Test-Time Visual In-Context Tuning

Jiahao Xie[1,2], Alessio Tonioni[3], Nathalie Rauschmayr[3], Federico Tombari[3], Bernt Schiele[1,2]
[1]Max Planck Institute for Informatics, SIC [2]VIA Research Center [3]Google
{jxie, schiele}@mpi-inf.mpg.de {alessiot, rauschmayr, tombari}@google.com

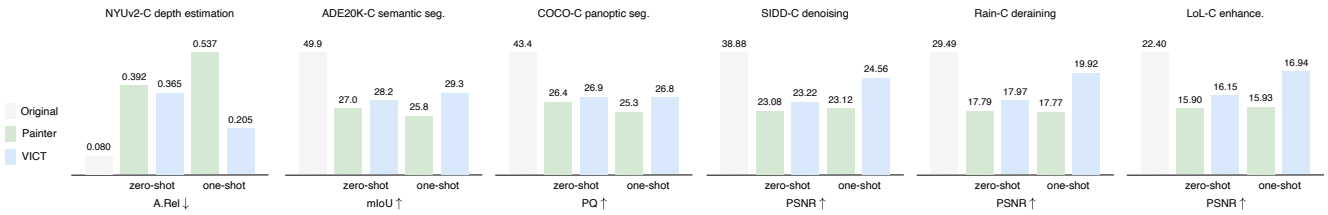

Figure 1. **Test-time visual in-context tuning (VICT) on six representative vision tasks under distribution shifts.** We benchmark the robustness of VICL with 15 common corruptions adopted in [23, 31], and report the averaged performance across all corruptions. Existing VICL models like Painter exhibit poor generalization capability to unseen new domains when the task prompts come from the training distribution (*i.e.*, *zero-shot*). Performances are even worse when given task prompts from the test distribution (*i.e.*, *one-shot*). By performing VICT at test time, we can significantly improve Painter in both zero-shot and one-shot manners.

Abstract

*Visual in-context learning (VICL), as a new paradigm in computer vision, allows the model to rapidly adapt to various tasks with only a handful of prompts and examples. While effective, the existing VICL paradigm exhibits poor generalizability under distribution shifts. In this work, we propose test-time **V**isual **I**n-**C**ontext **T**uning (**VICT**), a method that can adapt VICL models on the fly with a single test sample. Specifically, we flip the role between the task prompts and the test sample and use a cycle consistency loss to reconstruct the original task prompt output. Our key insight is that a model should be aware of a new test distribution if it can successfully recover the original task prompts. Extensive experiments on six representative vision tasks ranging from high-level visual understanding to low-level image processing, with 15 common corruptions, demonstrate that our VICT can improve the generalizability of VICL to unseen new domains. In addition, we show the potential of applying VICT for unseen tasks at test time. Code: https://github.com/Jiahao000/VICT.*

1. Introduction

Following the success of in-context learning (ICL) [2, 7, 21] in natural language processing (NLP), visual in-context learning (VICL) [5, 52] has shown promising performance in developing generalist models for vision tasks. Inspired by prompting in NLP that defines tasks using language sequences as a general interface, existing VICL works [5, 52] use images themselves as a natural interface for general-purpose visual perception. They formulate VICL as an image inpainting task, *i.e.*, given an input-output example describing a specific task (*i.e.*, the prompt) as well as an input image, they are assembled in a grid and the problem is casted as inpainting the missing part of the grid (*i.e.*, the prediction) consistently with the given task prompts. This enables adapting a pre-trained vision model to various downstream tasks with only a handful of prompts and examples.

By default, current VICL models are frozen during deployment. However, their performance might suffer in real-world deployment as the test distribution usually changes and deviates from the training one. This naturally raises a question: what is the generalizability of VICL models under distribution shifts? In this work, we investigate this aspect focusing on controllable shifts due to image corruptions. We observe that the existing VICL paradigm like Painter [52] exhibits poor generalizability to unseen new domains when the task prompt comes from the training distribution, as shown in Figure 1. Surprisingly, performances are even worse when given input-output task prompts belonging to the same distribution as the test input.

Driven by this observation, we propose to rely on test-time visual in-context tuning (VICT) to adapt VICL models on the fly to unseen distributions using only the given test sample. The motivation is that each test input offers a hint about the test distribution. Thus, we modify the VICL

model at test time to make full use of this hint by setting up a one-sample learning problem. Specifically, given input-output task prompts and the test input, we first use the VICL model to inpaint the output image. We then flip the role between the task prompts and the test sample, *i.e*., we treat the predicted test output as the prompt to the model and reconstruct the original output of task prompts. This allows us to tune the parameters of the whole model in a self-supervised manner that can be applied to arbitrary tasks. Our key insight is that a model should be aware of a new test distribution if it can successfully recover the original task prompts conditioned on its in-context inference. Such a cycle consistency supervision signal naturally exists in the context of VICL without requiring any additional training data or annotations, thus making it an appealing self-supervisory task for test-time visual in-context training.

We explore VICT in two settings: (i) zero-shot setting, where the task prompts are from the training distribution (*i.e*., clean images), and (ii) one-shot setting, where the task prompts are from the test distribution (*i.e*., corrupted images). Without loss of generality, we consider Painter [52] as our VICL model, for its simplicity in design and its wide applicability. As shown in Figure 1, our simple method leads to substantial improvements across 15 common corruptions [23, 31] on six representative vision tasks ranging from high-level visual understanding to low-level image processing, including depth estimation on NYUv2 [40], semantic segmentation on ADE20K [63], panoptic segmentation on COCO [27], image denoising on SIDD [1], image deraining on the merged deraining dataset [57], and low-light image enhancement on LoL [53].

Our main contributions are summarized as follows:

1) We propose a new cycle consistency task for test-time visual in-context tuning. To the best of our knowledge, we are the first to perform test-time training for VICL.

2) We contribute the first study on the generalizability of VICL under distribution shifts. We observe that the existing VICL paradigm exhibits poor generalizability to unseen new domains. Such a phenomenon can hardly be recovered even given input-output task prompts with the same distribution as the test input.

