tuzvfmyfn

January 30, 2025

[1]:
```
import keras

from keras.datasets import cifar10
from tensorflow.keras.preprocessing.image import ImageDataGenerator

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

import os

import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix, classification_report
import itertools

%matplotlib inline

#Loading required header files
```

[2]:
```
# Load the CIFAR-10 dataset
(x_train, y_train), (x_test, y_test) = cifar10.load_data()

# Print dataset shapes
print("x_train shape:", x_train.shape)
print("y_train shape:", y_train.shape)
print("x_test shape:", x_test.shape)
print("y_test shape:", y_test.shape)

# Print data types
print("x_train dtype:", x_train.dtype)
print("y_train dtype:", y_train.dtype)
```

```
x_train shape: (50000, 32, 32, 3)
y_train shape: (50000, 1)
x_test shape: (10000, 32, 32, 3)
y_test shape: (10000, 1)
```

```
x_train dtype: uint8
y_train dtype: uint8
```

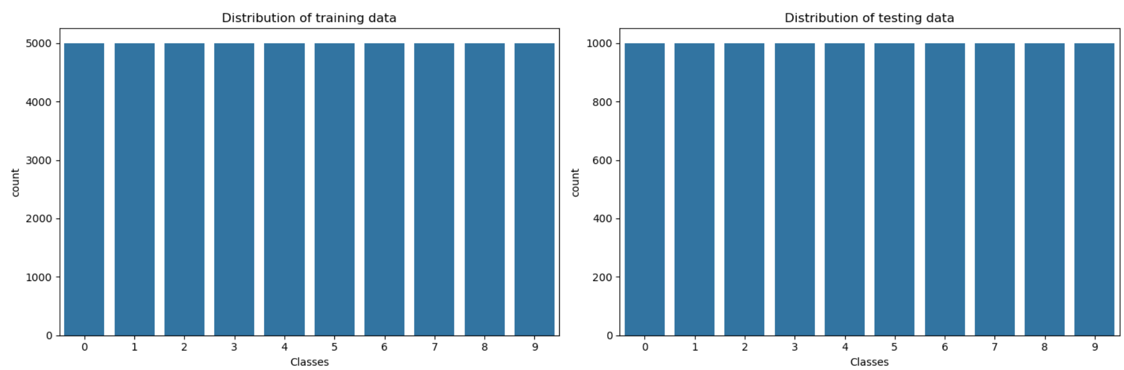
[3]:
```python
num_classes = 10

data_augmentation = False
```

[4]:
```python
#checking distribution
fig, axs = plt.subplots(1,2,figsize=(15,5))
sns.countplot(x=y_train.ravel(), ax = axs[0],legend=False)
axs[0].set_title("Distribution of training data")
axs[0].set_xlabel("Classes")

sns.countplot(x=y_test.ravel(), ax = axs[1],legend=False)
axs[1].set_title("Distribution of testing data")
axs[1].set_xlabel("Classes")

plt.tight_layout()
plt.show()
```



[5]:
```python
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255 #since picxcel max value is 255 dividing by it will normalize␣
↪the values
x_test /= 255
#one hot encoding the traget cvariablr
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils. to_categorical(y_test, num_classes)
```

[6]:
```python
model = Sequential()

model.add(Conv2D(filters = 32,kernel_size = (3,3), padding =␣
↪'same',activation="relu", input_shape=x_train.shape[1:]))
model.add(Conv2D(filters = 32,kernel_size = (3,3),activation="relu"))
```

```
model.add(MaxPooling2D(pool_size =(2,2)))
model.add(Dropout(.25))

model.add(Conv2D(filters = 64, kernel_size = (3,3), padding =␣
 ↪'same',activation="relu"))
model.add(Conv2D(filters = 64, kernel_size = (3,3), activation="relu"))
model.add(MaxPooling2D(pool_size = (2,2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(512,activation="relu"))


model.add(Dropout(0.5))

model.add(Dense(num_classes,activation="softmax" ))

model.summary()
```

```
Model: "sequential"

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

 conv2d_1 (Conv2D)           (None, 30, 30, 32)        9248

 max_pooling2d (MaxPooling2D  (None, 15, 15, 32)       0
 )

 dropout (Dropout)           (None, 15, 15, 32)        0

 conv2d_2 (Conv2D)           (None, 15, 15, 64)        18496

 conv2d_3 (Conv2D)           (None, 13, 13, 64)        36928

 max_pooling2d_1 (MaxPooling  (None, 6, 6, 64)         0
 2D)

 dropout_1 (Dropout)         (None, 6, 6, 64)          0

 flatten (Flatten)           (None, 2304)              0

 dense (Dense)               (None, 512)               1180160

 dropout_2 (Dropout)         (None, 512)               0
```

```
 dense_1 (Dense)             (None, 10)                5130

=================================================================
Total params: 1,250,858
Trainable params: 1,250,858
Non-trainable params: 0
_________________________________________________________________
```

