Monotonicity Regularization: Improved Penalties and Novel Applications to Disentangled Representation Learning and Robust Classification - Supplementary material

João Monteiro * **Mohamed Osama Ahmed**[1] **Hossein Hajimirsadeghi**[1] **Greg Mori**[1,2]

[1]Borealis AI
[2]Simon Fraser University

Abstract

We study settings where gradient penalties are used alongside risk minimization with the goal of obtaining predictors satisfying different notions of monotonicity. Specifically, we present two sets of contributions. In the first part of the paper, we show that different choices of penalties define the regions of the input space where the property is observed. As such, previous methods result in models that are monotonic only in a small volume of the input space. We thus propose an approach that uses mixtures of training instances and random points to populate the space and enforce the penalty in a much larger region. As a second set of contributions, we introduce regularization strategies that enforce other notions of monotonicity in different settings. In this case, we consider applications, such as image classification and generative modeling, where monotonicity is not a hard constraint but can help improve some aspects of the model. Namely, we show that inducing monotonicity can be beneficial in applications such as: (1) allowing for controllable data generation, (2) defining strategies to detect anomalous data, and (3) generating explanations for predictions. Our proposed approaches do not introduce relevant computational overhead while leading to efficient procedures that provide extra benefits over baseline models.

A ILLUSTRATIVE EXAMPLES ON THE SPHERE: MIXUP HELPS TO POPULATE THE SMALL VOLUME INTERIOR REGION

To further illustrate the issue discussed in the item 2 of Section 3.1 as well the effect of our proposal, we discuss a simple example considering random draws from the unit n-sphere, shown in Figure 1, i.e., the set of points $\mathcal{B} = \{x \in \mathbb{R}^n : ||x||_2 < 1\}$. We further consider a concentric sphere of radius $0 < r < 1$ given by $\mathcal{B}_r = \{x \in \mathbb{R}^n : ||x||_2 < r\}$. We are interested in the probability of a random draw from $\mathcal{B}$ to lie outside of $\mathcal{B}_r$, i.e.: $P(||x||_2 > r), x \sim \mathcal{D}(\mathcal{B})$, for some distribution $\mathcal{D}$. We start by defining $\mathcal{D}$ as the Uniform($\mathcal{B}$), which results in $P(||x||_2 > r) = 1 - r^n$. In Figure 2a, we can see that for growing n, $P(||x||_2 > r)$ is very large even if $r \approx 1$, which suggests most random draws will lie close to $\mathcal{B}$'s boundary.

We now evaluate the case where mixup is applied and random draws are taken in two steps: we first observe $y \sim$ Uniform($\mathcal{B}$), and then we perform mixup between y and the origin[1], i.e., $x = \lambda y$, $\lambda \sim$ Uniform($[0, 1]$). In this case, $P(||x||_2 > r) = (1 - r^n)(1 - r)$, which is shown in Figure 2b as a function of r for increasing n. We can then observe that even for large n, $P(||x||_2 > r)$ decays linearly with r, i.e., we populate the interior of $\mathcal{B}$ and x in this case follows a non-uniform distribution such that its norms histogram is uniform.

*Work done while interning at Borealis AI. Currently at ServiceNow

[1]Similar conclusions hold for any fixed point within $\mathcal{B}$. The origin is chosen for convenience.

Accepted for the 38th Conference on Uncertainty in Artificial Intelligence (UAI 2022).

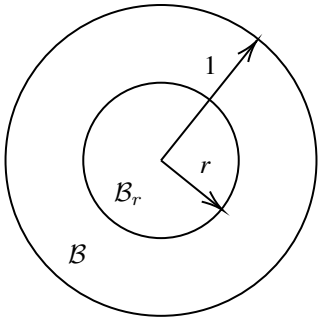
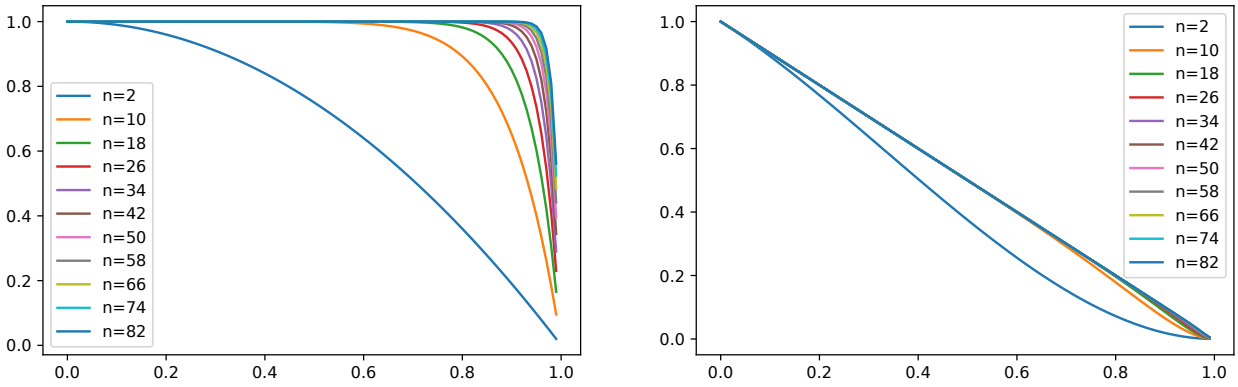


Figure 1: Illustration unit spheres $\mathcal{B}$ and $\mathcal{B}_r$ on the plane.



(a) $P(||x||_2 > r)$ as a function of r for various n and $x \sim$ Uniform($\mathcal{B}$).

(b) $P(||x||_2 > r)$ as a function of r for various n. In this case, $x = \lambda y$, $\lambda \sim$ Uniform($[0, 1]$), $y \sim$ Uniform($\mathcal{B}$).

Figure 2: Illustrative example showing that uniformly distributed draws on a unit sphere in $\mathbb{R}^n$ concentrate on its boundary for large n. Applying mixup populates the interior of the space.

B PROOF-OF-CONCEPT EVALUATION

We start by describing the approach we employ to generate data containing the properties required by our evaluation. Denote a design matrix by $X_{N\times D}$ such that each of its N rows corresponds to a feature vector within $\mathbb{R}^D$. In order to ensure the data lies in some manifold, we first obtain a low-dimensional synthetic design matrix given by $X'_{N\times d}$, where each entry is sampled randomly from Uniform($[-10, 10]$). We then expand it to $\mathbb{R}^D$ by applying the following transformation:

$$X = X'A, \tag{1}$$

where the expansion matrix given by $A_{d\times D}$ is such that each of its entries are independently drawn from Uniform($[0, 1]$). Throughout our experiments, $d = \lfloor 0.3D \rfloor$ was employed.

