THE MANIFOLD HYPOTHESIS FOR GRADIENT-BASED EXPLANATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

When are gradient-based explanations meaningful? We propose a *necessary* criterion: explanations need to be aligned with the *tangent space* of the data manifold. To test this hypothesis, we employ autoencoders to estimate and generate data manifolds. Across a range of different datasets – MNIST, EMNIST, CIFAR10, X-ray pneumonia and Diabetic Retinopathy detection – we demonstrate empirically that the more an explanation is aligned with the tangent space of the data, the more interpretable it tends to be. In particular, popular post-hoc explanation methods such as Integrated Gradients and SmoothGrad tend to align their results with the data manifold. The same is true for the outcome of adversarial training, which has been claimed to lead to more interpretable explanations. Empirically, alignment with the data manifold happens early during training, and to some degree even when training with random labels. However, we theoretically prove that good generalization of neural networks does not imply good or bad alignment of model gradients with the data manifold. This leads to a number of interesting follow-up questions regarding gradient-based explanations.

1 INTRODUCTION

A large number of algorithms aim to provide post-hoc explanations for the output of neural networks (Simonyan et al., 2014; Bach et al., 2015; Shrikumar et al., 2017; Ancona et al., 2018; Lim et al., 2021). Many of them are, directly or indirectly, based on the gradient with respect to the input (Smilkov et al., 2017; Sundararajan et al., 2017; Garreau & Mardaoui, 2021; Agarwal et al., 2021). A particularly interesting application for gradient-based input attribution methods is natural image classification. Despite recent attempts to provide a priori interpretable image classifiers, neural networks remain exceptionally successful at classification (Chen et al., 2018a). However, recent work has demonstrated that post-hoc explanation methods fail various sanity checks (Adebayo et al., 2018b;a; Kindermans et al., 2019; Arun et al., 2020), and some have even suggested that they should not be used at all (Rudin, 2019).

In this paper, we try to understand when and why gradient-based input attribution methods can be meaningful. To this end, we propose the following hypothesis:

Gradient-based explanations are more meaningful the more they are aligned with the tangent space of the data manifold.

Consider image classification. It is widely believed that natural image data concentrates around a low-dimensional image manifold (Goodfellow et al., 2016, Section 5.11.3). This image manifold captures the geometric structure of the data. In particular, the tangent space captures all components of an image that can be changed while still staying within the realm of natural images. If a gradient-based explanation approximately lies in the tangent space, this means that it highlights a meaningful way in which the different components of an image contribute to the prediction. If a gradient based explanation lies orthogonal to the tangent space, this means that it points in some direction that would not lead to realistic images, and a human would have a hard time to understand its meaning.

Our motivation for proposing this hypothesis is twofold. First, we believe that it is intuitive, and we provide empirical evidence in its support. Second, we hope that the hypothesis can provide a perspective on why obtaining gradient-based explanations might be more difficult than classification.

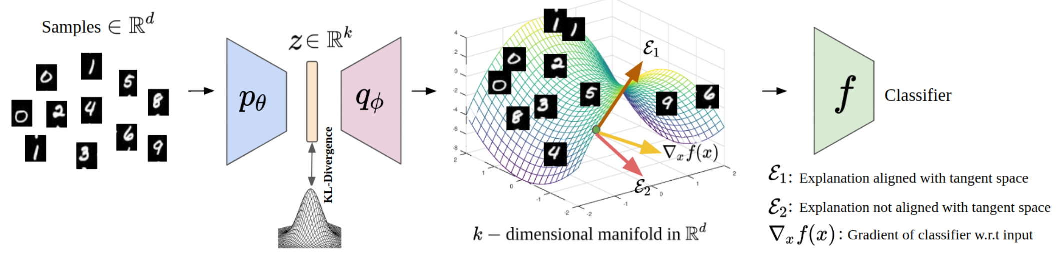
Samples $\in \mathbb{R}^d$
p_θ
$z \in \mathbb{R}^k$
KL-Divergence
q_ϕ
$\mathcal{E}_1$
$\nabla_x f(x)$
$\mathcal{E}_2$
k – dimensional manifold in $\mathbb{R}^d$
f
Classifier
$\mathcal{E}_1$: Explanation aligned with tangent space
$\mathcal{E}_2$: Explanation not aligned with tangent space
$\nabla_x f(x)$: Gradient of classifier w.r.t input

Figure 1: Conceptual overview of our approach. We first estimate the data manifold of an existing dataset with a variational autoencoder, then use the decoder as a generative model. On the generated data, we train a classifier f. For this classifier, we evaluate whether different gradient based explanations $\mathcal{E}_i$ align with the tangent space of the data manifold. Moving along an explanation aligned with the tangent space keeps us in the manifold, whereas moving along an orthogonal explanation takes us out of manifold. Our hypothesis is that the latter does not lead to meaningful explanations because it describes changes that lead to unnatural images.

To evaluate the hypothesis empirically, we employ autoencoders to estimate the data manifolds of five different datasets - MNIST, EMNIST, CIFAR10, X-ray pneumonia and diabetic retinopathy detection. As depicted in Figure 1, we also use variational autoencoders as generative models. This allows us to generate datasets with completely known manifold structure. With this approach, we provide qualitative and quantitative evidence that explanations that are more aligned with the tangent space of the data are more interpretable.

To study when and why model gradients are aligned with the tangent space of the data, we first show that the gradients of neural networks *at initialization* are unrelated to the structure of the data manifold. This means that the learning algorithm picks up some aspects of the structure of the data manifold during training. We show that this happens early during training, and to some extent even when training with random labels. Moreover, under standard training procedures, the alignment between model gradients and the data manifold deteriorates as the model increasingly fits the labels. This is avoided by l_2 adversarial training, which significantly aligns the model's gradients with the tangent space of the data. Is it always the case that a neural network that generalizes necessarily adapts its gradients to the data manifold, at least to some degree? The answer is no, as we show theoretically. Without further assumptions, the relation between data manifold and model gradients is ambiguous: The alignment between the two quantities can be arbitrarily good or bad.

The organization of the paper is as follows. Sec.2 formally introduces the manifold hypothesis and outlines our conceptual approach. Sec.3 evaluates the hypothesis on five datasets. Sec.4 discusses the effects of adversarial training and the evolution of model gradients over the course of training. Sec.5 contains a formal proof that generalization does not imply alignment with the data manifold, Sec.6 discusses the related work and Sec.7 discusses the implications of our results.

2 The Manifold Hypothesis

Our goal is to evaluate the following hypothesis: A gradient-based explanation $\mathcal{E} \in \mathbb{R}^d$ at a point $x \in \mathcal{M}$ is more meaningful the more it is aligned with the tangent space of the data manifold at x. Below we first give a background on data manifolds, tangent spaces and model gradients; then we detail our evaluation approach.

2.1 Background

Data manifolds and tangent spaces. A k-dimensional differentiable manifold $\mathcal{M} \subset \mathbb{R}^d$ is a subset of a d-dimensional space that locally resembles $\mathbb{R}^k$. At every point $x \in \mathcal{M}$, the tangent space $\mathcal{T}_x$ is a k-dimensional subspace of $\mathbb{R}^d$. The tangent space $\mathcal{T}_x$ consists of all directions v such that $x + v$, for $||v||$ small, is again close to the manifold. Manifolds and tangent spaces are the subject of differential geometry, to which we refer for a comprehensive introduction.

The long-standing hypothesis that natural image data concentrates around a low-dimensional image manifold is supported by a number of empirical studies (Weinberger & Saul, 2006; Fefferman et al., 2016). However, accurately learning the data manifolds of natural image datasets – manifold learning – is difficult and the exact properties of these manifolds remain unknown (Cayton, 2005; Aamari & Levrard, 2019). Shao et al. (2018) investigate the properties of manifolds generated by deep generative models and find that they have mostly low curvature.

Model gradients and the fraction of the gradient in tangent space. We consider neural networks that learn differentiable functions $f : \mathbb{R}^d \to \mathbb{R}^C$. Here C is the number of classes and the model prediction is given by $\arg\max_i f(x)_i$. The gradient of class i with respect to the input is given by $\text{grad}_i(x) = \frac{\partial (f(x)_i)}{\partial x}$. Unless mentioned otherwise, we always consider the model gradient with respect to the predicted class and before the softmax.

At every point x that lies on the data manifold $\mathcal{M}$, we can decompose the gradient into a part that lies in tangent space and a part that is orthogonal to it. Formally, we have $\text{grad}_i(x) = v_1 + v_2$ with $v_1 \in \mathcal{T}_x$, $v_2 \in \mathcal{T}_x^\perp$ and $v_1 \perp v_2$. Here v_1 is the part of the gradient that lies in the tangent space, and v_2 is the part of the gradient that is orthogonal to the tangent space. If the gradient completely lies in the tangent space, we have $v_2 = 0$. If the gradient is completely orthogonal to the tangent space, we have $v_1 = 0$. In practice, some part of the gradient will lie in the tangent space and another part be orthogonal to it, that is we have $v_1 \neq 0$ and $v_2 \neq 0$. To quantitatively measure how well the gradient is aligned with the tangent space, we compute the

$$\text{Fraction of the Gradient in Tangent Space} = \frac{\|v_1\|}{\|\text{grad}_i(x)\|} \in [0, 1]. \tag{1}$$

2.2 How do we know the data manifold?

To estimate whether an explanation is aligned with the tangent space we make use of autoencoders. The various variants of *variational autoencoders* (Kingma & Welling, 2013; Higgins et al., 2017), allow to estimate the data manifolds of existing datasets. Importantly, they also allow to *generate* datasets with completely known manifold structure (Algorithm 1). We make use of two related approaches that we term the *generative approach* and the *reconstructive approach*. In both approaches, we first train an autoencoder on the orignal dataset.

