Annapoornima S

225229101 ¶

Lab 14: Classification of CIFAR-10 data with Data Augmentation

Step-1:

In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import keras
from keras.datasets import cifar10
from keras.utils import to_categorical
from keras.backend import categorical_crossentropy
from keras.models import Sequential
from keras.layers import Dense, Activation, Dropout, Flatten, Conv2D, MaxPooling2D
from keras.optimizers import RMSprop
from keras.preprocessing.image import ImageDataGenerator
```

In [2]:

```
import warnings
warnings.filterwarnings("ignore")
```

Step-2:

In [3]:

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

In [4]:

```
print('Shape of X_train is {}'.format(X_train.shape))
print('Shape of X_test is {}'.format(X_test.shape))
print('Shape of y_train is {}'.format(y_train.shape))
print('Shape of y_test is {}'.format(y_test.shape))
```

```
Shape of X_train is (50000, 32, 32, 3)
Shape of X_test is (10000, 32, 32, 3)
Shape of y_train is (50000, 1)
Shape of y_test is (10000, 1)
```

Step-3:

In [5]:

```
num_classes =10
y_train = to_categorical(y_train, num_classes)
y_test = to_categorical(y_test, num_classes)
```

Step-4:

In [6]:

```
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train /= 255
X_test /= 255
```

In [7]:

```
print('Shape of one sample of X_train is {}'.format(X_train[37].shape))
print('Shape of one sample of y_train is {}'.format(y_train[37].shape))
```

```
Shape of one sample of X_train is (32, 32, 3)
Shape of one sample of y_train is (10,)
```

In [8]:

```
X_train[37]
```

Out[8]:

```
array([[[0.37254903, 0.4117647 , 0.49803922],
         [0.34509805, 0.38039216, 0.47058824],
         [0.3372549 , 0.3764706 , 0.4627451 ],
         ...,
         [0.39607844, 0.45490196, 0.5647059 ],
         [0.35686275, 0.42352942, 0.53333336],
         [0.4117647 , 0.4862745 , 0.6156863 ]],

        [[0.32156864, 0.3529412 , 0.43137255],
         [0.29411766, 0.3254902 , 0.40784314],
         [0.29803923, 0.32941177, 0.40784314],
         ...,
         [0.36862746, 0.4       , 0.48235294],
         [0.2       , 0.23921569, 0.3137255 ],
         [0.32941177, 0.38039216, 0.47843137]],

        [[0.3019608 , 0.33333334, 0.40392157],
         [0.2901961 , 0.31764707, 0.38431373],
         [0.2784314 , 0.30588236, 0.37254903],
         ...,
         [0.2784314 , 0.2901961 , 0.3372549 ],
         [0.18431373, 0.20392157, 0.24705882],
         [0.34509805, 0.37254903, 0.43529412]],

        ...,

        [[0.38039216, 0.37254903, 0.28235295],
         [0.36078432, 0.36078432, 0.27058825],
         [0.38039216, 0.3647059 , 0.27450982],
         ...,
         [0.3372549 , 0.35686275, 0.25490198],
         [0.36862746, 0.38039216, 0.28235295],
         [0.3529412 , 0.38039216, 0.2784314 ]],

        [[0.37254903, 0.3529412 , 0.25490198],
         [0.32941177, 0.3372549 , 0.23137255],
         [0.34901962, 0.34901962, 0.24313726],
         ...,
         [0.3764706 , 0.38039216, 0.29803923],
         [0.4       , 0.3764706 , 0.3019608 ],
         [0.38039216, 0.36862746, 0.28627452]],

        [[0.35686275, 0.32941177, 0.24705882],
         [0.3254902 , 0.31764707, 0.22352941],
         [0.32156864, 0.31764707, 0.21568628],
         ...,
         [0.39215687, 0.3764706 , 0.30588236],
         [0.4117647 , 0.38039216, 0.3137255 ],
         [0.42352942, 0.4       , 0.3254902 ]]], dtype=float32)
```

In [9]:

```
y_train[37]
```

Out[9]:

```
array([0., 0., 0., 0., 0., 0., 0., 1., 0., 0.], dtype=float32)
```

Step-5:

In [10]:

```
model = Sequential()
model.add(Conv2D(32, (5,5), strides=(2,2), padding='same', input_shape=X_train.shape[1:]
model.add(Activation('relu'))
model.add(Conv2D(32, (5,5), strides=(2,2)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(512))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes))
model.add(Activation('softmax'))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 16, 16, 32)        2432

 activation (Activation)     (None, 16, 16, 32)        0

 conv2d_1 (Conv2D)           (None, 6, 6, 32)          25632

 activation_1 (Activation)   (None, 6, 6, 32)          0

 max_pooling2d (MaxPooling2  (None, 3, 3, 32)          0
 D)

 dropout (Dropout)           (None, 3, 3, 32)          0

 flatten (Flatten)           (None, 288)               0

 dense (Dense)               (None, 512)               147968

 activation_2 (Activation)   (None, 512)               0

 dropout_1 (Dropout)         (None, 512)               0

 dense_1 (Dense)             (None, 10)                5130

 activation_3 (Activation)   (None, 10)                0

=================================================================
Total params: 181162 (707.66 KB)
Trainable params: 181162 (707.66 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Step-6:

In [11]:

```
# Load and preprocess the CIFAR-10 dataset
(X_train, y_train), (X_test, y_test) = cifar10.load_data()
```

In [12]:

```
# Convert class vectors to binary class matrices (one-hot encoding)
num_classes = 10
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)
```

In [13]:

```
# Build the neural network model
model = Sequential()
model.add(Conv2D(32, (3, 3), padding='same', input_shape=X_train.shape[1:]))
model.add(Activation('relu'))
model.add(Conv2D(32, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(64, (3, 3), padding='same'))
model.add(Activation('relu'))
model.add(Conv2D(64, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(512))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes))
model.add(Activation('softmax'))
```

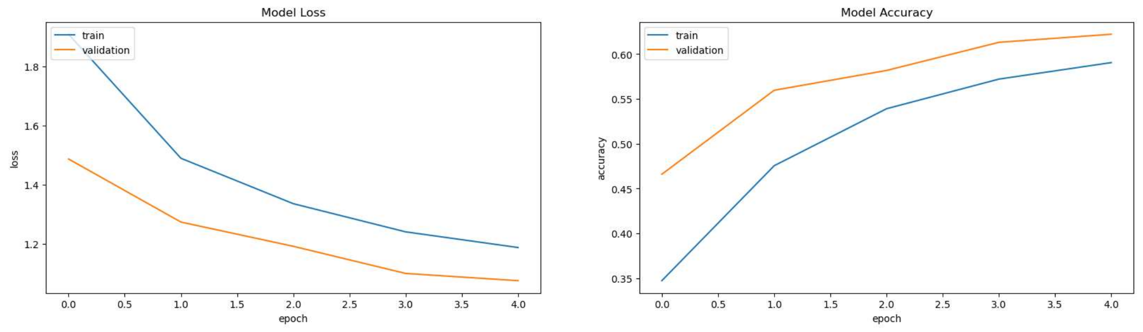
In [14]:

```
# Compile the model
opt = RMSprop(learning_rate=0.0005)
model.compile(loss='categorical_crossentropy', optimizer=opt, metrics=['accuracy'])

# Train the model
batch_size = 32
epochs = 5
history = model.fit(X_train, y_train, batch_size=batch_size, epochs=epochs, verbose=2, v

# Evaluate the model on the test data
score = model.evaluate(X_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Epoch 1/5
1250/1250 - 129s - loss: 1.9097 - accuracy: 0.3473 - val_loss: 1.4871 - v
al_accuracy: 0.4660 - 129s/epoch - 103ms/step
Epoch 2/5
1250/1250 - 168s - loss: 1.4896 - accuracy: 0.4755 - val_loss: 1.2735 - v
al_accuracy: 0.5597 - 168s/epoch - 134ms/step
Epoch 3/5
1250/1250 - 171s - loss: 1.3358 - accuracy: 0.5391 - val_loss: 1.1914 - v
al_accuracy: 0.5818 - 171s/epoch - 137ms/step
Epoch 4/5
1250/1250 - 149s - loss: 1.2404 - accuracy: 0.5721 - val_loss: 1.0995 - v
al_accuracy: 0.6132 - 149s/epoch - 119ms/step
Epoch 5/5
1250/1250 - 155s - loss: 1.1873 - accuracy: 0.5906 - val_loss: 1.0750 - v
al_accuracy: 0.6222 - 155s/epoch - 124ms/step
Test loss: 1.0911281108856201
Test accuracy: 0.6169000267982483
```

In [15]:

```
fig = plt.figure(figsize=(20, 5))
fig.add_subplot(1,2,1)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper left')
fig.add_subplot(1,2,2)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper left')
plt.show()
```