Target values for the function f to be approximated are defined as sums of functions of scalar arguments applied independently over each dimension. We thus select a set of dimensions $M \in [D]$ with respect to which f is to be monotonic, i.e.:

$$f(x) = \sum_{i\in M} g_i(x_i) + \sum_{j\in \bar{M}} h_j(x_j), \tag{2}$$

and every $g_i : \mathbb{R} \mapsto \mathbb{R}$ is increasing monotonic, while every $h_i : \mathbb{R} \mapsto \mathbb{R}$ is not monotonic.

We then create two evaluation datasets. One of them, referred to as the validation set, is identically distributed with respect to X since it is obtained following the same procedure discussed above. In order to simulate covariate-shift, we create a test

set by changing the expansion matrix A to a different one.

$$X_{val} = X'_{val}A, \quad X_{test} = X'_{test}A_{test}, \tag{3}$$

where A_{test} will be given by entry-wise linear interpolations between A, used to generate the training data, and a newly sampled expansion matrix A': $A_{test} = \alpha A' + (1 - \alpha)A$. The parameter $\alpha \in [0, 1]$, set to 0.8 in the reported evaluation, controls the shift between A_{test} and A_{test} in terms of the Frobenius norm, which in turn enables the control of how much the test set shifts relative to the training data.

We thus trained models to approximate f for spaces of increasing dimensions as well as for an increasing number of dimensions with respect to which f is monotonic. Results are reported in Table 1 in terms of RMSE on the two evaluation datasets, and in terms of monotonicity in Table 2 where $\hat{\rho}$ is computed both on random points and on the shifted test set. Entries in the tables correspond to the centers of 95% confidence intervals resulting from 20 independent training runs.

We highlight the two following observations regarding the prediction performances shown in table 1: different models present consistent performances across evaluations, which suggests different monotonicity-enforcing penalties do not significantly affect prediction accuracy. Moreover, the proposed approach used to generate test data under covariate-shift is effective given the gap in performance consistently observed between the validation and the test partitions. In terms of monotonicity, results in Table 2 suggest that Ω_{random} and Ω_{train} are only effective on either random or data points, which seems to aggravate when the dimension D grows. Ω_{mixup}, on the other hand, is effective on both sets of points, and continues to work well for growing D. Furthermore, covariate-shift significantly affects Ω_{train} for higher-dimensional cases, while Ω_{mixup} performs well in such a case.

$\lvert M\rvert/D$	*20/100*		*40/200*		*80/400*		*100/500*	
	Valid. RMSE	Test RMSE	Valid. RMSE	Test RMSE	Valid. RMSE	Test RMSE	Valid. RMSE	Test RMSE
Non-mon.	0.007	0.107	0.006	0.082	0.007	0.087	0.011	0.146
Ω_{random}	0.008	0.117	0.006	0.081	0.007	0.093	0.012	0.125
Ω_{train}	0.008	0.115	0.006	0.086	0.007	0.089	0.012	0.134
Ω_{mixup}	0.008	0.114	0.007	0.084	0.008	0.088	0.012	0.134

Table 1: Prediction performance of models trained on generated data in spaces of growing dimension (D) and number of monotonic dimensions ($|M|$). Different regularization strategies do not affect prediction performance. The performance gap consistently observed across the evaluation sets highlights the shift between the two sets of points. The lower the values of RMSE the better.

$\lvert M\rvert/D$	*20/100*		*40/200*		*80/400*		*100/500*	
	$\hat{\rho}_{random}$	$\hat{\rho}_{test}$	$\hat{\rho}_{random}$	$\hat{\rho}_{test}$	$\hat{\rho}_{random}$	$\hat{\rho}_{test}$	$\hat{\rho}_{random}$	$\hat{\rho}_{test}$
Non-mon.	99.90%	99.99%	97.92%	94.96%	98.47%	96.56%	93.98%	90.01%
Ω_{random}	0.00%	3.49%	0.00%	4.62%	0.01%	11.36%	0.02%	19.90%
Ω_{train}	1.30%	0.36%	4.00%	0.58%	9.67%	0.25%	9.25%	5.57%
Ω_{mixup}	0.00%	0.35%	0.00%	0.44%	0.00%	0.26%	0.00%	0.42%

Table 2: Fraction of monotonic points $\hat{\rho}$ for models trained on generated data in spaces of growing dimension (D) and number of monotonic dimensions ($|M|$). Different regularization strategies is effective on only one of $\hat{\rho}_{random}$ or $\hat{\rho}_{test}$, while Ω_{mixup} seems effective throughout conditions. The lower the values of $\hat{\rho}$ the better.

C MODELS AND TRAINING DETAILS FOR EXPERIMENTS REPORTED IN SECTION 4

For the case of CIFAR-10, WideResNets [Zagoruyko and Komodakis, 2016] are used. The models are initialized randomly and trained both with and without the monotonicity penalty. Standard stochastic gradient descent (SGD) implements the parameters update rule with a learning rate starting at 0.1, being decreased by a factor of 10 on epochs 10, 150, 250, and 350. Training is carried out for a total of 600 epochs with a batch size of 64. For ImageNet, on the other, training consists of fine tuning a pre-trained ResNet-50, where the fine-tuning phase included the monotonicity penalty. We do so by training the model for 30 epochs on the full ImageNet training partition. In this case, given that the label set $\mathcal{Y}$ is relatively large, using the standard ResNet-50 would result in small slices S_k. To avoid that, we add an extra final convolution layer with

Model	$\arg\max_{k \in \mathcal{Y}} h(x)_k$	$\arg\max_{k \in \mathcal{Y}} T_k(x)$
	10%	
WideResNet	85.68%	16.35%
*Mono*WideResNet	85.77%	82.21%
	30%	
WideResNet	92.12%	14.51%
*Mono*WideResNet	92.42%	88.88%
	60%	
WideResNet	94.51%	10.08%
*Mono*WideResNet	94.86%	93.81%

Table 3: Top-1 accuracy obtained by both standard and group monotonic models on sub-samples of CIFAR-10. Predicition performance obtained by classifiers defined by the total activations is upper bounded by the performance obtained at the output layer for monotonic models.

$W = 15K$. Training is once more carried out with SGD using a learning rate set to 0.001 in this case, and reduced by a factor of 5 at epoch 20. In both cases, the group monotonicity property is enforced at the last convolutional layer. Other hyperparameters such as the strength γ of the monotonicity penalty as well as the inverse temperature μ used to compute Ω_{group} are set to 1 and 50 for the case of CIFAR-10, and to 5 and 10 for the case of ImageNet. Both momentum and weight decay are further employed and their corresponding parameters are set to 0.9 and 0.0001. For MNIST classifiers, training is performed for 20 epochs using a batch size of 64 and the Adadelta optimizer [Zeiler, 2012] with a learning rate of 1.

