Image Classification using deep learning

March 10, 2024

1 Convolution Neural Network in Keras with python on a CIFAR-10 dataset

[1]:
```
!pip install tensorflow
!pip install keras
```

```
Requirement already satisfied: tensorflow in d:\anaconda\lib\site-packages
(2.10.0)
Requirement already satisfied: absl-py>=1.0.0 in d:\anaconda\lib\site-packages
(from tensorflow) (1.3.0)
Requirement already satisfied: astunparse>=1.6.0 in d:\anaconda\lib\site-
packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=2.0 in d:\anaconda\lib\site-packages
(from tensorflow) (22.10.26)
Requirement already satisfied: gast<=0.4.0,>=0.2.1 in d:\anaconda\lib\site-
packages (from tensorflow) (0.4.0)
Requirement already satisfied: google-pasta>=0.1.1 in d:\anaconda\lib\site-
packages (from tensorflow) (0.2.0)
Requirement already satisfied: h5py>=2.9.0 in d:\anaconda\lib\site-packages
(from tensorflow) (3.7.0)
Requirement already satisfied: keras-preprocessing>=1.1.1 in
d:\anaconda\lib\site-packages (from tensorflow) (1.1.2)
Requirement already satisfied: libclang>=13.0.0 in d:\anaconda\lib\site-packages
(from tensorflow) (14.0.6)
Requirement already satisfied: numpy>=1.20 in d:\anaconda\lib\site-packages
(from tensorflow) (1.23.5)
Requirement already satisfied: opt-einsum>=2.3.2 in d:\anaconda\lib\site-
packages (from tensorflow) (3.3.0)
Requirement already satisfied: packaging in d:\anaconda\lib\site-packages (from
tensorflow) (22.0)
Requirement already satisfied: protobuf<3.20,>=3.9.2 in d:\anaconda\lib\site-
packages (from tensorflow) (3.19.6)
Requirement already satisfied: setuptools in d:\anaconda\lib\site-packages (from
tensorflow) (67.4.0)
Requirement already satisfied: six>=1.12.0 in d:\anaconda\lib\site-packages
(from tensorflow) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in d:\anaconda\lib\site-packages
(from tensorflow) (2.1.0)
```

