Supplementary for Paper – ENVEDIT: Environment Editing for Vision-and-Language Navigation

Jialu Li Hao Tan Mohit Bansal
UNC Chapel Hill
{jialuli, airsplay, mbansal}@cs.unc.edu

1. Overview

In this supplementary, we first describe the implementation details of our editing methods and VLN training in Sec. 2 and explain the evaluation metrics we use in Sec. 3. We show the back translation results for models trained with CLIP-ViT-32 and CLIP-ViT-16 features on different created environments in Sec. 4, and adapt our ENVEDIT to another pretrained VLN agent in Sec. 5. We explore the impact of image preprocessing during feature extraction in Sec. 6. We then explore curriculum learning for training on both the original environments and the created environments in Sec. 7. Furthermore, we show the performance of training with the environments created by masking out different numbers of classes in Sec. 8. Moreover, we include our agent's performance for each different unseen environments in Sec. 9 and analyze the performance variation when adapting different edited environments to different VLN base agents in Sec. 10. Lastly, we show qualitative examples from our created environments in Sec. 11. Limitations of our work is discussed in Sec. 12.

2. Implementation Details

In the environment creation stage, the style transfer model is directly adopted from [9]. We train the image synthesis model for 10 epochs on the training environments in Room-to-Room dataset with the default hyper-parameters from [11]. The model is trained to generate new environments given the semantic segmentation of the original environments. In the Vision-and-Language Navigation training, we use the default hyper-parameters from [7, 14] when training their models on R2R dataset. On the RxR dataset, we replace the bi-directional LSTM based instruction encoder in [13] with a cased multi-lingual $\text{BERT}_{\text{BASE}}$ [4]. The instruction encoder is optimized with AdamW [10] and linear-decayed learning rate (peak at 4e-5). Other parameters are the same as in [13]. In the back translation stage, the speaker and the agent are trained with the same hyper-parameters as in [14].

3. Evaluation Metrics

We evaluate our model with six metrics: (1) Success Rate (SR): whether the agent could stop within 3 meters of the target. (2) Success Rate weighted by Path Length (SPL) [1], which penalizes navigation with long paths. (3) Trajectory Length (TL): path length in meters. (4) Navigation Error (NE): the distance between the target and the agent's end point in meters. (5) normalized Dynamic Time Warping (nDTW) [8]: penalizes agents for deviating from the reference path. (6) success rate weighted by normalized Dynamic Time Warping (sDTW) [8]: only consider the nDTW score for successful paths and set the score to 0 for failed paths. SR, SPL are the main metrics for evaluation on R2R dataset, and nDTW, sDTW are the main metrics for RxR dataset.

4. Back Translation Results

We show the performance of training with the original environment and one type of newly-augmented environments (i.e., E_{st}, E_{is_1}, E_{is_2}, $E^m_{is_1}$, $E^m_{is_2}$) on R2R dataset in Table 1. Back translation with the style-aware speaker is applied in these experiments. We can see that training with any of our created environments improves the baseline model by a large margin, which is consistent with the performance before back translation. Specifically, after back translation, E_{is_1} works the best in SR for ViT-B/32 features, and $E^m_{is_1}$ works the best in SR for ViT-B/16 features, outperforming the baseline model by 4.9% and 3.1% respectively. Besides, after back translation, training on environments created with image synthesis method gets higher performance than environments created with style transfer approach. Specifically, model trained on environments created with image synthesis method improves 3.5% with back translation while model trained on environments created with style transfer approach improves 2.1%[1]. This indicates models trained with different environments benefit differently from back translation.

