from tensorflow.keras import layers

from tensorflow.keras import models

model = models.Sequential()

model.add(layers.Conv2D(32, (3, 3), activation='relu', input\_shape=(28, 28, 1)))

model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(64, (3, 3), activation='relu'))

model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(64, (3, 3), activation='relu'))

from tensorflow.keras import layers from tensorflow.keras import models

**TensorFlow.keras library se 'layers' aur 'models' modules import kiye gaye hain.**

model = models.Sequential()

**'Sequential' model banaya gaya hai, jo layers ko ek ke baad ek linear sequence mein define karta hai.**

model.add(layers.Conv2D(32, (3, 3), activation='relu', input\_shape=(28, 28, 1)))

**'Conv2D' layer add kiya gaya hai jis mein 32 filters hain. Har ek filter 3x3 kernel ka hai. 'relu' activation function istemaal kiya gaya hai.**

**'input\_shape' argument se input ka shape define kiya gaya hai. Yahan pe input ka shape (28, 28, 1) hai, yani 28x28 pixels ka grayscale image hai.**

model.add(layers.MaxPooling2D((2, 2)))

**'MaxPooling2D' layer add kiya gaya hai jis mein (2, 2) window size ka pooling kiya jata hai.**

**Max pooling ka matlab hota hai ke har window mein se maximum value ko select karke output layer mein pass karna.**

model.add(layers.Conv2D(64, (3, 3), activation='relu'))

**Fir se ek 'Conv2D' layer add kiya gaya hai jismein 64 filters hain. Har ek filter 3x3 kernel ka hai. 'relu' activation function istemaal kiya gaya hai.**

model.add(layers.MaxPooling2D((2, 2)))

**Fir se 'MaxPooling2D' layer add kiya gaya hai jis mein (2, 2) window size ka pooling kiya jata hai.**

model.add(layers.Conv2D(64, (3, 3), activation='relu'))

**Aur ek 'Conv2D' layer add kiya gaya hai jismein 64 filters hain. Har ek filter 3x3 kernel ka hai. 'relu' activation function istemaal kiya gaya hai.**

Explanation:

* Code mein sabse pehle 'layers' aur 'models' modules TensorFlow.keras se import kiye gaye hain.
* Fir 'Sequential' model banaya gaya hai, jo layers ko ek ke baad ek linear sequence mein define karta hai.
* Phir se model mein layers add kiye jaate hain.
* Pehle 'Conv2D' layer add kiya gaya hai jismein 32 filters hain. Har ek filter 3x3 kernel ka hai. 'relu' activation function istemaal kiya gaya hai.
* Fir 'MaxPooling2D' layer add kiya gaya hai jis mein (2, 2) window size ka pooling kiya jata hai.
* Fir se 'Conv2D' layer add kiya gaya hai jismein 64 filters hain. Har ek filter 3x3 kernel ka hai. 'relu' activation function istemaal kiya gaya hai.
* Fir se 'MaxPooling2D' layer add kiya gaya hai jis mein (2, 2) window size ka pooling kiya jata hai.
* Phir se ek 'Conv2D' layer add kiya gaya hai jismein 64 filters hain. Har ek filter 3x3 kernel ka hai. 'relu' activation function istemaal kiya gaya hai.

Yeh code ek Convolutional Neural Network (CNN) model define kar raha hai, jo 28x28 pixels ke grayscale images ko classify karne ke liye design kiya gaya hai. Is model mein convolutional layers aur max pooling layers ka istemaal kiya gaya hai, jisse image features extract hote hain aur spatial information retain hoti hai.

model.add(layers.Flatten())

model.add(layers.Dense(64, activation='relu'))

model.add(layers.Dense(10, activation='softmax'))

model.add(layers.Flatten())

**'Flatten' layer add kiya gaya hai. Yeh layer input data ko ek linear array mein convert kar deti hai.**

**Isse input image ka spatial structure khatm ho jata hai aur use flat vector form mein convert kar diya jata hai.**

model.add(layers.Dense(64, activation='relu'))

**'Dense' layer add kiya gaya hai jismein 64 neurons hain. Yeh layer fully connected (poora tarah se jude hue) hai, jiske har neuron se har input connected hota hai.**

**'relu' activation function istemaal kiya gaya hai. ReLU function, yani Rectified Linear Unit, positive values ko as it is rakhta hai aur negative values ko zero banata hai.**

model.add(layers.Dense(10, activation='softmax'))

**Fir ek aur 'Dense' layer add kiya gaya hai jismein 10 neurons hain. Yeh layer bhi fully connected hai.**

**'softmax' activation function istemaal kiya gaya hai. Softmax function input values ko probability distribution mein convert karta hai, jisse har neuron ka output probability hai.**

Explanation:

* Code mein pehle 'Flatten' layer add kiya gaya hai. Is layer ka kaam hai input data ko ek linear array mein convert karna. Yani, input image ka spatial structure khatm ho jata hai aur use flat vector form mein convert kar diya jata hai.
* Fir ek 'Dense' layer add kiya gaya hai jismein 64 neurons hain. Yeh layer fully connected hai, jiske har neuron se har input connected hota hai. Is layer mein 'relu' activation function istemaal kiya gaya hai. ReLU function, yani Rectified Linear Unit, positive values ko as it is rakhta hai aur negative values ko zero banata hai. Isse non-linearity introduce hoti hai aur complex patterns ko model karne ki capability badhti hai.
* Fir ek aur 'Dense' layer add kiya gaya hai jismein 10 neurons hain. Yeh layer bhi fully connected hai. Is layer mein 'softmax' activation function istemaal kiya gaya hai. Softmax function input values ko probability distribution mein convert karta hai, jisse har neuron ka output probability hai. Yani, yeh layer classification task ke liye hai, jahaan har neuron output ko ek class ke probability ke roop mein represent karta hai.

