Intrinsic Analysis of Learned Representations in Encoder-Decoder Architectures

Shashi Durbha[1,2], Thomas Goerttler[1], Eduardo Vellasques[2], Jan Hendrik Stockemer[2] and Klaus Obermayer[1,3]

[1]*Technische Universität Berlin, Chair of Neural Information Processing, Germany*
[2]*SAP SE, Germany*
[3]*Bernstein Center for Computational Neuroscience Berlin, Germany*

Abstract
Encoder-decoder architectures have widespread use, both in research and the industry. Recently, similarity analysis applied to the representations of neural networks has contributed to a better understanding of the architectures. Previous work has found that for two instances (under different initialization) of the same layer of a given model, the learned representation becomes more and more dissimilar the farther away that layer is from the input layer. Since the encoder-decoder is often mirrored (causing a representational bottleneck), we investigate how the representation changes and if the objective of reconstructing the input influences this. Using representation similarity analysis, we find out that corresponding layers from the encoder and decoder are not very similar to each other and are more similar to their neighbor layers. In addition, our experiments show that except for classification tasks, the representations of the same decoder with different initialization become more and more similar the closer a layer is to the output layer. Some of our analysis includes comparing the average distances between the layers to the average distance of the current layer clusters, the impact of varying the latent dimension as well as having multiple bottleneck layers on the representational consistency.

Keywords
Neural Networks, Encoder-Decoder Architectures, Representational Similarity Analysis (RSA), Representational Consistency,

1. Introduction

One of the first successful cases of training deep architectures involved successively training (and consequently stacking) denoising autoencoders [1]. However, until very recently, much has been hypothesized about how learning occurs as information propagates through the different layers. Notwithstanding, the research community has shed some light on that subject in the last few years. For example, Ansuini et al. [2] applied intrinsic dimension analysis to examine the geometrical properties of the representations learned by a deep neural network.

Popal et al. [3] use Representational Similarity Analysis (RSA) to analyze representation learning in CNNs. In RSA, a Representational Dissimilarity Matrix (RDM) is employed in

LWDA'22: Lernen, Wissen, Daten, Analysen. October 05–07, 2022, Hildesheim, Germany
thomas.goerttler@tu-berlin.de (T. Goerttler); eduardo.vellasques@sap.com (E. Vellasques); jan.hendrik.stockemer@sap.com (J. H. Stockemer); klaus.obermayer@tu-berlin.de (K. Obermayer)
https://www.thomas.goerttler.de/ (T. Goerttler); https://www.ni.tu-berlin.de/ (K. Obermayer)
0000-0002-1437-0235 (T. Goerttler); 0000-0002-5057-6142 (K. Obermayer)

order to characterize a system's inner stimulus representations in terms of pairwise response differences [4]. In a technique called Representational Consistency (RC), Mehrer et al. [5], apply techniques until then only used in the field of neuroscience in order to compare different instances of the same neural network under random initialization. More specifically, they employ Pearson correlation analysis in order to quantify the variance between RDMs of the different instances. One important takeaway from their research is that by relying on multiple instances of the same architecture, it is possible to derive insights about representation learning in a statistically quantifiable way.

In this paper, we apply RSA to the representations of (convolutional) autoencoders and observe their consistency. We observe that in contrast to the supervised classifiers examined in Mehrer et al. [5], where consistency decreases with each layer stacked, for the autoencoder, the consistency reaches a minimum in the bottleneck layer and increases again in the decoder part.

We also investigate the influence of the size of the latent layer. The larger it is, the more consistent it stays. Analyzing a single model, we see that the layers of a network are more similar to the neighbor layers than to their corresponding mirror layers.

2. Background

In this section, we describe the encoder-decoder architectures and introduce representational similarity techniques.

2.1. Encoder-Decoder Architectures

Encoder-decoder architectures are networks in which an encoder takes a variable-length sequence as the input and transforms it into a state with a fixed shape. The decoder then maps the encoded state of a fixed shape to a variable-length sequence. The final encoder layer from which the decoder maps the encoded state is known as the latent dimension layer, which has the least number of neurons among all the layers of the network.

Encoder-Decoder architectures have widespread use in Natural Language Processing (NLP) and Image Denoising. In NLP, the common approach is to use sequential models such as Recurrent Neural Networks (RNNs) as encoder/decoder.