[7]:
```
opt = keras.optimizers.RMSprop(learning_rate = 0.0001, decay = 0.000001)

model.compile(loss = "categorical_crossentropy", optimizer = opt,
              metrics = ['accuracy'])
```

[8]:
```
# Set parameters
batch_size = 64  # Adjusted batch size for better CPU performance
epochs = 100

# Check if using data augmentation
if not data_augmentation:
    print("Not using data augmentation.")
    history = model.fit(x_train, y_train, batch_size=batch_size, epochs=epochs,
                        validation_data=(x_test, y_test), workers=4,␣
 ↪use_multiprocessing=True) #using multiple core for better CPU performance
else:
    print("Using realtime data augmentation.")

    # Initialize ImageDataGenerator with augmentation settings
    datagen = ImageDataGenerator(width_shift_range=0.1, height_shift_range=0.1,␣
 ↪horizontal_flip=True)

    # Fit the generator to the training data (not necessary if augmentation␣
 ↪parameters are fixed)
    datagen.fit(x_train)

    # Train the model using the augmented data
    history = model.fit(datagen.flow(x_train, y_train, batch_size=batch_size,␣
 ↪epochs=epochs,
                                     validation_data=(x_test, y_test),␣
 ↪workers=4, use_multiprocessing=True))
```

```
Not using data augmentation.
Epoch 1/100
782/782 [==============================] - 133s 168ms/step - loss: 1.8923 -
accuracy: 0.3073 - val_loss: 1.6324 - val_accuracy: 0.4078
Epoch 2/100
782/782 [==============================] - 142s 182ms/step - loss: 1.6026 -
accuracy: 0.4171 - val_loss: 1.4816 - val_accuracy: 0.4726
```

```
Epoch 3/100
782/782 [==============================] - 111s 143ms/step - loss: 1.4646 -
accuracy: 0.4676 - val_loss: 1.3915 - val_accuracy: 0.4984
Epoch 4/100
782/782 [==============================] - 112s 143ms/step - loss: 1.3590 -
accuracy: 0.5117 - val_loss: 1.2822 - val_accuracy: 0.5385
Epoch 5/100
782/782 [==============================] - 113s 145ms/step - loss: 1.2894 -
accuracy: 0.5394 - val_loss: 1.2219 - val_accuracy: 0.5706
Epoch 6/100
782/782 [==============================] - 119s 152ms/step - loss: 1.2244 -
accuracy: 0.5632 - val_loss: 1.2124 - val_accuracy: 0.5665
Epoch 7/100
782/782 [==============================] - 115s 147ms/step - loss: 1.1764 -
accuracy: 0.5850 - val_loss: 1.1837 - val_accuracy: 0.5925
Epoch 8/100
782/782 [==============================] - 118s 150ms/step - loss: 1.1267 -
accuracy: 0.6010 - val_loss: 1.0791 - val_accuracy: 0.6171
Epoch 9/100
782/782 [==============================] - 151s 193ms/step - loss: 1.0889 -
accuracy: 0.6187 - val_loss: 1.0161 - val_accuracy: 0.6453
Epoch 10/100
782/782 [==============================] - 146s 185ms/step - loss: 1.0463 -
accuracy: 0.6304 - val_loss: 0.9871 - val_accuracy: 0.6540
Epoch 11/100
782/782 [==============================] - 121s 155ms/step - loss: 1.0157 -
accuracy: 0.6444 - val_loss: 0.9883 - val_accuracy: 0.6589
Epoch 12/100
782/782 [==============================] - 128s 163ms/step - loss: 0.9821 -
accuracy: 0.6553 - val_loss: 0.9578 - val_accuracy: 0.6666
Epoch 13/100
782/782 [==============================] - 130s 167ms/step - loss: 0.9512 -
accuracy: 0.6675 - val_loss: 1.0064 - val_accuracy: 0.6521
Epoch 14/100
782/782 [==============================] - 126s 161ms/step - loss: 0.9266 -
accuracy: 0.6757 - val_loss: 0.8914 - val_accuracy: 0.6899
Epoch 15/100
782/782 [==============================] - 139s 177ms/step - loss: 0.9084 -
accuracy: 0.6835 - val_loss: 0.8507 - val_accuracy: 0.6986
Epoch 16/100
782/782 [==============================] - 130s 166ms/step - loss: 0.8881 -
accuracy: 0.6889 - val_loss: 0.8638 - val_accuracy: 0.7006
Epoch 17/100
782/782 [==============================] - 125s 160ms/step - loss: 0.8635 -
accuracy: 0.6989 - val_loss: 0.8252 - val_accuracy: 0.7119
Epoch 18/100
782/782 [==============================] - 124s 159ms/step - loss: 0.8453 -
accuracy: 0.7060 - val_loss: 0.9144 - val_accuracy: 0.6839
```

```
Epoch 19/100
