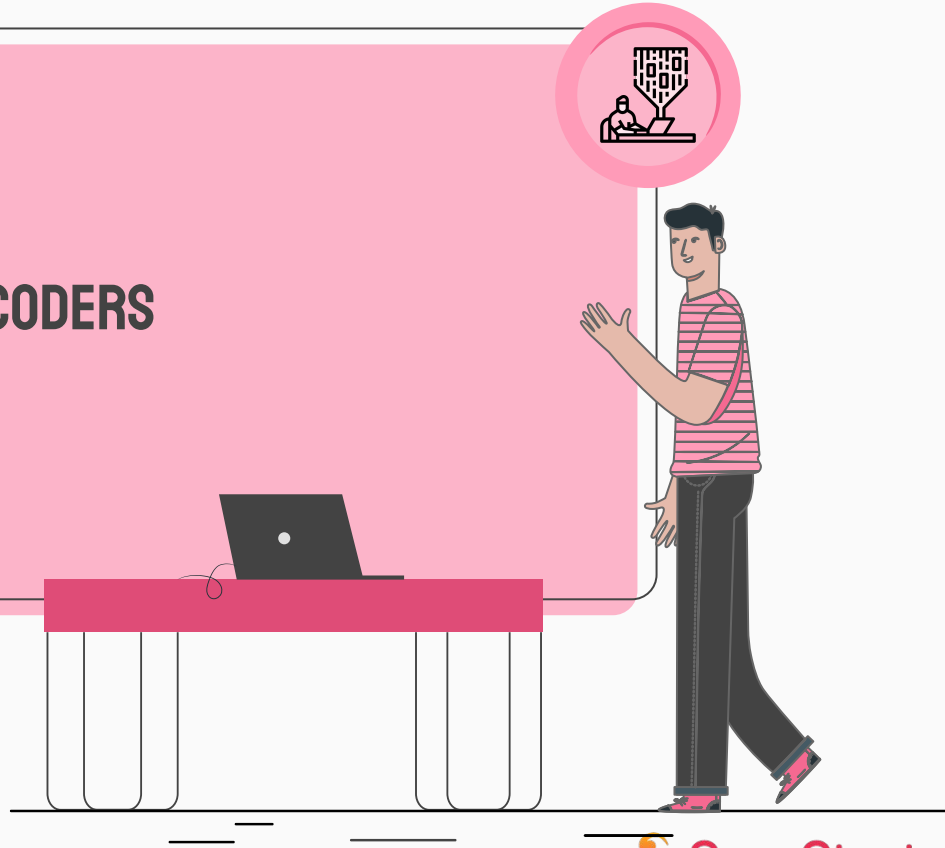
AUTOENCODERS

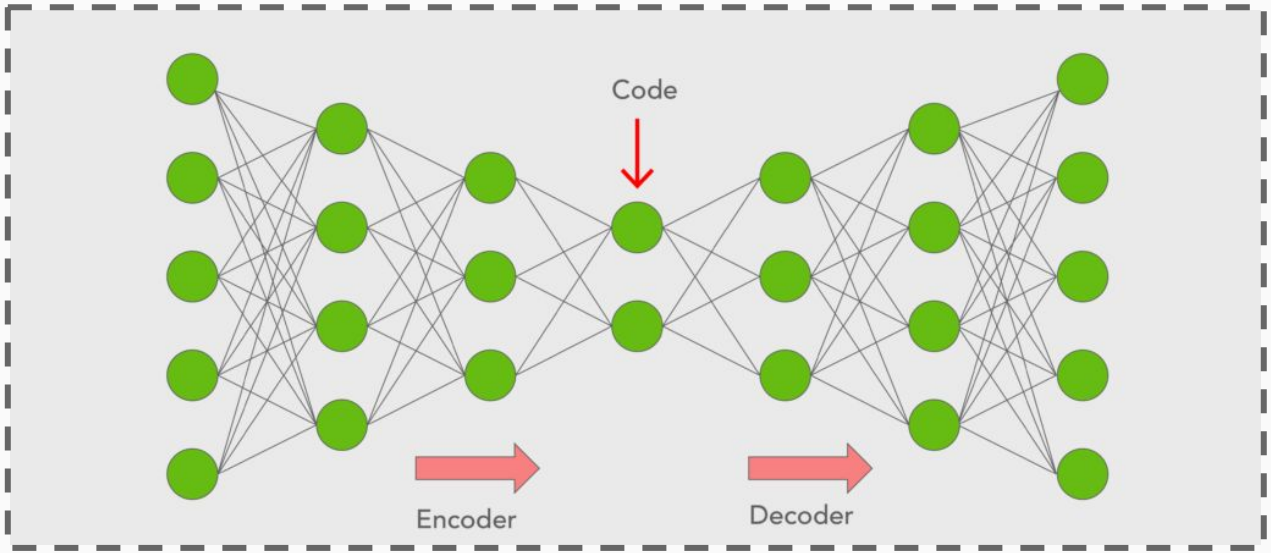


WHAT ARE AUTOENCODERS?

An autoencoder is a neural network that learns data representations in an unsupervised manner.

The structure consists of,

1. An encoder, which learns the compact (or compressed) representation of input data
2. A decoder, which reconstructs the input data by decompressing the compact representation

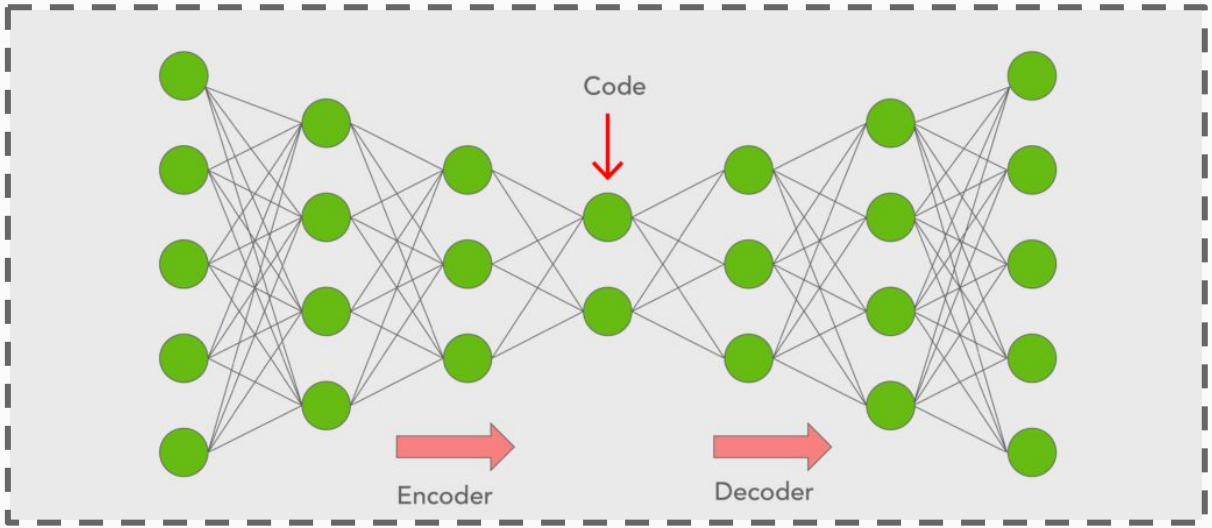


SureStart

AUTOENCODERS

An autoencoder has 3 main features –

1. It is data-specific: Autoencoders will only be able to compress data similar to what they have been trained on. Autoencoders work well if correlations exists within input features; it performs poorly if the all-input features are independent.



2. It is lossy: The quality of decompressed outputs will be degraded compared to the original inputs (similar to MP3 or JPEG compression).