2.1.1. Autoencoders

Baldi [6] describes Autoencoders as *'simple learning circuits which aim to transform inputs into outputs with the least possible amount of distortion'*. Goodfellow et al. [7] describe autoencoders as a special type of neural network which comes under the class of *unsupervised* networks. Such type of model is trained using a reconstruction loss. It is a common practice to add noise to the input. This leads to a learned representation that is robust to noise. Overall, the network may be viewed as consisting of two parts: (i) an encoder ($h = f(x)$), and (ii) a decoder ($r = g(h)$). The representational bottleneck forces the model to discard redundant information, so it can learn the useful properties of the data.

2.2. Representational Similarity Analysis (RSA) and Representational Consistency (RC)

Representational Similarity Analysis (RSA) was first proposed in the field of neuroscience [4], as a way to compare computational models to brain-activity data. Recently, it has also become a tool to analyze representations of deep convolutional networks. Mehrer et al. [5] firstly proposed using RSA to analyze the behavior of Convolutional Neural Networks (CNN) under different initializations. More recently, Goertler and Obermayer [8] employed the same technique to analyze model-agnostic meta-learning. The main building block of RSA is the Representation Dissimilarity Matrix (RDM), which is a symmetric matrix containing pairwise measurements for elements of two representation vectors (given a set of test stimuli). More formally, for any given pair of representation vectors h_i and h_j, the RDM is defined as:

$$RDM_{i,j} = \delta(h_i, h_j) \tag{1}$$

where $\delta(h_i, h_j)$ is a dissimilarity function. Kriegeskorte et al [4] proposes using 1 - correlation as dissimilarity function. Next to RSA, there also exist similar approaches, e.g. CKA [9] or CCA [10] which have also been used to analyze the similarity of deep networks [11, 12].

Given an RDM, it is possible to analyze how well the distribution of representational distances generalizes across network instances. This can be done using Representational Consistency (RC), which is defined as the shared variance between the upper triangle of a given RDM matrix. More specifically, given a set of instances $X \in \{x_1, ..., x_n\}$, and a layer $g_i()$, the RC for layer i (RC_i) is defined as:

$$Z = \frac{\|X\|(\|X\| - 1)}{2} \tag{2}$$

$$RC_i = \frac{\sum_{j<k} PCC(g_i(x_j), g_i(x_k))}{Z} \tag{3}$$

where $\|X\|$ is the number of instances in X, and PCC is the Pearson correlation coefficient. The RC for a given model is defined as the average RC of all the layers in that model.

The intuition behind RC is that the closer the value is to 1, the higher the consistency for that particular layer. For example, the input layer is always 1 because the inputs are the same for all the different instances of the network. As we progress deeper into the network layers, we see a continuous drop in consistency until we reach the bottleneck layer.

Multidimensional scaling (MDS) [13] gives a concise 2D projection of representation vectors. Since an RDM contains distances between instances of representation vectors, it can be used as a way to project these instances in a 2D space, such that similar instances are projected closer than dissimilar instances.

3. Experimental setup

The goal of our experiments is to get a better understanding of the representational behavior in autoencoder architectures. To that end we analyze representation similarity and consistency under slight variations of the experimental setup described below:

Index	Type	Output Shape
1	Input Layer	784
2	Dense$_1$	512
3	Dense$_2$	256
4	Dense$_3$	128
5	Dense$_4$	64
6	Bottleneck$_1$	8
7	Bottleneck$_2$	8
8	Bottleneck$_3$	8
9	Dense$_8$	64
10	Dense$_9$	128
11	Dense$_{10}$	256
12	Dense$_{11}$	512
13	Output Layer	784

(a) autoencoder for MNIST

Index	Type	Output Shape
1	Input Layer	(32, 32, 3)
2	Convolution$_1$	(32, 32, 32)
3	MaxPooling$_1$	(16, 16, 32)
4	Convolution$_2$	(16, 16, 64)
5	MaxPooling$_2$	(8, 8, 64)
6	Convolution$_3$	(8, 8, 128)
7	Flatten	8192
8	Bottleneck$_1$	12
9	Bottleneck$_2$	12
10	Bottleneck$_3$	12
11	Dense	8192
12	Reshape	(8, 8, 128)
13	Convolution$_4$	(8, 8, 64)
14	UpSampling$_1$	(16, 16, 64)
15	Convolution$_5$	(16, 16, 32)
16	UpSampling$_2$	(32, 32, 32)
17	Output Layer	(32, 32, 3)

(b) convolutional autoencoder for CIFAR10

Table 1
Basic architectures for the experiments. In some experiments we included additional layers as indicated.