782/782 [==============================] - 122s 156ms/step - loss: 0.8278 -
accuracy: 0.7124 - val_loss: 0.9245 - val_accuracy: 0.6828
Epoch 20/100
782/782 [==============================] - 133s 170ms/step - loss: 0.8122 -
accuracy: 0.7161 - val_loss: 0.8040 - val_accuracy: 0.7212
Epoch 21/100
782/782 [==============================] - 129s 165ms/step - loss: 0.7994 -
accuracy: 0.7219 - val_loss: 0.7843 - val_accuracy: 0.7290
Epoch 22/100
782/782 [==============================] - 158s 202ms/step - loss: 0.7800 -
accuracy: 0.7284 - val_loss: 0.8516 - val_accuracy: 0.7137
Epoch 23/100
782/782 [==============================] - 1752s 2s/step - loss: 0.7687 -
accuracy: 0.7337 - val_loss: 0.7754 - val_accuracy: 0.7316
Epoch 24/100
782/782 [==============================] - 94s 120ms/step - loss: 0.7591 -
accuracy: 0.7360 - val_loss: 0.7600 - val_accuracy: 0.7404
Epoch 25/100
782/782 [==============================] - 98s 125ms/step - loss: 0.7503 -
accuracy: 0.7404 - val_loss: 0.7904 - val_accuracy: 0.7313
Epoch 26/100
782/782 [==============================] - 101s 129ms/step - loss: 0.7349 -
accuracy: 0.7455 - val_loss: 0.8020 - val_accuracy: 0.7269
Epoch 27/100
782/782 [==============================] - 105s 134ms/step - loss: 0.7233 -
accuracy: 0.7502 - val_loss: 0.7555 - val_accuracy: 0.7397
Epoch 28/100
782/782 [==============================] - 111s 142ms/step - loss: 0.7185 -
accuracy: 0.7530 - val_loss: 0.7126 - val_accuracy: 0.7569
Epoch 29/100
782/782 [==============================] - 1524s 2s/step - loss: 0.7124 -
accuracy: 0.7542 - val_loss: 0.7214 - val_accuracy: 0.7553
Epoch 30/100
782/782 [==============================] - 115s 147ms/step - loss: 0.7023 -
accuracy: 0.7584 - val_loss: 0.7222 - val_accuracy: 0.7520
Epoch 31/100
782/782 [==============================] - 115s 147ms/step - loss: 0.6953 -
accuracy: 0.7584 - val_loss: 0.7486 - val_accuracy: 0.7467
Epoch 32/100
782/782 [==============================] - 107s 137ms/step - loss: 0.6868 -
accuracy: 0.7635 - val_loss: 0.7302 - val_accuracy: 0.7577
Epoch 33/100
782/782 [==============================] - 117s 149ms/step - loss: 0.6791 -
accuracy: 0.7667 - val_loss: 0.7155 - val_accuracy: 0.7598
Epoch 34/100
782/782 [==============================] - 109s 139ms/step - loss: 0.6780 -
accuracy: 0.7671 - val_loss: 0.6986 - val_accuracy: 0.7634
```

```
Epoch 35/100
782/782 [==============================] - 113s 144ms/step - loss: 0.6686 -
accuracy: 0.7680 - val_loss: 0.7142 - val_accuracy: 0.7598
Epoch 36/100
782/782 [==============================] - 115s 147ms/step - loss: 0.6610 -
accuracy: 0.7733 - val_loss: 0.6881 - val_accuracy: 0.7696
Epoch 37/100
782/782 [==============================] - 117s 149ms/step - loss: 0.6652 -
accuracy: 0.7739 - val_loss: 0.6897 - val_accuracy: 0.7674
Epoch 38/100
782/782 [==============================] - 124s 158ms/step - loss: 0.6538 -
accuracy: 0.7752 - val_loss: 0.6639 - val_accuracy: 0.7718
Epoch 39/100
782/782 [==============================] - 119s 152ms/step - loss: 0.6504 -
accuracy: 0.7778 - val_loss: 0.6775 - val_accuracy: 0.7707
Epoch 40/100
782/782 [==============================] - 120s 153ms/step - loss: 0.6417 -
accuracy: 0.7808 - val_loss: 0.7556 - val_accuracy: 0.7543
Epoch 41/100
782/782 [==============================] - 118s 152ms/step - loss: 0.6367 -
accuracy: 0.7808 - val_loss: 0.6912 - val_accuracy: 0.7699
Epoch 42/100
782/782 [==============================] - 122s 156ms/step - loss: 0.6410 -
accuracy: 0.7836 - val_loss: 0.6804 - val_accuracy: 0.7712
Epoch 43/100
782/782 [==============================] - 121s 155ms/step - loss: 0.6323 -
accuracy: 0.7844 - val_loss: 0.6685 - val_accuracy: 0.7755
Epoch 44/100
782/782 [==============================] - 125s 160ms/step - loss: 0.6298 -
accuracy: 0.7836 - val_loss: 0.6438 - val_accuracy: 0.7830
Epoch 45/100
782/782 [==============================] - 128s 164ms/step - loss: 0.6234 -