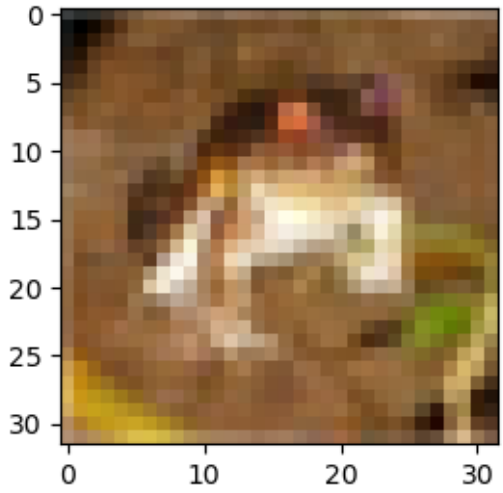
```
Requirement already satisfied: typing-extensions>=3.6.6 in d:\anaconda\lib\site-
packages (from tensorflow) (4.8.0)
Requirement already satisfied: wrapt>=1.11.0 in d:\anaconda\lib\site-packages
(from tensorflow) (1.14.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in
d:\anaconda\lib\site-packages (from tensorflow) (0.27.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in d:\anaconda\lib\site-
packages (from tensorflow) (1.50.0)
Requirement already satisfied: tensorboard<2.11,>=2.10 in d:\anaconda\lib\site-
packages (from tensorflow) (2.10.1)
Requirement already satisfied: tensorflow-estimator<2.11,>=2.10.0 in
d:\anaconda\lib\site-packages (from tensorflow) (2.10.0)
Requirement already satisfied: keras<2.11,>=2.10.0 in d:\anaconda\lib\site-
packages (from tensorflow) (2.10.0)
Requirement already satisfied: wheel<1.0,>=0.23.0 in d:\anaconda\lib\site-
packages (from astunparse>=1.6.0->tensorflow) (0.37.1)
Requirement already satisfied: google-auth<3,>=1.6.3 in d:\anaconda\lib\site-
packages (from tensorboard<2.11,>=2.10->tensorflow) (2.14.0)
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in
d:\anaconda\lib\site-packages (from tensorboard<2.11,>=2.10->tensorflow) (0.4.6)
Requirement already satisfied: markdown>=2.6.8 in d:\anaconda\lib\site-packages
(from tensorboard<2.11,>=2.10->tensorflow) (3.4.1)
Requirement already satisfied: requests<3,>=2.21.0 in d:\anaconda\lib\site-
packages (from tensorboard<2.11,>=2.10->tensorflow) (2.28.1)
Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0 in
d:\anaconda\lib\site-packages (from tensorboard<2.11,>=2.10->tensorflow) (0.6.1)
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in
d:\anaconda\lib\site-packages (from tensorboard<2.11,>=2.10->tensorflow) (1.8.1)
Requirement already satisfied: werkzeug>=1.0.1 in d:\anaconda\lib\site-packages
(from tensorboard<2.11,>=2.10->tensorflow) (2.2.2)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in d:\anaconda\lib\site-
packages (from google-auth<3,>=1.6.3->tensorboard<2.11,>=2.10->tensorflow)
(5.2.0)
Requirement already satisfied: pyasn1-modules>=0.2.1 in d:\anaconda\lib\site-
packages (from google-auth<3,>=1.6.3->tensorboard<2.11,>=2.10->tensorflow)
(0.2.8)
Requirement already satisfied: rsa<5,>=3.1.4 in d:\anaconda\lib\site-packages
(from google-auth<3,>=1.6.3->tensorboard<2.11,>=2.10->tensorflow) (4.7.2)
Requirement already satisfied: requests-oauthlib>=0.7.0 in d:\anaconda\lib\site-
packages (from google-auth-
oauthlib<0.5,>=0.4.1->tensorboard<2.11,>=2.10->tensorflow) (1.3.1)
Requirement already satisfied: importlib-metadata>=4.4 in d:\anaconda\lib\site-
packages (from markdown>=2.6.8->tensorboard<2.11,>=2.10->tensorflow) (4.11.3)
Requirement already satisfied: charset-normalizer<3,>=2 in d:\anaconda\lib\site-
packages (from requests<3,>=2.21.0->tensorboard<2.11,>=2.10->tensorflow) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in d:\anaconda\lib\site-packages
(from requests<3,>=2.21.0->tensorboard<2.11,>=2.10->tensorflow) (3.4)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in d:\anaconda\lib\site-
```

```
packages (from requests<3,>=2.21.0->tensorboard<2.11,>=2.10->tensorflow)
(1.26.14)
Requirement already satisfied: certifi>=2017.4.17 in d:\anaconda\lib\site-
packages (from requests<3,>=2.21.0->tensorboard<2.11,>=2.10->tensorflow)
(2022.12.7)
Requirement already satisfied: MarkupSafe>=2.1.1 in d:\anaconda\lib\site-
packages (from werkzeug>=1.0.1->tensorboard<2.11,>=2.10->tensorflow) (2.1.1)
Requirement already satisfied: zipp>=0.5 in d:\anaconda\lib\site-packages (from
importlib-metadata>=4.4->markdown>=2.6.8->tensorboard<2.11,>=2.10->tensorflow)
(3.11.0)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in d:\anaconda\lib\site-
packages (from pyasn1-modules>=0.2.1->google-
auth<3,>=1.6.3->tensorboard<2.11,>=2.10->tensorflow) (0.4.8)
Requirement already satisfied: oauthlib>=3.0.0 in d:\anaconda\lib\site-packages
(from requests-oauthlib>=0.7.0->google-auth-
oauthlib<0.5,>=0.4.1->tensorboard<2.11,>=2.10->tensorflow) (3.2.2)


[notice] A new release of pip is available: 23.2.1 -> 24.0
[notice] To update, run: python.exe -m pip install --upgrade pip
```

