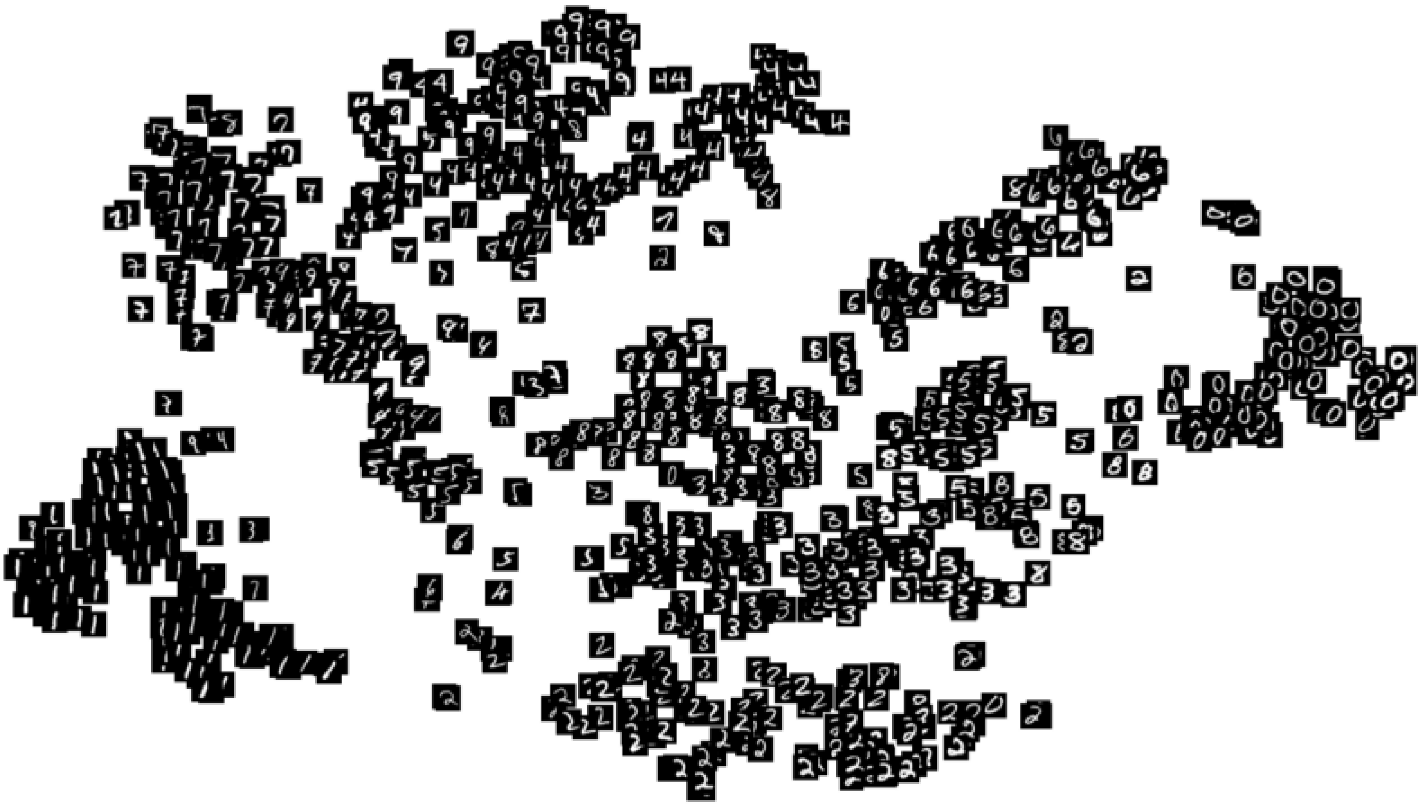
Understanding Latent Space in Machine Learning

Source: https://scribe.rip/understanding-latent-space-in-machine-learning-de5a7c687d8d

Learn a fundamental, yet often 'hidden,' concept of deep learning



✍Source: Hackernoon, https://hackernoon.com/latent-space-visualization-deep-learning-bits-2-bd09a46920df

What is Latent Space?

> If I have to describe latent space in one sentence, it simply means a representation of compressed data.

