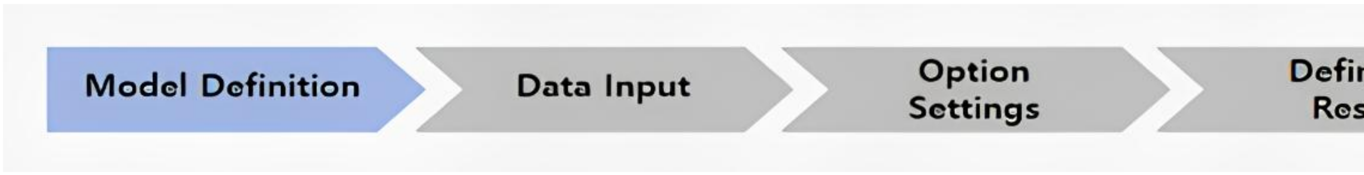
Chapter 5 AI

5.1 AI Wizard

5.1.1 User interface

ECMiner™ AI wizard provides a step-by-step interface for defining neural network models and configuring model options. The stream configuration is divided into four steps as follows.



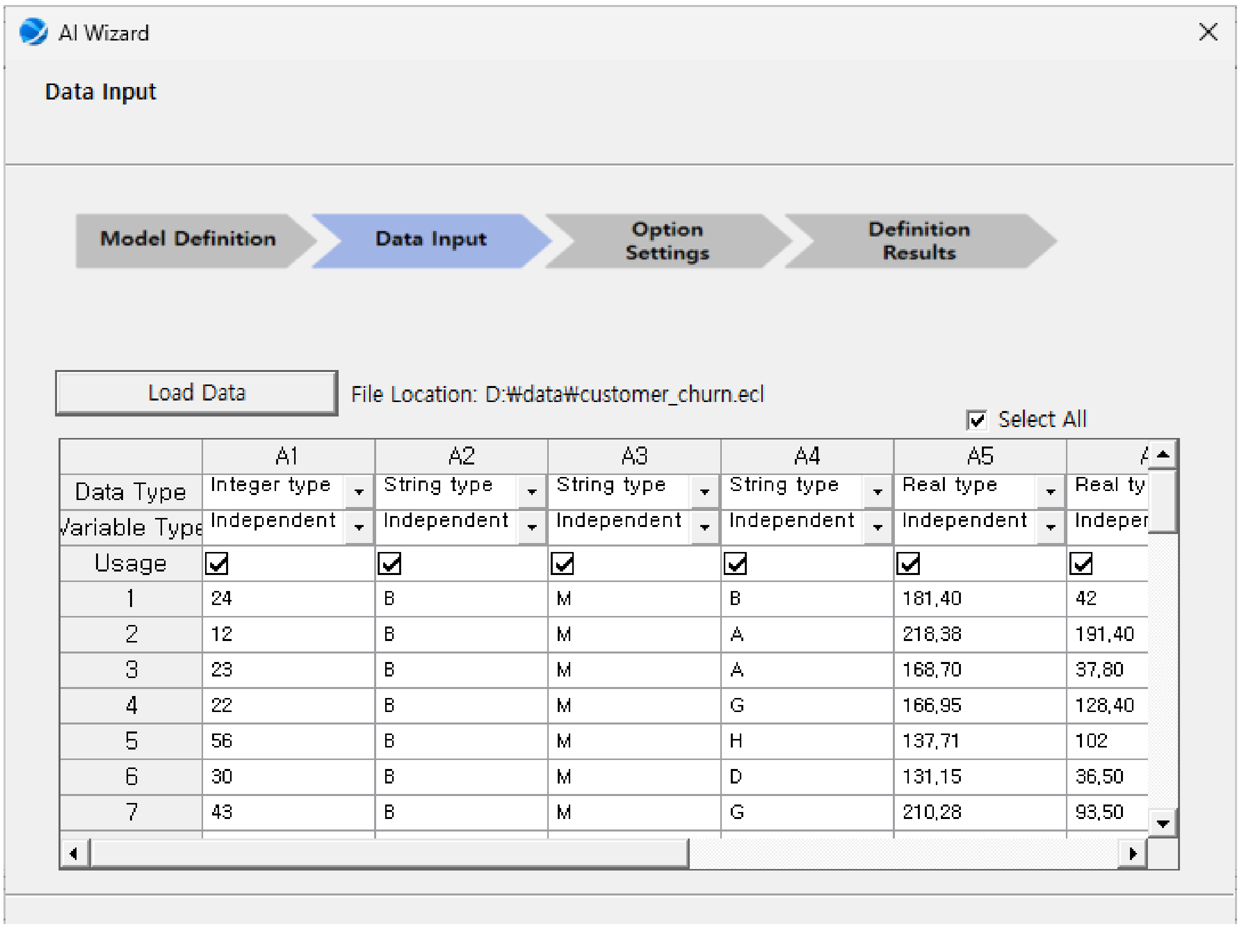
Step 1. Model Definition

Select the neural network model to train. There are four available models.

Category	Description	Supported Models	Applications
DNN	DNN is an artificial neural network with a multi-layer structure, capable of learning nonlinear patterns through multiple hidden layers. It uses deep layers to model complex relationships and is applied in various fields.	DNN, AutoEncoder	Regression analysis, Classification analysis, Feature extraction
CNN	CNN uses convolutional layers to detect spatial patterns, making it highly effective for image processing. It reduces the number of parameters by using local connections and shared weights. It is widely used in image classification, object detection, and other applications.	Lenet, AlexNet, VGGNet, ResNet, EfficientNet	Image analysis
RNN	RNN is a neural network designed for recurrent structure that remember previous information in sequential data. By repeatedly processing previous computation results along with the current input, it can model the flow of data over time.	SimpleRNN, LSTM, GRU	Sequential data prediction analysis
Language Model	Language Model (LM) is used for tasks like text generation, translation, and speech recognition. Model that learns from text data to predict the probability distribution of words or sentences. It is used to generate the next word or understand a sentence in a given context, and is applied in various natural language processing tasks such as translation, chatbots, and more.	Seq2Seq, Seq2Seq w/attention, Transformer	Translation, Text generation

Step 2. Data Input

Specify the data type, variable type, and whether each column should be used.



