Efficient Verification of Neural Networks against LVM-based Specifications

Harleen Hanspal
Imperial College London & Safe Intelligence
London, United Kingdom
h.hanspal21@imperial.ac.uk

Alessio Lomuscio
Safe Intelligence
London, United Kingdom
alessio@safeintelligence.ai

Abstract

The deployment of perception systems based on neural networks in safety critical applications requires assurance on their robustness. Deterministic guarantees on network robustness require formal verification. Standard approaches for verifying robustness analyse invariance to analytically defined transformations, but not the diverse and ubiquitous changes involving object pose, scene viewpoint, occlusions, etc. To this end, we present an efficient approach for verifying specifications definable using Latent Variable Models that capture such diverse changes. The approach involves adding an invertible encoding head to the network to be verified, enabling the verification of latent space sets with minimal reconstruction overhead. We report verification experiments for three classes of proposed latent space specifications, each capturing different types of realistic input variations. Differently from previous work in this area, the proposed approach is relatively independent of input dimensionality and scales to a broad class of deep networks and real-world datasets by mitigating the inefficiency and decoder expressivity dependence in the present state-of-the-art.

1. Introduction

The deployment of perception systems based on neural networks in safety-critical applications requires assurance on their performance, notably accuracy and robustness. Formal verification contributes to this requirement by providing provable and deterministic guarantees that a network meets a given *specification*. Typically, specifications are mathematically expressed constraints on the network's intended input/output and may encode desirable properties, such as robustness to noise (including adversarial attacks) [35], geometric changes [1,2], bias-field changes [14] and beyond.

While the above is useful, practical applications require robustness against diverse changes in a scene, including changes in the pose of objects, viewpoints, occlusions, etc. Such changes cannot be efficiently mathematically defined, but may be encoded from data by using generative models. For instance, [11, 12, 28, 34] use generative models to generate novel in-domain images for data augmentation, adversarial training or evaluating network generalisation; [34] additionally derives formal conditions for a latent space set to necessarily contain sufficient perturbations for it to be trusted for adversarial training and robustness checks. All these approaches either provide statistical robustness measures, or generate attacks based on gradient-search, which is not guaranteed to find an attack if one exists.

Popular for network robustification and empirical evaluation, latent space sets are seldom used as inputs for verification due to the valid concern over the lack of mathematical guarantees on the completeness of the specifications they encode. Therefore, we reiterate that this work is most useful for changes that are difficult to mathematically define. We additionally argue that formal verification of latent space-based specifications can be more valuable than their empirical evaluation. This is because the latent space is a continuous domain and countably infinite number of inputs can be mapped to and reconstructed from a latent space set. Therefore, no amount of testing, or search in the latent space can provide guarantees against all the variations encoded in it. To the best of our knowledge, only [20,27] encode specifications in a latent space and propose architectures to verify them.

There are, however, two difficulties with verification in the latent space. The first concerns the scalability of verification methods; the second relates to the quality of reconstructions affecting the verification outcomes. In this paper, we focus on alleviating these two concerns. Specifically, we propose a novel, invertible encoder-based pipeline for verifying latent space sets, that lends two key benefits of:

- Computational efficiency and relative independence to input dimensionality,
- Verification outcomes' independence to reconstructions, and precise counterexamples with high recall.

We focus our analysis on pose and attribute variations in vision inference tasks, but the approach is likely extendable

to other variations, domains and tasks. Next, we recall key notions for network verification and discuss the existing relevant work, before presenting and validating our method in subsequent sections.

2. Background and Related Work

Neural network verification. Given a network N and an input-output specification $(X_{\text{des}}, Y_{\text{des}})$, the network verification problem is the decision problem of determining whether

$$N(x) \in Y_{\text{des}}, \forall x \in X_{\text{des}}.$$

Verification methods, such as those based on SMT [19], MILP [4, 10, 22], Interval Propagation [15, 16, 32, 33], SDP [3, 29] or Lipschitz analysis [39], propagate a *set* X_{des} in a single forward pass through N and determine whether or not the output set of N for X_{des} lies in Y_{des}. These methods mostly support input specifications defined in:

- *Data space*, such that X_{des} is a norm ball around an input x, i.e., $\mathcal{B}_\epsilon(x) = \{x' \mid \|x' - x\| < \epsilon\}$. The norm is defined in metric spaces such as ℓ_p [2, 16, 19] or Wasserstein [36], and the ϵ is adequately set to include intended photometric and geometric variations, albeit along with considerable unrealistic instances in case of large variations.
- *Parameter space*, such that, given an input x and a set of transformations T, X_{des} is defined by prepending parametric transform layers $N_T(\alpha)$, where α determines the transformation extent, to the network; $X_{\text{des}} = N_T(\alpha)(x)$. This definition is more precise in capturing planar geometric changes [26] and photometric changes, such as smooth illumination variations [14], in an input.

While these specifications are important, they do not efficiently capture non-planar and semantic transformations.

LVM-based specifications. Latent Variable Models (LVMs) for data generation map a typically lower-dimensional, continuous latent distribution to the input space. This mapping and its converse is learnt by LVM's decoder-encoder pair. Additionally, conditional training can disentangle latent dimensions to precisely capture *specific* dataset features and variations [5, 7, 23]. We refer to specifications which use LVM encoders to map input variations and define query input sets as LVM-based specifications.

Related work. As mentioned in Section 1, most formal verification approaches do not use generative models. Only [20, 27] use an autoencoder in a verification setup to prove whether or not the output of a network N is consistent for the image set produced by an *independent* autoencoder (e, d) from the interpolations in its latent space. Formally, [27] establishes whether

$$(N \circ d) \circ e(x) \in Y_{\text{des}}, \forall x \in \{X \mid e(X) \in Z_{\text{des}}\},$$

where $Z_{\text{des}} \in \mathbb{R}^{\text{latent\_dim}}$ is a segment joining latent encodings of samples in X_{des}. Since propagating Z_{des} through a useful decoder is computationally expensive, deterministic verification using this approach is reported to not scale beyond small networks. To overcome this, the authors a) heuristically subsume and approximate line segments by axes-aligned boxes, and b) turn specifications into probabilistic statements that determine whether a vector sampled uniformly from Z_{des} generates a correctly classified image.