Algorithm 1 Construct a classification problem with known manifold structure.

```
Require: Dataset $X = (x_i, y_i)_{i=1}^n$.
Require: Dimension of latent space $k \in [d]$.
  Train variational autoencoder $q_\phi(p_\theta(x))$ on $X$.
  Latent $\hat{z}_i \sim p_\theta(x_i)$, Reconstruction $\hat{x}_i \sim q_\phi(\hat{z}_i)$.
  Let $c : \mathbb{R}^{k+d} \to [C]$ solve $(\hat{z}_i, \hat{x}_i) \to y_i$.
  Sample $n$-times from the prior $\tilde{z}_i \sim \mathcal{N}(0, \mathbf{I}_k)$.
  The dataset is $\tilde{x}_i \sim q_\phi(\tilde{z}_i)$, $\tilde{y}_i = c(\tilde{z}_i, \tilde{x}_i)$.
  for $i \in [n]$ do                ▷ Compute tangent spaces
    for $l \in [d]$ do
      for $m \in [k]$ do
        $t_{i,l,m} = \frac{\partial (q_\phi)_l}{\partial z_m}(\tilde{z}_i)$
      end for
    end for
    $\mathcal{T}_{\tilde{x}_i} = \text{span} < \begin{pmatrix} t_{i,1,0} \\ \vdots \\ t_{i,d,0} \end{pmatrix}, \dots, \begin{pmatrix} t_{i,1,k} \\ \vdots \\ t_{i,d,k} \end{pmatrix} >$
  end for
```

The generative approach to create datasets with a completely known manifold structure. To generate a dataset with completely known manifold structure, we have to train a variational- or another generative autoencoder (Tagasovska et al., 2019). After training, we pass the original dataset through the autoencoder. Then we train an additional classifier that reproduces the original labels from latent codes and reconstructed images. Equipped with this labeling function, we sample from the prior and use decoder and labeling function to generate a dataset. If the decoder is differentiable, we can compute the tangent space at each datapoint x (Shao et al., 2018; Anders et al., 2020).

The reconstructive approach to create datasets with an estimated manifold structure. The main limitation of the generative approach is that we might not be able to obtain high-quality samples with reasonably small latent spaces. While there have been great advances in generative modeling, state-of-the-art models like hierarchical variational autoencoders (Vahdat & Kautz, 2020) require very large latent spaces, i.e. $k \approx d$. For our analysis, it is however critical that $\sqrt{k/d}$ is small – with $k = d$, the fraction of the gradient in tangent space is always 1. To evaluate our hypothesis on real-world high-dimensional image data where it is difficult to obtain realistic samples with not-too-large latent spaces, we rely on estimating the tangent space. That is we simply pass the original dataset through the autoencoder and take the reconstucted images with the original labels as our new dataset.

3 Putting the hypothesis to the test

Explanation algorithms. We consider four gradient-based input attribution methods: The gradient (Simonyan et al., 2014), Integrated Gradients (Sundararajan et al., 2017), Input $\times$ Gradient (Ancona et al., 2018), and SmoothGrad (Smilkov et al., 2017). The motivation behind Integrated Gradients is axiomatic. The motivation behind SmoothGrad is to reduce noise in the gradient. All four methods provide explanations as vectors in $\mathbb{R}^d$. We can evaluate how each method is aligned with the tangent space of the data manifold by computing the fraction of the explanation method in tangent space. While other methods also provide explanations as vectors in $\mathbb{R}^d$, we restrict ourselves to these four methods because they are directly related to the gradient with respect to the input, which is our main object of investigation.

Experimental setting. Given a dataset, obtained either with the generative or the reconstructive approach, we train a neural network to minimize the test error. For this network, we then evaluate how gradients and other gradient-based explanation methods relate to the data manifold. To evaluate whether an explanation is meaningful, we use qualitative evaluations as demonstrated in (Simonyan et al., 2014; Bach et al., 2015; Sundararajan et al., 2017; Smilkov et al., 2017). We also rely on the literature that demonstrates the utility of Integrated Gradients and SmoothGrad for diabetic retinopathy detection (Sayres et al., 2019; Van Craenendonck et al., 2020).

When we quantitatively evaluate the fraction of an explanation in tangent space (1), we need to account for the fact that even a random vector has a non-zero fraction in tangent space. A random vector is by definition completely unrelated to the structure of the data manifold. The expected fraction of a random vector that lies in any k-dimensional subspace is $\sqrt{k/d}$. In our MNIST32 task, for example, $d = 1024$, $k = 10$ and $\sqrt{10/1024} \approx 0.1$. Thus, we could only say that a gradient-based explanation is systematically related to the tangent space of the data manifold if, on average, the fraction of the explanation in tangent space is significantly larger than 0.1.

Datasets. We evaluate our hypothesis several datasets. This includes (i) MNIST32 and MNIST256, two variants of the MNIST dataset (LeCun et al., 1998) with 10 classes and 60000 grayscale training images and 10000 grayscale test images of size 32×32 and 256×256, respectively. (ii) EMNIST128, a variant of the EMNIST dataset (Cohen et al., 2017) that extends MNIST with handwritten letters and has over 60 classes, (iii) the CIFAR10 dataset (Krizhevsky et al., 2009). We also evaluate our hypothesis on two real world high dimensional image datasets: X-ray Pneumonia (Kermany et al., 2018) and Diabetic Retinopathy Detection [1]. Both tasks have been used before to study the properties of post-hoc explanation methods (Rajaraman et al., 2019; Luján-García et al., 2020; Amyar et al., 2020; Arun et al., 2020; Chetoui & Akhloufi, 2020; Van Craenendonck et al., 2020). All further details on the datasets are provided in appendix A.

3.1 Evaluation on grayscale images: MNIST32, MNIST256 and EMNIST128

We first demonstrate on MNIST32 that the part of the gradient that lies in tangent space is meaningful, whereas the part of the gradient that is orthogonal to the tangent space is not meaningful. The MNIST32 dataset was generated with a β-TCVAE (Chen et al., 2018b), trained on MNIST as described in Algorithm 1 (details in appendix A). The data lies on a completely known 10-dimensional image manifold in a 1024-dimensional space. On this dataset, we train a simple neural network with two convolutional and two fully connected layers to a test accuracy greater than 99%.

Figure 2 depicts model gradients for images from the test set of MNIST32 where the fraction of the gradient in tangent space is particularly large. The gradients are decomposed into the part of the gradient that lies in tangent space (second row) and the part of the gradient that is orthogonal to the tangent space (third row). It can be seen quite clearly that the part of the gradient that lies in tangent space has meaningful structure that in particular relates to prediction that we aim to explain. In case of the number 3 (fourth column), regions that would complete an 8 have negative attribution. In case of the number 8 (ninth column), pixels that distinguish the number from a 3 have a positive attribution instead. Similar patterns can be observed for other numbers, too. In contrast, the part of the gradient that is orthogonal to the tangent space does not have meaningful structure. In line with

[1]Dataset at https://www.kaggle.com/c/diabetic-retinopathy-detection

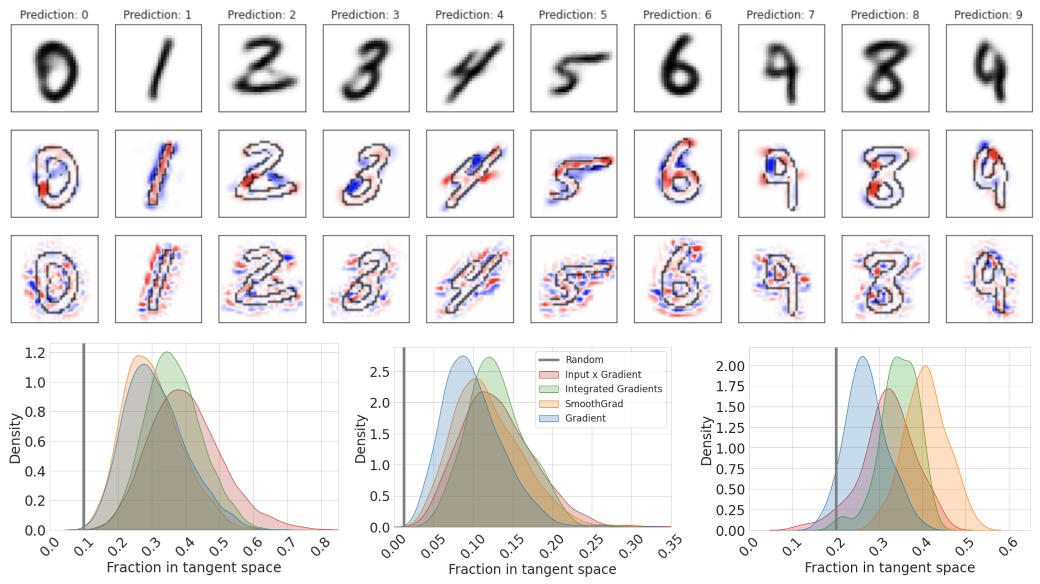

Figure 2: (First row) depicts images from the test set of MNIST32. (Second row) depicts the part of the gradient that lies in tangent space for these images. (Third row) depicts the part of gradient that is orthogonal to the tangent space for these images. Blue corresponds to negative attribution, red to positive attribution. (Bottom row) depicts the fraction of four different explanation methods in tangent space. (Bottom left) MNIST32. (Bottom center) MNIST256. (Bottom right) EMNIST128. Vertical gray line shows the expected fraction of a random vector in tangent space.

our hypothesis, it consists of seemingly unrelated spots of positive and negative attribution. We do not address the question whether the part of the gradient that is orthogonal to the tangent space at x is best described as noise or has a structural component. Shah et al. (2021) show that model gradients are not able to differentiate instance-specific features from other task-relevant features that are not pertinent to a given input instance. In this respect, note that part of the gradient that is orthogonal to the tangent space *at* x might lie within the tangent space of *other* instances on the data manifold.