3) We conduct extensive experiments on six representative vision tasks across 15 common corruptions. VICT significantly improves Painter in both zero-shot and one-shot manners. Our zero-shot or one-shot VICT can even outperform Painter trained with more few-shot corrupted examples. We also explore the potential of applying VICT for unseen tasks at test time, further demonstrating its promise.

2. Related Work

In-context learning. *In-context learning* has been extensively explored in the NLP literature after the introduction of large language models. Seminal works in the field like GPT-3 [7] go as far as claiming that language models are indeed few-shot learners. All recent language models like [13, 46, 47] have the ability to adapt their behavior based on few-shot examples provided in their context window or directly to follow simple user instructions [10]. Similar concepts have been extended to multimodal models with seminal works like Flamingo [2] showing few-shot capabilities by providing examples as interleaved text and images in the context window. Some of these capabilities are now available for extremely large commercial multimodal large language models [35, 42, 43].

Apart from using languages as the general interface, a recent line of work introduces purely *visual in-context learning* (VICL) by training vision generalist models that can perform arbitrary visual tasks following one or few visual examples provided at inference time. Two representative works developed in parallel are: (i) MAE-VQGAN [5], which trains a variant of MAE [22] on a dataset of figures extracted from academic papers, and (ii) Painter [52], which uses a similar idea but trains its model on a set of standard academic benchmark datasets. More recently, this strategy has been expanded in LVM [3], where even more visual datasets are collected to train an autoregressive generative model. We base our work on Painter [52] due to its simplicity in design and its wide applicability. It is the one that provides both the available code and model weights, and has already been extensively evaluated across several standard benchmarks.

Generalization under distribution shifts. Machine learning models suffer from performance drops when tested on a data distribution different from the one they are trained on [17, 49, 50, 61]. The lower the drop, the more we define a model robust or able to generalize to distribution shifts. Common strategies to increase model robustness include using heavy data augmentations [11, 23, 44, 62] or extremely large training distributions going up to web scale [33, 37, 59]. Nevertheless, most of the experiments are done in a classification setting. To the best of our knowledge, there is no previous work exploring the robustness of VICL models. We are the first to counteract performance drops in VICL under distribution shifts.

Test-time training. An alternative way of counteracting performance drops due to distribution shifts is test-time training [6, 8, 16, 28, 41, 48, 51]. This accounts to unfreezing the model at test time and fine-tuning it on the target distribution (or a single sample from it) through self-supervision. Early works propose this paradigm using self-supervised pretext tasks (*e.g*., rotation prediction [18]) to explicitly improve generalization under image corruptions [41] or improve in specific tasks where self-supervised losses can be clearly defined like depth estimation [26, 36, 45], reinforcement learning [20, 32], tracking [15, 38], and NLP [4, 56]. More recently, the field

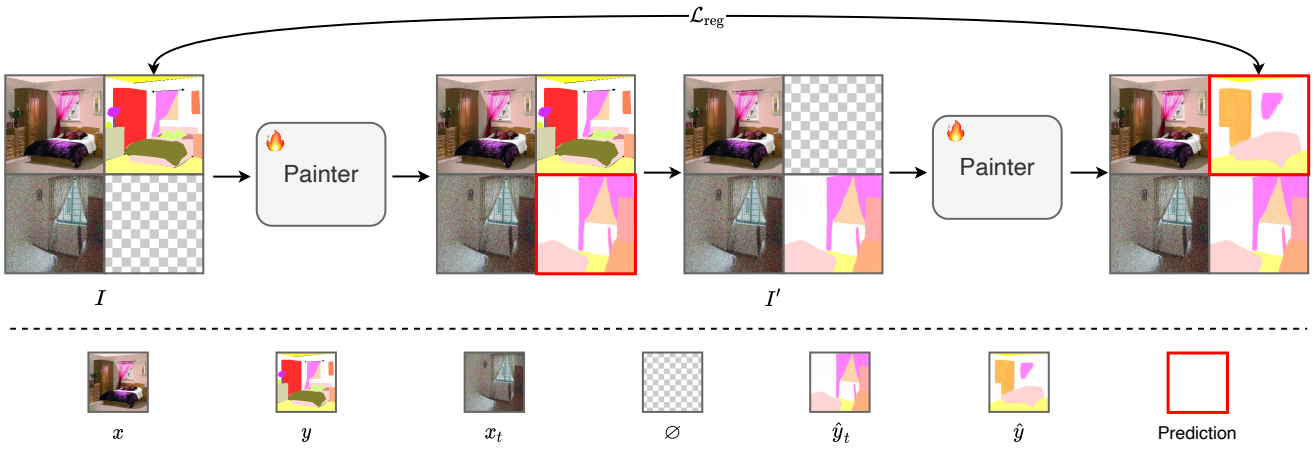

Figure 2. **Overview of our VICT pipeline.** Given a pair of task prompts (x, y) and a test input image x_t, we first construct a four-cell grid-like image canvas $I = (x, y, x_t, \varnothing)$, with an empty cell at the bottom right. We then feed I into the VICL model (*e.g.*, Painter) to predict the test output $\hat{y}_t$. Afterward, we flip the role between input-output task prompts and input-output test samples, *i.e.*, we provide the predicted $\hat{y}_t$ as the *prompt* to the model, recreating a new four-cell grid-like image canvas $I' = (x, \varnothing, x_t, \hat{y}_t)$, with an empty cell at the top right. The new I' is fed into the same model to predict the task prompt output $\hat{y}$. We finally optimize the model by minimizing the distance between $\hat{y}$ and y via a standard regression loss.

has realized that MAE [22] provides a very strong self-supervised pretext task and has therefore been employed in the context of test-time training [16]. Finally, some recent works [14, 24, 30, 39, 58] also employ the test-time training paradigm for vision-language models to improve their generalization. To the best of our knowledge, ours is the first work to explore test-time training applied to VICL models, where previous self-supervised pretext tasks [9, 12, 18, 22, 25, 34, 54, 55, 60] cannot be employed due to their constraints on the single-image context.

3. Methodology

Our visual in-context tuning (VICT) is a simple yet effective test-time training approach to adapt visual in-context learning (VICL) models on the fly. In this section, we first introduce some preliminaries on VICL in Section 3.1. We then detail our VICT pipeline in Section 3.2.