Imagine a large dataset of handwritten digits (0–9) like the one shown above. Handwritten images of the same number (i.e. images that are 3's) are the most similar to each other compared to other images of different numbers (i.e. 3s vs. 7s). But can we train an algorithm to recognize these similarities? *How?*

If you have trained a model to *classify digits*, then you have *also* trained the model to learn the 'structural similarities' between images. In fact, this is how the model is able to classify digits in the first place- by learning the features of each digit.

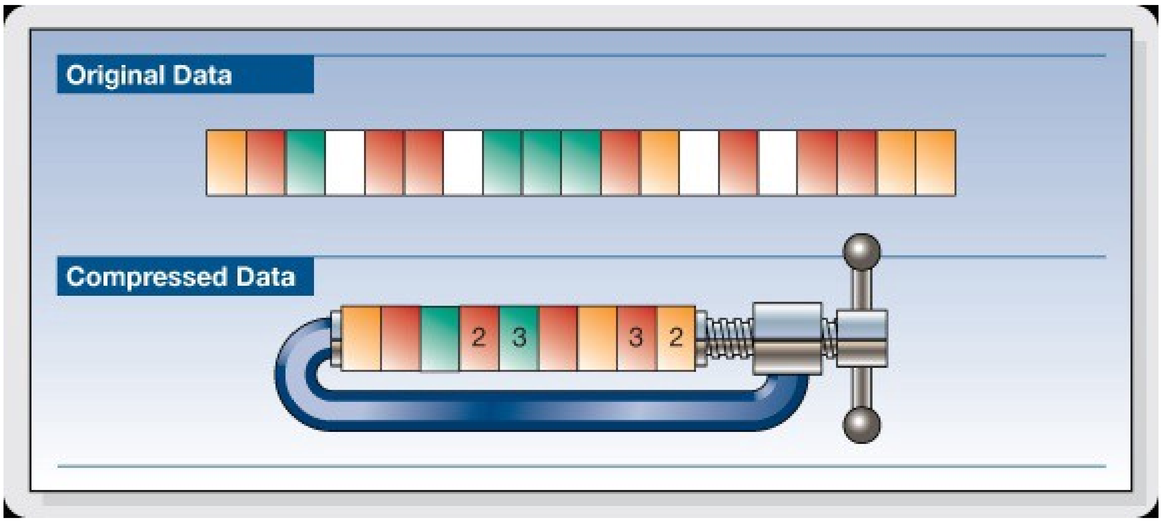
If it seems that this process is 'hidden' from you, it's because it is. Latent, by definition, means "hidden."

The concept of "latent space" is *important* because it's utility is at the core of 'deep learning' — *learning the features of data and simplifying data representations for the purpose of finding patterns.*

Intrigued? Let's break latent space down bit by bit:

Why do we compress data in ML?

Data compression is defined as the process of encoding information using fewer bits than the original representation. This is like taking a 19D data point (need 19 values to define unique point) and squishing all that information into a 9D data point.

