Jake Callahan, Taylor Paskett CS 674
February 20, 2021 Dr. Wingate

Analyzing Information Loss of Invertible Variational Autoencoders

Jake Callahan, Taylor Paskett

February 20, 2021

1 Introduction

As we gain access to more and more unstructured data, unsupervised learning methods become more and more vital. Among existing unsupervised deep learning models, variational autoencoders have become prominent. An autoencoder uses an unlabelled dataset to create more compactly encoded representations of the data. This is done by introducing a bottleneck to the architecture, as the following image (courtesy of [3]) depicts:

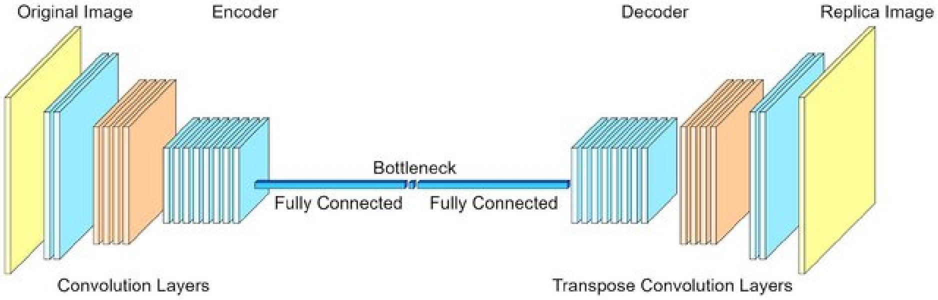


Figure 1: Illustration of Bottleneck in Autoencoder

Traditionally, autoencoders are trained by requiring that the original image and the replica image are nearly the same, measured using a reconstruction loss [1]. After training, we can extract an encoded version of a test image by inputting the test image to the encoder and extracting the value at the bottleneck layer.

More recently, information-theoretic concepts such as mutual information and entropy have been used to better understand autoencoders [5]; and in some cases, information-theoretic concepts can even be used in place of gradient descent to train neural networks [2].

Since information theory helps us understand and train autoencoders, we are interested in studying the information-theoretic properties of a new type of autoencoder that is built using invertible neural networks (INNs).

2 Invertible Autoencoders

We examine the work of Nguyen, Ardizzone, and Köthe, in their paper, ”Training Invertible Neural Networks as Autoencoders” [4].

In the following figure from Nguyen et al's work, they explain the fundamental building block of INNs: the invertible coupling layer. Just like a ResNet is built by stacking residual blocks, INNs are built by stacking invertible coupling layers. Theoretically, an INN can be built to any depth by stacking any number of invertible coupling layers.

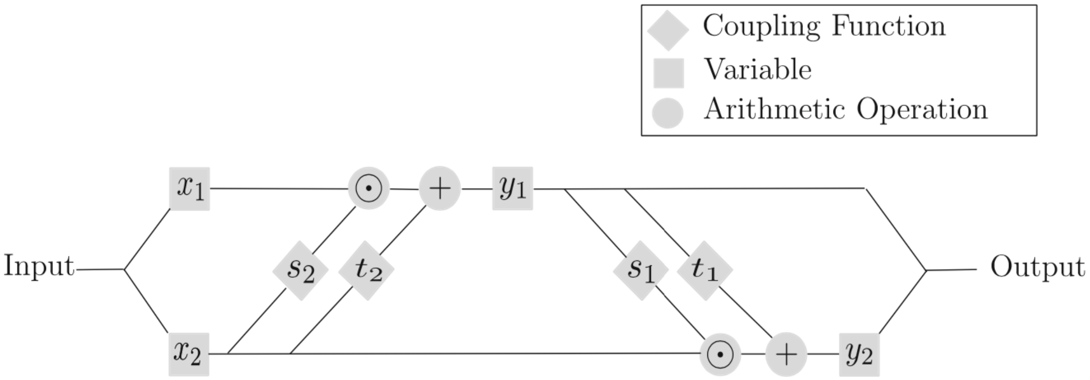


Figure 2: Visualization of the Invertible Coupling Layer

Although traditional neural networks are universal function approximators, INNs are limited to approximating bijective functions. The question, then, is this: if INNs can only approximate bijective functions, how can an INN become an autoencoder? Since the goal of an autoencoder is to encode data in a smaller-dimensional latent space, how can a bijection achieve this?

