Deep Learning

Autoencoders

Autoencoders: Outline

- Generative Adversarial Network
  - Discriminative vs. Generative Modeling,
  - Architecture of GAN,
  - Types of GANs.
- **Autoencoders**
  - **Types,**
  - **Linear autoencoder,**
  - **Undercomplete vs Overcomplete autoencoders,**
  - **Regularized autoencoders: Denoising and Sparse autoencoder.**
  - **Contractive autoencoder,**
  - **Convolutional autoencoder.**
  - **Compare GANs and autoencoders.**

Autoencoders: Why Deep Learning

- Need of Deep Learning
  - Autoencoders are used to help reduce the noise in data.
  - The process of compressing input data, encoding it, and then reconstructing it as an output.
  - Autoencoders allow you to reduce dimensionality and focus only on areas of real value.
- Objectives
  - Aim of an auto-encoder is to learn a compressed, distributed representation (encoding) for a set of data.
  - The purpose of auto encoders is dimensionality reduction.

Autoencoder: Introduction

- The purpose of autoencoders is dimensionality reduction.
- The hidden layer form a kind of encoding of the input.
- DL models to create realistic and novel images from scratch or based on given input data.
- The aim of an auto-encoder is to learn a compressed, distributed representation (encoding) for a set of data.
- DL models such as **VAEs and** GANs are used for Image generation.
- The general architecture of an autoencoder includes:
  - an encoder layer, code (bottleneck) layer, and decoder layer.
- It learns two functions:
  - an encoding function that transforms the input data, and
  - a decoding function that recreates the input data from the encoded representation.
- Example:
  - If input is a 100 dimensional vector, and
  - you have 60 neurons in the hidden layer,
  - then the auto encoder algorithm will replicate the input as a 100 dimensional vector in the output layer,
  - in the process giving you a 60 dimensional vector that encodes your input.

Autoencoder: Introduction

- Encoder
  - Input layer take raw input data
  - The hidden layers progressively reduce the dimensionality of the input, capturing important features and patterns. These layer compose the encoder.
  - The bottleneck layer (latent space) is the final hidden layer, where the dimensionality is significantly reduced. This layer represents the compressed encoding of the input data.
- Decoder
  - The bottleneck layer takes the encoded representation and expands it back to the dimensionality of the original input.
  - The hidden layers progressively increase the dimensionality and aim to reconstruct the original input.
  - The output layer produces the reconstructed output, which ideally should be as close as possible to the input data.
  - The loss function used during training is typically a reconstruction loss, measuring the difference between the input and the reconstructed output. Common choices include mean squared error (MSE) for continuous data or binary cross-entropy for binary data.
- During training, the autoencoder learns to minimize the reconstruction loss, forcing the network to capture the most important features of the input data in the bottleneck layer.

Autoencoder: Introduction

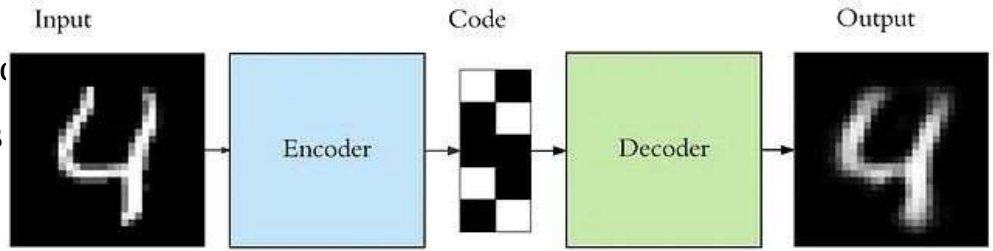
- **During training**, the autoencoder learns to minimize the reconstruction loss, forcing the network to capture the most important features of the input data in the bottleneck layer.
- **After the training process**, only the encoder part of the autoencoder is retained to encode a similar type of data used in the training process. The different ways to constrain the network are:
- Keep small Hidden Layers: If the size of each hidden layer is kept as small as possible, then the network will be forced to pick up only the representative features of the data thus encoding the data.
- Regularization: In this method, a loss term is added to the cost function which encourages the network to train in ways other than copying the input.
- Denoising: Another way of constraining the network is to add noise to the input and teach the network how to remove the noise from the data.
- Tuning the Activation Functions: This method involves changing the activation functions of various nodes so that a majority of the nodes are dormant thus, effectively reducing the size of the hidden layers.

Autoencoder: Introduction

- **During training**, the autoencoder learns to minimize the reconstruction loss, forcing the network to capture the most important features of the input data in the bottleneck layer.
- **After the training process**, only the encoder part of the autoencoder is retained to encode a similar type of data used in the training process. The different ways to constrain the network are:
- **Keep small Hidden Layers:** If the size of each hidden layer is kept as small as possible, then the network will be forced to pick up only the representative features of the data thus encoding the data.
- **Regularization:** In this method, a loss term is added to the cost function which encourages the network to train in ways other than copying the input.
- **Denoising:** Another way of constraining the network is to add noise to the input and teach the network how to remove the noise from the data.
- **Tuning the Activation Functions:** This method involves changing the activation functions of various nodes so that a majority of the nodes are dormant thus, effectively reducing the size of the hidden layers.