File Location: D:₩data₩customer_churn.ecl

	A1	A2	A3	A4	A5
Data Type	Integer type	String type	String type	String type	Real type	Real ty
Variable Type	Independent	Independent	Independent	Independent	Independent	Indeper
Usage	✓	✓	✓	✓	✓	✓
1	24	B	M	B	181,40	42
2	12	B	M	A	218,38	191,40
3	23	B	M	A	168,70	37,80
4	22	B	M	G	166,95	128,40
5	56	B	M	H	137,71	102
6	30	B	M	D	131,15	36,50
7	43	B	M	G	210,28	93,50

NOTE:

1) Keep in mind of the data formats according to the chosen neural network models.

2) Preview is available for up to 50 items.

Step 3. Option Settings

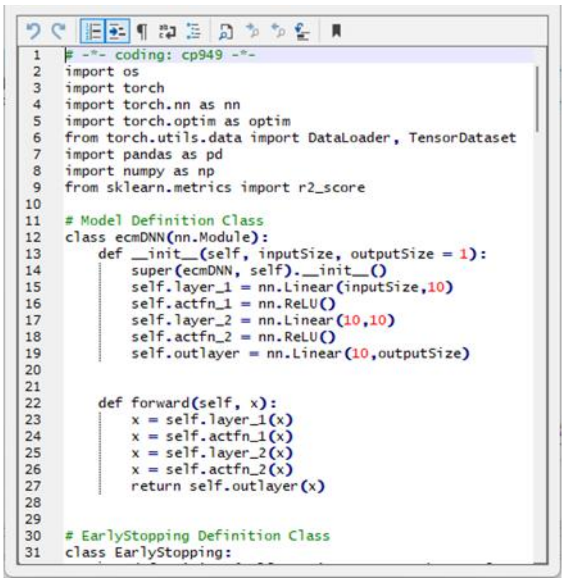
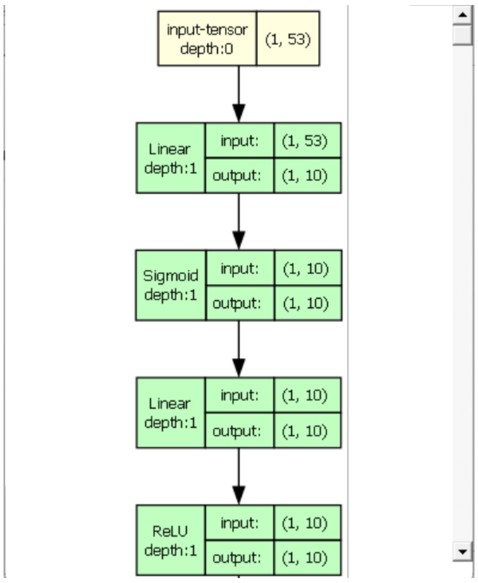
Define the structure of the neural network. Options vary according to the chosen model. The following options are general settings for all models.

Options group	Options Name	Description	Note
Network Option	Number of Hidden Layers	Set the number of hidden layers in a neural network. The deeper the neural network, the more sophisticated the model can be produced, but the higher the risk of overfitting.	1~10
	Layer #1~n Number of Nodes	Set the number of nodes in each hidden layer.	integer
	Activation Function	Set the activation function. Activation function is a mathematical transformation applied to its input.	Linear, Sigmoid, Tanh, ReLU, LeakyReLU, ELU
Train Option	Epochs	Set the number of iterations.	integer
	Batch Size	Set the number of data samples to be used in a single training step.	integer
	Learning Rate	Set the percentage to update the weights at each learning stage. Excessively large values can disrupt learning, while excessively small values can slow it down.	0<real number≤1
	Target Loss	Specify the target loss; stops learning when the value of the loss function in the model reaches the target loss.	0<real number≤1

	Optimization Method	Set up an algorithm to update the weight of the model.	SGD, RMSprop, Adagrad, Adam
	Scaling Method	Set how to standardize your input data. Improve learning stability and performance.	Min-Max Scaler (0,1), Min-Max Scaler (-1,1), Standard Scaler, None

Step 4. Definition Results

Review of the options set by the user is available, along with the neural network python code and a diagram of the neural network structure.

Neural network python code	Neural network structure
	

```python
# -*- coding: cp949 -*-
import os
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import DataLoader, TensorDataset
import pandas as pd
import numpy as np
from sklearn.metrics import r2_score

# Model Definition Class
class ecmDNN(nn.Module):
    def __init__(self, inputSize, outputSize = 1):
        super(ecmDNN, self).__init__()
        self.layer_1 = nn.Linear(inputSize,10)
        self.actfn_1 = nn.ReLU()
        self.layer_2 = nn.Linear(10,10)
        self.actfn_2 = nn.ReLU()
        self.outlayer = nn.Linear(10,outputSize)


    def forward(self, x):
        x = self.layer_1(x)
        x = self.actfn_1(x)
        x = self.layer_2(x)
        x = self.actfn_2(x)
        return self.outlayer(x)


# EarlyStopping Definition Class
class EarlyStopping:
```

5.1.2 DNN

An artificial neural network with a multi-layer structure, capable of learning nonlinear patterns through multiple hidden layers. It uses deep layers to model complex relationships and is applied in various fields.