D ENFORCING GROUP MONOTONICITY UNDER SMALL SAMPLES

Using CIFAR-10, we further evaluate how the proposed group monotonicity penalty behaves in data-constrained settings, i.e., we check whether or not the property can be enforced under small sample regimes. We do so by sub-sampling the original training data by randomly selecting a fraction of the training images uniformly across classes. We then train the same WideResNet for the same computation budget in terms of number of iterations as the models trained in the complete set of images. The learning rate schedule also matches that of the training on the full dataset in that the learning rate is reduced at exactly the same iterations across all training cases. Results are reported in Table 3 for sub-samples corresponding to 10%, 30%, and 60% of CIFAR-10. Results are consistent across the three sets of results in showing that predictions obtained from the total activation of feature slices approximate the prediction performance of the underlying model for the case of group monotonic predictors, i.e., the extent to which the underlying model is able to accurately predict correct classes upper bound the resulting "level of monotonicity". In simple terms, the better the classifier, the more group monotonic it can be made.

E SELECTING FEATURE MAPS TO COMPUTE VISUAL EXPLANATIONS

Approaches based on Class Activation Maps (CAM) such as Grad-CAM and its variations [Selvaraju et al., 2017, Chattopadhay et al., 2018] seek to extract *explanations* from convolutional models. By explanation we mean to refer to indications of properties of the data implying the predictions of a given model. Under such a framework, one can obtain so-called explanation heat-maps through the following steps: (1) Compute a weighted sum of activations of feature maps in a chosen layer; (2) Upscale the results in order to match the dimensions of the input data; (3) Superimpose results onto the input data. Specifically for the case of applications to image data, following those steps results in highlighting the patches of the input that were deemed relevant to yield the observed predictions. Different approaches were then introduced in order to define the weights used in the first step. A very common choice is to use the total gradient of the output corresponding to the prediction with respect to activations of each feature map.

For the case of group monotonic classifiers, we are interested in verifying whether one can define useful explanation heat-maps by considering only the feature slices corresponding to the predicted class, i.e., for a given input pair (x, y), we compute explanation heat-maps considering only its corresponding feature activation slice $S_y(x)$. We thus design an experiment to evaluate the effectiveness of such an approach by using external auxiliary classifiers to perform predictions from test data that was occluded using explanation heat-maps obtained using different models and sets of representations. In other words, we use the explanation maps to remove from the data the parts that were not indicated as relevant. We then assume that good explanation maps will be such that classifiers are able to correctly classify occluded data since relevant

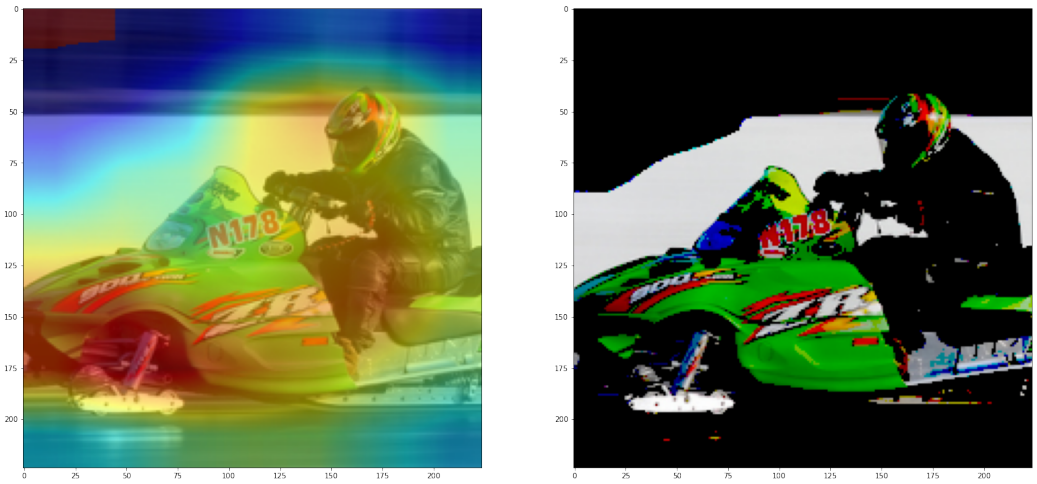


Figure 3: Example of explanation heat-map and corresponding occlusion obtained with Grad-CAM and a ResNet-50 trained on ImageNet. The example belongs to the validation set and corresponds to the class *snowmobile*.

Model (h)	Aux. classifier			
	ResNext-50	MobileNet-v3	VGG-16	SqueezeNet
Reference perf.	77.62%	74.04%	71.59%	58.09%
ResNet-50	72.94%	68.31%	67.34%	49.95%
*Mono*ResNet-50	72.88%	68.75%	66.99%	48.92%
*Mono*ResNet-50 (Constrained)	72.44%	66.55%	66.92%	45.83%

Table 4: Top-1 accuracy of auxiliary classifiers evaluated on data created by occluding patches deemed irrelevant by explanation heat-maps given by different models. The performance of monotonic classifiers when constrained to consider only the feature maps within the slice corresponding to their prediction is further reported and shown to closely math the performance of cases where the full set of features is considered.

patches are conserved. In further details, occlusions are computed by first applying a CAM operator given a model h and data x, which results in a heat-map with entries in $[0, 1]$. We then use such a heat-map as a multiplicative mask to get an occluded version of x, denoted x', i.e.:

$$x' = \text{CAM}(x, h) \circ x, \tag{4}$$

where the operator $\circ$ indicates element-wise multiplication. An example of such a procedure is shown in Figure 3. We apply the above procedure to all of the validation data, and use resulting points to then assess the prediction performance of auxiliary classifiers.

Explanation maps are computed using the same models discussed in Section 4.2.1 for ImageNet. The CAM operator corresponds to a variation of Grad-CAM++ [Chattopadhay et al., 2018] where the model activations are directly employed for weighing feature maps rather than the gradients. We consider 4 auxiliary pre-trained classifiers corresponding to ResNext-50 [Xie et al., 2017], MobileNet-v3 [Howard et al., 2019], VGG-16 [Simonyan and Zisserman, 2014], and SqueezeNet [Iandola et al., 2016]. Results are reported in Table 4 which also include the reference performance of the auxiliary classifiers on the standard validation set in order to provide an idea of the gap in performance resulting from removing parts of test images via occlusion. We highlight the performance reported in the last row of the Table. In that case, explanation maps for the group monotonic model are computed from only the features of the class slice, which is enough to match the performance of a standard ResNet-50 with full access to the features. This suggests that representations learned by group monotonic models are such that all the information required to explain a given class is contained in the slice reserved for that class.