accuracy: 0.7884 - val_loss: 0.6949 - val_accuracy: 0.7659
Epoch 46/100
782/782 [==============================] - 122s 157ms/step - loss: 0.6212 -
accuracy: 0.7891 - val_loss: 0.6553 - val_accuracy: 0.7802
Epoch 47/100
782/782 [==============================] - 124s 158ms/step - loss: 0.6223 -
accuracy: 0.7886 - val_loss: 0.6710 - val_accuracy: 0.7740
Epoch 48/100
782/782 [==============================] - 122s 156ms/step - loss: 0.6125 -
accuracy: 0.7928 - val_loss: 0.6352 - val_accuracy: 0.7879
Epoch 49/100
782/782 [==============================] - 125s 159ms/step - loss: 0.6119 -
accuracy: 0.7936 - val_loss: 0.6695 - val_accuracy: 0.7786
Epoch 50/100
782/782 [==============================] - 125s 160ms/step - loss: 0.6077 -
accuracy: 0.7933 - val_loss: 0.6418 - val_accuracy: 0.7834
```

```
Epoch 51/100
782/782 [==============================] - 123s 158ms/step - loss: 0.6073 -
accuracy: 0.7937 - val_loss: 0.6556 - val_accuracy: 0.7751
Epoch 52/100
782/782 [==============================] - 124s 158ms/step - loss: 0.6042 -
accuracy: 0.7962 - val_loss: 0.6415 - val_accuracy: 0.7842
Epoch 53/100
782/782 [==============================] - 128s 164ms/step - loss: 0.5959 -
accuracy: 0.8011 - val_loss: 0.6485 - val_accuracy: 0.7837
Epoch 54/100
782/782 [==============================] - 128s 164ms/step - loss: 0.5984 -
accuracy: 0.7981 - val_loss: 0.6875 - val_accuracy: 0.7788
Epoch 55/100
782/782 [==============================] - 126s 161ms/step - loss: 0.5920 -
accuracy: 0.7979 - val_loss: 0.6484 - val_accuracy: 0.7819
Epoch 56/100
782/782 [==============================] - 125s 160ms/step - loss: 0.5955 -
accuracy: 0.8002 - val_loss: 0.6262 - val_accuracy: 0.7932
Epoch 57/100
782/782 [==============================] - 126s 162ms/step - loss: 0.5883 -
accuracy: 0.8014 - val_loss: 0.6695 - val_accuracy: 0.7776
Epoch 58/100
782/782 [==============================] - 125s 160ms/step - loss: 0.5888 -
accuracy: 0.7994 - val_loss: 0.6349 - val_accuracy: 0.7873
Epoch 59/100
782/782 [==============================] - 128s 163ms/step - loss: 0.5824 -
accuracy: 0.8012 - val_loss: 0.6337 - val_accuracy: 0.7873
Epoch 60/100
782/782 [==============================] - 123s 158ms/step - loss: 0.5820 -
accuracy: 0.8027 - val_loss: 0.6200 - val_accuracy: 0.7947
Epoch 61/100
782/782 [==============================] - 128s 163ms/step - loss: 0.5785 -
accuracy: 0.8058 - val_loss: 0.6174 - val_accuracy: 0.7956
Epoch 62/100
782/782 [==============================] - 126s 161ms/step - loss: 0.5815 -
accuracy: 0.8047 - val_loss: 0.6571 - val_accuracy: 0.7816
Epoch 63/100
782/782 [==============================] - 124s 159ms/step - loss: 0.5764 -
accuracy: 0.8064 - val_loss: 0.6478 - val_accuracy: 0.7885
Epoch 64/100
782/782 [==============================] - 128s 164ms/step - loss: 0.5726 -
accuracy: 0.8061 - val_loss: 0.6116 - val_accuracy: 0.7966
Epoch 65/100
782/782 [==============================] - 126s 162ms/step - loss: 0.5794 -
accuracy: 0.8062 - val_loss: 0.6548 - val_accuracy: 0.7893
Epoch 66/100
782/782 [==============================] - 124s 159ms/step - loss: 0.5694 -
accuracy: 0.8083 - val_loss: 0.6390 - val_accuracy: 0.7920
```

```
Epoch 67/100
782/782 [==============================] - 128s 163ms/step - loss: 0.5735 -
accuracy: 0.8063 - val_loss: 0.6546 - val_accuracy: 0.7868
Epoch 68/100
782/782 [==============================] - 129s 165ms/step - loss: 0.5664 -
accuracy: 0.8090 - val_loss: 0.6011 - val_accuracy: 0.8007
Epoch 69/100
782/782 [==============================] - 128s 164ms/step - loss: 0.5612 -
accuracy: 0.8104 - val_loss: 0.6131 - val_accuracy: 0.7969
Epoch 70/100
782/782 [==============================] - 130s 166ms/step - loss: 0.5639 -
accuracy: 0.8111 - val_loss: 0.6695 - val_accuracy: 0.7912
Epoch 71/100
782/782 [==============================] - 132s 169ms/step - loss: 0.5618 -
accuracy: 0.8105 - val_loss: 0.6283 - val_accuracy: 0.7933
Epoch 72/100
