Cleaning the Streets of Montevideo

December 7th, 2022

Thomas Wimmer
TUM, IP Paris
thomas.wimmer@ip-paris.fr

Lucas McIntyre
École Polytechnique
lucas.mcintyre@polytechnique.edu

Abstract

Detecting and localizing litter in public places is one of the challenges that must be overcome to enable the use of advanced cleaning robots in the city of tomorrow. In this project, we first trained a classification model that can detect the accumulation of litter, and then used class activation maps to evaluate the predictions of the model. Ultimately, we explored different methods to digitally "clean" the streets of Montevideo, namely the usage of a CycleGAN model and image inpainting.

1. Introduction

The shift from human operation to more autonomous systems is one of the most important trends, especially for services that are particularly labor-intensive and physically demanding. An example of such work, which is usually physically demanding but not necessarily cognitively demanding, is the identification and cleaning of garbage scenes in cities. Automated approaches to the challenge of automated cleaning already exist at the household level with "smart" robotic vacuum cleaners, but the implementation of these machines is generally very limited in terms of artificial intelligence. It is therefore not straightforward to transfer these robots to cleaning waste scenes in cities (although several approaches to automation already exist, e.g. by Veolia). The main reason for this is that garbage scenes in cities are usually very complex and the identification of garbage scenes requires a deep understanding of the environment in which the robot operates.

The goal of this project is to develop a system capable of autonomously identifying garbage in urban scenes and to explore the explainability of the system's decisions. In addition, the system should also be able to "clean up" the trash scenes by removing the trash from the images.

2. Identification of Dirty Scenes

The identification of dirty scenes can be modeled as a binary classification task (clean/dirty). The dataset used in this project is described in the next section. Different approaches can be used for this standard computer vision task. The methods used in this project are described in section 2.2.

2.1. Dataset

The dataset is composed of 3416 images of urban scenes showing road bins in Montevideo, Uruguay. The photos are gathered from Google Street View, social networks (Twitter & Facebook), individual's camera roll, and from pormibarrio.uy, a collaborative project from DATA Uruguay aimed at cleaning the streets of Montevideo. Starting from an idea to help the uruguayan community, some people talked on a Twitter thread and this discussion led to a Kaggle challenge based on this dataset. It was even talked about in the country's news, notably in journals like *El Pais*, *El Observador*, and *La mañana en casa*.

The pictures fit into two categories: some show clean streets, with waste nicely contained inside the bins, some show dirty streets with containers overflowing with garbage. They vary in size depending on their source. For our project we used the latest dataset (clean-dirty-garbage-containers-V6.1). It is already split into two folders, test (1197 items) and train (2219 items), for us. Each folder is split into dirty and clean images, with: 601 clean and 596 dirty test images, and 1207 clean and 1012 dirty train images.

2.2. Methods

We decided to use CNNs pre-trained on the ImageNet dataset, available through the torchvision library, and optimize them for the downstream task of dirty scene classification. From the wide range of available models, we decided to use the ResNet models [1] as a base and the EfficientNet models [4] as a more sophisticated model architecture.

The fine-tuning task requires replacing the fully-connected layers in the end of the respective networks with

new layers that are ending in a single output neuron (instead of the previously used 1000 outputs for the ImageNet classification task). We designed our models to be as flexible as possible and experimented with different network depths and widths for the fully-connected part in the end of the networks, as well as different scales of the ResNet / EfficientNet architecture used. We further introduced dropout layers that are interleaved with the other layers and can help preventing overfitting and thus act as a regularization for the NN.

To find the optimal initial learning rate, we utilized the LR-Finder module that implements methods to find the optimal LR as described in [3]. Since fine-tuning neural networks needs a careful handling of the learning rate to not "unlearn" weights and overfit on the smaller dataset, we also implemented the use of an adaptive LR scheduler that decreases the learning rate as soon as a plateau is reached or the validation loss is even increasing.

Another important measure that one can take to avoid overfitting to the training set and to improve generalization of the model is the usage of data augmentation. We used several different augmentation strategies:

- Random horizontal flips
- Random rotations (up to 10°)
- Random affine transformations (introducing shear and scaling)
- Color jitter
- Random transformation to grayscale images
- Random erasing of image parts

The training procedure itself can also be considered a hyperparameter that needs to be tuned. We have experimented with different training methods. Freezing the pre-trained convolutional layers during the initial training steps and performing an initial training only on the randomly initialized new fully connected layers proved to be useful. However, fine-tuning the weights of the entire network (including the convolutional layers) was shown to have a large positive impact on the performance of the model. We experimented with more sophisticated training strategies that included training on augmented and non-augmented data, and alternating between freezing and unfreezing convolutional layers.

