FAST TEST-TIME ADAPTATION USING HINTS

Anonymous authors
Paper under double-blind review

ABSTRACT

We propose a framework for adapting neural networks to distribution shifts at test-time. The primary idea is to leverage proper adaptation objectives based on known general properties of the target task, e.g. multi-view geometry for 3D tasks, or hierarchical structure for semantic tasks. These properties can be instantiated as adaptation signals at test-time, which we refer to as "hints". These hints are robust to distribution shifts, thus, they make adaptation more reliable compared to existing test-time adaptation methods, e.g. entropy minimization. Next, we show that this optimization during test-time can be *amortized* using a side-network, thus, making the adaptation orders of magnitude faster. We call this variant of test-time adaption Rapid Network Adaptation (RNA). We demonstrate consistent improvements over the baselines on diverse tasks (depth, optical flow, semantic segmentation, classification), datasets (Taskonomy, Replica, ScanNet, COCO, ImageNet) and distribution shifts (Common Corruptions, 3D Common Corruptions, cross-datasets).

1 INTRODUCTION

Neural networks are unreliable against distribution shifts. Examples of such shifts include blur due to camera motion, object occlusions, changes in weather conditions and lighting. Dealing with such shifts is difficult mainly because they are *numerous* and *unpredictable*. Therefore, *training-time* strategies that attempt to take anticipatory measures for every possible shift (e.g., augmenting the training data accordingly or changing the architecture with corresponding robustness inductive biases) have inherent limitations. This is the main motivation behind *test-time* adaptation methods, which instead aim to adapt to such shifts as they occur. In other words, these methods choose *adaptation* over *anticipation*. In this work, we propose a test-time adaptation framework that aims to answer the following questions: **1.** What should be used as the test-time optimization objective? **2.** How should a model be adapted?

For the first question, we observe that existing test-time adaptation methods are insufficient (Boudiaf et al., 2022; Gandelsman et al., 2022). These methods aim to be adaptation strategies that work well for any task. Therefore, they employ proxy losses that are too general and largely *task-unaware* (e.g., reducing entropy Wang et al. (2020)). Due to this misalignment between such task-unaware objectives and the target task's original objective, the improvement in one may not translate to the other (see Fig. 2). Instead, we observe that known properties of the target task and/or the world from which the data is collected can be used for adaptation. For example, geometric computer vision tasks follow multi-view geometry constraints or semantic categories in a recognition task often follow a hierarchical structure (See Fig. 1). Inspired by the early investigations on this topic in the community Abu-Mostafa (1995), we call such properties "hints", and the instantiations of these hints, *task-aware proxies*. We empirically investigate the benefits of using them for test-time adaptation.

For the second question, we propose two ways to use these proxies. The first is as an optimization objective, similar to previous works (Wang et al., 2020; Zhang et al., 2021; Gandelsman et al., 2022). While this works, it is unnecessarily inefficient and unconducive for real-world applications. We show that this optimization process can actually be amortized using a single forward-pass from a side-network, yielding orders of magnitude faster adaptation (See Fig. 1, Sec. 4.1).

Contributions. We demonstrate that for many fundamental computer vision tasks (e.g., depth estimation, semantic segmentation, optical flow, image classification), test-time hints are readily available. We also empirically show that for these tasks, incorporating even the simplest hints into the

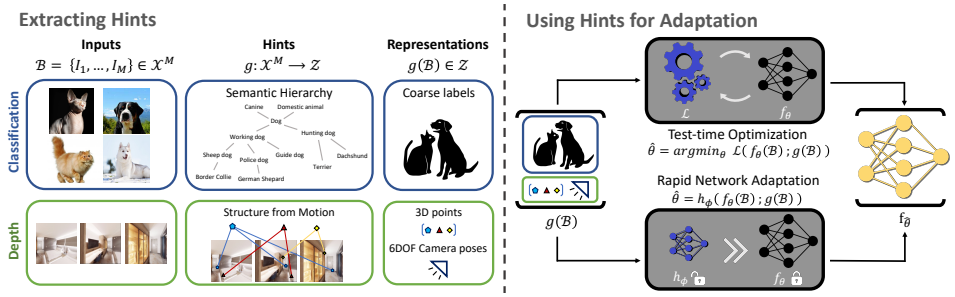

Figure 1: Test-time adaptation using hints. Adaptation signals based on Hints are properties of the target task known to us. For example, for depth estimation, the theorems of multi-view geometry hold for all inputs, and for classification all semantic categories follow a hierarchical structure. ***Left:*** When presented with a batch of inputs $\mathcal{B}$, one can use a hint g to represent such properties explicitly in a form $g(\mathcal{B})$ that can later be processed by a downstream test-time adaptation process. ***Right:*** Given a pre-trained model f_θ, we explore two ways of adapting it to a distribution shift at the test-time: **1.** Test-time Optimization (TTO), where hint representations are used as *proxy losses* $\mathcal{L}(*, g(\mathcal{B}))$ that are minimized to update the parameters θ, and **2.** where hint representations take the form of *proxy inputs* passed to a side-network $h_\phi(*, g(\mathcal{B}))$ that predicts updates to θ in a single forward-pass. This is a form of amortizing the optimization process of TTO, and consequently, gaining efficiency. We call it *Rapid Network Adaptation* (RNA). We demonstrate in Sec. 4 that using even the simplest hints can result in a striking performance increase compared to existing methods.

test-time adaptation process can provide a striking performance increase compared to existing methods. Finally, we propose a fast way to adapt a given computer vision model using hints.

2 Shortcomings of Existing Test-Time Adaptation Methods

Existing adaptation methods optimize either part or all of a model's parameters to minimize different proxy losses, which we refer to as test-time optimization (TTO). Common proxy losses include entropy of the predictions (Wang et al., 2020), errors on self-supervised pretext tasks such as rotation prediction (Sun et al., 2020), or spatial autoencoding (Gandelsman et al., 2022), and consistency measures between different middle domains (Yeo et al., 2021; Zamir et al., 2020). While posing the adaptation process as an optimization problem is a general and powerful approach, optimizing a quantity at test-time that is not well aligned with the target task's loss may lead to unstable behaviour. To empirically illustrate this, Fig. 2 (left) shows how the original loss on the target task changes as different proxy losses from the literature are minimized. In all cases the proxy loss decreases, however the degree to which this translates to an improvement on the target loss varies. Therefore, it can be concluded that *successful optimization of existing task-unaware proxy losses does not necessarily lead to better performance on the target task*.

A possible solution to the above problem is to find the optimal hyperparameters for TTO, e.g. batch size, optimizers. However, as shown in Boudiaf et al. (2022), even if one tunes the hyperparameters on a distribution shift (assuming access to the ground truth labels), different distribution shifts require different hyperparameters. Thus, one also has to anticipate the shift that will occur, *defeating the purpose of test-time adaptation*. In the rest of the paper, we show that using proxy losses that are *task-aware* results in lower sensitivity to hyperparameters. We also show that this optimization process can be amortized using a side-network, thus removing the need to tune any hyperparameters at test-time while being much faster.

3 Test-Time Adaptation Using Hints

This section provides examples of *hints* for common computer vision tasks, and discusses methods to incorporate them in the test-time adaptation process.

3.1 Examples of Hints For Computer Vision Tasks

Empirically, we find that *even hints that are extremely sparse in information content can work surprisingly well for adaptation*. Figure 3 shows this point with a proof-of-concept experiment where

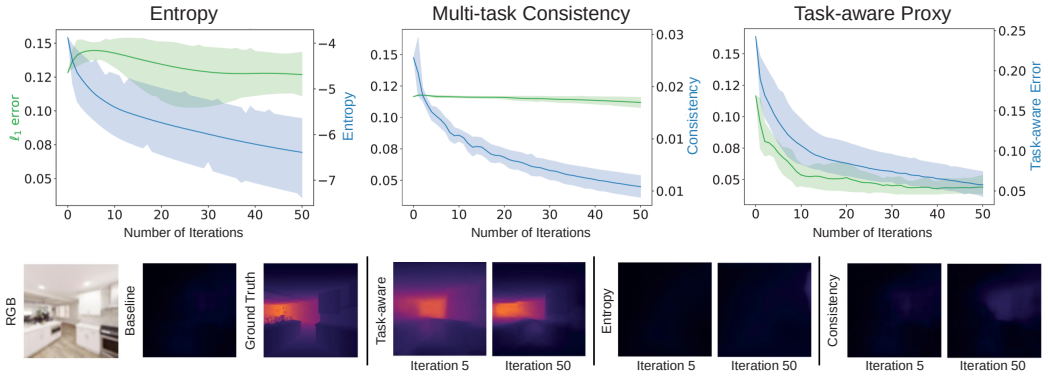

Figure 2: Existing proxy losses are task-unaware, thus, optimizing them does not necessarily improve performance on the target task. ***Left:*** We show the results of adapting a pre-trained depth estimation model to a defocus blur corruption by optimizing different proxy losses: prediction entropy (Wang et al., 2020), multi-task consistency (Zamir et al., 2020), and a task-aware proxy loss obtained from a hint (Sec. 3). The plots show how the ℓ_1 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. Optimization converges quickly for the task-aware loss, and is the only one where the proxy loss correlates well with improvements on the target task. ***Right:*** Visualizations of how predictions change with increasing numbers of SGD iterations. The distribution shift causes the baseline predictions to collapse to low values (i.e., black images). Consistent with the loss plots, optimizing task-unaware losses does not improve these collapsed predictions, unlike the task-aware proxy loss.

a monocular depth estimation model is being provided with an increasing number of pixels with ground truth (GT) depth annotations. The model is adapted by optimizing the error solely computed on these few annotated pixels. The plot shows the ℓ_1 error averaged over all pixels at the end of optimization, for different levels of GT supervision. The largest decrease in error occurs from 0% to 0.01% supervision, which corresponds to using only 6 pixels with GT supervision. We observe in Fig. 3 (right) that the adapted model recovers the coarse geometric outline of the scene (e.g., the walls, door, bed) from completely collapsed predictions.

We now discuss how such hints can be obtained under realistic assumptions for different computer vision tasks. Our exposition roughly divides computer vision tasks into two groups: geometric and semantic tasks.

Geometric Tasks. The field of multi-view geometry and its theories, rooted in the 3D spatial structure of the world, provide a rich source of hints for geometric tasks. We provide experiments on three examples: monocular depth estimation, dense optical-flow estimation, and dense 3D reconstruction. For all these tasks, we use COLMAP (Schönberger & Frahm, 2016) to run structure from motion (SFM), which provides: **1.** 3D coordinates of a sparse set of keypoints, **2.** matches between pixels that correspond to the projections of these keypoints across images, **3.** 6DOF camera poses. For depth estimation, we employ the z-coordinates of the sparse set of keypoints visible from each image as a hint. For dense optical flow estimation, we perform keypoint matching across images (which returns sparse optical flow). For dense 3D reconstruction, in addition to the previous two sources of hints, we employ the consistency between depth and optical flow predictions as another hint.

Semantic Tasks. We also show results on semantic segmentation and image classification tasks. For semantic segmentation, we follow existing works on active annotation tools (Cheng et al., 2022; Shin et al., 2021; Papadopoulos et al., 2017) and use a low number of click annotations for each class as a hint. For classification, we use the hierarchical structure of semantic classes, and use coarse labels generated from the WordNet tree (Miller, 1994), similar to Huh et al. (2016b).

3.2 METHOD

Notation. We use $\mathcal{X}$ to denote the image domain, and $\mathcal{Y}$ to denote the target domain for a given task. A hint is defined as a function $g : \mathcal{X}^M \to \mathcal{Z}$ that maps a batch of images $\mathcal{B} = \{I_1, ..., I_M\} \in \mathcal{X}^M$ to a vector $g(\mathcal{B}) = z \in \mathcal{Z}$ that represents the hint. This function g is given and not learnt. We use $f_\theta : \mathcal{X} \to \mathcal{Y}$ to denote the model to be adapted, where θ denotes the weights. We denote the model before and after adaptation as f_θ and $f_{\hat{\theta}}$ respectively, where the latter is used to get the final

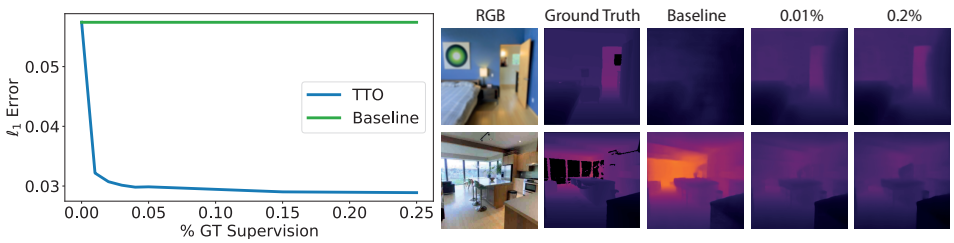

Figure 3: Simple hints are sufficient for successful adaptation. We adapt a pre-trained model to a distribution shift by optimizing our task-aware proxy loss via test-time optimization (TTO). We perform episodic optimization (Wang et al., 2020), i.e. after optimizing on a batch of inputs, its weights are reset back to that of the pre-trained model's weights. ***Left***: We show the ℓ_1 error vs % GT supervision for monocular depth prediction. x% GT supervision means each image has access to x% pixels of the GT to be used for TTO. The error is averaged over all corruptions from Common Corruptions for all severities. Having *only 0.01% supervision (6 pixels) per image* and with only 32 images, TTO can reduce the error by 44%. ***Right***: Predictions on two random query images from Taskonomy with increasing supervision. Even when the initial predictions have completely collapsed (see Baseline predictions), significant geometric structure can be recovered.

predictions after adaptation. $\mathcal{L}$ and $\mathcal{D}$ denote the original training loss and training dataset of f_θ. As illustrated in Fig. 1, we explore two ways of adapting f_θ, which we call *test-time optimization (TTO)* and *rapid network adaptation (RNA)*.

In TTO, hint representations parameterize a space of *proxy loss functions* $\mathcal{L}(f_\theta(\mathcal{B}), z)$. For example, $\mathcal{L}$ can be the ℓ_1-norm for monocular depth and cross-entropy for semantic segmentation. The weights of f are adapted by running SGD on the objective: $\hat{\theta} = \arg\min_\theta \mathcal{L}(f_\theta(\mathcal{B}), z)$.

In RNA, hint representations act as *proxy inputs* to a side-network denoted as h_ϕ. Unlike TTO, h_ϕ does not update θ by running SGD at test-time, but is instead 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 Amos (2022). The training objective for h_ϕ is $\min_\phi \mathbb{E}_\mathcal{D}\left[\mathcal{L}(f_{\hat{\theta}(\phi)}(\mathcal{B}), y)\right]$, where $(\mathcal{B}, y) \sim \mathcal{D}$ is a training batch sampled from $\mathcal{D}$.

We use feature-wise modulation (Dumoulin et al., 2018) to implement h_ϕ, and insert k Feature-wise Linear Modulation (FiLM) layers Perez et al. (2018) into the network f_θ. Each FiLM layer performs the operation $\text{FiLM}(x_i; \gamma_i, \beta_i) = \gamma_i \odot \mathbf{x}_i + \beta_i$, where $\mathbf{x_i}$ is the activation of layer i. h_ϕ is a convolutional network that takes as input the hint representation $g(\mathcal{B})$ defined above and outputs the coefficients $\{\gamma_i, \beta_i\}$ of all k FiLM layers. We also note that h_ϕ is trained on the same dataset $\mathcal{D}$ as f_θ, therefore it is never exposed to distribution shifts unlike TTO.

