Rapid Network Adaptation:
Learning to Adapt Neural Networks Using Test-Time Feedback

Teresa Yeo  Oğuzhan Fatih Kar  Zahra Sodagar  Amir Zamir
Swiss Federal Institute of Technology Lausanne (EPFL)
https://rapid-network-adaptation.epfl.ch/

Abstract

We propose a method for adapting neural networks to distribution shifts at test-time. In contrast to training-time robustness mechanisms that attempt to anticipate and counter the shift, we create a closed-loop system and make use of test-time feedback signal to adapt a network on the fly. We show that this loop can be effectively implemented using a learning-based function, which realizes an amortized optimizer for the network. This leads to an adaptation method, named Rapid Network Adaptation (RNA), that is notably more flexible and orders of magnitude faster than the baselines. Through a broad set of experiments using various adaptation signals and target tasks, we study the generality, efficiency, and flexibility of this method. We perform the evaluations using various datasets (Taskonomy, Replica, ScanNet, Hypersim, COCO, ImageNet), tasks (depth, optical flow, semantic segmentation, classification), and distribution shifts (Cross-datasets, 2D and 3D Common Corruptions) with promising results.

1. Introduction

Neural networks are found to be unreliable against distribution shifts [23, 35, 43, 42, 29]. Examples of such shifts include blur due to camera motion, object occlusions, changes in weather conditions and lighting, etc. The training-time strategies to deal with this issue attempt to anticipate the shifts that may occur and counter them at the training stage – for instance, by augmenting the training data or updating the architecture with corresponding robustness inductive biases. As the possible shifts are numerous and unpredictable, this approach has inherent limitations. This is the main motivation behind test-time adaptation methods, which instead aim to adapt to such shifts as they occur and recover from failure. In other words, these methods choose adaptation over anticipation (see Fig. 1). In this work, we propose a test-time adaptation framework that aims to perform an efficient adaptation of a given main network using a feedback signal.

Figure 1: Adaptive vs non-adaptive neural network pipelines. Top: In order to be robust, non-adaptive methods include training-time interventions that anticipate and counter the distribution shifts that will occur at test-time (e.g., via data augmentation). The learned model, \( f_\theta \), is frozen at test-time, thus upon encountering an out-of-distribution input, its predictions may collapse. Bottom: Adaptive methods create a closed-loop and use an adaptation signal at test-time. The adaptation signal is a quantity that can be computed at test-time from the environment. \( h_\phi \) acts as a “controller” by taking in an error feedback, computed from the adaptation signal and model predictions, to adapt \( f_\theta \) accordingly. It can be implemented as a (i) standard optimizer (e.g., using SGD) or (ii) neural network. The former is equivalent to test-time optimization (TTO), while the latter aims to amortize the optimization process, by training a controller network to adapt \( f_\theta \) – thus, it can be more efficient and flexible. In this work, we study the latter approach and show its efficiency and flexibility.

This ICCV paper is the Open Access version, provided by the Computer Vision Foundation. Except for this watermark, it is identical to the accepted version; the final published version of the proceedings is available on IEEE Xplore.
cess, and consequently, unconducive for real-world applications. It also results in a rigid framework as the update mechanism is fixed to be the same as the training process of neural networks (SGD). We show this process can be effectively amortized using a learning-based feed-forward controller network, which yields orders of magnitude faster results (See Fig. 1, Sec. 4.3). In addition, it provides flexibility advantages as the controller is implemented using a neural network and can be engineered to include arbitrary inductive biases and desired features that could counter the suboptimalities of the adaptation signal.

2. Related Work

Our work focuses on how to adapt a neural network efficiently at test-time on a range of tasks and adaptation signals. We give an overview of relevant topics. See the supplementary Sec. 7 for more related works.

Robustness methods anticipate the distribution shift that can occur and incorporate inductive biases into the model to help it generalize. Popular methods include data augmentation [61, 107, 57, 37, 103, 101, 34, 43], self-/pre-training [36, 95, 25, 72, 96, 75, 32], architectural changes [19, 8, 82, 62, 55] or ensembling [48, 67, 99, 73, 40, 68]. We focus on adaptation mechanisms and identifying practical adaptation signals that can be used at test-time.

Amortized optimization methods make use of learning to improve (e.g., speed-up) the solution of optimization problems, particularly for settings that require repeatedly solving similar instances of the same underlying problem [49, 11, 4, 106, 26, 94, 60, 14, 3]. Fully amortized optimization methods model the shared structure between past instances of solved problems to regress the solution to a new problem [31, 21, 52]. As adapting to distribution shifts can be cast as solving an optimization problem at test-time, our method can be seen as an amortized solution.