3.1. Training

Except for the experiment where we were attempting to analyse the change in consistency with the epochs increasing logarithmically till 256, all the other models were trained using *early stopping*, monitoring the *validation loss* with a *patience* of 10.

Additionally, for the experiments conducted on representational consistency - for a Keras-based architecture for an autoencoder, the default kernel initialization is generally a *Glorot-Uniform* for a dense/convolutional layer. So to change this default initialization, the experiments in the thesis use a *Glorot-Normal* instead to see the difference obtained.

3.2. Architectures

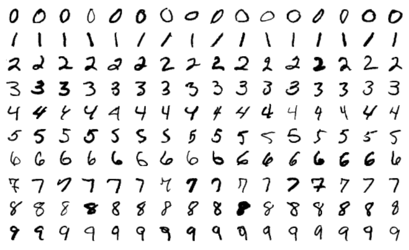
In our experiments, we use two different architectures, one with only fully connected layers and one with also convolutional ones. In Table 1a, we see the base architecture of the fully connected networks, whereas, in Table 1b, the convolutional architecture is depicted. In several experiments, we extend the networks as indicated.

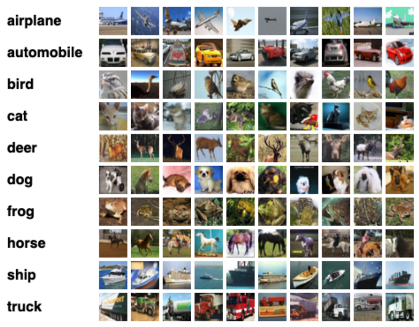
3.3. Datasets

Below is a short description of the datasets used in the experiments conducted in the paper.

3.3.1. MNIST (Modified National Institute of Standards and Technology)

The MNIST database [14] consists of 70,000 hand-written grayscale digits of size (28×28), which are divided into training and testing, with a ratio of 6:1 along with their own labels. It contains ten (digits from 0-9) different classes with an equal number of images in each class. It is a very common and widely used database in the field of ML and is also very easily accessible through the Keras dataset library. Because of these reasons, it is one of the default benchmark datasets in the field of ML. When working with a regular autoencoder, the pixel values in these datasets were binarized using a threshold of 255 and flattened out to a vector of size 784. The main motivation behind using this dataset was to compare the results between both the architectures, regular and convolutional autoencoders. Image 1a shows a sample image of how the dataset looks.


(a) Sample image from the MNIST dataset [14]



(b) Sample CIFAR10 images [15]

Figure 1: MNIST and CIFAR10 datasets sample images.

3.4. CIFAR10 (Canadian Institute For Advanced Research)

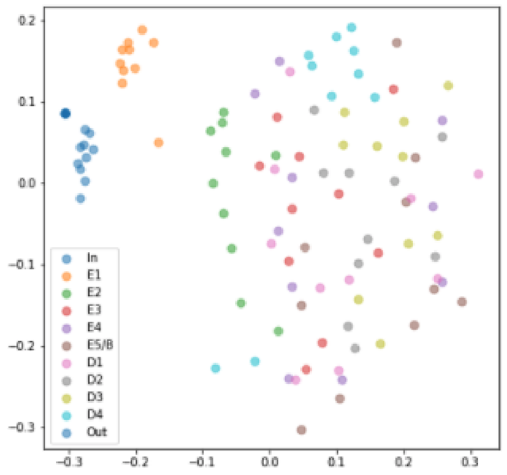
Similar to the MNIST dataset, CIFAR10 [16] is a database of 60,000 images having ten different classes with 6000 images in each class. It is divided into training and testing, with a ratio of 5:1 along with its own labels. Unlike the MNIST dataset, the image size for the CIFAR10 dataset is (32×32) with three RGB channels. The image in figure 1b shows a sample image of how this dataset looks.