Yeh code CNN model ke end mein fully connected layers (Dense layers) ko add kar raha hai, jisse CNN ki output features ko flatten karke use karke classification task ko perform kiya jaa sake.

train\_images = train\_images.reshape((60000, 28, 28, 1))

train\_images = train\_images.astype('float32') / 255

test\_images = test\_images.reshape((10000, 28, 28, 1))

test\_images = test\_images.astype('float32') / 255

train\_labels = to\_categorical(train\_labels)

test\_labels = to\_categorical(test\_labels)

model.compile(optimizer='rmsprop',

              loss='categorical\_crossentropy',

              metrics=['accuracy'])

model.fit(train\_images, train\_labels, epochs=5, batch\_size=64)

train\_images = train\_images.reshape((60000, 28, 28, 1))

**'train\_images' ko reshape kiya gaya hai, taaki uska shape (60000, 28, 28, 1) ho jaye. Yani, har ek image ka shape 28x28 pixels ka hai aur ek channel (grayscale) hai.**

train\_images = train\_images.astype('float32') / 255

**'train\_images' ko 'float32' data type mein convert kiya gaya hai aur phir 255 se divide karke normalize kiya gaya hai.**

**Isse train\_images ke pixel values 0 aur 1 ke beech mein scale ho jate hain, jismein 0 black aur 1 white hota hai.**

test\_images = test\_images.reshape((10000, 28, 28, 1))

**'test\_images' ko reshape kiya gaya hai, taaki uska shape (10000, 28, 28, 1) ho jaye. Yani, har ek image ka shape 28x28 pixels ka hai aur ek channel (grayscale) hai.**

test\_images = test\_images.astype('float32') / 255

**'test\_images' ko 'float32' data type mein convert kiya gaya hai aur phir 255 se divide karke normalize kiya gaya hai.**

**Isse test\_images ke pixel values 0 aur 1 ke beech mein scale ho jate hain, jismein 0 black aur 1 white hota hai.**

train\_labels = to\_categorical(train\_labels) test\_labels = to\_categorical(test\_labels)

**'train\_labels' aur 'test\_labels' ko one-hot encoding kiya gaya hai. Yani, unki categorical values ko binary vectors mein convert kiya gaya hai.**

**Isse classification task mein sahi label ke probabilities ko model asaani se predict kar sakta hai.**

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

**Model compile kiya gaya hai. 'optimizer' mein 'rmsprop' use kiya gaya hai, 'loss' mein 'categorical\_crossentropy' use kiya gaya hai.**

**'rmsprop' optimizer gradient descent technique ka ek variant hai, jo model ke weights ko update karta hai.**

**'categorical\_crossentropy' loss function multiclass classification task ke liye istemaal hoti hai.**

model.fit(train\_images, train\_labels, epochs=5, batch\_size=64)

**Model ko 'train\_images' aur 'train\_labels' dataset se fit kiya gaya hai. 'epochs' argument se total training epochs (iterations) define kiye gaye hain.**

**'batch\_size' argument se har training step mein kitne samples ko process kiya jaye, yeh define kiya gaya hai.**

Explanation:

* Pehle 'train\_images' ko reshape kiya gaya hai, taki uska shape (60000, 28, 28, 1) ho jaye. Yani, har ek image ka shape 28x28 pixels ka hai aur ek channel (grayscale) hai.
* Fir 'train\_images' ko 'float32' data type mein convert kiya gaya hai aur 255 se divide karke normalize kiya gaya hai, jisse pixel values 0 aur 1 ke beech mein scale ho jate hain.
* Usi tarah se 'test\_images' ko bhi reshape kiya gaya hai, phir 'float32' data type mein convert kiya gaya hai aur normalize kiya gaya hai.
* 'train\_labels' aur 'test\_labels' ko one-hot encoding kiya gaya hai. Yani, unki categorical values ko binary vectors mein convert kiya gaya hai.
* Fir model compile kiya gaya hai. 'optimizer' mein 'rmsprop' use kiya gaya hai, 'loss' mein 'categorical\_crossentropy' use kiya gaya hai.
* 'model.fit' function se model ko 'train\_images' aur 'train\_labels' dataset se fit kiya gaya hai. 'epochs' argument se total training epochs (iterations) define kiye gaye hain aur 'batch\_size' argument se har training step mein kitne samples ko process kiya jaye, yeh define kiya gaya hai.