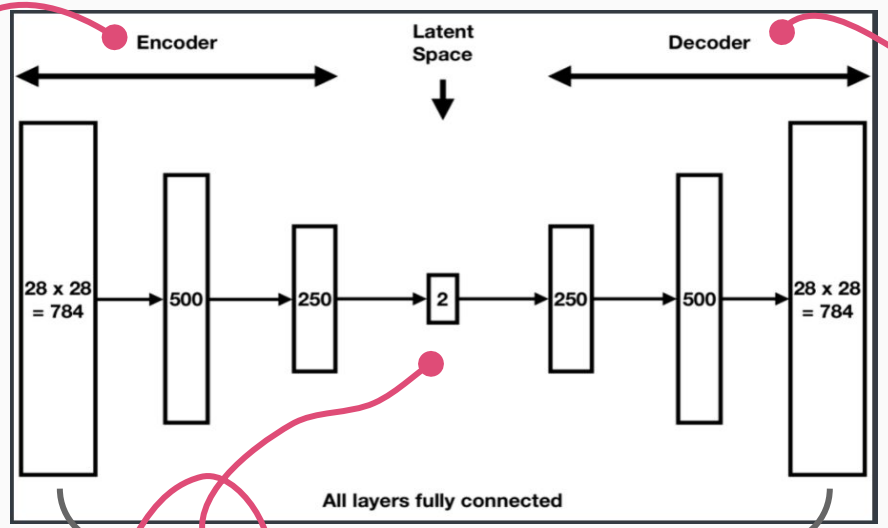
3. It learns automatically: Autoencoders learn automatically, but it needs appropriate training data.

SureStart

MAIN COMPONENTS OF AN AUTOENCODER

An **encoder** or recognition network that converts the inputs to an internal representation.

A **decoder** or generative network that converts the internal representation into efficient data representations.



Latent space: The middle layer (hidden layer) of the autoencoder is called a latent space. It is also referred to as code, latent variables, or latent representation.

The latent space of the network contains fewer units compared to the input or output layers in order to hold the reduced representation of the input.

A **distance function** to calculate the information loss between the compressed representation and the decompressed representation of data (i.e. a "loss" function).

SureStart

WHY TO USE AUTOENCODERS?

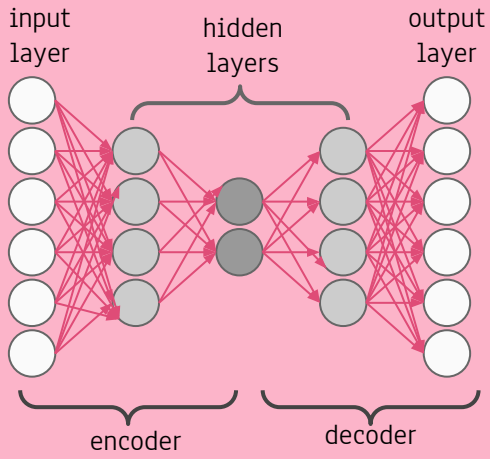
- **Dimensionality reduction** for data visualization: the process of reducing the number of random variables under consideration by obtaining a set of principal variables.
- **Data denoising**: A denoising autoencoder (DAE) is one that receives a corrupted data point as input and is trained to predict the original, uncorrupted data point as its output. (ex. images and audio).
- **Feature estimation**: Autoencoders help to learn important hidden features in the process to reduce reconstruction error. During encoding, a new set of combinations of original features are generated.
- **Generation of synthetic data**: Autoencoders are capable of randomly generating new data that looks very similar to training data (e.g., Generative Adversarial Networks, or GANs).

Some specific Applications:

1. Anomaly detection
2. Image inpainting
3. Information retrieval
4. Unsupervised pre-training of deep neural networks

SureStart

BASIC AUTOENCODER ARCHITECTURES



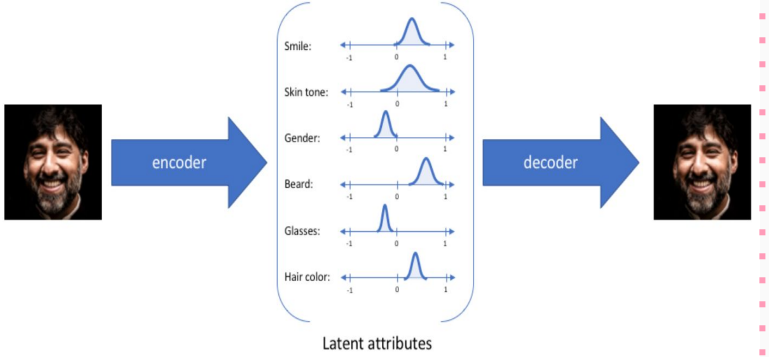
An autoencoder typically has the same architecture as a multi-layer perceptron, except that the number of neurons in the output layer must be equal to the number of inputs.