To provide quantitative evidence for our hypothesis we look at the fraction in tangent space (Equation 1) of different gradient-based attribution methods. If the hypothesis is correct, we would expect attribution methods that are known to provide better explanations to have a higher fraction in tangent space. The bottom-left of Figure 2 depicts the distribution of the fraction in tangent space for model gradients, SmoothGrad, Integrated Gradients and Input $\times$ Gradient over all images from the test set. All four methods provide explanations whose fraction in tangent space is considerably larger than random. In particular, the mean fraction of the gradient in tangent space is significantly larger than random (t-test, $p < 0.001$). As expected under the hypothesis, Integrated Gradients and Input $\times$ Gradient improve upon the gradient. SmoothGrad seems to have little effect in this task.

As an experiment, we increase the dimension of the MNIST32 images to 256×256 by bilinear upsampling. This preserve the structure of the data manifold while increasing the dimension of the ambient space, i.e. on MNIST256 the ratio $\sqrt{k/d}$ is only 0.012. To measure the effect on the fraction of model gradients in tangent space, we train a ResNet18 (He et al., 2016) to a test accuracy greater than 99% and then compute again the fraction in tangent space of various gradient-based attribution methods (bottom-center of Figure 2). At first glance, the picture seems similar to that of MNIST32 on the left – the only modification being that SmoothGrad now improves upon the gradient. However, looking on the x-axis reveals that the fraction in tangent space of all methods is significantly reduced. Figure 8 in the Appendix shows that this also impacts the visual quality of explanations. We conjecture that, holding everything else fixed, aligning model gradients with the data manifold is harder as the ratio $\sqrt{k/d}$ decreases.

Finally, we present our results on EMNIST128. To create this dataset, we used the reconstructive approach. We resized the images form EMNIST to 128×128 and estimated the data manifold with a latent dimension of $k = 656$ (details are in appendix A.3). EMNIST128 has 60 classes and serves

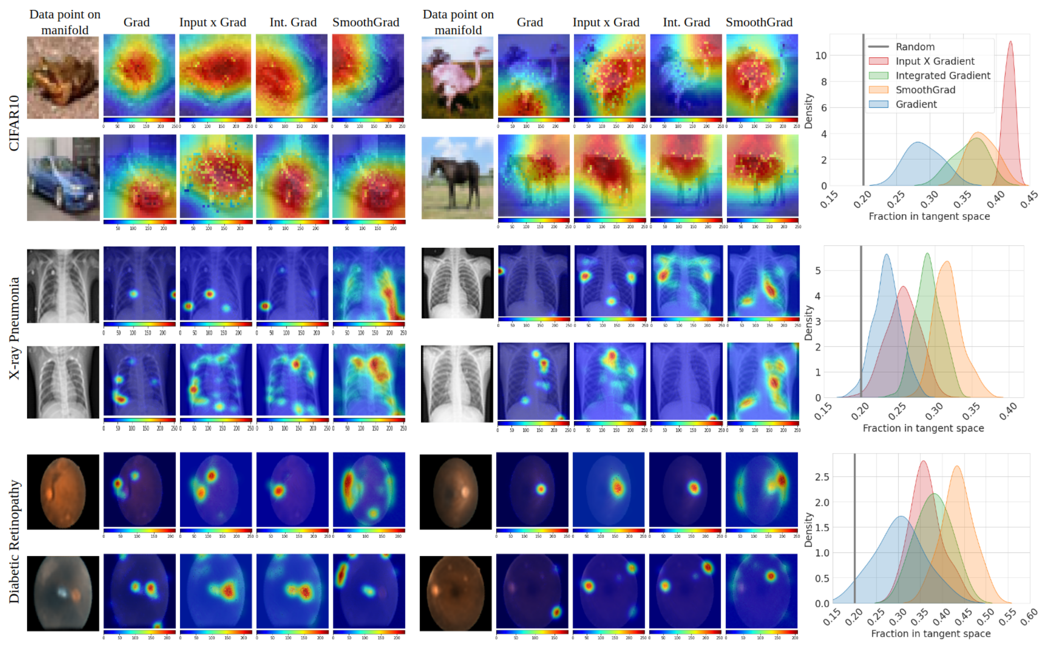

Figure 3: Comparing different explanation methods on (Top row) CIFAR10, (Middle row) X-Ray Pneumonia and (Bottom row) Diabetic Retinopathy detection. (Left part) qualitatively shows different explanations. (Right part) shows the distribution of the fraction in tangent space for all methods over the images from the test set. In high dimensional cases SmoothGrad has the highest component, whereas Input $\times$ Gradient has the highest component on CIFAR10.

as an example of a high-dimensional problem. We train a VGG16 network that yields an accuracy of 99%. Figure 2 (right) depicts again the fraction in tangent space of various explanation methods. SmoothGrad has the best performance, while again all post-hoc methods improve upon the gradient.

3.2 Experiments on CIFAR10 and Medical Diagnosis Datasets

We now turn to CIFAR10 and two medical diagnosis datasets, using the reconstructive approach. For CIFAR10, we trained a convolutional autoencoder with $k = 144$. A VGG16 classifier on the reconstructed images achieves an accuracy of 94.1%. For pneumonia and diabetic retinopathy detection, we resized the images to $1\times 256\times 224$ (i.e., $d = 57344$) and $3\times 224\times 224$ (i.e., $d = 150528$), respectively. We used a convolutional autoencoder with a latent dimension of 7168 and 6272, respectively. For both tasks we train ResNet18 classifier on the reconstructed images. It achieves an accuracy of 89% for pneumonia and 92% for retinopathy. Details are in Appendix A.

The top row of Figure 3 shows the performance of various gradient-based explanation methods on the reconstructed CIFAR10 dataset. The quantitative plot on the right shows that all post-hoc explanation methods have a higher fraction in tangent space than the gradient. Among all methods, Input $\times$ Gradient achieves the highest fraction in tangent space. The qualitative example on the left indicates that Input $\times$ Gradient indeed provides the most meaningful explanations. For instance, for the upper-left image of the frog, while other methods seem to focus on only some parts of frog along with the background, Input $\times$ Gradient focuses on the central region covering the entire frog. This is similar to the low-dimensional MNIST task (MNIST32), where Input $\times$ Gradient is among the best performing methods, indicating that SmoothGrad has little impact in lower dimensions.

Figure 3 middle row shows the performance of gradient-based explanation methods on the pneumonia dataset. The quantitative plots on the right show the distribution of the fraction in tangent space for different methods on all the test images. We observe that in general the fraction is highest for SmoothGrad and lowest for raw gradients. The qualitative examples on the left indicate that SmoothGrad indeed focuses on the relevant region near the lungs to make the correct predictions, whereas raw gradients do not seem to focus on the lungs. Similarly, Figure 3 bottom row shows the performance of gradient-based explanation methods on the diabetic retinopathy dataset. Again, from

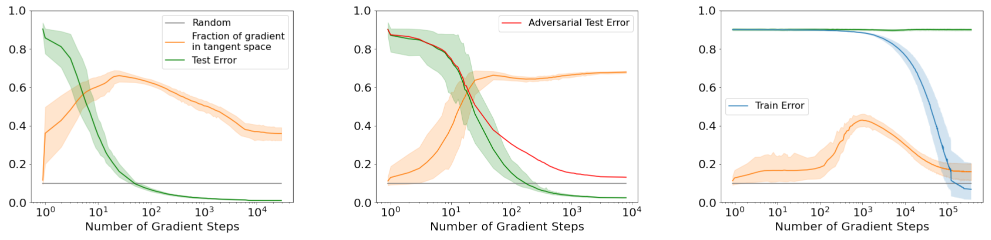

Figure 4: Fraction of gradient in tangent space evolving over the course of training. Mean and 90% confidence bounds. Gray line shows the expected fraction of a random vector in tangent space. Logarithmic x-axis. (Left) Training with Adam (Kingma & Ba, 2014). (Center) PGD Adv. Robust Training. (Right) Training with Adam and random labels.

qualitative examples shown on left we see that SmoothGrad correctly focuses on the region away from the pupil to explain the diagnosis but other methods, such as gradients, wrongly focus on the pupil to explain the decision. Both the qualitative and quantitative results indicate that SmoothGrad has a strong impact in high dimensions.

