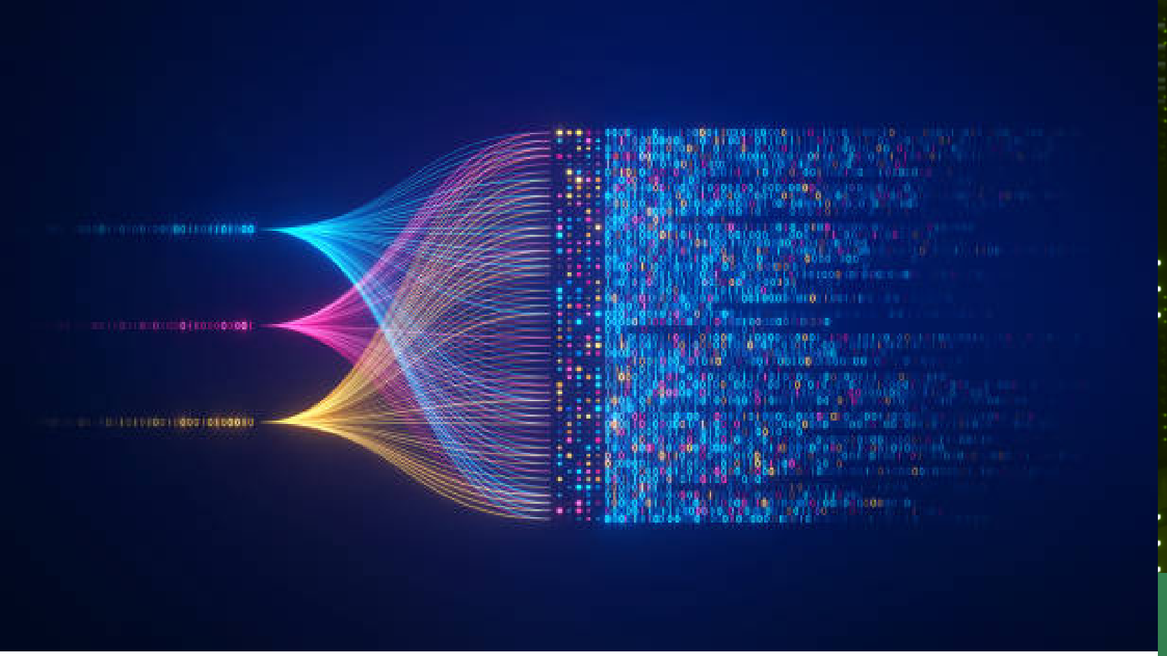
AUTOENCODER

neural network

DMYTRO BOROVYK

08.04.2024.

1. What is a neural network?
2. Autoencoders.
3. Denoising.



What is a neural network?

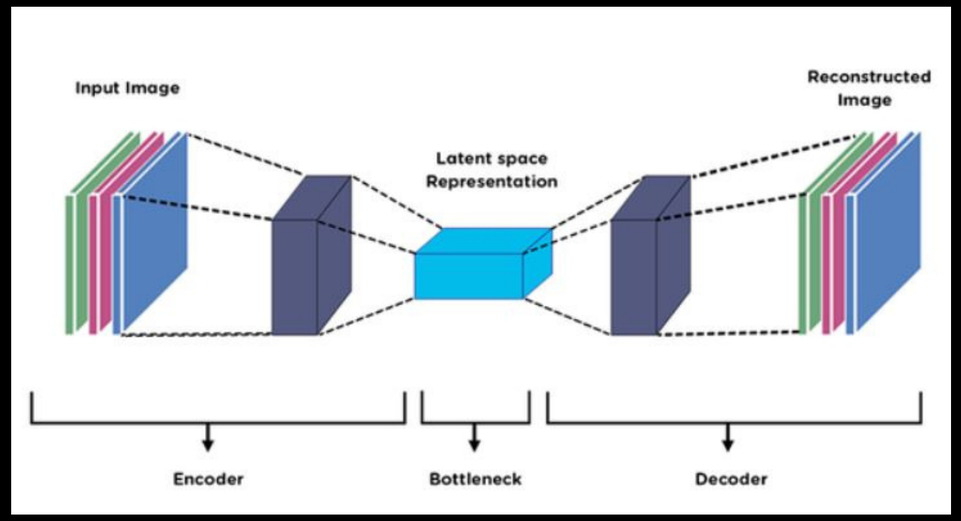
A neural network is a computational model inspired by the human brain, with interconnected neurons organized in layers. It processes input through activation functions and passes output to the next layer. Neural networks excel in tasks like pattern recognition and learning from large datasets, adjusting weights and biases during training to improve performance. They find applications in image recognition, natural language processing, and other fields due to their ability to handle complex relationships and adapt to new data, making them key tools in modern AI and machine learning.