The general approach of prepending a generative model as an independent input set generator for verifying a network, which we henceforth refer to as the Encoder-Decoder-Network (EDN) approach, suffers from the following limitations:

1. The network is verified only against images appropriately mapped by the encoder, but the encoding could be *loose* ($z \in Z_{\text{des}} \implies e^{-1}(z) \in X_{\text{des}}$) and *incomplete* ($x \in X_{\text{des}} \implies e(x) \in Z_{\text{des}}$).
2. The network is verified against reconstructions of the decoder which may not have the same quality as the dataset images in terms of sharpness and diversity. If the decoder does not reconstruct an intended attribute's variations, either because all variations of that attribute are encoded to the same latent vector, or because it is not expressive enough, verification may incorrectly suggest that the network is invariant to that attribute.
3. The input set Z_{des} is propagated through both the decoder and the network to be verified, thereby resulting in considerable computational cost.

In the rest of the paper, we propose a pipeline for verifying LVM-based specifications in a computationally lighter manner, and alleviate the last two shortcomings by construction. We do not solve the first limitation; but in Section 5.4, we present metrics to quantify the precision of the proposed specifications in terms of how well they capture the desired variations.

3. Efficient Verification in Latent Space

In this section we present an efficient approach to verify the robustness of neural networks against specifications capturing diverse and realistic input variations. Formally, given a network N and a variation v, we define the Semantic Verification Problem (SVP) as the network verification problem where X_{des} is the preimage of a set Z_{des} under a partially invertible encoder e:

$$N(x) \in Y_{\text{des}}, \forall x \in \{X \mid e(X) = Z_{\text{des}}\}.$$

Here, Z_{des} is a set in the encoder's output space that captures v and the input-output specification set is $(Z_{\text{des}}, Y_{\text{des}})$. In the following, we study the Semantic Robustness Verification Problem (SRVP), which is an instance of SVP that checks for network robustness, i.e., Y_{des} is a singleton. To solve SRVP, we propose the multi-head architecture shown in Figure 1.

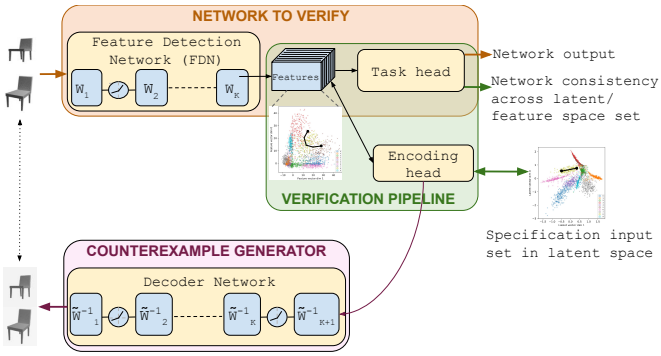


Figure 1. The proposed pipeline consists of an LVM, built around the network to be verified, that takes the network's intermediate features as its input. The LVM encoder is invertible and maps its input feature set to a spatially coherent latent manifold, where the input specification is defined.

Verification pipeline. The network N to be verified is divided into a *Feature Detection Network* (FDN), consisting of N's initial layers, and a *task head*, e.g., a classification or regression head. In our implementation, N is a classifier, but the approach applies similarly for other learning tasks. To define the specification, we add an LVM after the FDN. The encoder of this LVM is invertible and maps the intermediate features, output by FDN, to its latent space where the input specification is defined. Unlike the feature space, this latent space is queryable by using the decoder and can be made approximately bounded.

The verification path with this pipeline is limited to the inverse of LVM's encoding head and N's task head (green region in Figure 1). Differently from previous work, the decoder here is only used for training the LVM and as a counterexample generator to provide semantically meaningful counterexamples when verification outcomes are negative.

Specification inputs in latent space. We consider specifications focused on encoding the network's invariance to changes in an object-of-interest in the input. These changes include non-planar transformations, specific semantic changes of the object or its arbitrary task-orthogonal variations commonly occurring in the dataset. The learning tasks and datasets where the objects-of-interest and their variations are discernible against unimportant content in the input are well suited for such specifications.

We now describe the definitions we consider for the local latent space set Z_{des} against which the network is verified.

- **Segment.** We define Z_{des} to be a path connecting the encodings of two images $\{x_1, x_2\}$ that differ in the intended attributes or transforms (e.g., images of an object with different poses or backgrounds). Multiple definitions for Z_{des} are possible based on the traversal between the pair encodings, such as the maximum likelihood-based traversal [24], splines using multiple intermediate encodings, etc. With the Shortest Length Path (SLP), we get

  $$Z_{\text{des}} = e(x_1) + \alpha\left(e(x_2) - e(x_1)\right), \alpha \in [0,1]^{\text{latent\_dim}}.$$

  This Z_{des} is highly expressive since, in general, the image pair can be of any two images whose semantic interpolations should be consistently predicted by a network.
- **Axis.** We define Z_{des} to be the set generated by varying an encoding's disentangled dimensions, such as those trained to capture variations of a specific attribute or a single degree of freedom for pose change. For example, we may consider

  $$Z_{\text{des}} = [\max\{z_{\text{l}}, e(x)_i - \epsilon\}, \min\{e(x)_i + \epsilon, z_{\text{u}}\}], i \in I_C$$

  where I_C is the set of conditional dimension indices and $[z_{\text{l}}, z_{\text{u}}]$ are the limits containing a significant percentile of the distribution along these dimensions.

  Given conditional training, this Z_{des} can capture all variations of an image generated by an attribute or transform change, even when this image has no similar variations present in the dataset.
- **Region.** We define Z_{des} as a ℓ_∞-norm ball around an image encoding, i.e., $\mathcal{B}_\epsilon(z) = \{z' \mid ||z' - e(x)||_\infty < \epsilon\}$. Given a spatially coherent manifold, Z_{des} should capture the commonly occurring variations of an image.