Undercomplete autoencoder: The hidden layer (i.e., latent space) has a lower dimension than the input layer.

- This construction limits the amount of information that can flow through the network and forces it to learn the most important attributes (also called **latent attributes**) of the input data.
- This prevents simple memorization of the input values.

There are many variations of autoencoders, such as: *1. Convolutional autoencoder, 2. Variational autoencoder (VAE), 3. Denoise autoencoder, 4. Sparse autoencoder, 5. Contractive autoencoder (CAE)*

COMMON TYPES OF AUTOENCODERS



Convolutional Autoencoder

In this type of autoencoder, the encoder and decoder layers are convolution layers;

- Specifically, it consists of a stack of Conv2D and MaxPooling2D layers, where the latter max pooling is being used for spatial down-sampling.
- The decoder consists of a stack of Conv2D and UpSampling2D layers.

Variational Autoencoder

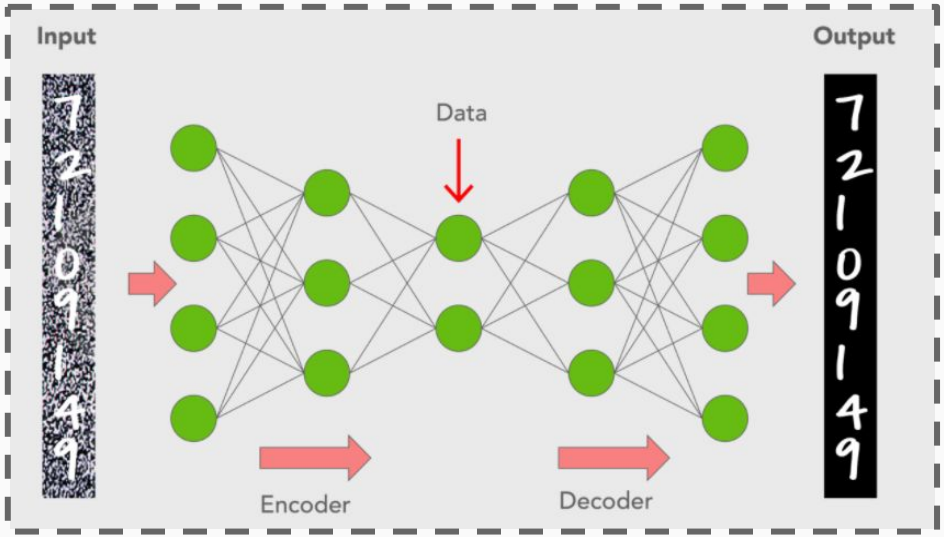
VAE is a "generative model" that can generate new images;

- The encoder constructs a probability distribution for each latent attribute.
- The latent attribute mapping is then fed into the decoder; enforcing a continuous, smooth latent space representation.
- From this the reconstructed 'lossy' output is generated.

SureStart

COMMON TYPES OF AUTOENCODERS

Denoising autoencoder: A feed forward neural network that learns to denoise images.
By doing so the neural network learns interesting features on the images used to train it. Then it can be used to extract features from images that are similar to the training set.



How?