3.1. Visual In-Context Learning

In-context learning is a new paradigm originating from large language models such as GPT-3 [7] in NLP. Unlike traditional learning paradigms, in-context learning formulates different NLP tasks as text completion tasks and makes predictions conditioned on one or many support examples provided to the model in the context window. Extending this paradigm to tasks requiring a dense visual output in the form of images is nontrivial and has not been addressed in the literature for a long time. Recently, [5, 52] formulate VICL as an image inpainting task by combining images and labels into a grid-like new image and using masked image modeling for pre-training, granting the models with the in-context learning ability.

Formally, let $S = \{(x_i, y_i)\}_{i=1}^{N}$ denote the set of support input-output examples (*a.k.a.*, task prompts) where x is an image and y is its visual label (*e.g.*, a segmentation mask).[1] Given S and a new test image x_t as input, VICL can be formulated as follows:

$$y_t = f_\theta(S, x_t), \tag{1}$$

where $f_\theta\,(*)$ is the VICL model parametrized with θ, y_t is the corresponding output label of x_t. Usually, we assume S and x_t are drawn from the same distribution. However, this is rarely the case for real-world deployment. Therefore, in this work, we mainly consider the scenario where a distribution shift like an image corruption occurs for x_t.

In the experiment section, we will show how the presence of such distribution shifts significantly degrades the performance of the VICL models. Moreover, we will additionally show how simple solutions like providing in-context examples from the target (corrupted) distribution would not fix the problem due to the limited generalization ability of the current VICL models.

3.2. Test-Time Visual In-Context Tuning

In this work, we propose to leverage test-time training [41] to counteract the distribution shifts between training and testing data, with the goal of making VICL models more robust. We argue that VICL models are particularly amenable to be optimized at test time on arbitrary tasks thanks to the availability of the N shots provided as context. The core intuition of our method is to use the provided in-context examples to define a self-supervised loss that can be used at

[1] Following previous works, we set $N = 1$ in practice.

Algorithm 1 Pseudocode of VICT in a PyTorch-like style.

```
# f: VICL model (e.g., Painter)
# theta_0: pre-trained model weights
# x: task prompt input
# y: task prompt output
# mask_token: mask token for the empty cell
# steps: number of test-time optimization steps

for x_t in loader: # load a test sample x_t
   f.params = theta_0 # initialize
   # test-time optimization
   for t in range(steps):
      # predict the test sample output
      y_t_pred = f((x, y, x_t, mask_token))
      # predict the task prompt output
      y_pred = f((x, mask_token, x_t, y_t_pred))

      # regression loss
      loss = SmoothL1Loss(y_pred, y)

      # update model
      loss.backward()
      update(f.params)

   # inference with the updated model
   y_t_pred = f((x, y, x_t, mask_token))
```

test time to train any VICL models on the fly on any task. The overall pipeline of our VICT is illustrated in Figure 2.

Specifically, given a pair of task prompts (x, y) from S and a test input x_t, we construct a grid-like image canvas with four cells, denoted by an ordered quadruple[2] $I = (x, y, x_t, \varnothing)$, where $\varnothing$ is an empty cell. We then feed I into the VICL model f_θ to predict the test output $\hat{y}_t$:

$$\hat{y}_t = f_\theta(I) = f_\theta((x, y, x_t, \varnothing)). \tag{2}$$

Afterward, with the obtained test output, we flip the role between input-output task prompts and input-output test samples, creating a new four-cell grid-like image canvas with the order of $I' = (x, \varnothing, x_t, \hat{y}_t)$. The new I' is fed into f_θ again to predict the task prompt output $\hat{y}$:

$$\hat{y} = f_\theta(I') = f_\theta((x, \varnothing, x_t, \hat{y}_t)). \tag{3}$$

We then use a simple regression loss in pixel space to optimize the model by minimizing the distance between $\hat{y}$ and y:

$$\theta_{x_t} = \arg\min_\theta \mathcal{L}(\hat{y}, y) \tag{4}$$

In practice, we follow [52] to use smooth-ℓ_1 [19] loss. Using this loss we can optimize the parameters θ of the model for a small numbers of steps for each x_t and, eventually, make a prediction on x_t as $f_{\theta_{x_t}}((x, y, x_t, \varnothing))$.

In our formulation, the gradient-based optimization for Equation 4 always starts from the initial pre-trained model weights θ_0 for each test input. When a new test input arrives, we discard θ_{x_t} and reset the weights to θ_0. We follow this strategy to not have any assumption on the test inputs and treat them independently at test time, *i.e.*, we do not assume that they come from the same distribution. Our generic formulation is completely agnostic on the VICL models considered, on the type of tasks being solved, and on the type of in-context samples provided to it.

The pseudo-code of VICT is in Algorithm 1.

[2]The image order is top left, top right, bottom left, and bottom right, respectively.

4. Experiments

4.1. Implementation Details

Tasks and datasets. Following [52], we evaluate the model performance on six representative vision tasks ranging from high-level visual understanding to low-level image processing, which includes depth estimation on NYUv2 [40], semantic segmentation on ADE20K [63], panoptic segmentation on COCO [27], image denoising on SIDD [1], image draining on the merged draining datasets [57], and low-light image enhancement on LoL [53]. To evaluate the robustness to distribution shifts of the model on these datasets, we follow the setup in [23, 31] to corrupt the aforementioned datasets and simulate hard distribution shifts. The corrupted datasets contain 15 types of corruptions, covering noise (gaussian noise, impulse noise, shot noise), blur (defocus blur, glass blur, motion blur, zoom blur), weather (fog, frost, snow), and digital (brightness, contrast, elastic transform, jpeg compression, pixelate) categories. Each type of corruption has five levels of severity. We denote the corrupted datasets as NYUv2-C, ADE20K-C, COCO-C, SIDD-C, Rain-C, and LoL-C, respectively. Due to the space constraints, the results in the main text are limited to the most severe level (*i.e.*, level 5) that corresponds to the strongest distribution shift. We provide the results on the other four levels in the supplementary material.