Step-7:

In [16]:

```
datagen = ImageDataGenerator(featurewise_center=False,
samplewise_center=False,

featurewise_std_normalization=False,
samplewise_std_normalization=False,
zca_whitening=False,
rotation_range=0,
width_shift_range=0.1,
height_shift_range=0.1,
horizontal_flip=True,
vertical_flip=False)

datagen.fit(X_train)
```

In [17]:

```
import tensorflow as tf
batch_size = 32
opt = tf.keras.optimizers.legacy.RMSprop(lr=0.0005, decay=1e-6)
model.compile(loss='categorical_crossentropy', optimizer=opt, metrics=['accuracy'])
```

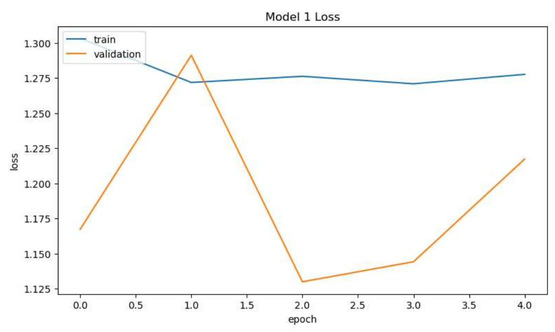
In [18]:

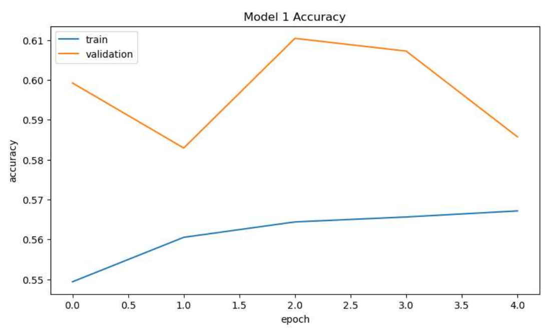
```
history1 = model.fit_generator(datagen.flow(X_train, y_train, batch_size=32),
steps_per_epoch=X_train.shape[0] // batch_size,
epochs=5,

validation_data=(X_test, y_test))
```

```
Epoch 1/5
1562/1562 [==============================] - 140s 88ms/step - loss: 1.303
3 - accuracy: 0.5494 - val_loss: 1.1673 - val_accuracy: 0.5992
Epoch 2/5
1562/1562 [==============================] - 147s 94ms/step - loss: 1.271
9 - accuracy: 0.5605 - val_loss: 1.2913 - val_accuracy: 0.5829
Epoch 3/5
1562/1562 [==============================] - 136s 87ms/step - loss: 1.276
3 - accuracy: 0.5644 - val_loss: 1.1299 - val_accuracy: 0.6104
Epoch 4/5
1562/1562 [==============================] - 137s 88ms/step - loss: 1.271
0 - accuracy: 0.5656 - val_loss: 1.1442 - val_accuracy: 0.6072
Epoch 5/5
1562/1562 [==============================] - 138s 88ms/step - loss: 1.277
7 - accuracy: 0.5671 - val_loss: 1.2174 - val_accuracy: 0.5857
```

In [19]:

```
fig = plt.figure(figsize=(20, 5))
fig.add_subplot(1,2,1)
plt.plot(history1.history['loss'])
plt.plot(history1.history['val_loss'])
plt.title('Model 1 Loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper left')
fig.add_subplot(1,2,2)
plt.plot(history1.history['accuracy'])
plt.plot(history1.history['val_accuracy'])
plt.title('Model 1 Accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper left')
plt.show()
```