Evaluation To evaluate the performance of the trained models, we compute several different metrics on the predefined test set:

- True Positive Rate (Recall): $TPR = \frac{TP}{P}$
- False Positive Rate: $FPR = \frac{FP}{N}$
- True Negative Rate: $TNR = \frac{TN}{N}$

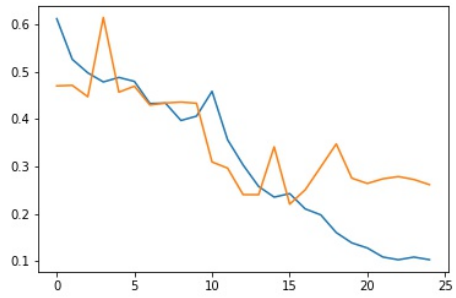


Figure 1. Training (blue) and Validation (orange) loss during the training of the EfficientNet-B0.

- False Negative Rate: $FNR = \frac{FN}{P}$
- Precision $= \frac{TP}{PP}$
- Accuracy $= \frac{TP+TN}{P+N}$
- F1 Score $= \frac{2TP}{2TP+FP+FN}$

In our case, the most important of the first four first mentioned metrics is probably the "true positive" (recall) rate, since it is more important to detect all trash piles and accidentally mark some clean bins than to detect only parts of the trash piles and miss some.

Besides evaluating the performance quantitatively, we can also do a qualitative analysis through visualizing failure cases (such as false negatives or false positives).

2.3. Results

Achieving an accuracy of 0.9347 on this task seems to be a reasonably good result. The network trained with the EfficientNet-B0 as the pre-trained convolutional basis model proved to perform slightly better than the ResNet models, especially since it has a higher recall, which we identified as one of the key metrics for this task. It proved beneficial to freeze the weights of the pre-trained model at the beginning of training and start training the classifier on non-augmented images. However, the training procedure must also include fine-tuning of the convolutional layers, as this greatly improves performance. This can be seen in Figure 1, where fine-tuning of the convolutional layers on augmented data starts at epoch 10 and a significant decrease in losses is observed. However, care must be taken not to overfit these weights to the data. From the broad range of experiments that were carried out, we could observe that using a scaled version of the pre-trained CNNs (as e.g. EfficientNet-B5) did not bring a significant improvement in performance while taking more resources to train.

As for the visualization of false classifications, we can see that the dataset itself is not really clean and therefore it

Table 1. Evaluation of a selection of trained models to identify garbage piles.

Base model	EfficientNet-B0	ResNet-50	ResNet-18
Classifier	[1024, 256, 32]	[512, 128, 32]	[512, 256, 64]
Training strategy	Alternating Freezing / Augmentation	Alternating Freezing / Augmentation	No Freezing
TPR (Recall)	**0.9563**	0.9378	0.9261
FPR	0.0867	**0.0783**	0.0800
TNR	0.9133	**0.9217**	0.9200
FNR	**0.0437**	0.0622	0.0739
Precision	0.9163	**0.9223**	0.9199
Accuracy	**0.9347**	0.9297	0.9230
F1	**0.9359**	0.9300	0.9229

Figure 2. Examples of false predictions by the EfficientNet-B0 classifier. The top row shows false positives (scenes are not labeled as dirty but network predicts them as dirty), while the bottom row shows false negatives (actual dirt / garbage not detected).

may be impossible to achieve perfect prediction accuracy. If we look at the results in Figure 2, we can see that some images are classified as "clean" even though there are small piles of garbage next to the bins (e.g. in the third and fourth images in the top row). The fifth image in the top row shows an example that is very challenging for the models because the lighting in the scene is very poor, resulting in a low-contrast image. Although we have done rigorous data augmentation to avoid these cases, the model still sometimes has problems with them. On the other hand, we can observe that in some pictures, there are annotations possibly from other sources (e.g. the red lines in the second image of the bottom row). These minor annotations should not "confuse" the model at best, but are still not realistic, as they would not occur in a real scenario.

3. Network Interpretation

Identifying images that contain garbage piles and quantifying the performance by accuracy and other metrics is already a nice achievement. However, there exist other methods of verifying whether and what the networks have learned. In addition, we might want to actually localize the garbage piles in the images. For these tasks, we can make use of class activation maps which were first introduced in [5] and further developed in [2]. The following section explains the method used in this project in detail, while results of the analysis can be found in Section 3.2.