[1] Performance without back translation is in Table 1 in the main paper

	Environment Components			ViT-B/32				ViT-B/16			
Models	**Style**	**Appearance**	**Object**	**TL**	**NE↓**	**SR↑**	**SPL↑**	**TL**	**NE↓**	**SR↑**	**SPL↑**
EnvDrop* [13]	✗	✗	✗	16.128	4.794	55.6	49.7	18.844	4.429	57.9	50.6
E_{st}	✓	✗	✗	15.912	4.335	60.2	**53.8**	15.916	4.344	60.3	53.3
E_{is_1}	✗	✓	✗	17.495	**4.325**	**60.5**	53.1	16.984	4.387	59.3	52.3
E_{is_2}	✓	✓	✗	17.945	4.650	58.6	51.1	18.189	4.423	59.1	51.1
$E^m_{is_1}$	✗	✓	✓	17.956	4.531	58.5	51.0	17.989	4.232	**60.8**	54.2
$E^m_{is_2}$	✓	✓	✓	17.283	4.405	58.2	51.1	16.204	**4.181**	60.4	**54.4**

Table 1. Performance of training the agent with one kind of our edited environments and back translation. Results are on R2R val-unseen set. ViT-B/32(16) indicate image features extracted with different CLIP-ViT models [12]. "*" indicates reproduced results. ✓ indicates the environment component in the new environment is different from the original environment, while ✗ indicates the same.

Models	**TL**	**NE↓**	**SR↑**	**SPL↑**
⟳BERT-p365* [7]	12.425	4.104	62.1	56.3
E_{st}-p365	12.551	4.054	62.4	56.3
E_{is_1}-p365	11.749	3.930	62.6	**57.4**
$E^m_{is_1}$-p365	11.739	**3.917**	**62.7**	56.7
⟳BERT-16 [7]	11.564	4.019	61.9	57.1
E_{st}-16	12.233	3.880	63.5	57.6
E_{is_1}-16	11.956	**3.808**	**64.9**	**58.1**
$E^m_{is_1}$-16	12.023	3.937	63.2	57.4

Table 2. Performance of applying our proposed method to SotA VLN agents on R2R validation unseen set. "-16" indicates image features extracted with ViT-B/16, "-p365" indicates image features extracted with ResNet [6] pre-trained on ImageNet [3] and fine-tuned on Place365 [15]. "*" indicates reproduced results.

5. Performance on Rec-BERT

In this section, we show that our ENVEDIT is complementary to other VLN pre-training methods. We enhance the VLN pre-traind model [7] with our methods and illustrate the improvements on R2R dataset.

The model architecture of [7] is based on transformer. The weights of their model are initialized with PREVALENT [5], a multi-modal transformer pre-trained on in-domain VLN data. In both works, image features are extracted with the ResNet [6] which is pre-trained on ImageNet [3] and fine-tuned on Place365 [15]. We first show results using the same image features as their work, and further explore adapting ViT-B/16 features for their model.

As shown in Table 2, when trained with "place365" features, augmenting the original environment with $E^m_{is_1}$ could improve the baseline by 0.5% in SR and 1.1% in SPL. Augmenting with the other two environments could also improve the baseline by 0.5% in both SR and SPL. This demonstrates the effectiveness of adapting our method to other strong SotA VLN models.

To use "ViT-B/16" features, we map the 512 dimension feature to 2048 with a linear layer with dropout.[2] Though this way of adapting features could not fully take advantage of in-domain pre-training [5] on "places365" features, the baseline "⟳BERT-16" shows competitive performance compared with "⟳BERT-place365" on both SR and SPL. Augmenting the original environments with E_{is_1} achieves the best performance, improving the baseline by 3.0% in SR and 1.0% in SPL. Similar improvements are observed for the other two kinds of new environments (i.e., 1.6% in SR for E_{st} and 1.3% in SR for $E^m_{is_1}$), validating the effectiveness of our proposed approach other over SotA methods.

6. Image Preprocessing Variants

In this section, we explore whether image preprocessing will influence the performance when adapting ENVEDIT to SotA pre-trained VLN agents ⟳BERT [7]. Our experiments are based on features extracted with ResNet [6] pre-trained on ImageNet [3] and fine-tuned on Place365 [15].

Our image input is of $640 \times 480 \times 3$. In the first preprocessing, we resize the image to $256 \times 256 \times 3$ and then crop the image at the center to $224 \times 224 \times 3$ before normalization, which is consistent with the image preprocessing when evaluating on ImageNet classification task. We then directly adopt the features from the last pooling layer of the ResNet. We use this preprocessing in our main paper. In the second preprocessing, instead of transforming the image size to match the required ResNet input size, we follow [2] to change the last pooling layer in ResNet to global pooling, and does not further do any resizing on input image except normalization. This preprocessing is also used in previous VLN works.