4. Results

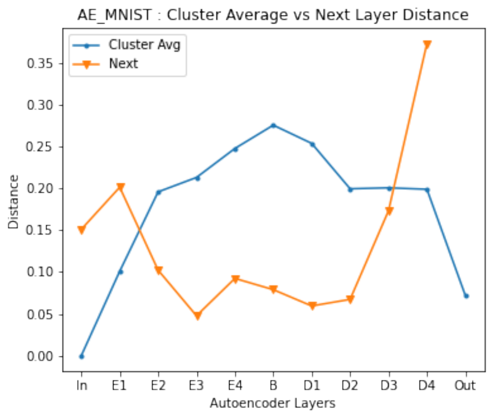
In this section, we depict our results.

4.1. Representational similarity in autoencoders

In our first experiment (Figure 2a), we show all layers of ten differently initialized instances embedded into two dimensions according to their respective similarities. The input and the output layers are comparatively closer to each other, and all the other layers are quite far apart. We can conclude that the points of the same layer get further apart from each other as we move deeper into the layers of the network, and they start getting closer to each other again as we reach the end. The input representations are obviously all the same. The next layer (Encoder1 - E1 in the plot in Figure 2a) has all the points very close to each other, except maybe one point,



(a) Sample MNIST dataset MDS plots for 10 models



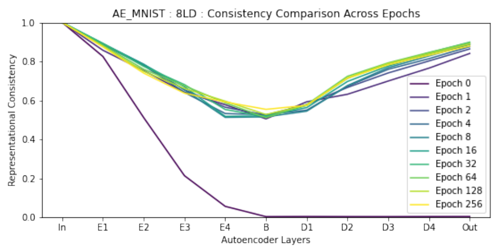
(b) Sample comparison between *cluster average* versus *next layer* distance.

Figure 2: MDS plot form MNIST, and comparision.

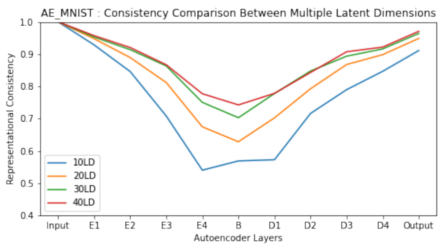
which is comparatively far away from all the other points with the same color. This becomes more frequent as we keep going deeper into these layers.

In Figure 2b, we observe that except for the first and last layers, the distance to the successive (next) layer is smaller the the overall cluster average.

4.2. Representational consistency in autoencoders



(a) Representational consistency: logarithmically increasing epochs comparison for a Fully-connected autoencoder with 8 neurons in the bottleneck layers.



(b) Comparison between multiple latent dimensions for an autoencoder architecture on an MNIST dataset

Figure 3: Representational consistency regarding epochs and different size of the latent space

Similar to Mehrer et al. [5] we also investigate the representation consistency described in Section 2.2. In Figure 3a, we see that the similarity of the representation increases after the bottleneck layer. This is different from the supervised case of Mehrer et al. [5] where the similarity diverges coming closer to the output layer. This is very interesting as this means that an autoencoder can reconstruct the representation despite having different encodings. In addition, the plot in Figure 3a shows the comparison inconsistencies when working with a varying number of epochs (from 0 to 256, with the epochs increasing logarithmically). Epoch 0, shown in the plot, is when the consistency is calculated without any prior training. While in the other trained cases for this network, we see that as we keep increasing the number of epochs, the consistencies start getting higher and gradually appear to be converging. A couple of key observations when conducting these experiments was that untrained networks behave as expected, i.e., the consistency decreases as we progress through the layers. We also observe that the convergence with regards to consistency is really fast, where we see the typical autoencoder behavior after a single epoch itself.

4.2.1. Impact of varying Latent dimension on representational consistency

In this experiment, we observe the representation consistency when changing the dimension of the latent vector and letting everything be the same.