3.1. Methods

Class activation maps can help interpret the reasoning of CNNs by backtracking the gradients of the class of interest (in our case, there is only one class, namely "garbage") to an activation map (output of a convolutional layer/block) and projecting it back to the input image. The result is a heat map showing the regions in the input image that have the highest activation with respect to the predicted class in the selected convolutional layer. This idea becomes more clear with the examples shown in the next section.

We used an available implementation of the Grad-CAM method and adapted it to work with our implementation.

3.2. Results

Figure 3 presents the activation regions for several pictures, showing which parts of each image influenced the final decision of the network. In most cases (see true positives and true negatives, lines 5 to 8 of figure 3), we can see that our network manages to focus on the garbage piles when they exist, which constitutes a major success. When there is no garbage, there is sometimes some residual activation, but not enough to influence the final classification.

Let's now breakdown what happens for the different failure cases:

Some activation maps reveal mistakes or ambiguities in the labeling of the data. Indeed, in some cases of false positives (first two lines of figure 3), the classification algorithm points out the presence of waste outside containers in scenes that are a supposedly "clean"!

For false negatives (lines 3 and 4), we can see that the CNN output is "clean" even though the garbage is being detected by the algorithm. This shows good localization but not enough activation to influence the decision.

n Figure 4 it can also be seen that the activation maps of the classes vary between the different layers of the network. While in the first layers activation is still fairly distributed over the image with only small areas reaching low activation, activation is concentrated in larger areas with higher activation in deeper layers. This is similar to the general structure of CNNs and the receptive fields of the layers. While the first layers identify local patches that could be part of garbage piles, the deeper layers are able to merge these local features and identify the actual garbage in the scenes.

4. Cleaning dirty city images

We successfully implemented a neural network able to accurately recognize garbage piles and we were able to interpret its reasoning using activation maps. Our aim is now to use this knowledge to artificially "clean" the dirty scenes, using Cycle-Consistent Adversarial Networks (Cycle GAN). In other words, given our two unordered collections of dirty and clean scenes, we wanted to code an algorithm which can automatically "translate" an image from the dirty set to the clean set (and vice versa). As well as a nice visual computing achievement, this would help the community visualize the end goal of the uruguayan effort towards a clean environment.

The following section explains the method used for the coding of this network, and the results can be found in Section 4.2.

4.1. Methods

For this class of vision and graphics problems, the goal is usually to learn the mapping between an input image and an output image using a training set of aligned image pairs. However, for our task, paired training data is not available: we don't have pictures of exactly the same scene with and without waste.

We used a Cycle GAN, which is illustrated in Figure 5. This model includes two mappings G : X $\rightarrow$ Y and F : Y $\rightarrow$ X, joining the dirty images (X) and the clean ones (Y). The goal of the network is to learn the function G such that the distribution of cleaned images G(X) is indistinguishable from the distribution Y. We use two adversarial discriminators D_X and D_Y , where D_Y aims to distinguish between clean images $\{y\}$ and "cleaned" images G(x); in the same way, D_X aims to discriminate between dirty images $\{x\}$ and translated images F(y).

The pipeline includes two types of losses: the adversarial losses for matching the distribution of generated images to the data distribution in the target domain, and the cycle consistency losses to capture the intuition that if we translate from one domain to the other and back again we should arrive at where we started, e.g. x $\rightarrow$ G(x) $\rightarrow$ F(G(x)) $\approx x$

We then used these losses to carry out the training of an imported Pix2Pix model for image-to-image translation.

4.2. First results

We implemented the pipeline described in the paper [6] in two ways, once adapting an existing PyTorch implementation, once using an existing Tensorflow implementation to our project, to compare the results.

Unfortunately we encountered some difficulties when training the Cycle GAN. After 200 epochs, the generated image presented very little difference with the input, except a slight blur around the garbage (see figure 6). We found that this revealed various issues with our data and setup.

First of all, unlike the basic tutorial datasets of the paper, our dataset has problems which makes it hard to learn from. Some images have occlusions, some have bad lighting. The bins and garbage can be located all over the place, so there is no geometrical alignment as we can e.g. find in pictures of centered faces. Coupled with the fact that our work environment (Google Colab) wasn't suited for long trainings (1 hour train time for 20 epochs, constant timeouts, etc...) and knowing the difficulty of GAN training in general, we decided to leave the results as they were, and try another way of cleaning the dirty scenes.