The rest of our method is independent of the choice of Z_{des}.

Note again that since the specifications cannot be guaranteed to be complete by construction, we empirically ascertain Z_{des} to be sufficiently capturing the intended variations using metrics presented in Section 5.4. Also note that [34] gives theoretical guarantees that local epsilon balls in a sufficiently trained Conditional VAE's latent space, necessarily generate sufficient instances of local input perturbations. This further validates the use of *Axis* and *Region* queries for formal robustness checks and the use of VAEs as the LVM in our pipeline (details in the next section). The specifications can also be systematically inspected by means of sample reconstructions, representative results for which are shown in Figure 2.

Verification. Having formally defined Z_{des} to capture a variation v, the SRVP becomes a standard verification problem for a network composed of the inverted encoding head and the original network's task head. This problem can be solved by *any* standard NN verifier; we use VeriNet [15, 17], an open-source linear programming-based verification toolkit. The *invertibility of the encoding head* is necessary to guarantee that the feature vectors that map to Z_{des} are contained in the bounds of feature space set obtained during verification. In our queries, the output constraints Y_{des} encode invariance of N to v. When the query outcome is negative, the verifier provides a counterexample in Z_{des} which is then mapped to a counterexample in input space using the decoder. This *ideally* gives an instance of v against which the network is incorrect (see Section 5.3 for details).

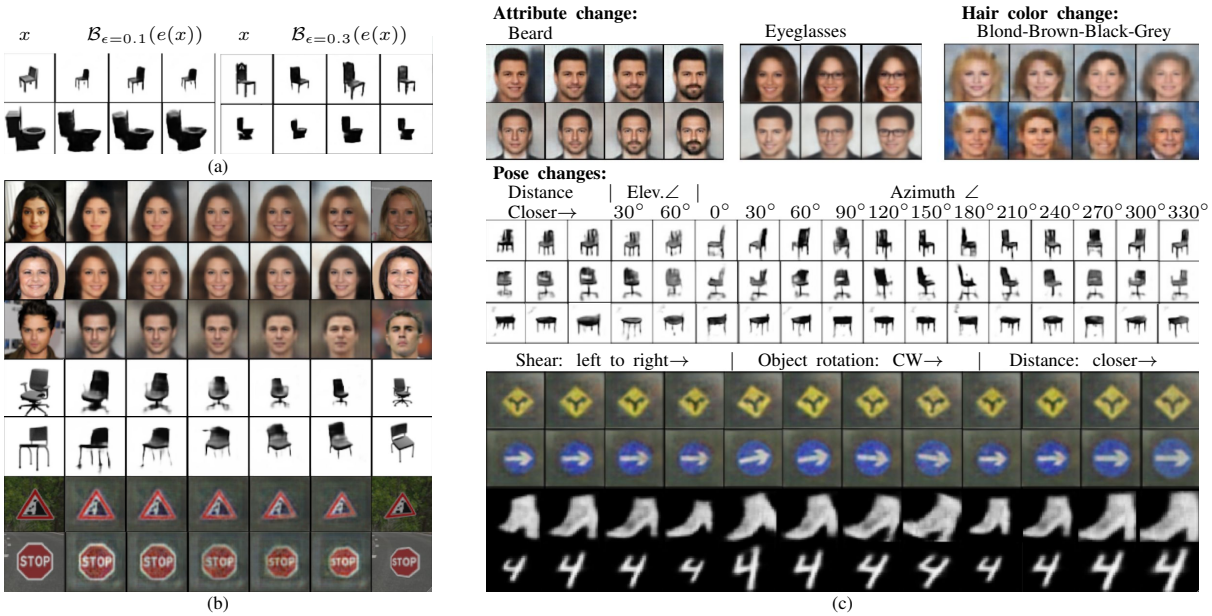


Figure 2. Reconstructions by our LVMs for *Region* specifications in (a), *Segment* specifications in (b) and *Axis* specifications in (c). (a) shows variations sampled from ϵ-balls around an image encoding; notice how a larger ϵ captures more varied changes of the input. The middle columns in (b) show reconstructions of samples from a segment joining the encodings of the dataset images in outer columns. In (c), all images are reconstructions and can be seen undergoing changes as specified conditional dimensions are varied.

Discussion. We evaluate the approach experimentally in Section 5; below we analyse it against the last two shortcomings of the EDN approach discussed in Section 2.

1. Observe that this approach directly verifies network features corresponding to latent space sets defined by mappings of the original images, rather than the reconstructed images. Thus, the pipeline ensures that the network is tested against at least the original image(s) that form the basis of a specification. Notice from Eq. 2, that this is not the case with the EDN approach if the decoder is not the exact inverse of the encoder. Moreover, since the encoding head is invertible, variations of an image mapping onto the same latent vector directly implies the stability of the network against these variations. Therefore, a many-to-one mapping is not a barrier for our approach, whereas it leads to insufficiently diverse input specifications with the EDN approach.
2. The proposed approach employs shorter paths for verification than the EDN approach. Since the encoding head is invertible, a latent space set need not be passed through the decoder and original network to obtain its corresponding feature set, and since most verification methods have an exponential complexity dependence on the number of unstable activations, the exclusion of the high-dimensional decoder and initial network layers from the verification path allows this approach to scale more favourably than the EDN approach.