Autoencoders

HOW THEY WORK?

Autoencoders are neural network architectures designed for unsupervised learning tasks. They aim to reconstruct the input data at the output, learning a compressed representation (encoding) of the input in the process. This compressed representation, called the latent space, captures essential features of the input data, useful for tasks like data denoising, dimensionality reduction, and anomaly detection.
Autoencoders are widely used in image, audio, and text data processing, where they can learn meaningful representations without requiring labeled data, making them valuable tools in various machine learning and deep learning applications.

Model autoencoder



Encoder

The encoder in an autoencoder transforms input into a compact representation, using layers with activation functions like ReLU or sigmoid. It reduces dimensionality to extract essential features and filter out noise

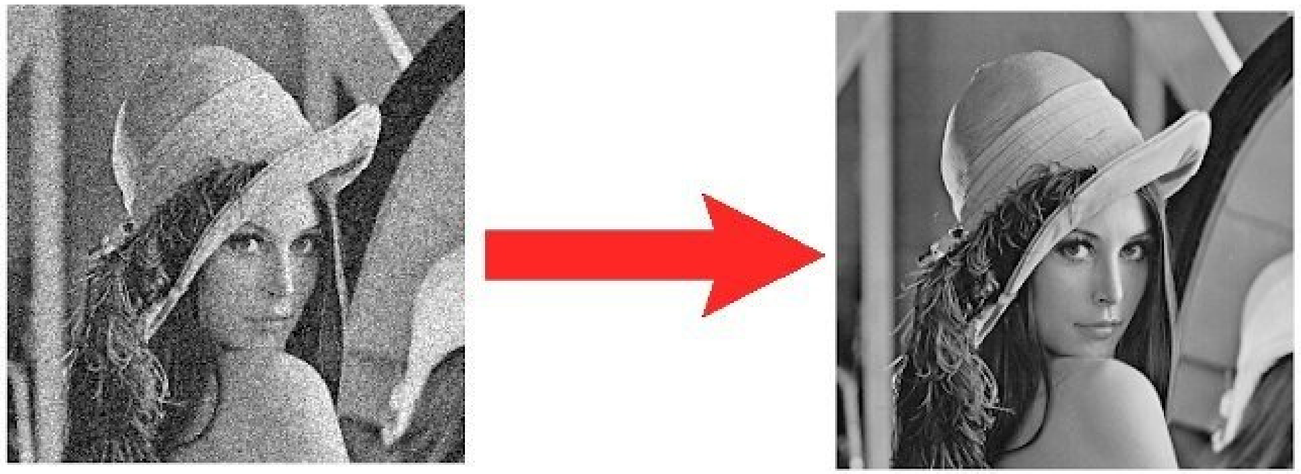
Bottleneck

The bottleneck, or latent space, is crucial in autoencoders, condensing input data while retaining key information. It compresses the encoder's output into a lower-dimensional space, enabling efficient storage and processing.

Decoder

The decoder in an autoencoder reconstructs input data from the compressed representation made by the encoder. It has multiple layers using activation functions like ReLU or sigmoid. These layers expand the latent representation to closely match the input data.

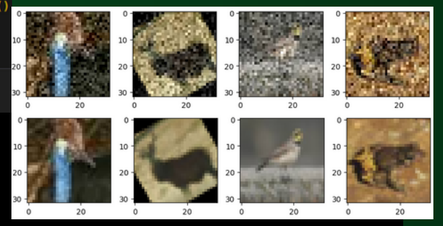
Denoising



Denoising is the task of removing noise from data to enhance its quality. In machine learning, denoising involves training models, like autoencoders, to distinguish between signal (information) and noise (unwanted variations). The model learns from pairs of noisy input and clean data during training, enabling it to remove noise from new data. This process improves data quality for analysis, applicable in image, audio, and general data preprocessing.

```python
# Load the dataset
(train_images, train_labels), (test_images, test_labels) = tf.keras.datasets.cifar10.load_data()
print('Sample dims: ', train_images.shape)
print('Test samples: ', test_images.shape)
```

```
Sample dims:  (50000, 32, 32, 3)
Test samples:  (10000, 32, 32, 3)
```



```python
#augmentation block
train_images_aug = []
train_labels_aug = []

augmentations = (
    iaa.Fliplr(1),
    iaa.Flipud(1),
    iaa.Affine(rotate=(-30))
)

#define amount of augmented images
num_augmented_samples = int(0.3 * len(train_images))
```

```
Sample dims:  (65000, 32, 32, 3)
Test samples:  (10000, 32, 32, 3)
```