We show the results for two preprocessing methods in Table 3. We can see that removing the image transformation in the preprocessing significantly improves the baseline performance (⟳BERT-1 vs. ⟳BERT-2). This is due to the baseline model is pre-trained on "place365" features extracted by [2], which does not use image resize and crop. We also observe that after training on both the original envi-

[2]Since [5] does not provide source code for their in-domain pre-training method, it's hard to start from pre-training a multi-modal transformer with "ViT-B/16" features and then transfer to [7] model.

Models	TL	NE↓	SR↑	SPL↑
↻BERT-1* [7]	12.421	4.294	59.5	53.6
E_{st}-1	12.551	4.054	62.4	56.3
E_{is_1}-1	11.749	3.930	62.6	**57.4**
$E^m_{is_1}$-1	11.739	**3.917**	**62.7**	56.7
↻BERT-2* [7]	11.327	4.028	61.8	56.3
E_{st}-2	11.760	3.921	62.8	56.6
E_{is_1}-2	12.319	**3.884**	**63.2**	**56.8**
$E^m_{is_1}$-2	12.034	4.016	62.3	56.3

Table 3. Performance of applying our proposed method to SotA VLN agents on R2R validation unseen set. "-1"/"-2" indicates using the first/second way of preprocess, "*" indicates results with our extracted "place365" features.

Models	TL	NE↓	SR↑	SPL↑
EnvDrop* [13]	15.86	4.73	55.1	48.8
E_{st}	16.59	4.69	58.2	51.5
E_o + E_{st} + CL	20.98	5.14	49.8	43.3
E_{st} + E_o + CL	16.08	4.63	56.8	50.5
E_{is_1}	17.69	4.76	56.4	48.9
E_o + E_{is_1} + CL	26.43	5.60	47.3	39.1
E_{is_1} + E_o + CL	17.27	4.95	55.3	48.0
$E^m_{is_1}$	14.46	4.67	57.3	51.1
E_o + $E^m_{is_1}$ + CL	26.54	5.77	48.4	38.9
$E^m_{is_1}$ + E_o + CL	17.31	4.83	55.5	48.2
E_{st} + E_{st_2} + E_o + CL	15.85	4.67	56.2	49.8
E_{is_1} + E_{st} + E_o + CL	17.47	4.83	56.2	49.1

Table 4. Performance of training the agent on the original environment and edited environment in different steps in curriculum learning on R2R validation unseen set. "-CL" indicates using curriculum learning, "*" indicates reproduced results.

ronment and our edited environment, the performance gap decreases (E_{is_1}-1 vs. E_{is_1}-2).

7. Curriculum Learning

In the main paper, we train the agent on both the original environment and the edited environment in the same batch. This section explores using curriculum learning to train the agent on the original environment and edited environment in different steps. Specifically, we train the models with two steps. In the first step, the model is trained on one of the newly created environments. In the second step, the model is trained on the original environments. Both steps contain 50,000 iterations with a batch size of 64. We use image features extracted with ViT-B/16 in this experiment. For simplicity, back translation is not applied in this experiment.

We notice that curriculum learning usually learns from easy samples first and gradually switch to hard samples. In our case, intuitively, easy samples should be the original environment and hard samples are our created environments.

Models	TL	NE↓	SR↑	SPL↑
$E^{m_1}_{is_1}$-16	14.46	4.67	57.3	51.1
$E^{m_2}_{is_1}$-16	17.01	4.86	55.6	48.7
$E^{m_3}_{is_1}$-16	17.05	4.94	54.6	48.4
$E^{m_4}_{is_1}$-16	16.54	4.95	54.2	47.7

Table 5. Performance of training on the environments created by masking out different number of objects on R2R validation unseen set. "-16" indicates image features extracted with ViT-B/16.

However, the performance of models first trained on original environment and then trained on newly created environments (as shown in Table 4) are much worse than training in the reverse way (e.g., "E_o + E_{st} + CL" vs. "E_{st}"). We attribute this to that the data in original environments are of higher quality.