4.3. The Inpaint method

In order to still have an algorithm that can "clean" the streets, we recalled that we do have an estimation of the location of garbage piles in the image and we used it in the

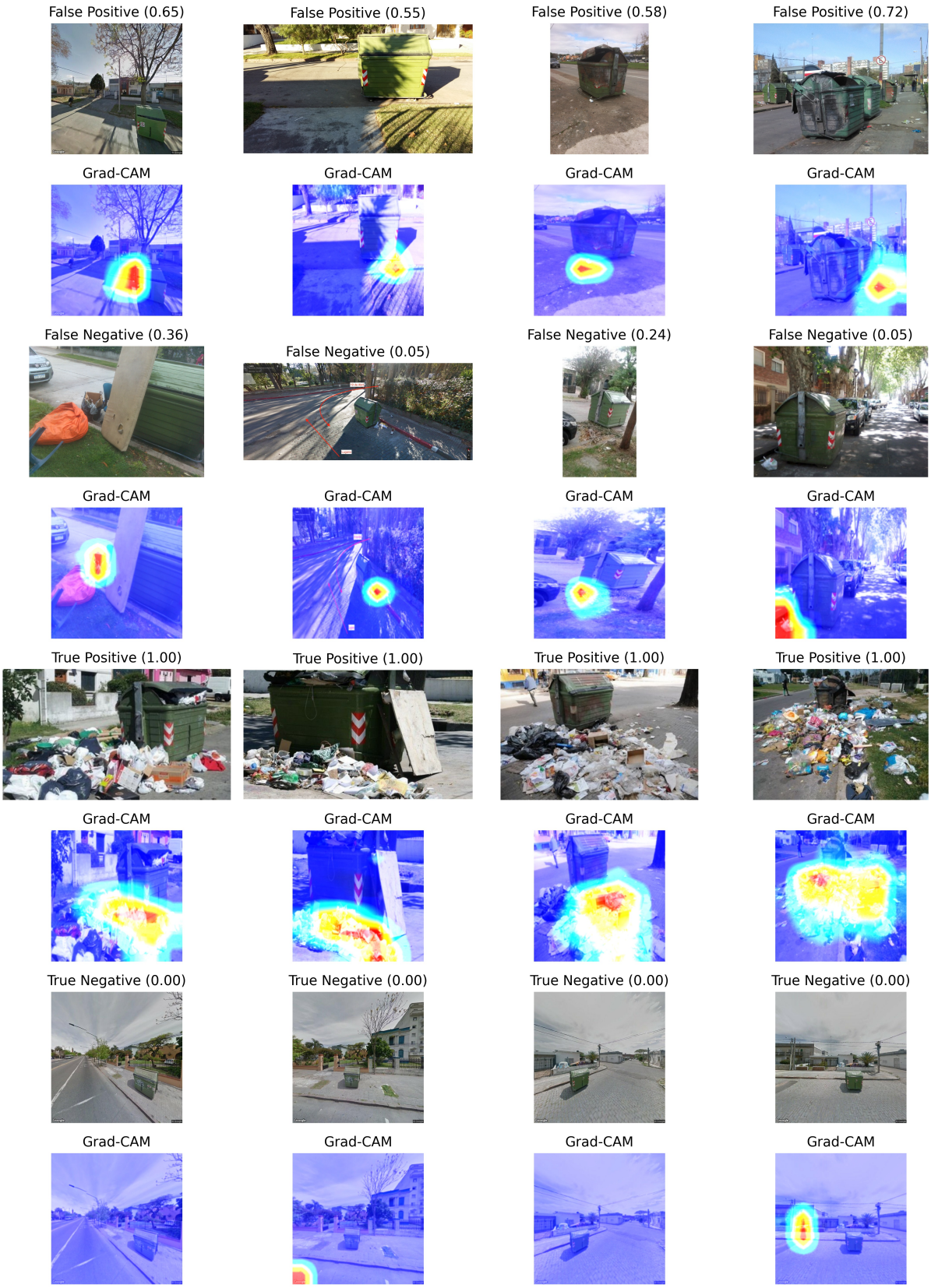


Figure 3. In-depth analysis of the predictions of our trained network.