Test-time adaptation methods for geometric tasks. Many existing frameworks, especially in geometric tasks such as aligning a 3D object model with an image of it, in effect instantiate a task-specific case of closed-loop optimization for each image [60, 106, 63]. Common sources of their adaptation quantity include sensor data [92, 98, 15, 89, 102, 18], structure from motion (SFM) [93, 47], motion [12], and photometric and multi-view consistency constraints (MVC) [58, 45]. Many of the latter methods often focus on depth prediction and they introduce losses that are task-specific, e.g., [93] optimize a photometric consistency loss. We differ by aiming to investigate a more general framework for test-time adaptation that can be applied to several tasks. For MVC, while we adopt the same losses as [58], we show under collapsed predictions, optimizing only MVC constraints is not sufficient for recovering predictions; depth predictions need to be adapted and this can be done efficiently using our framework (see Sec. 4.3).

Test-time adaptation methods for semantic tasks. Most of these works involve optimizing a self-supervised objective at test-time [85, 91, 54, 27, 28, 108, 9, 51, 100]. They differ in the choice of self-supervised objectives, e.g., prediction entropy [91], mutual information [51], and parameters optimized [9]. However, as we will discuss in Sec. 3.2, and as shown by [9, 27, 65], existing methods can fail silently, i.e., successful optimization of the adaptation signal loss does not necessarily result in better performance on the target task. We aim to have a more efficient and flexible method and also show that using proper adaptation signals results in improved performance.

Multi-modal frameworks are models that can use the information from multiple sources, e.g., RGB image, text, audio, etc., [13, 5, 86, 50, 2, 16, 72, 1, 6, 30]. Schematically, our method has similarities to multi-modal learning (as many amortized optimization methods do) since it simultaneously uses an input RGB image and an adaptation signal. The main distinction is that our method implements a particular process toward adapting a network to a shift using an adaptation signal from the environment – as opposed to a generic multi-modal learning.

3. Method

In Fig. 1, we schematically compared methods that incorporate robustness mechanisms at training-time (thus anticipating the distribution shift) with those that adapt to shifts at test-time. Our focus is on the latter. In this section, we first discuss the benefits and downsides of common adaptation methods (Sec. 3.1). We then propose an adaptation method that is fast and can be applied to several tasks (Sec. 3.1.1). To adapt, one also needs to be able to compute an adaptation signal, or proxy, at the test-time. In the second part of the section (Sec. 3.2), we study a number of practical adaptation signals for a number of tasks.

3.1. How to adapt at test-time?

An adaptive system is one that can respond to changes in its environment. More concretely, it is a system that can acquire information to characterize such changes, e.g., via an adaptation signal that provides an error feedback, and make modifications that would result in a reduction of this error (see Fig. 1). The methods for performing the adaptation of the network range from gradient-based updates, e.g., using SGD to fine-tune the parameters [85, 91, 27], to the more efficient semi-amortized [110, 88] and amortized approaches [90, 66, 77] (see Fig. 6 of [77] for an overview). As amortization methods train a controller network to substitute the explicit optimization process, they only require a forward pass at test-time. Thus, they are computationally efficient. Gradient-based approaches, e.g., TTO, can be powerful adaptation methods when the test-time signal is robust and well-suited for the task (see Fig. 4). How-
ever, they are inefficient and have the risk of failing silently and the need for carefully tuned optimization hyperparameters [9]. In this work, we focus on an amortization-based approach.

**Notation.** We use \( \mathcal{X} \) to denote the input image domain, and \( \mathcal{Y} \) to denote the target domain for a given task. We use \( f_\theta : \mathcal{X} \to \mathcal{Y} \) to denote the model to be adapted, where \( \theta \) denotes the model parameters. We denote the model before and after adaptation as \( f_\theta \) and \( f_\hat{\theta} \) respectively. \( \mathcal{L} \) and \( \mathcal{D} \) are the original training loss and training dataset of \( f_\theta \), e.g., for classification, \( \mathcal{L} \) will be the cross-entropy loss and \( \mathcal{D} \) the ImageNet training data. As shown in Fig. 1, \( h_\phi \) is a controller for \( f_\theta \). It can be an optimization algorithm, e.g., SGD, or a neural network. \( \phi \) denotes the optimization hyperparameters or the network’s parameters. The former case corresponds to TTO, and the latter is the proposed RNA, which will be explained in the next subsection. Finally, the function \( g : \mathcal{X}^M \to \mathcal{Z} \) returns the adaptation signal by mapping a set of images \( \mathcal{B} = \{I_1, ..., I_M\} \in \mathcal{X}^M \) to a vector \( g(\mathcal{B}) = z \in \mathcal{Z} \). This function \( g \) is given, e.g., for depth, \( g \) returns the sparse depth measurements from SFM.