test\_loss, test\_acc = model.evaluate(test\_images, test\_labels)

print(test\_acc)

print(test\_loss)

test\_loss, test\_acc = model.evaluate(test\_images, test\_labels)

**Model ko evaluate kiya gaya hai 'test\_images' aur 'test\_labels' dataset par. 'evaluate' function model ke performance metrics ko calculate karta hai.**

**'test\_loss' variable mein test loss value store ki gayi hai aur 'test\_acc' variable mein test accuracy value store ki gayi hai.**

print(test\_acc)

**'test\_acc' variable ki value print ki gayi hai. Yeh model ki test accuracy ko represent karti hai.**

print(test\_loss)

**'test\_loss' variable ki value print ki gayi hai. Yeh model ki test loss ko represent karti hai.**

Explanation:

* Code mein pehle 'model.evaluate' function ka istemaal kiya gaya hai. Isse model ko 'test\_images' aur 'test\_labels' dataset par evaluate kiya gaya hai.
* 'test\_loss' aur 'test\_acc' variables mein model ke test loss aur test accuracy values store ho gayi hain, jinhe 'model.evaluate' function return karta hai.
* Fir 'test\_acc' variable ki value print ki gayi hai, jisse model ki test accuracy result dikhai gayi hai.
* Usi tarah se 'test\_loss' variable ki value print ki gayi hai, jisse model ki test loss result dikhai gayi hai.

import numpy as np

from sklearn.metrics import precision\_score, recall\_score

# Get the predicted labels from the model

predictions = model.predict(test\_images)

predicted\_labels = np.argmax(predictions, axis=1)

# Convert the one-hot encoded test labels back to categorical labels

true\_labels = np.argmax(test\_labels, axis=1)

# Calculate precision and recall

precision = precision\_score(true\_labels, predicted\_labels, average='weighted')

recall = recall\_score(true\_labels, predicted\_labels, average='weighted')

print("Precision:", precision)

print("Recall:", recall)

import numpy as np from sklearn.metrics import precision\_score, recall\_score

**'numpy' library se 'np' alias ka import kiya gaya hai. Iska istemaal numerical computations ke liye hota hai.**

**'precision\_score' aur 'recall\_score' functions 'sklearn.metrics' module se import kiye gaye hain.**

predictions = model.predict(test\_images)

**Model se 'test\_images' dataset par predictions calculate kiye gaye hain.**

predicted\_labels = np.argmax(predictions, axis=1)

**'argmax' function se predictions se predicted labels nikale gaye hain. 'axis=1' argument specify kiya gaya hai takse predictions ka maximum value wale index ko select kiya jaye.**

true\_labels = np.argmax(test\_labels, axis=1)

**'argmax' function se 'test\_labels' ko categorical format mein convert kiya gaya hai. 'axis=1' argument specify kiya gaya hai takse one-hot encoded labels se categorical labels (class indices) ko select kiya jaye.**

precision = precision\_score(true\_labels, predicted\_labels, average='weighted')

**'precision\_score' function se precision score calculate kiya gaya hai, jismein true labels aur predicted labels as inputs diye gaye hain. 'average' argument mein 'weighted' specify kiya gaya hai, jisse weighted average precision score calculate kiya jayega.**

recall = recall\_score(true\_labels, predicted\_labels, average='weighted')

**'recall\_score' function se recall score calculate kiya gaya hai, jismein true labels aur predicted labels as inputs diye gaye hain. 'average' argument mein 'weighted' specify kiya gaya hai, jisse weighted average recall score calculate kiya jayega.**

print("Precision:", precision)

**'precision' variable ki value print ki gayi hai. Yeh precision score ko represent karti hai.**

print("Recall:", recall)

**'recall' variable ki value print ki gayi hai. Yeh recall score ko represent karti hai.**

Explanation:

* Pehle 'numpy' library se 'np' alias ka import kiya gaya hai, jiska istemaal numerical computations ke liye hota hai.
* Fir model se 'test\_images' dataset par predictions calculate kiye gaye hain, jiske results 'predictions' variable mein store ho gaye hain.
* 'argmax' function ka istemaal karke 'predictions' se predicted labels nikale gaye hain, jiske results 'predicted\_labels' variable mein store ho gaye hain.
* 'argmax' function se 'test\_labels' ko categorical format mein convert kiya gaya hai, jiske results 'true\_labels' variable mein store ho gaye hain.
* 'precision\_score' function ka istemaal karke true labels aur predicted labels se precision score calculate kiya gaya hai, jiske results 'precision' variable mein store ho gaya hai. 'average' argument mein 'weighted' specify kiya gaya hai, jisse weighted average precision score calculate kiya jata hai.
* Usi tarah se 'recall\_score' function ka istemaal karke true labels aur predicted labels se recall score calculate kiya gaya hai, jiske results 'recall' variable mein store ho gaya hai. 'average' argument mein 'weighted' specify kiya gaya hai, jisse weighted average recall score calculate kiya jata hai.
* 'Precision' aur 'Recall' ke values print ki gayi hain, jisse precision aur recall scores result dikhai gaye hain.