Autoencoder: Introduction

- Autoencoder is a type of ANN used to learn efficient codings of unlabeled data (unsupervised learning).
- An autoencoder learns two functions: an encoding function that transforms the input data, and a decoding function that recreates the input data from the encoded representation.
- Autoencoders are a specific type of feedforward neural networks where the input is the same as the output.
- They compress the input into a lower-dimensional code and then reconstruct the output from this representation.
- The code is a compact "summary" or "compression" of the input, also called the latent-space representation.
- An autoencoder consists of 3 components:
  - encoder,
  - code and
  - decoder.
- **Encoder** compresses the input and produces the co
- **Decoder** then reconstructs the input only using this

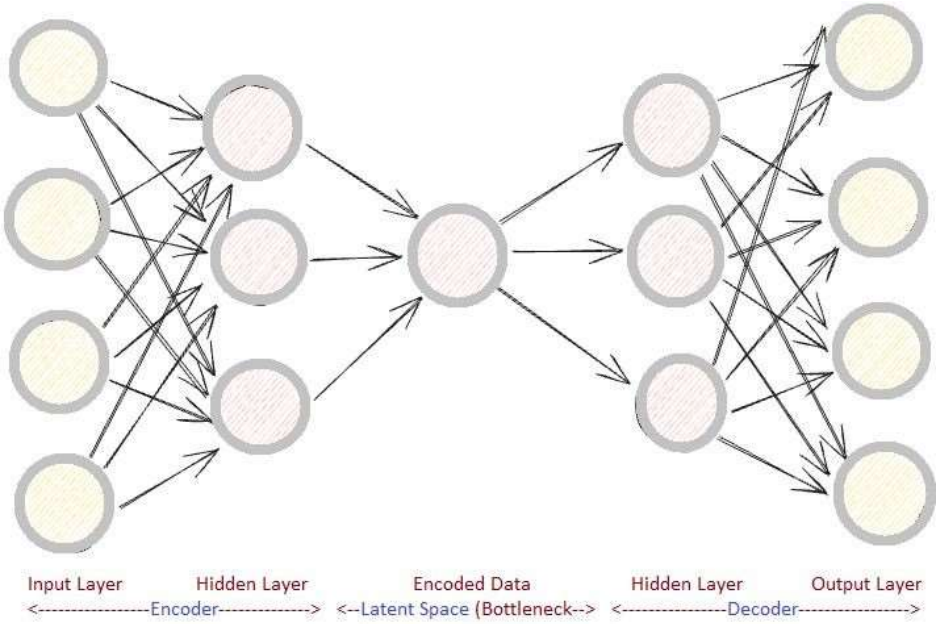


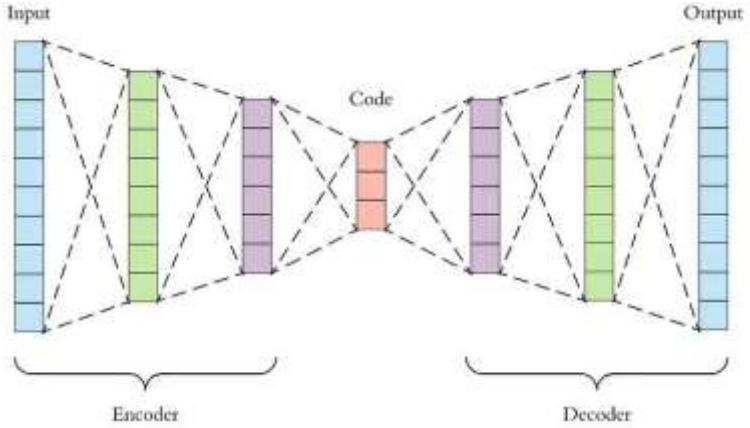
Autoencoder: Introduction

- To build an autoencoder we need 3 things: an encoding method, decoding method, and a loss function to compare the output with the target. We will explore these in the next section.
- Autoencoders helps in dimensionality reduction (or compression) algorithm with a couple of important **properties**:
- **Data-specific:** Autoencoders are only able to meaningfully compress data similar to what they have been trained on. Since they learn features specific for the given training data, they are different than a standard data compression algorithm like gzip. So we can't expect an autoencoder trained on handwritten digits to compress landscape photos.
- **Lossy:** The output of the autoencoder will not be exactly the same as the input, it will be a close but degraded representation. If you want lossless compression they are not the way to go.
- **Unsupervised:** To train an autoencoder we don't need to do anything fancy, just throw the raw input data at it. Autoencoders are considered an unsupervised learning technique since they don't need explicit labels to train on. But to be more precise they are self-supervised because they generate their own labels from the training data.