In conclusion, we find that the fraction of gradient based explanations in the tangent space is much higher than the expectation of fraction of random vectors in the tangent space. We also observe that for complex data manifolds (as described in this section), the alignment of different explanation methods is consistent with the literature (i.e., raw gradients being worse than other methods, with SmoothGrad being the best in high-dimensional setup).

4 When and why are gradients aligned with the data manifold?

When and why are model gradients aligned with the data manifold? That there is some degree of alignment is plausible: After all, finite differences between the points in the training data serve as rough approximations of the data manifold. For a more detailed perspective, we study how the fraction of model gradients in tangent space evolves over the course of training (Figure 4). The left part of Figure 4 depicts the evolution of model gradients over the course of training on the MNIST32 dataset and the model from Section 3.1. First, we see that the fraction of model gradients in tangent space *at initialization* is only as good as random. Second, and somewhat surprisingly, we see that the fraction in tangent space increases rapidly during the early steps of training, almost before the model starts to fit the labels (note that the x-axis in Figure 4 is logarithmic). Moreover, the relation between the data manifold and model gradients deteriorates as the model increasingly fits the labels.

4.1 The effect of adversarial training

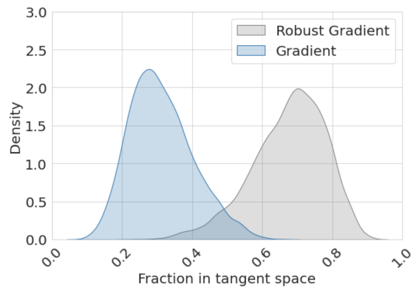

Figure 5: Adv. Robust Gradients.

Why does the relation between the data manifold and model gradients deteriorate as the model increasingly fits the labels? For insights on this question, we turn to training with adversarial perturbations. Previous work has noted that adversarial training aligns model gradients with human perception and the data manifold (Tsipras et al., 2018; Kim et al., 2019). In line with our hypothesis, we expect that adversarial training increases the fraction of model gradients in tangent space. An evaluation on the MNIST32 dataset shows that this is indeed the case. Figure 5 depicts the fraction of model gradients in tangent space, both for the model from section 3.1, and for the same architecture trained with projected gradient descent (PGD) against an l_2-adversary (Madry et al., 2017). Details regarding the adversarial training procedure are in Appendix A. The effect of adversarial training is quite remarkable: the mean fraction of gradients in tangent space is 0.68 for the adversarially trained model, compared with 0.31 for the standard model. The fraction of adversarially robust gradients in tangent space is not being improved by any of the post-hoc explanation methods considered in this paper. As depicted in the center part of Figure 4, the fraction of model gradients in tangent space does not deteriorate under adversarial training, but instead increases monotonically. We hypothesize that the observed behavior under standard training is due to the presence of non-robust features that are not aligned with the data manifold (Ilyas et al., 2019).

4.2 Training with random labels

There are two different ways in which a model might align its gradients with the data manifold: (1) The model leverages information about the data manifold in an unsupervised way. After all, the data manifold is an unsupervised object that exists independently of the labels. (2) The model leverages information about the data manifold implicitly through the labels. After all, in natural image classification problems, class boundaries might be implicitly aligned with the data manifold. We attempt to distinguish between these two different possibilities by training with random labels. The result of this experiment is depicted in the right part of Figure 4. First, the fraction of gradients in tangent space increase even when training with random labels. Second, it reaches its peak just before the network starts to fit the labels. Third, as the network increasingly (over-)fits the data, the relation between model gradients and the data manifold deteriorates until it remains hardly better than random. This experiment shows that the model does indeed, at least in part, leverage information about the data manifold in an unsupervised way. However, it also indicates that this relation might be hard to sustain if the labels are unrelated to the structure of the data manifold.

5 Generalization does not imply alignment with the manifold

We have already seen empirically that the relation between model gradients and the data manifold can exhibit considerable local variation (Figures 2 and 3). This is the case even for models that achieve an accuracy $> 99\%$ (Section 3.1). In this section, we are going to formally prove that a *single* neural network that achieves a test accuracy of 100% can exhibit an arbitrary amount of variation between its gradients and the data manifold (Theorem 1). This is illustrated in Figure 6. Figure 6 depicts the fraction of model gradients in tangent space for a two-layer neural network trained to solve a classification problem that lives on a 1-dimensional data manifold in $\mathbb{R}^3$. For this network, the distribution of model gradients in tangent space is bi-modal with two modes near 0 and 1. This means that (a) for 50% of observations, model gradients lie within the tangent space of the data manifold, and (b) for the remaining 50% of observations, model gradients are orthogonal to the tangent space of the data manifold.

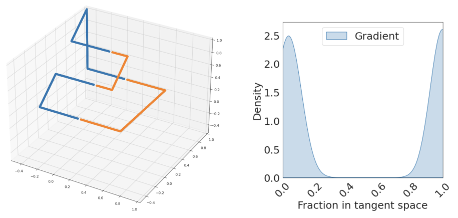

Figure 6: $\mathcal{M}_3$ and Simulation Results.

To formally prove this result, we leverage the recently demonstrated connections between the training dynamics of infinite width neural networks and Wasserstein gradient flow (Chizat & Bach, 2018; 2020). In particular, Chizat & Bach (2020) show that the limit of Wasserstein gradient flow, conditional on convergence, takes the form of a maximum-margin classifier

$$\nu^\star = \underset{\nu \in \mathcal{P}(\mathbb{S}^{d+1})}{\arg\max} \min_{(x,y) \in \mathcal{D}} y \cdot f(\nu, x) \qquad (2)$$

where $f(\nu, x) = \mathbb{E}_{(w,a,b)\sim\nu}\, w\phi(\langle a, x\rangle + b)$ and $\phi(x) = \max\{x, 0\}$. We show that there exists a manifold and corresponding classification such that the gradient of (2) is partly perfectly and partly not at all aligned with the manifold.

Theorem 1 (Generalization does not imply alignment with the manifold). *For every dimension $d > 1$, there exists a 1-dimensional manifold $\mathcal{M}_d \subset \mathbb{R}^d$ that does not lie in any proper subspace, a probability distribution $\mathcal{D}$ on $\mathcal{M}_d \times \{-1, 1\}$ and a maximum-margin classifier with zero test error given by equation (2) such that*

$$\mathbb{P}_{(x,y)\sim\mathcal{D}}\left(\frac{\partial f(\nu^\star, x)}{\partial x} \in \mathcal{T}_x\right) > 0.49 \qquad \textit{and} \qquad \mathbb{P}_{(x,y)\sim\mathcal{D}}\left(\frac{\partial f(\nu^\star, x)}{\partial x} \in \mathcal{T}_x^\perp\right) > 0.49.$$

The proof is in appendix B, we follow Shah et al. (2021). It would have been equally possible to provide examples where the gradient is always or never aligned with the data manifold. We choose this particular example because we believe that it is related to what is observed in practice.

6 Related Work

Explanation methods. Methods such as layer conductance (Dhamdhere et al., 2018), internal influence (Leino et al., 2018), and GradCam (Selvaraju et al., 2017) attempt to attribute model predictions to a certain layer and are aptly called *layer attribution methods*. The work in (Singla et al., 2020) proposed a method to explain the classifier by generating visual perturbation of an image by exaggerating or diminishing the semantic features that the classifier associates with a target label. Work on causal explanation such as (Schwab & Karlen, 2019) formulates the problem of explaining a machine learning model as a causal learning task that estimates the influence of certain inputs on the outputs of another machine learning model. Moreover, work such as (Yang & Song, 2020) uses models based on differentiable inductive logic programming that learns first-order logic to explain the data. The work in (Yeh et al., 2020) proposes a method to find set of concepts that are "sufficient" to explain predictions. Koh et al. (2020) build a model where concepts are built-in so that one can control influential concepts.

Evaluating explanations. It is important to objectively assess how good an explanation is. However, unavailability of ground-truth explanations makes it difficult to evaluate them (Samek et al., 2021). Evaluation is further complicated by the fact that explanations may not be reliable as they are susceptible to adversarial attacks (Heo et al., 2019; Dombrowski et al., 2019). Some of the existing work in this direction focus on (i) *Faithfulness/Sufficiency* and (ii) *Human interpretability*. Works on faithfulness propose that an explanation should reliably and comprehensively represent the local decision structure of the analyzed ML model. A practical technique to assess such property of the model, in case of image data, is "pixel-flipping" (Samek et al., 2016), it tests whether removing the features highlighted by the explanation (as most relevant) leads to a strong decay of the network prediction abilities. Works on human interpretability emphasize that the presented explanation techniques deliver results that are meaningful to the human (Samek et al., 2021). It is difficult to define human interpretability as different users may have different capabilities at reading explanations and at making sense of the features that support them (Ribeiro et al., 2016; Narayanan et al., 2018; Miller, 2019).

7 Discussion

Recently, there has been a lot of discussion about post-hoc explanation methods, with some authors claiming that they work relatively well (Samek et al., 2021), while others assert that they should not be used at all (Rudin, 2019). In this work, we focus on the particular case of gradient-based input attribution methods and propose a necessary criterion: explanations should be aligned with the tangent space of the data manifold.