Baselines. Without loss of generality, we consider Painter [52] as our VICL model for its simplicity in design and its wide applicability. We study VICT in two settings: (i) zero-shot setting, where the task prompts are from the training distribution (*i.e.*, clean images), and (ii) one-shot setting, where the task prompts are from the test distribution (*i.e.*, corrupted images). For the one-shot setting, apart from comparing with the frozen Painter using one-shot corrupted examples as task prompts, we also consider a baseline that further trains Painter with the one-shot corrupted samples using the same pre-training objective as in [52]. We also consider training Painter with more few-shot examples using the same settings to examine to what extent VICT can outperform a larger number of few-shot fine-tuning of Painter.

Training details. In all experiments, we use the pre-trained Painter based on ViT-Large provided by the authors of [52]. We perform VICT using an AdamW [29] optimizer, with betas as (0.9, 0.999), a weight decay of 0, a batch size of 1,

Table 1. **System-level comparison on six representative vision tasks across 15 corruptions.** Results are on corruption level 5. We consider two settings: (i) zero-shot setting, where the task prompts are from the training distribution (*i.e.*, clean images), and (ii) one-shot setting, where the task prompts are from the test distribution (*i.e.*, corrupted images). "avg" denotes the averaged results over 15 corruptions.

method	brigh	cont	defoc	elast	fog	frost	gauss	glass	impul	jpeg	motn	pixel	shot	snow	zoom	avg
(a) depth estimation NYUv2-C (A.Rel ↓)																
zero-shot setting:																
Painter	0.129	0.215	0.712	0.109	0.129	0.536	0.200	0.612	0.189	0.386	0.167	0.187	0.142	1.951	0.209	0.392
VICT	0.108	0.216	0.631	0.108	0.133	0.576	0.191	0.541	0.181	0.310	0.139	0.134	0.140	1.856	0.210	0.365
one-shot setting:																
Painter	0.126	0.285	0.743	0.109	0.132	0.853	0.901	0.622	0.901	0.392	0.174	0.194	0.440	1.964	0.212	0.537
VICT	0.097	0.193	0.180	0.107	0.128	0.227	0.241	0.210	0.278	0.159	0.121	0.114	0.150	0.662	0.214	0.205
(b) semantic segmentation ADE20K-C (mIoU ↑)																
zero-shot setting:																
Painter	40.4	11.7	27.9	31.2	35.0	23.3	24.8	25.6	26.1	38.6	31.8	40.4	25.9	9.4	13.1	27.0
VICT	40.9	12.7	28.2	31.7	36.4	23.7	25.5	26.3	27.4	38.8	31.9	40.9	28.0	17.3	13.6	28.2
one-shot setting:																
Painter	40.9	11.7	27.5	31.3	35.0	22.1	19.7	25.8	20.3	38.5	31.3	40.3	23.2	6.5	13.3	25.8
VICT	41.7	21.0	28.8	32.2	37.0	24.7	24.4	27.0	25.2	39.7	32.8	41.5	26.6	21.7	14.9	29.3
(c) panoptic segmentation COCO-C (PQ ↑)																
zero-shot setting:																
Painter	38.1	15.5	25.5	30.0	33.3	26.0	24.9	23.8	25.5	31.9	27.6	34.9	26.3	19.8	12.5	26.4
VICT	38.7	15.7	25.8	30.3	33.9	26.7	25.3	24.3	25.9	32.4	27.7	35.4	26.7	21.9	12.7	26.9
one-shot setting:																
Painter	38.1	14.2	24.8	30.0	33.1	25.2	21.5	23.7	22.0	31.5	27.5	34.6	23.8	17.2	12.1	25.3
VICT	38.6	16.9	25.4	30.4	33.8	26.6	24.0	24.6	24.5	32.3	27.4	35.0	25.2	24.9	12.4	26.8
(d) denoising SIDD-C (PSNR ↑)																
zero-shot setting:																
Painter	8.72	25.49	32.48	33.20	16.86	8.79	17.50	33.18	17.50	23.18	29.47	34.84	25.27	9.86	29.80	23.08
VICT	8.86	25.65	32.91	32.96	17.00	8.84	17.58	33.00	17.50	24.86	29.68	34.76	25.21	9.75	29.67	23.22
one-shot setting:																
Painter	9.12	25.16	31.99	33.23	16.68	9.24	17.99	33.14	18.33	22.73	29.30	34.87	24.69	10.50	29.76	23.12
VICT	13.94	25.32	32.41	34.45	16.45	11.15	20.13	32.96	19.93	26.80	30.11	35.01	24.93	14.90	29.95	24.56
(e) deraining Rain-C (PSNR ↑)																
zero-shot setting:																
Painter	10.74	14.15	19.54	20.10	13.91	12.49	19.85	19.80	20.16	24.86	18.52	22.81	20.11	11.89	17.93	17.79
VICT	10.89	14.24	19.58	20.31	13.94	12.62	20.42	19.83	20.67	24.97	18.56	23.05	20.47	12.05	17.98	17.97
one-shot setting:																
Painter	11.04	13.95	19.61	20.15	13.82	12.87	19.51	19.87	19.58	24.85	18.62	22.88	19.68	12.16	17.95	17.77
VICT	17.38	14.69	20.36	21.44	14.67	17.69	22.33	20.48	22.24	25.31	19.05	23.92	21.37	19.53	18.28	19.92
(f) low-light enhancement LoL-C (PSNR ↑)																
zero-shot setting:																
Painter	16.26	13.43	18.80	19.83	11.24	11.62	12.50	18.82	12.91	15.87	18.01	21.12	17.75	13.77	16.60	15.90
VICT	16.39	13.57	19.24	19.84	11.37	11.64	12.51	19.23	13.00	16.00	18.11	21.20	19.16	13.92	17.06	16.15
one-shot setting:																
Painter	16.74	13.50	18.85	19.83	11.02	11.76	13.14	18.95	13.74	15.84	18.15	21.12	16.54	13.19	16.59	15.93
VICT	18.19	15.03	18.99	19.88	11.76	12.01	14.95	19.22	14.99	16.12	18.23	21.09	19.84	16.96	16.91	16.94