Autoencoder: Autoencoder Architecture

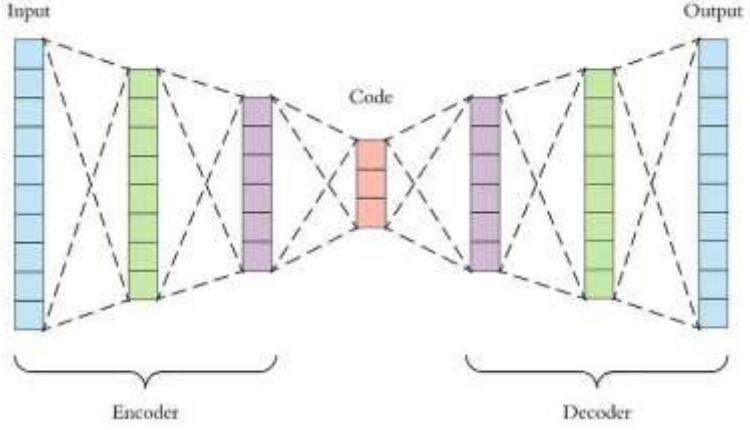
- Both the encoder and decoder are fully-connected feedforward NN.
- Autoencoders are trained the same way as ANNs via backpropagation.
- Code is a single layer of an ANN with the dimensionality of our choice.
- The number of nodes in the code layer (code size) is a hyperparameter that we set before training the autoencoder.





Autoencoder: Architecture

- This is a more detailed visualization of an autoencoder.
- First the input passes through the encoder, which is a fully-connected ANN, to produce the code.
- The decoder, which has the similar ANN structure, then produces the output only using the code.
- The goal is to get an output identical with the input.
- Note that the decoder architecture is the mirror image of the encoder.
- This is not a requirement but it's typically the case.
- The only requirement is the dimensionality of the input and output needs to be the same.
- Anything in the middle can be played with.

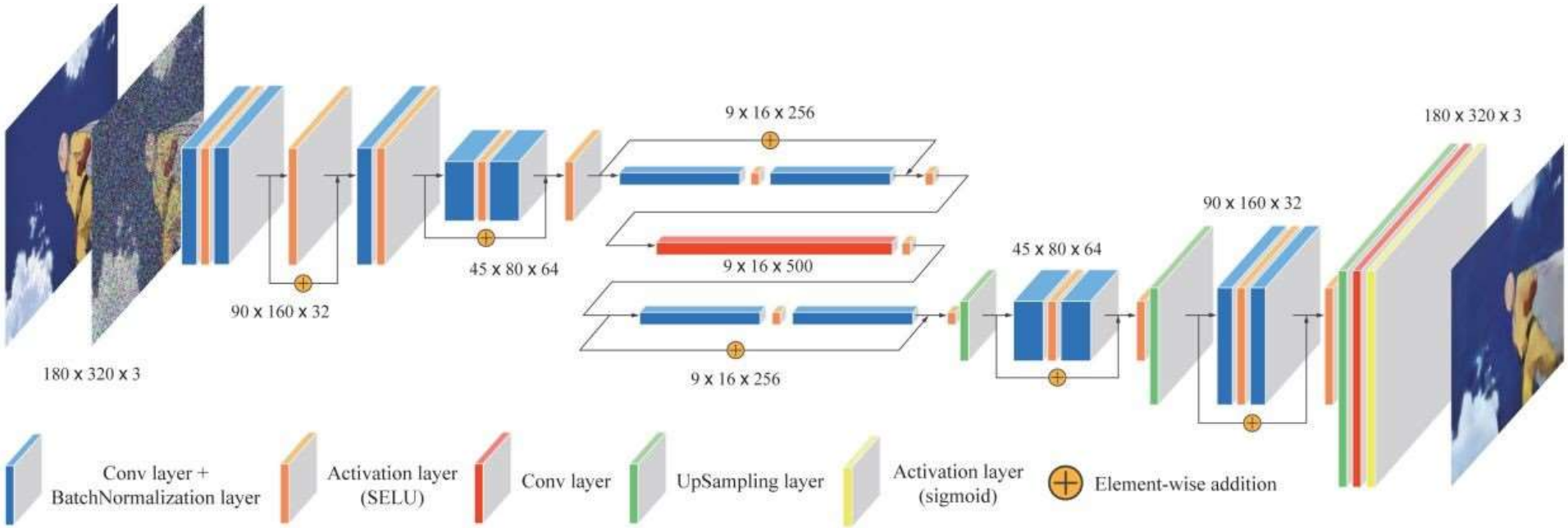


Autoencoder: Architecture

- There are 4 hyperparameters that we need to set before training an autoencoder:
- **Code size:** number of nodes in the middle layer. Smaller size results in more compression.
- **Number of layers:** the autoencoder can be as deep as we like. In the figure above we have 2 layers in both the encoder and decoder, without considering the input and output.
- **Number of nodes per layer:** the autoencoder architecture we're working on is called a stacked autoencoder since the layers are stacked one after another.
    - Usually stacked autoencoders look like a "sandwitch".
    - # nodes per layer decreases with each subsequent layer of the encoder, and increases back in the decoder.
    - Also the decoder is symmetric to the encoder in terms of layer structure.
    - As noted above this is not necessary and we have total control over these parameters.
- **Loss function:** we either use mean squared error (mse) or binary crossentropy.
    - If the input values are in the range [0, 1] then we typically use crossentropy,
    - otherwise we use the mean squared error.