4 Experiments

We demonstrate that our approach consistently outperforms the baselines for different distribution shifts (Common Corruptions Hendrycks & Dietterich (2019), 3D Common Corruptions Kar et al. (2022a), cross-dataset evaluations), over different tasks (monocular depth, image classification, semantic segmentation, optical flow) and datasets (Taskonomy Zamir et al. (2018), Replica Straub et al. (2019), ImageNet Deng et al. (2009), COCO Lin et al. (2014), ScanNet Dai et al. (2017)). Our development code is provided in the supplementary material for reference.

4.1 Evaluations on Monocular Depth

Model. We use a UNet Ronneberger et al. (2015) model trained on Taskonomy Zamir et al. (2018) which includes 4 million real images of indoor scenes. In all the experiments, we use the same architecture for our method and the baselines. See A.2.1 for further training details.

Evaluation datasets and supervision.

Taskonomy under Common Corruptions (Taskonomy-CC) with masked GT: Common Corruptions applied on the test set of Taskonomy. As running SFM to get sparse depth on the Taskonomy test set is expensive, we simulate this by uniformly masking the GT (0.01-0.1% valid pixels, unless stated otherwise).

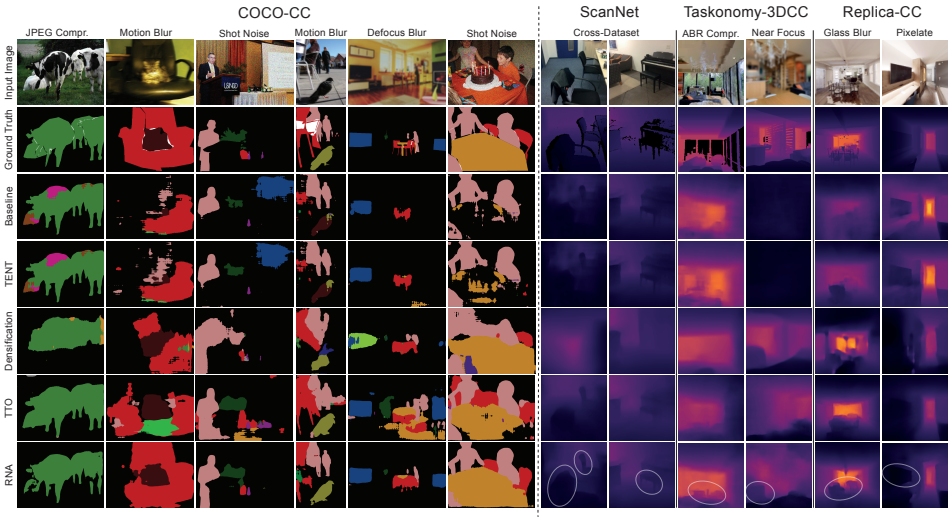

Figure 4: Qualitative results of our proposed method vs the baselines for semantic segmentation on random query images on COCO-CC (left) and depth on images from ScanNet, Taskonomy-3DCC and Replica-CC (right). For semantic segmentation, we assume access to 15 pixels per class. For Taskonomy-3DCC, we assume access to 0.05% of GT (30 pixels per image). See Fig. 5 for results on different supervision levels. For ScanNet and Replica-CC, supervision is from SFM (Schönberger & Frahm, 2016). The predictions from our proposed method is shown in the last two rows. They are noticeably more accurate compared to the baselines. Comparing TTO and RNA, RNA's predictions are less noisy for segmentation, and sharper than TTO for depth (see the ellipses) while significantly faster. See Fig. 16, 25-27 in the Appendix for more results.

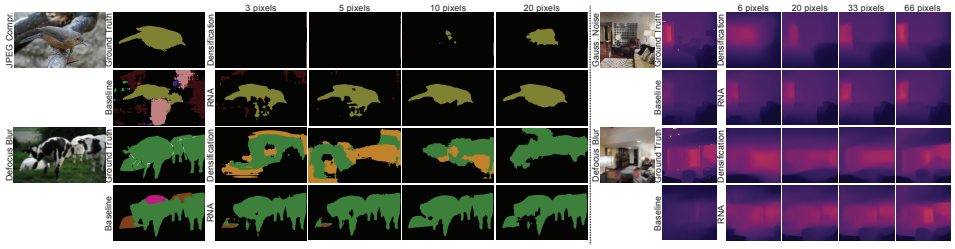

Figure 5: Qualitative results of our proposed method vs densification control baseline for semantic segmentation on random query images on COCO-CC (left) and monocular depth on images from Taskonomy-CC (right). Our method notably improves the prediction quality using as few as 3 pixels for segmentation and 6 pixels for depth estimation as supervision. See Fig. 8 for the quantitative results.

Taskonomy under 3D Common Corruptions (Taskonomy-3DCC) with masked GT: 3D Common Corruptions applied on the test set of Taskonomy. 3DCC incorporates the geometry of the scene into the transforms, resulting in more realistic corruptions.

Replica with sparse depth from SFM: We generate 7 episodes, a total of 300 images, in different apartments from Replica and run SFM to get sparse depth (0.18% valid depth pixels).

Replica under Common Corruptions (Replica-CC) with sparse depth from SFM: We use the same episodes as above, apply the corruptions, then run SFM to get noisy sparse depth (0.16% valid depth pixels).

ScanNet with sparse depth from SFM: We use the publicly available dataset from Roessle et al. (2022). It comes with sparse depth points computed with SFM (0.04% valid depth pixels).

Our setups. We evaluate following variants of the adaptation mechanisms proposed in Sec. 3.2.

TTO (episodic): We adapt the Baseline model to each episode by optimizing the proxy loss (see Sec. 3.2) at test-time. Its weights are reset back to those of the Baseline model after optimizing on that batch. For the Replica and ScanNet datasets, an episode consists of all the images generated from the same scene or apartment, thus, the batch size is given by the number of images in that episode, and 32 for Taskonomy. See App. A.2.1 for more details.

TTO (online): We continually adapt to a distribution shift defined by a corruption and severity. We assume that the test data arrives in a stream, and each data point has the same distribution

shift, e.g. Gaussian noise with a fixed standard deviation. Concretely, the difference with the episodic case is that the model weights are not reset after each iteration.

RNA: This uses the side model described in Sec. 3.2 to adapt the Baseline model where $\hat{\theta}(\phi) = h_\phi(z)$, i.e. model predictions are not passed.

RNA-Feedback: We also train the side model with the predictions from the Baseline model where $\hat{\theta}(\phi) = h_\phi(f_\theta(\mathcal{B}), z)$, i.e. both hints z and the model predictions $f_\theta(\mathcal{B})$ are used for adaptation. Note that this RNA variant requires two forward passes, unlike RNA, but it empirically improves the results in some cases (See Fig. 8).

RNA-Scratch: Instead of freezing the weights of the Baseline model, we train it jointly with the side-network. Note that this variant requires longer training than vanilla RNA.

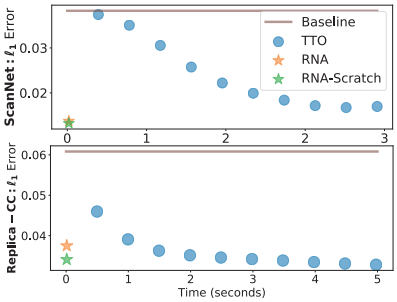

Figure 6: RNA achieves similar performance as TTO while being orders of magnitude faster. We compare how the ℓ_1 errors of the proposed adaptation mechanisms decrease over time. The errors are averaged over all episodes (and all corruptions for Replica-CC). RNA only requires a forward pass at test-time, 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*. RNA (scratch) gives a further boost in performance at the expense of longer training.

Dataset	Taskonomy			Replica		ScanNet
Shift	None	CC	3DCC	CDS	CC	CDS
Baseline UNet	2.68	5.74	4.75	1.75	6.08	3.30
Densification	1.72	1.72	1.72	2.50	4.19	2.35
TENT Wang et al. (2020)	5.51	5.51	4.48	2.03	6.09	4.03
TTO-Edges Sun et al. (2020)	2.70	5.69	4.74	1.73	6.14	3.28
TTO (Episodic)	1.62	2.99	2.31	1.72	3.31	1.85
TTO (Online)	1.13	1.48	1.34	1.82	3.16	1.85
RNA-Feedback	1.12	1.68	1.49	1.72	4.26	1.77
RNA	1.28	1.79	1.61	1.59	3.75	1.69
RNA-Scratch	1.07	1.35	1.26	1.58	3.50	1.66

Table 1: Quantitative evaluations of our proposed framework vs the baselines. ℓ_1 errors on the depth prediction task. (Lower is better. Multiplied by 100 for readability.) 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 supervision from Taskonomy is masked GT (fixed at 0.05% valid pixels) while that from Replica and ScanNet is sparse depth from SFM. Our variants notably outperform the baselines. See App. A.2.2 for the losses for different corruption types, supervison levels, and the results of applying RNA to other proxies (entropy and edges).

Baselines. We evaluate the following baselines:

Baseline: A network that maps from RGB to depth, with no test-time adaptation.

Densification: A network that maps from the given supervision to depth. This is a control baseline and shows what can be learned from the test-time supervision alone, without employing RGB information.

Episodic TTO with Entropy supervision (TENT): We adapt a Baseline UNet model trained with log-likelihood loss by optimizing the entropy of the predictions. This is to reveal the effectiveness of entropy as a supervision signal as proposed in Wang et al. (2020).

Episodic TTO with Sobel Edges supervision (TTO-Edges): We adapt the Baseline UNet model trained with an additional decoder that predicts a self-supervised task, similar to Sun et al. (2020). We choose to predict Sobel edges as it has been shown to be robust to certain shifts (Yeo et al., 2021). We optimize the error of the edges predicted by the model and edges extracted from the RGB image. This is to reveal the effectiveness of Sobel edge error as a supervision signal.

4.1.1 Comparisons of Adaptation Methods

RNA is efficient. Being able to adapt efficiently at test-time is crucial for some real-world problems. In Fig. 6, we compare the runtime of adaption with RNA and TTO. On average, for a given episode, RNA is able to attain similar performance as TTO in 0.01s, compared to TTO's 3-5s.

Furthermore, RNA's training is also efficient as it only requires training a small model i.e. 5% of the Baseline UNet model's parameters. Training RNA takes 36hrs on average on a single V100 gpu. Thus, RNA has a fixed overhead, but small added cost at test-time.

RNA's predictions are sharper than TTO. This can be seen in the last two rows of Fig. 4. As seen in Sec. 3.2, RNA was only trained with GT, while TTO was trained at test-time with sparse and possibly noisy supervision, thus TTO is more likely to lose details.

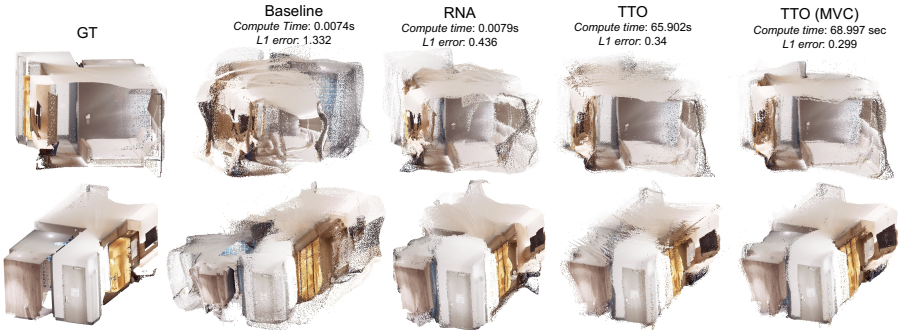

Figure 7: Visualizations for dense 3D reconstruction. Using appropriate hints from multi-view geometry can recover accurate 3D reconstructions. We report the average ℓ_1 error between ground truth 3D coordinates and the estimated ones. The titles above each column refers to the depth model used to get the reconstruction, as described in Sec. 4.1. TTO (MVC) corresponds to the predictions after multi-view consistency optimization as described in Sec.4.2. It can be seen that RNA and TTO improve the reconstructions over the baselines. Adding multi-view consistency hints further improves reconstructions. See the figure in App. A.4 for more examples and the error maps.

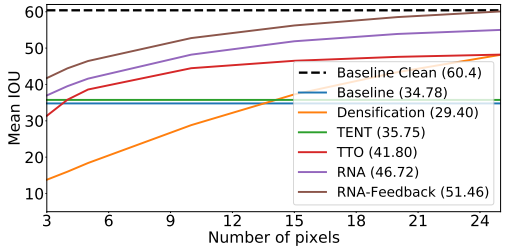

Figure 8: Quantitative evaluations using COCO-CC for semantic segmentation task. Each point shows the mean IOU over 15 corruptions and 5 severities. RNA significantly improves over baselines even with 3 pixel annotation per class. Black dashed line shows the mean IOU of baseline model for *clean* validation images, and is provided as a reference. Numbers in the legend denote averages over all supervision pixel counts. See App. A.5 for a breakdown.

Dataset	Clean	ImageNet-C	ImageNet-3DCC	ImageNet-V2
Baseline	23.85	61.66	54.97	37.15
Densification	95.50	95.50	95.50	95.50
TENT	24.67	46.19	47.13	37.07
TTO	24.72	**40.62**	42.90	36.77
RNA	**16.72**	41.21	**40.37**	**25.53**

Table 2: Quantitative evaluations of our proposed framework vs the baselines on ImageNet classification task. We evaluate on the clean validation set, ImageNet-C, ImageNet-3DCC, and ImageNet-V2. We report average error percentages for 1000-way classification task over all corruptions and severities. For TTO and RNA, we use 45-coarse labels as supervision. This corresponds to a 22x coarser supervision level compared to the 1000 classes that we are evaluating on. Our proposed methods significantly outperform the baselines. See App. A.6 for more results on other coarse sets and details.

TTO has an advantage over RNA under (3D)CC for high severities. RNA performs better than TTO for low severities (see App. A.2.2 for more details). However, as it was not exposed to any corruptions during training time, its performance gets worse than TTO at higher severities, which is exposed to corruptions at test-time.

We also trained a single model that takes as input a concatenation of the RGB image and sparse supervision. However, its average performance on Taskonomy-CC was 42.5% worse than RNA's (see App. A.2.2). Among the baselines, 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 Fig. 4, its predictions are missing fine-grained details. The results in Tab. 1 also confirm that task-unaware supervision signals, e.g. entropy or Sobel edges, are not as effective.

4.2 Evaluations on Dense 3D Reconstruction

This experiment combines multiple hints from multi-view geometry to solve a dense 3D reconstruction task robustly, under realistic assumptions about what information is available at test-time.

Evaluation data and models. We use 110 consecutive images that are part of a single navigation episode from hotel_0 from the Replica Dataset. These images have been corrupted by Gaussian noise. This results in depth predictions that have collapsed, thus, reconstructions that are unusable (see Fig. 7). The weights of the baseline depth model and a pre-trained optical flow model, RAFT (Teed & Deng, 2020), are adapted.

Test-Time Optimization details: First we adapt the weights of the depth and optical flow models independently. The results from this adaptation can be found in Sec. 4.1 (for depth) and App. A.3 (for optical flow). Next, the two models are adapted to make their predictions consistent with each other. This is achieved using the same process as Luo et al. (2020). See App. A.4 for details.

Results. Figure 7 shows the point cloud visualizations before and after test-time adaptation. Thus, using appropriate hints can allow test-time adaptation to recover a reasonably accurate and useful (e.g., for navigation without collisions) reconstruction from an unusable one.

4.3 Evaluations on Semantic segmentation

Model. We use an FCN Long et al. (2015) with a ResNet50 backbone using pre-trained weights from PyTorch Paszke et al. (2019). It is trained on a subset of COCO training dataset corresponding to 20 categories present in the Pascal VOC dataset [1].

Evaluation dataset and supervision. We perform our experiments on 5k validation images from COCO distorted with 15 corruptions over 5 severities from Common Corruptions. For supervision, we randomly generate pixel click annotations for each class in a given image.