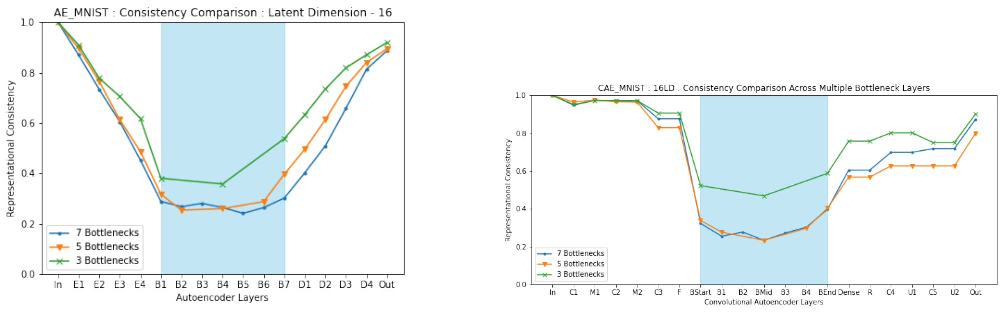
Based on the results of the experiments shown in Figure 3b, we can see a clear correlation between the representational consistency and the number of neurons we have in the latent dimension layers. We notice that as the number of neurons increases, the representational consistency of the bottleneck layer also increases.

We can interpret that with the increase in the number of neurons of the bottleneck layer from 10 to 40 though these values are comparatively less, the information held in the previous layers has to be less compressed in the case of the 40 neurons than in the case of the ten neurons. We would always attribute some amount of loss in the case of information transfer from the previous encoder layer to the bottleneck layer, but it would be comparatively less in the case of the 40 neurons than the ten neurons, which automatically correlates to a more consistent reproducibility of the input image in the output and hence a better consistency comparison in the final output layer as well.

4.2.2. Representational consistency on multiple bottleneck layers

When training architectures with multiple bottleneck layers (3, 5 and 7), the below experiments were conducted with ten differently initialized instances for all the experiments: (i) Fully-connected autoencoder: MNIST dataset with 8 and 16 neurons in the bottleneck layers respectively, (ii) Convolutional autoencoder: MNIST dataset with 8 and 16 neurons in the bottleneck layers respectively, (iii) Convolutional autoencoder: CIFAR10 dataset with 12 and 25 neurons in the bottleneck layers respectively. The results of these experiments can be seen in the plots in Figure 4:

The above plots show the comparisons while working with a varying number of bottleneck layers for an MNIST dataset for a Fully-connected autoencoder as well as a convolutional autoencoder, with 16 neurons in the bottleneck layer. A common latent dimension and dataset is chosen so that the networks are more comparable. We see, in both cases, that there is not much separating the consistencies of the networks with five and seven bottleneck layers. However, the network with three bottleneck layers has a much higher consistency at the bottleneck level



(a) Fully-connected autoencoder with 16 neurons in the bottleneck layers.

(b) Convolutional autoencoder with 16 neurons in the bottleneck layers.

Figure 4: Representational consistencies comparisons for an MNIST dataset working with different networks for a varying number of bottleneck layers

as well as overall in both cases. In the case of the networks with five and seven bottleneck layers, we see the consistencies flattening out in the region shaded blue in the above plots. And though this cannot be confirmed with a guarantee as yet, this trend has more or less been consistent regardless of the dataset and the network. A possible, interesting future work for this thesis would be to replicate these results on much larger and more complicated datasets.

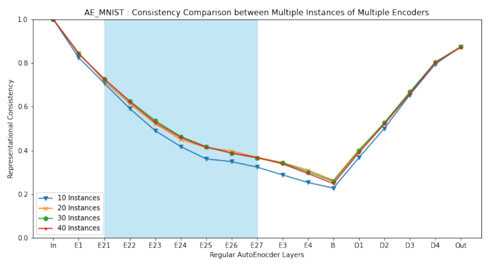
4.2.3. Dependency of consistency on the number of trained model instances

To check the dependence of the consistency on the number of trained model instances with different random initial weights, two experiments were performed. For the fully-connected autoencoder trained on MNIST as well as the convolutional autoencoder trained on CIFAR-10 we compared the consistency values in the different layers for different number of model instances. The results are shown in Figure 5a for the fully connected autoencoder and Figure 5b for the convolutional autoencoder. The fully connected autoencoder, tested for 10, 20, 30 and 40 model instances, shows convergence at 20 instances. Due to technical limitations the convolutional autoencoder was only tested for 5, 10, 15, and 20 model instances. While the consistency is not convergent at these instance numbers, we can see the same pattern as for the fully connected autoencoder and assume that using more than 20 model instances would not result in relevant changes.