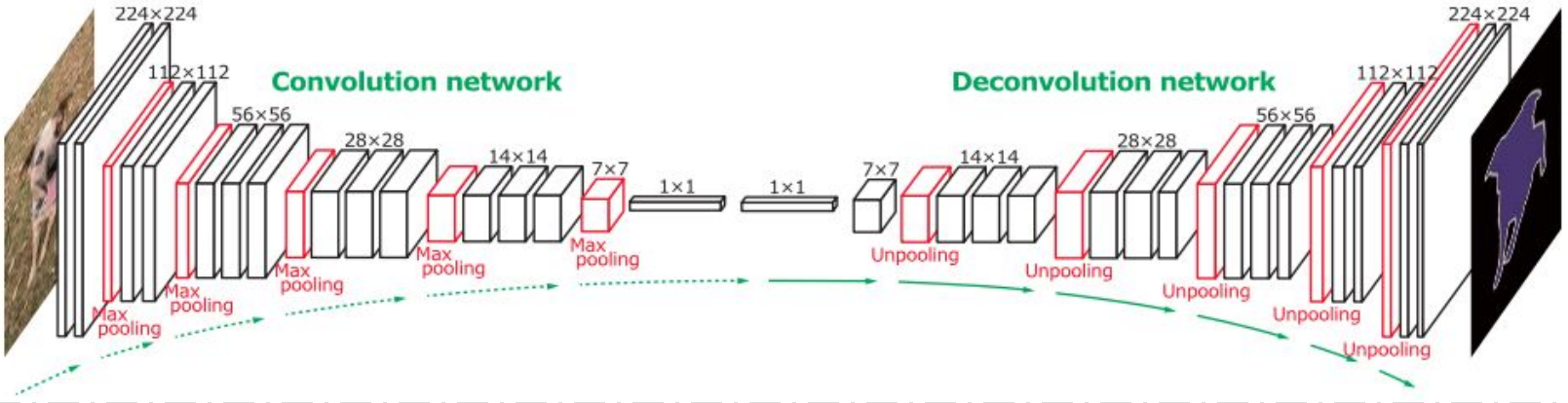
Denoising autoencoders add some noise to the input image, feed the corrupted data into autoencoder, measure reconstruction loss against the original image and recover the original undistorted input.

The model learns a vector field for mapping the input data towards a lower-dimensional latent space that describes the natural data to cancel out the added noise. The encoder thus extracts the most salient features and learns a more robust representation of the input data.

SureStart

UPSAMPLING AND DOWNSAMPLING

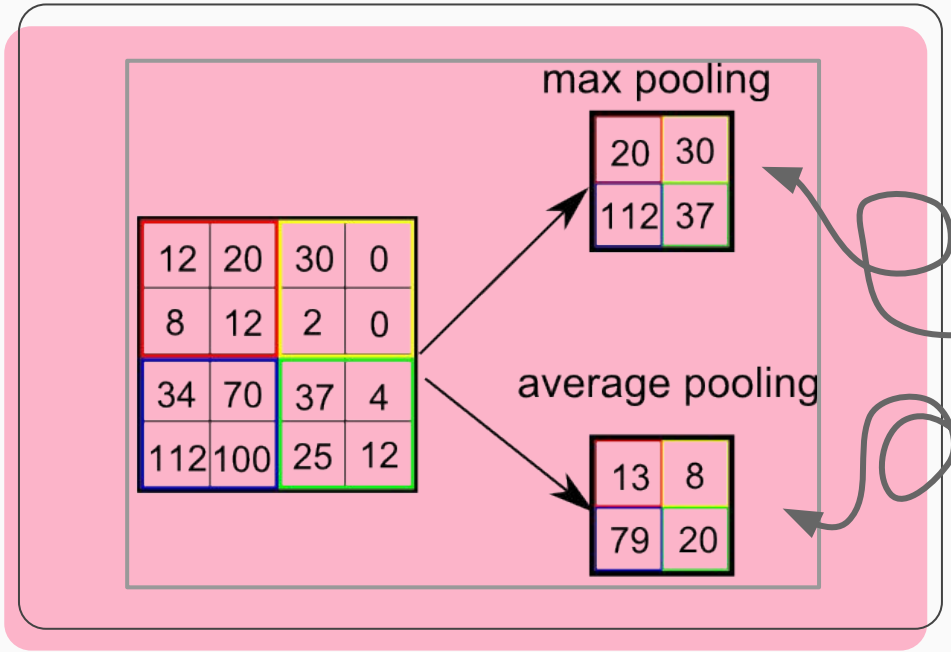
The shown CNN network performs downsampling in the encoder and upsampling in the decoder.



Downsampling is commonly implemented via max pooling operation with $n > 1$ stride.
Upsampling is commonly implemented via an unpooling operation.