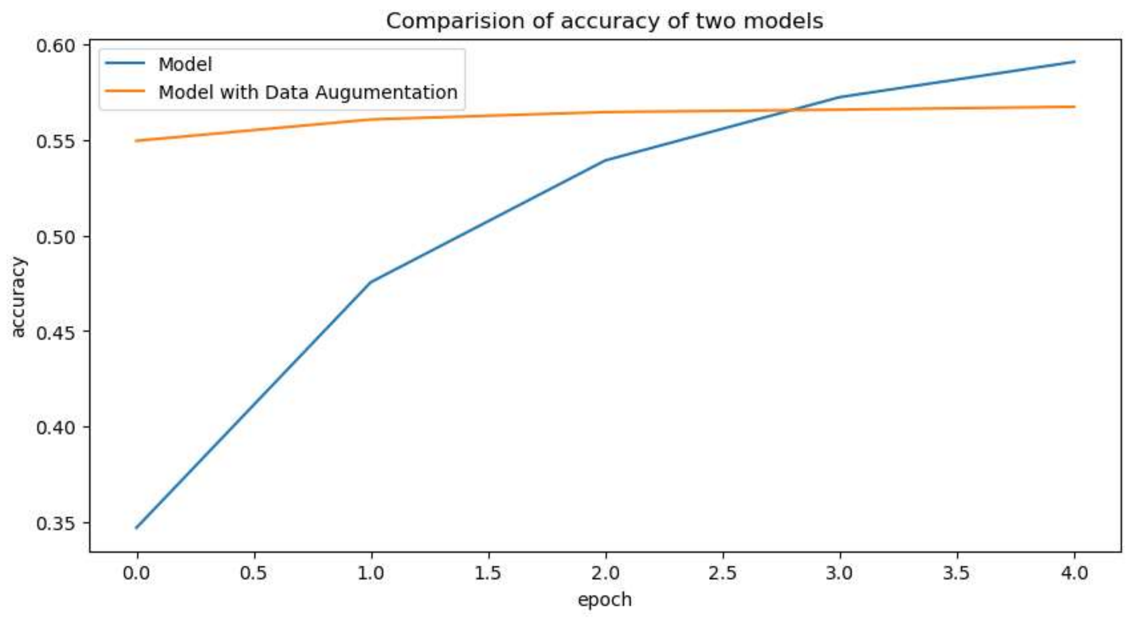




Step-8:

In [20]:

```
plt.figure(figsize=(10, 5))
plt.plot(history.history['accuracy'])
plt.plot(history1.history['accuracy'])
plt.title('Comparision of accuracy of two models')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['Model', 'Model with Data Augumentation'], loc='upper left')
plt.show()
```



Step-9:

In [21]:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, Activation, MaxPooling2D, Flatten, Dense

# Define the input shape based on your data
input_shape = (128, 128, 3)  # Replace with your actual input dimensions

model2 = Sequential()
model2.add(Conv2D(32, (5, 5), strides=(1, 1), padding='same', activation='relu', input_s
model2.add(Conv2D(32, (5, 5), strides=(1, 1)))
model2.add(Activation('relu'))
model2.add(MaxPooling2D(pool_size=(2, 2)))
model2.add(Conv2D(32, (5, 5), strides=(1, 1), padding='same', activation='relu'))
model2.add(Conv2D(32, (5, 5), strides=(1, 1)))
model2.add(Activation('relu'))
model2.add(MaxPooling2D(pool_size=(2, 2)))

# Flatten layer should produce an output that matches the input shape of the Dense layer
model2.add(Flatten())
model2.add(Dense(512, activation='relu'))  # Updated to specify activation here
model2.add(Dense(num_classes, activation='softmax'))  # Updated to specify activation he
model2.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_6 (Conv2D)           (None, 128, 128, 32)      2432      
                                                                 
 conv2d_7 (Conv2D)           (None, 124, 124, 32)      25632     
                                                                 
 activation_10 (Activation)  (None, 124, 124, 32)      0         
                                                                 
 max_pooling2d_3 (MaxPoolin  (None, 62, 62, 32)        0         
 g2D)                                                            
                                                                 
 conv2d_8 (Conv2D)           (None, 62, 62, 32)        25632     
                                                                 
 conv2d_9 (Conv2D)           (None, 58, 58, 32)        25632     
                                                                 
 activation_11 (Activation)  (None, 58, 58, 32)        0         
                                                                 
 max_pooling2d_4 (MaxPoolin  (None, 29, 29, 32)        0         
 g2D)                                                            
                                                                 
 flatten_2 (Flatten)         (None, 26912)             0         
                                                                 
 dense_4 (Dense)             (None, 512)               13779456  
                                                                 
 dense_5 (Dense)             (None, 10)                5130      
                                                                 
=================================================================
Total params: 13863914 (52.89 MB)
Trainable params: 13863914 (52.89 MB)
Non-trainable params: 0 (0.00 Byte)
```

In [22]:

```
import tensorflow as tf  # Import TensorFlow's Keras submodule
batch_size = 32
opt = tf.keras.optimizers.legacy.RMSprop(lr=0.0005, decay=1e-6)  # Use the legacy optimi
model2.compile(loss='categorical_crossentropy', optimizer=opt, metrics=['accuracy'])
```

In [ ]:

```
```