F EXAMPLES OF EXPLANATION HEAT-MAPS AND OCCLUDED DATA

In Figure 4, we show examples of explanation heat-maps obtained using different approaches. Corresponding occlusions resulting from the different approaches are shown in 5.

G ANALYSIS OF COLOR SEQUENCES FOR GENERATED DATA

We performed a set of experiments in order to evaluate whether some kind of ordering could be observed once we generate data for increasing values of z, specifically on dimensions that correspond to colors. To do that, we created an increasing sequence of values by defining a uniform grid in $[0, 1]$ with 50 steps. We then encoded a particular image, but decoded latent vectors after substituting the z value in the dimension corresponding to *floor color* by the values in the sequence.

Generated sequences of images are shown in Figures 7 and 8 for the base and monotonic models, respectively. In each such a case, we plot the images on the left, and bottom-left patches of size 10x10 so as to highlight the color sequences that we observe with such an approach. Surprisingly, we observed that monotonic models tend to generate colors in a sequence that matches the HUE circle for RGB images, represented in Figure 6 for reference. Besides visually verifying that to be the case across a number of generated examples, in Table 3 in Section 4.1 we check the fraction of the dataset where such sequences of patches are sorted in terms of their HUE angles.

H EXAMPLES OF DATA GENERATED WITH STANDARD AND MONOTONIC MODELS

We illustrate data generated for linear trajectories in the latent space of standard and monotonic models. To do that, we start from a fixed image, and modify one generative factor at a time. We then generate images by feeding the decoder with points in the linear trajectory between the outputs of the encoder for the pair of images. Generated data for each modified factor are shown in Figures 9, 10, 11, 12, 13, and 14.

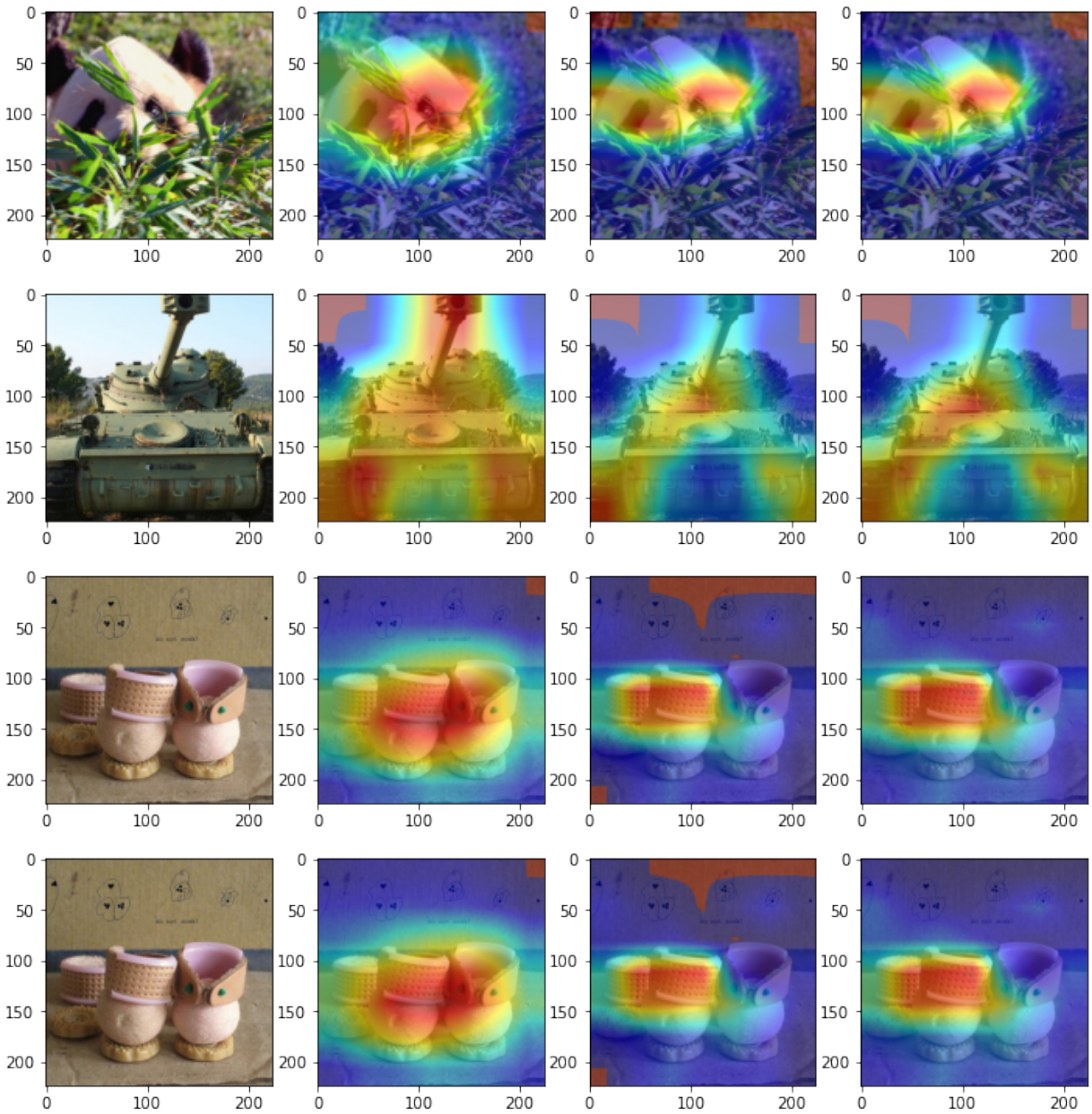


Figure 4: Examples of explanation heat-maps superimposed onto images. From left to right we have the original image, results obtained from a ResNet-50, a *mono*ResNet-50, and a *mono*ResNet-50 where the CAM operator only access the slice corresponding to the underlying class. All are obtained with Grad-CAM.