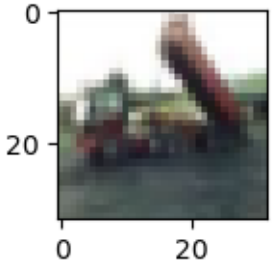
```
Requirement already satisfied: keras in d:\anaconda\lib\site-packages (2.10.0)


[notice] A new release of pip is available: 23.2.1 -> 24.0
[notice] To update, run: python.exe -m pip install --upgrade pip
```

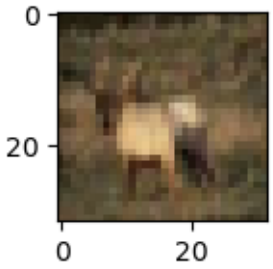
[2]:
```
# importing the neccessary libraries and loading the cifar10 dataset
import numpy as np
import pandas as pd
from keras.datasets import cifar10

(train_X,train_Y),(test_X,test_Y)=cifar10.load_data()
```

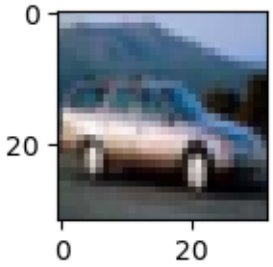
[3]:
```
# plotting some images from dataset
import matplotlib.pyplot as plt
n=6
plt.figure(figsize=(20,10))
for i in range(n):
    plt.subplot(330+1+i)
    plt.imshow(train_X[i])
    plt.show()
```