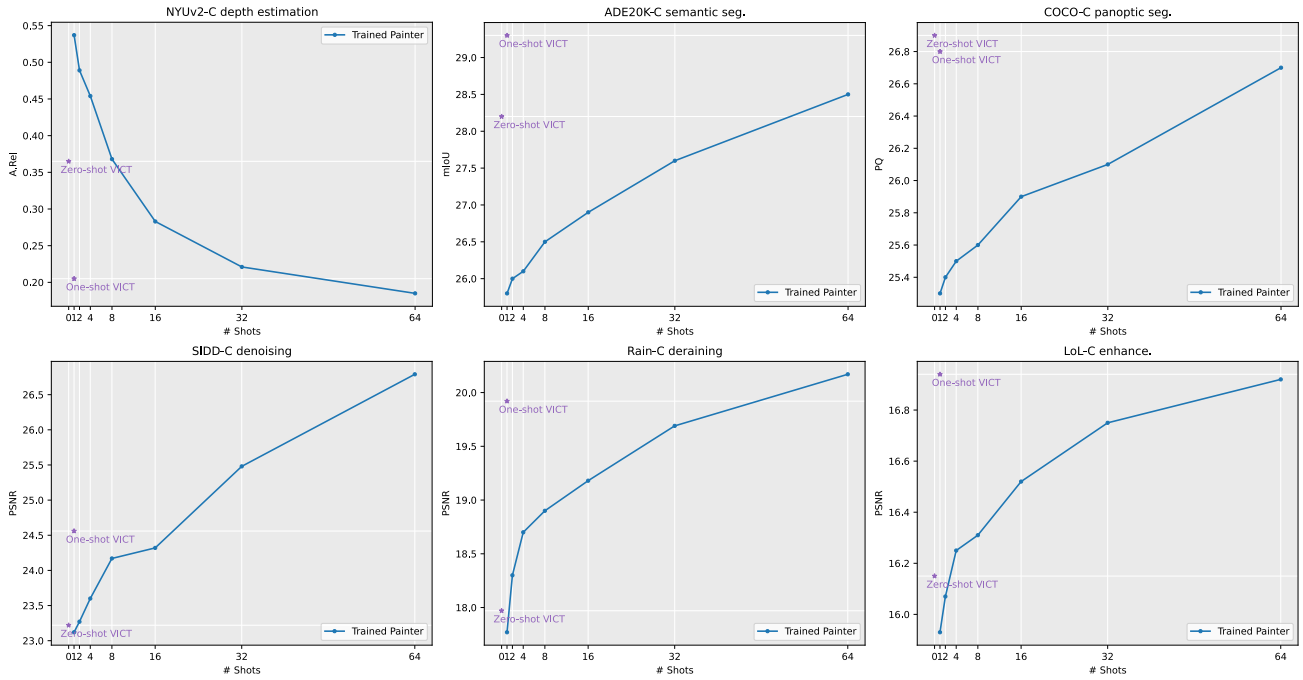

Figure 3. **Comparison with few-shot Painter on six vision tasks with corruptions.** We randomly corrupt a certain number of images in the training set, using 1, 2, 4, 8, 16, 32, and 64 shots for training and deploying the model in the full corrupted test sets. We report the final results averaged across 15 corruptions. Our zero-shot or one-shot VICT can outperform Painter trained with more few-shot examples.

and a fixed learning rate of 1e-6. Unless otherwise specified, we train each test sample for 60 steps. The choice of 60 steps is purely computational and more steps are likely to further improve performance, judging from the positive trend observed in Figure 4. Note that we do not apply any data augmentations for VICT as many data augmentations are in fact distribution shifts in our evaluation benchmarks. Training with them is analogous to training on the test distributions. Therefore, to solely study the generalization ability to new test distributions, we purposely choose not to use any data augmentations, even though they could improve our results at face value. Every iteration of VICT is performed on the same concatenated grid-like image, with each sub-image resized as 448×448, which is the same as what we later use for in-context inference. During VICT, we only optimize the encoder weights. We have experimented with training both the encoder and the decoder and found that the difference is negligible, which is also consistent with the observations in [16, 41]. A detailed comparison is provided in the experiment section.

4.2. Main Results

System-level comparison. We compare VICT with Painter under 15 common corruptions as introduced in Section 4.1. We perform experiments on six representative vision tasks covering high-level visual understanding and low-level image processing, which includes depth estimation on NYUv2-C, semantic segmentation on ADE20K-C, panoptic segmentation on COCO-C, image denoising on SIDD-C, image draining on Rain-C, and low-light image enhancement on LoL-C.

Table 1 reports the results. First of all, we observe that Painter exhibits poor generalization ability under common corruptions. For example, on ADE20K-C semantic segmentation, Painter only achieves an average performance of 27.0 mIoU, which is 22.9 mIoU lower than that on the clean validation set (49.9 mIoU as reported in Figure 1). Nevertheless, our VICT outperforms Painter by clear margins in both zero-shot and one-shot settings across different tasks. When it comes to the zero-shot setting, VICT achieves an average of -0.027 A.Rel on NYUv2-C depth estimation, +1.2 mIoU on ADE20K-C semantic segmentation, +0.5 PQ on COCO-C panoptic segmentation, +0.14 PSNR on SIDD-C image denosing, +0.18 PSNR on Rain-C image draining, and +0.25 PSNR on LoL-C low-light enhancement, respectively.

The performance gains of VICT over Painter are further increased when we shift from the zero-shot setting to the one-shot counterpart. Specifically, VICT achieves an average of -0.332 A.Rel on NYUv2-C depth estimation, +3.5 mIoU on ADE20K-C semantic segmentation, +1.5 PQ on COCO-C panoptic segmentation, +1.44 PSNR on SIDD-C image denoising, +2.15 PSNR on Rain-C image draining, and +1.01 PSNR on LoL-C low-light enhancement, respectively. Besides, for the Painter baseline, using the

Table 2. **Test-time optimization on different modules.** We use semantic segmentation on ADE20K-C for the ablation. Training only the encoder performs similarly to training both the encoder and decoder, regardless of the zero-shot and one-shot settings.

