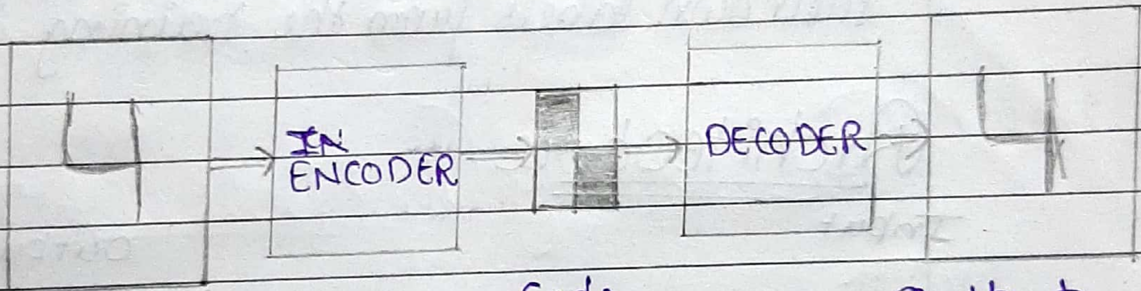
Autoencoders

Page No.
Date

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. The encoder compress the input and produces the code, the decoder then reconstructs the input only using this code.



Input Code Output

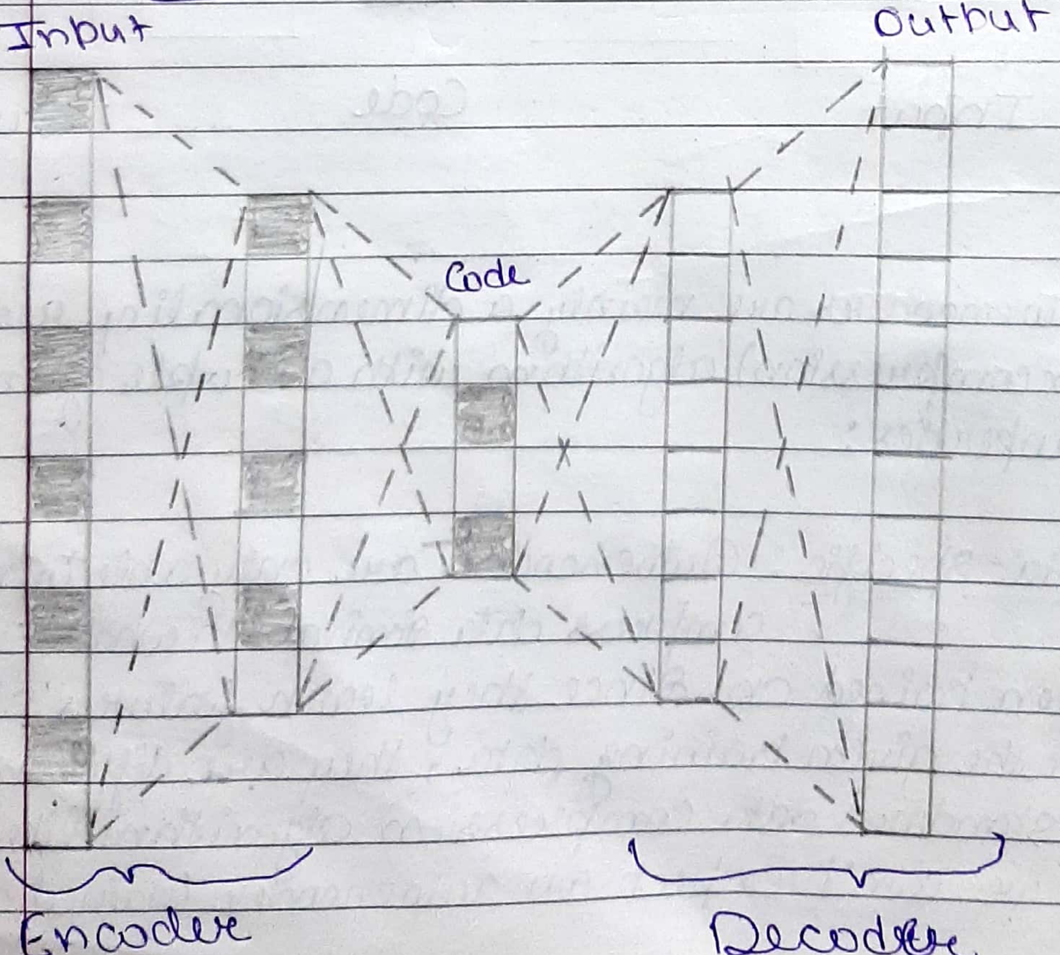
Autoencoders are mainly a dimensionality reduction (or compression) algorithm with a couple of important properties:

- 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 grip. 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.

