Variational Encoder-Decoder Models for Prediction

# Encoder-Decoder Model

The encoder-decoder architecture is a fundamental framework in deep learning and has been widely used in various applications, including machine translation, image captioning, speech recognition, and more. It is particularly popular in sequence-to-sequence tasks where the input and output data are of variable lengths.

Here's a high-level overview of how the encoder-decoder model works:

1. Encoder:

- The encoder takes the input sequence (e.g., a sentence in natural language, an image, or a time series) and processes it into a fixed-length representation, often referred to as a "context vector" or "latent representation."

- It consists of one or more layers of neural networks (e.g., recurrent neural networks, long short-term memory networks, or transformers) that encode the input information.

- The encoder's goal is to capture the essential information from the input sequence and compress it into a meaningful context vector.

2. Decoder:

- The decoder takes the context vector produced by the encoder and generates an output sequence.

- Like the encoder, it also consists of one or more layers of neural networks, but it operates in a generative manner, producing one element of the output sequence at a time.

- During generation, the decoder can be autoregressive, where it conditions each output on the previously generated elements, or it can use other mechanisms depending on the specific task.

The training process for an encoder-decoder model often involves supervised learning, where the model is trained to minimize the difference between its generated output and the target output (ground truth) for a given input sequence. Various techniques, such as teacher forcing and attention mechanisms, are used to make the training more effective and improve the quality of generated sequences.

Encoder-decoder models have shown remarkable success in many natural language processing tasks, including machine translation (e.g., Google Translate), text summarization, and question-answering systems. They can also be applied to other domains, such as image-to-text generation and speech recognition.

# Encoder-Decoder For Prediction (Seq2Seq)

The encoder-decoder model can make predictions because it is designed to learn the mapping between an input sequence and an output sequence, and once trained, it can generate meaningful output sequences based on the input it receives. The key to its predictive capability lies in its training process and architecture:

1. Training Process: During training, an encoder-decoder model is exposed to a dataset containing pairs of input sequences and their corresponding target output sequences. It learns to minimize the difference (often measured using a loss function) between its predicted output and the true target output.

2. Learned Representations: The encoder component of the model learns to encode the essential information from the input sequence into a fixed-length context vector or representation. This representation captures the salient features and context of the input.

3. Decoder Generation: The decoder, conditioned on this context vector, learns to generate output sequences step by step. It uses its learned knowledge to predict the next element in the output sequence based on the context and previously generated elements (in the case of autoregressive models). This process continues until the entire output sequence is generated.

4. Generalization: Through the training process, the model generalizes from the training data, learning patterns, relationships, and dependencies between input and output sequences. It learns to make predictions not just for the specific examples it has seen during training but also for new, unseen input sequences.

5. Attention Mechanisms: Many encoder-decoder models use attention mechanisms, which allow the model to focus on different parts of the input sequence when generating each element of the output. This attention mechanism helps the model make more contextually relevant predictions.

6. Fine-Tuning and Inference: After training, the encoder-decoder model can be fine-tuned if necessary. During inference (prediction), you provide the model with an input sequence, and it uses its learned knowledge to generate an output sequence based on what it has learned during training.

In summary, the encoder-decoder model's predictive ability comes from its ability to learn meaningful representations from input sequences during training and then use those representations to generate output sequences that are consistent with what it has learned. This makes it a versatile framework for tasks like machine translation, text generation, image captioning, and more, where generating coherent and contextually relevant predictions is essential.

# Variational Autoencoder (VAE)

A Variational Autoencoder (VAE) is a generative probabilistic model and a type of autoencoder that is designed to learn a probabilistic mapping between input data and a latent space, allowing it to generate new data samples. VAEs are particularly useful for tasks like image generation, data compression, and generative modeling.

Here's an overview of the key components and concepts of a Variational Autoencoder:

1. Encoder:

- The encoder takes input data (e.g., images, text, or any other type of data) and maps it to a probability distribution in the latent space. Instead of directly encoding the input into a fixed vector as in a traditional autoencoder, the encoder produces two vectors: a mean vector (μ) and a variance vector (σ) that parameterizes a multivariate Gaussian distribution in the latent space.

2. Sampling:

- To generate samples from the latent space, the VAE samples from the Gaussian distribution are defined by the mean and variance vectors (μ and σ). This sampling step introduces stochasticity into the model.

3. Reparameterization Trick:

- To make the model differentiable, the VAE uses a reparameterization trick. Instead of directly sampling from the Gaussian distribution, it samples from a standard Gaussian (with mean 0 and variance 1) and then scales and shifts the samples using the mean and variance from the encoder. This allows for gradient-based optimization during training.

