NEURAL NETWORKS ASSIGNMENT-3

Deep Learning Image Classification with CNN

Name: Saharsha Muddagauni STD ID: 700742790

GITHUB LINK: https://github.com/sxm26790/ASSIGNMENT3_NEURAL

To perform image classification with a CNN, we'll use the Keras library, which provides an easy-to-use API for building and training deep learning models. Let's go through the implementation step by step:

Import the required libraries:

```
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


np.random.seed(7)
```

Load and preprocess the data: We'll use the CIFAR-10 dataset, which contains 50,000 training images and 10,000 test images of 10 different classes.

```
(X_train, y_train), (X_test, y_test) = cifar10.load_data()
```

```
    Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
    170498071/170498071 [==============================] - 3s 0us/step
```

```
X_train = X_train.astype('float32') / 255.0
X_test = X_test.astype('float32') / 255.0


y_train = np_utils.to_categorical(y_train)
y_test = np_utils.to_categorical(y_test)
num_classes = y_test.shape[1]
```

Build the CNN 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), 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'))


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"
    _________________________________________________________________
     Layer (type)                Output Shape              Param #
    =================================================================
     conv2d (Conv2D)             (None, 32, 32, 32)        896

     dropout (Dropout)           (None, 32, 32, 32)        0

     conv2d_1 (Conv2D)           (None, 32, 32, 32)        9248

     max_pooling2d (MaxPooling2D  (None, 16, 16, 32)       0
     )

     flatten (Flatten)           (None, 8192)              0

     dense (Dense)               (None, 512)               4194816

     dropout_1 (Dropout)         (None, 512)               0

     dense_1 (Dense)             (None, 10)                5130
    =================================================================
    Total params: 4,210,090
    Trainable params: 4,210,090
    Non-trainable params: 0
    _________________________________________________________________
    None
```

```
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 [==============================] - 164s 105ms/step - loss: 1.7276 - accuracy: 0.3737 - val_loss: 1.4764 - val_accuracy: 0.4738
    Epoch 2/5
    1563/1563 [==============================] - 168s 108ms/step - loss: 1.4083 - accuracy: 0.4939 - val_loss: 1.2520 - val_accuracy: 0.5621
    Epoch 3/5
    1563/1563 [==============================] - 168s 108ms/step - loss: 1.2226 - accuracy: 0.5638 - val_loss: 1.1403 - val_accuracy: 0.5962
    Epoch 4/5
    1563/1563 [==============================] - 167s 107ms/step - loss: 1.0943 - accuracy: 0.6109 - val_loss: 1.0451 - val_accuracy: 0.6321
    Epoch 5/5
    1563/1563 [==============================] - 169s 108ms/step - loss: 0.9879 - accuracy: 0.6515 - val_loss: 1.1353 - val_accuracy: 0.6127
    <keras.callbacks.History at 0x781917f1de40>
```

```
scores = model.evaluate(X_test, y_test, verbose=0)
print("Accuracy: %.2f%%" % (scores[1]*100))
```

```
    Accuracy: 61.27%
```

```
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_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)                                                             
                                                                     
     conv2d_4 (Conv2D)           (None, 16, 16, 64)        18496     
                                                                     
     dropout_3 (Dropout)         (None, 16, 16, 64)        0         
                                                                     
     conv2d_5 (Conv2D)           (None, 16, 16, 64)        36928     
                                                                     
     max_pooling2d_2 (MaxPooling  (None, 8, 8, 64)         0         
     2D)                                                             
                                                                     
     conv2d_6 (Conv2D)           (None, 8, 8, 128)         73856     
                                                                     
     dropout_4 (Dropout)         (None, 8, 8, 128)         0         
                                                                     
     conv2d_7 (Conv2D)           (None, 8, 8, 128)         147584    
                                                                     
     max_pooling2d_3 (MaxPooling  (None, 4, 4, 128)        0         
     2D)                                                             
                                                                     
