**import** numpy **as** np  
**from** keras.datasets **import** cifar10  
**from** keras.models **import** Sequential  
**from** keras.layers **import** Dense, Dropout, Flatten  
**from** keras.constraints **import** maxnorm  
**from** keras.optimizers **import** SGD  
**from** keras.layers.convolutional **import** Conv2D, MaxPooling2D  
**from** keras.utils **import** np\_utils

[6]

np.random.seed(7)

[7]

(X\_train, y\_train), (X\_test, y\_test) = cifar10.load\_data()

[8]

X\_train = X\_train.astype('float32') / 255.0  
X\_test = X\_test.astype('float32') / 255.0

[9]

y\_train = np\_utils.to\_categorical(y\_train)  
y\_test = np\_utils.to\_categorical(y\_test)  
num\_classes = y\_test.shape[1]

[10]

model = Sequential()  
model.add(Conv2D(32, (3, 3), input\_shape=(32, 32, 3), padding='same', activation='relu', kernel\_constraint=maxnorm(3)))  
model.add(Dropout(0.2))  
model.add(Conv2D(32, (3, 3), activation='relu', padding='same', kernel\_constraint=maxnorm(3)))  
model.add(MaxPooling2D(pool\_size=(2, 2), padding='same'))  
model.add(Flatten())  
model.add(Dense(512, activation='relu', kernel\_constraint=maxnorm(3)))  
model.add(Dropout(0.5))  
model.add(Dense(num\_classes, activation='softmax'))

[11]

sgd = SGD(learning\_rate=0.01, momentum=0.9, decay=1e-6)  
model.compile(loss='categorical\_crossentropy', optimizer=sgd, metrics=['accuracy'])  
print(model.summary())

Model: "sequential\_1"  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
 Layer (type) Output Shape Param #   
=================================================================  
 conv2d\_2 (Conv2D) (None, 32, 32, 32) 896   
   
 dropout\_2 (Dropout) (None, 32, 32, 32) 0   
   
 conv2d\_3 (Conv2D) (None, 32, 32, 32) 9248   
   
 max\_pooling2d\_1 (MaxPooling (None, 16, 16, 32) 0   
 2D)   
   
 flatten\_1 (Flatten) (None, 8192) 0   
   
 dense\_2 (Dense) (None, 512) 4194816   
   
 dropout\_3 (Dropout) (None, 512) 0   
   
 dense\_3 (Dense) (None, 10) 5130   
   
=================================================================  
Total params: 4,210,090  
Trainable params: 4,210,090  
Non-trainable params: 0  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
None

[12]

epochs = 5  
batch\_size = 32  
model.fit(X\_train, y\_train, validation\_data=(X\_test, y\_test), epochs=epochs, batch\_size=batch\_size)

Epoch 1/5  
1563/1563 [==============================] - 19s 7ms/step - loss: 1.7232 - accuracy: 0.3746 - val\_loss: 1.4776 - val\_accuracy: 0.4563  
Epoch 2/5  
1563/1563 [==============================] - 10s 6ms/step - loss: 1.3675 - accuracy: 0.5117 - val\_loss: 1.2470 - val\_accuracy: 0.5551  
Epoch 3/5  
1563/1563 [==============================] - 10s 6ms/step - loss: 1.2071 - accuracy: 0.5716 - val\_loss: 1.1232 - val\_accuracy: 0.6047  
Epoch 4/5  
1563/1563 [==============================] - 10s 7ms/step - loss: 1.0855 - accuracy: 0.6136 - val\_loss: 1.1554 - val\_accuracy: 0.5928  
Epoch 5/5  
1563/1563 [==============================] - 10s 7ms/step - loss: 0.9709 - accuracy: 0.6583 - val\_loss: 0.9986 - val\_accuracy: 0.6550

<keras.callbacks.History at 0x7f689d6d65e0>

[13]

scores = model.evaluate(X\_test, y\_test, verbose=0)  
print("Accuracy: %.2f%%" % (scores[1]\*100))

Accuracy: 65.50%

[14]

**import** numpy **as** np  
**from** keras.datasets **import** cifar10  
**from** keras.models **import** Sequential  
**from** keras.layers **import** Dense, Dropout, Flatten  
**from** keras.layers.convolutional **import** Conv2D, MaxPooling2D  
**from** keras.constraints **import** maxnorm  
**from** keras.utils **import** np\_utils  
**from** keras.optimizers **import** SGD  
  
*# Fix random seed for reproducibility*  
np.random.seed(7)  
  
*# Load data*  
(X\_train, y\_train), (X\_test, y\_test) = cifar10.load\_data()  
  
*# Normalize inputs from 0-255 to 0.0-1.0*  
X\_train = X\_train.astype('float32') / 255.0  
X\_test = X\_test.astype('float32') / 255.0  
  
*# One hot encode outputs*  
y\_train = np\_utils.to\_categorical(y\_train)  
y\_test = np\_utils.to\_categorical(y\_test)  
num\_classes = y\_test.shape[1]  
  