The objective of this paper is *not* to claim that the gradients of existing models provide good explanations or that any particular post-hoc explanation method works especially well. Instead, our goal is to convince the reader that the manifold hypothesis for gradient-based explanations deserves consideration, and to start exploring the potential implications of this hypothesis. We hope that our hypothesis can provide a novel perspective on gradient-based explanations, and to serve a criterion for future investigations. As we demonstrate in Section 5, the question of whether model gradients are aligned with the data manifold is amendable to theoretical analysis. Future work might ask under what conditions adversarial training causes model gradients to be aligned with the data manifold, or when Integrated Gradients can of improve the fraction of model gradients in tangent space. Moreover, while current models and algorithms provide only imperfect alignment, it is an open question whether this is due to the fact that we have not yet found the right algorithm, or because the problem is harder than classification.

Although we are, to the best of our knowledge, the first to provide a systematic evaluation of our hypothesis, we believe that it is implicit in previous works. Projections of model gradients on estimates of the tangent space have long been performed in the literature (Kim et al., 2019; Dombrowski et al., 2019). Our criterion might also be implicitly related to other sanity checks for explanations (Adebayo et al., 2018a). For example, since model gradients at random initialization are unrelated to the data manifold, an explanation that does not change when model weights are being randomized can also not lie in the tangent space of the data manifold.

8 Ethics and Reproducability Statements

Explainability is an important demand when it comes to applications of machine learning, for example in medicine or in societal contexts. This has been stressed by the European Data Protection Regulation, and will be an important aspect of future AI legislation. However, the field of explainable machine learning is still widely open, and formal guarantees about explanation algorithms rarely exist. Our work takes one step towards understanding conceptually how particular explanation methods work. Many more such steps will be necessary before we actually understand which kind of guarantees can possibly be given for explanation algorithms.

Regarding our experimental setup: The properties of the data manifolds of natural image datasets depend on data collection practices and other design choices made during dataset creation. In this work, we use standard open-sourced datasets that do not require ethical approval.

All details regarding our experimental setup can be found in appendix A. The same is true for all model architectures, as well as the details of neural network training. Python code for computing the tangent space and our model architectures can be found in appendix D. A formal proof of Theorem 1 is in appendix B.

References

Eddie Aamari and Clément Levrard. Nonasymptotic rates for manifold, tangent space and curvature estimation. *The Annals of Statistics*, 2019.

Julius Adebayo, Justin Gilmer, Ian Goodfellow, and Been Kim. Local explanation methods for deep neural networks lack sensitivity to parameter values. *arXiv preprint arXiv:1810.03307*, 2018a.

Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. *arXiv preprint arXiv:1810.03292*, 2018b.

Sushant Agarwal, Shahin Jabbari, Chirag Agarwal, Sohini Upadhyay, Zhiwei Steven Wu, and Himabindu Lakkaraju. Towards the unification and robustness of perturbation and gradient based explanations. *arXiv preprint arXiv:2102.10618*, 2021.

Amine Amyar, Romain Modzelewski, Hua Li, and Su Ruan. Multi-task deep learning based ct imaging analysis for covid-19 pneumonia: Classification and segmentation. *Computers in Biology and Medicine*, 2020.

Marco Ancona, Enea Ceolini, Cengiz Öztireli, and Markus Gross. Towards better understanding of gradient-based attribution methods for deep neural networks. *International Conference on Learning Representations*, 2018.

Christopher Anders, Plamen Pasliev, Ann-Kathrin Dombrowski, Klaus-Robert Müller, and Pan Kessel. Fairwashing explanations with off-manifold detergent. In *International Conference on Machine Learning*, 2020.

Nishanth Arun, Nathan Gaw, Praveer Singh, Ken Chang, Mehak Aggarwal, Bryan Chen, Katharina Hoebel, Sharut Gupta, Jay Patel, Mishka Gidwani, et al. Assessing the (un) trustworthiness of saliency maps for localizing abnormalities in medical imaging. *arXiv preprint arXiv:2008.02766*, 2020.

Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLOS ONE*, 2015.

Christopher P Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. Understanding disentangling in β-vae. *arXiv preprint arXiv:1804.03599*, 2018.

Lawrence Cayton. Algorithms for manifold learning. *Univ. of California at San Diego Tech. Rep*, 2005.

Chaofan Chen, Oscar Li, Chaofan Tao, Alina Jade Barnett, Jonathan Su, and Cynthia Rudin. This looks like that: deep learning for interpretable image recognition. *arXiv preprint arXiv:1806.10574*, 2018a.

Ricky TQ Chen, Xuechen Li, Roger Grosse, and David Duvenaud. Isolating sources of disentanglement in variational autoencoders. *arXiv preprint arXiv:1802.04942*, 2018b.

Mohamed Chetoui and Moulay A Akhloufi. Explainable diabetic retinopathy using efficientnet. In *IEEE Engineering in Medicine & Biology Society (EMBC)*, 2020.

Lenaic Chizat and Francis Bach. On the global convergence of gradient descent for over-parameterized models using optimal transport. *arXiv preprint arXiv:1805.09545*, 2018.

Lenaic Chizat and Francis Bach. Implicit bias of gradient descent for wide two-layer neural networks trained with the logistic loss. In *Conference on Learning Theory*, 2020.

Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. Emnist: Extending mnist to handwritten letters. In *2017 International Joint Conference on Neural Networks (IJCNN)*, 2017.

Kedar Dhamdhere, Mukund Sundararajan, and Qiqi Yan. How important is a neuron? *arXiv preprint arXiv:1805.12233*, 2018.

Ann-Kathrin Dombrowski, Maximillian Alber, Christopher Anders, Marcel Ackermann, Klaus-Robert Müller, and Pan Kessel. Explanations can be manipulated and geometry is to blame. *Neural Information Processing Systems*, 2019.

Charles Fefferman, Sanjoy Mitter, and Hariharan Narayanan. Testing the manifold hypothesis. *Journal of the American Mathematical Society*, 2016.

Damien Garreau and Dina Mardaoui. What does lime really see in images? *arXiv preprint arXiv:2102.06307*, 2021.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

Seyyed Hossein Hasanpour, Mohammad Rouhani, Mohsen Fayyaz, and Mohammad Sabokrou. Lets keep it simple, using simple architectures to outperform deeper and more complex architectures. *arXiv preprint arXiv:1608.06037*, 2016.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE conference on computer vision and pattern recognition*, 2016.

Juyeon Heo, Sunghwan Joo, and Taesup Moon. Fooling neural network interpretations via adversarial model manipulation. *Neural Information Processing Systems*, 2019.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *ICLR*, 2017.

Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. *arXiv preprint arXiv:1905.02175*, 2019.

Daniel S Kermany, Michael Goldbaum, Wenjia Cai, Carolina CS Valentim, Huiying Liang, Sally L Baxter, Alex McKeown, Ge Yang, Xiaokang Wu, Fangbing Yan, et al. Identifying medical diagnoses and treatable diseases by image-based deep learning. *Cell*, 172(5):1122–1131, 2018.

Beomsu Kim, Junghoon Seo, and Taegyun Jeon. Bridging adversarial robustness and gradient interpretability. *arXiv preprint arXiv:1903.11626*, 2019.

Pieter-Jan Kindermans, Sara Hooker, Julius Adebayo, Maximilian Alber, Kristof T Schütt, Sven Dähne, Dumitru Erhan, and Been Kim. The (un) reliability of saliency methods. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. 2019.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. Concept bottleneck models. In *International Conference on Machine Learning*, 2020.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

Klas Leino, Shayak Sen, Anupam Datta, Matt Fredrikson, and Linyi Li. Influence-directed explanations for deep convolutional networks. In *2018 IEEE International Test Conference*, 2018.

Dohun Lim, Hyeonseok Lee, and Sungchan Kim. Building reliable explanations of unreliable neural networks: Locally smoothing perspective of model interpretation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2021.

Juan Eduardo Luján-García, Cornelio Yáñez-Márquez, Yenny Villuendas-Rey, and Oscar Camacho-Nieto. A transfer learning method for pneumonia classification and visualization. *Applied Sciences*, 2020.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial intelligence*, 2019.

Menaka Narayanan, Emily Chen, Jeffrey He, Been Kim, Sam Gershman, and Finale Doshi-Velez. How do humans understand explanations from machine learning systems? an evaluation of the human-interpretability of explanation. *arXiv preprint arXiv:1802.00682*, 2018.

Sivaramakrishnan Rajaraman, Sema Candemir, George Thoma, and Sameer Antani. Visualizing and explaining deep learning predictions for pneumonia detection in pediatric chest radiographs. In *Medical Imaging 2019: Computer-Aided Diagnosis*, 2019.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 2019.

Wojciech Samek, Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, and Klaus-Robert Müller. Evaluating the visualization of what a deep neural network has learned. *IEEE transactions on neural networks and learning systems*, 2016.

Wojciech Samek, Grégoire Montavon, Sebastian Lapuschkin, Christopher J Anders, and Klaus-Robert Müller. Explaining deep neural networks and beyond: A review of methods and applications. *Proceedings of the IEEE*, 2021.