A concern over the proposed approach could be that the network to be verified is partly used to define the input specification that it is verified for. However, consider the FDN-encoding head pair as an independent encoder, and the encoding head inverse as a compute efficient analogue of the decoder-FDN pair in the EDN pipeline. From this perspective, the proposed approach is identical to the traditional EDN-based approaches for robustness validation. Additionally, the proposed approach benefits from:

- Added flexibility to use a deep decoder for manifold learning and sharper counterexample generation without impacting verification outcomes. Verification also ensures maximal recall in finding counterexamples as opposed to a gradient search-based approach.
- Relative flexibility in choosing the network layer after which the encoding head is added and thereby, how many of the network's posterior layers feature in the verification path. This decision can depend on the scalability and implementation support of the chosen verification backend or the feasibility of learning a latent space of given dimensionality (see Section 5.2).
- Independence with respect to the FDN architecture's depth and type (residual, recurrent, non-standard convolutional, etc.), and possibly to input dimensionality[1], leading to an easier extension of the verification approach to newer architectures.

[1]Independence to input dimensionality is conditional on the nature of the FDN and its last layer being fully-connected (fc); however, since most vision inference tasks, such as classification, regression and detection, involve dimensionality reduction and network's posterior layers are usually fc, the specification input dimensionality can almost always be made lower than that of input data.

4. Implementation

In this section we describe our implementation of the proposed verification pipeline [13].

Encoding head. In terms of design choices, all common LVMs for data generation, including VAEs, Normalising Flows (NFs) and GANs, can be used in the pipeline if the following holds:

- The FDN of the network to be verified can serve as the backbone of the LVM encoder (e.g., if the FDN involves a dimensionality change then NFs cannot serve as the LVM).
- The verification backend can verify the inverted encoding head of the pipeline.

Given any LVM, Z_{des} is defined in *the output space of the LVM encoding head* in ways described in Section 3. A VAE fits the pipeline particularly well given the backing of the theoretical analysis in [34], and as it regularises the underlying latent space to resemble a prior. This prior (normal, von Mises-Fisher distribution in $\mathcal{N}$-VAE [21], $\mathcal{S}$-VAE [8] resp.) is light-tailed, so the latent space can be approximately bounded or truncated, and the epsilons in our specifications have bounded domain. Note that the latent variable z in a VAE is not the output of the encoder but a random variable conditioned on its outputs ($\mu, \sigma \in \mathbb{R}^{\text{latent\_dim}}$) by reparameterisation [21]. Since reparameterisation cannot be inverted, Z_{des} is defined using the vector $[\mu, \sigma]$ instead of z, albeit with no change in Z_{des}.

In contrast, GANs do not use an encoder to learn a latent distribution and instead take a prior distribution as input. However, when using a GAN as the LVM, we still add an encoding head after the FDN to transform the unconstrained FDN output to a desired prior for the GAN and define input-parameterized local specifications.

LVM design. As discussed previously, our pipeline features an encoding head after the FDN of the network to be analysed. Since the encoding head must not undergo dimensionality change, the dimension of the flattened FDN output is the dimension of the encoding head layers and the latent space where the input specification is defined. Typically, a shorter FDN implies larger latent space dimensionality and verification involves more network layers.

We use two types of feed-forward encoding heads: one consists of pairs of an invertible activation and fully-connected (FC) layers; the other consists of alternating affine-coupling layers [9]. The former is easier to train and supported by most NN verifiers, but its invertibility to sufficient precision needs to be validated by checking the singular values of the individual layers' weight matrices. Affine-coupling layer is invertible-by-construction but not supported by standard NN verifiers. To support its verification, we replace its typically input dependent scaling and translation unit with an input independent learnt vector and an FC layer respectively. This modified affine-coupling layer is equivalent to an FC layer in expressiveness, but is more challenging to train.

One can increase the encoding head depth with the pairs described above until the LVM is adequately trained; in our experience, a single layered encoding head sufficed for good reconstructions. The decoder is always a deep upscaling network mirroring the pipeline encoder.

LVM training. In addition to the negative log likelihood or GAN loss, we conditionally train latent dimensions [5, 23] for *Axis* specifications. We use Cross-Entropy (CE) loss for discrete attributes. For instance, we train one dimension with binary CE loss so that $[0, 1]$ for this dimension corresponds to variations from *no attribute* to *complete attribute*. Note that we cannot use the common conditional generation approach of augmenting encodings with attribute labels [31] because the entire encoding must be inferred from the FDN output for invertibility. Therefore, differently from usual practice, our VAEs for conditional generation do not take attributes as priors to condition on but rather learn them with supervision from data.

Continuum in transformations is important when verifying against pose changes, which CE loss may not provide. Therefore, dimensions capturing continuous changes are trained as *graphics code* as per the procedure in [23].

Note that it may not be possible to train an LVM by adding an encoding head to the extreme end of every network, as network's task-agnostic information may be lost till that layer. However, if an LVM can be trained by adding an encoding head after the K^{th} network layer, it is inexpensive to train a new LVM from after the $(K-k)^{th}, k > 0$, layer, with the frozen original network.

Datasets preparation. To train the LVMs for our experiments, we used public datasets with custom dataloading to have attribute and transform labels for every input. We only used datasets for which realistic variations can be produced with labels, as it is required for conditional training and forming precise queries for pipeline evaluation (see Section 5.4). We worked with five datasets: CelebA [25], Traffic Signs Recognition (TSRD), 3D Objects (3DOD), (Fashion)MNIST [38]. For TSRD, we took traffic signs and city scene crops, apply planar (rotation, scale, photometric) and spatial (perspective) transformations on the former and blend them into the latter during dataloading. For 3DOD, we took 3D models from ModelNet10 dataset [37] and render their images from different distances, azimuth and elevation angles in Blender [6]. For all trainings, the dataset was balanced such that every class had adequate samples of the variation being verified against.

5. Experimental Analysis

We report on our experiments to validate the proposed approach, with focus on answering three key questions:

Q1. Can an LVM added to a classifier generate an appropriate latent manifold and good reconstructions?

Q2. Can the proposed pipeline efficiently verify deep networks?

Q3. Can the specifications, discussed in Section 3, encode intended input variations?

To assess Q1, we refer to the reconstructions presented in Figure 2 and the low values of mean reconstruction error and FID score [18] reported in Table 1. To assess Q2, we refer to the verification outcomes for deep networks of different architectures reported in Sections 5.1, 5.2. To assess Q3, we refer to the values of metrics quantifying the precision of a specification in capturing intended variations, discussed in Section 5.4. With this overview, we now discuss the experiments in greater detail.