5.1.2.1 DNN

Overview

DNN (Deep Neural Networks) is an extension of traditional neural networks, such as Multi-Layer Perceptron (MLP), incorporating multiple hidden layers to learn complex patterns. This neural network can learn complex patterns from different types of data by using advanced methods. DNNs are used for tasks such as classification, prediction, and pattern recognition.

Components

- **Input Layer**

  This layer is responsible for receiving the input data into the neural network.

- **Hidden Layers**

  These layers process the data, extracting complex features and patterns through multiple stages.

- **Output Layer**

  The final layer produces the prediction or classification results based on the learned features.

Working Principle

The network learns by adjusting the weights and activation functions of connections between nodes based on the data it is trained on, making it highly effective at handling complex and high-dimensional data.

Options

Options group	Options Name	Description	Note
Network Option	Number of Hidden Layers	Set the number of hidden layers. The deeper the neural network, the more sophisticated the model can be produced, but the higher the risk of overfitting.	1~10
	Layer #1~n Number of Nodes	Set the number of nodes in each hidden layer.	integer
	Activation Function	Set the activation function. Activation function is a mathematical transformation applied to its input.	Linear, Sigmoid, Tanh, ReLU, LeakyReLU, ELU
Train Option	Epochs	Set the number of iterations.	integer
	Batch Size	Set the number of data samples to be used in a single training step.	ingeter
	Learning Rate	Set the percentage to update the weights at each learning stage. Excessively large values can disrupt learning, while excessively small values can slow it down.	0<real number≤1
	Target Loss	Specify the target loss; stops learning when the value of the loss function in the model reaches the target loss.	0<real number≤1
	Optimization Method	Set up an algorithm to update the weight of the model.	SGD, RMSprop, Adagrad, Adam
	Scaling Method	Set how to standardize your input data. Improve learning stability and performance.	Min-Max Scaler (0,1),

			Min-Max Scaler (-1,1), Standard Scaler, None

5.1.2.2 AutoEncoder

Overview

An Autoencoder is a type of neural network consisting of an encoder that compresses the input data and a decoder that reconstructs the original data from the compressed representation. This model is mainly used to learn the key features of the data and for dimensionality reduction.

Components

- **Encoder**

  Converts input data into a lower-dimensional representation.

- **Decoder**

  Expands the lower-dimensional representation back to the original dimensions to reconstruct the data.

- **Loss Function**

  Measures the difference between the original data and the reconstructed data, guiding the learning process.

Working Principle

The Autoencoder learns by extracting important features from the data using the encoder, then reconstructing the original data using the decoder. This process, which falls under unsupervised learning, is useful for tasks such as noise removal and dimensionality reduction. By focusing on the most relevant features, Autoencoders can efficiently compress and reconstruct data.

Options

Options group	**Options Name**	**Description**	**Note**
Network Option	Number of Hidden Layers	Set the number of hidden layers. The deeper the neural network, the more sophisticated the model can be produced, but the higher the risk of overfitting.	1~10
	Feature Hidden Layer #1~n: Number of Nodes	Sets the number of nodes in Feature hidden layer. more nodes can produce more sophisticated models, but also higher computational costs.	integer
	Feature Hidden Layer Activation Function	Sets the activation function of Feature hidden layer. The activation function adds nonlinearity to the model, allowing it to learn more complex patterns.	Linear, Sigmoid, Tanh, ReLU, LeakyReLU, ELU
	Symmetric Hidden Layer #1~n: Number of Nodes	Sets the number of nodes in the symmetric hidden layers. The number of nodes should gradually decrease from the first hidden layer to the feature hidden layer.	integer
	Symmetric Hidden Layer #1~n: Activation Function	Sets the activation function for the symmetric hidden layers. The activation function adds nonlinearity to the model, allowing it to learn more complex patterns.	Linear, Sigmoid, Tanh, ReLU, LeakyReLU, ELU
Train Option	Epochs	Set the number of iterations.	integer
	Batch Size	Set the number of data samples to be used in a single training step.	ingeter
	Learning Rate	Set the percentage to update the weights at each learning stage. Excessively large values	0<real number≤1

		can disrupt learning, while excessively small values can slow it down.	
	Target Loss	Specify the target loss; stops learning when the value of the loss function in the model reaches the target loss.	0<real number≤1
	Optimization Method	Set up an algorithm to update the weight of the model.	SGD, RMSprop, Adagrad, Adam
	Scaling Method	Set how to standardize your input data. Improve learning stability and performance.	Min-Max Scaler (0,1), Min-Max Scaler (-1,1), Standard Scaler, None

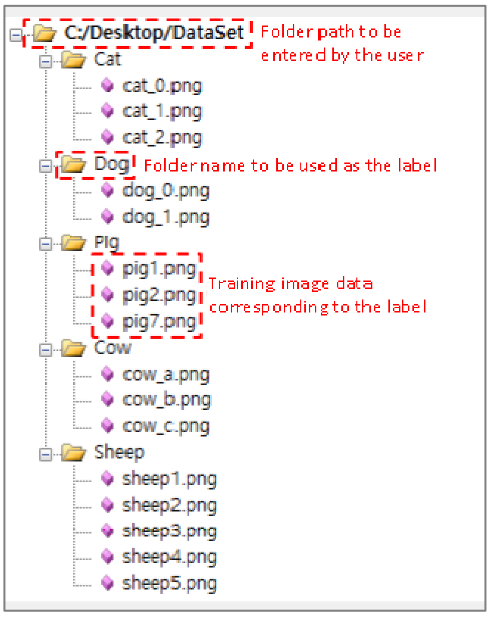
5.1.3 CNN

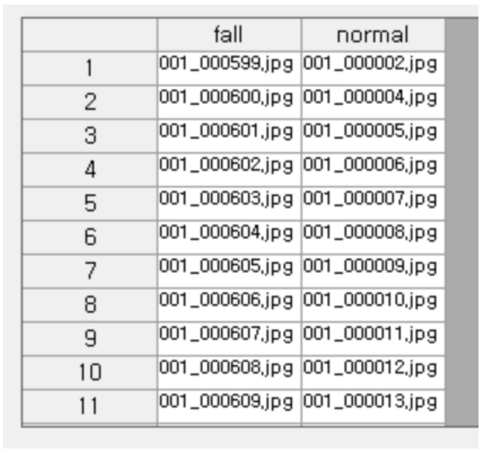
Neural network specialized in processing high-dimensional data such as images. By utilizing convolution and pooling operations to extract features, it effectively recognizes patterns while preserving spatial information. It is widely used in image classification, object detection, and other applications.