SureStart

DOWNSAMPLING



Downsampling aims to reduces the spatial dimensions of the input (image).

In CNN, pooling layers execute downsampling by certain mathematical operations such as average pooling or max-pooling.

Downsampling helps to capture essential structural features of the represented images without getting hampered by the fine details.

Max Pooling: The largest element is taken from feature map.

Average Pooling: It calculates the average of the elements in a predefined size image section.

SureStart

UPSAMPLING TECHNIQUES

Upsampling process *upsamples* the compressed image; transform a small input into a large image output.

In Generative Adversarial Networks, or GANs, two common types of layers can be used in the upsampling technique:

1. **Transpose convolutional layer** (Conv2DTranspose) performs an inverse convolution operation.
   This layer have learnable parameters. It upsample input and learn how to fill in details during the model training process.

2. **Upsample layer** (UpSampling2D/ Unpooling) simply doubles the dimensions of the input.
   This layer doesn't have any weights to learn.

In Encoder-Decoder networks, unpooling layer is a commonly used as upsampling technique.
The most widely used techniques are:

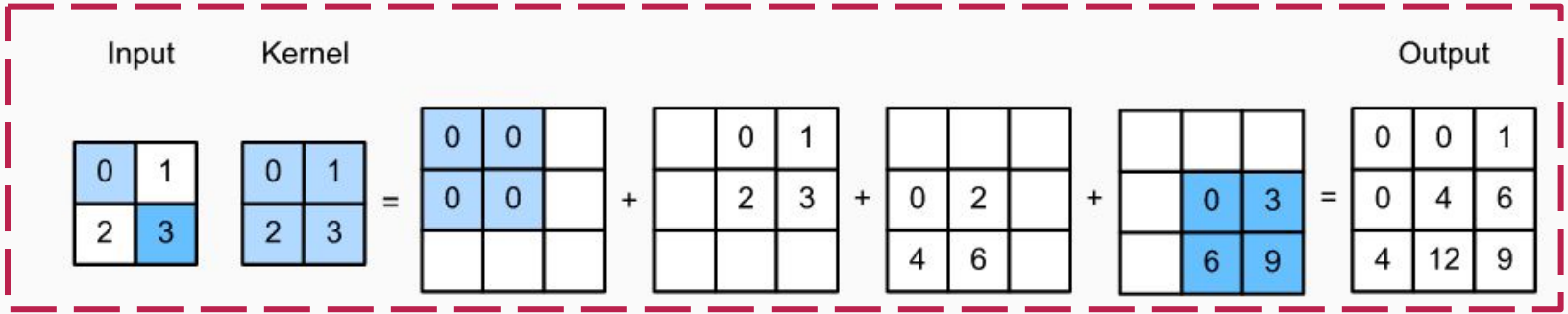
- Nearest Neighbor
- Bed of Nails
- Bi-Linear Interpolation
- Max-unpooling

SureStart

TRANSPOSE CONVOLUTIONAL LAYER

The basic operation that goes in a transposed convolution is explained below:

- Let's consider a 2 x 2 encoded feature map which needs to be upsampled to 3 x 3 feature map.
- The operation needs kernel of size 2 x 2 with unit stride and zero padding.
- Now the elements of the input feature map is multiplied with every element of the kernel as shown in the figure.
- This process is done for all the remaining elements of the input feature map as depicted in the figure.
- It can be noticed that some of the elements of the resulting upsampled feature maps are overlapping. To solve this issue, we simply add the elements of the overlapping positions.
- The resulting output will be the final upsampled feature map having the required spatial dimensions of 3 x 3.



SureStart

UNPOOLING TECHNIQUES

NEAREST NEIGHBOR

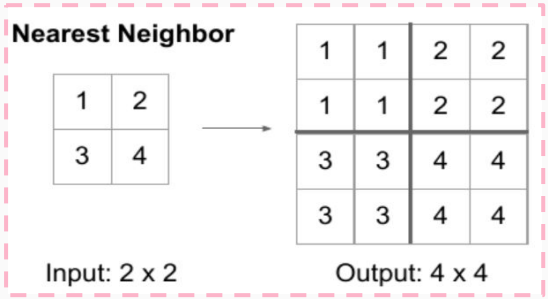
simply repeat every element.

