NAME: VEENA ROLL NO:225229145 PDL Lab9: Image Classification using CNN for CIFAR-10 data

```
#1 import library
from __future__ import print_function
import keras
from keras.datasets import cifar10
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, Flatten, Conv2D, MaxPooling2D
from keras.utils import np_utils
from tensorflow.keras.optimizers import RMSprop
import matplotlib.pyplot as plt
%matplotlib inline
```

```
#2.Load the data and print the shape of training and test samples
(x_train, Y_train), (x_test, y_test) = cifar10.load_data()
print('x_train shape:', x_train.shape)
print(x_train.shape[0], 'train sampple')
print(x_test.shape[0], 'test samples')
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170498071/170498071 [==============================] - 2s 0us/step
x_train shape: (50000, 32, 32, 3)
50000 train sampple
```

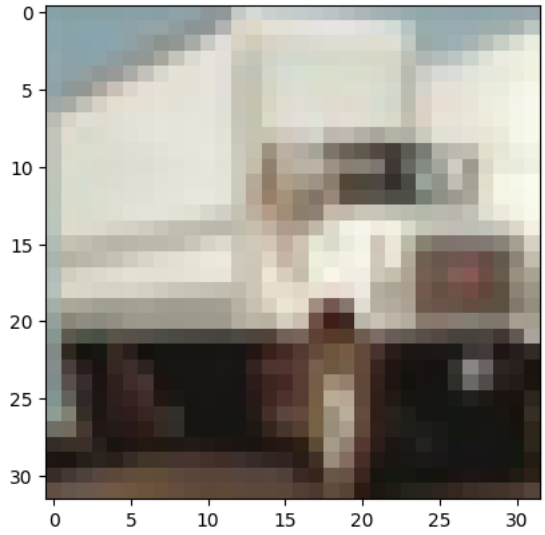


```
x_train[444].shape
```

```
(32, 32, 3)
```

```
#4.Display one image using imshow()function
print(Y_train[444])
plt.imshow(x_train[444])
```

```
[9]
<matplotlib.image.AxesImage at 0x7a48ddeddc90>
```




```
#5. Convert y_train and y_test into categorical values
num_classes = 10
Y_train = keras.utils.to_categorical(Y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)
```

```
Y_train[444]
```

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

```
#6. convert train data into float and scale
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /=255
x_test /=255
```

```
#7.Build your frist CNN
from tensorflow.keras.models import Sequential
```

```
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dropout, Flatten, Dense

INPUT_SHAPE = (32, 32, 3)

model = Sequential()
model.add(Conv2D(filters=32, kernel_size=(5, 5), strides=(2, 2), activation='relu', padding='same', input_shape=INPUT_SHAPE))
model.add(Conv2D(filters=32, kernel_size=(5, 5), strides=(2, 2), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(rate=0.25))
model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(Dropout(rate=0.5))
model.add(Dense(10, activation='softmax'))

model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_1 (Conv2D)           (None, 16, 16, 32)        2432

 conv2d_2 (Conv2D)           (None, 8, 8, 32)          25632

 max_pooling2d (MaxPooling2D  (None, 4, 4, 32)         0
 )

 flatten (Flatten)           (None, 512)               0

 dense (Dense)               (None, 512)               262656

 dropout_1 (Dropout)         (None, 512)               0

 dense_1 (Dense)             (None, 10)                5130

=================================================================
Total params: 295,850
Trainable params: 295,850
Non-trainable params: 0
_________________________________________________________________
```



```
#8.print summary and verify the configuration
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_1 (Conv2D)           (None, 16, 16, 32)        2432

 conv2d_2 (Conv2D)           (None, 8, 8, 32)          25632

 max_pooling2d (MaxPooling2D  (None, 4, 4, 32)         0
 )

 dropout (Dropout)           (None, 4, 4, 32)          0

 flatten (Flatten)           (None, 512)               0

 dense (Dense)               (None, 512)               262656

 dropout_1 (Dropout)         (None, 512)               0

 dense_1 (Dense)             (None, 10)                5130

=================================================================
Total params: 295,850
Trainable params: 295,850
Non-trainable params: 0
_________________________________________________________________
```

```
#  the model architecture
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(128, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dense(10, activation='softmax')
])
```

```
])

# Compile the model using legacy RMSprop optimizer
model.compile(optimizer=tf.keras.optimizers.legacy.RMSprop(learning_rate=0.0005, decay=1e-6),
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# Fit and validate the model
history = model.fit(x_train, y_train, batch_size=32, epochs=15, verbose=2, validation_split=0.2)
```

```
Epoch 1/15
1250/1250 - 68s - loss: 1.6233 - accuracy: 0.4098 - val_loss: 1.4626 - val_accuracy: 0.4709 - 68s/epoch - 55ms/step
Epoch 2/15
1250/1250 - 65s - loss: 1.2318 - accuracy: 0.5630 - val_loss: 1.1520 - val_accuracy: 0.5990 - 65s/epoch - 52ms/step
Epoch 3/15
1250/1250 - 65s - loss: 1.0417 - accuracy: 0.6344 - val_loss: 1.0217 - val_accuracy: 0.6439 - 65s/epoch - 52ms/step
Epoch 4/15
1250/1250 - 65s - loss: 0.9138 - accuracy: 0.6828 - val_loss: 0.9591 - val_accuracy: 0.6722 - 65s/epoch - 52ms/step
Epoch 5/15
1250/1250 - 65s - loss: 0.8120 - accuracy: 0.7165 - val_loss: 0.8856 - val_accuracy: 0.6965 - 65s/epoch - 52ms/step
Epoch 6/15
1250/1250 - 65s - loss: 0.7277 - accuracy: 0.7465 - val_loss: 0.9489 - val_accuracy: 0.6815 - 65s/epoch - 52ms/step
Epoch 7/15
1250/1250 - 65s - loss: 0.6531 - accuracy: 0.7750 - val_loss: 0.8757 - val_accuracy: 0.7082 - 65s/epoch - 52ms/step
Epoch 8/15
1250/1250 - 65s - loss: 0.5856 - accuracy: 0.7956 - val_loss: 0.9053 - val_accuracy: 0.7086 - 65s/epoch - 52ms/step
Epoch 9/15
1250/1250 - 64s - loss: 0.5256 - accuracy: 0.8174 - val_loss: 0.8797 - val_accuracy: 0.7191 - 64s/epoch - 51ms/step
```



```
val_accuracy: 0.7239 - 65s/epoch - 52ms/step
1250/1250 - 65s - loss: 0.4152 - accuracy: 0.8546 - val_loss: 0.9368 - val_accuracy: 0.7182 - 65s/epoch - 52ms/step
Epoch 12/15
1250/1250 - 66s - loss: 0.3706 - accuracy: 0.8718 - val_loss: 0.9479 - val_accuracy: 0.7298 - 66s/epoch - 53ms/step
Epoch 13/15
1250/1250 - 65s - loss: 0.3218 - accuracy: 0.8900 - val_loss: 1.0388 - val_accuracy: 0.7202 - 65s/epoch - 52ms/step
Epoch 14/15
1250/1250 - 65s - loss: 0.2813 - accuracy: 0.9026 - val_loss: 1.1034 - val_accuracy: 0.7225 - 65s/epoch - 52ms/step
Epoch 15/15
1250/1250 - 64s - loss: 0.2464 - accuracy: 0.9155 - val_loss: 1.1972 - val_accuracy: 0.7094 - 64s/epoch - 51ms/step
```

```
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Test loss: 1.2195886373519897
Test accuracy: 0.703499972820282
```