Data Input Format

The folder names will be designated as training labels, and the training image data corresponding to each label must be stored within the respective folders.

The folder structure for training data should be as follows:





	fall	normal
1	001_000599.jpg	001_000002.jpg
2	001_000600.jpg	001_000004.jpg
3	001_000601.jpg	001_000005.jpg
4	001_000602.jpg	001_000006.jpg
5	001_000603.jpg	001_000007.jpg
6	001_000604.jpg	001_000008.jpg
7	001_000605.jpg	001_000009.jpg
8	001_000606.jpg	001_000010.jpg
9	001_000607.jpg	001_000011.jpg
10	001_000608.jpg	001_000012.jpg
11	001_000609.jpg	001_000013.jpg

5.1.3.1 LeNet

Overview

LeNet is one of the early convolutional neural networks, and was originally designed for handwritten digit recognition, specifically for the MNIST dataset. Though its structure is simple, it introduced the fundamental ideas and structural characteristics of modern CNNs.

Components

- **Convolutional Layers**

  These layers extract low-level features from the input image.

- **Average Pooling Layers**

  These layers reduce the size of feature maps and improve robustness by downsampling the data.

- **Fully Connected Layers**

  Based on the extracted features, these layers make the final classification decision.

Working Principle

LeNet extracts features through convolution and pooling layers, and then uses the fully connected layers to predict the final class.

Options

The architecture of the LeNet model is provided in a fixed form.

Options group	Options Name	Description	Note
Network Option	Number of input channels	The number of color channels in an image; a grayscale image has 1 channel, while an RGB image has 3 channels.	1
	Width	The width of the input data (image) in pixels.	32
	Height	The height of the input data (image) in pixels.	32
	Convolutional Layers	A layer that extracts features from the input data. The first layer receives input with a single channel and generates 6 feature maps, while the second layer takes these as input and produces 16 feature maps.	2
	Fully Connected layers	A fully connected layer in the neural network, which performs the final prediction based on the given features.	3
Train Option	Epochs	Set the number of iterations.	integer
	Batch Size	Set the number of data samples to be used in a single training step.	ingeter
	Learning Rate	Set the percentage to update the weights at each learning stage. Excessively large values can disrupt learning, while excessively small values can slow it down.	0<real number≤1

	Target Loss	Specify the target loss; stops learning when the value of the loss function in the model reaches the target loss.	0<real number≤1
	Optimization Method	Set up an algorithm to update the weight of the model.	SGD, RMSprop, Adagrad, Adam
	Scaling Method	Scaling Method is set to Min-Max Scaler (0,1).	Min-Max Scaler (0,1)

5.1.3.2 AlexNet

Overview

AlexNet is an expanded and deeper version of LeNet, which gained worldwide recognition when it won the 2012 ImageNet competition. This model was one of the first to apply modern deep learning techniques and demonstrated the power of deep learning with GPUs and large datasets for image classification tasks.

Components

- **Convolutional Layers**

  These layers extract high-level features from the images.

- **ReLU Activation Function**

  It helps solve non-linear problems and accelerates training by enabling faster convergence.

- **Max Pooling Layers**

  These layers reduce the size of the data while preserving important features.

- **Dropout**

  This technique helps prevent overfitting during training by randomly disabling a portion of the neurons.

Working Principle

AlexNet extracts complex image features through deep convolutional layers and reduces data size using max pooling layers. The ReLU activation function accelerates training by enabling faster convergence. Dropout is also used to prevent overfitting during training. Afterward, fully connected layers process the high-level features for classification.

Options

The architecture of the AlexNet model is provided in a fixed form.

Options group	Options Name	Description	Note
Network Option	Number of input channels	The number of color channels in an image; a grayscale image has 1 channel, while an RGB image has 3 channels.	3
	Width	The width of the input data (image) in pixels.	227
	Height	The height of the input data (image) in pixels.	227
	Convolutional Layers	A layer that extracts features from the input data, generating feature maps based on the output of the previous layer. The 5 layers sequentially produce feature maps with 64, 128, 256, and 512 channels.	5
	Fully Connected layers	A fully connected layer in the neural network, which performs the final prediction based on the given features.	3
Train Option	Epochs	Set the number of iterations.	integer
	Batch Size	Set the number of data samples to be used in a single training step.	ingeter
	Learning Rate	Set the percentage to update the weights at each learning stage. Excessively large values can disrupt learning, while excessively small values can slow it down.	0<real number≤1

	Target Loss	Specify the target loss; stops learning when the value of the loss function in the model reaches the target loss.	0<real number≤1
	Optimization Method	Set up an algorithm to update the weight of the model.	SGD, RMSprop, Adagrad, Adam
	Scaling Method	Scaling Method is set to Min-Max Scaler (0,1).	Min-Max Scaler (0,1)

5.1.3.3 VGGNet

Overview

VGGNet is known for its simple architecture, where convolutional and pooling layers are stacked uniformly. This model demonstrated that increasing the depth of the network with smaller convolutional filters (3x3) can improve performance. It is especially popular for transfer learning tasks.