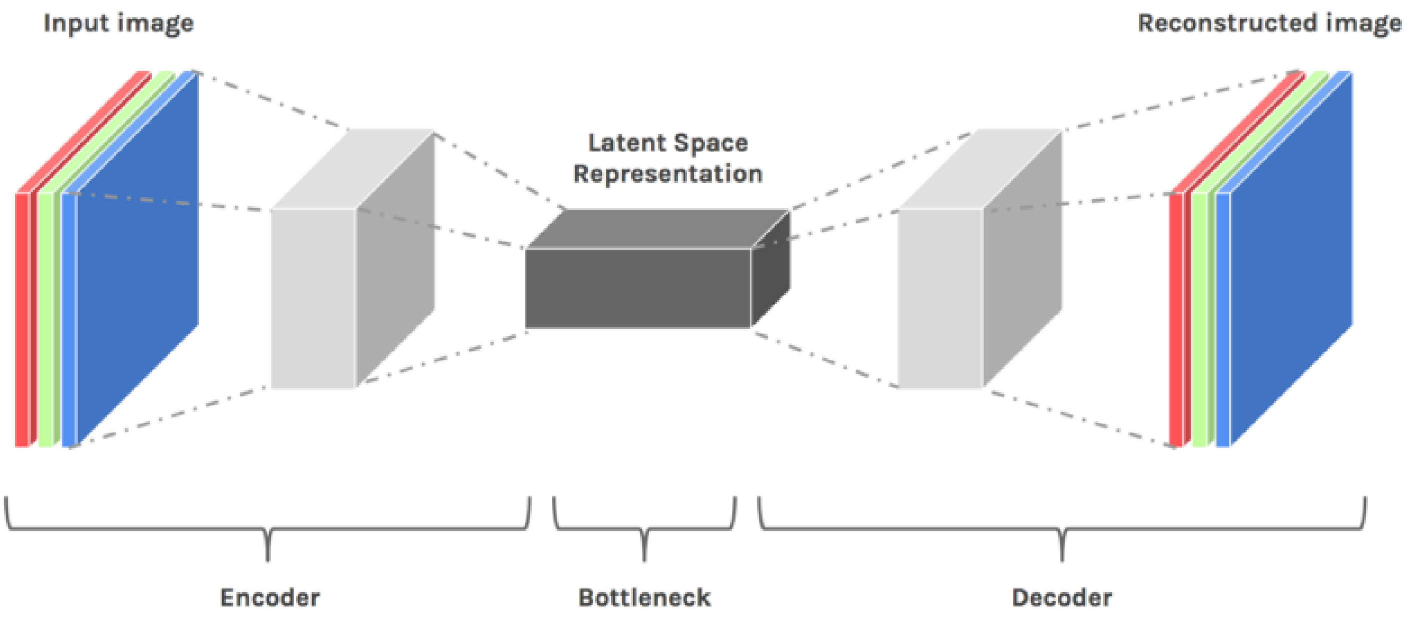


✍Illustration of compression. Source: Faust 2013

More often than not, data is compressed in machine learning to *learn important information about data points*. Let me explain with an example.

Say we would like to train a model to classify an image using a fully convolutional neural network (FCN). (i.e. output digit number given image of digit). As the model 'learns', it is simply learning *features* at each layer (edges, angles, etc.) and attributing a combination of features to a specific output.

But each time the model learns through a data point, the *dimensionality* of the image is first *reduced* before it is ultimately increased. (see Encoder and Bottleneck below). When the dimensionality is reduced, we consider this a form of lossy compression.



✍Depiction of convolutional neural network. Source: Source: Hackernoon Latent Space Visualization.

Because the model is required to then *reconstruct* the compressed data (see Decoder), it must learn to store *all relevant information* and disregard the noise. This is the value of compression- it allows us to get rid of any extraneous information, and only focus on the most important features.

This ‘compressed state’ is the Latent Space Representation of our data.

What do I mean by space?

You may be wondering why we call it a latent *space*. After all, compressed data, at first glance, may not evoke any sort of “space.”

But here’s the parallel.

In this rather simplistic example, let’s say our original dataset are images with dimensions 5 x 5 x 1. We will set our latent space dimensions to be 3 x 1, meaning our compressed data point is a vector with 3-dimensions.