Rory Sayres, Ankur Taly, Ehsan Rahimy, Katy Blumer, David Coz, Naama Hammel, Jonathan Krause, Arunachalam Narayanaswamy, Zahra Rastegar, Derek Wu, et al. Using a deep learning algorithm and integrated gradients explanation to assist grading for diabetic retinopathy. *Ophthalmology*, 2019.

Patrick Schwab and Walter Karlen. Cxplain: Causal explanations for model interpretation under uncertainty. *Neural Information Processing Systems*, 2019.

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 *IEEE International Conference on Computer Vision*, 2017.

Harshay Shah, Prateek Jain, and Praneeth Netrapalli. Do input gradients highlight discriminative features? *arXiv preprint arXiv:2102.12781*, 2021.

Hang Shao, Abhishek Kumar, and P Thomas Fletcher. The riemannian geometry of deep generative models. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018.

Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *International Conference on Machine Learning*, 2017.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *In Workshop at International Conference on Learning Representations*, 2014.

Sumedha Singla, Brian Pollack, Junxiang Chen, and Kayhan Batmanghelich. Explanation by progressive exaggeration. In *International Conference on Learning Representations*, 2020.

Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*, 2017.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International Conference on Machine Learning*, 2017.

Natasa Tagasovska, Damien Ackerer, and Thibault Vatter. Copulas as high-dimensional generative models: Vine copula autoencoders. *arXiv preprint arXiv:1906.05423*, 2019.

Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. *arXiv preprint arXiv:1805.12152*, 2018.

Arash Vahdat and Jan Kautz. Nvae: A deep hierarchical variational autoencoder. *arXiv preprint arXiv:2007.03898*, 2020.

Toon Van Craenendonck, Bart Elen, Nele Gerrits, and Patrick De Boever. Systematic comparison of heatmapping techniques in deep learning in the context of diabetic retinopathy lesion detection. *Translational vision science & technology*, 2020.

Kilian Q Weinberger and Lawrence K Saul. Unsupervised learning of image manifolds by semidefinite programming. *International journal of computer vision*, 2006.

Yuan Yang and Le Song. Learn to explain efficiently via neural logic inductive learning. In *International Conference on Learning Representations*, 2020.

Chih-Kuan Yeh, Been Kim, Sercan Arik, Chun-Liang Li, Tomas Pfister, and Pradeep Ravikumar. On completeness-aware concept-based explanations in deep neural networks. *Neural Information Processing Systems*, 2020.

A Model Architectures and Training Details

A.1 MNIST32

We first describe the creation of the MNIST32 dataset. We autoencoded the original MNIST dataset with a β-TCVAE (Chen et al., 2018b) and the same architecture as Burgess et al. (2018). The hyperparameters were $\alpha = \gamma = 1$, $\beta = 6$. We use `https://github.com/YannDubs/disentangling-vae`. On the reconstructed images, we trained a SimpleNet-V1 to replicate the original labels (Hasanpour et al., 2016). Training with Adam and a learning rate of $1e-3$ allowed to replicated the labels of the test images with an accuracy of 96%. To increase the quality of the generated images, we additionally applied rejection sampling based on the the softmax score of the class predicted by the SimpleNet. Every sample from the autoencoder was accepted with probability p^2_{softmax}. Random samples from the MNIST32 dataset are depicted in the left part of Figure 7.

On the MNIST32 dataset, we trained the default model architecture from `https://github.com/pytorch/examples/tree/master/mnist`. We trained for 50 epochs with Adam, an initial learning rate of $1e-4$ and learning rate decay of $1e-1$ after 10 epochs each.

Adversarially robust training on MNIST32 was perfomed as follows. We trained the same model architecture against an l_2-adversary with projected gradient descent (PGD). For each gradient step, the size of the adversarial perturbation ϵ was randomly chosen from $[1, 4, 8]$ and we took 100 iterations with a step size of $\alpha = 2.5\epsilon/100$ each (Madry et al., 2017).

To overfit the MNIST32 dataset with random labels, we disabled the dropout layers of the neural network. We then trained for 3000 epochs with Adam, and intial learning rate of $1e-4$ and learning rate decay of $1e-1$ after 1000 epochs each.

A.2 MNIST256

To create the MNIST256 dataset, we appended a bilinear upsampling layer to the decoder that was used to generate the MNIST32 dataset. Note that bilinear upsampling is differentiable, which is required to compute the tangent spaces. Random samples from the MNIST256 dataset are depicted in the right part of Figure 7.

On the MNIST256 dataset, we trained a ResNet18 for 50 epochs with Adam, an initial learning rate of $1e-2$ and a learning rate decay of $1e-1$ after 10 epochs each.

A.3 EMNIST128

The EMNIST dataset is a set of handwritten character digits derived from the NIST Special Database 19 and converted to a 28×28 pixel image format and dataset structure that directly matches the MNIST dataset. We used the dataset as available from PyTorch `https://pytorch.org/vision/stable/datasets.html#emnist`. The images were resized to 128×128 to make it a high-dimensional problem and we used a subset consisting of 60 classes (in contrast to other experiments where number of classes are typically low). We trained an autoencoder using the reconstruction approach and encoder-decoder architecture as shown in Section D using Adam optimizer with learning rate set to 1e-4, decayed over 200 epochs using cosine annealing. We then train a VGG network to perform the classification, in a similar manner.

A.4 CIFAR10

The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. We use the dataset available directly from the PyTorch dataloaders as described here `https://pytorch.org/vision/stable/datasets.html#cifar`. To learn the manifold, we use the reconstruction approach using the autoencoder shown in Section D with the latent dimension set to be 144 with $\sqrt{k/d} \approx 0.20$. We use the Adam optimizer with learning rate set to 1e-4 decayed using cosine annealing over 200 epochs to learn the autoencoder. We then use the VGG16 to learn a classifier again trained using Adam optimizer with learning rate set to 1e-4 decayed using cosine annealing.

A.5 Pneumonia Detection

The original dataset at https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia contains high-resolution chest X-ray images with 2 classes: Normal and Pneumonia (with pneumonia being of two types viral and bacterial, but within class distinction of pneumonia is not considered in this problem). The problem is posed as a binary classification problem to decide between the normal and an abnormal class (pneumonia). The images were resized to $1 \times 256 \times 224$ (i.e., 57344 dimensional data) and the autoencoder shown in Section D is used to learn the manifold of the images where the latent dimension is reduced to $8 \times 28 \times 32$ (i.e., 7168) with $\sqrt{k/d} = 0.20$, we then fine tune a Resnet18 model (previously trained on ImageNet) to perform the classification using a learning rate of 1e-4, decayed with cosine annealing over 200 epochs and using Adam optimizer.

A.6 Diabetic Retinopathy Detection

The original 3 channel (RGB) fundus image dataset at https://www.kaggle.com/c/diabetic-retinopathy-detection contains 5 classes with varying degrees of diabetic retinopathy. We posed the problem as a binary classification problem to decide between the normal and an abnormal class. The images were resized to $3 \times 224 \times 224$ (i.e., 150528 dimensional data) and the autoencoder shown in Section D is used to learn the manifold of the images where the latent dimension is reduced to $8 \times 28 \times 28$ (i.e., 6272) with $\sqrt{k/d} = 0.20$, we then fine tune a Resnet18 model (previously trained on ImageNet) to perform the classification using a learning rate of 2e-4, decayed with cosine annealing over 150 epochs and using Adam optimizer.

B Proof of Theorem 1

Proof. Let $d > 1$. We begin by defining the manifold. Let

$$t_{\max} = \begin{cases} 4(d-2) & \text{if } d \text{ is even} \\ 4(d-1) & \text{if } d \text{ is odd.} \end{cases} \tag{3}$$

For $t \in [0, t_{\max}]$, consider the continuous curve $f(t)$ that walks along the edges of the shifted hypercube, alternating between the first and other dimensions

$$f(t) = \begin{cases} (-1/2 + (t - \lfloor t \rfloor), \underbrace{1, \dots, 1}_{\lfloor t/2 \rfloor}, \underbrace{0, \dots, 0}_{d-1-\lfloor t/2 \rfloor})^t & \text{if } \lfloor t \rfloor \mod 4 = 0 \\ (1/2, \underbrace{1, \dots, 1}_{\lfloor t/2 \rfloor}, t - \lfloor t \rfloor, \underbrace{0, \dots, 0}_{d-2-\lfloor t/2 \rfloor})^t & \text{if } \lfloor t \rfloor \mod 4 = 1 \\ (1/2 - (t - \lfloor t \rfloor), \underbrace{1, \dots, 1}_{\lfloor t/2 \rfloor}, \underbrace{0, \dots, 0}_{d-1-\lfloor t/2 \rfloor})^t & \text{if } \lfloor t \rfloor \mod 4 = 2 \\ (-1/2, \underbrace{1, \dots, 1}_{\lfloor t/2 \rfloor}, t - \lfloor t \rfloor, \underbrace{0, \dots, 0}_{d-2-\lfloor t/2 \rfloor})^t & \text{if } \lfloor t \rfloor \mod 4 = 3. \end{cases} \tag{4}$$

In all dimensions, $f(t)$ starts at $(-1/2, 0, \dots, 0)^t$. If d is even, $f(t)$ ends at $(-1/2, 1, \dots, 1, 0)^t$. If d is odd, $f(t)$ ends at $(-1/2, 1, \dots, 1)^t$. In even dimensions, connect the endpoint $(-1/2, 1, \dots, 1, 0)^t$ to the starting point via straight lines to the corner points $(\sqrt{d-1}/2, 1, \dots, 1, 0)^t$, $(\sqrt{d-1}/2, 1, \dots, 1, 1)^t$, and $(-1/2, 1, \dots, 1, 1)^t$ In odd dimensions, connect the endpoint $(-1/2, 1, \dots, 1)^t$ to the starting point via straight lines to the corner points $(-1/2, 2/3, \dots, 2/3)^t$, $(-1/2 + \sqrt{d-1}/2, 2/3, \dots, 2/3)^t$, $(-1/2 + \sqrt{d-1}/2, 1/3, \dots, 1/3)^t$, $(-1/2, 1/3, \dots, 1/3)^t$. The whole point of this construction is to obtain a closed connected curve that does not lie in any proper subspace and that walks exactly half of the time along the first coordinate, and the rest of the time orthogonal to it. By smoothing the corners of this connected curve, we obtain a smooth connected manifold $\mathcal{M}$.