Components

- **Convolutional Layers**

  These layers use kernels of the same size to extract features from the input data.

- **Max Pooling Layers**

  These layers simplify the feature maps and enhance computational efficiency.

Working Principle

VGGNet is built by stacking several layers of 3x3 convolutions and 2x2 max pooling. The idea behind using small filters is that multiple small filters (3x3) can capture more complex features while reducing computational complexity.

Options

The architecture of the VGGNet model is provided in a fixed form.

Options group	Options Name	Description	Note
Network Option	Number of input channels	The number of color channels in an image; a grayscale image has 1 channel, while an RGB image has 3 channels.	3
	Width	The width of the input data (image) in pixels.	224
	Height	The height of the input data (image) in pixels.	224
	Convolutional Layers	A layer that extracts features from the input data, generating feature maps based on the output of the previous layer. The 13 layers sequentially produce feature maps with 64, 128, 256, and 512 channels.	13
	Fully Connected layers	A fully connected layer in the neural network, which performs the final prediction based on the given features.	3
Train Option	Epochs	Set the number of iterations.	integer
	Batch Size	Set the number of data samples to be used in a single training step.	integer
	Learning Rate	Set the percentage to update the weights at each learning stage. Excessively large values can disrupt learning, while excessively small values can slow it down.	0<real number≤1
	Target Loss	Specify the target loss; stops learning when the value of the loss function in the model reaches the target loss.	0<real number≤1
	Optimization Method	Set up an algorithm to update the weight of the model.	SGD, RMSprop, Adagrad, Adam

	Scaling Method	Scaling Method is set to Min-Max Scaler (0,1).	Min-Max Scaler (0,1)

5.1.3.4 ResNet

Overview

ResNet features residual connections that were introduced to solve the vanishing gradient problem in deep neural networks. This architecture allows the network to improve performance as the layers get deeper, enabling it to learn more complex patterns without losing effectiveness.

Components

- **Residual Blocks**

  Each block contains a set of convolutional layers, but with the addition of skip connections that allow the input to bypass certain layers and be added directly to the output. This helps address the problem of vanishing gradients in very deep networks.

Working Principle

ResNet uses the skip connection technique which passes the input through multiple residual blocks and adds the original input back to the output. This allows the model to learn residual functions (the difference between the desired output and the input), rather than the entire output directly. This approach enables the training of deep networks without suffering from the vanishing gradient problem.

Options

The architecture of the ResNet model is provided in a fixed form.

Options group	Options Name	Description	Note
Network Option	Number of input channels	The number of color channels in an image; a grayscale image has 1 channel, while an RGB image has 3 channels.	3

	Width	The width of the input data (image) in pixels.	224
	Height	The height of the input data (image) in pixels.	224
	Convolutional Layers	A layer that extracts features from the input data, generating feature maps based on the output of the previous layer. The 20 layers sequentially produce feature maps with 64, 128, 256, and 512 channels.	20
	Fully Connected layers	A fully connected layer in the neural network, which performs the final prediction based on the given features.	1
Train Option	Epochs	Set the number of iterations.	integer
	Batch Size	Set the number of data samples to be used in a single training step.	integer
	Learning Rate	Set the percentage to update the weights at each learning stage. Excessively large values can disrupt learning, while excessively small values can slow it down.	0<real number≤1
	Target Loss	Specify the target loss; stops learning when the value of the loss function in the model reaches the target loss.	0<real number≤1
	Optimization Method	Set up an algorithm to update the weight of the model.	SGD, RMSprop, Adagrad, Adam
	Scaling Method	Scaling Method is set to Min-Max Scaler (0,1).	Min-Max Scaler (0,1)

5.1.3.5 EfficientNet

Overview

EfficientNet introduces a method for balancing the scaling of the network's depth, width, and resolution to achieve higher performance with fewer parameters compared to traditional models.

Components

- **Compound Scaling**

  Uniformly scales depth, width, and resolution using a fixed set of scaling coefficients, optimizing the model’s efficiency.

Working Principle

EfficientNet works by applying the compound scaling method to find the optimal balance between depth, width, and resolution. Instead of manually tuning each of these hyperparameters separately, EfficientNet scales them in a way that maximizes performance while minimizing computational costs.

Options

The architecture of the EfficientNet model is provided in a fixed form.

Options group	Options Name	Description	Note
Network Option	Number of input channels	The number of color channels in an image; a grayscale image has 1 channel, while an RGB image has 3 channels.	3
	Width	The width of the input data (image) in pixels.	240
	Height	The height of the input data (image) in pixels.	240
	Convolutional Layers	A layer that extracts features from the input data, generating feature maps based on the output of the previous layer. The 50 layers sequentially produce feature maps with 64, 128, 256, and 512 channels.	50

	Fully Connected layers	A fully connected layer in the neural network, which performs the final prediction based on the given features.	1
Train Option	Epochs	Set the number of iterations.	integer
	Batch Size	Set the number of data samples to be used in a single training step.	integer
	Learning Rate	Set the percentage to update the weights at each learning stage. Excessively large values can disrupt learning, while excessively small values can slow it down.	0<real number≤1
	Target Loss	Specify the target loss; stops learning when the value of the loss function in the model reaches the target loss.	0<real number≤1
	Optimization Method	Set up an algorithm to update the weight of the model.	SGD, RMSprop, Adagrad, Adam
	Scaling Method	Scaling Method is set to Min-Max Scaler (0,1).	Min-Max Scaler (0,1)

5.1.4 RNN

Neural network constructed with a recurrent structure that can remember previous information in sequential data. By repeatedly processing previous computation results along with the current input, it can model the flow of data over time.