module	brigh	cont	defoc	elast	fog	frost	gauss	glass	impul	jpeg	motn	pixel	shot	snow	zoom	avg
zero-shot setting:																
Encoder	40.9	12.7	28.2	31.7	36.4	23.7	25.5	26.3	27.4	38.8	31.9	40.9	28.0	17.3	13.6	28.2
Both	40.9	12.5	28.2	31.7	36.4	23.8	25.7	26.5	27.4	38.8	31.9	41.0	27.9	16.9	13.5	28.2
one-shot setting:																
Encoder	41.7	21.0	28.8	32.2	37.0	24.7	24.4	27.0	25.2	39.7	32.8	41.5	26.6	21.7	14.9	29.3
Both	41.7	21.0	28.7	32.1	36.9	24.4	24.4	27.0	25.3	39.6	33.0	41.6	26.8	21.5	15.1	29.3

one-shot setting does not always perform better than the zero-shot setting. The average performances are even degraded in most cases, *e.g.*, depth estimation on NYUv2-C (0.392 A.Rel *vs*. 0.537 A.Rel), semantic segmentation on ADE20K-C (27.0 mIoU *vs*. 25.8 mIoU), panoptic segmentation on COCO-C (26.4 PQ *vs*. 25.3 PQ), image draining on Rain-C (17.79 PSNR *vs*. 17.77 PSNR). In contrast, for VICT, using the one-shot setting usually performs better than the zero-shot setting. It is worth mentioning that our zero-shot VICT can even outperform the one-shot Painter, further demonstrating the effectiveness of VICT.

Comparison with few-shot Painter. In previous experiments, we mainly compare with a frozen Painter using either clean or corrupted examples as the task prompts. Here, we further consider a scenario where more few-shot examples from the test distribution are available. Specifically, we randomly corrupt a certain number of images in the training set, using 1, 2, 4, 8, 16, 32, and 64 shots to train Painter respectively and deploying the model in the full corrupted test sets.

The results are shown in Figure 3. Both our zero-shot VICT and one-shot VICT can consistently outperform the one-shot Painter. When more few-shot examples are available for training, our VICT can still outperform Painter below a certain threshold. We observe such a threshold varies for different tasks, with high-level visual understanding tasks having a higher threshold. It is worth noting that in high-level visual understanding tasks like ADE20K-C semantic segmentation and COCO-C panoptic segmentation, both our zero-shot and one-shot variants of VICT nearly match or even outperform the 64-shot Painter. In other words, with only one or simply no labeled test sample(s), VICT can achieve similar performances as models that use significantly more labeled test samples to train. This phenomenon is rather appealing as in practice it is rare to know test distribution in advance, let alone obtaining a large number of labeled test samples.

4.3. Further Analysis

Effect of test-time optimization modules. We study the effect of optimizing different modules at test time. We consider two parameter groups: (i) the encoder, and (ii) both the encoder and decoder. We use semantic segmentation on ADE20K-C as an example task for analysis. The results are shown in Table 2. Training only the encoder performs similarly to training the entire model, regardless of the zero-shot and one-shot settings. This makes sense since the encoder acts as a feature extractor, which plays a key role in determining the representation quality. In contrast, the decoder is only responsible for mapping the latent representation back to its original resolution, which is more specialized for reconstruction but less relevant for semantics. Thus, we choose only to optimize the encoder at test time.

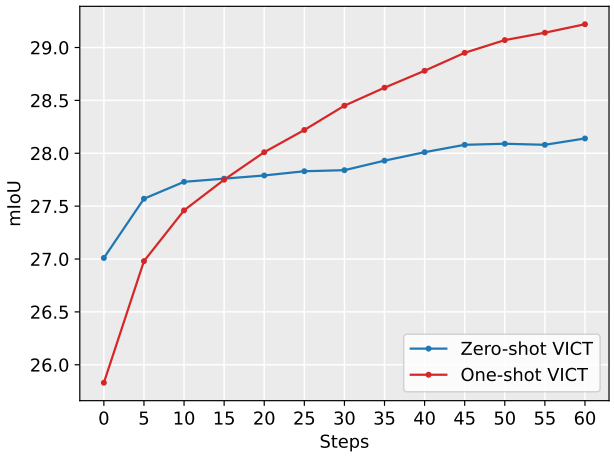

Figure 4. **Analysis on the trade-off between efficiency and accuracy.** We use semantic segmentation on ADE20K-C for the ablation. VICT benefits from more training steps, while at the cost of linearly increased training time.

Effect of test-time optimization steps. We study the effect of test-time optimization steps in Figure 4. VICT benefits from more training steps, which keeps improving performance even after 60 steps. However, it should be noted that the runtime of our VICT is linearly increased with the number of training steps. For reference, it takes around 0.4 seconds per step per test sample on a single A100 GPU. Thus, in practice, one may decide the number of training steps according to the pre-defined cost budget for deployment.

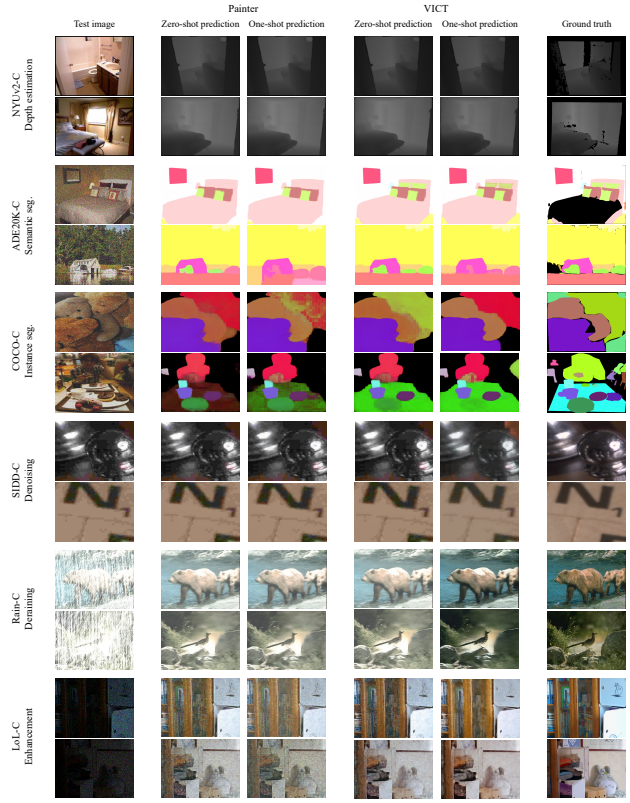

Figure 5. **Visualizations of test examples and predictions for six main tasks with corruptions.** We visualize both zero-shot and one-shot settings for Painter and VICT. Zoom in for best view.