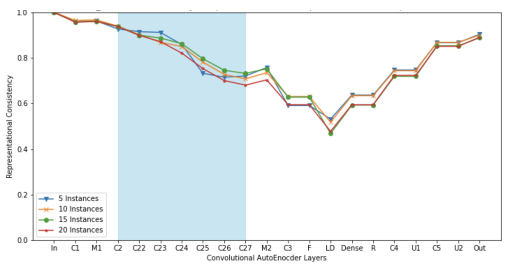
4.2.4. Effect of increased layer numbers on the representation consistency

Figure 6 shows all the representational consistencies in a single plot for the MNIST dataset for a convolutional autoencoder. We can see here that, regardless of which layer has how many multiple layers in other networks, the overall highest consistency was shown by the network, which has no multiple numbers for any of the layers.

The basic motivation behind this above experiment was to see how multiple adding layers of a certain type of layer, before and after the bottleneck layer, impacted the representational consistencies across multiple instances. A small note here is that when working with a convo-



(a) Fully connected Autoencoder on MNIST



(b) Convolutional Autoencoder on CIFAR-10

Figure 5: Calculating representation consistency with varying number of randomly initialize model instances

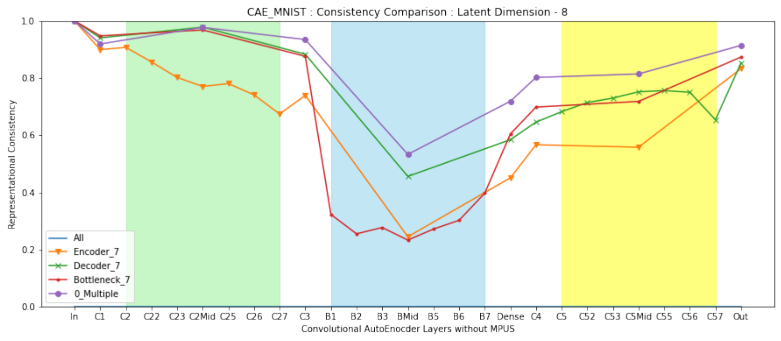


Figure 6: Representational consistency for all the multiple layers together. The orange line indicates multiple encoders. The green line indicates multiple decoders. The red line indicates multiple bottleneck layers, and the final purple line indicates a consistent plot for a network without any multiple layers. The green shade in the plot indicates all the multiple encoder layers. The blue shade indicates all the multiple bottleneck layers, and the yellow shade indicates all the multiple decoder layers.

lutional autoencoder, the bottleneck layer was a dense layer, but the other multiple layers (C2 and C5) were all convolutional layers.

4.3. Comparison of encoder and decoder

So far, we have only looked at the representation consistency regarding ten different models with different random initialization. In this section, we want to look at individual models to understand single models better and observe if mirrored layers share a structure. In Figure 7 are the MDS plots for two samples that were obtained using a CIFAR10 dataset on a convolutional autoencoder with 12 neurons in its bottleneck layer.

A slight distance is seen between the input and output. However, it is important to understand here that the motive behind performing these experiments was not to find the best possible solution as to which parameters or architectures give the most accurate results. Hence, it was chosen to work with neurons (8 and 16 neurons in the case of an MNIST dataset and 12 and 25

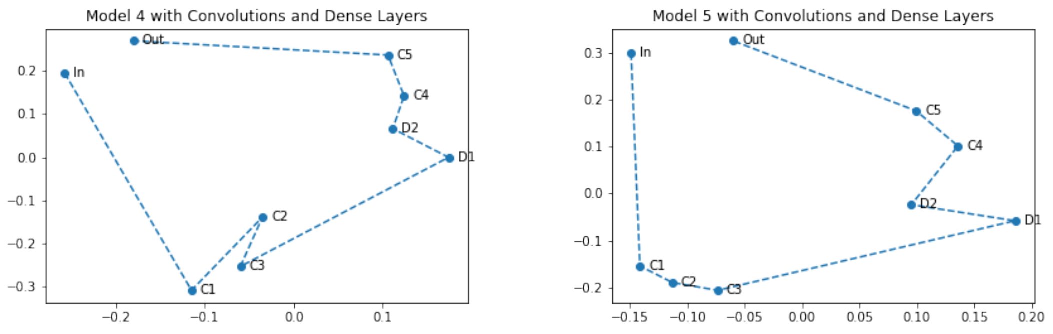
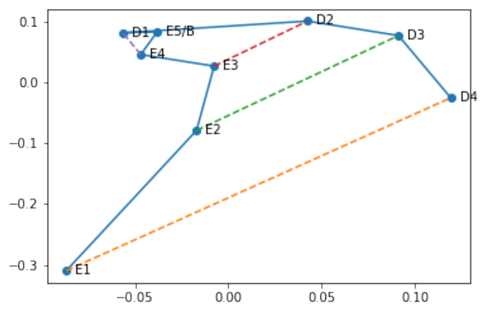