5.1.4.1 SimpleRNN

Overview

RNN (Recurrent Neural Network) is a type of neural network specialized for processing sequence data. This structure maintains the hidden states that captures information about previous inputs, allowing it to model the flow of data over time. It is widely used in fields such as natural language processing and time series analysis.

Components

- **Recurrent Node**

  At each sequence step, it takes the input and the previous step's state, and outputs the current state.

- **State Vector**

  This stores and transfers the current and past information of the sequence.

- **Output Layer**

  The final result is generated from the output of the recurrent node.

Working Principle

At each time step, the input and the previous state combine to pass information through the network, modeling the temporal dependencies of the sequence. This allows the network to capture the dynamics of time-series data or sequential information.

Options

Options group	Options Name	Description	Note
Network Option	Number of Hidden Layers	Sets the number of hidden layers inside the RNN cell. The deeper the neural network, the more sophisticated the model can be produced, but the higher the risk of overfitting.	1~10
	Number of Nodes	Sets the number of nodes in the hidden state within the RNN cell. The larger the number of	integer

		nodes, the more sophisticated the model can be created, but the computational cost also increases.	
	Activation Function	Set the activation function. Activation function is a mathematical transformation applied to its input.	Linear, Sigmoid, Tanh, ReLU, LeakyReLU, ELU
	Drop-Out	Sets a random percentage of the nodes to be deactivated at each learning stage. When used properly, the model can learn more generalized features and prevent overfitting.	0<real number≤1
Train Option	Epochs	Set the number of iterations.	integer
	Batch Size	Set the number of data samples to be used in a single training step.	integer
	Learning Rate	Set the percentage to update the weights at each learning stage. Excessively large values can disrupt learning, while excessively small values can slow it down.	0<real number≤1
	Target Loss	Specify the target loss; stops learning when the value of the loss function in the model reaches the target loss.	0<real number≤1
	Optimization Method	Set up an algorithm to update the weight of the model.	SGD, RMSprop, Adagrad, Adam
	Scaling Method	Set how to standardize your input data. Improve learning stability and performance.	Min-Max Scaler (0,1), Min-Max Scaler (-1,1),

			Standard Scaler, None

5.1.4.2 LSTM

Overview

LSTM (Long Short-Term Memory) is a type of RNN designed to solve the long-term dependency problem. Compared to basic RNNs, LSTM has a more complex structure and excels at retaining information over long periods.

Components

- **Input Gate**

  Controls the amount of new information that is added to the cell state from the current input.

- **Forget Gate**

  Determines how much of the previous state information should be forgotten.

- **Output Gate**

  Controls how much of the current state should be output to the next layer.

- **Cell State**

  The central component that stores past information and updates it over time.

Working Principle

LSTM selectively removes unnecessary information and retains the important information through the operation of each gate. This allows to capture long-term dependencies and retrain information over long sequences, overcoming the vanishing gradient problem.

Options

Options group	Options Name	Description	Note

Network Option	Number of Hidden Layers	Sets the number of hidden layers inside the RNN cell. The deeper the neural network, the more sophisticated the model can be produced, but the higher the risk of overfitting.	1~10
	Number of Nodes	Sets the number of nodes in the hidden state within the RNN cell. The larger the number of nodes, the more sophisticated the model can be created, but the computational cost also increases.	integer
	Activation Function	Set the activation function. Activation function is a mathematical transformation applied to its input.	Linear, Sigmoid, Tanh, ReLU, LeakyReLU, ELU
	Drop-Out	Sets a random percentage of the nodes to be deactivated at each learning stage. When used properly, the model can learn more generalized features and prevent overfitting.	0<real number≤1
Train Option	Epochs	Set the number of iterations.	integer
	Batch Size	Set the number of data samples to be used in a single training step.	integer
	Learning Rate	Set the percentage to update the weights at each learning stage. Excessively large values can disrupt learning, while excessively small values can slow it down.	0<real number≤1
	Target Loss	Specify the target loss; stops learning when the value of the loss function in the model reaches the target loss.	0<real number≤1

	Optimization Method	Set up an algorithm to update the weight of the model.	SGD, RMSprop, Adagrad, Adam
	Scaling Method	Set how to standardize your input data. Improve learning stability and performance.	Min-Max Scaler (0,1), Min-Max Scaler (-1,1), Standard Scaler, None

5.4.1.3 GRU

Overview

GRU (Gated Recurrent Unit) is a type of recurrent neural network with a simplified structure, making it computationally efficient. It can maintain information over long periods while processing complex sequence data, making it widely used in tasks like natural language processing and time series analysis.

Components

- **Update Gate**

  Determines how much of the previous state's information should be retained.

- **Reset Gate**

  Decides how much of the previous information should be forgotten.

- **Recurrent Node**

  Combines the input with the previous step's state and outputs the current state.

Working Principle

By selectively retaining only the necessary information through the update and reset gates, GRU effectively models the temporal dependencies in sequences.