4.4. Qualitative Results

Results on main tasks. We visualize some test examples and predictions from the validation set for different tasks with corruptions, including depth estimation, semantic segmentation, instance segmentation, image denoising, image deraining, and low-light image enhancement. As shown in Figure 5, VICT can make more accurate predictions than Painter on all tasks.

Results on unseen tasks. In this work, we mainly consider the distribution shifts with corruptions and conduct experiments on the tasks that are seen in training. Here, we further verify whether VICT can generalize to unseen tasks. We explore this capability via visualizations. More quantitative results are provided in the supplementary material. Figure 6 provides examples of two unseen tasks including foreground object segmentation and colorization. Both Painter and our VICT can generalize to the foreground segmentation task. This makes sense as similar segmentation tasks (*e.g.*, semantic segmentation and instance segmentation) have been seen during the training stage of Painter. Nevertheless, our VICT further reduces noises and produces finer masks. However, Painter cannot generalize to the colorization task that is totally unseen during training, producing only grayscale images or even wrong predictions from other tasks (*e.g.*, the last row). In contrast, our VICT can produce decent colorful results. This further demonstrates the potential of applying VICT for unseen tasks at test time.

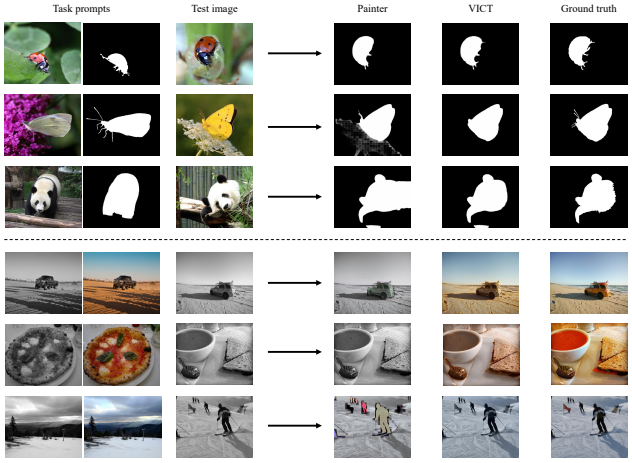

Figure 6. **Visualizations of test examples and predictions for unseen tasks.** The visualized tasks include foreground object segmentation and colorization. Painter cannot generalize to totally unseen tasks like colorization, whereas VICT can make decent color predictions. Zoom in for best view.

5. Conclusion

Generalization under distribution shifts is the central theme of deep learning. In this work, we have studied the robustness of VICL models and found that the existing VICL paradigm exhibits poor generalization capability to unseen new domains. Based on this observation, we proposed a simple yet effective test-time visual in-context tuning method to adapt VICL models to a single test sample on the fly. We hope our explorations can pave the way for improving the generalizability of VICL.

Limitations and future work. Our study has several limitations: 1) Since we perform training for each test sample, our method is slower than the baseline applying a fixed model at test time, which is a common limitation for test-time training. Inference speed might be improved through better architectural designs, training techniques, optimizers, and hyper-parameters. However, it has not been the focus of this paper. 2) Our proposed cycle consistency supervision is a general self-supervised task. However, we cannot guarantee that it will produce useful gradients for every single test distribution. We only focus on several representation vision tasks and most popular corruption benchmarks for distribution shifts. More tasks and benchmarks can be studied for future research. One possible extension of this work is to apply similar test-time supervision in multi-modal in-context learning, where cycle consistency supervision also exists. We leave this exploration for future work.

References

[1] Abdelrahman Abdelhamed, Stephen Lin, and Michael S Brown. A high-quality denoising dataset for smartphone cameras. In *CVPR*, 2018. 2, 4

[2] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. In *NeurIPS*, 2022. 1, 2

[3] Yutong Bai, Xinyang Geng, Karttikeya Mangalam, Amir Bar, Alan L Yuille, Trevor Darrell, Jitendra Malik, and Alexei A Efros. Sequential modeling enables scalable learning for large vision models. In *CVPR*, 2024. 2

[4] Pratyay Banerjee, Tejas Gokhale, and Chitta Baral. Self-supervised test-time learning for reading comprehension. In *NAACL*, 2021. 2

[5] Amir Bar, Yossi Gandelsman, Trevor Darrell, Amir Globerson, and Alexei Efros. Visual prompting via image inpainting. In *NeurIPS*, 2022. 1, 2, 3

[6] Alexander Bartler, Andre Bühler, Felix Wiewel, Mario Döbler, and Bin Yang. Mt3: Meta test-time training for self-supervised test-time adaption. In *AISTATS*, 2022. 2

[7] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *NeurIPS*, 2020. 1, 2, 3

[8] Dian Chen, Dequan Wang, Trevor Darrell, and Sayna Ebrahimi. Contrastive test-time adaptation. In *CVPR*, 2022. 2

[9] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020. 3

[10] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *JMLR*, 2024. 2

[11] Samuel Dodge and Lina Karam. Quality resilient deep neural networks. *arXiv preprint arXiv:1703.08119*, 2017. 2

[12] Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *ICCV*, 2015. 3

[13] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. 2

[14] Chun-Mei Feng, Kai Yu, Yong Liu, Salman Khan, and Wangmeng Zuo. Diverse data augmentation with diffusions for effective test-time prompt tuning. In *ICCV*, 2023. 3

[15] Yang Fu, Sifei Liu, Umar Iqbal, Shalini De Mello, Humphrey Shi, and Jan Kautz. Learning to track instances without video annotations. In *CVPR*, 2021. 2

[16] Yossi Gandelsman, Yu Sun, Xinlei Chen, and Alexei Efros. Test-time training with masked autoencoders. In *NeurIPS*, 2022. 2, 3, 6

[17] Robert Geirhos, Carlos RM Temme, Jonas Rauber, Heiko H Schütt, Matthias Bethge, and Felix A Wichmann. Generalisation in humans and deep neural networks. In *NeurIPS*, 2018. 2