Besides, as shown in Table 4, the model trained with curriculum learning gets lower performance compared with directly training the agent on both the original environment and the edited environment in the same batch. This indicates that more advanced curriculum learning (e.g., a better-designed approach to rank the difficulty of the samples) is needed to get higher performance.

We further explore using curriculum learning to train the agent on multiple edited environments and the original environment. Specifically, we train the models with three steps. In the first two steps, the agent is trained on two different edited environments. In the last step, the agent is trained on the original environments. All three steps contain 50,000 iterations with a batch size of 64. For simplicity, back translation is not applied in this experiment as well.

As shown in Table 4, we first create a new environment E_{st_2} with style transfer approach. The style is sampled from a multivariate normal distribution. We train the agent on E_{st} and E_{st_2} in the first two steps sequentially in curriculum learning. We observe that the performance ("E_{st} + E_{st_2} + E_o + CL") is lower than only training on E_{st}. Similar results are observed for training on environments E_{is_1} and E_{st} in the first two steps in curriculum learning ("E_{is_1} + E_{st} + E_o + CL"). We further experiment with adaptive curriculum learning (aCL) to learn from easy examples to hard examples. The difficulty is measured with path generation probability and variance. We use Bayesian optimization to find the best hyperparameter for aCL based on the agent's performance of the first 10k iterations. The best model achieves 57.6%, which is still lower than E_{st} only. We believe how to combine the three editing environments during training to get better performance is a non-trivial problem and requires further investigation in future work.

ID	Environment	EnvDrop-16* [13]		E_{st}-16		E_{is_1}-16		$E^m_{is_1}$-16	
		SR	SPL	SR	SPL	SR	SPL	SR	SPL
1		59.3	50.4	59.7	52.3	63.3	52.1	51.3	52.1
2		50.4	42.8	51.1	43.8	49.7	39.7	52.5	47.1
3		77.8	59.8	75.6	61.1	77.8	59.9	64.4	51.5
4		56.8	48.7	59.9	52.0	59.4	50.0	63.5	55.4
5		42.7	37.1	57.7	50.9	48.3	42.4	52.7	47.0
6		49.0	45.3	50.7	46.6	50.7	43.0	49.3	45.5
7		56.1	53.6	50.6	46.1	52.8	49.2	53.3	50.2
8		49.5	45.3	47.0	42.8	51.6	47.6	48.0	43.8
9		69.1	60.1	71.4	62.1	67.0	58.5	66.3	59.9
10		60.7	55.5	68.0	60.1	60.3	54.2	65.0	58.2
11		16.7	14.4	33.3	27.2	16.7	14.2	55.6	46.6

Table 6. The results of our ENVEDIT on different environments in validation unseen set. "-16" indicates image features extracted with ViT-B/16.

ID	Environment	EnvDrop-16* [13]		EnvDrop+BT		⟳BERT-p365* [7]	
		SR	SPL	SR	SPL	SR	SPL
1		59.3	50.4	61.3	51	67.3	62.2
2		50.4	42.8	61	50.3	58.9	53.9
3		77.8	59.8	71.1	50.6	77.8	66.1
4		56.8	48.7	60.9	52	63	56.8
5		42.7	37.1	51.7	43.7	55.3	50.6
6		49.0	45.3	55.3	49.4	50.7	47.8
7		56.1	53.6	50.6	49	56.1	53.5
8		49.5	45.3	55.2	50.4	63.8	59.2
9		69.1	60.1	66.3	57.9	71.1	77.9
10		60.7	55.5	58.3	53.5	65.3	60.6
11		16.7	14.4	22.2	18.9	61.1	57.3

Table 7. The results of different VLN baseline models on different environments in validation unseen set. "-16" indicates image features extracted with ViT-B/16.

8. Editing Different Number of Objects

We edit the objects in the original environments by randomly masking out some classes in the semantic segmentation (as discussed in Sec.4.2 in the main paper) In this section, we explore the impact of editing different numbers of objects in the original environments. Specifically, we randomly mask out 1 to 4 classes in the semantic segmentation during the inference time of environment editing, and create four environments respectively ($E^{m_1}_{is_1}$, $E^{m_2}_{is_1}$, $E^{m_3}_{is_1}$, $E^{m_4}_{is_1}$), where $E^{m_i}_{is_1}$ indicates masking out i classes in the semantic segmentation. $E^{m_1}_{is_1}$ is the environment $E^m_{is_1}$ used in the main paper.