Options

Options group	Options Name	Description	Note
Network Option	Number of Hidden Layers	Sets the number of hidden layers inside the RNN cell. The deeper the neural network, the more sophisticated the model can be produced, but the higher the risk of overfitting.	1~10
	Number of Nodes	Sets the number of nodes in the hidden state within the RNN cell. The larger the number of nodes, the more sophisticated the model can be created, but the computational cost also increases.	integer
	Activation Function	Set the activation function. Activation function is a mathematical transformation applied to its input.	Linear, Sigmoid, Tanh, ReLU, LeakyReLU, ELU
	Drop-Out	Sets a random percentage of the nodes to be deactivated at each learning stage. When used properly, the model can learn more generalized features and prevent overfitting.	0<real number≤1
Train Option	Epochs	Set the number of iterations.	integer
	Batch Size	Set the number of data samples to be used in a single training step.	integer
	Learning Rate	Set the percentage to update the weights at each learning stage. Excessively large values can disrupt learning, while excessively small values can slow it down.	0<real number≤1

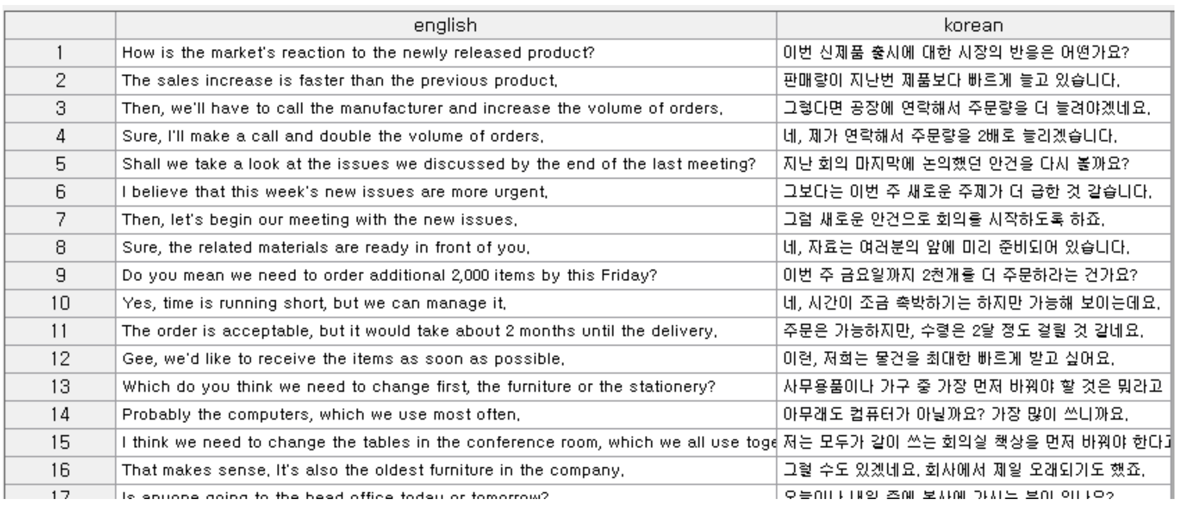
	Target Loss	Specify the target loss; stops learning when the value of the loss function in the model reaches the target loss.	0<real number≤1
	Optimization Method	Set up an algorithm to update the weight of the model.	SGD, RMSprop, Adagrad, Adam
	Scaling Method	Set how to standardize your input data. Improve learning stability and performance.	Min-Max Scaler (0,1), Min-Max Scaler (-1,1), Standard Scaler, None

5.1.5 Language Model

Model that learns from text data to predict the probability distribution of words or sentences. It is used to generate the next word or understand a sentence in a given context, and is applied in various natural language processing tasks such as translation, chatbots, and more.

Data Input Format

The training data should be in the form of input-output pairs.



	english	korean
1	How is the market's reaction to the newly released product?	이번 신제품 출시에 대한 시장의 반응은 어떤가요?
2	The sales increase is faster than the previous product.	판매량이 지난번 제품보다 빠르게 늘고 있습니다.
3	Then, we'll have to call the manufacturer and increase the volume of orders.	그렇다면 공장에 연락해서 주문량을 더 늘려야겠네요.
4	Sure, I'll make a call and double the volume of orders.	네, 제가 연락해서 주문량을 2배로 늘리겠습니다.
5	Shall we take a look at the issues we discussed by the end of the last meeting?	지난 회의 마지막에 논의했던 안건을 다시 볼까요?
6	I believe that this week's new issues are more urgent.	그보다는 이번 주 새로운 주제가 더 급한 것 같습니다.
7	Then, let's begin our meeting with the new issues.	그럼 새로운 안건으로 회의를 시작하도록 하죠.
8	Sure, the related materials are ready in front of you.	네, 자료는 여러분의 앞에 미리 준비되어 있습니다.
9	Do you mean we need to order additional 2,000 items by this Friday?	이번 주 금요일까지 2천개를 더 주문하라는 건가요?
10	Yes, time is running short, but we can manage it.	네, 시간이 조금 촉박하기는 하지만 가능해 보이는데요.
11	The order is acceptable, but it would take about 2 months until the delivery.	주문은 가능하지만, 수령은 2달 정도 걸릴 것 같네요.
12	Gee, we'd like to receive the items as soon as possible.	이런, 저희는 물건을 최대한 빠르게 받고 싶어요.
13	Which do you think we need to change first, the furniture or the stationery?	사무용품이나 가구 중 가장 먼저 바꿔야 할 것은 뭐라고
14	Probably the computers, which we use most often.	아무래도 컴퓨터가 아닐까요? 가장 많이 쓰니까요.
15	I think we need to change the tables in the conference room, which we all use toge	저는 모두가 같이 쓰는 회의실 책상을 먼저 바꿔야 한다고
16	That makes sense. It's also the oldest furniture in the company.	그럴 수도 있겠네요. 회사에서 제일 오래되기도 했죠.
17	Is anyone going to the head office today or tomorrow?	오늘이나 내일 중에 본사에 가시는 분이 있나요?

5.1.5.1 Seq2Seq

Overview

The Seq2Seq model is designed to convert an input sequence into an output sequence using an encoder-decoder architecture. This model is widely used in tasks such as machine translation and automatic summarization, effectively handling complex sequence data.

Components

- **Encoder**

  Encodes the input sequence into a fixed-size state vector.

- **Decoder**

  Uses the state vector received from the encoder to generate the output sequence.

- **Context Vector**

  The vector created by the encoder that compresses all the information from the input sequence.

