一張含有 文字, 螢幕擷取畫面, 收據, 字型 的圖片

自動產生的描述

The architecture settings for each module in the MNIST classification task are as follows:

1. Transmitter: The transmitter module is responsible for encoding the input data. It consists of a dense layer with an output dimension of 64. The activation function used in this layer is the hyperbolic tangent (Tanh) function, which outputs a value between -1 and 1. This helps to normalize the output of the layer and control the exploding gradients problem.

2. Receiver (Classifier): The receiver module is responsible for decoding the transmitted data and classifying it. It consists of three dense layers. The first layer has an output dimension of 1024 and uses the Rectified Linear Unit (ReLU) activation function, which outputs the input directly if it is positive, otherwise, it will output zero. The second layer has an output dimension of 256 and also uses the ReLU activation function. The final layer has an output dimension of 10 (corresponding to the 10 digits in the MNIST dataset) and uses the softmax activation function, which outputs a probability distribution over the 10 classes, making it suitable for multi-class classification tasks.

3. Receiver (Decoder): The receiver also has a decoder module, which is responsible for reconstructing the original input data from the encoded data. It consists of a dense layer with an output dimension of 784 (corresponding to the 28x28 pixel images in the MNIST dataset). The activation function used in this layer is the Tanh function, which helps to normalize the output and control the exploding gradients problem.

Code:

from keras.datasets import mnist

from keras.models import Model

from keras.layers import Input, Dense

from keras.utils import to\_categorical

# Load MNIST dataset

(x\_train, y\_train), (x\_test, y\_test) = mnist.load\_data()

# Preprocess data

x\_train = x\_train.reshape(-1, 784).astype('float32') / 255

x\_test = x\_test.reshape(-1, 784).astype('float32') / 255

y\_train = to\_categorical(y\_train, 10)

y\_test = to\_categorical(y\_test, 10)

# Transmitter

input\_img = Input(shape=(784,))

encoded = Dense(64, activation='tanh')(input\_img)

# Receiver (Classifier)

x = Dense(1024, activation='relu')(encoded)

x = Dense(256, activation='relu')(x)

classifier\_output = Dense(10, activation='softmax')(x)

# Receiver (Decoder)

decoded = Dense(784, activation='tanh')(encoded)

# Define the models

encoder = Model(input\_img, encoded)

classifier = Model(input\_img, classifier\_output)

autoencoder = Model(input\_img, decoded)

# Compile the models

classifier.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])

autoencoder.compile(optimizer='adam', loss='mean\_squared\_error')

# Train the models

classifier.fit(x\_train, y\_train, epochs=10, batch\_size=256, validation\_data=(x\_test, y\_test))

autoencoder.fit(x\_train, x\_train, epochs=10, batch\_size=256, validation\_data=(x\_test, x\_test))

一張含有 文字, 螢幕擷取畫面, 字型, 數字 的圖片

自動產生的描述

Code:

from keras.models import Model

from keras.layers import Input, Dense, Conv2D, MaxPooling2D, UpSampling2D, Flatten, Reshape

from keras.layers import add, Activation

# Function for creating a residual block

def residual\_block(input\_tensor, filters):

x = Conv2D(filters, (3, 3), padding='same')(input\_tensor)

x = Activation('relu')(x)

x = Conv2D(filters, (3, 3), padding='same')(x)

output\_tensor = add([x, input\_tensor])

output\_tensor = Activation('relu')(output\_tensor)

return output\_tensor

# Transmitter

input\_img = Input(shape=(32, 32, 3))

x = Conv2D(128, (3, 3), activation='relu', padding='same')(input\_img)

x = Conv2D(128, (3, 3), activation='relu', padding='same')(x)

x = residual\_block(x, 128)

x = Conv2D(8, (3, 3), activation='relu', padding='same')(x)

x = Conv2D(8, (3, 3), activation='relu', padding='same')(x)

x = Conv2D(8, (3, 3), activation='relu', padding='same')(x)

x = Flatten()(x)

encoded = Dense(128, activation='tanh')(x)

# Receiver (Classifier)

x = Dense(512, activation='relu')(encoded)

x = Reshape((4, 4, 32))(x)

x = Conv2D(512, (3, 3), activation='relu', padding='same')(x)

x = residual\_block(x, 512)

x = MaxPooling2D((2, 2), padding='same')(x)

x = Flatten()(x)

classifier\_output = Dense(10, activation='softmax')(x)