The main contribution of Nguyen, et al. was to provide a creative solution to this problem. They do so by ”zero-padding” the output of the INN. For example, suppose our inputs have a dimension of 1000, and we wish to create a bottleneck of size 20. To train our INN, we give it the input x, obtaining $\text{INN}(x) = y$. Then, we replace all but the first 20 entries of y with zero, call this new output $\hat{y}$. Finally, we run the inverse model: $\text{INN}^{-1}(\hat{y}) = \hat{x}$. Then, we seek to minimize the reconstruction loss

$$\mathscr{L}(x, \hat{x}).$$

In so doing, we have effectively created a bottleneck. The zero-padded $\hat{y}$ has only 20 nonzero entries, and these entries can be thought of as our latent space encoding.

Nguyen et al. obtained very promising results. Testing with MNIST, CIFAR-10, and CelebA, they showed that, compared to traditional autoencoders:

1. INNs required less training epochs;
2. INNs could achieve better reconstruction losses across most bottleneck sizes;
3. INNs required less trainable parameters than their classic autoencoder counterparts.

The reason for 3 is primarily because the INN architecture is constant with respect to the bottleneck size, where classic autoencoders must change the number of trainable parameters depending on the bottleneck size. In their discussion of their results, they hypothesize the following:

> We already established, that if a DNN does not learn a bijective function, information loss occurs during the forward process making the inverse process ambiguous. The INN solves this ambiguity problem by introducing latent variables z containing all the information lost during the forward process.... Therefore, we hypothesize that INNs have no intrinsic information loss contrary to DNNs and the findings of Yu et al. do not apply to INNs. In other words, INNs are not bound to a maximal number of layers (depth) after which only suboptimal results can be achieved.

This is a bold hypothesis, and we will investigate it using information theory. Primarily, we seek to answer the following questions:

- How much information entropy is gained or lost at each layer of an INN?
- What about a classic autoencoder?
- What is the KL-divergence between the distribution of test inputs and reconstructed outputs?

3 Methods, Results

3.1 Methods

3.1.1 Experimental Setup

In order to answer these questions, we utilized the same models and architecture employed by Nguyen et al. We chose to compare the information loss across each layer in a classical model with its INN counterpart using two different datasets: MNIST and CIFAR-10. We chose these datasets so that our results could be comparable to the findings of Nguyen et al. Although we hoped to also used the CelebA data to more closely match the process in [4], computational and time constraints prevented us from examining this dataset as well.

We were primarily interested in three metrics for each model and dataset: Shannon entropy of the output at each layer, the relative entropy (measured by KL-Divergence) of the input and output at each layer, and the relative entropy between the model input and reconstructed model output. We measured these metrics on four models total: An Invertible Neural Net and a classical autoencoder on the CIFAR-10 dataset, and an Invertible Neural Net and a classical autoendcoder on the MNIST dataset.

We partitioned the datasets into a training and test set, trained the models, and then exmained our information metrics on the test set to see how much information the models would retain on unseen data. By examining these three metrics, we hoped to find a clear indicator of whether or not INNs limit information loss to a higher degree than classical autoencoders.

3.1.2 Model Architecture

In general, we used the extremely well-designed model code and architecture provided by Nguyen et al. and FrEIA. The only modification we made was adding the ability to track layer output at each layer in FrEIA's ReversibleGraphNet class. Apart from this, the model architecture was completely identical to that employed in [4].

The INN autoencoders for both the MNIST and CIFAR-10 dataset followed identical architectures: One Haar multiplex layer to reshape the data, three convolutional coupling layers with hidden size 128, one fully connected coupling layer with hidden size 1000, and a final Haar reshaping layer. The classical autoencoder for the CIFAR-10 dataset contained five convolutional layers with kernel size 3 and stride 1 and one fully-connected layer. The MNIST classical autoencoder contained four fully-connected layers (hidden size 512, 256, 128, 20) followed by ReLU functions. Further details can be found in [4].

The INN autoencoders were modified so that the model stored its output at each layer as an internal method. The classical autoencoders were accessed and called at each layer in order to generate the layer output.

3.2 Results

3.2.1 Layer-wise Entropy

Figure 3:

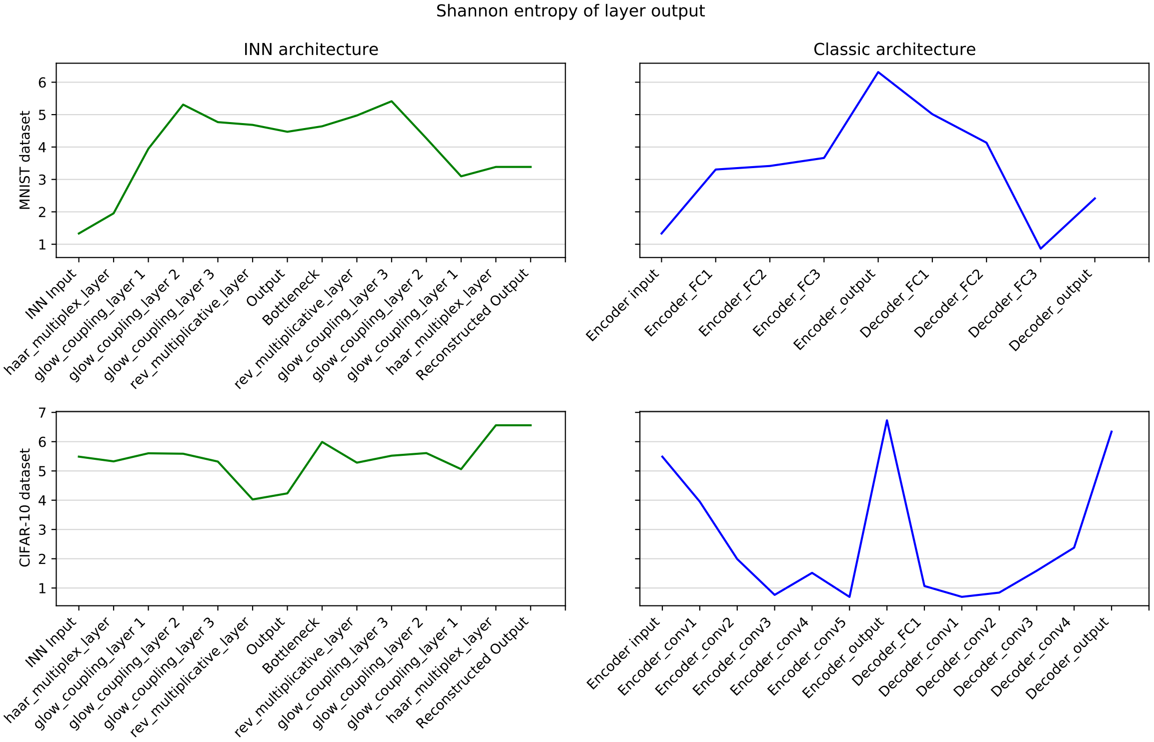
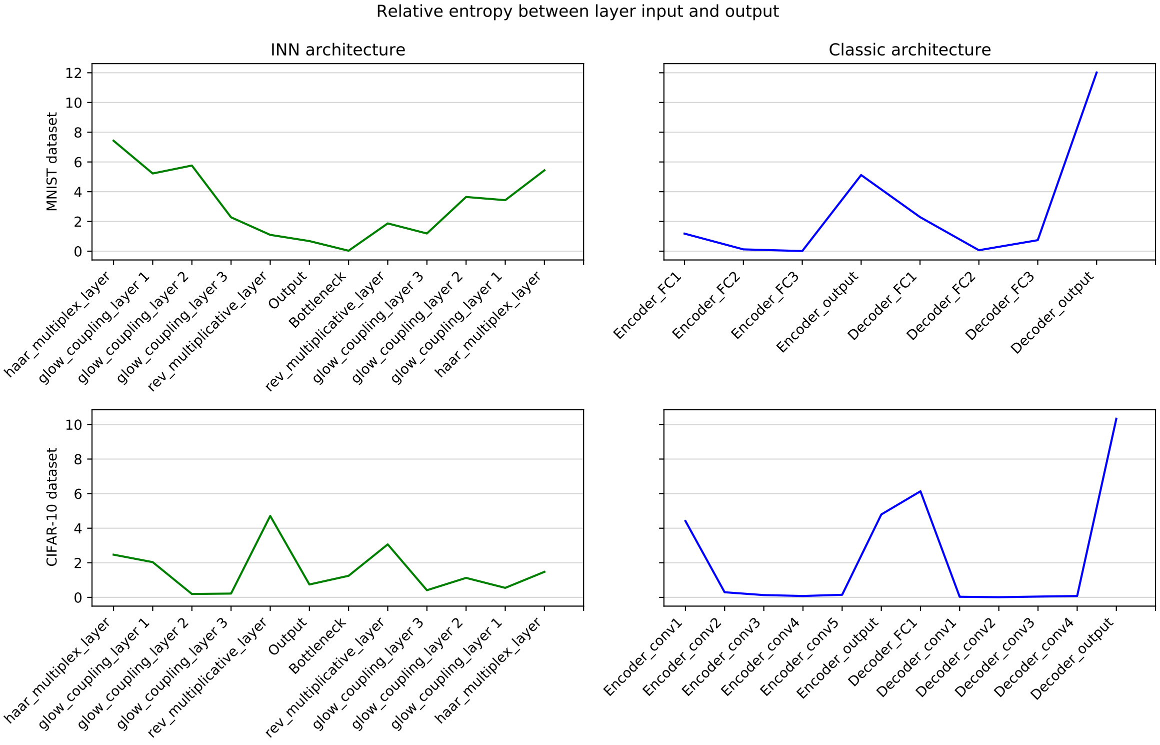