Let $U_{\mathcal{M}}$ be the uniform distribution on $\mathcal{M}$. Let $\mathcal{D}_x$ be given by $\mathcal{D}_x(A) = U_{\mathcal{M}}(A/M)/(1-U_{\mathcal{M}}(M))$ where $M = \{x \in \mathcal{M} : |x_1| < \epsilon\}$. Let the label be given by $y = \operatorname{sign} x_1$. The separating hyperplane with maximum margin is $x_1 = 0$. We claim that $\nu^\star = \frac{1}{2}\delta_{\theta_0^\star} + \frac{1}{2}\delta_{\theta_1^\star}$, $\theta_0^\star = \left(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2(1+\epsilon^2)}}, 0, \ldots, 0, \frac{\epsilon}{\sqrt{2(1+\epsilon^2)}}\right)^t$, $\theta_1^\star = \left(\frac{-1}{\sqrt{2}}, \frac{-1}{\sqrt{2(1+\epsilon^2)}}, 0, \ldots, 0, \frac{\epsilon}{\sqrt{2(1+\epsilon^2)}}\right)^t$, is a maximizer of

$$\underset{\nu \in \mathcal{P}(\mathbb{S}^{d+1})}{\arg\max} \min_{(x,y)\in\mathcal{D}} y \cdot f(\nu, x). \tag{5}$$

By Proposition 12 in (Chizat & Bach, 2020), we have to show that there exists a measure $p^\star$ on $\mathcal{M}$ (the support vectors) such that

$$\text{Support}(\nu^\star) \in \underset{(w,a,b)\in\mathbb{S}^{d+1}}{\arg\max} \mathbb{E}_{(x,y)\sim p^\star} \left(y \cdot w\phi(\langle a, x\rangle + b)\right) \tag{6}$$

and

$$\text{Support}(p^\star) \in \underset{(x,y)\in\mathcal{D}}{\arg\min} \mathbb{E}_{(w,a,b)\sim\nu^\star} \left(y \cdot w\phi(\langle a, x\rangle + b)\right). \tag{7}$$

We claim that $p^\star$ is given by

$$p^\star = \frac{1}{2}\delta_{(-\epsilon,0,\ldots,0)^t} + \frac{1}{2}\delta_{(\epsilon,0,\ldots,0)^t}. \tag{8}$$

We first show (6). It holds that

$$\mathbb{E}_{(x,y)\sim p^\star} \left(y \cdot w\phi(\langle a, x\rangle + b)\right) = \frac{w}{2}\left(\phi(a_1\epsilon + b) - \phi(-a_1\epsilon + b)\right).$$

We differentiate two cases. Note that $\theta_0^\star$ achieves an objective larger than zero, hence $a_1 \neq 0$.

Case 1, $a_1 > 0$**.** If $a_1 > 0$, then $\phi(a_1\epsilon + b) > \phi(-a_1\epsilon + b)$. This implies $b \geq 0$ and $b \leq a_1\epsilon$. The maximization problem can then be written as

$$\begin{aligned} \max_{w,a_1,b} \quad & \frac{w}{2}a_1\epsilon + \frac{w}{2}b \\ \text{subject to} \quad & w^2 + a_1^2 + b^2 = 1 \\ & 0 \leq b \leq a_1\epsilon \\ & a_1 > 0. \end{aligned}$$

For ϵ small enough, the unique solution is given by $b = a_1\epsilon$, $w = \frac{1}{\sqrt{2}}$ and $a_1 = \frac{1}{\sqrt{2(1+\epsilon^2)}}$, i.e. by $\theta_0^\star$. The objective is $1/(2\sqrt{1+\epsilon^2})$.

Case 2, $a_1 < 0$**.** If $a_1 < 0$, then $\phi(a_1\epsilon + b) < \phi(-a_1\epsilon + b)$. This implies $b \geq 0$ and $b \leq -a_1\epsilon$. The maximization problem can now be written as

$$\begin{aligned} \max_{w,a_1,b} \quad & -\frac{w}{2}a_1\epsilon + \frac{w}{2}b \\ \text{subject to} \quad & w^2 + a_1^2 + b^2 = 1 \\ & 0 \leq b \leq -a_1\epsilon \\ & a_1 < 0. \end{aligned}$$

For ϵ small enough, the unique solution is given by $b = -a_1\epsilon$, $w = \frac{1}{\sqrt{2}}$, and $a_1 = \frac{1}{\sqrt{2(1+\epsilon^2)}}$, i.e. by $\theta_1^\star$. The objective is again $1/(2\sqrt{1+\epsilon^2})$. This shows (6).

We now show (7). Explicit computation shows

$$\mathbb{E}_{(w,a,b)\sim\nu^\star} \left(y \cdot w\phi(\langle a, x\rangle + b)\right) = \frac{y}{\sqrt{2}}\phi\left(\frac{x_1+\epsilon}{\sqrt{2(1+\epsilon^2)}}\right) - \frac{y}{\sqrt{2}}\phi\left(\frac{-x_1+\epsilon}{\sqrt{2(1+\epsilon^2)}}\right).$$

For $y = 1$, $x_1 \geq \epsilon$ and the second term vanishes. The minimum is then attained iff $x_1 = \epsilon$. For $y = -1$, $x_1 \leq \epsilon$ and the first term vanishes. The minimum is then attained iff $x_1 = -\epsilon$. This proves (7).

We now compute the gradient of f. We have

$$f(\nu^\star, x) = \frac{1}{\sqrt{2}}\phi\left(\frac{x_1+\epsilon}{\sqrt{2(1+\epsilon^2)}}\right) - \frac{1}{\sqrt{2}}\phi\left(\frac{-x_1+\epsilon}{\sqrt{2(1+\epsilon^2)}}\right). \tag{9}$$

Thus, for $i > 1$,

$$\frac{\partial f(\nu^\star, x)}{\partial x_i} = 0. \tag{10}$$

For $i = 1$,

$$\frac{\partial f(\nu^\star, x)}{\partial x_1} = \frac{1}{2\sqrt{1+\epsilon^2}}. \tag{11}$$

Thus, the gradient of f is constant and a multiple of $(1, 0, \dots, 0)^t$.

Except at the corners, the tangent space of $\mathcal{M}$ is either given by span$\langle\, (1, 0, \dots, 0)^t\rangle$ or orthogonal to $(1, 0, \dots, 0)^t$. The proof is completed by noting that it is orthogonal to $(1, 0, \dots, 0)^t$ with probability 0.5, that we can smooth the corners in regions of arbitrarily small measure, and by choosing ϵ arbitrarily small. $\square$

C Plots and Figures

C.1 Figure Creation Details

Figures 2 and 8 were created by normalizing both vectors to unit norm and trimming the pixels with the largest 0.005% absolute values. Images within the same column lie within the same color space, i.e. equal colors imply equal values along the respective coordinates.

To create Figure 3 we use the following function which overlays the explanation maps on top of the images.

```python
def overlay_saliency_map(i1_rec, smap, alpha=0.5, thr=0.75, sigma=10,
    wt_lowhigh=0.2):
    smap = (smap-np.min(smap))/(np.max(smap) - np.min(smap))
    smap = np.mean(smap, axis=2)
    mask = smap > thr*np.max(smap)
    smap_high = mask*smap
    smap_bg = (1-mask)*smap
    ##
    smap_high = gaussian_filter(smap_high, sigma)
    smap_high = (smap_high-np.min(smap_high))/(np.max(smap_high) - np.min
    (smap_high))
    smap_bg = (smap_bg-np.min(smap_bg))/(np.max(smap_bg) - np.min(smap_bg
    ))
    smap = wt_lowhigh*smap_bg + (1-wt_lowhigh)*smap_high
    smap = (smap-np.min(smap))/(np.max(smap) - np.min(smap))
    smap = np.uint8(smap*255)
    ##
    smap = cv2.applyColorMap(smap, colormap=cv2.COLORMAP_JET)
    smap = cv2.cvtColor(smap, cv2.COLOR_BGR2RGB)
    ##
    i1_rec = np.uint8(i1_rec*255)
    smap = cv2.addWeighted(i1_rec, alpha, smap, 1-alpha, 0)
    return smap
```

Listing 1: Function used to create Figure 3.