4. Latent Space:

- The latent space, often denoted as "z," is a lower-dimensional space where each point corresponds to a representation of the input data. It is this space where the VAE learns to encode the essential features of the data.

5. Decoder:

- The decoder takes a sample from the latent space and maps it back to the data space, attempting to reconstruct the input data. Like the encoder, the decoder is a neural network.

6. Objective Function:

- The training objective for a VAE combines two terms: a reconstruction loss and a regularization term. The reconstruction loss encourages the decoder to generate data similar to the input, while the regularization term encourages the latent space to follow a standard Gaussian distribution. This regularization term is known as the Kullback-Leibler (KL) divergence.

The main advantage of VAEs is their ability to generate new data samples by sampling from the learned latent space. Because the latent space follows a Gaussian distribution, you can generate data samples by sampling from this distribution and passing the samples through the decoder.

VAEs have been used for various applications, including image generation, image denoising, data compression, and more. They are part of the broader family of generative models and have contributed to advances in generative modeling and unsupervised learning in machine learning and deep learning.

## VAEs can be implemented in prediction tasks

Variational Autoencoders (VAEs) are primarily designed for generative tasks, such as generating new data samples from a learned latent space. While VAEs are not typically used for traditional prediction tasks like regression or classification, they can be adapted and combined with other models to incorporate predictive capabilities. Here are some ways VAEs can be used in prediction tasks:

1. Anomaly Detection:

- VAEs can be employed for anomaly detection in which the model learns to reconstruct normal data samples accurately. If a data point cannot be reconstructed well, it is considered an anomaly. This can be seen as a form of outlier prediction rather than traditional prediction tasks.

2. Imputation:

- In scenarios with missing data, VAEs can be used to impute missing values by generating plausible data points to fill in the gaps. This is useful for data preprocessing but doesn't involve predicting future or unseen data.

3. Time Series Prediction with Extensions:

- While VAEs alone are not well-suited for time series prediction tasks, variations of VAEs like Variational Recurrent Autoencoders (VRAEs) or Temporal Variational Autoencoders (TVAEs) have been proposed to handle time series data. These models can capture temporal dependencies and are used for tasks like time series forecasting or imputing missing values in time series data.

4. Combining VAEs with Predictive Models:

- VAEs can be combined with traditional predictive models like recurrent neural networks (RNNs) or convolutional neural networks (CNNs). The VAE can be used to learn a latent representation of the input data, and this representation can then be fed into another model for prediction. This hybrid approach can leverage the feature extraction capabilities of VAEs for predictive tasks.

It's essential to note that while VAEs can be part of a predictive system, their primary strength lies in generative modeling, representation learning, and data generation tasks. For many traditional prediction tasks, other models like feedforward neural networks, recurrent neural networks (RNNs), or convolutional neural networks (CNNs) are often more suitable due to their direct mapping from inputs to predictions.

# Variational Encoder-Decoder

An autoencoder is a special encoder-decoder model that is designed to reconstruct the input, i.e., the model’s input and the expected output are the same. VAE is an autoencoder using the reparameterization trick. We can introduce the reparameterization trick into normal encoder-decoder models so that they become variational encoder-decoders (VEDs).

## VED For Candlestick Chart Generation

We can design VEDs for candlestick chart generation, which is a sequence-to-sequence prediction task (Seq2Seq). Specifically, the model will take a candlestick chart (e.g., 20 trading days) as the input and output a candlestick chart of the next few trading days (e.g., the candlestick chart of the next 5 trading days.).