Working Principle

The encoder transforms the information from the input sequence into a context vector. Whereas the decoder then helps to generate the output sequence from the context vector. This process establishes the semantic connection between the input and output, enabling the model to perform complex sequence transformation tasks.

Options

Options group	Options Name	Description	Note
Network Option	Input Size	Sets the size of the dimensions to be used as input to the model.	128, 256, 512, 1024
	Number of Hidden Layers	Sets the number of hidden layers in a neural network.	
	Number of Nodes	Sets the number of nodes in the hidden state within the LSTM cell of Encoder/Decoder.	128, 256, 512, 1024
	Sequence Maximum Length	Sets the size of the level to reduce words. The greater the relationship between words, but the calculation cost increases.	
	Drop-Out	Sets a random percentage of the nodes to be deactivated at each learning stage. When used properly, the model can learn more generalized features and prevent overfitting.	0<real number≤1
Train Option	Epochs	Set the number of iterations.	integer
	Batch Size	Set the number of data samples to be used in a single training step.	integer
	Learning Rate	Set the percentage to update the weights at each learning stage. Excessively large values can disrupt learning, while excessively small values can slow it down.	0<real number≤1

	Target Loss	Specify the target loss; stops learning when the value of the loss function in the model reaches the target loss.	0<real number≤1
	Optimization Method	Set up an algorithm to update the weight of the model.	SGD, RMSprop, Adagrad, Adam
	Scaling Method	Scaling Method is set to Min-Max Scaler (0,1).	Min-Max Scaler (0,1)

5.1.5.2 Seq2Seq with attention

Overview

Attention is a mechanism that evaluates the importance of different parts of the input sequence, allowing the model to focus more on certain parts when transforming the input sequence into an output sequence. It is applied to models like Seq2Seq in tasks such as machine translation and automatic summarization.

Components

- **Query**

  Information derived from the current output word being generated, based on the input sequence.

- **Key**

  Each element of the input sequence, used to evaluate its similarity with the query.

- **Value**

  The actual information of the input sequence, which contributes to the output according to the attention weights.

Working Principle

The query is compared with the keys to measure their similarity, and attention weights are calculated. These weights are then multiplied by the values and used to generate the final output. This process allows each output word to focus on the most relevant part of the input sequence, helping to prevent information loss, which is a limitation of traditional RNNs.

Options

Options group	**Options Name**	**Description**	**Note**
Network Option	Input Size	Sets the size of the dimensions to be used as input to the model.	128, 256, 512, 1024
	Number of Hidden Layers	Sets the number of hidden layers.	
	Number of Nodes	Sets the number of nodes in the hidden state within the LSTM cell of Encoder/Decoder.	128, 256, 512, 1024
	Number of Heads	Sets the number of heads used by the attention mechanism.	4,8,16
	Sequence Maximum Length	Sets the size of the level to reduce words. The greater the relationship between words, but the calculation cost increases.	
	Drop-Out	Sets a random percentage of the nodes to be deactivated at each learning stage. When used properly, the model can learn more generalized features and prevent overfitting.	$0<$real number≤ 1
Train Option	Epochs	Set the number of iterations.	integer
	Batch Size	Set the number of data samples to be used in a single training step.	integer
	Learning Rate	Set the percentage to update the weights at each learning stage. Excessively large values	$0<$real number≤ 1

		can disrupt learning, while excessively small values can slow it down.	
	Target Loss	Specify the target loss; stops learning when the value of the loss function in the model reaches the target loss.	0<real number≤1
	Optimization Method	Set up an algorithm to update the weight of the model.	SGD, RMSprop, Adagrad, Adam
	Scaling Method	Scaling Method is set to Min-Max Scaler (0,1).	Min-Max Scaler (0,1)

5.1.5.3 Transformer

Overview

The core of the Transformer model lies in the self-attention mechanism. This mechanism allows it to process all parts of the input sequence simultaneously and capture long-range dependencies efficiently.

Components

- **Multi-head Attention**

  Processes information from different positions in parallel, integrating diverse perspectives and capturing various aspects of the input.

- **Feed-forward Neural Networks**

  Applied identically at each position to further process information within the sequence.

- **Positional Encoding**

  Provides the model with information about the position of each element in the sequence, helping preserve the sequence order.

Working Principle

The Transformer can process sequences without relying on traditional RNNs or CNNs, significantly improving computational efficiency and learning speed. By evaluating all sequence elements globally, it excels at understanding context and capturing long-range dependencies.

Options

Options group	Options Name	Description	Note
Network Option	Input Size	Sets the size of the dimensions to be used as input to the model.	128, 256, 512, 1024
	Number of Hidden Layers	Sets the number of layers of Encoder/Decoder.	
	Number of Heads	Sets the number of heads used by the attention mechanism.	4,8,16
	Sequence Maximum Length	Sets the size of the level to reduce words. The greater the relationship between words, but the calculation cost increases.	
	Drop-Out	Sets a random percentage of the nodes to be deactivated at each learning stage. When used properly, the model can learn more generalized features and prevent overfitting.	0<real number≤1
Train Option	Epochs	Set the number of iterations.	integer
	Batch Size	Set the number of data samples to be used in a single training step.	integer
	Learning Rate	Set the percentage to update the weights at each learning stage. Excessively large values can disrupt learning, while excessively small values can slow it down.	0<real number≤1

	Target Loss	Specify the target loss; stops learning when the value of the loss function in the model reaches the target loss.	0<real number≤1
	Optimization Method	Set up an algorithm to update the weight of the model.	SGD, RMSprop, Adagrad, Adam
	Scaling Method	Scaling Method is set to Min-Max Scaler (0,1).	Min-Max Scaler (0,1)

NOTE:

In the Multi-Head Self-Attention (MHSA) layer, the input vector must be evenly distributed across each head, so the input size (dimension) should be divisible by the number of heads.