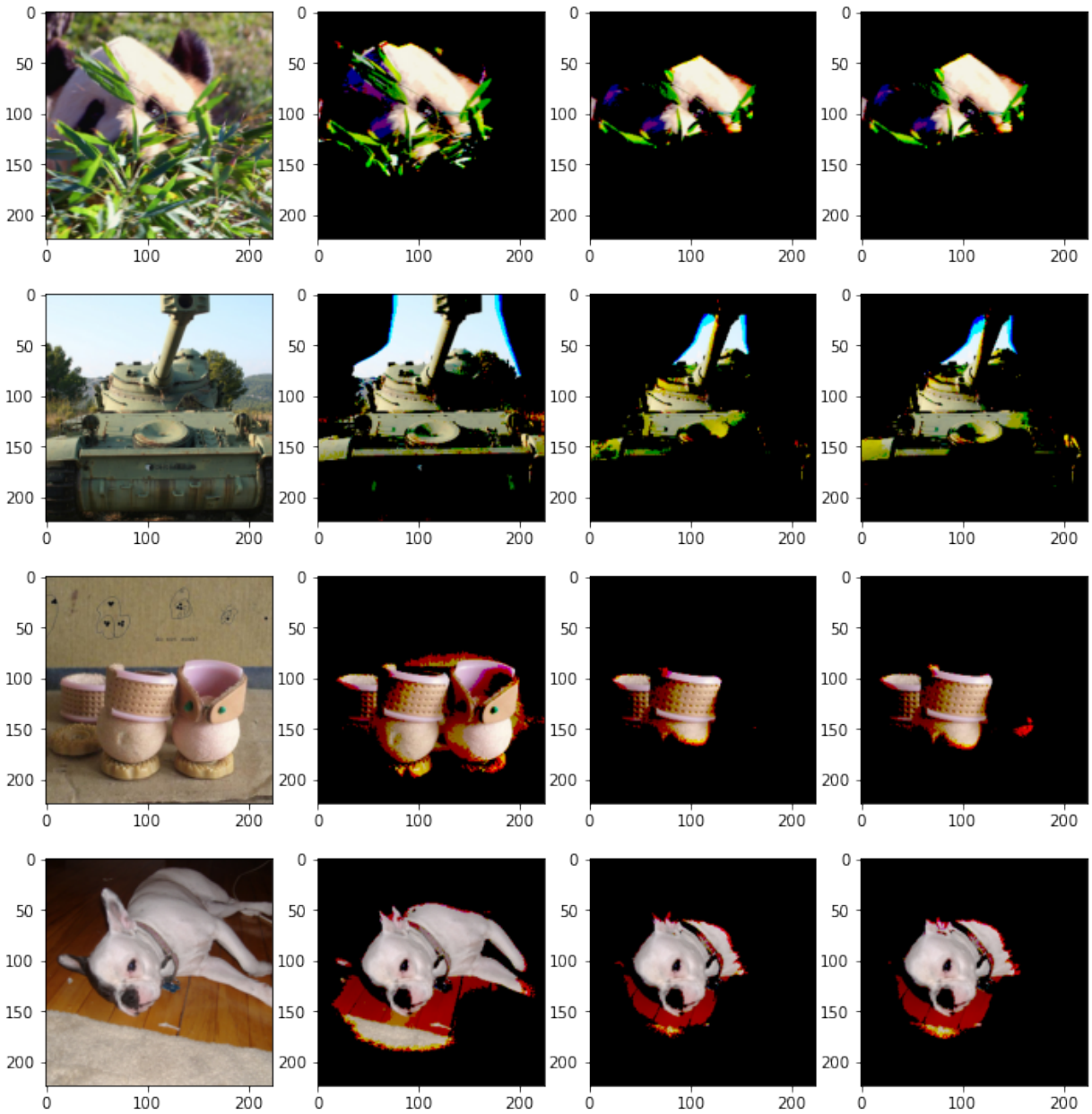


Figure 5: Examples of occluded data using explanation heat-maps. From left to right we have the original image, results obtained from a ResNet-50, a *mono*ResNet-50, and a *mono*ResNet-50 where the CAM operator only access the slice corresponding to the underlying class. All are obtained with Grad-CAM.

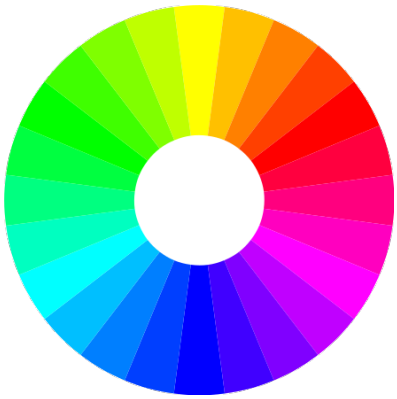
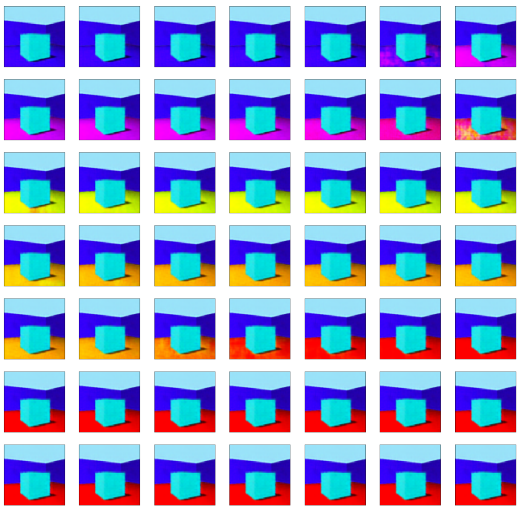
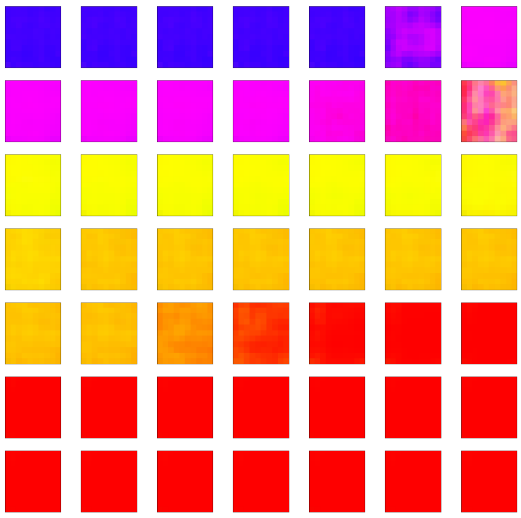


Figure 6: HUE circle of RGB images. Original image from: `https://en.wikipedia.org/wiki/Hue`.