*# Create the model*  
model = Sequential()  
model.add(Conv2D(32, (3, 3), input\_shape=(32, 32, 3), padding='same', activation='relu', kernel\_constraint=maxnorm(3)))  
model.add(Dropout(0.2))  
model.add(Conv2D(32, (3, 3), activation='relu', padding='same', kernel\_constraint=maxnorm(3)))  
model.add(MaxPooling2D(pool\_size=(2, 2)))  
model.add(Conv2D(64, (3, 3), activation='relu', padding='same', kernel\_constraint=maxnorm(3)))  
model.add(Dropout(0.2))  
model.add(Conv2D(64, (3, 3), activation='relu', padding='same', kernel\_constraint=maxnorm(3)))  
model.add(MaxPooling2D(pool\_size=(2, 2)))  
model.add(Conv2D(128, (3, 3), activation='relu', padding='same', kernel\_constraint=maxnorm(3)))  
model.add(Dropout(0.2))  
model.add(Conv2D(128, (3, 3), activation='relu', padding='same', kernel\_constraint=maxnorm(3)))  
model.add(MaxPooling2D(pool\_size=(2, 2)))  
model.add(Flatten())  
model.add(Dropout(0.2))  
model.add(Dense(1024, activation='relu', kernel\_constraint=maxnorm(3)))  
model.add(Dropout(0.2))  
model.add(Dense(512, activation='relu', kernel\_constraint=maxnorm(3)))  
model.add(Dropout(0.2))  
model.add(Dense(num\_classes, activation='softmax'))  
  
*# Compile model*  
epochs = 5  
learning\_rate = 0.01  
decay\_rate = learning\_rate / epochs  
sgd = SGD(lr=learning\_rate, momentum=0.9, decay=decay\_rate, nesterov=False)  
model.compile(loss='categorical\_crossentropy', optimizer=sgd, metrics=['accuracy'])  
print(model.summary())  
  
*# Fit the model*  
history = model.fit(X\_train, y\_train, validation\_data=(X\_test, y\_test), epochs=epochs, batch\_size=32)  
  
*# Evaluate the model*  
scores = model.evaluate(X\_test, y\_test, verbose=0)  
print("Accuracy: %.2f%%" % (scores[1] \* 100))

Model: "sequential\_2"  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
 Layer (type) Output Shape Param #   
=================================================================  
 conv2d\_4 (Conv2D) (None, 32, 32, 32) 896   
   
 dropout\_4 (Dropout) (None, 32, 32, 32) 0   
   
 conv2d\_5 (Conv2D) (None, 32, 32, 32) 9248   
   
 max\_pooling2d\_2 (MaxPooling (None, 16, 16, 32) 0   
 2D)   
   
 conv2d\_6 (Conv2D) (None, 16, 16, 64) 18496   
   
 dropout\_5 (Dropout) (None, 16, 16, 64) 0   
   
 conv2d\_7 (Conv2D) (None, 16, 16, 64) 36928   
   
 max\_pooling2d\_3 (MaxPooling (None, 8, 8, 64) 0   
 2D)   
   
 conv2d\_8 (Conv2D) (None, 8, 8, 128) 73856   
   
 dropout\_6 (Dropout) (None, 8, 8, 128) 0   
   
 conv2d\_9 (Conv2D) (None, 8, 8, 128) 147584   
   
 max\_pooling2d\_4 (MaxPooling (None, 4, 4, 128) 0   
 2D)   
   
 flatten\_2 (Flatten) (None, 2048) 0   
   
 dropout\_7 (Dropout) (None, 2048) 0   
   
 dense\_4 (Dense) (None, 1024) 2098176   
   
 dropout\_8 (Dropout) (None, 1024) 0   
   
 dense\_5 (Dense) (None, 512) 524800   
   
 dropout\_9 (Dropout) (None, 512) 0   
   
 dense\_6 (Dense) (None, 10) 5130   
   
=================================================================  
Total params: 2,915,114  
Trainable params: 2,915,114  
Non-trainable params: 0  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
None  
Epoch 1/5  
1563/1563 [==============================] - 15s 9ms/step - loss: 1.9322 - accuracy: 0.2796 - val\_loss: 1.6108 - val\_accuracy: 0.4168  
Epoch 2/5  
1563/1563 [==============================] - 13s 9ms/step - loss: 1.5375 - accuracy: 0.4379 - val\_loss: 1.4261 - val\_accuracy: 0.4795  
Epoch 3/5  
1563/1563 [==============================] - 13s 9ms/step - loss: 1.3979 - accuracy: 0.4918 - val\_loss: 1.3406 - val\_accuracy: 0.5164  
Epoch 4/5  
1563/1563 [==============================] - 13s 8ms/step - loss: 1.3128 - accuracy: 0.5217 - val\_loss: 1.2901 - val\_accuracy: 0.5367  
Epoch 5/5  
1563/1563 [==============================] - 13s 9ms/step - loss: 1.2504 - accuracy: 0.5459 - val\_loss: 1.1804 - val\_accuracy: 0.5735  
Accuracy: 57.35%

[18]

*# Predict the first 4 images of the test data*  
predictions = model.predict(X\_test[:4])  
*# Convert the predictions to class labels*  
predicted\_labels = numpy.argmax(predictions, axis=1)  
*# Convert the actual labels to class labels*  
actual\_labels = numpy.argmax(y\_test[:4], axis=1)  
  
*# Print the predicted and actual labels for the first 4 images*  
print("Predicted labels:", predicted\_labels)  
print("Actual labels:   ", actual\_labels)

1/1 [==============================] - 0s 21ms/step  
Predicted labels: [3 8 8 8]  
Actual labels: [3 8 8 0]

[19]

**import** matplotlib.pyplot **as** plt  
  
*# Plot the training and validation loss*  
plt.plot(history.history['loss'])  
plt.plot(history.history['val\_loss'])  
plt.title('Model Loss')  
plt.ylabel('Loss')  
plt.xlabel('Epoch')  
plt.legend(['train', 'val'], loc='upper right')  
plt.show()  
  
*# Plot the training and validation accuracy*  
plt.plot(history.history['accuracy'])  
plt.plot(history.history['val\_accuracy'])  
plt.title('Model Accuracy')  
plt.ylabel('Accuracy')  
plt.xlabel('Epoch')  
plt.legend(['train', 'val'], loc='lower right')  
plt.show()

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generated

Github link :