5.1. Verifying segments joining input encodings

To verify robustness against variations reflected in an image pair, we generated *Segment* specifications capturing object pose and viewpoint changes. For CelebA, we also verify specifications capturing the classifier's invariance to task-orthogonal attributes. These queries are similar to the queries in [27] which uses the EDN approach and reports successful *deterministic* verification of 23.8% queries for a ConvMed model with 64k activations. Based on the classifier layer separating the FDN and task head, multiple verification pipelines were trained for each classifier. The verification outcomes for all queries and pipelines are reported in Table 1, from which the following observations can be drawn:

- The mean reconstruction errors and FID scores for the multiple SRVP pipelines per classifier are low with consistent verification outcomes, indicating that latent spaces of a considerable range of dimensionality can be successfully learnt and verified against.
- The SRVP pipelines require an order of magnitude less time than the EDN pipelines to solve safe queries, and produce fewer trivial output bounds or undecided cases. This remains the case with the inclusion of more network layers in the verification path.
- The verified robust accuracy tightly lower bounds the adversarial accuracy computed against gradient-ascent-based latent space interpolation attacks [28].

5.2. Verifying along latent dimensions

As discussed in Section 3, *Axis* specifications can be used to verify an input against the entire range of a task-orthogonal change (e.g., style, or pose). Figure 3 shows representative results for verifying these specifications where the local input set is increased for individual latent dimensions of an encoding. For many inputs, conditional dimensions were verifiably robust for almost their entire range, while dimensions without conditional training were found to be robust for much smaller epsilons. Decoded samples from latent dimensions in Figure 4 corroborate these outcomes. The object class change was not observed for all non-conditional dimensions, but conditional dimensions were more consistent in preserving the object class. However, with increase in latent dimensionality, as with a shorter FDN, it became harder to balance the conditional and standard LVM training to achieve both good reconstructions and distinct attribute changes with class consistency. This explains the reduced transform detections for higher dimensional manifolds reported in Figure 3. In Section 5.4, we present a validity metric to assess the success of conditional manifold training and in turn, derive expectation on the usefulness of *Axis* specifications defined in the manifold.

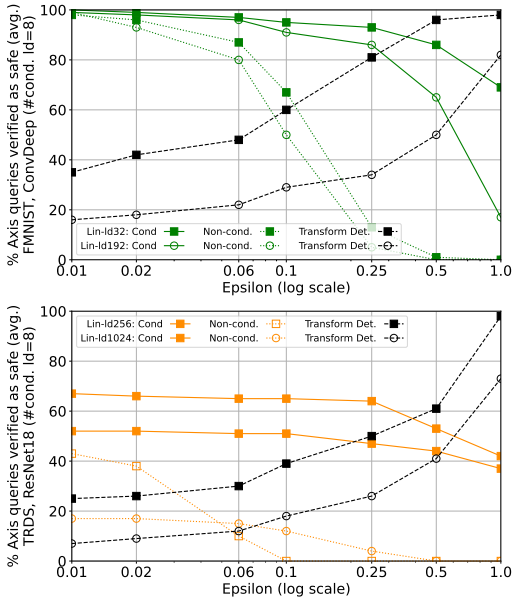


Figure 3. Both graphs show mean percentage of *Axis* queries verified as safe (out of 200 tested) for local intervals of $\pm\epsilon$ for LVMs of different latent dimensionality. The percentage of reconstructions of samples along the conditional axes, around the ϵs reported on the x-axis, in which a learnt transform detector detects the intended transform, is also plotted. The axes were trained with (B)CE loss with 0,1 mapping to the base and the attribute added image resp.

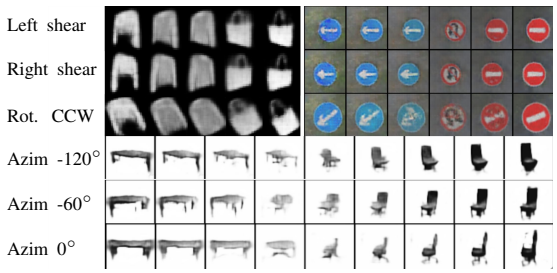


Figure 4. All images above are generated by an LVM from a single image per dataset. The columns feature samples along non-conditional dimensions, and each row has a constant conditional axis value which enforces rendering of the specified transform in all its reconstructions.