Autoencoder: Architecture

- Autoencoder



Autoencoder: Architecture

- We can make autoencoder very powerful by increasing # layers, nodes per layer and most importantly the code size
- Increasing these hyperparameters will let the autoencoder to learn more complex codings.
- Be careful while making it too powerful. Then it will
  - Simply learn to copy its inputs to the output, without learning any meaningful representation
  - Just mimic the identity function.
  - Reconstruct training data perfectly, but result in overfitting without being able to generalize to new instances
- This is why we prefer a “sandwitch” architecture, and deliberately keep the code size small.
- Since the coding layer has a lower dimensionality than the input data, the autoencoder is said to be **undercomplete**.
- It won’t be able to directly copy its inputs to the output, and will be forced to learn intelligent features.
- For example:
  - Digit “1” is somewhat straight line & digit “0” is circular, it will learn fact and encode it in compact form.
- If the input data was completely random without any internal correlation or dependency,
  - then an undercomplete autoencoder won’t be able to recover it perfectly

Autoencoder: Types of Autoencoders

- **Types of Autoencoders**
  - Undercomplete autoencoders: one of the simplest types of autoencoders.
  - Vanilla Autoencoders: Basic autoencoders that efficiently encode and decode data.
  - Denoising Autoencoders: Improved robustness to noise and irrelevant information.
  - Sparse Autoencoders: Learn more compact and efficient data representations.
  - Contractive Autoencoders: Generate representations less sensitive to minor data variations.
  - Variational Autoencoders (for generative modelling): Generate new data points that resemble the training data.
  - Convolutional Autoencoder
- The choice of autoencoder depends on the specific task and data characteristics.

Autoencoder: Undercomplete Autoencoder

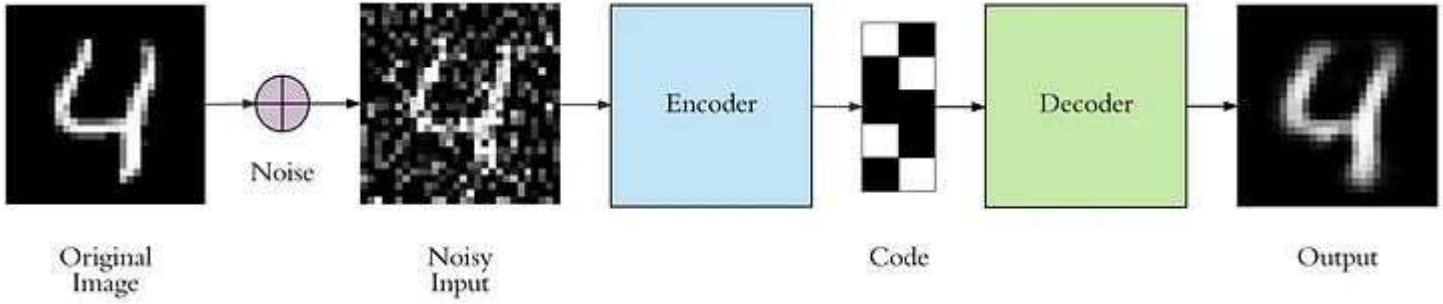
- An undercomplete autoencoder is one of the simplest types of autoencoders.
- Undercomplete autoencoder takes in an image and tries to predict the same image as output, thus reconstructing the image from the compressed bottleneck region.
- Undercomplete autoencoders are truly unsupervised as they do not take any form of label, the target being the same as the input.
- The primary use of autoencoders like such is the generation of the latent space or the bottleneck, which forms a compressed substitute of the input data and can be easily decompressed back with the help of the network when needed.
- This form of compression in the data can be modeled as a form of dimensionality reduction.

Autoencoder: Undercomplete Autoencoder

- This is the simplest version of an autoencoder.
- In this case, we don’t have an explicit regularization mechanism, but we ensure that the size of the bottleneck is always lower than the original input size to avoid overfitting.
- This type of configuration is typically used as a dimensionality reduction technique (more powerful than PCA since its also able to capture non-linearities in the data).
- Under complete autoencoders is an unsupervised neural network that you can use to generate a compressed version of the input data.
- It is done by taking in an image and trying to predict the same image as output, thus reconstructing the image from its compressed bottleneck region.
- The primary use for autoencoders like these is generating a latent space or bottleneck, which forms a compressed substitute of the input data and can be easily decompressed back with the help of the network when needed.

Autoencoder: Denoising Autoencoders

- Keeping the code layer small forced our autoencoder to learn an intelligent representation of the data.
- There is another way to force the autoencoder to learn useful features,
  - which is adding random noise to its inputs and making it recover the original noise-free data.
- This way the autoencoder can't simply copy the input to its output because the input also contains random noise.
- We are asking it to subtract the noise and produce the underlying meaningful data.
- This is called a denoising autoencoder.