Baselines. Similar to monocular depth estimation experiments, we include TENT and densification models as the baselines. See the App. A.5 for training details.

Results. We demonstrate quantitatively in Fig. 8 that TTO and RNA variants make use of the available sparse information to significantly improve upon baselines. Figure 5 (left) qualitatively shows the improvement made by our method with only 3 pixels as supervision. Figure 4 (left) shows further qualitative comparisons where RNA consistently outperforms the baselines.

4.4 Evaluations on Image classification

Model. We used a ResNet-50 pre-trained on ImageNet Deng et al. (2009) from PyTorch [2].

Evaluation dataset and supervision. We perform the evaluations on the clean ImageNet validation set, ImageNet-C (Hendrycks & Dietterich, 2019), ImageNet-3DCC (Kar et al., 2022b), and ImageNet-V2 (Recht et al., 2019). For supervision, we generate 45 coarse labels from the 1000-way ImageNet labels, i.e. making the labels 22x coarser, using the WordNet tree (Miller, 1994), similar to Huh et al. (2016a). See App. A.6 for more details and results for other coarse label sets.

Baselines. We include TENT (Wang et al., 2020) as the test-time optimization baseline and densification as the control baseline, which is the fixed mapping from the coarse labels to fine labels.

Results. Table 2 shows that the proposed method improves significantly over the baselines for all evaluation datasets. Thus, coarse supervision provides a useful signal for adaptation while requiring much less efforts than full annotation (Xu et al., 2021). See App. A.6 for more results.

5 Related Work

Our framework has two main components, **1.** defining useful test-time adaptation signals, **2.** exploring ways to use these signals. We give an overview of the topics relevant to these components.

Encoding inductive biases using task-aware losses. Adding losses to the target task loss function is a flexible way of encoding biases, e.g. energy should be conserved when predicting planetary movements (Alet et al., 2021), forecasts in foreign exchange markets should be symmetric across different currencies (Abu-Mostafa, 1995), depth predictions from video frames should be geometri-

[1] See here for details.

[2] Accessed from here.

cally consistent (Luo et al., 2020). We call these losses task-aware, as they are *tailored* to each task (See Sec. 3). Our paper shares a similar motivation by using task-aware losses for adaptation.

Test-time adaptation methods. Most of these works involve optimizing a self-supervised objective at test-time (Sun et al., 2020; Wang et al., 2020; Liu et al., 2021; Gandelsman et al., 2022; Gao et al., 2022; Zhang et al., 2021; Boudiaf et al., 2022; Liang et al., 2020). They differ in the choice of self-supervised objectives, e.g. prediction entropy (Wang et al., 2020), mutual information (Liang et al., 2020), and parameters optimized. However, as discussed in Sec. 3, and shown by Boudiaf et al. (2022); Gandelsman et al. (2022), existing methods can ***fail silently***, i.e. successful optimization of the proposed proxy loss does not necessarily result in better performance on the target task. We propose *task-aware losses* that boosts adaptation performance (See Sec. 3). The most similar work to our sparse depth proxy is that of Wang et al. (2019). They propose optimizing sparse depth at test-time to adapt a model, using simulated sensor data as supervision. We differ in assuming only access to RGB images to attain supervision via SFM (Schönberger & Frahm, 2016). Furthermore, we also show that this optimization process can be amortized (See Sec. 3.2) resulting in similar performance as performing optimization but at a fraction of the time.

Weak supervision for semantic tasks uses imperfect e.g. sparse, noisy supervision for learning. In the case of semantic segmentation, examples include scribbles (Lin et al., 2016), sparse annotations (Bearman et al., 2016; Papadopoulos et al., 2017; Shin et al., 2021; Zhi et al., 2021; Cheng et al., 2022). For classification, coarse labels are employed in different works (Xu et al., 2021; Huh et al., 2016a). These papers show that models trained with imperfect supervision can achieve similar performance to those trained with perfect supervision, i.e. full ground truth. We show that this imperfect supervision can also be used at test-time, resulting in significant performance improvements.

Conditioning methods uses *auxilliary inputs* to adapt a model. A popular method that has been adopted in many different domains e.g. style transfer (Dumoulin et al., 2016; Ghiasi et al., 2017), few shot learning (Oreshkin et al., 2018; Requeima et al., 2019), and settings involves performing feature-wise normalization (Dumoulin et al., 2018; Perez et al., 2018). The underlying principle involves training a model to use the auxilliary information to predict scaling and shift parameters that will be applied to the features of the target model. We use this mechanism in our framework and show that it is expressive, efficient, and able to generalize to unseen shifts.

Robustness methods *anticipate* the distribution shift that can occur and incorporate inductive biases into the model to help it generalize. Popular methods include data augmentations (Madry et al., 2017; Hendrycks et al., 2019; 2021; Kar et al., 2022a), pre-training (Eftekhar et al., 2021; Ranftl et al., 2022), architectural changes (Cohen & Welling, 2016) or ensembling from diverse cues (Yeo et al., 2021; Jain et al., 2022). In our case, we focus on adaptation mechanisms and identifying useful adaptation signals that can be used at test-time.

6 Conclusion and Limitations

We propose using proxies that are *task-aware*, making adaptation via *test-time optimization* (TTO) reliable. We show that this optimization process can be amortized with a side-network, which we call *rapid network adaptation* (RNA). We show empirically that RNA generalizes to distribution shifts, and is orders of magnitude faster than TTO.

We briefly discuss some of the limitations of our framework:

Finding hints for a given task: We demonstrate the usefulness of hints for several core vision tasks, but there are still many others which can benefit from hints, e.g. object detection or pose estimation. Finding such hints requires knowledge of the target task. We believe this work can open up a promising direction for incorporating this knowledge into adaptation process.

Analysis of different amortization methods. While we do not extensively explore different ways of amortizing TTO with a side-network, e.g. HyperNetworks (Ha et al., 2016), or cross-attention (Vaswani et al., 2017), such efforts could be worthwhile. We leave it to future work.

Combining proxies from different sources. Our framework can easily be extended to take in multiple proxies, e.g. depth from LiDAR sensors, in addition to the sparse depth from SFM. These proxies have their own strength and weaknesses, thus, combining them can result in more robust predictions.

REFERENCES

Yaser S Abu-Mostafa. Hints. *Neural computation*, 7(4):639–671, 1995.

Ferran Alet, Maria Bauza, Kenji Kawaguchi, Nurullah Giray Kuru, Tomás Lozano-Pérez, and Leslie Kaelbling. Tailoring: encoding inductive biases by optimizing unsupervised objectives at prediction time. *Advances in Neural Information Processing Systems*, 34:29206–29217, 2021.

Brandon Amos. Tutorial on amortized optimization for learning to optimize over continuous domains. *arXiv preprint arXiv:2202.00665*, 2022.

Amy Bearman, Olga Russakovsky, Vittorio Ferrari, and Li Fei-Fei. What's the point: Semantic segmentation with point supervision. In *European conference on computer vision*, pp. 549–565. Springer, 2016.

Malik Boudiaf, Romain Mueller, Ismail Ben Ayed, and Luca Bertinetto. Parameter-free online test-time adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8344–8353, 2022.

Bowen Cheng, Omkar Parkhi, and Alexander Kirillov. Pointly-supervised instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2617–2626, 2022.

Taco Cohen and Max Welling. Group equivariant convolutional networks. In *International conference on machine learning*, pp. 2990–2999. PMLR, 2016.

Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5828–5839, 2017.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255. Ieee, 2009.

Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pp. 224–236, 2018.

Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. A learned representation for artistic style. *arXiv preprint arXiv:1610.07629*, 2016.

Vincent Dumoulin, Ethan Perez, Nathan Schucher, Florian Strub, Harm de Vries, Aaron Courville, and Yoshua Bengio. Feature-wise transformations. *Distill*, 2018. doi: 10.23915/distill.00011. https://distill.pub/2018/feature-wise-transformations.

Ainaz Eftekhar, Alexander Sax, Jitendra Malik, and Amir Zamir. Omnidata: A scalable pipeline for making multi-task mid-level vision datasets from 3d scans. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10786–10796, 2021.

Yossi Gandelsman, Yu Sun, Xinlei Chen, and Alexei A Efros. Test-time training with masked autoencoders. *arXiv preprint arXiv:2209.07522*, 2022.

Jin Gao, Jialing Zhang, Xihui Liu, Trevor Darrell, Evan Shelhamer, and Dequan Wang. Back to the source: Diffusion-driven test-time adaptation. *arXiv preprint arXiv:2207.03442*, 2022.

Golnaz Ghiasi, Honglak Lee, Manjunath Kudlur, Vincent Dumoulin, and Jonathon Shlens. Exploring the structure of a real-time, arbitrary neural artistic stylization network. *arXiv preprint arXiv:1705.06830*, 2017.

David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019.

Dan Hendrycks, Norman Mu, Ekin D Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. Augmix: A simple data processing method to improve robustness and uncertainty. *arXiv preprint arXiv:1912.02781*, 2019.

Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, et al. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8340–8349, 2021.

Minyoung Huh, Pulkit Agrawal, and Alexei A. Efros. What makes imagenet good for transfer learning?, 2016a. URL https://arxiv.org/abs/1608.08614.

Minyoung Huh, Pulkit Agrawal, and Alexei A Efros. What makes imagenet good for transfer learning? *arXiv preprint arXiv:1608.08614*, 2016b.

Saachi Jain, Dimitris Tsipras, and Aleksander Madry. Combining diverse feature priors. In *International Conference on Machine Learning*, pp. 9802–9832. PMLR, 2022.

Oğuzhan Fatih Kar, Teresa Yeo, Andrei Atanov, and Amir Zamir. 3d common corruptions and data augmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18963–18974, 2022a.

Oğuzhan Fatih Kar, Teresa Yeo, and Amir Zamir. 3d common corruptions for object recognition. In *ICML 2022 Shift Happens Workshop*, 2022b.

Jian Liang, Dapeng Hu, and Jiashi Feng. Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. In *International Conference on Machine Learning*, pp. 6028–6039. PMLR, 2020.

Di Lin, Jifeng Dai, Jiaya Jia, Kaiming He, and Jian Sun. Scribblesup: Scribble-supervised convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3159–3167, 2016.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, pp. 740–755. Springer, 2014.

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 thrive? *Advances in Neural Information Processing Systems*, 34:21808–21820, 2021.

Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3431–3440, 2015.

Xuan Luo, Jia-Bin Huang, Richard Szeliski, Kevin Matzen, and Johannes Kopf. Consistent video depth estimation. *ACM Transactions on Graphics (ToG)*, 39(4):71–1, 2020.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

George A. Miller. WordNet: A lexical database for English. In *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*, 1994. URL https://aclanthology.org/H94-1111.

Boris Oreshkin, Pau Rodríguez López, and Alexandre Lacoste. Tadam: Task dependent adaptive metric for improved few-shot learning. *Advances in neural information processing systems*, 31, 2018.

Dim P Papadopoulos, Jasper RR Uijlings, Frank Keller, and Vittorio Ferrari. Extreme clicking for efficient object annotation. In *Proceedings of the IEEE international conference on computer vision*, pp. 4930–4939, 2017.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pp. 8024–8035, 2019.

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(3), 2022.

Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? In *International Conference on Machine Learning*, pp. 5389–5400. PMLR, 2019.

Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. *arXiv preprint arXiv:1904.09237*, 2019.

James Requeima, Jonathan Gordon, John Bronskill, Sebastian Nowozin, and Richard E Turner. Fast and flexible multi-task classification using conditional neural adaptive processes. *Advances in Neural Information Processing Systems*, 32, 2019.

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 Vision and Pattern Recognition*, pp. 12892–12901, 2022.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-assisted Intervention*, pp. 234–241. Springer, 2015.

Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Gyungin Shin, Weidi Xie, and Samuel Albanie. All you need are a few pixels: semantic segmentation with pixelpick. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1687–1697, 2021.

Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J. Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, Anton Clarkson, Mingfei Yan, Brian Budge, Yajie Yan, Xiaqing Pan, June Yon, Yuyang Zou, Kimberly Leon, Nigel Carter, Jesus Briales, Tyler Gillingham, Elias Mueggler, Luis Pesqueira, Manolis Savva, Dhruv Batra, Hauke M. Strasdat, Renzo De Nardi, Michael Goesele, Steven Lovegrove, and Richard Newcombe. The Replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019.

Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *International conference on machine learning*, pp. 9229–9248. PMLR, 2020.

Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *European conference on computer vision*, pp. 402–419. Springer, 2020.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. In *International Conference on Learning Representations*, 2020.

Tsun-Hsuan Wang, Fu-En Wang, Juan-Ting Lin, Yi-Hsuan Tsai, Wei-Chen Chiu, and Min Sun. Plug-and-play: Improve depth prediction via sparse data propagation. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 5880–5886. IEEE, 2019.

Yuanhong Xu, Qi Qian, Hao Li, Rong Jin, and Juhua Hu. Weakly supervised representation learning with coarse labels. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10593–10601, 2021.

Teresa Yeo, Oğuzhan Fatih Kar, and Amir Zamir. Robustness via cross-domain ensembles. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 12189–12199, October 2021.

Amir Zamir, Alexander Sax, Teresa Yeo, Oğuzhan Kar, Nikhil Cheerla, Rohan Suri, Zhangjie Cao, Jitendra Malik, and Leonidas Guibas. Robust learning through cross-task consistency. *arXiv preprint arXiv:2006.04096*, 2020.

Amir R Zamir, Alexander Sax, William Shen, Leonidas J Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling task transfer learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3712–3722, 2018.

Marvin Zhang, Sergey Levine, and Chelsea Finn. Memo: Test time robustness via adaptation and augmentation. *arXiv preprint arXiv:2110.09506*, 2021.

Shuaifeng Zhi, Tristan Laidlow, Stefan Leutenegger, and Andrew J Davison. In-place scene labelling and understanding with implicit scene representation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 15838–15847, 2021.

A APPENDIX

A.1 OVERVIEW

We provide further details and evaluations about our method in the following sections.

- Additional details and results for monocular depth (Appendix A.2)
- Additional details and results for optical flow (Appendix A.3)
- Additional details and results for dense 3D reconstruction (Appendix A.4)
- Additional details and results for semantic segmentation (Appendix A.5)
- Additional details and results for image classification (Appendix A.6)

A.2 MONOCULAR DEPTH

A.2.1 TRAINING DETAILS

All networks for our method and baselines use the same UNet architecture Ronneberger et al. (2015) and were trained with AMSGrad Reddi et al. (2019), with a learning rate of 5×10^{-4}, weight decay 2×10^{-6} and batch size of 64. The h models were trained by randomly masking the GT, with sparsity ratio ranging from 0% to 0.025%.

To simulate the sparse depth from SFM, we use the model from DeTone et al. (2018) to extract keypoints locations. This creates a sparse binary mask that we apply on the GT. We then add Gaussian Noise to the masked GT to account for outliers at test-time.

We assume that we are given a function f_θ to be adapted. The choice of function g that extracts the adaptation signal is discussed in Sec. 3.1.

Test-Time Opimization (TTO) minimizes the following loss function:

$$\min_{\hat{\theta}} \mathcal{L} = \sum_{n=1}^{N} \| f_\theta(x_n) \odot m_n, z_n \|_1. \tag{1}$$

ℓ is the ℓ_1 norm for depth. N is the number of datapoints, $\odot$ is the element-wise product and $\hat{\theta}$ is the subset of parameters of f_θ that is updated i.e. $\hat{\theta} \subseteq \theta$.

Rapid Network Adaptation (RNA). We aim to fuse the additional information provided by g at test-time into f_θ via a small network, h_ϕ. This is also called conditioning as the computations done by f_θ will be *conditioned* on the information provided by g.