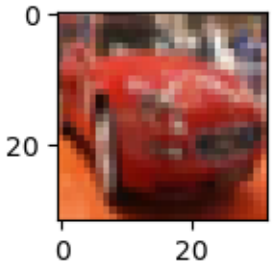

0
5
10
15
20
25
30
0
10
20
30


0
20
0
20


0
20
0
20





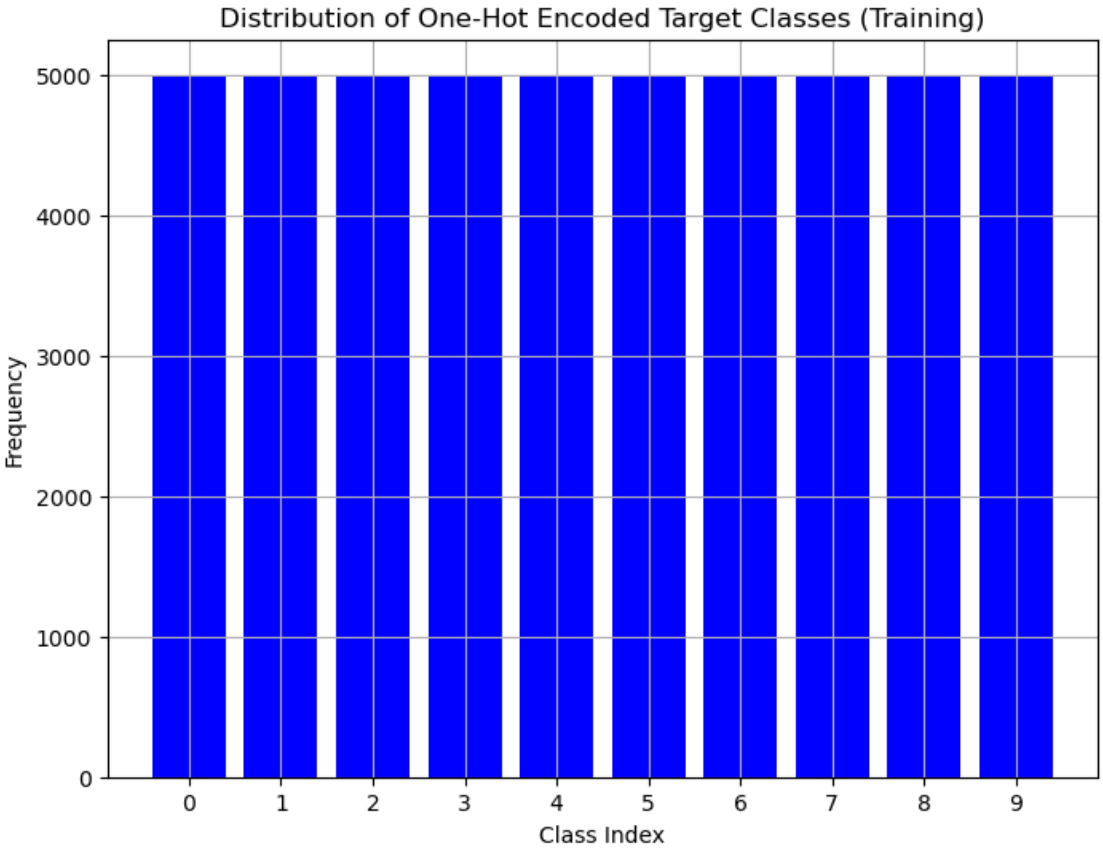


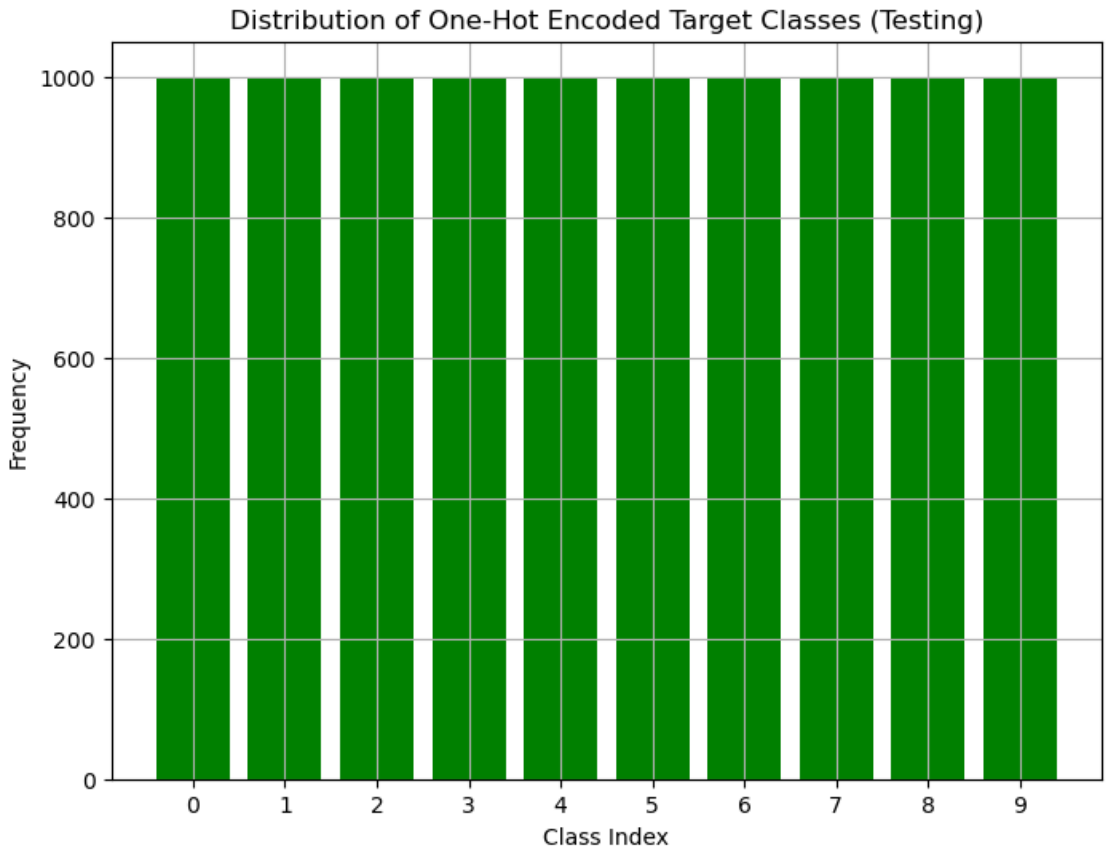
[4]:
```
# importing the required layers to create CNN architecture
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from keras.layers import Flatten
from keras.constraints import maxnorm
from keras.optimizers import SGD
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
from keras.utils import np_utils
```