Autoencoder: Denoising Autoencoders

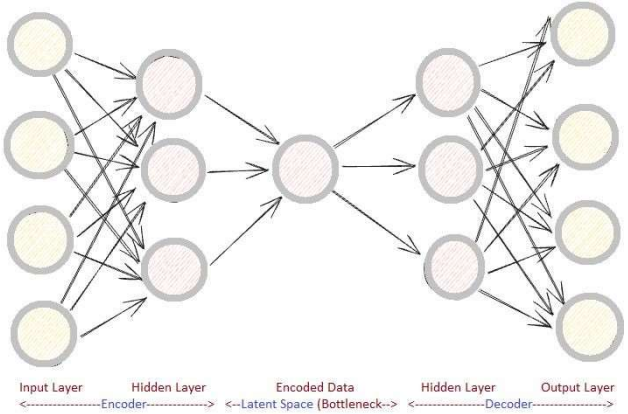
- With Denoising Autoencoders, the input and output of the model are no longer the same.
- For example, the model could be fed some low-resolution corrupted images and work for the output to improve the quality of the images.
- In order to assess the performance of the model and improve it over time, we would then need to have some form of labeled clean image to compare with the model prediction.
- Want to remove noise from an image; If so, then denoising autoencoders are useful.
- Denoising autoencoders are similar to regular autoencoders in that they take an input and produce an output.
- However, they differ because they don't have the input image as their ground truth.
- Instead, they use a noisy version.
- It is because removing image noise is difficult when working with images.
- Adenoising autoencoder, we feed the noisy idea into our network and
  - let it map it into a lower-dimensional manifold where filtering out noise becomes much more manageable.
- The loss function usually used with these networks is L2 or L1 loss.

Autoencoder: Denoising Autoencoder

- Denoising autoencoder works on a partially corrupted input and trains to recover the original undistorted image.
- As mentioned above, this method is an effective way to constrain the network from simply copying the input and
  - thus learn the underlying structure and important features of the data.
- **Advantages**
  - This type of autoencoder can extract important features and reduce the noise or the useless features.
  - It can be used as a form of data augmentation,
    - the restored images can be used as augmented data thus generating additional training samples.
- **Disadvantages**
  - Selecting right type and level of noise to introduce can be challenging and may require domain knowledge.
  - Denoising process can result into loss of some information that is needed from the original input.
  - This loss can impact accuracy of the output.

Autoencoder: Sparse Autoencoder

- A Sparse Autoencoder is quite similar to an Undercomplete Autoencoder, but their main difference lies in how regularization is applied.
- In fact, with Sparse Autoencoders, we don’t necessarily have to reduce the dimensions of the bottleneck, but we use a loss function that tries to penalize the model from using all its neurons in the different hidden layers (Figure).
- This penalty is commonly referred to as a sparsity function, and it's quite different from traditional regularization techniques since it doesn’t focus on penalizing the size of the weights but the number of nodes activated.
- In this way, different nodes could specialize for different input types and be activated/deactivated depending on the specifics of the input data.
- This sparsity constraint can be induced by using L1 Regularization and KL divergence,
  - effectively preventing the model from overfitting.



Autoencoder: Sparse Autoencoder

- Sparse autoencoders are controlled by changing the number of nodes at each hidden layer.
- Since it is impossible to design a neural network with a flexible number of nodes at its hidden layers, sparse autoencoders work by penalizing the activation of some neurons in hidden layers.
- It means that a penalty directly proportional to the number of neurons activated is applied to the loss function.
- As a means of regularizing the neural network, the sparsity function prevents more neurons from being activated.
- There are two types of regularizers used:
  - The L1 Loss method is a general regularizer we can use to add magnitude to the model.
  - The KL-divergence method considers the activations over a collection of samples at once rather than summing them as in the L1 Loss method. We constrain the average activation of each neuron over this collection.

Autoencoder: Sparse Autoencoder

- **Sparse Autoencoder** contains more hidden units than the input but only a few are allowed to be active at once.
  - This property is called the sparsity of the network.
- The sparsity of the network can be controlled by either manually zeroing the required hidden units,
  - tuning the activation functions or by adding a loss term to the cost function.
- **Advantages**
  - Sparsity constraint in sparse autoencoders helps in filtering out noise and irrelevant features during the encoding process.
  - These autoencoders often learn important and meaningful features due to their emphasis on sparse activations.
- **Disadvantages**
  - The choice of hyperparameters play a significant role in the performance of this autoencoder.
  - Different inputs should result in the activation of different nodes of the network.
  - The application of sparsity constraint increases computational complexity.

Autoencoder: Sparse Autoencoders

- Two ways to force the autoencoder to learn useful features:
  - keeping the code size small and
  - denoising autoencoders.
- Third method is using regularization.
  - We can regularize the autoencoder by using a sparsity constraint such that
    - only a fraction of the nodes would have nonzero values, called active nodes.
- So, we add a penalty term to the loss function such that only a fraction of the nodes become active.
- This forces the autoencoder to represent each input as a combination of small number of nodes, and demands it to discover interesting structure in the data.
- This method works even if the code size is large, since only a small subset of the nodes will be active at any time.