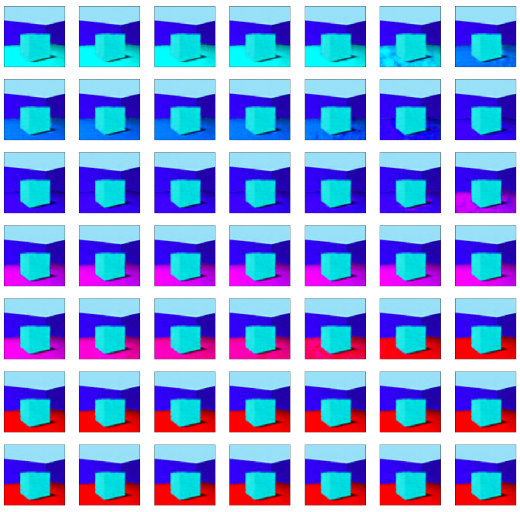


(a) Data for increasing values for the latent dimension associated to *floor color*.

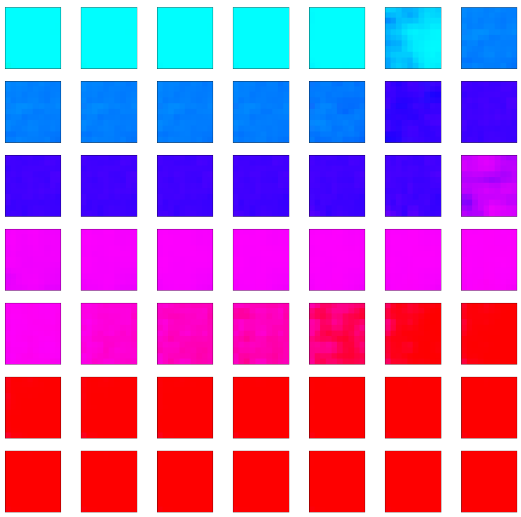


(b) Bottom-left 10x10 patches of generated images.

Figure 7: Data generated by *standard model* for traversals of z on the dimension corresponding to *floor color*

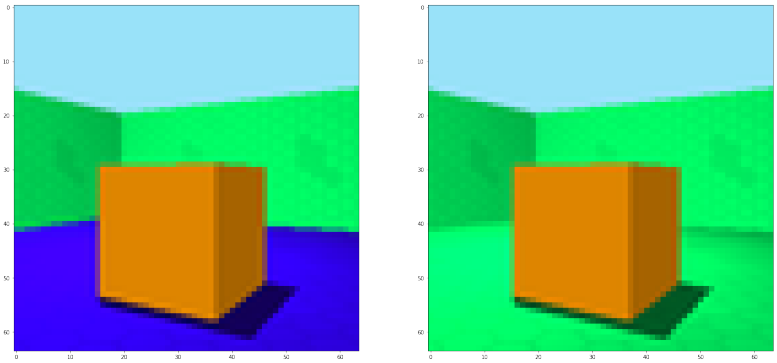


(a) Data for increasing values for the latent dimension associated to *floor color*.

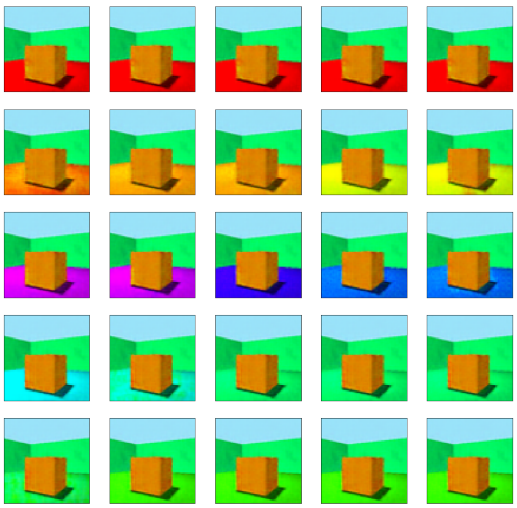


(b) Bottom-left 10x10 patches of generated images.

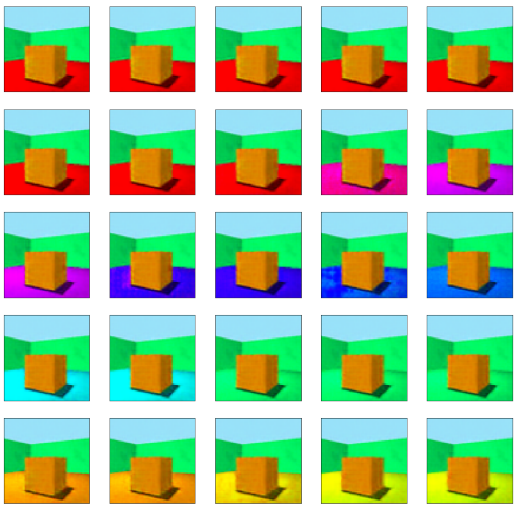
Figure 8: Data generated by *monotonic model* for traversals of z on the dimension corresponding to *floor color*



(a) Input pair.

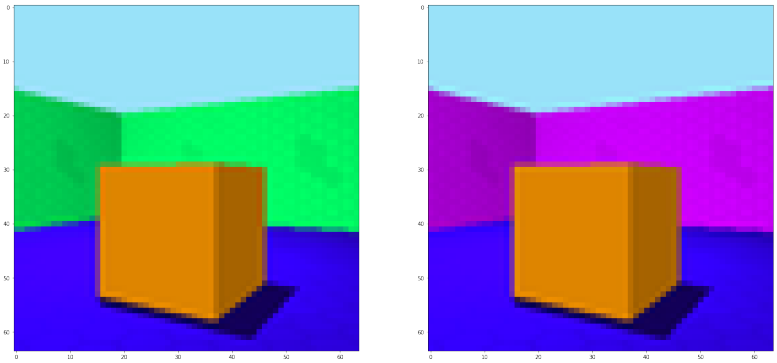


(b) Data generated by standard model.

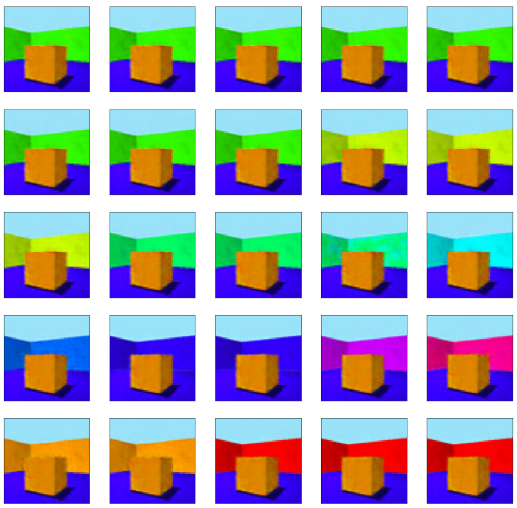


(c) Data generated by monotonic model.