782/782 [==============================] - 133s 170ms/step - loss: 0.5608 -
accuracy: 0.8123 - val_loss: 0.6180 - val_accuracy: 0.7967
Epoch 73/100
782/782 [==============================] - 143s 183ms/step - loss: 0.5604 -
accuracy: 0.8153 - val_loss: 0.6558 - val_accuracy: 0.7921
Epoch 74/100
782/782 [==============================] - 197s 252ms/step - loss: 0.5562 -
accuracy: 0.8144 - val_loss: 0.6027 - val_accuracy: 0.7988
Epoch 75/100
782/782 [==============================] - 138s 177ms/step - loss: 0.5539 -
accuracy: 0.8134 - val_loss: 0.6244 - val_accuracy: 0.7922
Epoch 76/100
782/782 [==============================] - 130s 167ms/step - loss: 0.5595 -
accuracy: 0.8126 - val_loss: 0.6237 - val_accuracy: 0.7990
Epoch 77/100
782/782 [==============================] - 138s 176ms/step - loss: 0.5556 -
accuracy: 0.8139 - val_loss: 0.6069 - val_accuracy: 0.8019
Epoch 78/100
782/782 [==============================] - 169s 216ms/step - loss: 0.5578 -
accuracy: 0.8133 - val_loss: 0.6235 - val_accuracy: 0.8005
Epoch 79/100
782/782 [==============================] - 162s 207ms/step - loss: 0.5539 -
accuracy: 0.8154 - val_loss: 0.6022 - val_accuracy: 0.8039
Epoch 80/100
782/782 [==============================] - 133s 170ms/step - loss: 0.5442 -
accuracy: 0.8169 - val_loss: 0.6299 - val_accuracy: 0.7953
Epoch 81/100
782/782 [==============================] - 131s 168ms/step - loss: 0.5490 -
accuracy: 0.8173 - val_loss: 0.6212 - val_accuracy: 0.7989
Epoch 82/100
782/782 [==============================] - 128s 163ms/step - loss: 0.5448 -
accuracy: 0.8181 - val_loss: 0.6493 - val_accuracy: 0.7912
```

```
Epoch 83/100
782/782 [==============================] - 125s 160ms/step - loss: 0.5471 -
accuracy: 0.8167 - val_loss: 0.6570 - val_accuracy: 0.7883
Epoch 84/100
782/782 [==============================] - 128s 164ms/step - loss: 0.5419 -
accuracy: 0.8194 - val_loss: 0.6107 - val_accuracy: 0.7997
Epoch 85/100
782/782 [==============================] - 130s 166ms/step - loss: 0.5430 -
accuracy: 0.8207 - val_loss: 0.6182 - val_accuracy: 0.7947
Epoch 86/100
782/782 [==============================] - 127s 163ms/step - loss: 0.5414 -
accuracy: 0.8193 - val_loss: 0.6128 - val_accuracy: 0.7987
Epoch 87/100
782/782 [==============================] - 138s 177ms/step - loss: 0.5458 -
accuracy: 0.8175 - val_loss: 0.6226 - val_accuracy: 0.7950
Epoch 88/100
782/782 [==============================] - 156s 200ms/step - loss: 0.5362 -
accuracy: 0.8204 - val_loss: 0.6178 - val_accuracy: 0.8049
Epoch 89/100
782/782 [==============================] - 174s 223ms/step - loss: 0.5351 -
accuracy: 0.8218 - val_loss: 0.6945 - val_accuracy: 0.7826
Epoch 90/100
782/782 [==============================] - 4318s 6s/step - loss: 0.5391 -
accuracy: 0.8205 - val_loss: 0.6419 - val_accuracy: 0.7913
Epoch 91/100
782/782 [==============================] - 98s 126ms/step - loss: 0.5372 -
accuracy: 0.8219 - val_loss: 0.6340 - val_accuracy: 0.7936
Epoch 92/100
782/782 [==============================] - 111s 143ms/step - loss: 0.5322 -
accuracy: 0.8201 - val_loss: 0.6407 - val_accuracy: 0.7983
Epoch 93/100
782/782 [==============================] - 114s 146ms/step - loss: 0.5335 -
accuracy: 0.8208 - val_loss: 0.6139 - val_accuracy: 0.8006
Epoch 94/100
782/782 [==============================] - 111s 143ms/step - loss: 0.5344 -
accuracy: 0.8221 - val_loss: 0.6296 - val_accuracy: 0.7975
Epoch 95/100
782/782 [==============================] - 4057s 5s/step - loss: 0.5358 -
accuracy: 0.8233 - val_loss: 0.6082 - val_accuracy: 0.8043
Epoch 96/100
782/782 [==============================] - 97s 124ms/step - loss: 0.5347 -
accuracy: 0.8227 - val_loss: 0.6165 - val_accuracy: 0.8021
Epoch 97/100
782/782 [==============================] - 97s 123ms/step - loss: 0.5324 -
accuracy: 0.8236 - val_loss: 0.6302 - val_accuracy: 0.8033
Epoch 98/100
782/782 [==============================] - 98s 125ms/step - loss: 0.5314 -
accuracy: 0.8219 - val_loss: 0.6272 - val_accuracy: 0.8012
```