As shown in Table 5, the performance consistently decreases while masking out more object classes in the environments. These results demonstrate the importance of balancing between the matches with the original instruc-

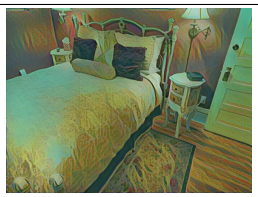
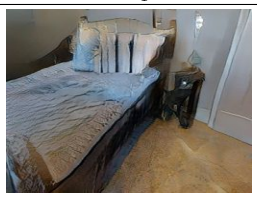
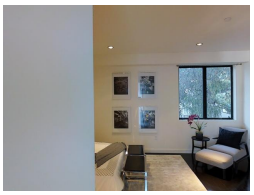
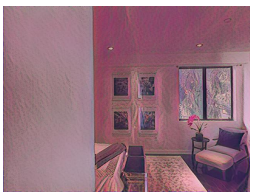
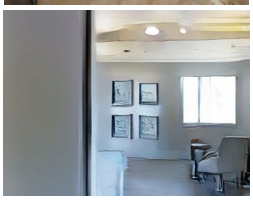
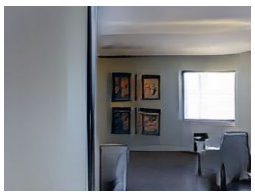
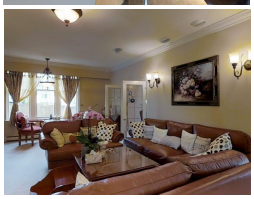
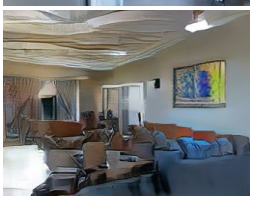
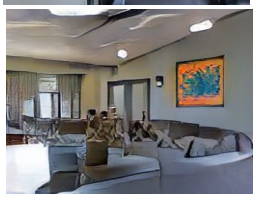
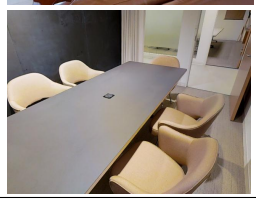
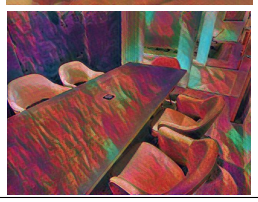
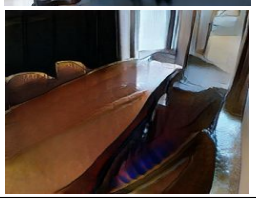
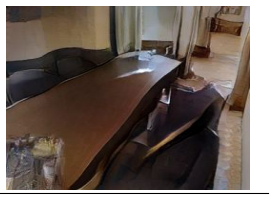
ID	Original	E_{st}	E_{is_1}	$E^m_{is_1}$
1				
2				
3				
4				
5				
6				

Table 8. Qualitative Examples of our created environments E_{st}, E_{is_1}, $E^m_{is_1}$.

tions and the diversity of new environments, and we find that masking out 1 object class would empirically give the best results in our experimental setup.

9. Detailed Results on Different Environments

We show the results of the overall averaging performance on all unseen environments in the main paper. In this section, we want to have a detailed look at the performance of our model on different environments and explore which unseen environments benefit more from our ENVEDIT. As shown in Table 6, we observe that training on E_{st} could improve the performance on Environment 5 by 15.0% in SR and 13.8% in SPL, and improve the performance on Environment 11 by 7.2% in SR and 4.6% in SPL. When training on E_{is_1}, the Environment 1 gets the largest improvement. A possible reason is that this environment differ from the original environment mainly in object appearance, and our ENVEDIT creates environments with new object appearance, and thus generalize better. Besides, this indicates that environments in the validation unseen set benefit differently from training on different created environments, and the newly created environments contain complementary information for generalization. This points to a future direction where the agent learns from multiple new environments to generalize better to unseen environments.