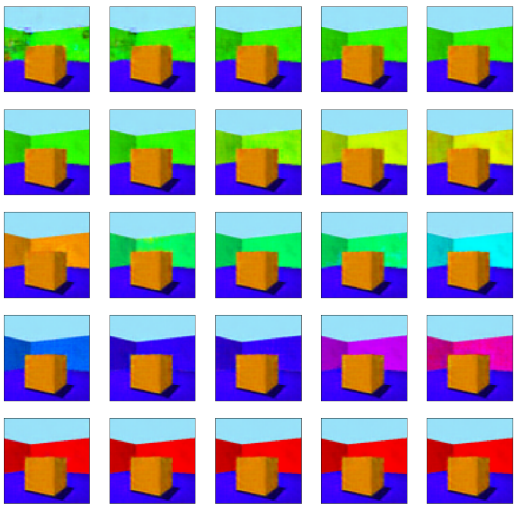
Figure 9: Generating data by moving along the line passing over latent representation for inputs for which a single factor is different. Generative factor changing: floor color.



(a) Input pair.

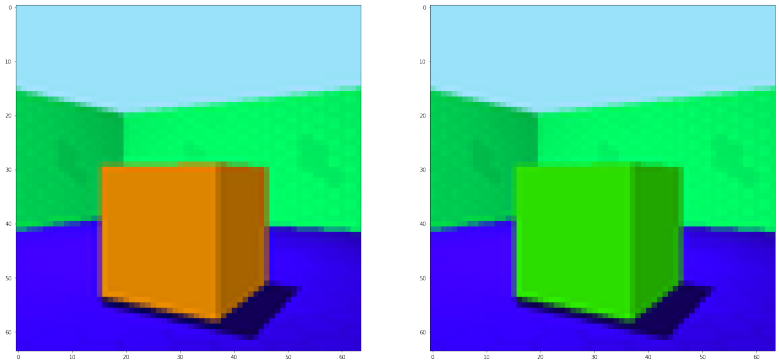


(b) Data generated by standard model.

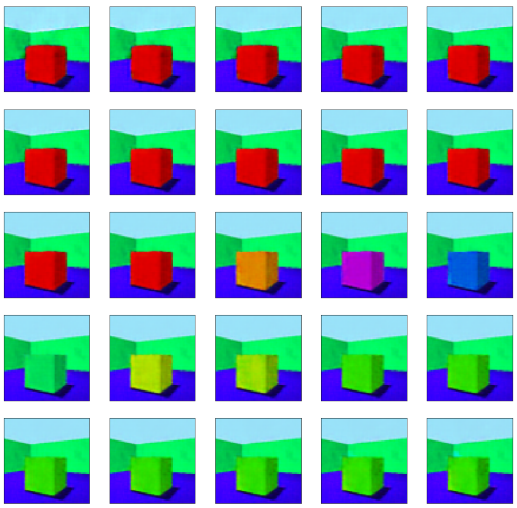


(c) Data generated by monotonic model.

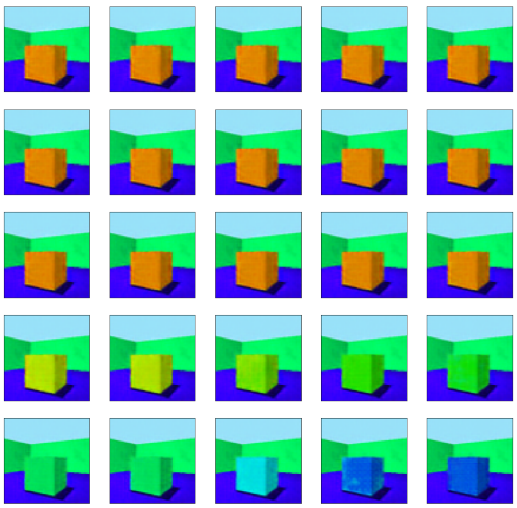
Figure 10: Generating data by moving along the line passing over latent representation for inputs for which a single factor is different. Generative factor changing: wall color.



(a) Input pair.

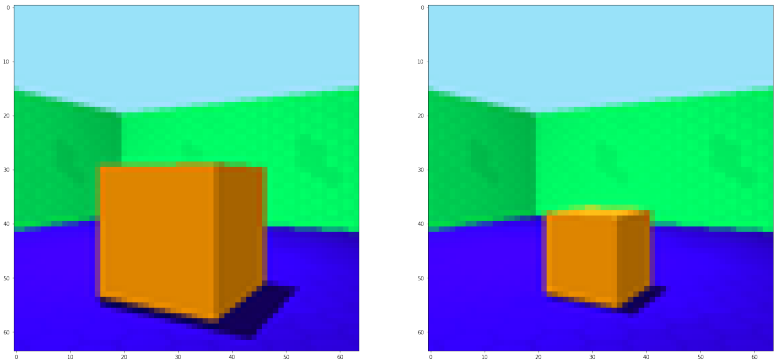


(b) Data generated by standard model.

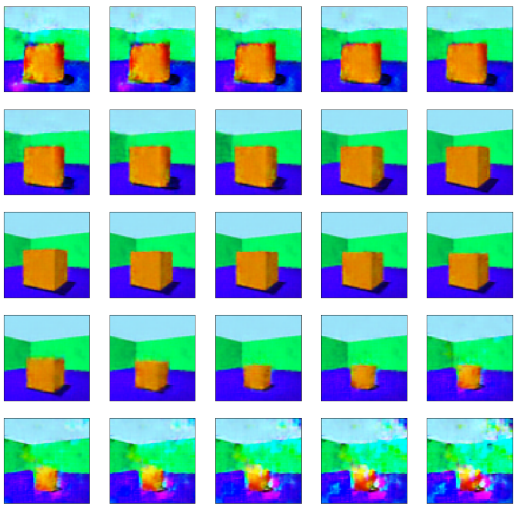


(c) Data generated by monotonic model.

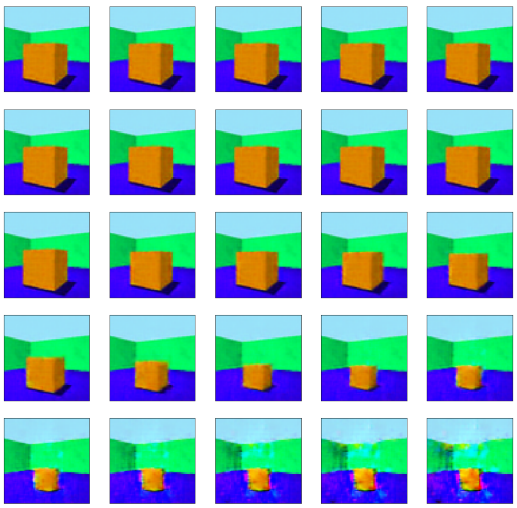
Figure 11: Generating data by moving along the line passing over latent representation for inputs for which a single factor is different. Generative factor changing: object color.