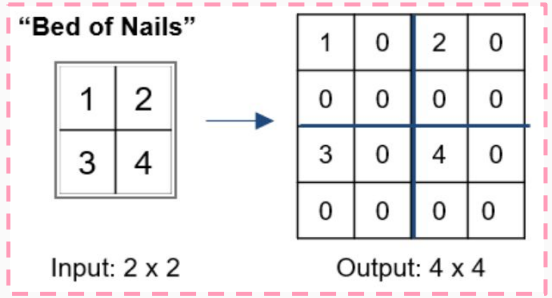
BED OF NAILS

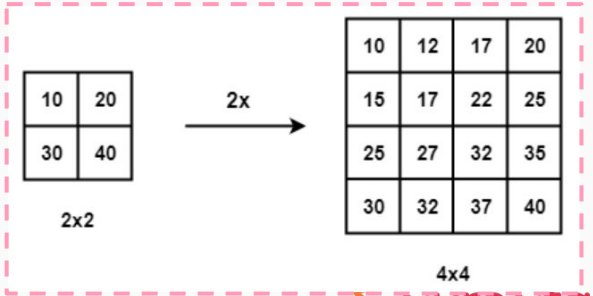
place the value in a particular position in the output, filling the rest with zeros. The below example places the input values in the upper left corner.

BI-LINEAR INTERPOLATION

take the 4 nearest pixel value of the input pixel and perform a weighted average based on the distance of the four nearest cells smoothing the output.



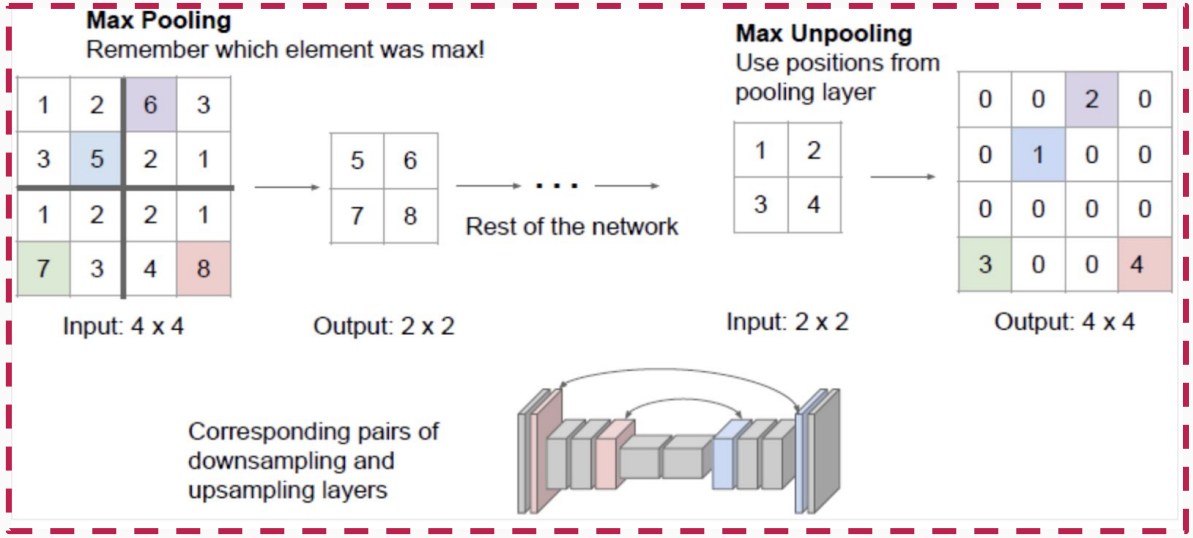




SureStart

UNPOOLING TECHNIQUES

The Max-Pooling layer in CNN takes the maximum among all the values in the kernel.



To perform max-unpooling -

- First, the index of the maximum value is saved for every max-pooling layer during the encoding step.
- The saved index is then used during the decoding step where the input pixel is mapped to the saved index, filling zeros everywhere else.

SureStart