C.2 Additional Plots and Figures

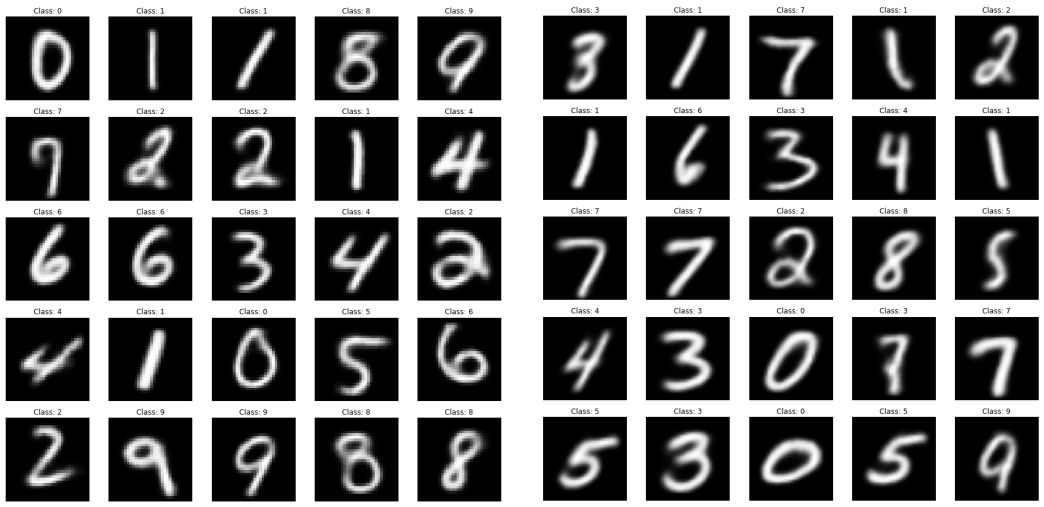

Figure 7: Random samples from the generated datasets. Left: MNIST32. Right: MNIST256.

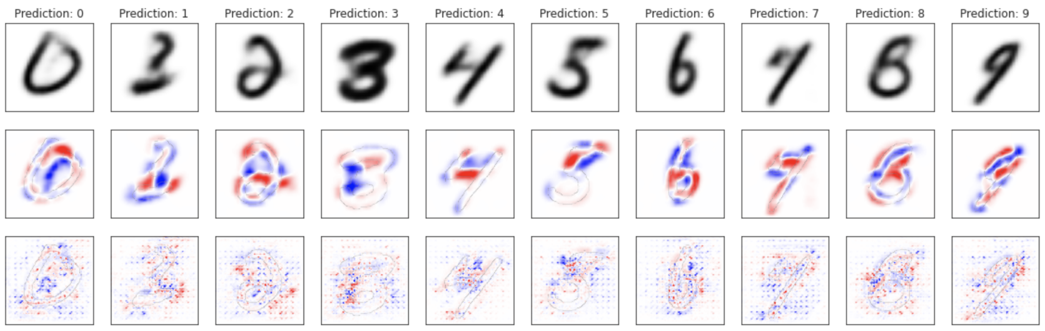

Figure 8: Images from the test set of the synthetic MNIST256 task. The second row depicts the part of the gradient that lies in tangent space. The third row depicts the part of the gradient that is orthogonal to the tangent space. Blue corresponds to negative attribution, red to positive attribution.

D Code Listings

```python

def compute_tangent_space(NetAE, z, device='cuda'):
    """ compute the tangent space of a generative model at  z

    NetAE: The decoder. A pytorch module that implements decode(z)
    z: pytorch tensor (latent vector)

    Returns: vectors that span the tangent space (tangent space dim, model output dim). These vectors correspond 1:1 to the latent dimensions of z.
    """
    assert len(z.shape) == 1, "compute_tangent_space: batch dimension in z is not supported. z has to be a 1-dimensional vector"
    NetAE.to(device)
    z = z.to(device)
    latent_dim = z.shape[0]
    z.requires_grad = True
    out = NetAE.decode(z)
    out = out.squeeze()      # remove singleton batch dimension
    output_shape = out.shape # store original output shape
    out = out.reshape(-1)    # and transform the output into a vector
    tangent_space = torch.zeros((latent_dim, out.shape[0]))
    for i in range(out.shape[0]):
        out[i].backward(retain_graph=True)
        tangent_space[:, i] = z.grad
        z.grad.zero_()
    tangent_space = tangent_space.reshape((-1, *output_shape)) # tangent space in model output shape
    return tangent_space
```

Listing 2: Code used to compute the tangent space of a generative model.

```python
class MNIST32Model(nn.Module):
    def __init__(self):
        super(MNIST32Model, self).__init__()
        self.conv1 = nn.Conv2d(1, 32, 3, 1)
        self.conv2 = nn.Conv2d(32, 64, 3, 1)
        self.dropout1 = nn.Dropout(0.25)
        self.dropout2 = nn.Dropout(0.5)
        self.fc1 = nn.Linear(12544, 128)
        self.fc2 = nn.Linear(128, 10)

    def forward(self, x):
        x = self.conv1(x)
        x = F.relu(x)
        x = self.conv2(x)
        x = F.relu(x)
        x = F.max_pool2d(x, 2)
        x = self.dropout1(x)
        x = torch.flatten(x, 1)
        x = self.fc1(x)
        x = F.relu(x)
        x = self.dropout2(x)
        x = self.fc2(x)
        return x
```

Listing 3: The model trained on MNIST32.

```python

class ResBlock(nn.Module):
    def __init__(self, in_channel, channel):
        super().__init__()

        self.conv = nn.Sequential(
            nn.ReLU(),
            nn.Conv2d(in_channel, channel, 3, padding=1),
            nn.ReLU(inplace=True),
            nn.Conv2d(channel, in_channel, 1),
        )

    def forward(self, input):
        out = self.conv(input)
        out += input

        return out
```

Listing 4: Convolutional residual block.

```python
class Encoder(nn.Module):
    def __init__(self, in_channel, channel, n_res_block, n_res_channel, stride):
        super().__init__()

        elif stride == 8:
            blocks = [
                nn.Conv2d(in_channel, channel//2, 4, stride=2, padding=1),
                nn.ReLU(inplace=True),
                nn.Conv2d(channel//2, channel//2, 4, stride=2, padding=1),
                nn.ReLU(inplace=True),
                nn.Conv2d(channel//2, channel, 4, stride=2, padding=1),
                nn.ReLU(inplace=True),
                nn.Conv2d(channel, channel, 3, padding=1),
            ]

        elif stride == 4:
            blocks = [
                nn.Conv2d(in_channel, channel//2, 4, stride=2, padding=1),
                nn.ReLU(inplace=True),
                nn.Conv2d(channel//2, channel, 4, stride=2, padding=1),
                nn.ReLU(inplace=True),
                nn.Conv2d(channel, channel, 3, padding=1),
            ]

        for i in range(n_res_block):
            blocks.append(ResBlock(channel, n_res_channel))

        blocks.append(nn.ReLU(inplace=True))

        self.blocks = nn.Sequential(*blocks)

    def forward(self, input):
        return self.blocks(input)
```

Listing 5: Convolutional encoder.

```
class Decoder(nn.Module):
    def __init__(
        self, in_channel, out_channel, channel, n_res_block, n_res_channel, stride
    ):
        super().__init__()

        blocks = [nn.Conv2d(in_channel, channel, 3, padding=1)]

        for i in range(n_res_block):
            blocks.append(ResBlock(channel, n_res_channel))

        blocks.append(nn.ReLU(inplace=True))

        elif stride == 8:
            blocks.extend(
                [
                    nn.ConvTranspose2d(channel, channel//2, 4, stride=2, padding=1),
                    nn.ReLU(inplace=True),
                    nn.ConvTranspose2d(channel//2, channel//2, 4, stride=2, padding=1),
                    nn.ReLU(inplace=True),
                    nn.ConvTranspose2d(
                        channel//2, out_channel, 4, stride=2, padding=1
                    ),
                ]
            )

        elif stride == 4:
            blocks.extend(
                [
                    nn.ConvTranspose2d(channel, channel//2, 4, stride=2, padding=1),
                    nn.ReLU(inplace=True),
                    nn.ConvTranspose2d(
                        channel//2, out_channel, 4, stride=2, padding=1
                    ),
                ]
            )

        self.blocks = nn.Sequential(*blocks)

    def forward(self, input):
        return self.blocks(input)
```

Listing 6: Convolutional decoder.

```
class AutoEncoder(nn.Module):
    def __init__(self, in_channel, out_channel, channel, n_res_block, n_res_channel, stride):
        super().__init__()

        self.E = Encoder(in_channel, channel, n_res_block, n_res_channel, stride)
        self.D = Decoder(channel, out_channel, channel, n_res_block, n_res_channel, stride)

    def forward(self, x):
        z = self.E(x)
        x_rec = self.D(z)
        return x_rec, z

    def encode(self, x):
        return self.E(x)

    def decode(self, z):
        return self.D(z)
```

Listing 7: Convolutional autoencoder used to estimate data manifolds.