[5]:
```
# Convert the pixel values of the dataset to float type
# normalize the dataset
train_x=train_X.astype('float32')
test_X=test_X.astype('float32')

train_X=train_X/255.0
test_X=test_X/255.0
```

[6]:
```
# one hot encoding for the target classes
train_Y=np_utils.to_categorical(train_Y)
test_Y=np_utils.to_categorical(test_Y)

num_classes=test_Y.shape[1]
```

[7]:
```
# Plotting train_Y
plt.figure(figsize=(8, 6))
plt.bar(range(num_classes), train_Y.sum(axis=0), color='blue')
plt.title('Distribution of One-Hot Encoded Target Classes (Training)')
plt.xlabel('Class Index')
plt.ylabel('Frequency')
plt.xticks(range(num_classes))
plt.grid(True)
plt.show()

# Plotting test_Y
plt.figure(figsize=(8, 6))
plt.bar(range(num_classes), test_Y.sum(axis=0), color='green')
plt.title('Distribution of One-Hot Encoded Target Classes (Testing)')
plt.xlabel('Class Index')
plt.ylabel('Frequency')
plt.xticks(range(num_classes))
plt.grid(True)
plt.show()
```


Distribution of One-Hot Encoded Target Classes (Training)
Frequency
Class Index
0
1000
2000
3000
4000
5000
0
1
2
3
4
5
6
7
8
9



[8]:
```
# Create the sequential model and add the layers
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(Flatten())
model.add(Dense(512,activation='relu',kernel_constraint=maxnorm(3)))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))
```

[9]:
```
# Configure the optimizer and compile the model
sgd=SGD(lr=0.01,momentum=0.9,decay=(0.01/25),nesterov=False)

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

```
D:\anaconda\lib\site-
packages\keras\optimizers\optimizer_v2\gradient_descent.py:111: UserWarning: The
`lr` argument is deprecated, use `learning_rate` instead.
  super().__init__(name, **kwargs)
```

[10]:
```
# view the model summary
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
_________________________________________________________________
```

[11]:
```
# train the model
model.fit(train_X,train_Y,
    validation_data=(test_X,test_Y),
    epochs=10,batch_size=32)
```

```
Epoch 1/10
1563/1563 [==============================] - 122s 78ms/step - loss: 1.7188 -
accuracy: 0.3804 - val_loss: 1.3933 - val_accuracy: 0.4951
Epoch 2/10
1563/1563 [==============================] - 111s 71ms/step - loss: 1.3521 -
accuracy: 0.5120 - val_loss: 1.2088 - val_accuracy: 0.5750
Epoch 3/10
1563/1563 [==============================] - 105s 67ms/step - loss: 1.1880 -
accuracy: 0.5745 - val_loss: 1.1283 - val_accuracy: 0.5963
```