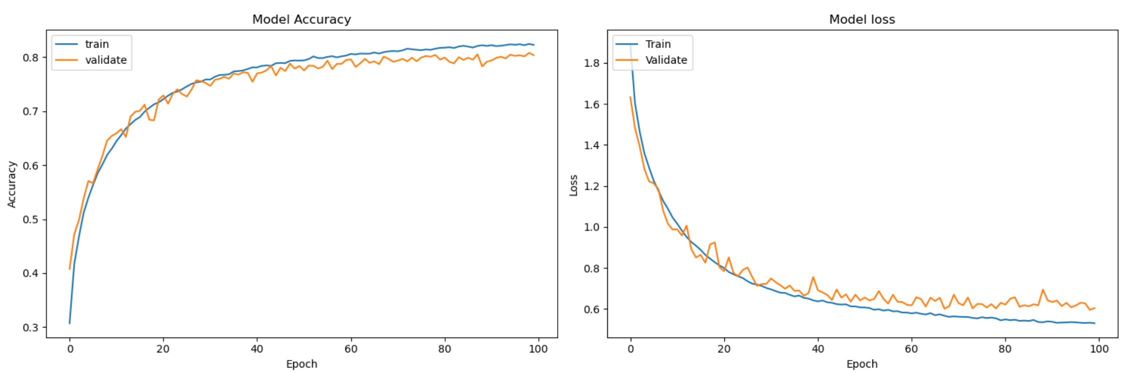
```
Epoch 99/100
782/782 [==============================] - 107s 136ms/step - loss: 0.5326 -
accuracy: 0.8242 - val_loss: 0.5953 - val_accuracy: 0.8077
Epoch 100/100
782/782 [==============================] - 109s 139ms/step - loss: 0.5300 -
accuracy: 0.8224 - val_loss: 0.6039 - val_accuracy: 0.8034
```