In case of *Region* specifications, where the local ep-

Dataset (size)	Verifying inv. to *	Network	Accuracies* Std.	Adv.	Pipeline** Type #Ver.actv.layers (actvs.)	Verification outcomes (100 std.safe queries) Safe(s†)	Unsafe(s)	Undec.(s)	Rob.acc	Recons.outcomes RMSerr⋆	FIDscore
Runs for efficiency comparison of EDNs (with decoders of different depths) vs. SRVPs (with encoding heads after different n/w layers).											
CelebA				64.2%	ED*tiny*N-32 8 (5k)	61 (**6.8**)	39 (3.94)	0	56%	0.064	218.61
				65%	ED*deep*N-32 9 (27k)	64 (**22.3**)	29 (31.6)	7 (64)	59%	0.053	107.42
(64x64)		ConvSmall	92%	69%	ED*resnet18*N-32 11 (0.2M)	0	42 (370)	58 (802)	0%	0.051	89.62
		(6k ReLUs)		91%	SRVP Lin1-32 1 (64)	99 (**0.4**)	1 (0.01)	0	91%	0.058	114
				89%	SRVP Lin1-64 2 (192)	94 (**0.5**)	6 (0.02)	0	85.5%	0.042	112
	Head Flip			87%	SRVP Lin1-192 3 (576)	88 (**1.1**)	12 (0.08)	0	81%	0.035	105
	(gender cla.)			89%	SRVP Lin1-392 4 (1.3k)	91 (**2.7**)	9 (0.11)	0	84%	0.031	102
				74%	ED*tiny*N-32 8 (86k)	59 (**32.8**)	41 (20)	0	56.6%	0.07	264
				81%	ED*deep*N-32 9 (0.1M)	66 (**35.3**)	22 (23.5)	12 (64)	63%	0.06	223
(128x128)		ConvBig	96%	77%	ED*resnet18*N-32 11 (475k)	0	21 (410)	79 (816)	0%	0.06	124
		(72k ReLUs)		93%	SRVP Lin1-64 1 (64)	95 (**0.4**)	5 (0.07)	0	91%	0.23	252
				95%	SRVP Lin1-192 2 (256)	96 (**0.8**)	4 (0.09)	0	92%	0.15	225
				94%	SRVP Lin1-392 3 (648)	96 (**1.9**)	4 (0.11)	0	92%	0.04	202
SRVP runs with encoding heads of different dimensions for different n/w architectures and datasets											
(64x64)		ResNet18	84%	82%	SRVP Lin1-256 1 (512)	95 (2.6)	5 (0.1)	0	80%	0.022	104
TRDS	Planar Pose	ResNet18	92.5%	91%	SRVP Lin1-256 2 (1k)	96 (2.1)	4 (0.21)	0	89%	0.03	116
(64x64)	(sign cla.)			89%	SRVP Lin1-1024 2 (1.5k)	94 (3.8)	6 (0.46)	0	87%	0.03	106
		MobileNetv2	82%	78%	SRVP Lin1-640 1 (640)	93 (2.1)	7 (0.42)	0	76%	0.09	133
3DOD	Spatial Pose	ResNet18	97%	94%	SRVP Lin1-256 2 (1k)	94 (1.9)	6 (0.2)	0	91%	0.04	106
(64x64)	(obj cla.)			90%	SRVP Lin1-1024 2 (1.5k)	92 (4.7)	8 (0.45)	0	89%	0.02	98

$*$ Std. and Adv. accuracy are the network's accuracy against clean, augmented data and latent-space PGD attacks [28] (α=2e-3, #iters=2000) resp. The PGD attacks are found for $N \circ d(z)$ in EDN pipelines and $e^{-1} \circ N_{\text{head}}(z)$ in SRVP pipelines for $z \in Z_{\text{segment}}$.
$**$ The suffixed integer in the pipeline type is the dimensionality of its latent space and is equal to its verification input. Higher dimensions for a SRVP pipeline for the same network indicate an earlier split and the inclusion of more layers in the verification path.
† Seconds (s) reported are the mean query solving time. All queries were run on a 24 core, 256GB CPU with timeout of 60 or 800 seconds.
$\star$ The reconstruction error is per-pixel for normalised float [-1,1] images. The FID score is computed by using pytorch-fid package [30].
All pipelines for the same network were trained until comparable reconstruction outcomes, were tested for the same queries using the same backend and use $\mathcal{N}$-VAE as the LVM so that all of them have an approximately normal latent manifold distribution.

Table 1. Verification results for *Segment* queries; representative specification inputs for the above are shown in Figure 2b.

silon ball is increased uniformly in all latent dimensions, we could verify robustness for a maximum $\epsilon = 0.01$ (for $<$10% queries) for SRVP Lin-32 pipelines.

5.3. Counterexamples analysis

Counterexamples are instances of the input specification set that falsify the robustness property under investigation. In our approach, the verifier finds a latent space counterexample $z^* \in Z_{\text{des}}$ for every unsafe query such that $N_{\text{head}} \circ e^{-1}(z^*)$ demonstrates non-robustness. However, since the LVM decoder is not the exact inverse of its encoder, the decoded counterexample could be spurious, i.e., $N \circ d(z^*)$ may not be a counterexample. Therefore, we decode every z^* and label it a *valid* counterexample iff its reconstruction is also a counterexample for the network.

Similar to [16], in the case of a spurious counterexample z^*, we run a local gradient search for a valid one, taking z^* as the starting point. In comparison to starting from a mapped vector, this local gradient-based search around z^* takes much fewer iterations to reach a valid counterexample in most cases, and results in high precision in finding *valid* and in-domain counterexamples (see Figure 5).

Dataset	Valid-to-all ceg. ratio	Valid-to-all ceg. ratio-w-PGD search (full search #iters, refinement #iters)
CelebA	0.34	1 (174, 2.75)
TRDS	0.85	0.92 (385, 11)
FMNIST	0.7	0.96 (150, 9.6)

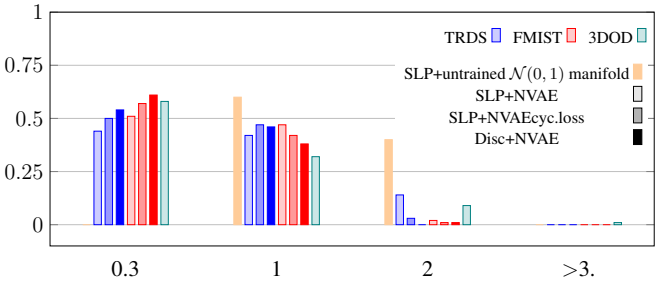
Figure 5. The table reports the mean ratios of valid counterexamples (ceg.) to all cegs found by SRVP pipelines for network $N_{\text{head}} \circ e^{-1}$ through verification of 100 *Segment* queries, with and without PGD-based search in the second and third column resp. The full search and refinement iterations are median PGD iterations of step-size 1e-3 for success in the latent/attribute space attack [28] and in the local search for a valid ceg around a verification-returned spurious ceg resp. Rows in the figure below show reconstructions for some unsafe *Segment* and *Axis* queries with their valid cegs in the rightmost column.

5.4. Validating the LVM based pipeline

Having assessed the capabilities of our approach in terms of scaling to deep networks and the fidelity of returned counterexamples, we now evaluate its precision in sufficiently capturing the intended variations in the proposed specifications. For this, we consider two latent space-based validity metrics as described below.