```
Epoch 4/10
1563/1563 [==============================] - 105s 67ms/step - loss: 1.0731 -
accuracy: 0.6182 - val_loss: 1.0463 - val_accuracy: 0.6313
Epoch 5/10
1563/1563 [==============================] - 106s 68ms/step - loss: 0.9822 -
accuracy: 0.6508 - val_loss: 1.0172 - val_accuracy: 0.6388
Epoch 6/10
1563/1563 [==============================] - 105s 67ms/step - loss: 0.8975 -
accuracy: 0.6818 - val_loss: 0.9752 - val_accuracy: 0.6612
Epoch 7/10
1563/1563 [==============================] - 107s 68ms/step - loss: 0.8252 -
accuracy: 0.7062 - val_loss: 0.9616 - val_accuracy: 0.6609
Epoch 8/10
1563/1563 [==============================] - 108s 69ms/step - loss: 0.7600 -
accuracy: 0.7304 - val_loss: 0.9258 - val_accuracy: 0.6802
Epoch 9/10
1563/1563 [==============================] - 107s 69ms/step - loss: 0.7024 -
accuracy: 0.7505 - val_loss: 0.9341 - val_accuracy: 0.6823
Epoch 10/10
1563/1563 [==============================] - 108s 69ms/step - loss: 0.6545 -
accuracy: 0.7684 - val_loss: 0.9217 - val_accuracy: 0.6878
```

[11]:
```
<keras.callbacks.History at 0x2509ffdcd30>
```

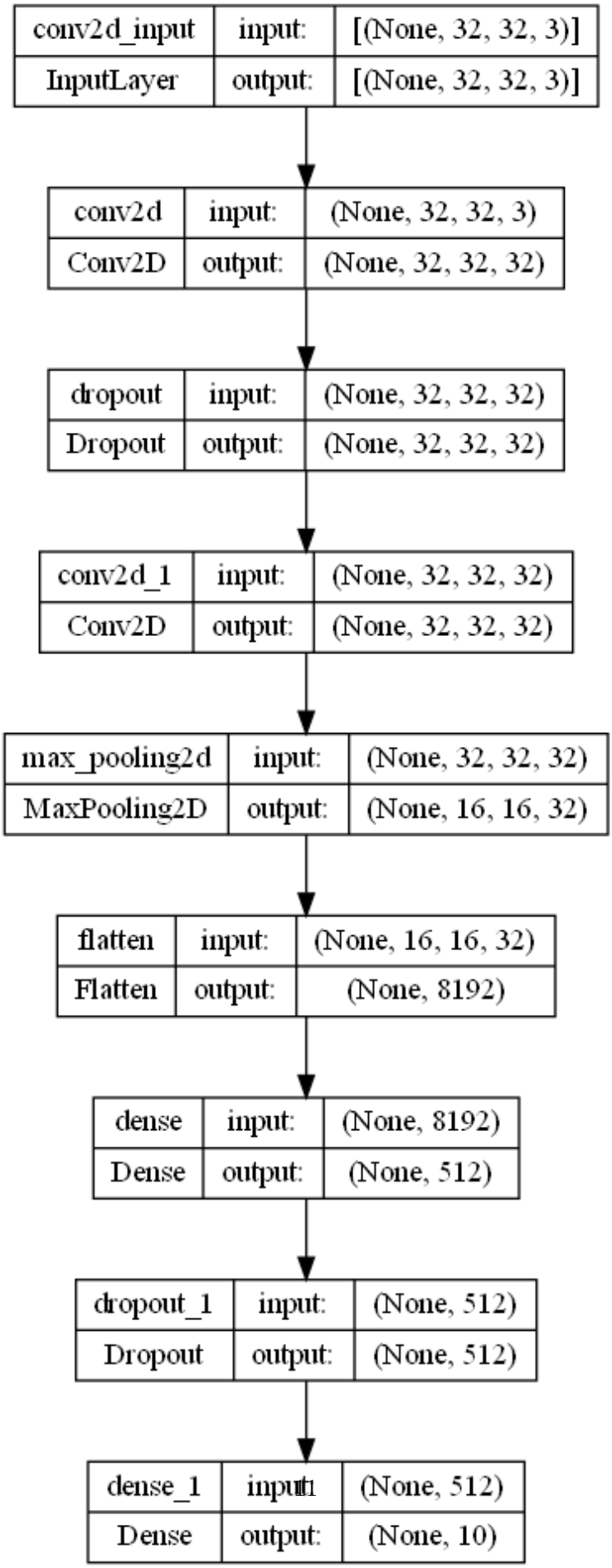
[12]:
```
# calculating the accuracy on testing data
_,acc=model.evaluate(test_X,test_Y)
print(acc*100)
```