     flatten_1 (Flatten)         (None, 2048)              0         
                                                                     
     dropout_5 (Dropout)         (None, 2048)              0         
                                                                     
     dense_2 (Dense)             (None, 1024)              2098176   
                                                                     
     dropout_6 (Dropout)         (None, 1024)              0         
                                                                     
     dense_3 (Dense)             (None, 512)               524800    
                                                                     
     dropout_7 (Dropout)         (None, 512)               0         
                                                                     
     dense_4 (Dense)             (None, 10)                5130      
                                                                     
    =================================================================
    Total params: 2,915,114
    Trainable params: 2,915,114
    Non-trainable params: 0
    _________________________________________________________________
    /usr/local/lib/python3.10/dist-packages/keras/optimizers/legacy/gradient_descent.py:114: UserWarning: The `lr` argument is deprecated, use
      super().__init__(name, **kwargs)
    None
    Epoch 1/5
    1563/1563 [==============================] - 342s 218ms/step - loss: 1.9313 - accuracy: 0.2843 - val_loss: 1.7142 - val_accuracy: 0.3886
    Epoch 2/5
    1563/1563 [==============================] - 345s 221ms/step - loss: 1.5361 - accuracy: 0.4434 - val_loss: 1.4258 - val_accuracy: 0.4928
    Epoch 3/5
```

```
import numpy
# 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)
```

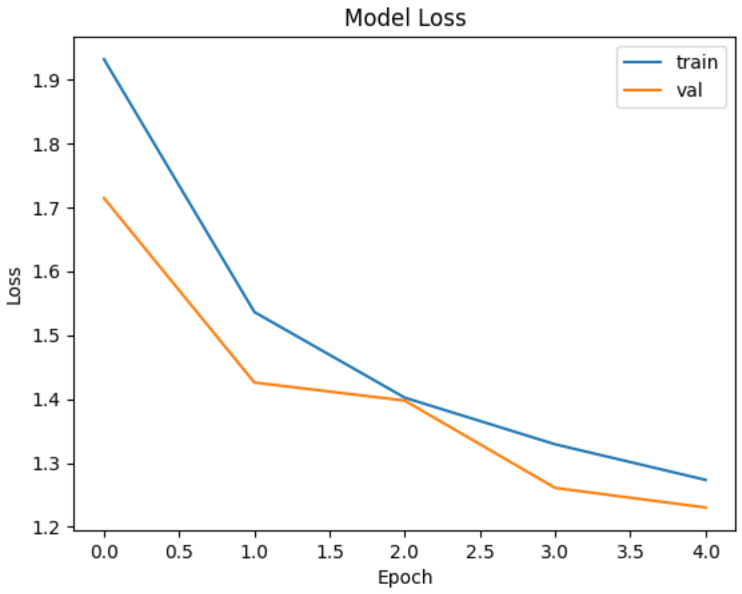
```
    1/1 [==============================] - 0s 202ms/step
    Predicted labels: [3 1 8 0]
    Actual labels:    [3 8 8 0]
```

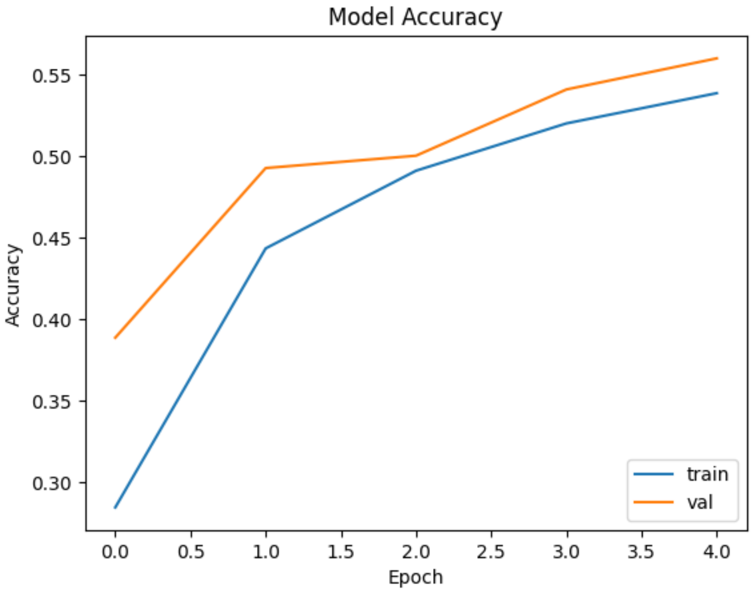
```
import matplotlib.pyplot as plt

# 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()
```


Model Loss
train
val
Loss
Epoch


Model Accuracy
Accuracy
train
val
Epoch