The assumption behind *Segment* specifications is that the in-domain interpolations of an image pair map to the interpolations of its encoding pair. To validate this, we measure the distance of the encodings of the desired interpolations of an image pair to the path joining its encoding pair. In practice, for validating pose invariance queries, we generate images with varying extents of a transform, construct a path between the encodings of the image pair with the most extreme mutual transform, and compute the distance between the encodings of the intermediate transforms and their projection on this path. This distance is divided by square-root of latent dimensionality to generalise across manifolds of different dimensions. We compute this metric for the following pairs of encoding paths and VAE manifolds used in our pipelines, and report its mean value distribution below.

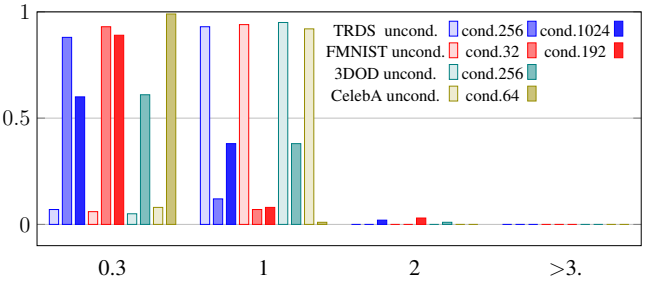
1. Shortest Length Path (SLP) + untrained normal distribution manifold (baseline),
2. SLP + $\mathcal{N}$-VAE,
3. SLP + $\mathcal{N}$-VAE trained with a latent space-based cycle consistency loss that encourages its encoder-decoder to be an inverse pair,
4. Real-vs-fake discriminator-guided path + $\mathcal{N}$-VAE.



Normalised histogram of distances of intermediate transform encodings to Z_{des}, i.e., $\mathbf{d}(e(x_i), proj_{Z_{\text{des}}} e(x_i))/\sqrt{m}, e(.) \in \mathbb{R}^m$.

In comparison to the primitive baseline, the metric values with $\mathcal{N}$-VAEs are low for most queries as desired, and further reduce on incentivising consistency in their mappings, as in 3., and making the encoding traversals more precise and segmented, as in 4. Although we do not observe notable changes in the verification outcomes with different $\mathcal{N}$-VAEs, this metric could be used to guide the construction of input specifications. Notice that this metric is the latent space-based analogue of the pixel space distance used to define input specifications and evaluate generators for EDN-based verification [20].

The motivation behind *Axis* specifications is the disentanglement of conditional dimensions in capturing specified variations. To assess disentanglement, we compute the ratio of the highest value among the incorrect conditional dimensions to the value of the correct conditional dimension of an encoding for a discrete attribute. For the continuous transforms encoded as graphics code [23], an equivalent metric is the ratio of the maximum variance among the incorrect conditional dimensions to the variance in the correct conditional dimension, computed for a batch of encodings. Below we report the mean value distribution of this metric for normally and conditionally trained $\mathcal{N}$-VAEs. As expected, conditional training results in low metric values as desired, while normal training results in similar values for most dimensions and metric values closer to unity.



Normalised histogram of maximum incorrect-to-correct dimension ratios for conditionally trained LVMs. Numbers in the legend entries denote manifold dimensionality.

These latent space-based metrics are more relevant to our approach than the pixel space-based or the attribute detection-based analyses relevant to the EDN approach. This is because the effectiveness of verification with our approach depends more on the spatial coherence and precision of the LVM encodings, than its reconstruction capabilities.

6. Conclusions

In this work we introduced an efficient and precise approach for verifying specifications in the latent space. By means of the proposed approach, we could effectively assess and verify deep networks against pose changes, and provide counterexamples that inform of their fragilities.

In terms of limitations, we discussed that the ease of learning a conditional latent manifold, and the computational gains of our verification approach can be affected by the increase of layers in the network head of our pipeline. However, since the largest activation layers can be excluded from the verification path with our approach, it still fares better than the existing approach. A general limitation of latent-space based verification approaches is their necessity for an adequate LVM, which, as discussed for our approach, may require the original network to be trained as part of the proposed pipeline. While we did not find these limitations to hinder the application of our approach to the practical usecases demonstrated in this work, further experiments should be conducted to better assess its generality.

References

[1] M. Akintunde, A. Lomuscio, L. Maganti, and E. Pirovano. Reachability analysis for neural agent-environment systems. In *Proceedings of the 16th International Conference on Principles of Knowledge Representation and Reasoning (KR18)*, pages 184–193. AAAI Press, 2018. 1

[2] M. Balunovic, M. Baader, G. Singh, T. Gehr, and M. Vechev. Certifying geometric robustness of neural networks. In *Advances in Neural Information Processing Systems (NeurIPS19)*, pages 15313–15323. Curran Associates, Inc., 2019. 1, 2

[3] B. Batten, P. Kouvaros, A. Lomuscio, and Y. Zheng. Efficient neural network verification via layer-based semidefinite relaxations and linear cuts. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI21)*, pages 2184–2190. ijcai.org, 2021. 2

[4] E. Botoeva, P. Kouvaros, J. Kronqvist, A. Lomuscio, and R. Misener. Efficient verification of neural networks via dependency analysis. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI20)*, pages 3291–3299. AAAI Press, 2020. 2

[5] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Proceedings of the 30th International Conference on Neural Information Processing Systems (NIPS2016)*, page 2180–2188, 2016. 2, 5

[6] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. 5

[7] E. Cunningham, A.D. Cobb, and S. Jha. Principal component flows. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162, pages 4492–4519. PMLR, 2022. 2

[8] T. R. Davidson, L. Falorsi, N.D. Cao, T. Kipf, and J.M. Tomczak. Hyperspherical variational auto-encoders. In *Conference on Uncertainty in Artificial Intelligence*, 2018. 5

[9] L. Dinh, J. Sohl-Dickstein, and S. Bengio. Density estimation using real NVP. In *Proceedings of the 5th International Conference on Learning Representations (ICLR17)*. OpenReview.net, 2017. 5