```
313/313 [==============================] - 4s 12ms/step - loss: 0.9217 -
accuracy: 0.6878
68.77999901771545
```

[13]:
```
# save the model
model.save("model1_cifar_10epoch.h5")
```

[14]:
```
from keras.models import load_model
from keras.utils import plot_model

# Load the saved model
model = load_model("model1_cifar_10epoch.h5")

# Visualize the model architecture
plot_model(model, to_file='model1_cifar.png', show_shapes=True,␣
 ↪show_layer_names=True)
```

[14]:


conv2d_input | input: | [(None, 32, 32, 3)]
InputLayer | output: | [(None, 32, 32, 3)]
conv2d | input: | (None, 32, 32, 3)
Conv2D | output: | (None, 32, 32, 32)
dropout | input: | (None, 32, 32, 32)
Dropout | output: | (None, 32, 32, 32)
conv2d_1 | input: | (None, 32, 32, 32)
Conv2D | output: | (None, 32, 32, 32)
max_pooling2d | input: | (None, 32, 32, 32)
MaxPooling2D | output: | (None, 16, 16, 32)
flatten | input: | (None, 16, 16, 32)
Flatten | output: | (None, 8192)
dense | input: | (None, 8192)
Dense | output: | (None, 512)
dropout_1 | input: | (None, 512)
Dropout | output: | (None, 512)
dense_1 | input: | (None, 512)
Dense | output: | (None, 10)

[16]:
```
results={
   0:'aeroplane',
   1:'automobile',
   2:'bird',
   3:'cat',
   4:'deer',
   5:'dog',
   6:'frog',
   7:'horse',
   8:'ship',
   9:'truck'
}
from PIL import Image
import numpy as np
im=Image.open("C:/Users/Dharini/Downloads/cat.jpeg")
# the input image is required to be in the shape of dataset, i.e (32,32,3)

im=im.resize((32,32))
# Convert the image to a numpy array and normalize
im = np.array(im) / 255.0
# Reshape the image to match the input shape of the model
im = np.expand_dims(im, axis=0)
# Make predictions
predictions = model.predict(im)
pred_class_index = np.argmax(predictions[0])
# Print the predicted class
print("Predicted class:", results[pred_class_index])
```

```
1/1 [==============================] - 0s 94ms/step
Predicted class: cat
```

[18]:
```
import ipywidgets as widgets
from IPython.display import display, Image
import numpy as np
from keras.models import load_model
from PIL import Image as PILImage
import io  # Import io module for working with byte streams

# Load the trained model
model = load_model("model1_cifar_10epoch.h5")

# Define the class labels
results = {
   0: 'aeroplane',
   1: 'automobile',
```

```
    2: 'bird',
    3: 'cat',
    4: 'deer',
    5: 'dog',
    6: 'frog',
    7: 'horse',
    8: 'ship',
    9: 'truck'
}

# Create file upload widget
uploader = widgets.FileUpload(
    accept='.jpg,.jpeg,.png',  # Accept only image files
    multiple=False  # Allow only single file upload
)

# Create output widget for displaying results
output = widgets.Output()

# Function to predict the class of the uploaded image
def predict_image(change):
    output.clear_output()
    image = PILImage.open(io.BytesIO(uploader.data[-1]))
    image = image.resize((32, 32))
    image_data = np.array(image) / 255.0
    image_data = np.expand_dims(image_data, axis=0)
    prediction_probabilities = model.predict(image_data)[0]
    predicted_class_index = np.argmax(prediction_probabilities)
    predicted_class = results[predicted_class_index]

    with output:
        display(image)
        print("Predicted class:", predicted_class)

# Link file upload event to predict_image function
uploader.observe(predict_image, names='data')

# Display widgets
display(uploader, output)
```

```
FileUpload(value={}, accept='.jpg,.jpeg,.png', description='Upload')
```

```
Output()
```

```
1/1 [==============================] - 0s 45ms/step
1/1 [==============================] - 0s 22ms/step
```

2 Conclusion

- The model is correctly identifying the objects in the images, and the predicted classes match the ground truth labels.

[ ]: