Appendix for GANMEX: One-vs-One Attributions Guided by GAN-based Counterfactual Baselines

A. Implementation Details

A.1. Datasets and Classifiers

MNIST (LeCun and Cortes 2010) The classifier consists of two 6x6 CNN layers with a stride of 2, followed by a 256-unit fully connected layer, a dropout layer with $p = 0.5$, and the 10 output neurons. As shown in Springenberg et al. (2015) the stride>1 CNN achieved comparable performance with pooling layers. The classifier was trained for 50 epochs and achieve a test accuracy of 99.3%.

Street-View House Numbers (SVHN) (Netzer et al. 2011) We tested our models on the cropped version of SVHN and used the same model architecture with that of MNIST and achieved a test accuracy of 90.3% after 50 epochs of training.

CIFAR10 ((Krizhevsky 2009)) We trained a classifier consist of 4 repetitive units, with each unit constructed by two 3x3 CNN layers and a 2x2 average pooling layer, with each CNN layer followed by a batch normalization layer. The classifer achieved 87.8% test accuracy after 100 epochs of training.

apple2orange (Zhu et al. 2017) We train a classifier taking the original 256x256 image as input. The classifier was constructed by adding a global average pooling layer on top of MobileNet (Howard et al. 2017), and then followed by a dense layer of 1024 neurons and a dropout layer of $p = 0.5$ before the output neurons. The classifier was trained for 50 epochs and achieve a test accuracy of 87.7%.

A.2. Baseline Generation with GANMEX

Our baseline generation process is based on StarGAN (Choi et al. 2017). We used the Tensorflow-GAN implementation (https://github.com/tensorflow/gan) and made the following two modifications (Equation 9):

1. The class discriminator D_{cls} is replace by the target classifier S to be explained.
2. A similarity loss $\mathcal{L}_{sim}$ is added to the training objective function.

We train the GANMEX model for 100k steps for the MNIST and apple2orange datasets, 300k steps for the SVHN dataset, and 400k steps for the CIFAR10 dataset. Only the train split is used for training, and the attribution results and evaluation were done on the test split of the dataset.

A.3. Attribution Methods

For our experiments, we used DeepExplain (https://github.com/marcoancona/DeepExplain) for generating saliency maps with IG, DeepLIFT, and Occlusion. We modified the code base to use the score delta ($S_{c_o} - S_{c_t}$) instead of the original class score (S_{c_t}) and allowing replacing the zero baseline by custom baselines from GANMEX and MDTS. Expected gradient was separately implemented according to the formulation in Erion et al. (2019). We set the number of sampling steps to 200 for both IG and Expected Gradient, and used Occlusion-1 that only perturb the pixel itself (as supposed to perturbing the whole neighboring patch of pixels).

The DeepSHAP saliency maps were calculated using SHAP (https://github.com/slundberg/shap). We made similar modification to replace the original class score by the score delta and feed in the custom baseline instances.

In all saliency maps shown in the paper, blue color in indicates positive values and red color indicates negative values. We skipped Occlusion for large images (apple2orange) and also skipped SHAP for full dataset evaluations due to the computation resource constraints.

B. Additional Experiments

B.1. Dependency on the Target Class

In Figure 6, we presented the 1-vs-1 saliency maps generated with different target classes. We expect the feature attributions to be dependent for different target classes. However, 1-vs-1 saliency maps generated using zero baselines show almost the same attributions regardless of the target classes. GANMEX baselines corrected the behavior for IG, DeepLIFT and DeepSHAP, and produced attributions that depend on the target classes.

B.2. Attribution Sparsity

We calculated the Gini Index for measuring the sparseness of the saliency maps as proposed by Chalasani et al. (2018), where a larger score representing sparser saliency map is desired. Figure 7 shows our experiments comparing the different techniques with zero baselines vs. GANMEX as well