[10] S. Dutta, S. Jha, S. Sankaranarayanan, and A. Tiwari. Output range analysis for deep feedforward neural networks. In *NASA Formal Methods*, pages 121–138. Springer International Publishing, 2018. 2

[11] N. Fijalkow and M. K. Gupta. Verification of neural networks: Specifying global robustness using generative models. *CoRR*, abs/1910.05018, 2019. 1

[12] S. Gowal, C. Qin, P. Huang, T. Cemgil, K. Dvijotham, T. Mann, and P. Kohli. Achieving robustness in the wild via adversarial mixing with disentangled representations. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR20)*, pages 1208–1217. IEEE Computer Society, 2020. 1

[13] H. Hanspal and A. Lomuscio. Srvp codebase. `github.com/hh10/Efficient-Verification-of-NNs-against-LVM-based-Specifications`, 2023. Accessed: 2023-03-26. 5

[14] P. Henriksen, K. Hammernik, D. Rueckert, and A. Lomuscio. Bias field robustness verification of large neural image classifiers. In *Proceedings of the 32nd British Machine Vision Conference (BMVC21)*. BMVA Press, 2021. 1, 2

[15] P. Henriksen and A. Lomuscio. Efficient neural network verification via adaptive refinement and adversarial search. In *Proceedings of the 24th European Conference on Artificial Intelligence (ECAI20)*, pages 2513–2520. IOS Press, 2020. 2, 3

[16] P. Henriksen and A. Lomuscio. DEEPSPLIT: an efficient splitting method for neural network verification via indirect effect analysis. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI21)*, pages 2549–2555. ijcai.org, 2021. 2, 7

[17] P. Henriksen and A. Lomuscio. Verinet. `github.com/vas-group-imperial/VeriNet`, 2021. Accessed: 2023-03-26. 3

[18] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, G. Klambauer, and S. Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS17)*, page 6629–6640. Curran Associates Inc., 2017. 6

[19] G. Katz, C. Barrett, D. Dill, K. Julian, and M. Kochenderfer. Reluplex: An efficient SMT solver for verifying deep neural networks. In *Proceedings of the 29th International Conference on Computer Aided Verification (CAV17)*, volume 10426 of *Lecture Notes in Computer Science*, pages 97–117. Springer, 2017. 2

[20] S.M. Katz, A.L. Corso, C.A. Strong, and M.J. Kochenderfer. Verification of image-based neural network controllers using generative models. *Journal of Aerospace Information Systems*, 19:574–584, 2022. 1, 2, 8

[21] D.P. Kingma and M. Welling. Auto-encoding variational bayes, 2013. 5

[22] P. Kouvaros and A. Lomuscio. Towards scalable complete verification of relu neural networks via dependency-based branching. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI21)*, pages 2643–2650. ijcai.org, 2021. 2

[23] T.D. Kulkarni, W.F. Whitney, P. Kohli, and J. Tenenbaum. Deep convolutional inverse graphics network. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. 2, 5, 8

[24] S. Lin and R. Clark. Ladder: Latent data distribution modelling with a generative prior. In *British Machine Vision Conference (BMVC20)*, 2020. 3

[25] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015. 5

[26] A. Lomuscio and L. Maganti. An approach to reachability analysis for feed-forward relu neural networks. *arXiv preprint 1706.07351*, 2017. 2

[27] M. Mirman, A. Hägele, P. Bielik, T. Gehr, and M. Vechev. Robustness certification with generative models. In *Proceedings of the 42nd ACM SIGPLAN International Conference on*

Programming Language Design and Implementation, page 1141–1154. Association for Computing Machinery, 2021. 1, 2, 6

[28] H. Qiu, C. Xiao, L. Yang, X. Yan, H. Lee, and B. Li. Semanticadv: Generating adversarial examples via attribute-conditioned image editing. In *Computer Vision – ECCV 2020*, pages 19–37. Springer International Publishing, 2020. 1, 6, 7

[29] A. Raghunathan, J. Steinhardt, and P. Liang. Semidefinite relaxations for certifying robustness to adversarial examples. *arXiv preprint arXiv:1811.01057*, 2018. 2

[30] M. Seitzer. pytorch-fid: FID Score for PyTorch. `github.com/mseitzer/pytorch-fid`, August 2020. Version 0.2.1. 7

[31] K. Sohn, H. Lee, and X. Yan. Learning structured output representation using deep conditional generative models. In *Proceedings of the 29th International Conference on Neural Information Processing Systems (NIPS15)*, volume 28. Curran Associates, Inc., 2015. 5

[32] S. Wang, K. Pei, J. Whitehouse, J. Yang, and S. Jana. Formal security analysis of neural networks using symbolic intervals. In *Proceedings of the 27th USENIX Security Symposium (USENIX18)*, 2018. 2

[33] S. Wang, H. Zhang, K. Xu, X. Lin, S. Jana, C. Hsieh, and J. Kolter. Beta-crown: Efficient bound propagation with per-neuron split constraints for complete and incomplete neural network verification. *arXiv preprint arXiv:2103.06624*, 2021. 2

[34] E. Wong and J. Zico Kolter. Learning perturbation sets for robust machine learning. *CoRR*, abs/2007.08450, 2020. 1, 3, 5

[35] E. Wong and Z. Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *Proceedings of the 35th International Conference on Machine Learning (ICML18)*, pages 5286–5295. PMLR, 2018. 1

[36] E. Wong, F. Schmidt, and Z. Kolter. Wasserstein adversarial examples via projected Sinkhorn iterations. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 6808–6817. PMLR, 2019. 2

[37] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR15)*, pages 1912–1920. IEEE Computer Society, 2015. 5

[38] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747, 2017. 5

[39] H. Zhang, P. Zhang, and C. Hsieh. Recurjac: An efficient recursive algorithm for bounding jacobian matrix of neural networks and its applications. *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI19)*, 33(01):5757–5764, Jul. 2019. 2