Autoencoder: Variational Autoencoder

- In every type of Autoencoder considered so far, the encoder outputs a single value for each dimension involved. With Variational Autoencoders (VAE), we make this process instead probabilistic, creating a probability distribution for each dimension.
- The decoder can then sample a value from each distribution describing the different dimensions and construct the input vector, which it can then be used to reconstruct the original input data.
- One of the main applications of Variational Autoencoders is for generative tasks.
- In fact, sampling the latent model from distributions can enable the decoder to create new forms of outputs that were previously not possible using a deterministic approach.

Autoencoder: Variational Autoencoder

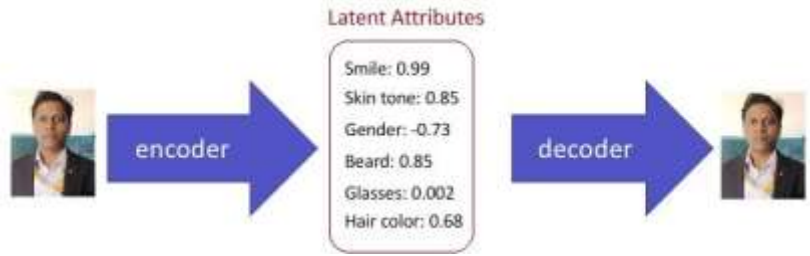
- **Variational autoencoder** makes strong assumptions about the distribution of latent variables and uses the Stochastic Gradient Variational Bayes estimator in the training process.
- **Advantages**
  - It is used to generate new data points that resemble the original training data.
  - These samples are learned from the latent space.
  - It is probabilistic framework to learn a compressed representation of the data that captures its underlying structure and variations, so it is useful in detecting anomalies and data exploration.
- **Disadvantages**
  - It use approximations to estimate the true distribution of the latent variables.
  - This approximation introduces some level of error, which can affect the quality of generated samples.
  - The generated samples may only cover a limited subset of the true data distribution.
    - This can result in a lack of diversity in generated samples.

Autoencoder: Variational Autoencoders

- **Variational Autoencoders (VAE):**
  - this is a more modern and complex use-case of autoencoders and we will cover them in another article.
  - VAE learns the parameters of the probability distribution modeling the input data, instead of learning an arbitrary function in the case of vanilla autoencoders.
  - By sampling points from this distribution we can also use the VAE as a generative model.
- **Variational Autoencoders (VAEs)** are a type of Generative Models in DL
  - that can learn to encode and decode data,
  - enabling generation of new & realistic data
- Encoding:
  - I/p data is passed through an encoder NN,
  - that learns to map the data to the lower dimension known as latent space.
  - Here the latent space captures the essential features from the input data.
- Sampling: In latent space, the random vectors are sampled to generate new data points.
- Decoder:
  - The sampled vectors from latent space are passed onto the decoder NN that reconstructs the original data.
  - Here the aim of the decoder is to minimise the reconstruction error.

Autoencoder: Variational Autoencoders

- This models that address a specific problem with standard autoencoders.
- When you train an autoencoder, it learns to represent the input just in a compressed form called the latent space or the bottleneck.
- However, this latent space formed after training is not necessarily continuous and, in effect, might not be easy to interpolate.
- It deal with this specific topic and express their latent attributes as a probability distribution, forming a continuous latent space that can be easily sampled and interpolated.
- Standard and variational autoencoders learn to represent the input just in a compressed form called the latent space or the bottleneck.
- Therefore, the latent space formed after training the model is not necessarily continuous and, in effect, might not be easy to interpolate.
- For example: This is what a variational autoencoder would learn from the input:



Autoencoder: Contractive Autoencoders

- Similar to other autoencoders, contractive autoencoders perform task of learning a representation of the image while passing it through a bottleneck and reconstructing it in the decoder.
- The contractive autoencoder also has a regularization term to prevent the network from learning the identity function and mapping input into the output.
- It work on the basis that similar inputs should have similar encodings and a similar latent space representation.
- It means that the latent space should not vary by a huge amount for minor variations in the input.
- To train a model that works along with this constraint, we have to ensure that the derivatives of the hidden layer activations are small with respect to the input data.
- The main idea behind Contractive Autoencoders is that given some similar inputs, their compressed representation should be quite similar (neighborhoods of inputs should be contracted in small neighborhood of outputs).
- In mathematical terms, this can be enforced by keeping input hidden layer activations derivatives small when fed similar inputs.

Autoencoder: Contractive Autoencoders

- Contractive Autoencoders
- The input is passed through a bottleneck in a contractive autoencoder and then reconstructed in the decoder.
- The bottleneck function is used to learn a representation of the image while passing it through.
- The contractive autoencoder also has a regularization term to prevent the network from learning the identity function and mapping input into output.
- To train a model that works along with this constraint,
  - we need to ensure that the derivatives of the hidden layer activations are small concerning the input.