### 3.1.1 Rapid Network Adaptation (RNA)

For adaptation, we use a neural network for \( h_\phi \). The adaptation signal and model predictions are passed as inputs to \( h_\phi \) and it is trained to regress the parameters \( \hat{\theta}(\phi) = h_\phi(f_\theta(\mathcal{B}), z) \). This corresponds to an objective-based amortization of the TTO process [3]. Using both the adaptation signal \( z \) and model prediction \( f_\theta(\mathcal{B}) \) informs the controller network about the potential errors of the model. The training objective for \( h_\phi \) is \( \min_{h_\phi} \mathbb{E}_\mathcal{D}[\mathcal{L}(f_{\hat{\theta}(\phi)}(\mathcal{B}), y)] \), where \( (\mathcal{B}, y) \sim \mathcal{D} \) is a training batch sampled from \( \mathcal{D} \). Note that the **original weights of \( f \) are frozen** and \( h_\phi \) is a small network, having only 5-20% of the number of parameters of \( f \), depending on the task. We call this method as rapid network adaptation (RNA) and experiment with different variants of it in Sec. 4.

There exist various options for implementing the amortization process, e.g., \( h_\phi \) can be trained to update the input image or the weights of \( f_\theta \). We choose to modulate the features of \( f_\theta \) as it has been shown to work well in different domains [24] and gave the best results. To do this, we insert \( k \) Feature-wise Linear Modulation (FiLM) layers [71] into \( f_\theta \) (see Fig. 2). Each FiLM layer performs: \( \text{FiLM}(x_i; \gamma_i, \beta_i) = \gamma_i \odot x_i + \beta_i \), where \( x_i \) is the activation of layer \( i \). \( h_\phi \) is a network that takes as input the adaptation signal \( z \) and model predictions and outputs the coefficients \( \{\gamma_i, \beta_i\} \) of all \( k \) FiLM layers. \( h_\phi \) is trained on the same dataset \( \mathcal{D} \) as \( f_\theta \), therefore, unlike TTO, it is never exposed to distribution shifts during training. Moreover, it is able to generalize to unseen shifts (see Sec. 4.3). See the supplementary for the full details, other RNA implementations we investigated, and a comparison of RNA against other approaches that aim to handle distribution shifts.

### 3.2 Which test-time adaptation signals to use?

While developing adaptation signals is not the main focus of this study and is independent of the RNA method, we need to choose some for experimentation. Existing test-time adaptation signals, or proxies, in the literature include prediction entropy [91], spatial autoencoding [27], and self-supervised tasks like rotation prediction [85], contrastive [54] or clustering [9] objectives. The more aligned the adaptation signal is to the target task, the better the performance on the target task [85, 54]. More importantly, a poor signal can cause the adaptation to fail silently [9, 27].

Figure 3 shows how the original loss on the target task changes as different proxy losses from the literature, i.e. entropy [91], consistency between different middle domains [99, 104] are minimized. In all cases, the proxy loss decreases, however, the improvement in the target loss varies. Thus, successful optimization of existing proxy losses does not necessarily lead to better performance on the target task. In this paper, we adopt a few practical and real-world signals for our study. Furthermore, RNA turns out to be less susceptible to a poor adaptation signal vs TTO (see supplementary Tab. 1). This is because RNA is a neural network trained to use these signals to improve the target task, as opposed to being fixed at being SGD, like in TTO.

#### 3.2.1 Employed test-time adaptation signals

We develop test-time adaptation signals for several geometric and semantic tasks as shown in Fig. 4. Our focus is not on providing an extensive list of adaptation signals, but rather on using practical ones for experimenting with RNA as well as demonstrating the benefits of using signals that are rooted in the known structure of the world and the task in hand. For example, geometric computer vision tasks naturally follow the multi-view geometry constraints, thus making that a proper candidate for approximating the test-time error, and consequently, an informative adaptation signal.

**Geometric Tasks.** The field of multi-view geometry and its theorems, rooted in the 3D structure of the world, provide a rich source of adaptation signals. We demonstrate
Figure 3: Adaptation using different signals. Not all improvements in proxy loss translates into improving the target task’s performance. We show the results of adapting a pre-trained depth estimation model to a defocus blur corruption by optimizing different adaptation signals: prediction entropy [91], a self-supervised task (sobel edge prediction error [99]), and sparse depth obtained from SFM. The plots show how the ℓ_1 target error with respect to ground-truth depth (green, left axis) changes as the proxy losses (blue, right axis) are optimized (shaded regions represent the 95% confidence intervals across multiple runs of stochastic gradient descent (SGD) with different learning rates). Only adaptation with the sparse depth (SFM) proxy leads to a reduction of the target error. This signifies the importance of employing proper signals in an adaptation framework. Furthermore, we show that RNA is less susceptible to poorer adaptation signal, which results in comparable or improved performance while being significantly faster (see supplementary Table 1).