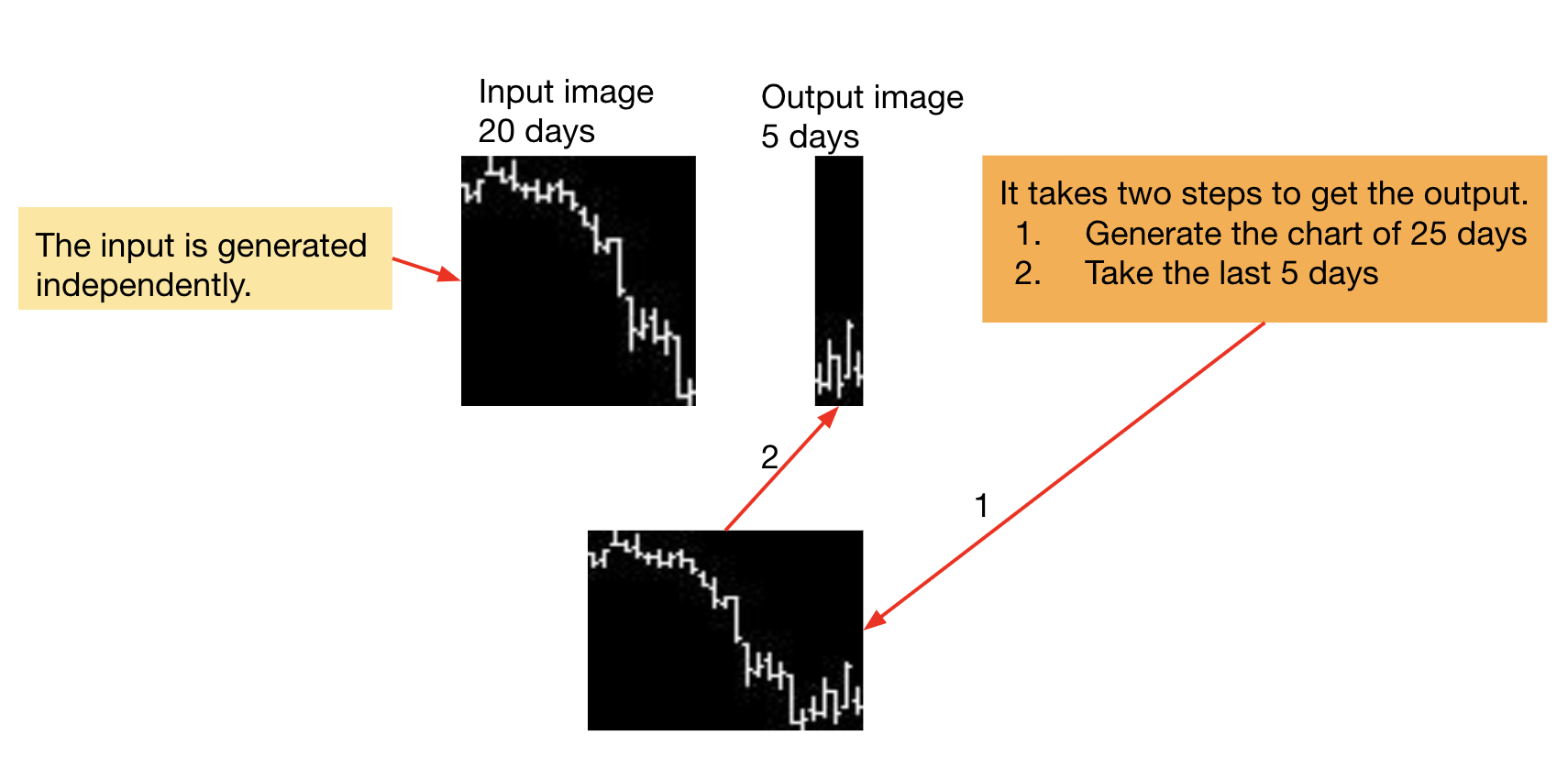
## The Shared Code (using PyTorch)

The shared program is the Python code of a VED model. It contains the following files,

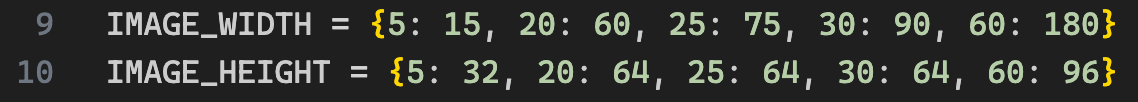
* train.py: it is for training the model and wraps all functions and variables related to model training;
* engine.py: it contains functions for training the model, such as loss functions and validation functions.
* model.py: it contains the definition of the variational encoder-decoder model.
* utilis.py: it contains the auxiliary functions such as image saving, plotting, etc.
* making\_chart.py: it is used to convert the stock prices into candle charts.
* run.SBASH: it is a bash file for running the training program on the HPC.
* make\_chart.SBASH: it is a bash file for running the making\_chart on the HPC.

## About the candlestick charts

Since it is a prediction task, it is important to have proper input charts and true output charts. The following figure shows how to generate the input and output images. The functions related to it are given in the making\_chart.py. The height of the input and output images are the same.



You may change the duration of the input and output images. When adjusting the durations, you need to check the dictionaries (IMAGE\_WIDTH and IMAGE\_HEIGHT) contain the keys of the durations you want to generate.



## Some potential improvements to the current VED model

1. Adding residual blocks into the model
2. Adjust the hyperparameters of the model, which include,
   1. The number of layers in the encoder and the decoder
   2. The kernel size of the CNN
   3. The dimensions of the context vector “Z”.