[18] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *ICLR*, 2018. 2, 3

[19] Ross Girshick. Fast r-cnn. In *ICCV*, 2015. 4

[20] Nicklas Hansen, Rishabh Jangir, Yu Sun, Guillem Alenyà, Pieter Abbeel, Alexei A Efros, Lerrel Pinto, and Xiaolong Wang. Self-supervised policy adaptation during deployment. In *ICLR*, 2021. 2

[21] Yaru Hao, Haoyu Song, Li Dong, Shaohan Huang, Zewen Chi, Wenhui Wang, Shuming Ma, and Furu Wei. Language models are general-purpose interfaces. *arXiv preprint arXiv:2206.06336*, 2022. 1

[22] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, 2022. 2, 3

[23] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *ICLR*, 2019. 1, 2, 4

[24] Adilbek Karmanov, Dayan Guan, Shijian Lu, Abdulmotaleb El Saddik, and Eric Xing. Efficient test-time adaptation of vision-language models. In *CVPR*, 2024. 3

[25] Wei Li, Jiahao Xie, and Chen Change Loy. Correlational image modeling for self-supervised visual pre-training. In *CVPR*, 2023. 3

[26] Zhi Li, Shaoshuai Shi, Bernt Schiele, and Dengxin Dai. Test-time domain adaptation for monocular depth estimation. In *ICRA*, 2023. 2

[27] 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 *ECCV*, 2014. 2, 4

[28] 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? In *NeurIPS*, 2021. 2

[29] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019. 4

[30] Xiaosong Ma, Jie Zhang, Song Guo, and Wenchao Xu. Swapprompt: Test-time prompt adaptation for vision-language models. In *NeurIPS*, 2023. 3

[31] Claudio Michaelis, Benjamin Mitzkus, Robert Geirhos, Evgenia Rusak, Oliver Bringmann, Alexander S Ecker, Matthias Bethge, and Wieland Brendel. Benchmarking robustness in object detection: Autonomous driving when winter is coming. *arXiv preprint arXiv:1907.07484*, 2019. 1, 2, 4

[32] Mirco Mutti and Aviv Tamar. Test-time regret minimization in meta reinforcement learning. In *ICML*, 2024. 2

[33] Muhammad Ferjad Naeem, Yongqin Xian, Xiaohua Zhai, Lukas Hoyer, Luc Van Gool, and Federico Tombari. Silc: Improving vision language pretraining with self-distillation. In *ECCV*, 2024. 2

[34] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *ECCV*, 2016. 3

[35] OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023. 2

[36] Matteo Poggi, Alessio Tonioni, Fabio Tosi, Stefano Mattoccia, and Luigi Di Stefano. Continual adaptation for deep stereo. *TPAMI*, 2021. 2

[37] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021. 2

[38] Mattia Segu, Bernt Schiele, and Fisher Yu. Darth: holistic test-time adaptation for multiple object tracking. In *ICCV*, 2023. 2

[39] Manli Shu, Weili Nie, De-An Huang, Zhiding Yu, Tom Goldstein, Anima Anandkumar, and Chaowei Xiao. Test-time prompt tuning for zero-shot generalization in vision-language models. In *NeurIPS*, 2022. 3

[40] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012. 2, 4

[41] 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 *ICLR*, 2020. 2, 3, 6

[42] Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023. 2

[43] Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024. 2

[44] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *IROS*, 2017. 2

[45] Alessio Tonioni, Fabio Tosi, Matteo Poggi, Stefano Mattoccia, and Luigi Di Stefano. Real-time self-adaptive deep stereo. In *CVPR*, 2019. 2

[46] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023. 2

[47] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023. 2

[48] Yun-Yun Tsai, Chengzhi Mao, and Junfeng Yang. Convolutional visual prompt for robust visual perception. In *NeurIPS*, 2023. 2

[49] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. In *ICLR*, 2019. 2

[50] Igor Vasiljevic, Ayan Chakrabarti, and Gregory Shakhnarovich. Examining the impact of blur on recognition by convolutional networks. *arXiv preprint arXiv:1611.05760*, 2016. 2

[51] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. In *ICLR*, 2021. 2

[52] Xinlong Wang, Wen Wang, Yue Cao, Chunhua Shen, and Tiejun Huang. Images speak in images: A generalist painter for in-context visual learning. In *CVPR*, 2023. 1, 2, 3, 4

[53] Chen Wei, Wenjing Wang, Wenhan Yang, and Jiaying Liu. Deep retinex decomposition for low-light enhancement. In *BMVC*, 2018. 2, 4

[54] Jiahao Xie, Xiaohang Zhan, Ziwei Liu, Yew-Soon Ong, and Chen Change Loy. Delving into inter-image invariance for unsupervised visual representations. *IJCV*, 2022. 3

[55] Jiahao Xie, Wei Li, Xiaohang Zhan, Ziwei Liu, Yew Soon Ong, and Chen Change Loy. Masked frequency modeling for self-supervised visual pre-training. In *ICLR*, 2023. 3

[56] Hai Ye, Yuyang Ding, Juntao Li, and Hwee Tou Ng. Robust question answering against distribution shifts with test-time adaptation: An empirical study. In *Findings of EMNLP*, 2022. 2

[57] Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, Ming-Hsuan Yang, and Ling Shao. Learning enriched features for real image restoration and enhancement. In *ECCV*, 2020. 2, 4

[58] Maxime Zanella and Ismail Ben Ayed. On the test-time zero-shot generalization of vision-language models: Do we really need prompt learning? In *CVPR*, 2024. 3

[59] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *ICCV*, 2023. 2

[60] Xiaohang Zhan, Jiahao Xie, Ziwei Liu, Yew-Soon Ong, and Chen Change Loy. Online deep clustering for unsupervised representation learning. In *CVPR*, 2020. 3

[61] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. Theoretically principled trade-off between robustness and accuracy. In *ICML*, 2019. 2

[62] Stephan Zheng, Yang Song, Thomas Leung, and Ian Goodfellow. Improving the robustness of deep neural networks via stability training. In *CVPR*, 2016. 2

[63] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *CVPR*, 2017. 2, 4