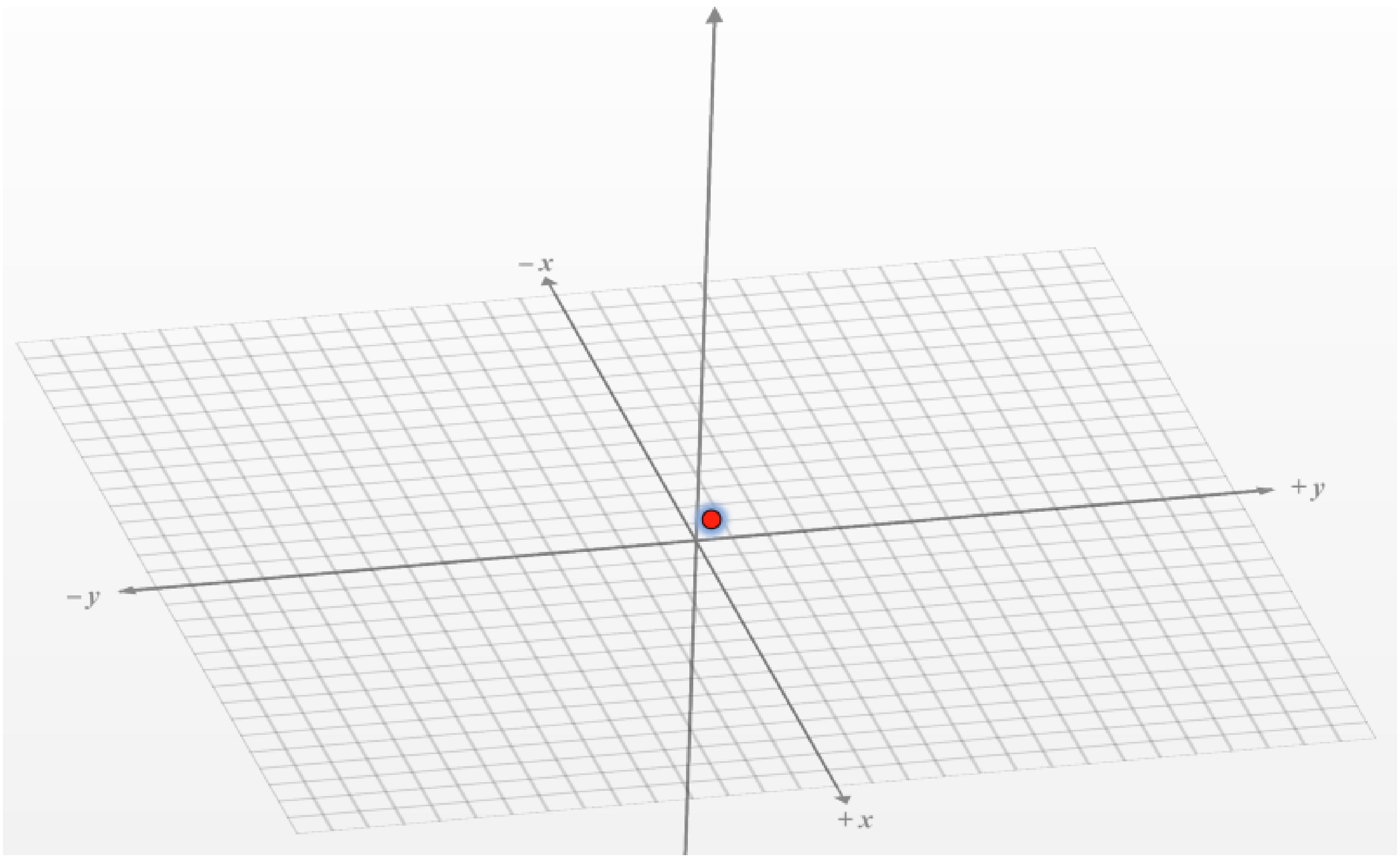
$$\begin{bmatrix} 1 & 0.5 & 0.8 & 0.2 & 0.7 \\ 0.2 & 0.11 & 0.78 & 0.3 & 0.2 \\ 1 & 0.01 & 0 & 0.9 & 0.56 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 0.4 & 0.2 & 0.1 & 0.01 \end{bmatrix}$$

✍Example 5x5x1 data

$$\begin{bmatrix} 0.4 \\ 0.3 \\ 0.8 \end{bmatrix}$$

✍Example compressed 3x1 data in ‘latent space’

Now, each compressed data point is uniquely defined by only 3 numbers. That means we can graph this data on a 3D Plane (One number is x, the other y, the other z).



✍Point (0.4, 0.3, 0.8) graphed in 3D space

This is the “space” that we are referring to.

> Whenever we graph points or think of points in latent space, we can imagine them as coordinates in space in which **points that are** *“similar”* **are closer together on the graph.**

A natural question that arises is how would we imagine space of 4D points or n-dimensional points, or even non-vectors (since the latent space representation is NOT required to be 2 or 3-dimensional vectors, and is oftentimes not since too much information would be lost).

The unsatisfying answer is, *we can’t*. We are 3-dimensional creatures that cannot fathom n-dimensional space (such that n > 3). However, there are tools such as t-SNE which can transform our higher dimensional latent space representations into representations that we *can* visualize (2D or 3D). (See **Visualizing Latent Space** section below.)

But you may be wondering, what *are* ‘similar’ images, and why does reducing the dimensionality of our data make similar images ‘closer’ together in space?

What do I mean by similar?

If we look at three images, two of a chair and one of a desk, we would easily say that the two chair images are the most *similar* whereas the desk is the most different from either of the chair images.

✍

✍

✍Two chairs and a desk.

But what makes these two chair images “more similar?” A chair has *distinguishable features* (i.e. back-rest, no drawer, connections between legs). These can all be ‘understood’ by our models by learning patterns in edges, angles, etc.

As explained, such features are packaged in the latent space representation of data.

Thus, as dimensionality is reduced, the ‘extraneous’ information which is distinct to each image (i.e. chair color) is ‘removed’ from our latent space representation, since only the most *important* features of each image are stored in the latent space representations.