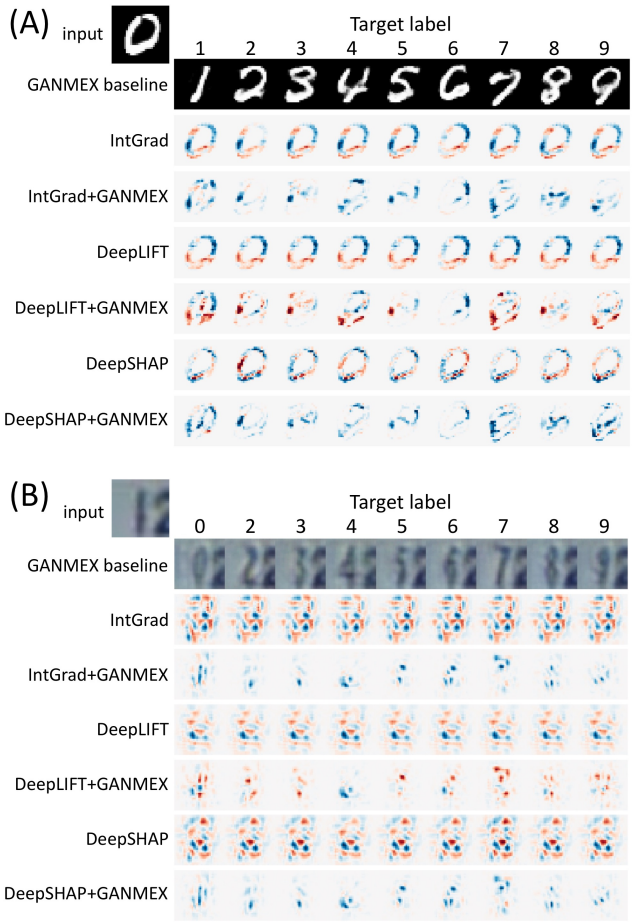

Figure 6: 1-vs-1 saliency maps using class-targeted baselines (GANMEX) vs non-class-targeted baselines (zero baselines).

as expected gradient. Other than IG/SVHN where GANMEX has a visible advantage, the results are roughly the same; we suspect that the sparseness of zero baseline attribution was benefited from incorrectly hiding key features, as shown in Figure 2. Expected gradient, on the other hand, consistently under-performs its counterpart (IG+GANMEX) in both datasets.

B.3. One-vs-all Attributions for Binary Classifiers

In Figure 4 and Figure 8 we compared the saliency map generated by DeepLIFT and IG with the zero input, max input, blurred image, with those generated with the GANMEX baselines on the apple2orange dataset. Conceptually apples and oranges both have round shapes but have different colors, so we would expect the saliency maps on a reasonably performing classifier to highlight the colors of the fruit, but not the shapes, and definitely not the background. With all baselines except GANMEX (Zero, Max, Blur), we commonly observe one of two mistakes in the saliency maps. The first mistake consists of highlighting the background, and the second highlights only the edge of the

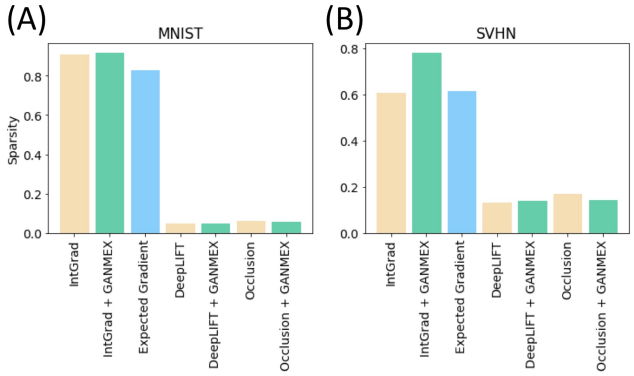

Figure 7: Gini indices, with the yellow bars representing saliency maps with zero baselines and the green bars representing that of GANMEX baselines.

apples providing false indications that the model based its decisions on the object shapes rather than the colors. Neither of these mistakes occur when using the GANMEX baselines as the background is never present and the full shape of the apples/oranges are highlighted.

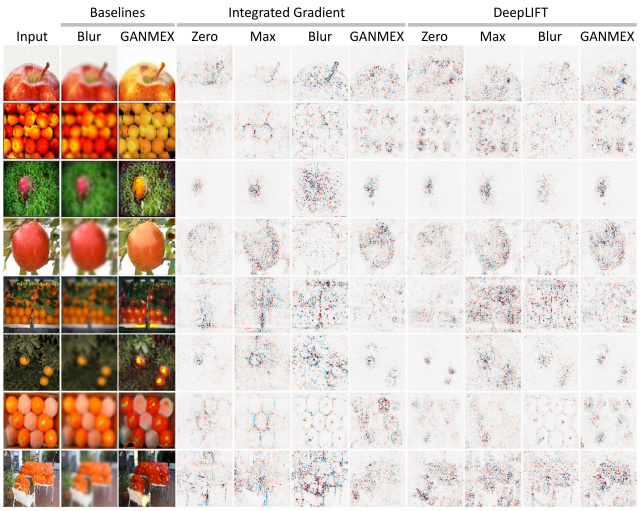

Figure 8: Additional examples of saliency maps for the classifier on the apple2orange dataset with four baseline choices: zero baseline (Zero), maximum value baseline (Max), blurred baseline (Blur), and GANMEX baseline (GANMEX).