In this paper, we use feature-wise modulations. Concretely, we insert k Feature-wise Linear Modulation (FiLM) layers into the network f_θ, we denote this network as f_θ. Each FiLM layer performs the following operation $\text{FiLM}(x_i; \gamma_i, \beta_i) = \gamma_i \odot \mathbf{x}_i + \beta_i$, where $\mathbf{x_i}$ is the activation of layer i. h_ϕ is trained to output the coefficients $\{\gamma_i, \beta_i\}$ of all k FiLM layers given g. h_ϕ is trained using the following loss function:

$$\min_{\phi} \mathcal{L} = \sum_{n=1}^{N} \| f_\theta(x_n; h_\phi(y_n \odot \bar{m}_n)) - y_n \|_1 \tag{2}$$

where $\bar{m}_n$ is the mask that simulates the sparsity pattern that will be encountered at test-time. If g extracts sparse depth via SFM, we use a feature extractor that samples keypoints to generate $\bar{m}$, and add Gaussian noise to $y_n \odot \bar{m}_n$ to account for potential outliers. At test-time, RNA only requires a forward pass through h_ϕ and f_θ.

Test-Time Optimization details. We use the same optimization parameters as in training time. Optimization is done for 10 iterations for each batch. With the exception of the TENT baseline, all parameters of the model were updated. For TENT, only the GroupNorm parameters were updates as it results in more stable optimization.

A.2.2 ADDITIONAL RESULTS

RNA with existing supervision signals. We show that one can train a side-network to also use supervision from existing sources e.g. prediction entropy and Sobel edges to get better performance at test time. As described in 4.1, to use supervision from entropy, we train a Baseline UNet model with NLL loss. The model outputs the prediction and its uncertainty estimate of the predictions. We show results with the model from Yeo et al. (2021) that calibrates the uncertainty estimates as it was shown to predict uncertainties that are better correlated with error,

ℓ_1 error for different levels of GT supervision. We show how the error changes with increasing GT supervision for our proposed methods and the baselines.

RNA performs better a than training a single model that takes as input the RGB image and sparse supervision concatenated. We call this model multi-domain. For 0.05% GT supervision, RNA has an average loss, over all distortions and severities of 0.0179 while the multi-domain model has an average loss of 0.0255.

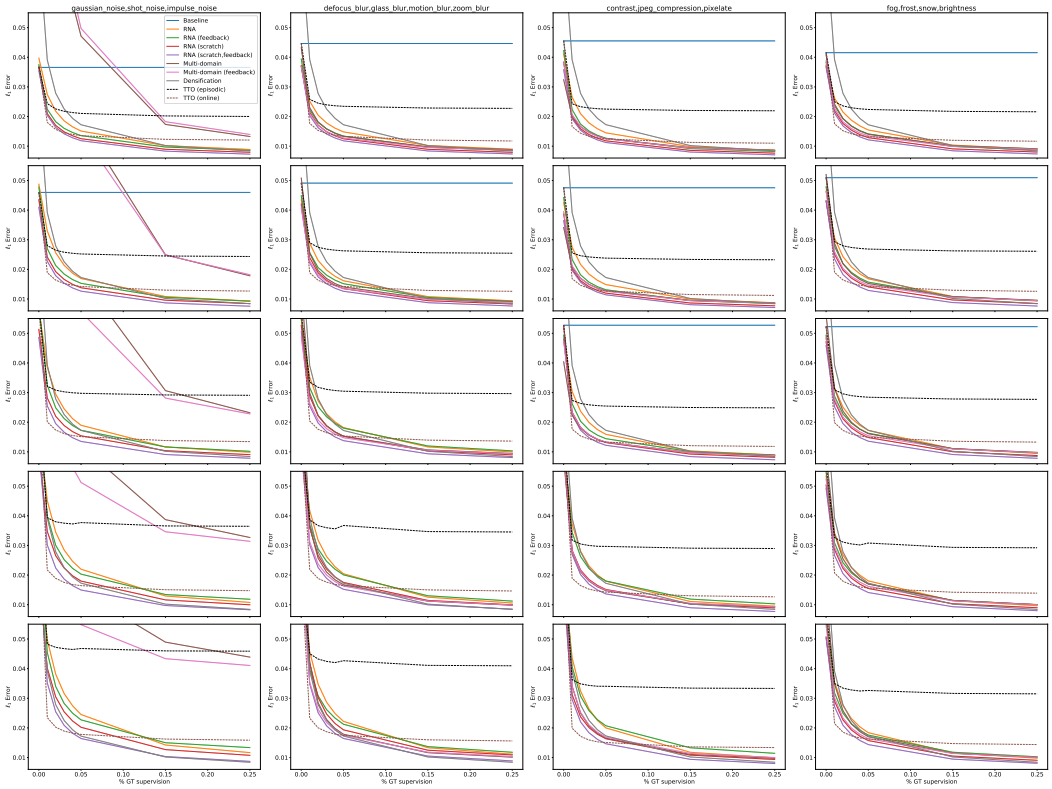

Figure 9: ℓ_1 error vs different levels of GT supervision. These results are on Taskonomy-CC data, as described in Sec 4.1. Each curve shows an average over all the distortions listed in the title.

Method/Supervision	Sobel edge	Entropy	Entropy (calibrated)	Sparse GT
RNA	0.29	0.07	0.12	4.06
TTO (GN)	-0.02	-0.11	0.04	0.55
TTO (F)	0.06	-1.21	0.06	2.76

Table 3: RNA can be used with existing supervision signals. F denotes TTO by opimizing all parameters and, GN, TTO by optimizing only GN parameters.

Qualitative results. We provide more qualitative results in Fig. 16 where our method outperforms the baselines qualitatively.

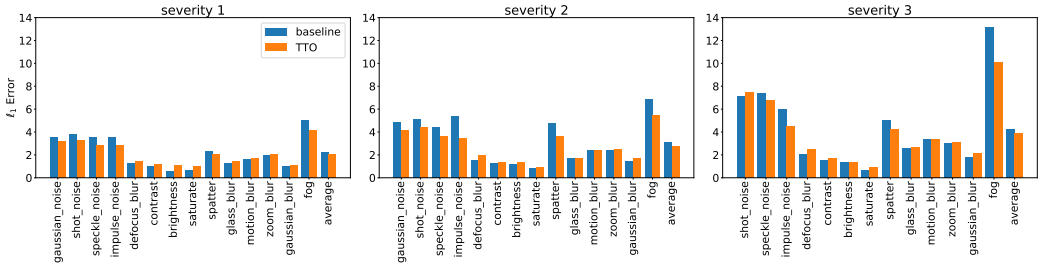

Figure 10: Quantiative results for optical flow. These results were attained from Replica+CC data, as described in Sec 4.1.

A.3 Optical Flow Experiments

To predict optical flow, we use a pre-trained RAFT model from Teed & Deng (2020). We use sparse optical flow as our supervision signal, attained from keypoint matching between images. We perform TTO to adapt the model. We use the same episodes from Replica+CC as described in Sec. 4.1. TTO was done for 10 iterations for each episode, all parameters of the RAFT model were updated. Figure 10 shows the results. Adaptation with TTO results in an 8.5% improvement over the baseline.

A.4 Dense 3D Reconstruction

Test-Time Optimization details. As mentioned in the main paper, we achieve multi-view consistency using the same process as Luo et al. (2020), where: **1.** every pixel is backprojected into 3D world-coordinates using the estimated depth and camera poses, **2.** optical-flow predictions are employed to establish dense correspondences across pixels, **3.** weights of the depth model are optimized to minimize the discrepancy between the estimated 3D world coordinates of corresponding pixels.

We provide additional qualitative results and the corresponding error images in Fig. 11.

A.5 Semantic Segmentation

A.5.1 Training Details

TTO. We optimize by SGD with 0.0001 learning rate, 0.9 momentum, batch size 2 for 10 iterations per batch.

TENT. Following Wang et al. (2020), we optimize by SGD with 0.0001 learning rate, 0.9 momentum, batch size 1. As TENT is unstable for online and multi-iteration optimization, we restart the model after each batch, i.e. episodic optimization, and run 1 iteration per batch.

TENT (all). In contrast to Wang et al. (2020) which only updates batch normalization parameters at test-time, we also included a TENT baseline that optimizes all parameters. We optimize by SGD with 0.00001 learning rate, 0.9 momentum, batch size 1. We run the optimization for each image for 10 iterations to be comparable to the TTO model. Note that we reduced learning rate by a factor of 10 as TENT was unstable. Since TENT and TENT (all) models perform similarly, we only show the TENT results in the main paper. See A.5.2 for all results.

RNA. As an encoder we used a small CNN with 3 downsampling blocks. We trained the FiLM generator with frozen segmentation model on the *clean* COCO training dataset with cross entropy losses. During the training we sparsify the target segmentation mask uniformly in [0,30] pixels to generate sparse ground truth inputs to the encoder. We optimize by Adam with 0.0001 learning rate and 0.0001 weight decay. We select the model with highest mean IOU on the clean validation set.

RNA-feedback. This model is trained with two forward passes. During the first pass the input to FiLM encoder is sparse ground truth and zeros as prediction. After this, in the second pass the encoder takes the sparse ground truth and the prediction from the previous pass as input. We

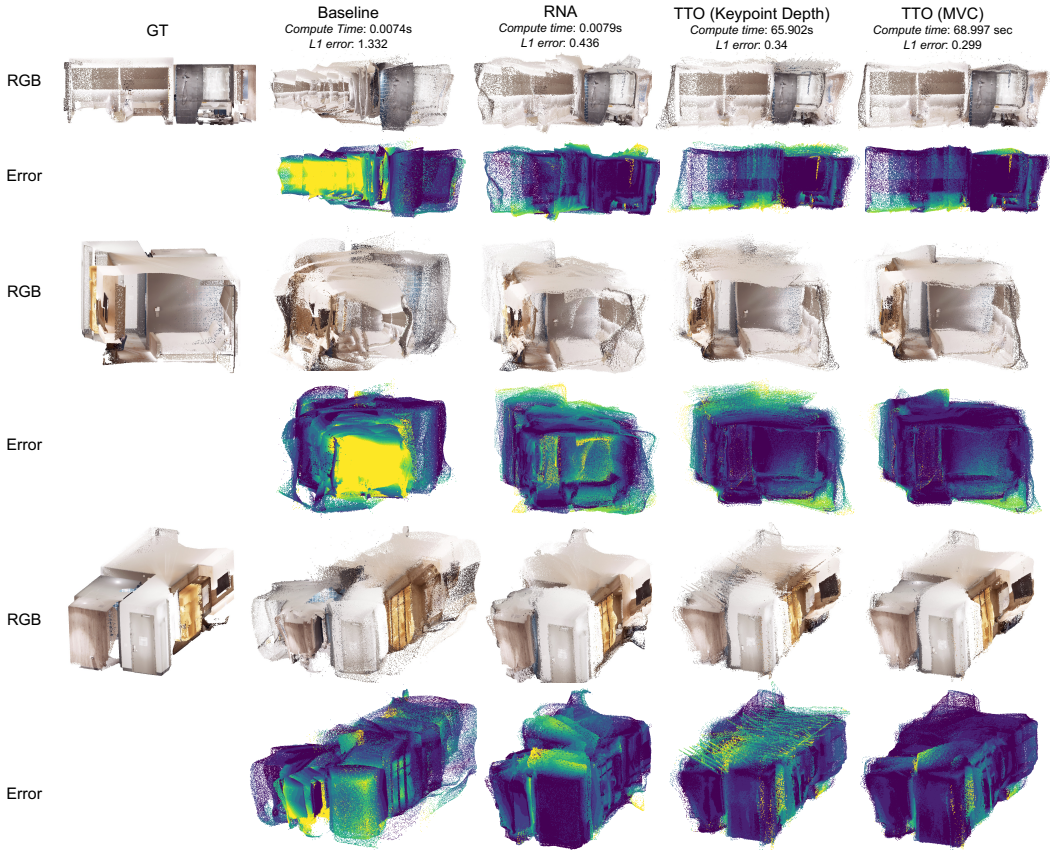

Figure 11: Extension of Fig. 7 to show more examples and error maps.

compute the cross-entropy loss for the prediction of second pass. We use the same hyperparameters as RNA and again select the model with highest mean IOU on the clean validation set.

Densification. It uses the same FCN-ResNet50 model as other baselines and is trained with the same setup as RNA variants.

A.5.2 Additional Results

Quantitative Results. We provide the results for mean IOU vs number of pixels for different severities in Fig. 17. Figures 18, 19, 20, 21, 22, 23, 24 give a breakdown of the performance for our methods and baselines for each corruption, severity level, and number of pixel annotations.

Qualitative Results. We include additional qualitative comparisons between our methods and baselines in Figures 25, 26, 27.

A.6 Image classification

A.6.1 Generating Coarse Label Sets

One method to generate coarse label sets using WordNet tree Miller (1994) is proposed in Huh et al. (2016a). This method and other clustering methods create imbalanced coarse labels and too many ImageNet classes are assigned to coarse labels that are either too coarse or too fine-grained (See Figure 12 and Table 8 for the statistics). As we aim to use coarse supervision to adapt models at test-time, we focus on generating more balanced coarse labels, as explained below.

To generate balanced coarse label sets, instead of going from top to bottom or bottom to top for a fixed depth in WordNet tree, we follow a different approach. For each ImageNet class we go up

until we get to a hypernym that has a certain number of hyponyms that are ImageNet classes. That number determines the coarsity level of the coarse label. To achieve this, we use a priority-based selection criteria where we define certain ranges to reach a given coarsity level. Using this approach, we created three different coarse sets with 26, 45, and 85 coarse labels. See Tables 4, 5, 6, 7 for the coarse labels and their IDs. The resulting sets are more balanced than the 127-label set provided in (Huh et al., 2016a). See Figures 12 13 14 15 and Table 8 for more details about the statistics of the coarse label sets.

n01428580 soft-finned fish	n02807260 bath linen	n03597469 jewelry	n04509592 uniform
n01482330 shark	n02856463 board	n03613592 key	n04571292 weight
n01495701 ray	n02858304 boat	n03619396 kit	n06595351 magazine
n01503061 bird	n02924116 bus	n03664943 ligament	n06793231 sign
n01629276 salamander	n02942699 camera	n03666917 lighter-than-air craft	n06874019 light
n01639765 frog	n02954340 cap	n03678362 litter	n07557434 dish
n01662784 turtle	n03035510 cistern	n03764276 military vehicle	n07560652 fare
n01674990 gecko	n03039947 cleaning implement	n03825080 nightwear	n07579575 entree
n01676755 iguanid	n03094503 container	n03837422 oar	n07582609 dip
n01685439 teiid lizard	n03101156 cooker	n03880531 pan	n07611358 frozen dessert
n01687665 agamid	n03122748 covering	n03896233 passenger train	n07612996 pudding
n01689411 anguid lizard	n03151500 cushion	n03906997 pen	n07680932 bun
n01691951 venomous lizard	n03183080 device	n03961939 platform	n07681926 cracker
n01692864 lacertid lizard	n03236735 dress	n03964744 plaything	n07683786 loaf of bread
n01693783 chameleon	n03241093 drill rig	n03990474 pot	n07707451 vegetable
n01694709 monitor	n03257586 duplicator	n04015204 protective garment	n07800740 fodder
n01697178 crocodile	n03278248 electronic equipment	n04077734 rescue equipment	n07829412 sauce
n01698434 alligator	n03294833 eraser	n04099429 rocket	n07882497 concoction
n01703569 ceratopsian	n03309808 fabric	n04100174 rod	n07891726 wine
n01726692 snake	n03405725 furniture	n04125853 safety belt	n07929519 coffee
n01767661 arthropod	n03414162 game equipment	n04128837 sailing vessel	n07930554 punch
n01861778 mammal	n03419014 garment	n04185071 sharpener	n09214060 bar
n01909422 coelenterate	n03441112 glove	n04194289 ship	n09287968 geological formation
n01922303 worm	n03446832 golf equipment	n04235291 sled	n09289709 globule
n01940736 mollusk	n03450516 gown	n04264914 spacecraft	n09820263 athlete
n02316707 echinoderm	n03472232 gymnastic apparatus	n04285622 sports implement	n10019552 diver
n02512938 food fish	n03476083 hairpiece	n04317420 stick	n10401829 participant
n02605316 butterfly fish	n03497657 hat	n04341686 structure	n11669921 flower
n02606384 damselfish	n03510583 heavier-than-air craft	n04377057 system	n12992868 fungus
n02638596 ganoid	n03513137 helmet	n04447443 toiletry	n13134947 fruit
n02642644 scorpaenid	n03528263 home appliance	n04451818 tool	n15074962 tissue
n02652668 plectognath	n03540267 hosiery	n04500060 turner	