As a result, as we reduce dimensionality, the representations of both chairs become less distinct and more similar. If we were to imagine them in space, they would be ‘closer’ together.

*Please note that the ‘closeness’ metric I have referred to throughout the article is an ambiguous term and *not* a definitive Euclidian distance, because there are multiple definitions of distance in space.

Why Does Latent Space Matter?

The latent space concept is definitely intriguing. But *how* is it used? *When* do we use it? And most importantly, *why*?

What we'll find is that the latent space is 'hidden' in many of our favorite image processing networks, generative models, etc.

Although the latent space is hidden from most, there *are* certain tasks in which understanding the latent space is not only helpful, but necessary.

Representation Learning

The latent space representation of our data contains all the important information needed to represent our original data point.

This representation *must* then represent the **features** of the original data.

In other words, the model *learns* the data features and simplifies its representation to make it easier to analyze.

This is at the core of a concept called **Representation Learning,** defined as a set of techniques that allow a system to discover the *representations* needed for *feature detection* or *classification* from *raw data.*

In this use case, our **latent space representations** are used to transform more complex forms of raw data (i.e. images, video), into simpler representations which are 'more convenient to process' and analyze.

Listed below are specific instances of representation learning.

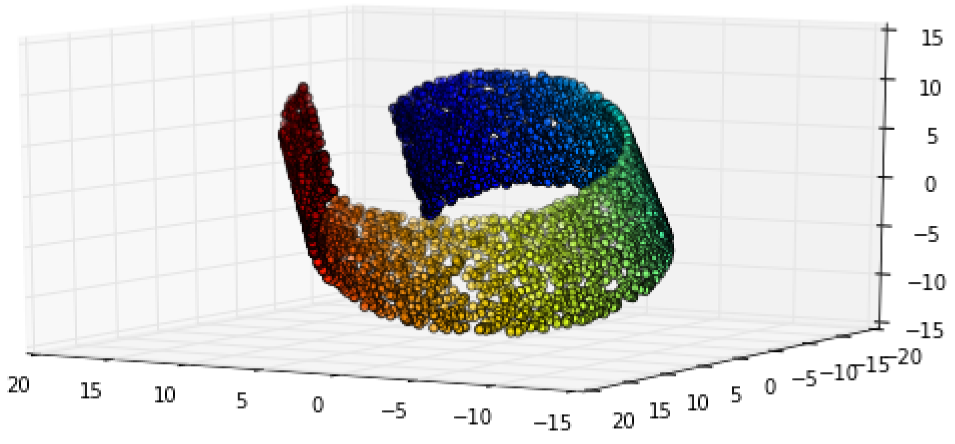
Manifolds

The latent space is an essential concept in **manifold learning**, a subfield of representation learning.

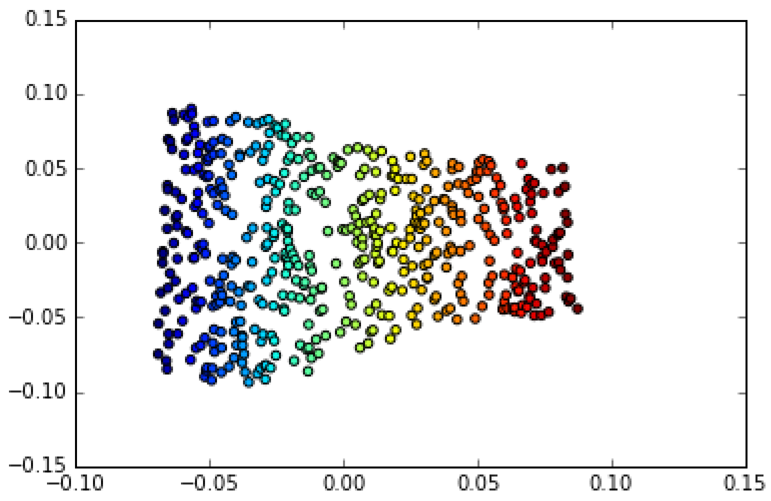
Manifolds in data science can be understood as groups or subsets of data that are 'similar' in some way.

These similarities, usually imperceptible or obscured in higher-dimensional space, can be discovered once our data has been represented in the latent space.

Take the example of a 'swiss roll' below.