Autoencoder: Convolutional Autoencoder

- Convolutional autoencoder
- To work with image data, Convolutional Autoencoders replace traditional feedforward neural networks with CNN for both the encoder and decoder steps.
- Updating type of loss function, etc., this type of Autoencoder can also be made,
  - for example, Sparse or Denoising, depending on your use case requirements.

Autoencoder: Convolutional Autoencoder

- Convolutional autoencoder is a type of autoencoder that use CNNs as their building blocks.
- The encoder consists of multiple layers that take a image or a grid as input and pass it through different convolution layers thus forming a compressed representation of the input.
- The decoder is the mirror image of the encoder it deconvolves the compressed representation and tries to reconstruct the original image.
- **Advantages**
  - It can compress high-dimensional image data into a lower-dimensional data.
  - This improves storage efficiency and transmission of image data.
  - Convolutional autoencoder can reconstruct missing parts of an image.
  - It can also handle images with slight variations in object position or orientation.
- **Disadvantages**
  - These autoencoder are prone to overfitting.
  - Proper regularization techniques should be used to tackle this issue.
  - Compression of data can cause data loss which can result in reconstruction of a lower quality image.

Autoencoder: Autoencoder vs Variational Autoencoder

- Autoencoder (AE)
  - Used to generate a compressed transformation of input in a latent space
  - The latent variable is not regularized
  - Picking a random latent variable will generate garbage output
  - The latent variable has a discontinuity
  - Latent variable is deterministic values
  - The latent space lacks the generative capability
- Variational Autoencoder (VAE)
  - Enforces conditions on the latent variable to be the unit norm
  - The latent variable in the compressed form is mean and variance
  - The latent variable is smooth and continuous
  - A random value of latent variable generates meaningful output at the decoder
  - I/p of decoder is stochastic and is sampled from a gaussian with mean and variance of o/p of the encoder.
  - Regularized latent space
  - The latent space has generative capabilities.

Autoencoder: Compare GANs and Autoencoders

- Compare GANs and Autoencoders

Topics	Generative Adversarial Networks (GANs)	Variational Autoencoders (VAEs)
Functionality	Composed of two models (a generator and a discriminator) that compete with each other. The generator creates fake samples and the discriminator attempts to distinguish between real and fake samples.	Composed of an encoder and a decoder. The encoder maps inputs to a latent space, and the decoder maps points in the latent space back to the input space.
Output Quality	Can generate high-quality, realistic outputs. Known for generating images that are hard to distinguish from real ones.	Generally produces less sharp or slightly blurrier images compared to GANs. However, this may depend on the specific implementation and problem domain.
Latent Space	Often lacks structure, making it hard to control or interpret the characteristics of the generated samples.	Creates a structured latent space which can be more easily interpreted and manipulated.
Training Stability	Training GANs can be challenging and unstable, due to the adversarial loss used in training.	Generally easier and more stable to train because they use a likelihood-based objective function.

Autoencoder: Compare GANs and Autoencoders

- Compare GANs and Autoencoders

Topics	Generative Adversarial Networks (GANs)	Variational Autoencoders (VAEs)
Use Cases	Great for generating new, creative content. Often used in tasks like image generation, text-to-image synthesis, and style transfer.	Useful when there's a need for understanding the data-generating process or controlling the attributes of the generated outputs. Often used in tasks like anomaly detection, denoising, or recommendation systems.
Activation function	In generator network, the final layer typically uses a Tanh activation function, mapping the output to a range that matches the preprocessed input data, usually from -1 to 1. For intermediate layers, GANs commonly employ activation functions like ReLU or LeakyReLU, effectively circumventing the vanishing gradient problem. This allows GANs to learn and propagate more diverse gradients back through the network. The discriminator network uses a Sigmoid activation function in its output layer	The encoder network in a VAE transforms the input data into two components: a mean and a standard deviation. Since the mean can span any real value and the standard deviation needs to be positive, these outputs typically forego activation functions. However, much like GANs, VAEs use activation functions such as ReLU or LeakyReLU in intermediate layers to introduce non-linearity and mitigate the vanishing gradient problem. The decoder, which maps points from the latent space back to the data space, can employ various activation functions in its final layer.

Autoencoder: Compare GANs and Autoencoders

- Compare GANs and Autoencoders

Topics	GAN	Autoencoder
Performance	GANS are a bit improved in this respect.	VAEs are regarded as more difficult to optimize.
Applications	Used mostly for super resolution, image to image translation, image generation etc.	Used mostly for image denoising and anomaly detection and generation.
Complexity	The simplicity of the adversarial training concept that rules GANS allows the GAN community to contribute more than VAE.	It has a higher degree of complexity in VAES theoretical basis in terms of probabilistic model and variational inference for approximating intractable integrals.
Complexity	GAN wants to synchronize the discriminator with the generator during the training to achieve greater results. Therefore, it is more complex to train.	In terms of training, VAE has an enforced tradeoff between mixing and power of reconstruction generation. So it's simpler to train.
Latent Space	GAN accomplishes the task to generate non blurry images as latent vectors come from random noise.	VAE generates a blurry image compared to GAN as latent vector is generated by encoder.
Loss Function	Has two loss functions generator's loss and discriminator loss.	Has error function to be minimized - KL divergence and reconstruction error.
Output	GAN finds a point in the generator discriminator to help achieve fake data as one tries to deceive the other.	VAE maximizes the probability of generated output with respect to input to get an output by compressing the input to latent space.