[9]:
```
def plotmodelhistory(history):
    fig,axs = plt.subplots(1,2, figsize = (15,5))
    axs[0].plot(history.history['accuracy'])
    axs[0].plot(history.history['val_accuracy'])
    axs[0].set_title('Model Accuracy')
    axs[0].set_ylabel('Accuracy')
    axs[0].set_xlabel('Epoch')
    axs[0].legend(['train','validate'], loc = 'upper left')

    axs[1].plot(history.history['loss'])
    axs[1].plot(history.history['val_loss'])
    axs[1].set_title('Model loss')
    axs[1].set_ylabel('Loss')
    axs[1].set_xlabel('Epoch')
    axs[1].legend(['Train','Validate'], loc = 'upper left')

    plt.tight_layout()
    plt.show()
```

[10]:
```
plotmodelhistory(history)
```



[11]:
```
scores = model.evaluate(x_test, y_test, verbose =1)
print('Test loss:', scores[0])
print('Test accuracy:', scores[1])

pred = model.predict(x_test)
```

```
313/313 [==============================] - 7s 22ms/step - loss: 0.6039 -
accuracy: 0.8034
Test loss: 0.6038806438446045
Test accuracy: 0.8033999800682068
313/313 [==============================] - 9s 26ms/step
```

[13]:
```python
Y_pred_classes = np.argmax(pred, axis = 1)
Y_true = np.argmax(y_test, axis = 1)

print(classification_report(Y_true,Y_pred_classes))
```

```
              precision    recall  f1-score   support

           0       0.82      0.85      0.84      1000
           1       0.90      0.89      0.90      1000
           2       0.79      0.63      0.70      1000
           3       0.66      0.65      0.66      1000
           4       0.70      0.85      0.77      1000
           5       0.83      0.61      0.71      1000
           6       0.76      0.92      0.83      1000
           7       0.84      0.86      0.85      1000
           8       0.86      0.91      0.89      1000
           9       0.90      0.85      0.87      1000

    accuracy                           0.80     10000
   macro avg       0.81      0.80      0.80     10000
weighted avg       0.81      0.80      0.80     10000
```