✍



✍3D Representation of Swiss Roll vs. 2D Representation of same data. Example from https://datascience.stackexchange.com/a/5698

In 3D, we know that there *are* groups of similar data points that exist, but it is much more difficult to delineate such groups with higher dimensional data.

> By reducing the dimensionality of our data to 2D, which in this case could be considered a 'latent space' representation, we are able to more easily distinguish the manifolds (groups of similar data) in our dataset.

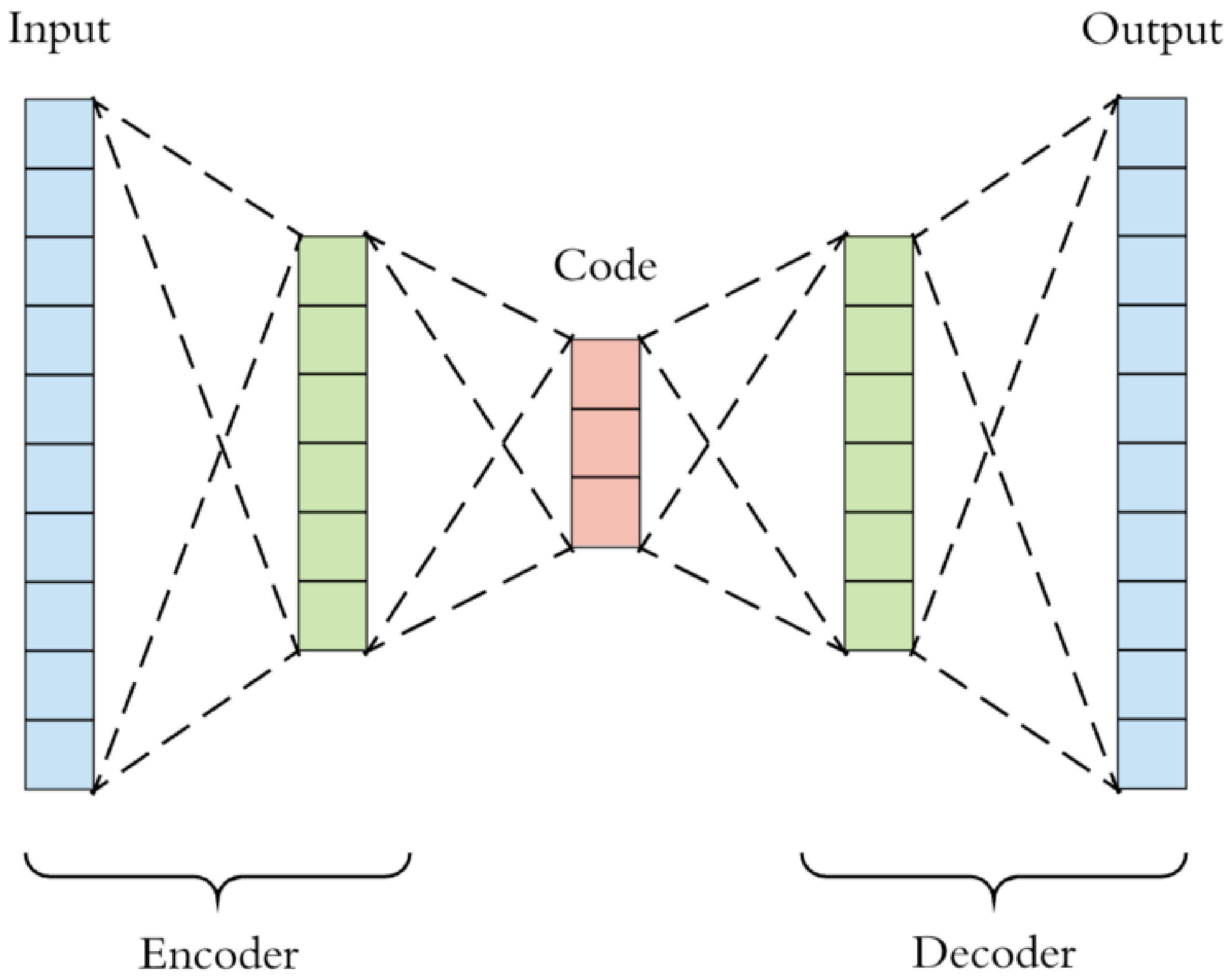
To learn more about manifolds and manifold learning, I recommend the following articles:

> **Manifolds in Data Science — A Brief Overview → https://towardsdatascience.com/manifolds-in-data-science-a-brief-overview-2e9dde9437e5** *Manifolds are an essential tool for representing data in higher dimensions. But what are manifolds, and how are they…*towardsdatascience.com → https://towardsdatascience.com/manifolds-in-data-science-a-brief-overview-2e9dde9437e5

> **2.2. Manifold learning — scikit-learn 0.22.1 documentation → https://scikit-learn.org/stable/modules/manifold.html** *Look for the bare necessities The simple bare necessities Forget about your worries and your strife I mean the bare…*scikit-learn.org → https://scikit-learn.org/stable/modules/manifold.html

Autoencoders and Generative Models

A common type of deep learning model that manipulates the 'closeness' of data in the latent space is the **autoencoder —** a neural network that acts as an identity function. In other words, an autoencoder learns to output whatever is inputted.



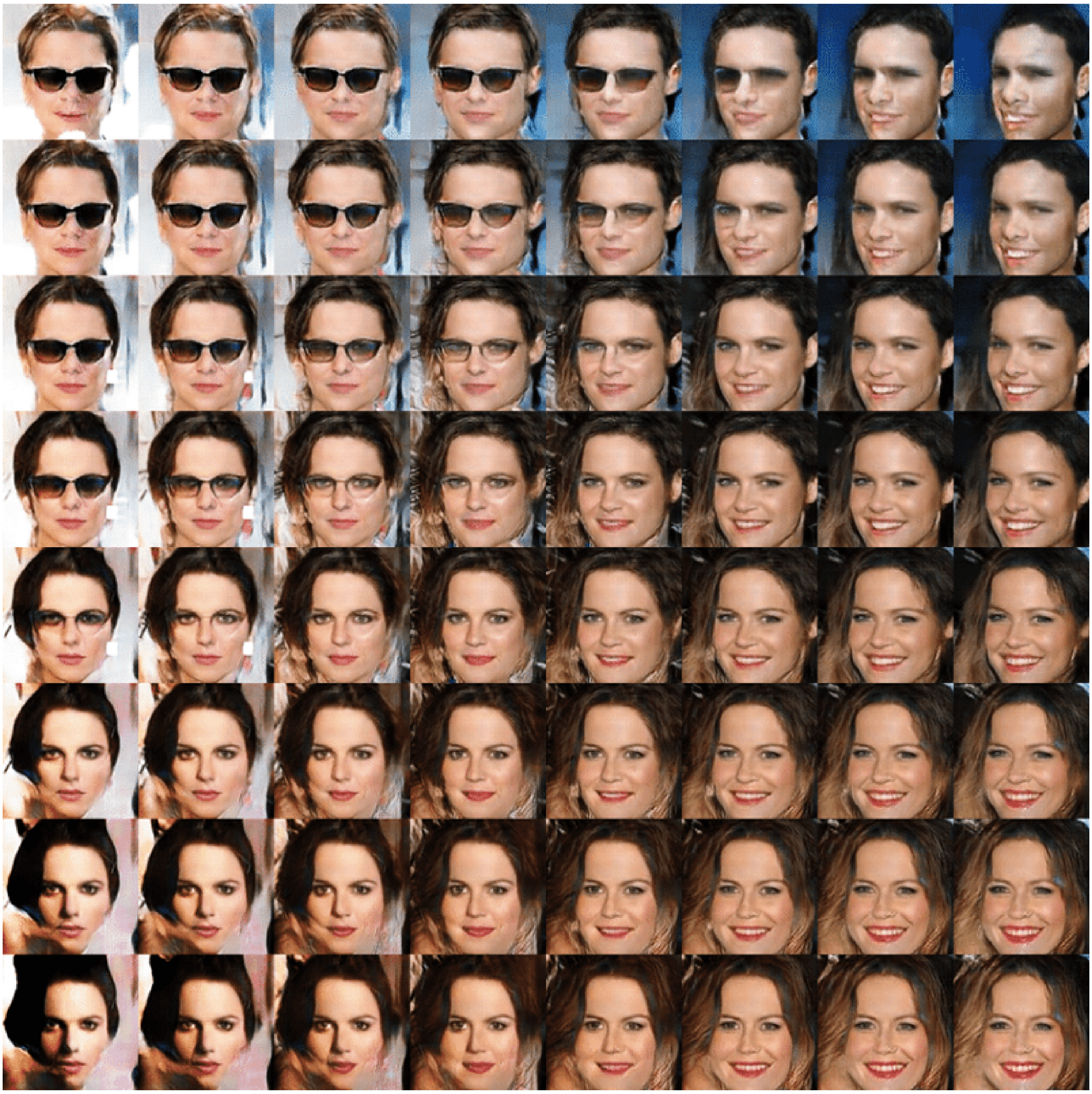
✍General architecture of an autoencoder

Now, if you're new to the field, you may be wondering, why in the world would we need a model that does this? It seems rather useless if all it outputs is itself…

Though this reasoning is valid, we don't care so much about what the model *outputs*. We care more about what the model *learns* in the process.