B.4. Ablation Studies

Here we analyzed the possibilities of using other GAN models. Different from StarGAN, most other image-to-image translation algorithms do not have a stand-alone class discriminator that can be swapped with a trained classifier. To simulate such restrictions, we trained a similar GAN model but with the class discriminator trained jointly with the generator from scratch. Figure 9.A shows that while the stand-

alone GAN yields similar baseline with GANMEX, both the baselines and saliency maps of the stand-alone GAN remains unperturbed under cascading randomization of the model. This indicates that the class-wise explanations provided by stand-alone GAN were not specific to the to-be-explained classifier.

The importance of the similarity loss in Equation 9 can be demonstrated on a colored-MNIST dataset, where we randomly assigned the digits with one of the three colors $\{\text{red}, \text{green}, \text{blue}\}$, with labels of the instances remain unchanged from the original MNIST labels of $\{0, ..., 9\}$. The classifier was trained with the same model architecture and training process as for MNIST.

The dataset demonstrated different modes (colors in this case) that are irrelevant to the labels, and we would expect the class-targeted baseline for x would be another instance that has the same color as x. Figure 9.B shows that the similarity loss is the crucial component for ensuring that the baseline has the same color with the input. Without the similarity loss, the generated baseline instance can easily have a different color with the original image. The reconstruction loss itself does not provide the same-mode constraint because a mapping of $G(\text{red}) \rightarrow \text{green}$ and $G(\text{green}) \rightarrow \text{red}$ does not get penalized by the reconstruction loss. While the reconstruction loss was not required for GANMEX and the same-mode constraint, we observed that some degrees of reconstruction loss help GANs converge faster.

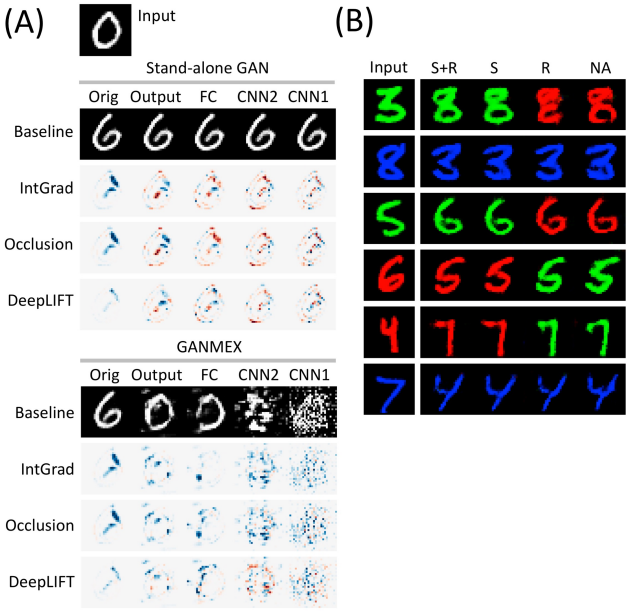

Figure 9: (A) Cascading randomization on baselines generated by a stand-alone GAN lead to little randomization on the saliency maps. (B) Colored-MNIST dataset. GAN baselines generated with both similarity loss and reconstruction loss (S+R), similarity loss only (S), reconstruction loss only (R), and none of those (NA). Only S+R and S successfully constrained the baselines in the same modes (colors) with the inputs.