Table 4: The classes used in the 127-coarse label set from Huh et al. (2016a). See Figure 12 for more details.

n00002137 abstraction	n02084071 dog	n03100490 conveyance	n03839993 obstruction
n00004475 organism	n02087122 hunting dog	n03122748 covering	n04014297 protective covering
n00007347 causal agent	n02087551 hound	n03125870 craft	n04081844 restraint
n00020090 substance	n02092468 terrier	n03183080 device	n04170037 self-propelled vehicle
n00020827 matter	n02098550 sporting dog	n03257877 durables	n04285146 sports equipment
n00021265 food	n02103406 working dog	n03278248 electronic equipment	n04341686 structure
n00021939 artifact	n02104523 shepherd dog	n03294048 equipment	n04447443 toiletry
n01503061 bird	n02120997 feline	n03297735 establishment	n04451818 tool
n01525720 oscine	n02159955 insect	n03309808 fabric	n04524313 vehicle
n01627424 amphibian	n02329401 rodent	n03405265 furnishing	n04530566 vessel
n01661091 reptile	n02370806 ungulate	n03405725 furniture	n04531098 vessel
n01661818 diapsid	n02394477 even-toed ungulate	n03414162 game equipment	n04576211 wheeled vehicle
n01674464 lizard	n02469914 primate	n03419014 garment	n04586932 wind instrument
n01726692 snake	n02484322 monkey	n03528263 home appliance	n07570720 nutriment
n01767661 arthropod	n02512053 fish	n03563967 implement	n07705931 edible fruit
n01844917 aquatic bird	n02528163 teleost fish	n03574816 instrument	n07707451 vegetable
n01861778 mammal	n02778669 ball	n03575240 instrumentality	n09287968 geological formation
n01886756 placental	n02913152 building	n03699975 machine	n12992868 fungus
n01905661 invertebrate	n02958343 car	n03733925 measuring instrument	n13134947 fruit
n02000954 wading bird	n03051540 clothing	n03738472 mechanism	
n02075296 carnivore	n03076708 commodity	n03791235 motor vehicle	
n02083346 canine	n03094503 container	n03800933 musical instrument	

Table 5: The classes used in the 85-coarse label set. See Figure 13 for more details.

n00002137 abstraction	n02075296 carnivore	n03093574 consumer goods	n03738472 mechanism
n00004475 organism	n02083346 canine	n03094503 container	n03800933 musical instrument
n00007347 causal agent	n02084071 dog	n03100490 conveyance	n04014297 protective covering
n00019128 natural object	n02087551 hound	n03122748 covering	n04081844 restraint
n00020827 matter	n02092468 terrier	n03183080 device	n04341686 structure
n00021939 artifact	n02098550 sporting dog	n03294048 equipment	n04524313 vehicle
n01503061 bird	n02103406 working dog	n03309808 fabric	n04531098 vessel
n01627424 amphibian	n02120997 feline	n03405265 furnishing	n09287968 geological formation
n01661091 reptile	n02370806 ungulate	n03563967 implement	n12992868 fungus
n01861778 mammal	n02441326 musteline mammal	n03574816 instrument	
n01886756 placental	n02469914 primate	n03575240 instrumentality	
n01905661 invertebrate	n02512053 fish	n03699975 machine	

Table 6: The classes used in the 45-coarse label set. See Figure 14 for more details.

n00002137 abstraction	n01661091 reptile	n03257877 durables	n04524313 vehicle
n00004475 organism	n01861778 mammal	n03294048 equipment	n04531098 vessel
n00007347 causal agent	n01905661 invertebrate	n03309808 fabric	n09287968 geological formation
n00020827 matter	n02075296 carnivore	n03405265 furnishing	n12992868 fungus
n00021939 artifact	n02512053 fish	n03563967 implement	n13134947 fruit
n01503061 bird	n03122748 covering	n03575240 instrumentality	
n01627424 amphibian	n03183080 device	n04341686 structure	

Table 7: The classes used in the 26-coarse label set. See Figure 15 for more details.

Coarse Label Set	Min	Max	Mean	Median	Mode	Standard Deviation
127-way	1	218	88.40	59.0	218	79.96
85-way	6	522	44.05	24.0	26	59.10
45-way	7	522	59.33	50.0	67	60.27
26-way	7	522	105.86	67.0	158	84.77

Table 8: The statistics of coarse label sets. See Figures 12 13 14 15 for more details.

A.6.2 TRAINING DETAILS

The baseline model used for classification is ResNet50 trained on ImageNet.

TTO. We optimize by SGD with 0.00025 learning rate, 0.9 momentum and batch size 64, following (Wang et al., 2020). The following loss function is used for TTO, which is a linear combination of cross-entropy loss on the summation of probabilities of all of the classes in the coarse label set, and the entropy of the predictions:

$$\min_{\phi} \mathcal{L} = -\log \sum_{c \in \mathbb{S}} p_c + w_e \sum_{c} -p_c \log p_c \tag{3}$$

where p_c is the probability of class c and $\mathbb{S}$ is the set of classes that are present in the coarse label.

TENT. We used the optimizer and parameters that were reported in Wang et al. (2020) to adapt TENT. For both TTO and TENT, we optimize the transformation parameters of the normalization layers and estimate the normalization statistics from the current batch. Note that for each batch we re-evaluate after the updates (in our experiments we run 1 iteration per batch) to get the final predictions.

RNA. We optimize by Adam with 0.0001 learning rate, 0.0001 weight decay, batch size 64 and for about 50 epochs. The FiLM generator we used has an encoder-decoder structure. The encoder has one hidden layer with 128 nodes and 64 nodes for the output layer. The decoder for each FiLM layer has one hidden layer with 64 nodes and the output size is equal to the number of FiLM layer parameters. The FiLM layers are inserted between normalization layers and ReLUs. During training all of the model parameters are fixed and only the FiLM generator parameters are being trained, and cross-entropy loss is minimized.

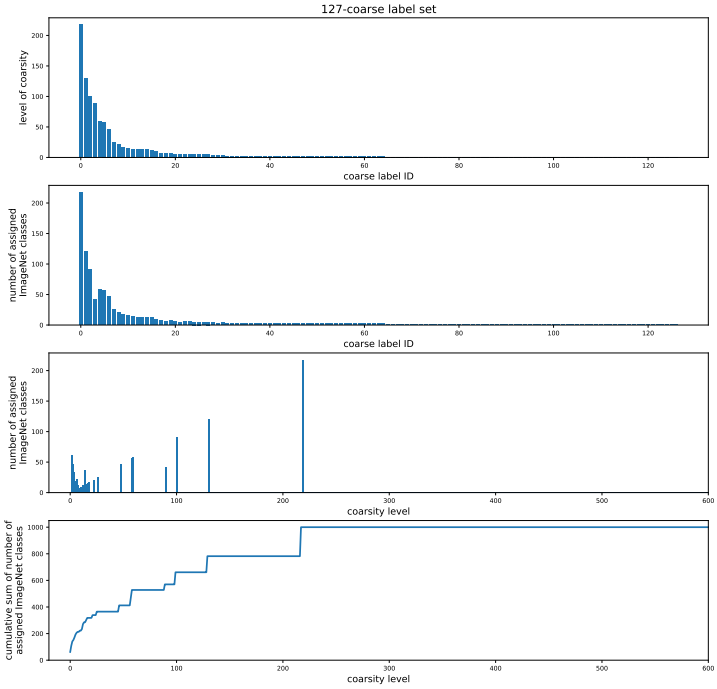

Figure 12: Distribution of 127-coarse label set from Huh et al. (2016a). 62 ImageNet classes are using coarse labels with coarsity level of 1, which means for 62 classes the hint is the correct label itself. 171 classes are using coarse labels with coarsity level of 5 or less, 222 classes are using coarse labels with coarsity level of 10 or less, and 218 classes are using the coarse labels with coarsity level of 200 or more.

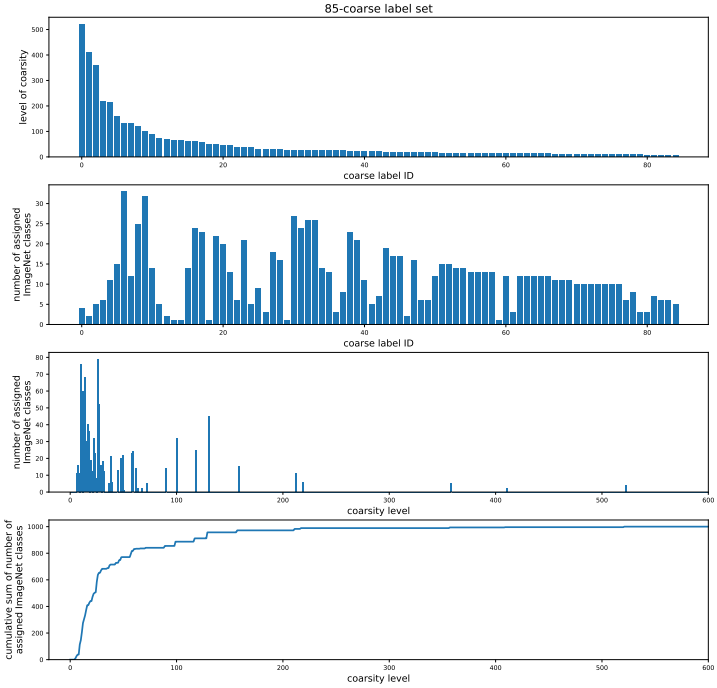

Figure 13: Distribution of 85-coarse label set. None of ImageNet classes are using coarse labels with coarsity level of 5 or less, 114 classes are using coarse labels with coarsity level of 10 or less, and 28 classes are using the coarse labels with coarsity level of 200 or more.

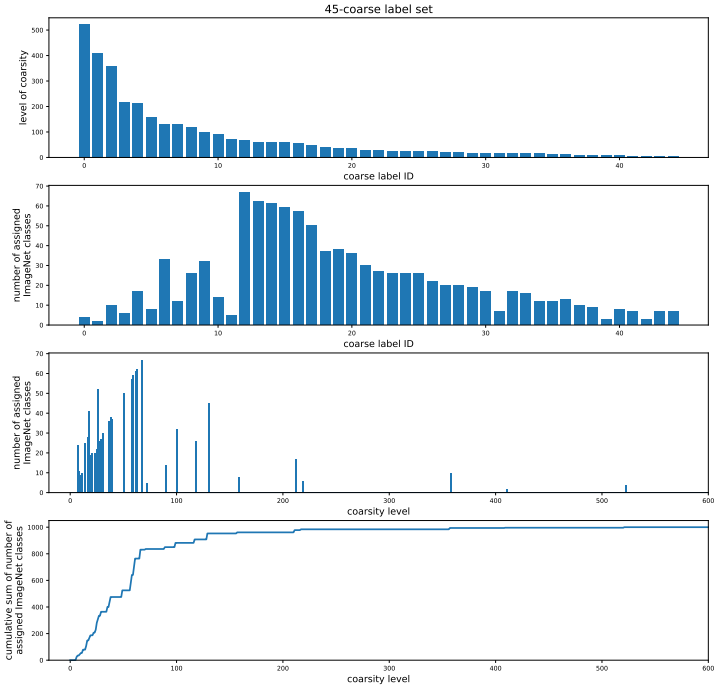

Figure 14: Distribution of 45-coarse label set. None of ImageNet classes are using coarse labels with coarsity level of 5 or less, 44 classes are using coarse labels with coarsity level of 10 or less, and 39 classes are using the coarse labels with coarsity level of 200 or more.

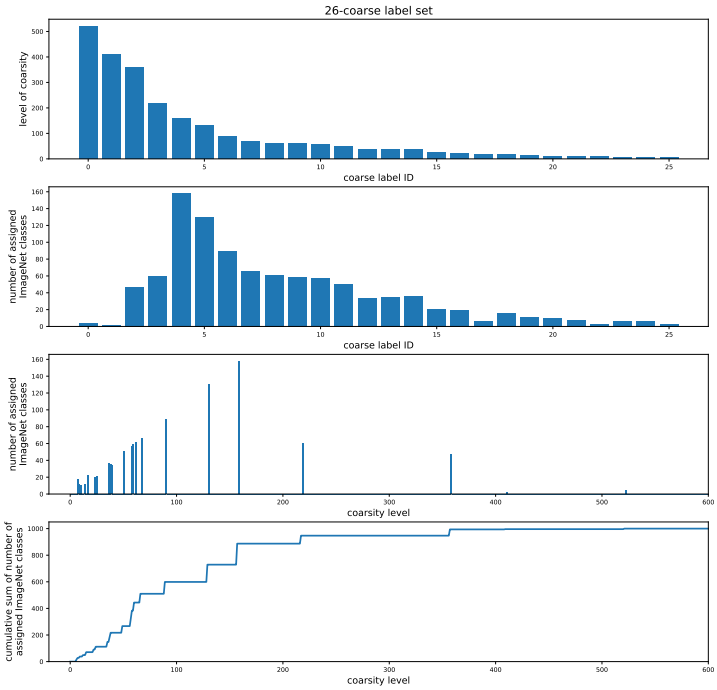

Figure 15: Distribution of 26-coarse label set. None of ImageNet classes are using coarse labels with coarsity level of 5 or less, 38 classes are using coarse labels with coarsity level of 10 or less, and 113 classes are using the coarse labels with coarsity level of 200 or more.

A.6.3 ADDITIONAL RESULTS

In Figures 28,29,30 we provide a breakdown of performance against individual corruptions from ImageNet-C and ImageNet-3DCC using 26-,45-, and 85-way coarse labels. We also included the results when we used 127-way coarse labels from Huh et al. (2016a) in Fig. 31. Note that this coarse set has imbalances as explained in A.6.1, yet our methods can still benefit from it.