We demonstrate that our approach consistently outperforms the baselines for adaptation to different distribution shifts (2D and 3D Common Corruptions [35, 43, 44], cross-datasets), over different tasks (monocular depth, image classification, semantic segmentation, optical flow) and datasets (Taskonomy [105], Replica [84], ImageNet [22], COCO [53], ScanNet [20], Hypersim [78]). The source code can be found on our project page.

4. Experiments

We evaluate the following baselines:

- **Baseline** model trained with log-likelihood loss by optimizing the proxy loss (see Tab. 1 for the adaptation signal used for each task.) at test-time. Its weights are reset after each iteration. We denote this as Baseline for brevity.

**Pre-Adaptation Baseline**: The network f_0 that maps from RGB to the target task, e.g., depth estimation, with no test-time adaptation. We denote this as Baseline for brevity.

**Densification**: A network that maps from the given adaptation signal for the target task to the target task, e.g., sparse depth from SFM to dense depth. This is a control baseline and shows what can be learned from the test-time supervision alone, without employing input image information or a designed adaptation architecture. See Sec. 4.3 for a variant which includes the image as an additional input.

**TTO (episodic)**: We adapt the Baseline model to each episode by optimizing the proxy loss (see Tab. 1 for the adaptation signal used for each task.) at test-time. Its weights are reset to the Baseline model’s after optimizing each batch, similar to [91, 108].

**TTO (online)**: We continually adapt to a distribution shift defined by a corruption and severity. Test data is assumed to arrive in a stream, and each data point has the same distribution shift, e.g., noise with a fixed standard deviation [91, 85]. The difference with TTO (episodic) is that the model weights are not reset after each iteration. We denote this as TTO for brevity.

**TTO with Entropy supervision (TENT [91])**: We adapt the Baseline model trained with log-likelihood loss by optimizing the entropy of the predictions. This is to reveal the effectiveness of entropy as a signal as proposed in [91].

**TTO with Sobel Edges supervision (TTO-Edges)**: We adapt the Baseline model trained with an additional decoder that predicts a self-supervised task, similar to [85]. We choose
Table 1: Overview of the experiments for different target tasks, adaptation methods, and adaptation signals. For each task, we list the adaptation signal (Sec. 3.2) that we use for adaptation. We also list the models that we adapt, and the out-of-distribution (OOD) data used for evaluations and the relevant baselines. When there are different options for adaptation signal, e.g., in the case of depth, the signal is denoted in italics followed by the corresponding OOD dataset. The weights for the semantic segmentation, classification and optical flow models were taken from PyTorch [70].

Masked ground truth (GT): We apply a random mask to the GT depth of the test image. We fixed the valid pixels to 0.05% of all pixels, i.e., similar sparsity as SFM (see the supplementary for other values). This a control proxy as it enables a better evaluation of the adaptation methods without conflating with the shortcomings of adaptation signals. It is also a scalable way of simulating sparse depth from real-world sensors, e.g., LiDAR, as also done in [92, 59, 41].

Click annotations: We generate click annotations over random pixels for each class in a given image using GT – simulating an active annotation pipeline. The number of pixels ranges from 3 to 25, i.e. roughly 0.01% of the total pixels, similar to [7, 69, 83, 109, 17].

Patch matching: To not use GT click annotations, for each test image, we first retrieve its k-NN images from the original clean training dataset using DINO features [10]. We then get segmentation masks on these k images. If the training dataset has labels for segmentation we use them directly, otherwise we obtain them from a pretrained network. For each of the k training images and test image, we extract non-overlapping patches. The features for each patch that lie inside the segmentation masks of the k training images are matched to the features of every patch in the test image. These matches are then filtered and used as sparse segmentation annotations. See Fig. 4 for illustration.

Coarse labels (WordNet): We generate 45 coarse labels from the 1000-way ImageNet labels, i.e. making the labels 2x coarser, using the WordNet tree [64], similar to [39]. See supplementary for more details on the construction and results for other coarse label sets.

Coarse labels (DINO k-NN): For each test image, we retrieve the k-NN images from the training dataset using DINO features [10]. Each of these k training images is associated with an ImageNet class, thus, combining k one-hot labels gives us a coarse label.

Keypoint matching: We perform keypoint matching across images to get sparse optical flow.

<table>
<thead>
<tr>
<th>Task</th>
<th>Adaptation signal</th>
<th>Adapted model</th>
<th>Training data</th>
<th>OOD evaluation data</th>
<th>Baselines</th>
</tr>
</thead>
<tbody>
<tr>
<td>Depth</td>
<td>SFM, masked GT</td>
<td>UNet [80], DPT [74]</td>
<td>Taskonomy</td>
<td>For SFM: Replica, Replica-CC, ScanNet, for masked GT: Taskonomy-CC, -3DCC, Hypersim</td>
<td>Pre-adaptation, densification, TENT, TTO-edges, TTO</td>
</tr>
<tr>
<td>3D reconstruction</td>
<td>SFM, keypoint matching, consistency</td>
<td>Depth, optical flow models</td>
<td>Depth, optical flow data</td>
<td>Replica-CC</td>
<td>Pre-adaptation, TTO</td>
</tr>
<tr>
<td>Semantic segmentation</td>
<td>Click annotations, patch matching</td>
<td>FCN [56]</td>
<td>COCO (20 classes from Pascal VOC)</td>
<td>For click annotations: COCO-CC, For patch-matching: ImageNet-C</td>
<td>Pre-adaptation, densification, TENT, TTO</td>
</tr>
</tbody>
</table>