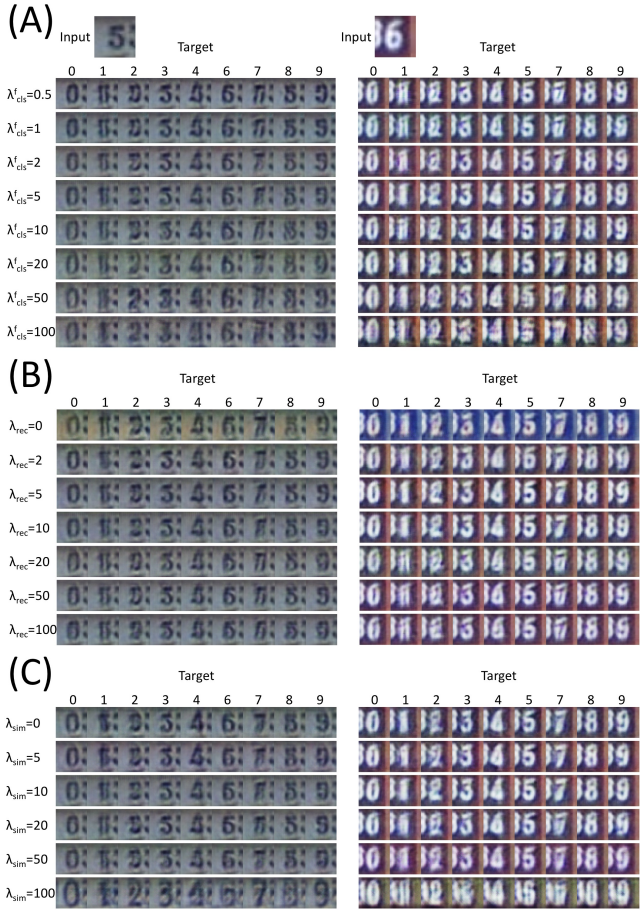

Figure 10: GANMEX baselines generated with various weights for the (A) classification loss, (B) similarity loss, and (C) reconstruction loss.

Hyper-parameter Analysis

We tested how the generated baselines change with respect to the hyperparemeters in the GANMEX loss function. The hyper-parameters, λ^f_{cls}, λ_{rec}, and λ_{sim}, presented in Equation 9 control the degrees of the classification loss, reconstruction loss, and similarity loss, respectively. We performed the hyper-parameter scan on the SVHN dataset as it has enough complex and yet simple enough for visually assessing the attribution.

Classification Loss (λ^f_{cls}) Low classification loss tended to make some transformation unsuccessful, and high classification loss introduced additional noise that make the images unrealistic (Figure 10.A).

Similarity Loss (λ_{rec}) Similarity loss is the key component for minimum distance optimization. As we have shown in Section B.4 and Figure 9.B, at zero similarity loss, the generator is only constraint by the reconstruction loss and can lead to incorrect font colors and background. High similarity loss, on the other hand, makes the baselines to be too similar to the original images (Figure 10.B).

Reconstruction Loss (λ_{sim}) As we have mentioned in

Section B.4 and Figure 9.B, reconstruction loss is not required for GANMEX, but it slightly helps GAN to converged. In contrast, high reconstruction loss can lead to incorrect outputs (Figure 10.C).

References

Chalasani, P.; Chen, J.; Chowdhury, A. R.; Jha, S.; and Wu, X. 2018. Concise Explanations of Neural Networks using Adversarial Training. *CoRR* abs/1810.06583. URL http://arxiv.org/abs/1810.06583.

Choi, Y.; Choi, M.; Kim, M.; Ha, J.; Kim, S.; and Choo, J. 2017. StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation. *CoRR* abs/1711.09020. URL http://arxiv.org/abs/1711.09020.

Erion, G. G.; Janizek, J. D.; Sturmfels, P.; Lundberg, S.; and Lee, S. 2019. Learning Explainable Models Using Attribution Priors. *CoRR* abs/1906.10670. URL http://arxiv.org/abs/1906.10670.

Howard, A. G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; and Adam, H. 2017. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *CoRR* abs/1704.04861. URL http://arxiv.org/abs/1704.04861.

Krizhevsky, A. 2009. Learning multiple layers of features from tiny images. Technical report.

LeCun, Y.; and Cortes, C. 2010. MNIST handwritten digit database URL http://yann.lecun.com/exdb/mnist/.

Netzer, Y.; Wang, T.; Coates, A.; Bissacco, A.; Wu, B.; and Ng, A. Y. 2011. Reading Digits in Natural Images with Unsupervised Feature Learning .

Springenberg, J.; Dosovitskiy, A.; Brox, T.; and Riedmiller, M. 2015. Striving for Simplicity: The All Convolutional Net. In *ICLR (workshop track)*. URL http://lmb.informatik.uni-freiburg.de/Publications/2015/DB15a.

Zhu, J.; Park, T.; Isola, P.; and Efros, A. A. 2017. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. *CoRR* abs/1703.10593. URL http://arxiv.org/abs/1703.10593.