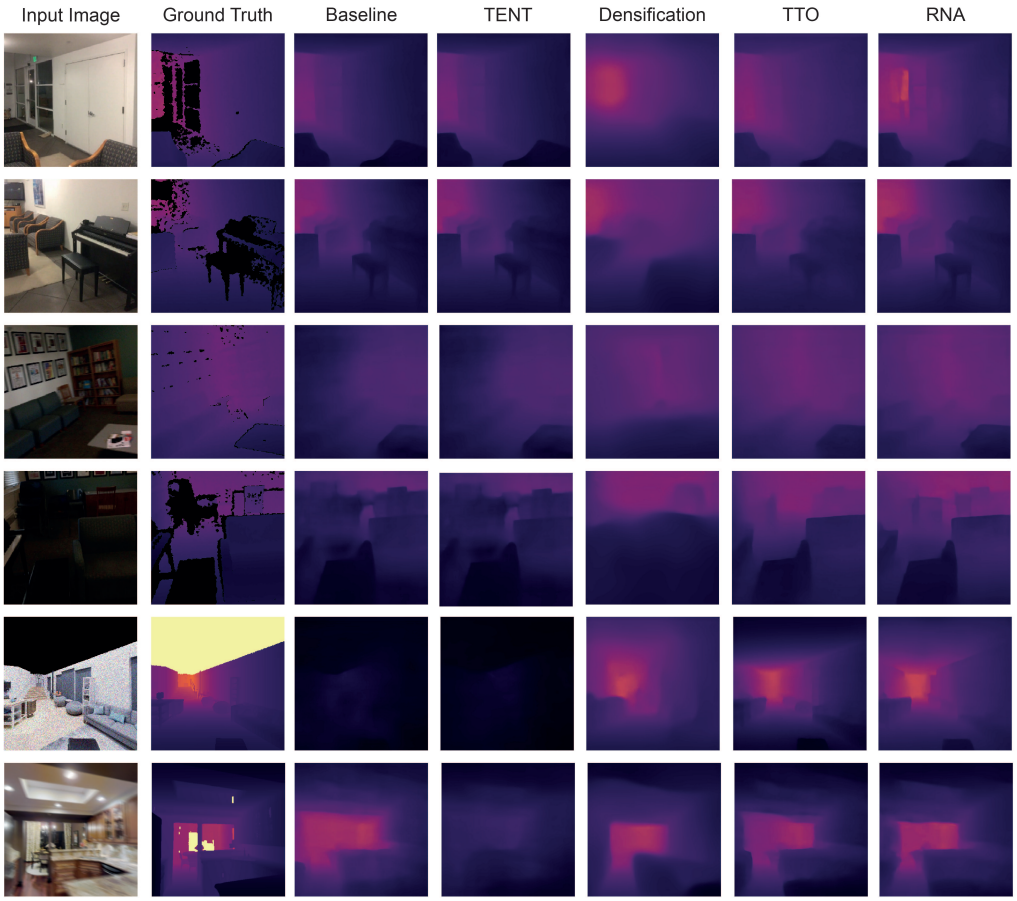

Figure 16: Supplementary results for monocular depth estimation. Qualitative comparison of our method vs baselines on query images from ScanNet, Replica, and Taskonomy datasets.

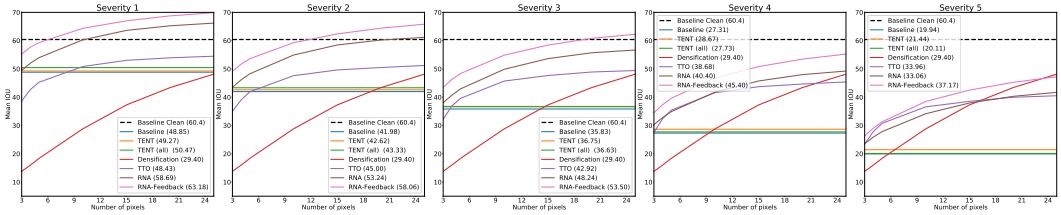

Figure 17: Supplementary results for semantic segmentation. Mean IOU vs number of pixels for different severities. Numbers in the legend denote the average over all pixel levels.

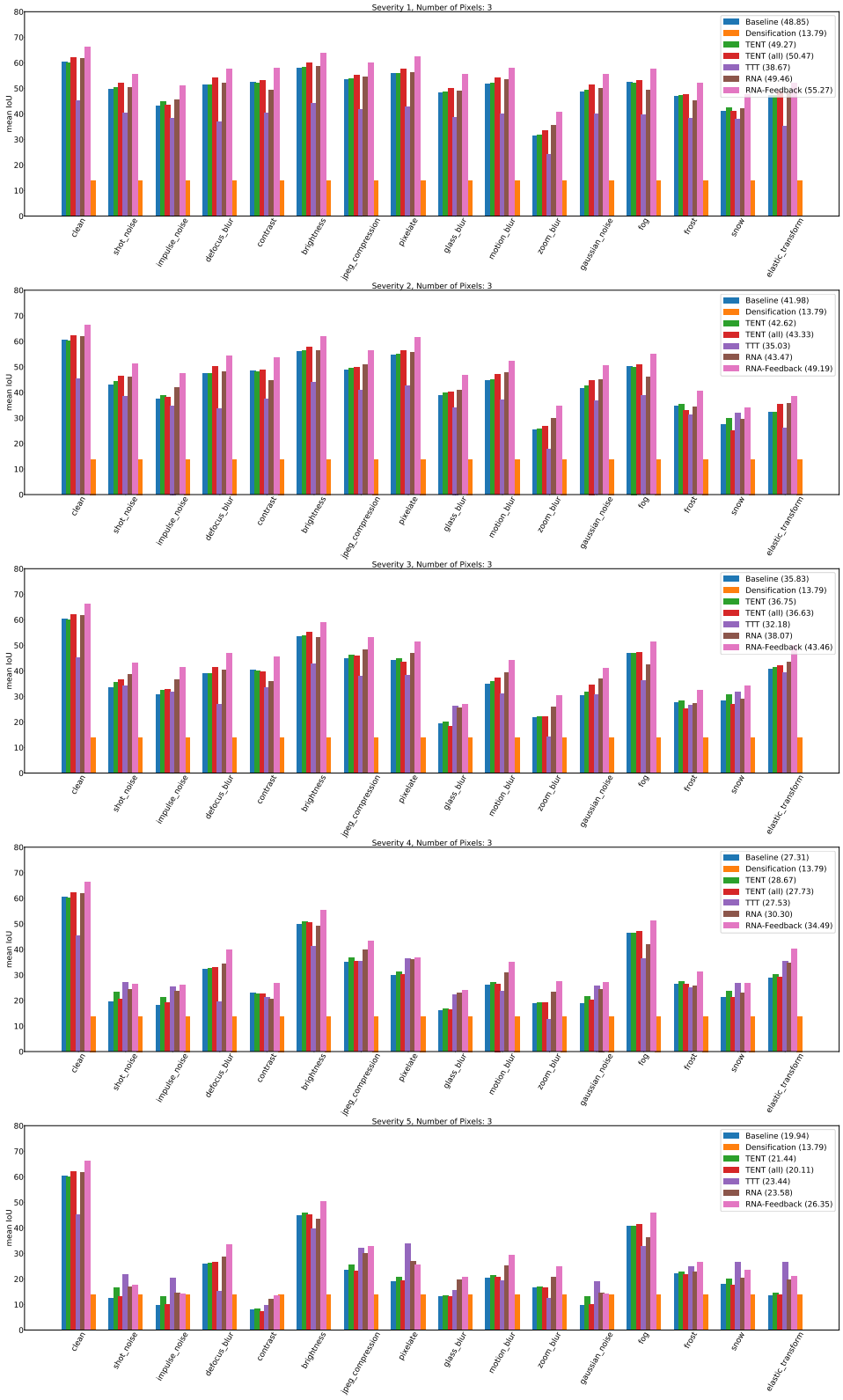

Figure 18: Supplementary results for semantic segmentation. Mean IOU vs individual corruptions for different severities when the number of pixels is 3. Numbers in the legend denote the average over the corruptions.

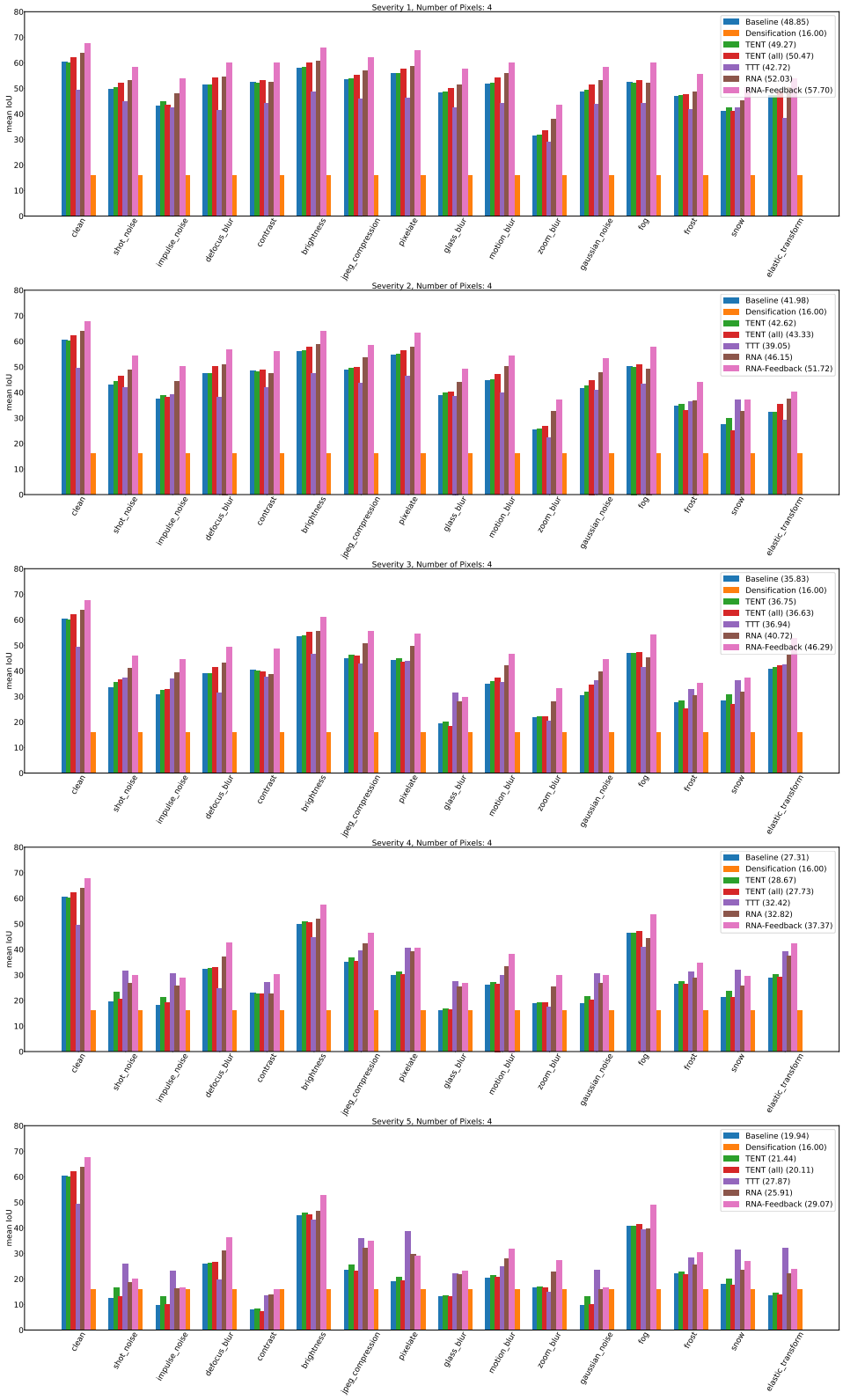

Figure 19: Supplementary results for semantic segmentation. Mean IOU vs individual corruptions for different severities when the number of pixels is 4. Numbers in the legend denote the average over the corruptions.

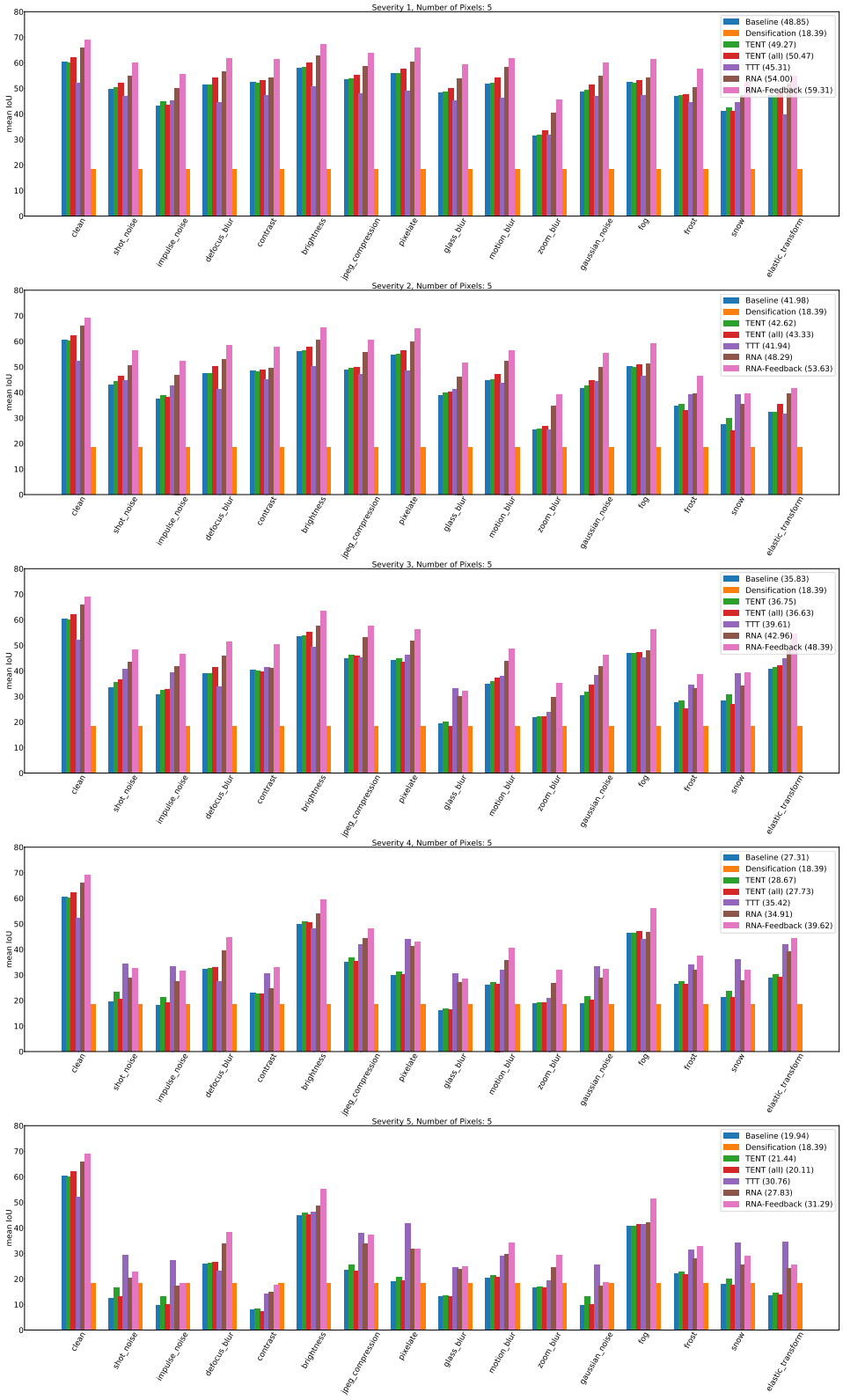

Figure 20: Supplementary results for semantic segmentation. Mean IOU vs individual corruptions for different severities when the number of pixels is 5. Numbers in the legend denote the average over the corruptions.