Figure 7: Sample instances MDS plots. Out of the numerous instances used in the experiments, these sample plots have been taken from the convolutional autoencoder experiments which were worked on a CIFAR10 dataset. As seen in the plots, the maxpooling and upsampling layers have been removed from the plot as they do not add anything significant to the direction of the network in general in the plot. Instead, all the convolutional (*C*) layers and dense layers (*D*) were kept along with the input and output layers. The *D1* we see in the plot is the only bottleneck layer used in this experiment. The second dense layer *D2* is used to rebuild the neurons for the convolutional layers.

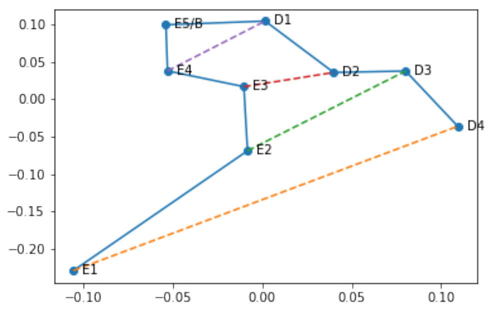
neurons in the case of a CIFAR10 dataset) that were not giving us pixelated outputs while at the same time, they were not overcompensating what is being attempted to study when conducting these experiments. Although the individual instance plots look similar in nature, we see some differences in the directions of the layers, which would be attributed to the different random initializations that take place inside these networks.

4.3.1. Analysing the impact on representational consistency when varying the latent dimension

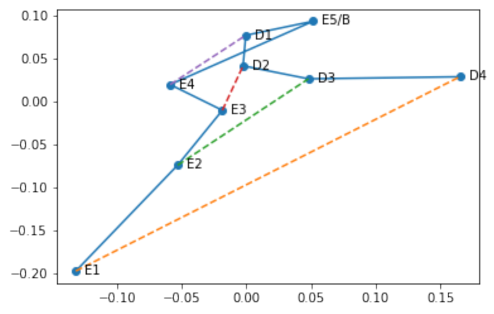
It is important to note that when the input and output layers are included in an instance plot, these two layers would always be close or far away from each other depending on the number of neurons chosen to work with for the latent dimension. This, however, can be misleading at times because of what the autoencoder attempts to do. For this very reason, the above plots of Figure 8 were included where the input and output layers were left out, and a simpler comparison was made between all the other layers of the network. As shown in the above Figure 7, we do not see much difference between different instances of the same latent dimension. For the current analysis in Figure 8, we see four different instance plots (all four of them are taken from the 3^{rd} instance of each experiment for a consistent comparison), where the instances have been plotted without the input and output layers with linearly increasing latent dimensions (i.e., 10, 20, 30 and 40 neurons). Even in such a scenario, we see consistency among the models. As previously discussed, we would normally expect a U-shape between the mirror layers of the model, but we see a sort of a *V*-shape forming between the mirror layers with the shortest distance between the final encoder layer and the first decoder layer as the distance keeps increasing all the way till the first encoder layer and the final decoder layer. This distance appears to be consistent across all the instances, even with an increasing latent dimension.