Figure 3 depicts the Shannon entropy of the output of each individual layer of the models. First, we examine the INN architectures. On MNIST, the entropy increases after the first two invertible coupling layers, then stays nearly constant, even after the bottleneck. Upon going through the model in reverse, the final two invertible coupling layers see the most decrease in entropy, and we find that our reconstructed output distribution has a slightly higher entropy than the original input distribution. On CIFAR-10, the entropy stays nearly constant, with the most noticeable spikes upon applying the bottleneck. This seems to indicate that on MNIST, the INN is more completely putting all useful information in the latent variable, whereas on CIFAR-10, there appears to be more information loss.

The classic architecture shows very noticeable spikes at the *Encoder_output* layer, which indicates that we expect a high amount of information from the values at these layers. This is exactly what we expect, since the encoded output is intended to store all of the important information while discarding random noise.

3.2.2 Layer-wise Relative Entropy

Figure 4:



The results of this metric are even more striking. We can see from Figure 4 that the relative entropies at each layer are much more constant in the INN autoencoders than they are in the classical autoencoders. Further, the classical autoencoders only see significant relative entropy values at the bottleneck layers and the output layers. This suggests that while both styles of autoencoder do a good job of preserving information from layer to layer, the classical autoencoder does it in an uneven way, providing almost no distributional change at non-bottleneck layers, and packing most of the distributional change into the bottleneck and outputs. In contrast, the INN autoencoder loses information at a more constant rate.

3.2.3 Relative entropy between inputs and outputs

The relative entropies between the inputs and outputs of each model are:

Dataset	INN	Classical
MNIST	10.46	10.25
CIFAR-10	0.76	0.28

It is clear to see that the relative information loss between input and output is very similar for each model. Surprisingly, the INN performs worse across both datasets. It is also interesting to note how drastically different the information loss is between the datasets. However, given that both the classical and INN architectures performed similarly on each dataset, it is probable that this discrepancy is due to properties of the datasets themselves and not the model architectures.

4 Conclusion

There are several interesting conclusions to draw from these results. First, it is clear that INN autoencoders do not preserve information as perfectly as hypothesized. In fact, they reconstruct images with information preservation similar to or worse than classical autoencoders. Interestingly, Nguyen et al. show that the reconstruction loss for INN autencoders is better than that of classical autoencoders. This suggests that the INNs could be overfitting more to the data while still losing a comparable amount of information.

However, INN autoencoders seem to lose information at a more constant rate than do their classical counterparts, whose layer-wise entropy changes drastically depending on the layer. This means that on average the INN autoencoder loses less information across the intermediate layers. Given that the idea of an INN is to learn bijections, this behavior should be expected.

In conclusion, it is clear that INN autoencoders handle information in a markedly different way than classical autoencoders, but it is not clear that they preserve any more information than their counterparts.

References

[1] Carl Doersch. Tutorial on variational autoencoders. 2021. URL http://arxiv.org/abs/1606.05908.

[2] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. 2019. URL http://arxiv.org/abs/1808.06670.

[3] Lukas Hollenstein, Lukas Lichtensteiger, Thilo Stadelmann, Mohammadreza Amirian, Lukas Budde, Jürg Meierhofer, Rudolf M. Füchslin, and Thomas Friedli. Unsupervised learning and simulation for complexity management in business operations. In *Applied Data Science*, pages 313–331. Springer International Publishing, 2019. doi: 10.1007/978-3-030-11821-1_17. URL https://doi.org/10.1007/978-3-030-11821-1_17.

[4] The-Gia Leo Nguyen, Lynton Ardizzone, and Ullrich Köthe. Training invertible neural networks as autoencoders. In *Lecture Notes in Computer Science*, pages 442–455. Springer International Publishing, 2019. doi: 10.1007/978-3-030-33676-9_31. URL https://doi.org/10.1007/978-3-030-33676-9_31.

[5] Shujian Yu and Jose C. Principe. Understanding autoencoders with information theoretic concepts. 2019. URL http://arxiv.org/abs/1804.00057.