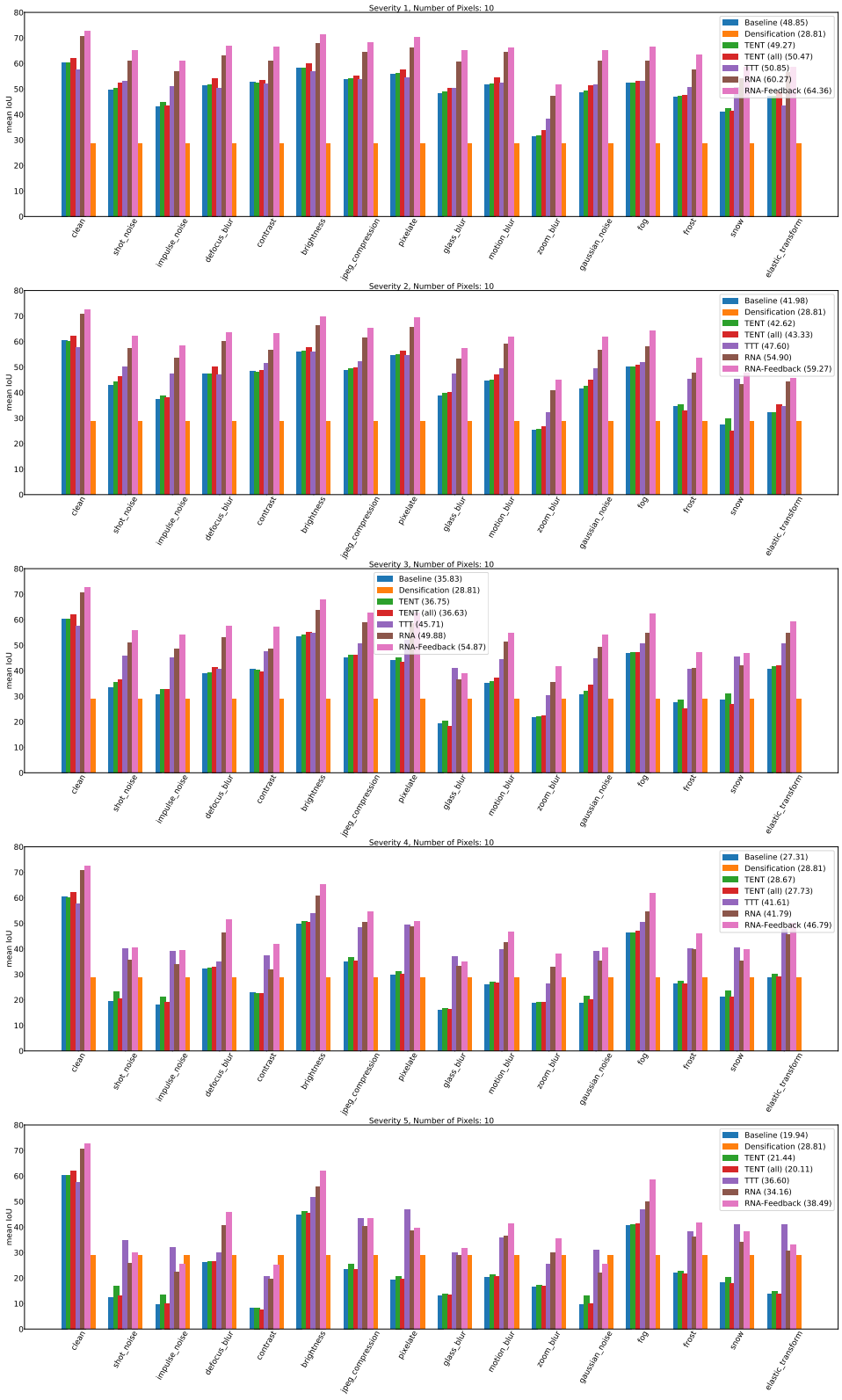

Figure 21: Supplementary results for semantic segmentation. Mean IOU vs individual corruptions for different severities when the number of pixels is 10. Numbers in the legend denote the average over the corruptions.

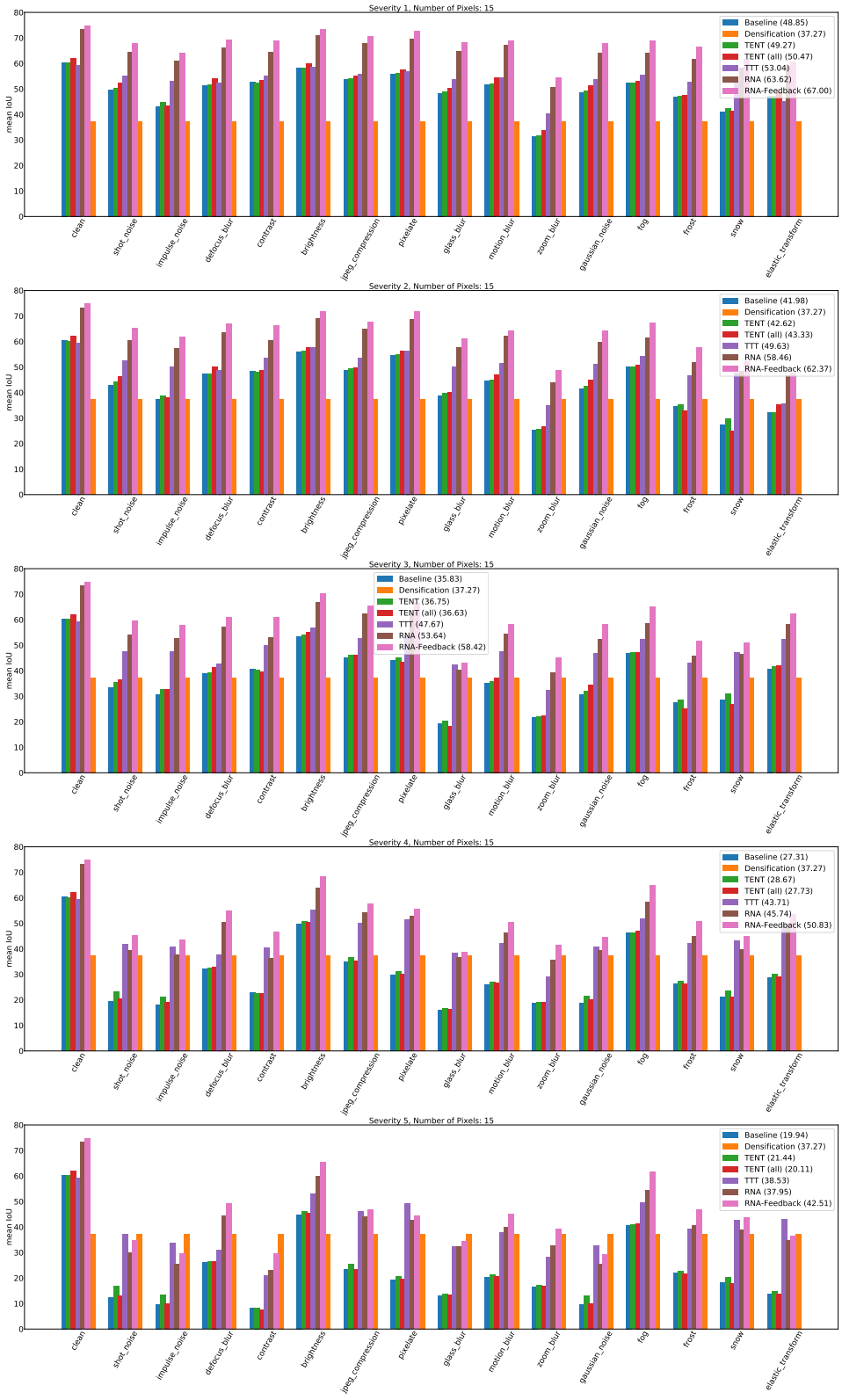

Figure 22: Supplementary results for semantic segmentation. Mean IOU vs individual corruptions for different severities when the number of pixels is 15. Numbers in the legend denote the average over the corruptions.

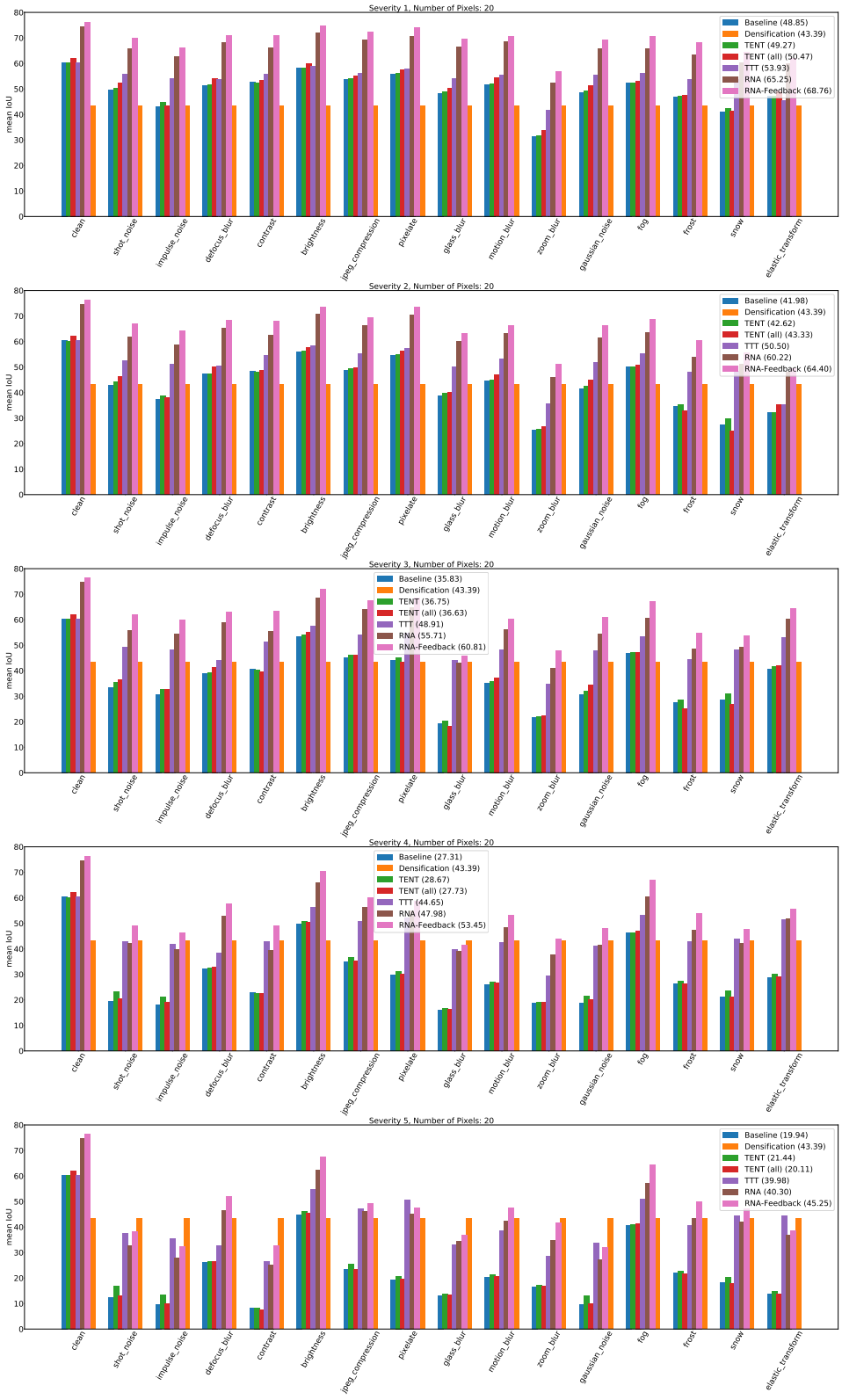

Figure 23: Supplementary results for semantic segmentation. Mean IOU vs individual corruptions for different severities when the number of pixels is 20. Numbers in the legend denote the average over the corruptions.

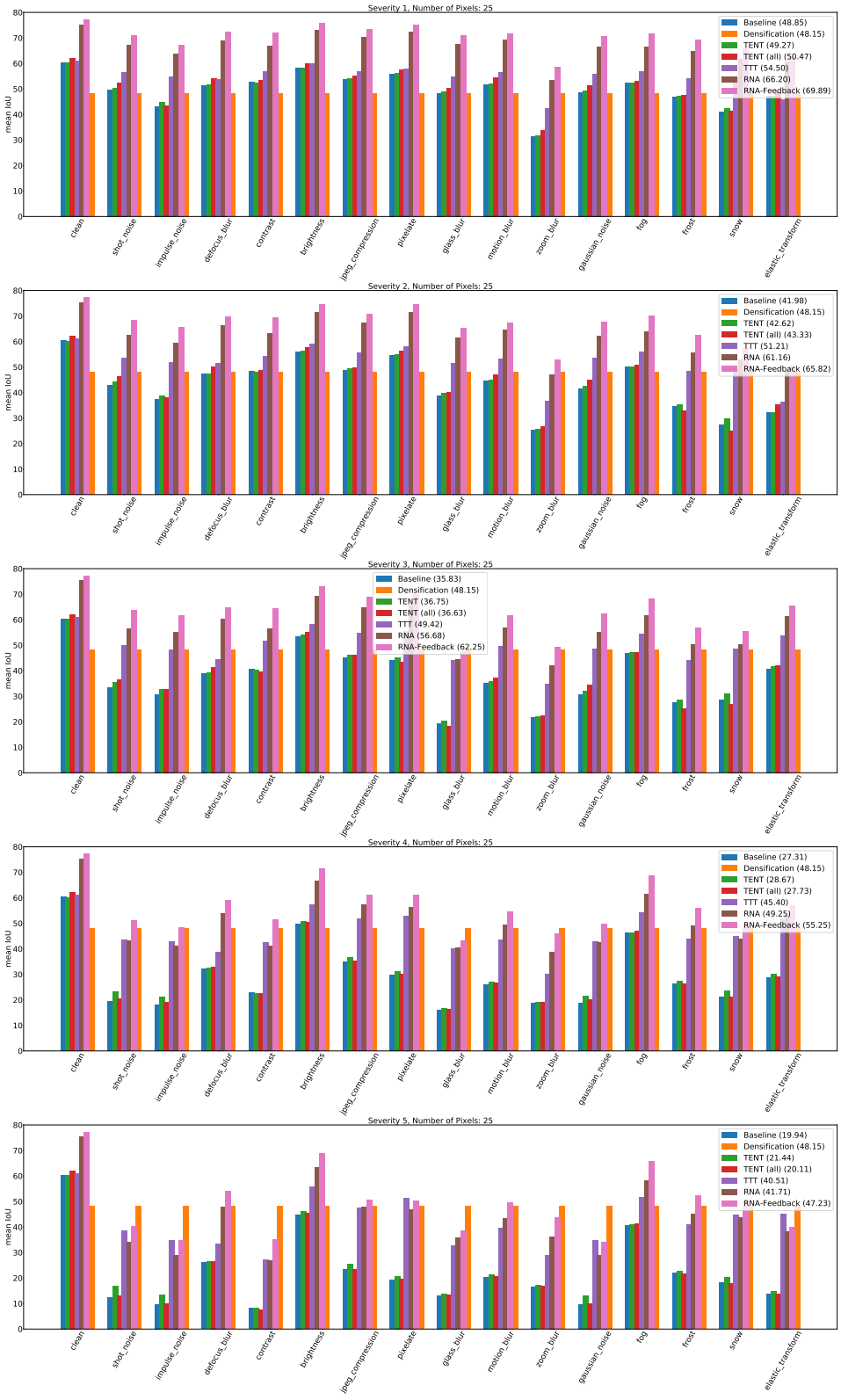

Figure 24: Supplementary results for semantic segmentation. Mean IOU vs individual corruptions for different severities when the number of pixels is 25. Numbers in the legend denote the average over the corruptions.

Figure 25: Supplementary results for semantic segmentation. Qualitative comparison of our method vs baselines for defocus blur and glass blur corruptions applied to COCO validation images.

Figure 26: Supplementary results for semantic segmentation. Qualitative comparison of our method vs baselines for JPEG compression and motion blur corruptions applied to COCO validation images.

Figure 27: Supplementary results for semantic segmentation. Qualitative comparison of our method vs baselines for shot noise corruption applied to COCO validation images.