(a) Input pair.

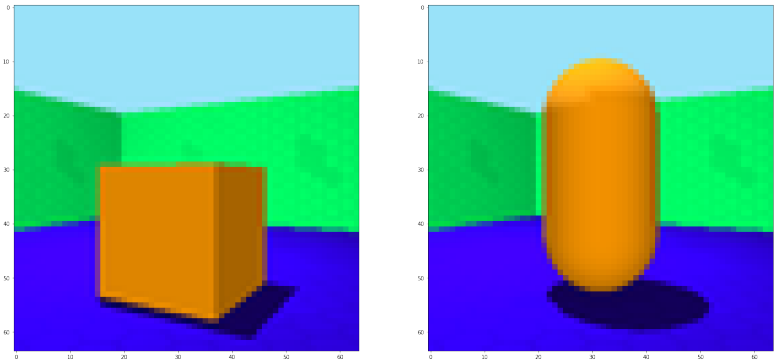


(b) Data generated by standard model.

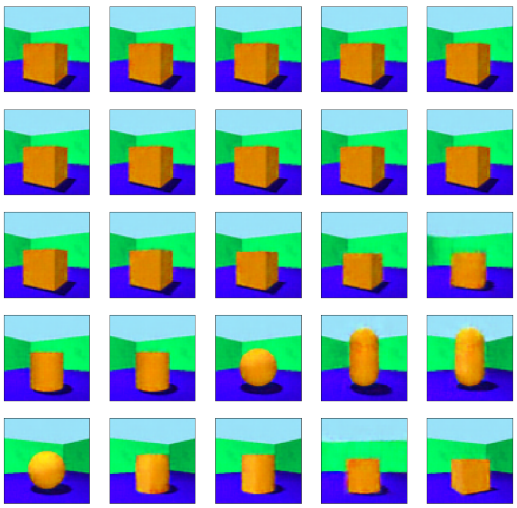


(c) Data generated by monotonic model.

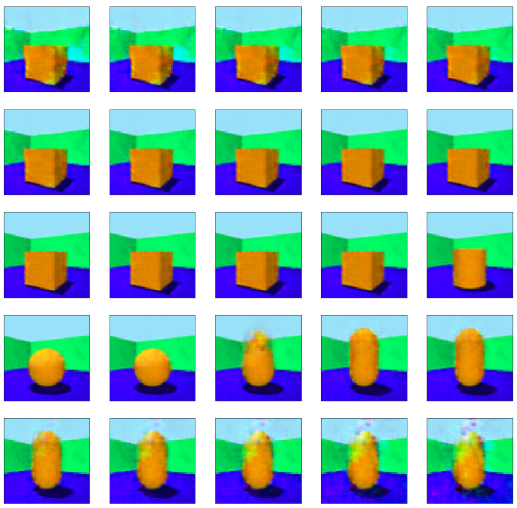
Figure 12: Generating data by moving along the line passing over latent representation for inputs for which a single factor is different. Generative factor changing: scale.



(a) Input pair.

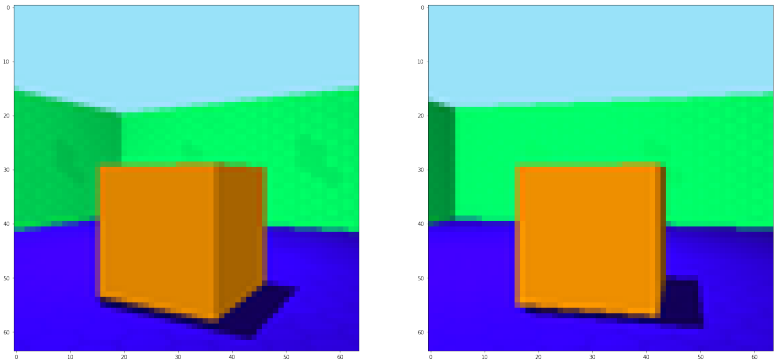


(b) Data generated by standard model.

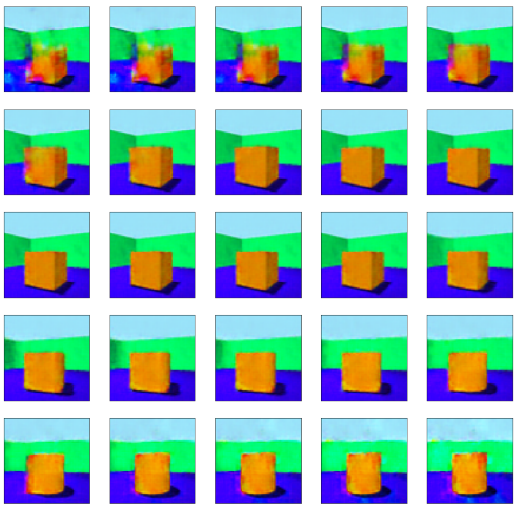


(c) Data generated by monotonic model.

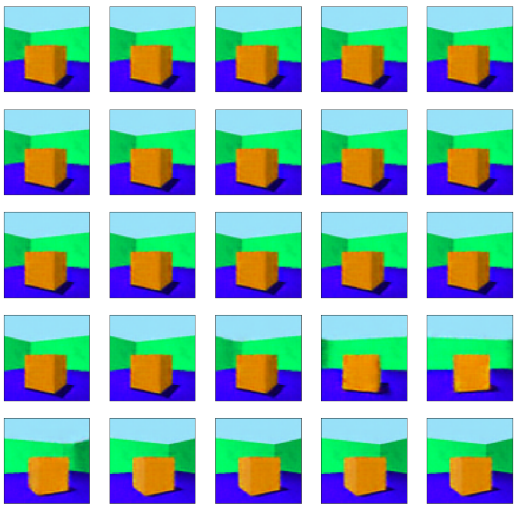
Figure 13: Generating data by moving along the line passing over latent representation for inputs for which a single factor is different. Generative factor changing: shape.



(a) Input pair.



(b) Data generated by standard model.



(c) Data generated by monotonic model.

Figure 14: Generating data by moving along the line passing over latent representation for inputs for which a single factor is different. Generative factor changing: orientation.

References

Aditya Chattopadhay, Anirban Sarkar, Prantik Howlader, and Vineeth N Balasubramanian. Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks. In *2018 IEEE winter conference on applications of computer vision (WACV)*, pages 839–847. IEEE, 2018.

Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1314–1324, 2019.

Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

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.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.