Figure 5: RNA can achieve similar performance as TTO in a much shorter time. We compare how the ℓ errors of the adaptation mechanisms decrease over wall-clock time (s). The errors are averaged over all episodes (and all corruptions for Replica-CC). RNA only requires a forward pass at test-time, while TTO requires multiple forward and backward passes. On ScanNet and Replica-CC, RNA takes 0.01s, while TTO takes 3s to achieve similar performance. Furthermore, RNA is not trained with test-time shifts unlike TTO, thus, it learned to use the additional supervision to adapt to unseen shifts.

Adaptation signal. As described in Sec. 3.2, we compute a broad range of test-time signals from the following processes. Each case describes a process applied on query image(s) in order to extract a test-time quantity.

Structure-from-motion (SFM): Given a batch of query images, we use COLMAP [81] to run SFM, which returns sparse depth. The percentage of valid pixels, i.e. depth measurements, is about 0.16% on Replica-CC and 0.18% on Replica. For ScanNet we use the pre-computed sparse depth from [79], which has about 0.04% valid pixels. As running SFM on corrupted images results in noisy sparse depth, we train hϕ to be invariant to noise [102, 79].

RNA configurations. At test-time, we first get the predictions of the Baseline model and compute the adaptation signal. The predictions and adaptation signal are then passed to hϕ which adapts f0 to f̂. The test images are then passed to f̂ to get the final predictions. We evaluate following variants of RNA.

RNA (frozen f): Baseline model weights, f0, are frozen when training hϕ. We call this variant RNA for brevity.

RNA (jointly trained f): In contrast to the frozen f variant, here we train hϕ jointly with the Baseline network. This variant requires longer training.

to predict Sobel edges as it has been shown to be robust to certain shifts [99]. We optimize the error of the edges predicted by the model and edges extracted from the RGB image to reveal the value of edge error as a supervision.
Figure 6: Qualitative results of RNA vs the baselines for semantic segmentation on random query images on COCO-CC (left) and depth on images from ScanNet, Taskonomy-3DCC and Replica-CC (right). For semantic segmentation, we use 15 pixel annotations per class. For Taskonomy-3DCC, we use sparse depth with 0.05% valid pixels (30 pixels per image). See Fig. 7 for results on different adaptation signal levels. For ScanNet and Replica-CC, the adaptation signal is sparse depth measurements from SFM [81] with similar sparsity ratios to Taskonomy-3DCC. The predictions with proposed adaptation signals are shown in the last two rows. They are noticeably more accurate compared to the baselines. Comparing TTO and RNA, RNA’s predictions are more accurate for segmentation, and sharper than TTO for depth (see the ellipses) while being significantly faster. See 4.2 and supplementary for more results.

4.2. Adaptation with RNA vs TTO

Here we summarize our observations from adapting with RNA vs TTO. As described, TTO represents the approach of closed-loop adaptation using the adaptation signal but without benefiting from amortization and learning (the adaptation process is fixed to be standard SGD). These observations hold across different tasks. See Sec. 4.3 for results.

RNA is efficient. As RNA only requires a forward pass at test-time, it is orders of magnitude faster than TTO and is able to attain comparable performance to TTO. In Fig. 5, we compare the runtime of adaption with RNA and TTO for depth prediction. On average, for a given episode, RNA obtains similar performance as TTO in 0.01s, compared to TTO’s 3-5s. Similarly, for dense 3D reconstruction, RNA is able to adapt in 0.008s compared to TTO’s 66s (see Fig. 9). This suggests a successful amortization of the adaptation optimization by RNA.

Furthermore, RNA’s training is also efficient as it only requires training a small model, i.e. 5-20% of the Baseline model’s parameters, depending on the task. Thus, RNA has a fixed overhead, and small added cost at test-time.

RNA’s predictions are sharper than TTO for dense prediction tasks. From the last two rows of Fig. 6, it can be seen that RNA retains fine-grained details. This is a noteworthy point and can be attributed to the fact that RNA benefits from a neural network, thus its inductive biases can be beneficial (and further engineered) for such advantages. This is a general feature that RNA, and more broadly using a learning-based function to amortize adaptation optimization, brings – in contrast to limiting the adaptation process to be SGD, as represented by TTO.