Layer (type)	Output Shape	Param #
input_layer_29 (InputLayer)	(None, 32, 32, 3)	0
conv2d_87 (Conv2D)	(None, 32, 32, 8)	224
max_pooling2d_58 (MaxPooling2D)	(None, 16, 16, 8)	0
conv2d_88 (Conv2D)	(None, 16, 16, 16)	1,168
max_pooling2d_59 (MaxPooling2D)	(None, 8, 8, 16)	0
conv2d_transpose_58 (Conv2DTranspose)	(None, 16, 16, 16)	2,320
conv2d_transpose_59 (Conv2DTranspose)	(None, 32, 32, 16)	2,320
conv2d_89 (Conv2D)	(None, 32, 32, 3)	435

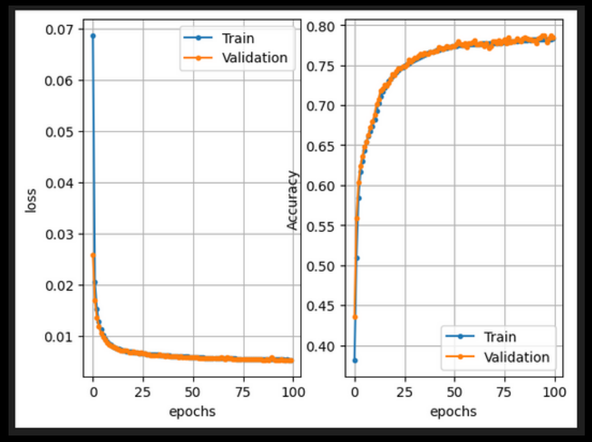
Total params: 6,467 (25.26 KB)

Trainable params: 6,467 (25.26 KB)

Non-trainable params: 0 (0.00 B)

```python
# Encoder
x = layers.Conv2D(8, (3, 3), activation="relu", padding="same")(input)
x = layers.MaxPooling2D((2, 2), padding="same")(x)
x = layers.Conv2D(16, (3, 3), activation="relu", padding="same")(x)
x = layers.MaxPooling2D((2, 2), padding="same")(x)

# Decoder
x = layers.Conv2DTranspose(16, (3, 3), strides=2, activation="relu", padding="same")(x)
x = layers.Conv2DTranspose(16, (3, 3), strides=2, activation="relu", padding="same")(x)
x = layers.Conv2D(3, (3, 3), activation="sigmoid", padding="same")(x)
```



In my project, I used the CIFAR-10 dataset initially comprising 60,000 images, which I divided into 50,000 for training and 10,000 for testing. Then, I augmented the dataset by adding 30% more images, resulting in a total of 65,000 images for training. Additionally, I applied blurring to all the images as part of the preprocessing steps.

The encoder section begins with a convolutional layer with 8 filters, each using a 3x3 kernel and ReLU activation, maintaining the input's spatial dimensions through "same" padding.
This is followed by a max-pooling layer with a 2x2 window and "same" padding, reducing spatial dimensions by half.
The decoder starts with a transposed convolutional layer with 16 filters, a 3x3 kernel, and ReLU activation, doubling the spatial dimensions through strides of 2 and "same" padding.
Another transposed convolutional layer with similar parameters further increases spatial dimensions.
The final layer uses a convolutional operation with 3 filters, a 3x3 kernel, sigmoid activation, and "same" padding to reconstruct the output, aiming for image-like quality.

The training process spanned 100 epochs, resulting in an accuracy exceeding 78%, which is quite decent for a model of this size with a modest number of Trainable params. Furthermore, there is no evidence of overfitting.

In this model, we're training an autoencoder on blurred images and comparing them to the original images using the Mean Squared Error (MSE) loss function. This process involves:

- Convolutional layers in the encoder to extract features and reduce dimensionality.
- Transposed convolutional layers in the decoder to upsample and reconstruct the images.
- Activation functions (ReLU for encoding, sigmoid for decoding) for non-linearity.
- The optimizer "Adam" to adjust model weights during training based on the loss.
- We set the batch size to 512 and train for 100 epochs, validating with a 20% split of the training data.

The objective is to train the model to identify features and restore images without blurring, leveraging the autoencoder's ability to learn a compressed representation of the input data.

Resuls:

Original images

Noisy images

Denoised images