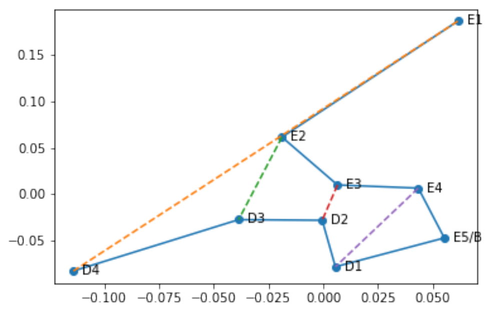
(a) Latent dimension with 10 neurons



(b) Latent dimension with 20 neurons



(c) Latent dimension with 30 neurons



(d) Latent dimension with 40 neurons

Figure 8: Comparing instance plots with varying Latent Dimension

5. Conclusion

In this paper, we demonstrated that for an encoder-decoder neural network, as information is propagated through encoder layers, there is a decrease in the representational consistency of the layer. And the opposite happens for the decoder layers. Such type of representational bottleneck forces the encoder to "prioritize" features, discarding features not relevant for reconstruction. We also demonstrated that for any given instance, the representations tend to diverge (from the input) as information is propagated through the encoder layers and then converge (towards the input) as information is propagated through the decoder.

It is also observed that even when we have multiple bottlenecks, a consistent line is not seen for that particular period. There was a dip even among the bottleneck layers. Furthermore, the dip was observed in the middle bottleneck layer in these instances with multiple bottleneck layers.

In practical terms, these findings validate the common intuition behind encoder-decoder neural networks (that such bottleneck provides some sort of "lossy compression"). For practitioners, such type of analysis can be used a helpful tool when debugging novel architectures.

References

[1] P. Vincent, H. Larochelle, Y. Bengio, P.-A. Manzagol, Extracting and composing robust features with denoising autoencoders, in: Proceedings of the 25th international conference on Machine learning, 2008, pp. 1096–1103.

[2] A. Ansuini, A. Laio, J. H. Macke, D. Zoccolan, Intrinsic dimension of data representations in deep neural networks, Advances in Neural Information Processing Systems 32 (2019).

[3] H. Popal, Y. Wang, I. R. Olson, A guide to representational similarity analysis for social neuroscience, Social Cognitive and Affective Neuroscience 14 (2019) 1243–1253.

[4] N. Kriegeskorte, M. Mur, P. A. Bandettini, Representational similarity analysis-connecting the branches of systems neuroscience, Frontiers in Systems Neuroscience 2 (2008) 4.

[5] J. Mehrer, C. J. Spoerer, N. Kriegeskorte, T. C. Kietzmann, Individual differences among deep neural network models, Nature communications 11 (2020) 1–12.

[6] P. Baldi, Autoencoders, unsupervised learning, and deep architectures, in: Proceedings of ICML workshop on unsupervised and transfer learning, JMLR Workshop and Conference Proceedings, 2012, pp. 37–49.

[7] I. Goodfellow, Y. Bengio, A. Courville, Y. Bengio, Deep learning, volume 1, MIT press Cambridge, 2016.

[8] T. Goerttler, K. Obermayer, Exploring the similarity of representations in model-agnostic meta-learning, arXiv preprint arXiv:2105.05757 (2021).

[9] S. Kornblith, M. Norouzi, H. Lee, G. E. Hinton, Similarity of neural network representations revisited, in: K. Chaudhuri, R. Salakhutdinov (Eds.), Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA, volume 97 of *Proceedings of Machine Learning Research*, PMLR, 2019, pp. 3519–3529.

[10] A. Morcos, M. Raghu, S. Bengio, Insights on representational similarity in neural networks with canonical correlation, Advances in Neural Information Processing Systems 31 (2018).

[11] S. Kornblith, T. Chen, H. Lee, M. Norouzi, Why do better loss functions lead to less transferable features?, in: M. Ranzato, A. Beygelzimer, Y. N. Dauphin, P. Liang, J. W. Vaughan (Eds.), Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual, 2021, pp. 28648–28662. URL: https://proceedings.neurips.cc/paper/2021/hash/f0bf4a2da952528910047c31b6c2e951-Abstract.html.

[12] T. Goerttler, K. Obermayer, Similarity of pre-trained and fine-tuned representations, arXiv preprint arXiv:2207.09225 (2022).

[13] J. O. Ramsay, Some statistical considerations in multidimensional scaling, Psychometrika 34 (1969) 167–182.

[14] Y. LeCun, C. Cortes, MNIST handwritten digit database (2010). URL: http://yann.lecun.com/exdb/mnist/.

[15] H. Xiao, K. Rasul, R. Vollgraf, Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, CoRR abs/1708.07747 (2017). URL: http://arxiv.org/abs/1708.07747. `arXiv:1708.07747`.

[16] A. Krizhevsky, Learning multiple layers of features from tiny images, Technical Report, 2009.