RNA generalizes to unseen shifts. RNA performs better than TTO for low severities (see supplementary for more details). However, as it was not exposed to any corruptions, the performance gap against TTO narrows at high severities as expected, which is exposed to corruptions at test-time.

We hypothesize that the generalization property of RNA is due to the following reasons. 1. Even though $f_\theta$ was trained to convergence, it does not achieve exactly 0 error. Thus, when $h_\phi$ is trained with a frozen $f_\theta$ with the training data, it can still learn to correct the errors of $f_\theta$, thus, adapting $f_\theta$. 2. Adaptation signals, by definition, are expected to be relatively robust to distribution shifts. Even though the RGB image has been corrupted or from a new domain, the adaptation signal is more aligned with the target task. Thus, the input to $h_\phi$ does not undergo a significant shift and is able to adapt $f_\theta$.

4.3. Experiments using Various Target Tasks

In this section, we provide a more comprehensive set of evaluations covering various target tasks and adaptation signals. In all cases, RNA is a fixed general framework without being engineered for each task and shows supportive results.
Table 2: Quantitative adaptation results on depth estimation. $\ell_1$ errors on the depth prediction task. (Lower is better. Multiplied by 100 for readability. The best models within 0.0003 error are shown in bold.) We generate distribution shifts by applying Common Corruptions (CC), 3D Common Corruptions (3DCC) and from performing cross-dataset evaluations (CDS). The results from CC and 3DCC are averaged over all distortions and severity levels on Taskonomy and 3 severity levels on Replica data. The adaptation signal from Taskonomy is masked GT (fixed at 0.05% valid pixels) while that from Replica and ScanNet is sparse depth from SFM. RNA and TTO notably outperform the baselines. RNA successfully matches the performance of TTO while being around 10 times faster. See supplementary for the losses for different corruption types, sparsity levels, and the results of applying RNA to other adaptation signals.

![Image](56x391 to 280x483)

Figure 7: Qualitative adaptation results on semantic segmentation on random query images on COCO-CC. RNA notably improves the prediction quality using error feedback from as few as 3 random pixels.

![Image](321x242 to 533x504)

Figure 9: Adaptation results for 3D reconstruction. Using appropriate adaptation signals from multi-view geometry can recover accurate 3D reconstructions. We report the average $\ell_1$ error between ground truth 3D coordinates and the estimated ones. The titles above each column refers to the depth model used to get the reconstruction. TTO+MVC corresponds to the predictions after multi-view consistency optimization. It can be seen that RNA and TTO improve the reconstructions over the baselines with RNA being significantly faster. See supplementary for more results and the corresponding error maps.

Depth. We demonstrate the results quantitatively in Tab. 2 and Fig. 5 and qualitatively in Fig. 6. In Tab. 2, we compare RNA against all baselines, and over several distribution shifts and different adaptation signals. Our RNA variants outperform the baselines overall. TTO (online) has a better performance than TTO (episodic) as it assumes a smoothly changing distribution shift, and it continuously updates the model weights. RNA (jointly trained $f$) has a better performance among RNA variants. This is reasonable as the target model is not frozen, thus, is less restrictive.

As another baseline, we trained a single model that takes as input a concatenation of the RGB image and sparse supervision, i.e., multi-modal input. However, its average performance on Taskonomy-CC was 42.5% worse than RNA’s (see sup. mat. Sec. 3.2). Among the baselines that do not adapt, densification is the strongest under distribution shift due to corruptions. This is expected as it does not take the RGB image as input, thus, it is not affected by the underlying distribution shift. However, as seen from the qualitative results in Figs. 6, 7, unlike RNA, densification is unable to predict fine-grained details (which quantitative metrics often do not well capture). We also show that the gap between RNA and densification widens with sparser supervision (see sup. mat. Fig. 1), which confirms that RNA is making use of the error feedback signal, to adapt $f$.
Table 3: Quantitative adaptation results on ImageNet (IN) classification task. We evaluate on the clean validation set, ImageNet-\{C,3DCC,V2\}. We report average error (%) for 1000-way classification task over all corruptions and severities. For the coarse labels with WordNet supervision, we use 45-coarse labels. For DINO k-NN, we set k = 20.