When we force a model to become an identity function, we are forcing it to store all of the data's relevant features in a compressed representation so that there is enough information in that compressed form such that the model can 'accurately' reconstruct it. *Sound familiar?* It should, because this compressed representation is our latent space representation (red block in image above).

We have seen how patterns can be more easily discovered in the latent space since similar data points will tend to cluster together, but we have not yet seen how we can sample points *from* this latent space to seemingly generate 'new' data.


✍Image generation through latent space interpolation. Source: *Bilinear interpolation on latent space for random noise vectors.* Figure 20

In the example above, we can generate different facial structures by *interpolating on latent space*, and using our model decoder to reconstruct the latent space representation into a 2D image with the same dimensions as our original input.

What do I mean by interpolating on latent space?

Let's say that I have compressed the chair images from the previous section into the following 2D vectors, [0.4, 0.5] and [0.45, 0.45]. Let's say the desk is compressed to [0.6, 0.75]. If I were to interpolate on latent space, I would sample points in latent space *between* the 'chair' cluster and the 'desk' cluster.

We can feed these sampled 2D vectors into the model's decoder, and voila! We get 'new' images that look like a morph between a chair and a desk. *new is in quotes because these generated images are not technically independent of the original data sample.

Below is an example of linear interpolation between two types of chairs in latent space.

Interpolation in Latent Space

✍Interpolation in Latent Space. Source: Hackernoon Latent Space Visualization.

Image generation is still an active area of research, and the latent space is an essential concept that must be understood. See the following articles for more use cases of generative models, and a hands-on example of latent space interpolation using a GAN (Generative Adversarial Network), another generative model that uses latent space representations.

> **18 Impressive Applications of Generative Adversarial Networks (GANs) → https://machinelearningmastery.com/impressive-applications-of-generative-adversarial-networks/** *A Generative Adversarial Network, or GAN, is a type of neural network architecture for generative modeling. Generative…*machinelearningmastery.com → https://machinelearningmastery.com/impressive-applications-of-generative-adversarial-networks/

> **How to Explore the GAN Latent Space When Generating Faces → https://machinelearningmastery.com/how-to-interpolate-and-perform-vector-arithmetic-with-faces-using-a-generative-adversarial-network/** *How to Use Interpolation and Vector Arithmetic to Explore the GAN Latent Space. Generative Adversarial Networks, or…* machinelearningmastery.com → https://machinelearningmastery.com/how-to-interpolate-and-perform-vector-arithmetic-with-faces-using-a-generative-adversarial-network/

Visualizing Latent Space

For more on latent space visualization, I recommend Hackernoon's article which provides a hands-on example of visualizing similarities between digit images in a 2D space with the t-SNE algorithm.

> **Latent space visualization — Deep Learning bits #2 → https://hackernoon.com/latent-space-visualization-deep-learning-bits-2-bd09a46920df** *Featured: interpolation, t-SNE projection (with gifs & examples!)*hackernoon.com → https://hackernoon.com/latent-space-visualization-deep-learning-bits-2-bd09a46920df

Key Takeaways

- The **latent space** is simply a representation of compressed data in which similar data points are closer together in space.
- Latent space is useful for learning data features and for finding simpler representations of data for analysis.
- We can understand patterns or structural similarities between data points by analyzing data in the latent space, be it through manifolds, clustering, etc.
- We can interpolate data in the latent space, and use our model's decoder to 'generate' data samples.

- We can visualize the latent space using algorithms such as t-SNE and LLE, which takes our latent space representation and transforms it into 2D or 3D.

While learning about the latent space, I was fascinated by this ‘hidden,’ yet essential concept. I hope that this article demystified the latent space representation, and provided the ‘deeper understanding’ of deep learning that I longed for as a novice.