Part-2. Model Improvements

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

INPUT_SHAPE = (32, 32, 3)
model1 = Sequential()

# Add layers to the model
model1.add(Conv2D(filters=8, kernel_size=(5, 5), strides=(1, 1), activation='relu', padding='same', input_shape=INPUT_SHAPE))
model1.add(Conv2D(filters=8, kernel_size=(5, 5), strides=(1, 1), activation='relu', padding='same'))
model1.add(MaxPooling2D(pool_size=(2, 2)))
model1.add(Conv2D(filters=8, kernel_size=(5, 5), strides=(1, 1), activation='relu', padding='same'))
model1.add(Conv2D(filters=8, kernel_size=(5, 5), strides=(1, 1), activation='relu', padding='same'))
model1.add(MaxPooling2D(pool_size=(2, 2)))
model1.add(Flatten())
model1.add(Dense(512, activation='relu'))
model1.add(Dense(10, activation='softmax'))

# Build the model
model1.build(input_shape=(None, 32, 32, 3))

model1.summary()
```

```
Model: "sequential_7"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_19 (Conv2D)          (None, 32, 32, 8)         608

 conv2d_20 (Conv2D)          (None, 32, 32, 8)         1608

 max_pooling2d_10 (MaxPoolin  (None, 16, 16, 8)        0
```

```
g2D)

conv2d_21 (Conv2D)          (None, 16, 16, 8)         1608

conv2d_22 (Conv2D)          (None, 16, 16, 8)         1608

max_pooling2d_11 (MaxPoolin  (None, 8, 8, 8)          0
g2D)

flatten_5 (Flatten)         (None, 512)               0

dense_10 (Dense)            (None, 512)               262656

dense_11 (Dense)            (None, 10)                5130

=================================================================
Total params: 273,218
Trainable params: 273,218
Non-trainable params: 0
_________________________________________________________________
```

```
model1.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
history1=model1.fit(x_train,y_train,batch_size=16,epochs=5,verbose=2,validation_split=0.2)
```

```
Epoch 1/5
2500/2500 - 132s - loss: 1.5579 - accuracy: 0.4301 - val_loss: 1.3218 - val_accuracy: 0.5276 - 132s/epoch - 53ms/step
Epoch 2/5
2500/2500 - 131s - loss: 1.2370 - accuracy: 0.5561 - val_loss: 1.1927 - val_accuracy: 0.5730 - 131s/epoch - 52ms/step
```



```
val_accuracy: 0.6158 - 128s/epoch - 51ms/step
2500/2500 - 131s - loss: 0.9543 - accuracy: 0.6586 - val_loss: 1.0808 - val_accuracy: 0.6233 - 131s/epoch - 53ms/step
Epoch 5/5
2500/2500 - 128s - loss: 0.8482 - accuracy: 0.6981 - val_loss: 1.2157 - val_accuracy: 0.6063 - 128s/epoch - 51ms/step
```

```
score = model1.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Test loss: 1.2265422344207764
Test accuracy: 0.6022999882698059
```

✓ 11s completed at 10:16 PM