<table>
<thead>
<tr>
<th>Adaptation Signal</th>
<th>Dataset</th>
<th>Clean</th>
<th>IN-C</th>
<th>IN-3DCC</th>
<th>IN-V2</th>
<th>Rel. Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entropy</td>
<td></td>
<td>37.85</td>
<td>64.66</td>
<td>54.97</td>
<td>37.15</td>
<td>1.00</td>
</tr>
<tr>
<td>Coarse labels (wordnet)</td>
<td></td>
<td>Demarcation</td>
<td>95.50</td>
<td>95.50</td>
<td>95.50</td>
<td>1.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TTO (online)</td>
<td>24.72</td>
<td>40.62</td>
<td>42.90</td>
<td>36.77</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RNA (frozen)</td>
<td>16.72</td>
<td>41.21</td>
<td>40.37</td>
<td>25.53</td>
</tr>
<tr>
<td>Coarse labels (DINO k-NN)</td>
<td></td>
<td>DINO (k-NN)</td>
<td>25.36</td>
<td>52.64</td>
<td>48.24</td>
<td>37.39</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TTO (online)</td>
<td>24.95</td>
<td>51.59</td>
<td>49.18</td>
<td>36.96</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RNA (frozen)</td>
<td>24.36</td>
<td>54.86</td>
<td>52.29</td>
<td>36.88</td>
</tr>
</tbody>
</table>

Semantic Segmentation. We experiment with click annotations and DINO patch matching as adaptation signals. Click annotations: In Fig. 8, we show how the IoU changes with the adaptation signal level on COCO-CC. As the Baseline and TENT do not make use of this signal, their IoU is a straight line. RNA clearly outperforms the baselines for all levels of adaptation signal. Figure 7 shows the qualitative results with increasing supervision, and Fig. 6 (left) a comparison against all baselines, demonstrating higher quality predictions with RNA.

DINO patch matching: We perform patch matching on DINO features (described in Sec. 4.1) to get the adaptation signal. As the patch matching process can be computationally expensive, we demonstrate our results on all cat classes in ImageNet and over one noise, blur and digital corruption for 3 levels of severity. We used the predictions of a pre-trained FCN on the clean images as pseudolabels to compute IoU. The mean IoU averaged over these corruptions and severities is 48.98 for the baseline model, 53.45 for TTO. RNA obtains a better IOU of 58.04, thus it can make use of the sparse annotations from DINO patch matching.

Image Classification. We experiment with coarse labels from WordNet and DINO k-NN as adaptation signals. Coarse labels (WordNet): Table 3 shows the results from using 45-coarse labels on ImageNet-\{C,3DCC,V2\}. This corresponds to 22x coarser supervision compared to the 1000 classes that we are evaluating on. TENT seems to have notable improvements in performance under corruptions for classification, unlike for semantic segmentation and depth. We show that using coarse supervision results in even better performance, about a further 5 pp reduction in error. Furthermore, on uncorrupted data, i.e. clean, and ImageNet-V2 [76], RNA gives roughly 10 pp improvement in performance compared to TTO. Thus, coarse supervision provides a useful signal for adaptation while requiring much less effort than full annotation [97]. See supplementary for results on other coarse sets.

Coarse labels (DINO k-NN): We also show results from using coarse sets generated from DINO k-NN retrieval. This is shown in the last 3 rows of Tab. 3. Both RNA and TTO use this coarse information to outperform the non-adaptive baselines. However, they do not always outperform TENT, which could be due to the noise in retrieval.

4.4. Ablations and additional results

Table 4: RNA works across different architectures of the main network $f_\theta$ such as DPT [74] and ConvNext [55]. Quantitative adaptation results on depth estimation and image classification on Taskonomy and ImageNet datasets, respectively. (Lower is better. $\ell_1$ errors for depth estimation are multiplied by 100 for readability.)

<table>
<thead>
<tr>
<th>Task (Arch.)</th>
<th>Depth (DPT [74])</th>
<th>Classification (ConvNext [55])</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shift</td>
<td>Clean</td>
<td>CC</td>
</tr>
<tr>
<td>Pre-adaptation Baseline</td>
<td>2.23</td>
<td>3.76</td>
</tr>
<tr>
<td>TTO (Baseline)</td>
<td>1.82</td>
<td>2.61</td>
</tr>
<tr>
<td>RNA (frozen)</td>
<td>1.13</td>
<td>1.56</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method/Shift</th>
<th>None</th>
<th>Taskonomy-CC</th>
<th>Taskonomy-3DCC</th>
<th>HyperSim</th>
<th>BlendedMVG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-adaptation</td>
<td>0.027</td>
<td>0.057</td>
<td>0.048</td>
<td>0.336</td>
<td>3.450</td>
</tr>
<tr>
<td>RNA (HyperNetwork-\theta)</td>
<td>0.019</td>
<td>0.041</td>
<td>0.033</td>
<td>0.257</td>
<td>2.587</td>
</tr>
<tr>
<td>RNA (FiLM-\theta)</td>
<td>0.010</td>
<td>0.039</td>
<td>0.033</td>
<td>0.279</td>
<td>2.636</td>
</tr>
<tr>
<td>RNA (FiLM-f)</td>
<td>0.013</td>
<td>0.024</td>
<td>0.020</td>
<td>0.198</td>
<td>2.310</td>
</tr>
</tbody>
</table>

Table 5: Implementations of different architectures for the controller network $h_\phi$. $\ell_1$ errors on the depth estimation task under distribution shifts are reported. The adaptation signal here is masked GT, fixed at 0.05% of valid pixels.

