Tushar Bhagat

In [10]:
```
import numpy as np
import tensorflow as tf
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.utils import plot_model
from tensorflow.keras.layers import Dense,Input
```

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

# Preprocess the data
x_train = x_train.astype('float32') / 255
x_test = x_test.astype('float32') / 255

# One-hot encode the labels
y_train = to_categorical(y_train, 10)
y_test = to_categorical(y_test, 10)
```

```
A local file was found, but it seems to be incomplete or outdated because the auto file hash does not match the original value o
f 6d958be074577803d12ecdefd02955f39262c83c16fe9348329d7fe0b5c001ce so we will re-download the data.
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170498071/170498071 ━━━━━━━━━━━━━━━━━━━━ 2748s 16us/step
```

In [12]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout


def create_model():
    model = Sequential([
        Input(shape=(32, 32, 3)),
        Conv2D(32, kernel_size=(3, 3), activation='relu'),
        MaxPooling2D(pool_size=(2, 2)),
        Conv2D(64, kernel_size=(3, 3), activation='relu'),
        MaxPooling2D(pool_size=(2, 2)),
        Conv2D(128, kernel_size=(3, 3), activation='relu'),
        MaxPooling2D(pool_size=(2, 2)),
        Flatten(),
        Dense(128, activation='relu'),
        Dropout(0.5),
        Dense(10, activation='softmax')
    ])
    return model
```

In [13]:
```
# Create the model
model = create_model()

# Compile the model
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# Train the model
history = model.fit(x_train, y_train,
                    epochs=20,
                    batch_size=64,
                    validation_split=0.2)
```

```
Epoch 1/20
625/625 ━━━━━━━━━━━━━━━━━━━━ 14s 21ms/step - accuracy: 0.2580 - loss: 1.9729 - val_accuracy: 0.4777 - val_loss: 1.4467
Epoch 2/20
625/625 ━━━━━━━━━━━━━━━━━━━━ 13s 20ms/step - accuracy: 0.4784 - loss: 1.4431 - val_accuracy: 0.5356 - val_loss: 1.2932
Epoch 3/20
625/625 ━━━━━━━━━━━━━━━━━━━━ 12s 20ms/step - accuracy: 0.5457 - loss: 1.2794 - val_accuracy: 0.5807 - val_loss: 1.1648
Epoch 4/20
625/625 ━━━━━━━━━━━━━━━━━━━━ 12s 20ms/step - accuracy: 0.5913 - loss: 1.1585 - val_accuracy: 0.6204 - val_loss: 1.0700
Epoch 5/20
625/625 ━━━━━━━━━━━━━━━━━━━━ 12s 20ms/step - accuracy: 0.6277 - loss: 1.0620 - val_accuracy: 0.6480 - val_loss: 1.0111
Epoch 6/20
625/625 ━━━━━━━━━━━━━━━━━━━━ 12s 20ms/step - accuracy: 0.6549 - loss: 0.9837 - val_accuracy: 0.6746 - val_loss: 0.9352
Epoch 7/20
625/625 ━━━━━━━━━━━━━━━━━━━━ 12s 20ms/step - accuracy: 0.6817 - loss: 0.9123 - val_accuracy: 0.6806 - val_loss: 0.8996
Epoch 8/20
625/625 ━━━━━━━━━━━━━━━━━━━━ 13s 20ms/step - accuracy: 0.6938 - loss: 0.8781 - val_accuracy: 0.6897 - val_loss: 0.9115
Epoch 9/20
625/625 ━━━━━━━━━━━━━━━━━━━━ 13s 20ms/step - accuracy: 0.7174 - loss: 0.8156 - val_accuracy: 0.6932 - val_loss: 0.8913
Epoch 10/20
625/625 ━━━━━━━━━━━━━━━━━━━━ 13s 20ms/step - accuracy: 0.7325 - loss: 0.7763 - val_accuracy: 0.7063 - val_loss: 0.8553
Epoch 11/20
625/625 ━━━━━━━━━━━━━━━━━━━━ 12s 20ms/step - accuracy: 0.7410 - loss: 0.7442 - val_accuracy: 0.7091 - val_loss: 0.8545
Epoch 12/20
625/625 ━━━━━━━━━━━━━━━━━━━━ 13s 21ms/step - accuracy: 0.7462 - loss: 0.7215 - val_accuracy: 0.7093 - val_loss: 0.8770
Epoch 13/20
625/625 ━━━━━━━━━━━━━━━━━━━━ 13s 20ms/step - accuracy: 0.7635 - loss: 0.6711 - val_accuracy: 0.7248 - val_loss: 0.8325
Epoch 14/20
625/625 ━━━━━━━━━━━━━━━━━━━━ 13s 21ms/step - accuracy: 0.7741 - loss: 0.6511 - val_accuracy: 0.7202 - val_loss: 0.8453
Epoch 15/20
625/625 ━━━━━━━━━━━━━━━━━━━━ 13s 20ms/step - accuracy: 0.7844 - loss: 0.6116 - val_accuracy: 0.7303 - val_loss: 0.8380
Epoch 16/20
625/625 ━━━━━━━━━━━━━━━━━━━━ 13s 20ms/step - accuracy: 0.7905 - loss: 0.5914 - val_accuracy: 0.7268 - val_loss: 0.8381
Epoch 17/20
625/625 ━━━━━━━━━━━━━━━━━━━━ 13s 20ms/step - accuracy: 0.8012 - loss: 0.5651 - val_accuracy: 0.7180 - val_loss: 0.8743
Epoch 18/20
625/625 ━━━━━━━━━━━━━━━━━━━━ 13s 20ms/step - accuracy: 0.8064 - loss: 0.5447 - val_accuracy: 0.7300 - val_loss: 0.8717
Epoch 19/20
625/625 ━━━━━━━━━━━━━━━━━━━━ 13s 20ms/step - accuracy: 0.8184 - loss: 0.5192 - val_accuracy: 0.7222 - val_loss: 0.9046
Epoch 20/20
625/625 ━━━━━━━━━━━━━━━━━━━━ 13s 20ms/step - accuracy: 0.8199 - loss: 0.5149 - val_accuracy: 0.7255 - val_loss: 0.9040
```