Autoencoder: Use Cases

- **Anomaly detection:** autoencoders can identify data anomalies using a loss function that penalizes model complexity. It can be helpful for anomaly detection in financial markets, where you can use it to identify unusual activity and predict market trends.
- **Data denoising image and audio:** autoencoders can help clean up noisy pictures or audio files. You can also use them to remove noise from images or audio recordings.
- **Image inpainting:** autoencoders have been used to fill in gaps in images by learning how to reconstruct missing pixels based on surrounding pixels. For example, if you're trying to restore an old photograph that's missing part of its right side, the autoencoder could learn how to fill in the missing details based on what it knows about the rest of the photo.
- **Information retrieval:** autoencoders can be used as content-based image retrieval systems that allow users to search for images based on their content.

Autoencoder: Applications of Autoencoders

- Unfortunately autoencoders are not widely used in real-world applications.
- As a compression method, they don't perform better than its alternatives,
  - for example jpeg does photo compression better than an autoencoder.
- And the fact that autoencoders are data-specific makes them impractical as a general technique.
- **Applications of Autoencoders**
- 1. File Compression: reduce the **dimensionality** of input data which we in common refer to as file compression.
- 2. Image De-noising: Autoencoders are also used as noise removal techniques
- 3. Image Transformation: transform B/W images to colored one and vice versa
- 4. Anomaly detection
- 5. Generation of image and time series data

Autoencoder: FAQs

- 1. What are autoencoders used for?
- Using an autoencoder, you can compress input data, encode it, and then reconstruct the data in a different format to reduce dimensionality. Autoencoders help you focus on only the most critical areas of your data.
- 2. How do autoencoders work?
- Autoencoders are neural networks that you can use to compress and reconstruct data. The encoder compresses input, and the decoder attempts to recreate the information from this compressed version.
- 3. Is autoencoder a CNN?
- Convolutional Autoencoders are autoencoders that use CNNs in their encoder/decoder parts. The term "convolutional" refers to convolving an image with a filter to extract information, which happens in a CNN.
- 4. Is the autoencoder supervised or unsupervised?
- Autoencoders can be used to learn a compressed representation of the input. Autoencoders are unsupervised, although they are trained using supervised learning methods.
- 5. When should we not use autoencoders?
- An autoencoder could misclassify input errors that are different from those in the training set or changes in underlying relationships that a human would notice. Another drawback is you may eliminate the vital information in the input data.

Summary

- The discriminator is typically a simple Convolution Neural Network (CNN) in simple architectures.
- **GANs** includes supervised vs. unsupervised learning and discriminative vs. generative modeling.
- GANs are an architecture for automatically training a generative model by treating the unsupervised problem as supervised and using both a generative and a discriminative model.
- GANs provide a path to sophisticated domain-specific data augmentation and a solution to problems that require a generative solution, such as image-to-image translation.
- **Autoencoders** are a very useful dimensionality reduction technique.
- They are very popular as a teaching material in introductory DL courses, most likely due to their simplicity.

Summary

- An autoencoder is an unsupervised learning technique for neural networks that learns efficient data representations (encoding) by training the network to ignore signal “noise.”
- Autoencoders can be used for image denoising, image compression, and, in some cases, even generation of image data.
- While autoencoders might seem easy at the first glance (as they have a very simple theoretical background), making them learn a representation of the input that is meaningful is quite difficult.
- Autoencoders like the undercomplete autoencoder and the sparse autoencoder do not have large scale applications in computer vision compared to VAEs and DAEs.

References

A. Text Books:

1. Satish Kumar, "Neural Networks: A Classroom Approach", McGraw Hill Education; 2ed, 2017.
2. Jacek M. Zurada, "Introduction to Artificial Neural Systems", West Publishing Company, 1092.
3. Ian Goodfellow, Yoshua Bengio and Aaron Courville, "Deep Learning", MIT Press Ltd, 2016
4. Li Deng and Dong Yu, "Deep Learning Methods and Applications", Now Publishers Inc., 2014.
5. Mykel J. Kochenderfer and Tim A. Wheeler, "Algorithms for Optimization", The MIT Press, Cambridge, Massachusetts London.

B. References:

6. Simon Haykin, "Neural Network - A Comprehensive Foundation", 2ed, Pearson Education, 2005.
7. S.N. Sivanandam and S.N. Deepa, "Principles of soft computing", Wiley India
8. François Chollet, "Deep learning with Python," New York: Manning, Vol. 361. 2018.
9. Douwe Osinga, "Deep Learning Cookbook", O′Reilly; 1st edition, 2018, SPD Publishers.

Thank You.