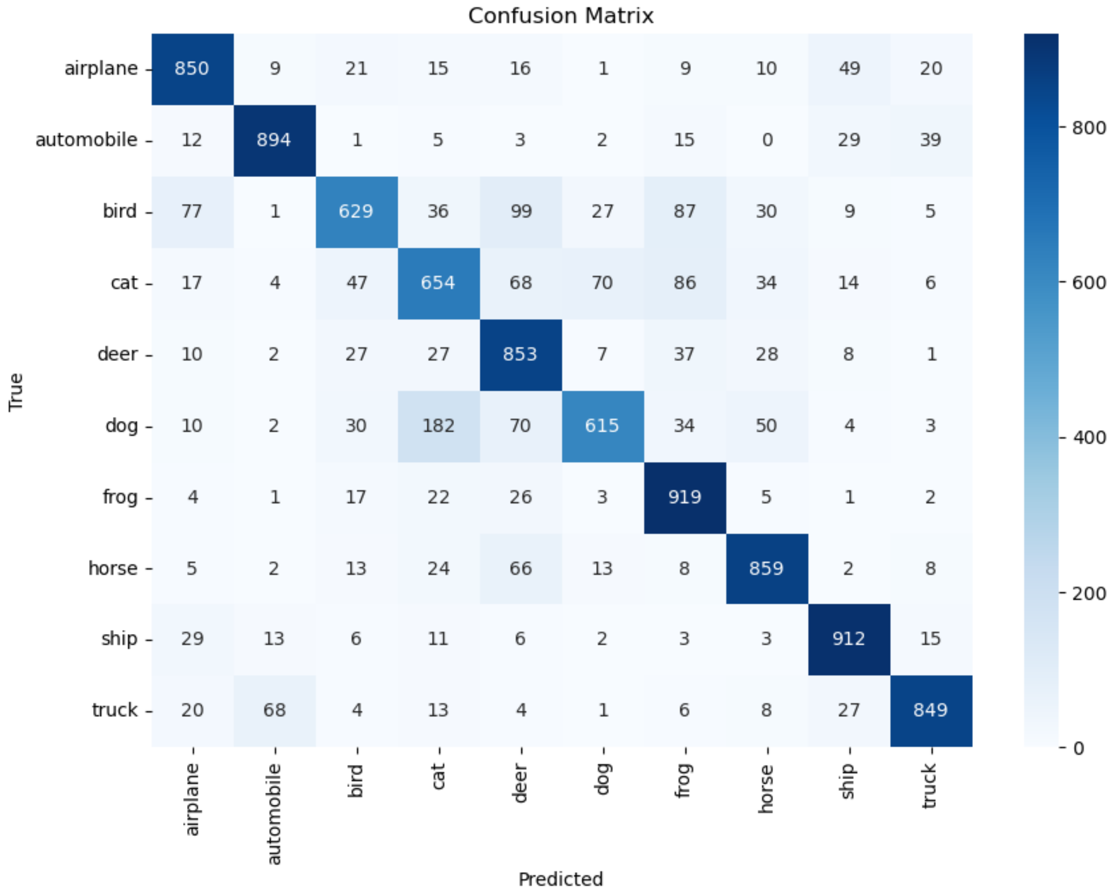
[21]:
```python
from sklearn.metrics import confusion_matrix, classification_report
import seaborn as sns
import matplotlib.pyplot as plt
class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog',␣
 ↪'horse', 'ship', 'truck']



# Generate confusion matrix
conf_matrix = confusion_matrix(Y_true, Y_pred_classes)

# Plot confusion matrix
plt.figure(figsize=(10, 7))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',␣
 ↪xticklabels=class_names, yticklabels=class_names)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
```

```
plt.show()
```



[22]:
```
test_loss, test_acc = model.evaluate(x_test, y_test, verbose=2)
print(f'Test accuracy: {test_acc:.4f}')
print(f'Test loss: {test_loss:.4f}')
```

```
313/313 - 6s - loss: 0.6039 - accuracy: 0.8034 - 6s/epoch - 19ms/step
Test accuracy: 0.8034
Test loss: 0.6039
```