Neural Networks & Deep Learning

Assignment-3

GAYATHRI KESHAMONI

STUDENT ID: 700742488

GITHUB LINK: https://github.com/GayathriKeshamoni/Neural-Assignment3--Gayathri-Keshamoni--700742488

VIDEO LINK: https://youtu.be/EuWezwCS47s

LessonOverview:

Inthislesson,wearegoingtodiscussImageclassificationwithCNN.

UseCaseDescription:

ImageClassificationwithCNN

1. Trainingthemodel
2. Evaluatingthemodel

Programmingelements:

1. About CNN
2. HyperparametersofCNN
3. ImageclassificationwithCNN

Inclassprogramming:

1. Followtheinstructionbelowandthenreporthowtheperformancechanged.(applyallatonce)

- Convolutionalinputlayer,32featuremapswitha sizeof3×3 andarectifieractivationfunction.
- Dropoutlayerat20%.
- Convolutionallayer,32featuremaps withasizeof3×3andarectifieractivationfunction.

- MaxPoollayerwithsize2×2.
- Convolutionallayer,64featuremapswithasizeof3×3andarectifieractivationfunction.
- Dropoutlayerat20%.
- Convolutionallayer,64featuremaps withasizeof3×3andarectifieractivationfunction.
- MaxPoollayerwithsize2×2.
- Convolutionallayer,128featuremapswithasizeof3×3andarectifieractivationfunction.
- Dropoutlayerat20%.
- Convolutionallayer,128featuremapswithasizeof3×3and arectifieractivationfunction.
- MaxPoollayerwithsize2×2.
- Flattenlayer.
- Dropoutlayerat20%.
- Fullyconnectedlayer with1024 unitsandarectifieractivationfunction.
- Dropoutlayerat20%.
- Fullyconnectedlayer with512unitsandarectifieractivation function.
- Dropoutlayerat20%.
- Fullyconnectedoutputlayerwith10unitsanda Softmaxactivation function

Didtheperformance change?

2. Predictthefirst4imagesofthetestdatausingtheabovemodel.Then,comparewitht heactuallabelforthose4 images to check whether or not the model has predicted correctly.
3. VisualizeLossandAccuracyusingthehistoryobject.

Solution:

- These are the output & result for the following:

In [5]:
```
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
```

In [6]:
```
np.random.seed(7)
```

In [7]:
```
(X_train, y_train), (X_test, y_test) = cifar10.load_data()
```

In [8]:
```
## 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
```

In [9]:
```
## 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]
```

In [10]:
```
## 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))
```

- Imported all the required packages.
- seed() method is used to initialize the random number generator.
- Loaded the data.
- Normalised the inputs in a range .
- Using np_utils.to_categorical(), converts a class vector (integers) to binary class matrix.
- Created models using Sequential().

In [10]:
```
## 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), 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'))
```

In [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
```

- SGD is Stochastic gradient descent. SGD() to find the model parameters that correspond to the best fit between predicted and actual outputs.

In [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
```

In [12]:
```
## Compile model
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_accur
acy: 0.4563
Epoch 2/5
1563/1563 [==============================] - 10s 6ms/step - loss: 1.3675 - accuracy: 0.5117 - val_loss: 1.2470 - val_accur
acy: 0.5551
Epoch 3/5
1563/1563 [==============================] - 10s 6ms/step - loss: 1.2071 - accuracy: 0.5716 - val_loss: 1.1232 - val_accur
acy: 0.6047
Epoch 4/5
1563/1563 [==============================] - 10s 7ms/step - loss: 1.0855 - accuracy: 0.6136 - val_loss: 1.1554 - val_accur
acy: 0.5928
Epoch 5/5
1563/1563 [==============================] - 10s 7ms/step - loss: 0.9709 - accuracy: 0.6583 - val_loss: 0.9986 - val_accur
acy: 0.6550
```

Out[12]: <keras.callbacks.History at 0x7f689d6d65e0>

- fit() method will fit the model to the input training instances.

In [13]:
```
## Evaluate the model
scores = model.evaluate(X_test, y_test, verbose=0)
print("Accuracy: %.2f%%" % (scores[1]*100))
```

```
Accuracy: 65.50%
```

- Evaluating the model by trying to calculating the accuracy.

In [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))
```

- Repeating the above process by changing the values and finding the accuracy value.

```
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)

 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_accur
acy: 0.4168
Epoch 2/5
1563/1563 [==============================] - 13s 9ms/step - loss: 1.5375 - accuracy: 0.4379 - val_loss: 1.4261 - val_accur
acy: 0.4795
Epoch 3/5
1563/1563 [==============================] - 13s 9ms/step - loss: 1.3979 - accuracy: 0.4918 - val_loss: 1.3406 - val_accur
acy: 0.5164
Epoch 4/5
1563/1563 [==============================] - 13s 8ms/step - loss: 1.3128 - accuracy: 0.5217 - val_loss: 1.2901 - val_accur
acy: 0.5367
Epoch 5/5
1563/1563 [==============================] - 13s 9ms/step - loss: 1.2504 - accuracy: 0.5459 - val_loss: 1.1804 - val_accur
acy: 0.5735
Accuracy: 57.35%
```

In [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]
```

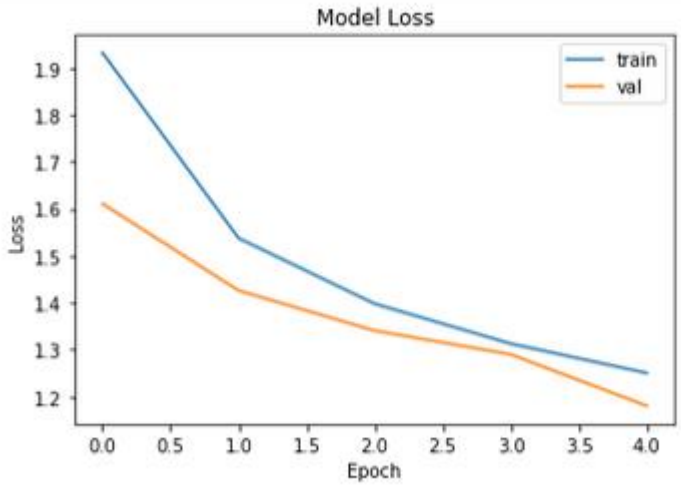
- predict() will perform predictions on the testing instances, based on the learned parameters during fit.
- Here tried to predict first 4 images of the test data and then convert them into class labels.

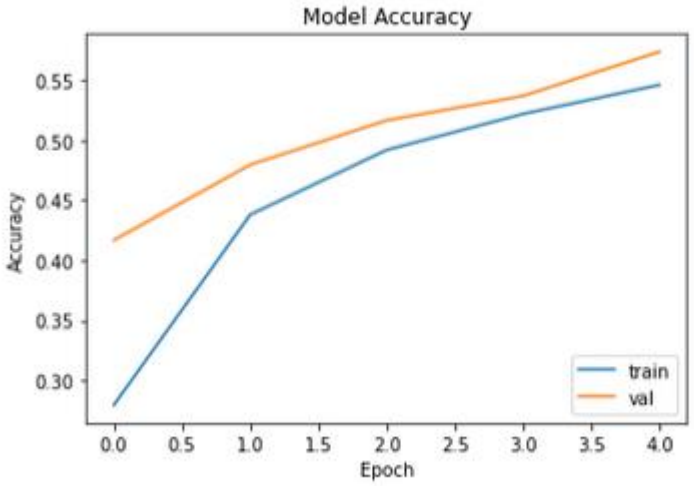
In [19]:
```
import matplotlib.pyplot as plt

# 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()
```

- Using the matplot, tried ploting the training and validation loss also plotted the accuracy of the same.


Model Loss
train
val
Loss
1.9
1.8
1.7
1.6
1.5
1.4
1.3
1.2
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Epoch


Model Accuracy
Accuracy
0.55
0.50
0.45
0.40
0.35
0.30
train
val
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Epoch