<table>
<thead>
<tr>
<th>Implementation of different architectures for the controller network $h_\phi$</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-adaptation</td>
<td>0.027</td>
<td>0.057</td>
<td>0.048</td>
<td>0.336</td>
</tr>
<tr>
<td>RNA (HyperNetwork-\theta)</td>
<td>0.019</td>
<td>0.041</td>
<td>0.033</td>
<td>0.257</td>
</tr>
<tr>
<td>RNA (FiLM-\theta)</td>
<td>0.010</td>
<td>0.039</td>
<td>0.033</td>
<td>0.279</td>
</tr>
<tr>
<td>RNA (FiLM-f)</td>
<td>0.013</td>
<td>0.024</td>
<td>0.020</td>
<td>0.198</td>
</tr>
</tbody>
</table>
The results of adaptation with these variants of RNA are shown in Table 5. The FiLM-f variant performed best, thus, we adopted it as our main architecture. See supplementary Sec. 2.2 for more details and a discussion on the trade-offs of the choices of implementing this closed-loop “control” system, namely those that make stronger model-based assumptions.

**Controlling for number of parameters.** We ran a control experiment where all methods have the same architecture, thus, same number of parameters. The results are in the supplementary Table 2. RNA still returns the best performance. Thus, its improvement over the baselines is not due to a different architecture or number of parameters but due to its test-time adaptation mechanism.

5. Approaches for handling distribution shifts

In this section, we provide a unified discussion about the approaches that aim to handle distribution shifts. Figure 10 gives an overview of how these approaches can be characterized. **Open-loop** systems predict $y$ by only using their inputs without receiving feedback. Training-time robustness methods, image modifications, and multi-modal methods fall into this category. These methods assume the learned model is frozen at the test-time. Thus, they aim to incorporate inductive biases at training time that could be useful against distribution shifts at the test-time. The **closed-loop** systems, on the other hand, are adaptive as they make use of a live error feedback signal that can be computed at test-time from the model predictions and an adaptation signal.

In the interest of space, we move the detailed discussion about each of these variants to Sec. 1.2 of the supplementary. In particular, elaboration of the distinction between a model-based vs model-free approach, denoising/image-modification methods, multi-modal methods, and our observations over comparing the conclusions of [46] el al. and our experiments are provided there.

6. Conclusion and Limitations

We presented RNA, a method for efficient adaptation of neural networks at test-time using a closed-loop formulation. It involves training a side network to use a test-time adaptation signal to adapt a main network. This network acts akin to a “controller” and adapts the main network based on the adaptation signal. We showed that this general and flexible framework can generalize to unseen shifts, and as it only requires a forward pass at test-time, it is orders of magnitude faster than TTO. We evaluated this approach using a diverse set of adaptation signals and target tasks. We briefly discuss the limitations and potential future works:

**Different instantiation of RNA and amortization.** While we experimented with several RNA variants (see supplementary for details), further investigation toward a stronger instantiation of RNA which can generalize to more shifts and handle more drastic changes, e.g. via building in a more explicit “model” of the shifts and environment (see the discussion about [46]), is important. In general, as the role of the controller network is to amortize the training optimization of the main network, the amortized optimization literature [3] is one apt resource to consult for this purpose.

**Hybrid mechanism for activating TTO in RNA.** TTO constantly adapts a model to a distribution shift, hence, in theory, it can adapt to any shift despite being comparatively inefficient. To have the best of both worlds, investigating mechanisms for selectively activating TTO within RNA when needed can be useful.

**Finding adaptation signals for a given task.** While the focus of this study was not on developing new adaption signals, we demonstrated useful ones for several core vision tasks, but there are many more. Finding these signals requires either knowledge of the target task so a meaningful signal can be accordingly engineered or core theoretical works on understanding how a proxy and target objectives can be “aligned” for training.

Acknowledgement: The authors thank Onur Beker. This work was partially supported by the ETH4D and EPFL EssentialTech Centre Humanitarian Action Challenge Grant.
References


ings of the IEEE/CVF International Conference on Computer Vision, pages 10786–10796, 2021. 2


[32] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 16000–16009, 2022. 2


[44] Oğuzhan Fatih Kar, Teresa Yeo, and Amir Zamir. 3d common corruptions for object recognition. In ICML 2022 Shift Happens Workshop, 2022. 4


[54] Yuejiang Liu, Parth Kothari, Bastien van Delft, Baptiste Bellot-Gurlet, Taylor Mordan, and Alexandre Alahi. Ttt++: When does self-supervised test-time training fail or


[79] Barbara Roessle, Jonathan T Barron, Ben Mildenhall, Pratul P Srinivasan, and Matthias Nießner. Dense depth priors for neural radiance fields from sparse input views. In *Proceedings of the IEEE/CVF Conference on Computer Vi-