Architecture



This is the a more detailed visualization of an autoencoder. fist 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 same. Anything in the middle can be played with

Hyperparameters

There are 3 hyperparameters that we need to set before training an autoencoder:

1) code size: number of nodes in the middle layer. Smaller size results in the more compression.

2) Number of layers: the autoencoder can be as deep as we like.

3) loss function: we either use mean squared error (mse) or binary crossentroy.

If the input values are in the range [0, 1] then we typically use crossentropy, otherwise we use the mean-squared error

Advice

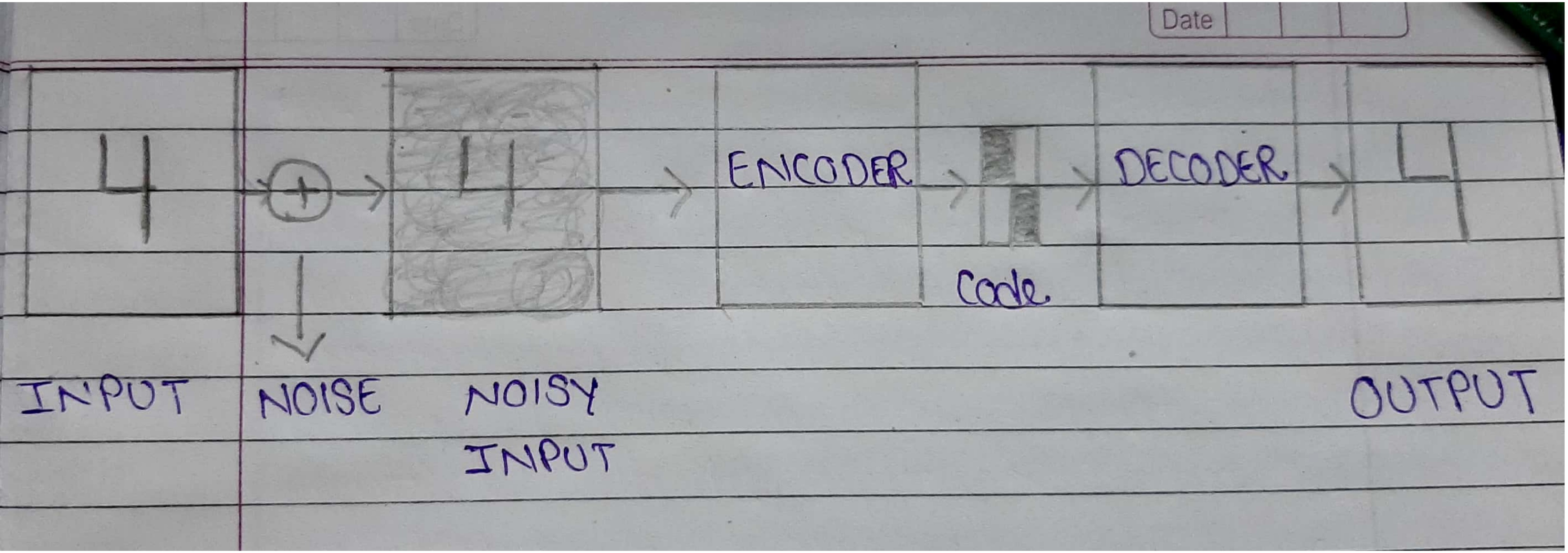
Page No. ______
Date

We have total control over the architecture of the autoencoder. We can make it very powerful by increasing the numbers of layers, nodes per layer and most importantly the code size. Increasing these hyperparameters will let the autoencoder to learn more complex codings. But we should be careful to not make it too powerful. Otherwise the autoencoder will simply learn to copy its inputs to the output, without learning any meaningful representation. It will just mimic the identity function. The autoencoder will reconstruct the training data perfectly, but it will be overfitting without being able to generalize to new instances, which is not what we want.

(b) Denoising Autoencoders

Keeping the code layer small forced our auto-encoders to learn an intelligent representation of the data. There is another way to force the auto-encoder 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 meaning data. This is called a denoising autoencoder.

Scanned by CamScanner


Date
4
ENCODER
Code
DECODER
4
INPUT
NOISE
NOISY
INPUT
OUTPUT