10. Analysis of Consistency Across Different Environments

In Sec 9, we show that different environments could benefit differently from different editing methods (i.e., Env5 gets 15% improvement by E_{st}, while Env11 gets 38.9% improvement by $E^m_{is_1}$). Similarly, in this section, we show that different VLN models (EnvDrop/EnvDrop+Back Translation(EnvDrop+BT)/⟳BERT-p365) will be better at generalization to different environments. As shown in Table 7, for VLN models trained on the original environment, while 'EnvDrop' generalizes bad to Env5 (42.7%), 'EnvDrop+BT' improves it by 9% (51.7%) and '⟳BERT-p365' improves it by 12.6% (55.3%) on Env5. Thus, the improvement brought by E_{st} might be less for 'EnvDrop+BT' and '⟳BERT-p365' compared with 'EnvDrop'. Similarly, $E^m_{is_1}$ can benefit 'EnvDrop+BT' more since 'EnvDrop+BT' still works bad on Env11 (22.2%). This explains why the best model for each VLN model and visual feature will be different. Nevertheless, picking any of the editing methods will improve the performance, which demonstrates the effectiveness of our environment-level data augmentation in tackling the data scarcity problem. Considering both simplicity and performance across different models, we recommend using E_{st} as the start point for future research.

11. Qualitative Examples for Generated Environments

We show some more examples in our generated environments in Table 8. We could see that the environments generated with the style transfer approach (denoted as E_{st} in Table 8) maintain the semantics of the original environments better, but the overall style is artistic. The environments generated with the image synthesis approach (denoted as E_{is_1} and $E^m_{is_1}$ in Table 8) bring more diversity in object appearance and are more close to real environments.

12. Limitations

We also note that there are some limitations of our work. First, this work explores two classic style transfer and image synthesis approaches in augmenting the VLN training data. More advanced image-to-image translation models can potentially generate environments with higher quality and thus bring further improvement. Besides, this work dedicatedly considers the Vision-and-Language Navigation task, but the proposed method can be potentially used in other embodied tasks and simulated environments with certain adaptations. We will explore other useful and interesting tasks in the future.

References

[1] Peter Anderson, Angel Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, et al. On evaluation of embodied navigation agents. *arXiv preprint arXiv:1807.06757*, 2018. 1

[2] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton Van Den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3674–3683, 2018. 2

[3] 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*, pages 248–255. Ieee, 2009. 2

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, June 2019. 1

[5] Weituo Hao, Chunyuan Li, Xiujun Li, Lawrence Carin, and Jianfeng Gao. Towards learning a generic agent for vision-and-language navigation via pre-training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13137–13146, 2020. 2

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 2

[7] Yicong Hong, Qi Wu, Yuankai Qi, Cristian Rodriguez-Opazo, and Stephen Gould. A recurrent vision-and-language bert for navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1643–1653, June 2021. 1, 2, 3, 4

[8] Gabriel Ilharco, Vihan Jain, Alexander Ku, Eugene Ie, and Jason Baldridge. General evaluation for instruction conditioned navigation using dynamic time warping. *NeurIPS Visually Grounded Interaction and Language (ViGIL) Workshop*, 2019. 1

[9] Philip TG Jackson, Amir Atapour Abarghouei, Stephen Bonner, Toby P Breckon, and Boguslaw Obara. Style augmentation: data augmentation via style randomization. In *CVPR Workshops*, pages 83–92, 2019. 1

[10] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. 1

[11] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2337–2346, 2019. 1

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

[13] Sheng Shen, Liunian Harold Li, Hao Tan, Mohit Bansal, Anna Rohrbach, Kai-Wei Chang, Zhewei Yao, and Kurt Keutzer. How much can clip benefit vision-and-language tasks? *ICLR*, 2022. 1, 2, 3, 4

[14] Hao Tan, Licheng Yu, and Mohit Bansal. Learning to navigate unseen environments: Back translation with environmental dropout. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2610–2621, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. 1

[15] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017. 2