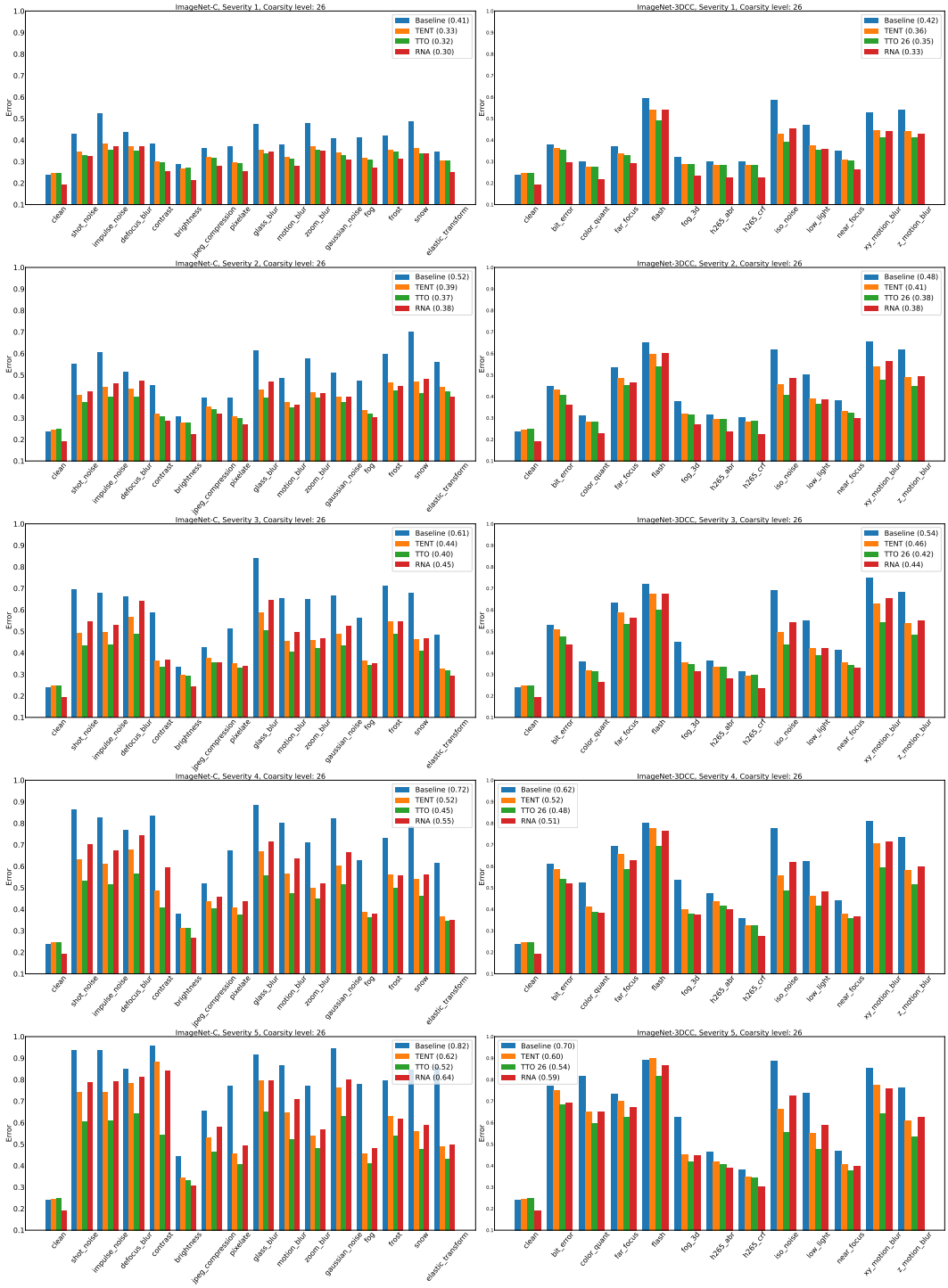

Figure 28: Supplementary results for ImageNet classification. Error for individual corruptions from ImageNet-C and ImageNet-3DCC. TTO and RNA uses 26-way coarse label supervision. Numbers in the legend denote the average over the corruptions.

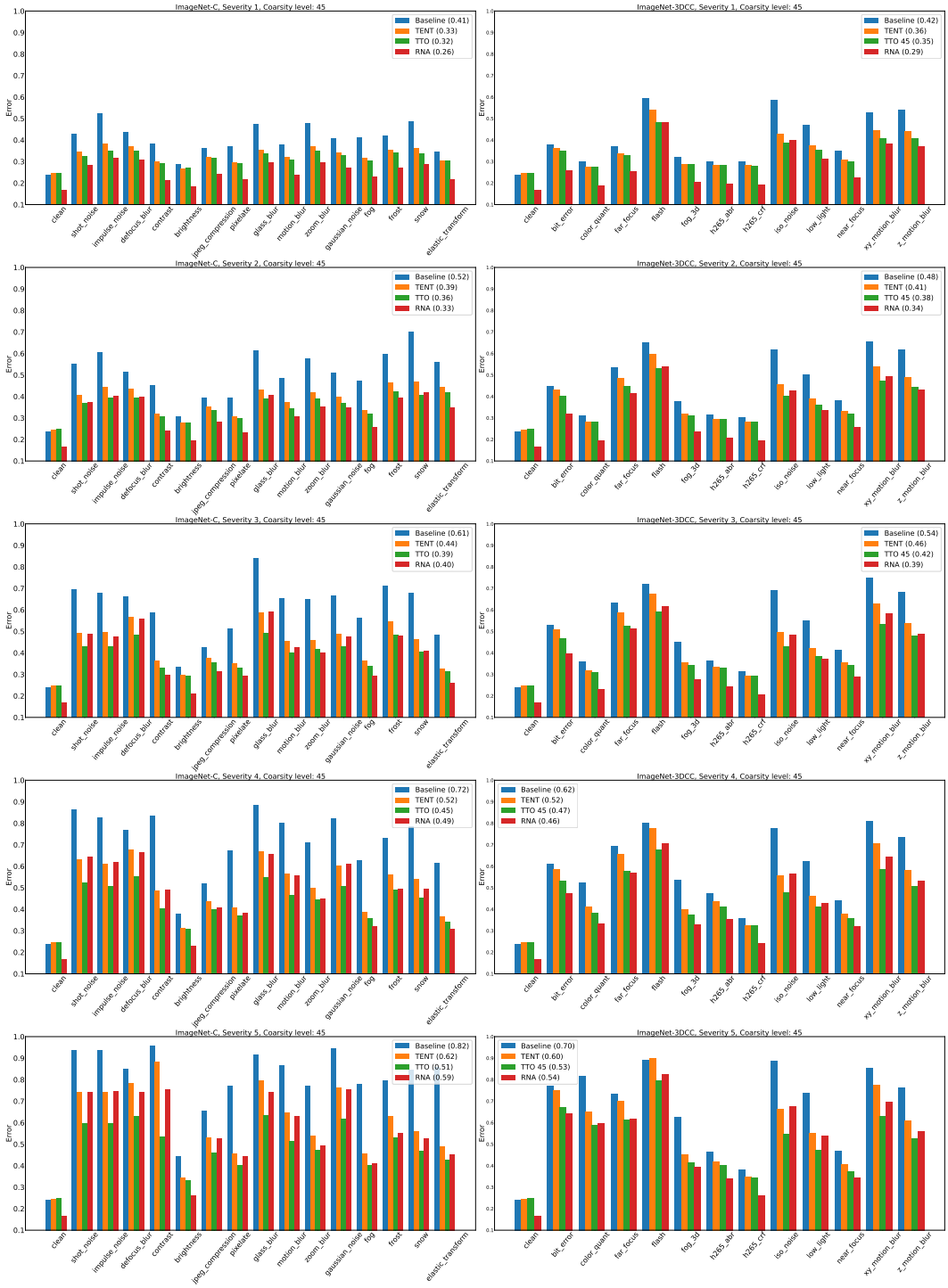

Figure 29: Supplementary results for ImageNet classification. Error for individual corruptions from ImageNet-C and ImageNet-3DCC. TTO and RNA uses 45-way coarse label supervision. Numbers in the legend denote the average over the corruptions.

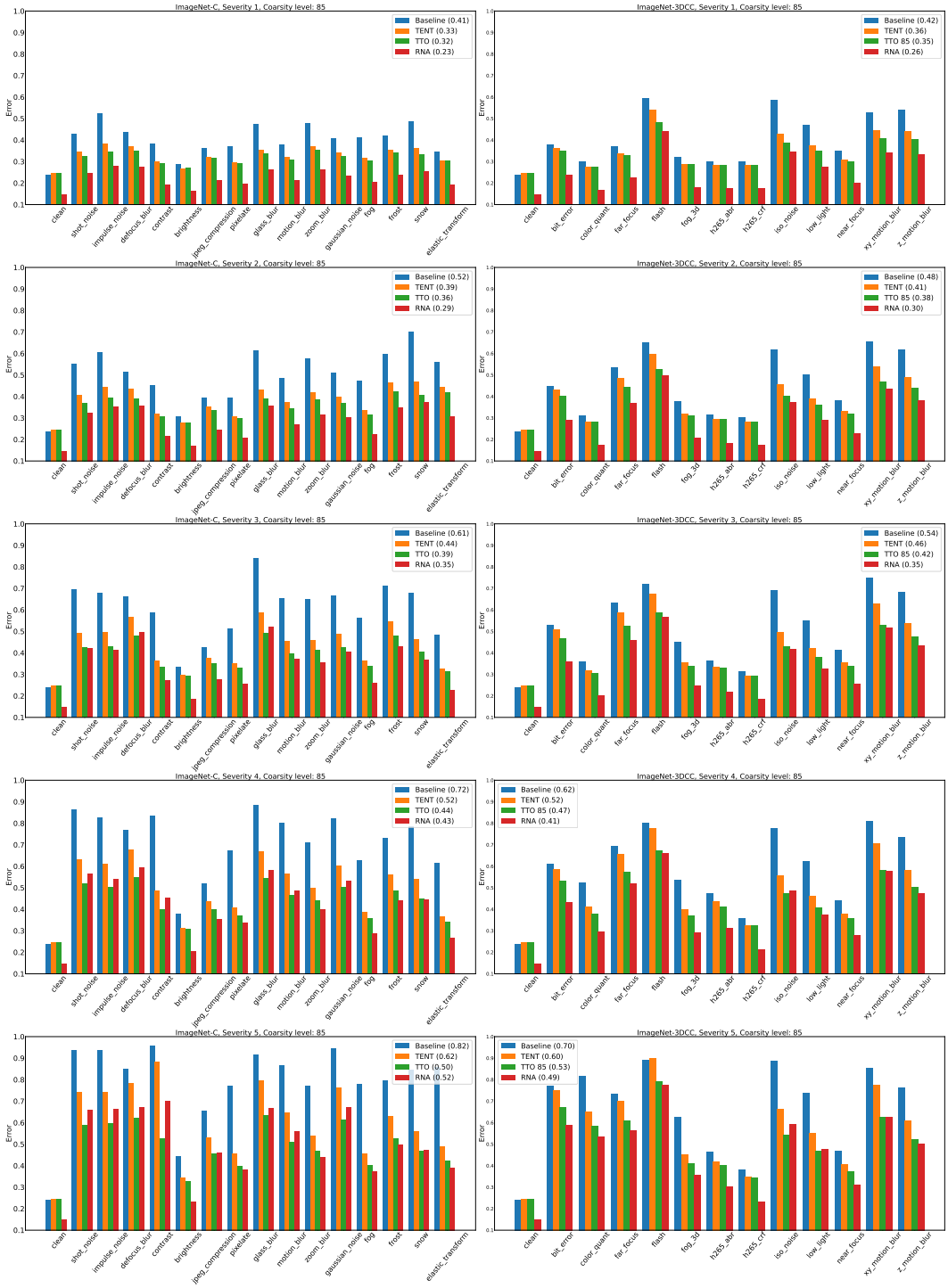

Figure 30: Supplementary results for ImageNet classification. Error for individual corruptions from ImageNet-C and ImageNet-3DCC. TTO and RNA uses 85-way coarse label supervision. Numbers in the legend denote the average over the corruptions.

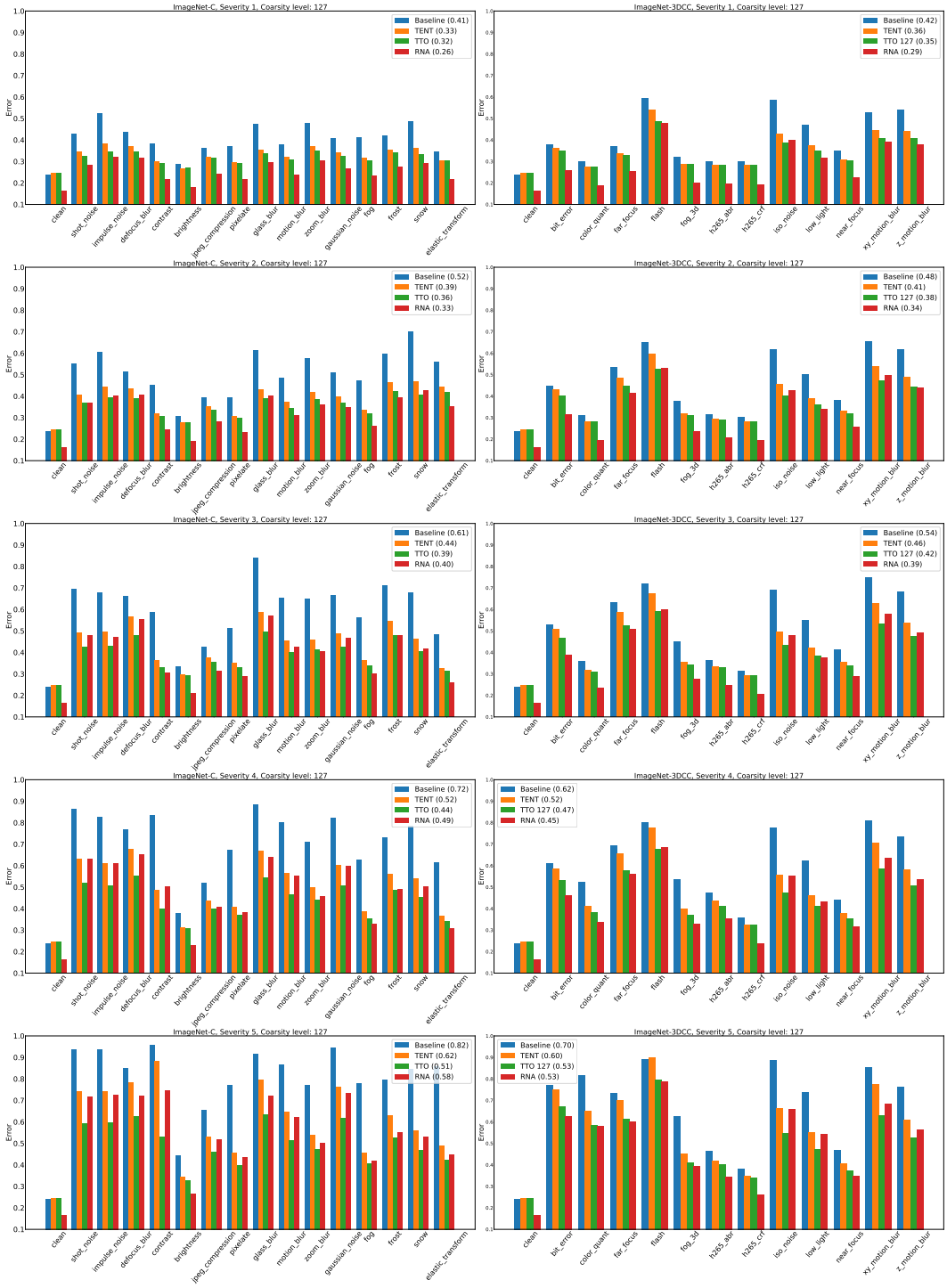

Figure 31: Supplementary results for ImageNet classification. Error for individual corruptions from ImageNet-C and ImageNet-3DCC. TTO and RNA uses 127-way coarse label supervision from Huh et al. (2016a). Numbers in the legend denote the average over the corruptions.