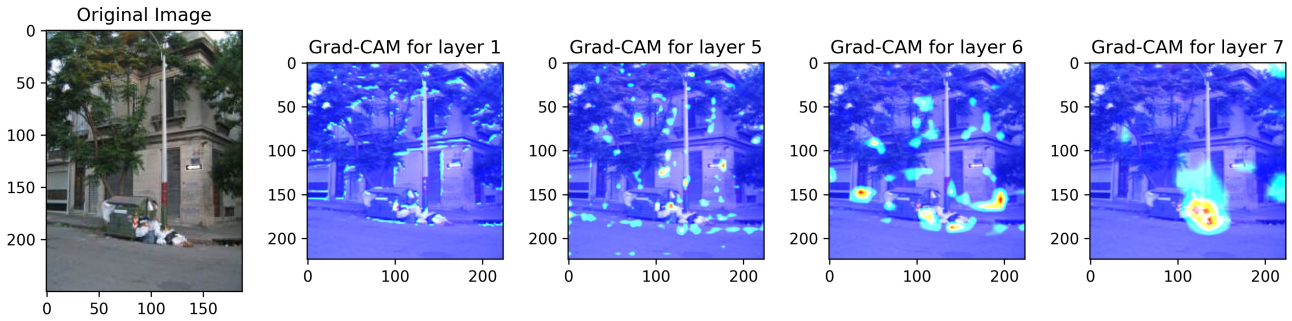
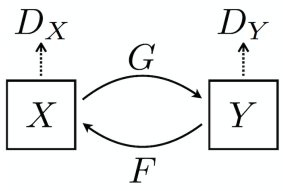


Figure 4. CAMs of different convolutional layers in the ResNet architecture.



(a)

Figure 5. (a) [X : set of dirty images, Y : set of clean images] Illustration of the two mapping functions and the associated adversarial discriminators D_Y and D_X.

Figure 6. Results after training the CycleGAN for 200 epochs.

following way: by thresholding the backprojected activation maps, we can identify the locations of garbage in the scenes. We can use the so-created masks in an image inpainting approach by removing the identified zones in the image and replacing it with the inpainting of a dedicated method. Since time and resources didn't allow for a neural solution, we decided to use the inpainting functionality from the OpenCV library. Some results are shown in Figure 7.

Although the algorithm does not produce high quality photorealistic results, it still works surprisingly well and produces reasonably good results. Since we use a classic inpainting method, the results are often blurry and especially with larger garbage piles this problem becomes visible, as can be seen in the last example in Figure 7.

5. Future Work

On the technical side, improving the model for artificially cleaning garbage scenes is an obvious choice for future work. Improving in this case means conducting further experiments to tune the parameters of the CycleGAN network that we know can work for similar tasks. On the other hand, we can drastically improve the results of the image inpainting method through utilizing an autoencoder network that was trained for image inpainting in a self-supervised manner on clean city scenes. Instead of using the class activation maps which are upsampled to the full image size from a hidden layer output and thus do not give clear edges, we could use a dedicated object d method. Since pixel-level annotations or bounding boxes are not available, training such a detection method would need to be carried out in a weakly-supervised manner.

Another improvement could be to have the classification model not only evaluate whether trash is present, but also estimate the amount of litter next to trash cans to help cleaners identify the dirtier scenes of the city. Weakly supervised training could prove beneficial in this case to avoid the large effort required to relabel the entire dataset.

As for the dataset itself, future efforts could include cleaning up the mislabeling for some images as well as capturing new data by having people send it in on site. In this regard, it might be beneficial to provide documentation of the dataset in Spanish.

Finally, an actual implementation of bots or simple cameras that are able to detect and report containers' statuses could actually bring research to practice. Now that we can accurately detect overflowing bins, programmers could develop and deploy a bot which might even be able to automatically clean scenes.

Figure 7. Results of the inpainting-based garbage removal algorithm.

6. Contributions

Detailed list of contributions to the project:

- Task 1: Data Loading (Thomas), Data Augmentation (Thomas), Model Loading & Training (Thomas and Lucas), Hyperparameter tuning (Thomas and Lucas), Evaluation and Visualization of Results (Thomas)
- Task 2: Adaptation of the CAM algorithm to our project (Thomas and Lucas), Visualization of CAMs for different network predictions (Thomas and Lucas)
- Task 3: Adaptation of Pytorch and Tensorflow versions of CycleGAN to our project (Lucas), Experimentation on data to understand the results (Thomas and Lucas), Implementation of the Inpaint method (Thomas and Lucas)

References

[1] 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. 1

[2] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017. 3

[3] Leslie N Smith. Cyclical learning rates for training neural networks. In *2017 IEEE winter conference on applications of computer vision (WACV)*, pages 464–472. IEEE, 2017. 2

[4] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019. 1

[5] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2921–2929, 2016. 3

[6] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017. 4