# Receiver (Decoder)

x = Dense(512, activation='tanh')(encoded)

x = Reshape((4, 4, 32))(x)

x = Conv2D(512, (3, 3), activation='relu', padding='same')(x)

x = UpSampling2D((2, 2))(x)

x = Conv2D(128, (3, 3), activation='relu', padding='same')(x)

x = Conv2D(128, (3, 3), activation='relu', padding='same')(x)

x = residual\_block(x, 128)

x = UpSampling2D((2, 2))(x)

x = Conv2D(64, (3, 3), activation='relu', padding='same')(x)

decoded = Conv2D(3, (3, 3), activation='sigmoid', padding='same')(x)

# Define the models

encoder = Model(input\_img, encoded)

classifier = Model(input\_img, classifier\_output)

autoencoder = Model(input\_img, decoded)

# Compile the models

classifier.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])

autoencoder.compile(optimizer='adam', loss='mean\_squared\_error')

一張含有 文字, 螢幕擷取畫面, 收據, 字型 的圖片

自動產生的描述

Code:

from keras.datasets import mnist, cifar10

from keras.utils import to\_categorical

from keras.models import Model

from keras.layers import Input, Dense, Conv2D, UpSampling2D, Flatten, Reshape

from keras.layers import add, Activation

from keras.optimizers import Adam

# Function for creating a residual block

def residual\_block(input\_tensor, filters):

x = Conv2D(filters, (3, 3), padding='same')(input\_tensor)

x = Activation('relu')(x)

x = Conv2D(filters, (3, 3), padding='same')(x)

output\_tensor = add([x, input\_tensor])

output\_tensor = Activation('relu')(output\_tensor)

return output\_tensor

# Load and preprocess the MNIST data

(x\_train, \_), (x\_test, \_) = mnist.load\_data()

x\_train = x\_train.astype('float32') / 255.

x\_test = x\_test.astype('float32') / 255.

x\_train = x\_train.reshape((len(x\_train), np.prod(x\_train.shape[1:])))

x\_test = x\_test.reshape((len(x\_test), np.prod(x\_test.shape[1:])))

# Define the MNIST model

input\_img = Input(shape=(784,))

x = Dense(784, activation='tanh')(input\_img)

mnist\_model = Model(input\_img, x)

# Compile the MNIST model

mnist\_model.compile(optimizer=Adam(), loss='mean\_squared\_error')

# Train the MNIST model

mnist\_model.fit(x\_train, x\_train,

epochs=50,

batch\_size=256,

shuffle=True,

validation\_data=(x\_test, x\_test))

# Load and preprocess the CIFAR-10 data

(x\_train, \_), (x\_test, \_) = cifar10.load\_data()

x\_train = x\_train.astype('float32') / 255.

x\_test = x\_test.astype('float32') / 255.

x\_train = x\_train.reshape((len(x\_train), np.prod(x\_train.shape[1:])))

x\_test = x\_test.reshape((len(x\_test), np.prod(x\_test.shape[1:])))

# Define the CIFAR-10 model

input\_img = Input(shape=(3072,))

x = Dense(512, activation='tanh')(input\_img)

x = Reshape((4, 4, 32))(x)

x = Conv2D(512, (3, 3), activation='relu', padding='same')(x)

x = UpSampling2D((2, 2))(x)

x = Conv2D(256, (3, 3), activation='relu', padding='same')(x)

x = residual\_block(x, 256)

x = UpSampling2D((2, 2))(x)

x = Conv2D(128, (3, 3), activation='relu', padding='same')(x)

x = residual\_block(x, 128)

x = UpSampling2D((2, 2))(x)

x = Conv2D(64, (3, 3), activation='relu', padding='same')(x)

x = residual\_block(x, 64)

decoded = Conv2D(3, (3, 3), activation='sigmoid', padding='same')(x)

cifar\_model = Model(input\_img, decoded)

# Compile the CIFAR-10 model

cifar\_model.compile(optimizer=Adam(), loss='mean\_squared\_error')

# Train the CIFAR-10 model

cifar\_model.fit(x\_train, x\_train,

epochs=50,

batch\_size=256,

shuffle=True,

validation\_data=(x\_test, x\_test))