In [14]:
```
model.summary()
```

Model: "sequential_1"

Layer (type)	Output Shape	Param #
conv2d_3 (Conv2D)	(None, 30, 30, 32)	896
max_pooling2d_3 (MaxPooling2D)	(None, 15, 15, 32)	0
conv2d_4 (Conv2D)	(None, 13, 13, 64)	18,496
max_pooling2d_4 (MaxPooling2D)	(None, 6, 6, 64)	0
conv2d_5 (Conv2D)	(None, 4, 4, 128)	73,856
max_pooling2d_5 (MaxPooling2D)	(None, 2, 2, 128)	0
flatten_1 (Flatten)	(None, 512)	0
dense_2 (Dense)	(None, 128)	65,664
dropout_1 (Dropout)	(None, 128)	0
dense_3 (Dense)	(None, 10)	1,290

Total params: 480,608 (1.83 MB)

Trainable params: 160,202 (625.79 KB)

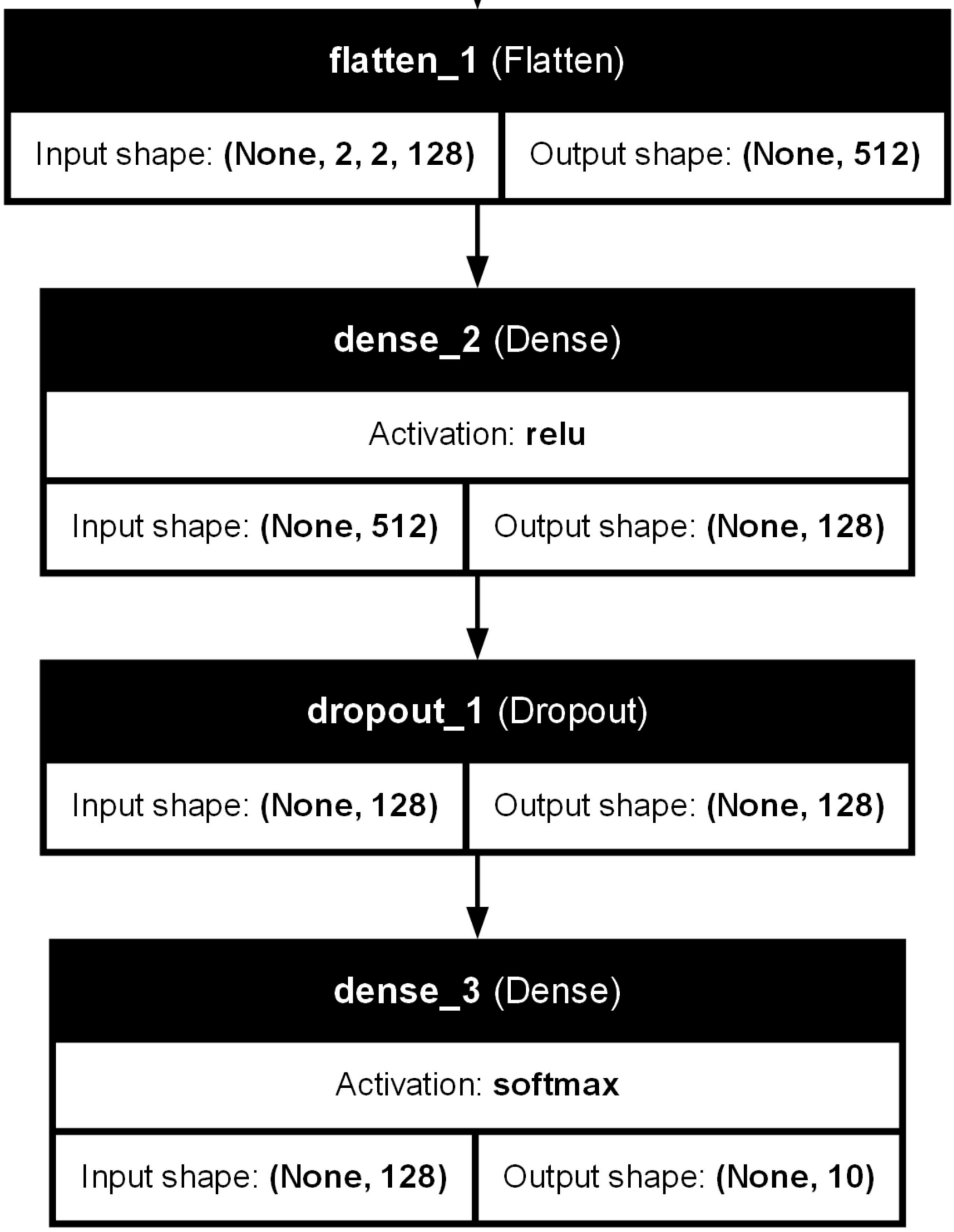
Non-trainable params: 0 (0.00 B)

Optimizer params: 320,406 (1.22 MB)

In [9]:
```
#plotting the model
plot_model(model,to_file='C:/Users/tmbha/Downloads/DLprac3_model.png',show_shapes=True,show_layer_names=True,show_layer_activatio
```

Out[9]:

conv2d_3 (Conv2D)
Activation: relu
Input shape: (None, 32, 32, 3)
Output shape: (None, 30, 30, 32)
max_pooling2d_3 (MaxPooling2D)
Input shape: (None, 30, 30, 32)
Output shape: (None, 15, 15, 32)
conv2d_4 (Conv2D)
Activation: relu
Input shape: (None, 15, 15, 32)
Output shape: (None, 13, 13, 64)
max_pooling2d_4 (MaxPooling2D)
Input shape: (None, 13, 13, 64)
Output shape: (None, 6, 6, 64)
conv2d_5 (Conv2D)
Activation: relu
Input shape: (None, 6, 6, 64)
Output shape: (None, 4, 4, 128)
max_pooling2d_5 (MaxPooling2D)
Input shape: (None, 4, 4, 128)
Output shape: (None, 2, 2, 128)



In [15]:

```python
import matplotlib.pyplot as plt

# Evaluate the model on test data
test_loss, test_accuracy = model.evaluate(x_test, y_test)
print(f'Test Loss: {test_loss:.4f}')
print(f'Test Accuracy: {test_accuracy:.4f}')

# Plot training & validation loss
plt.figure(figsize=(14, 6))

plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()

# Plot training & validation accuracy
plt.subplot(1, 2, 2)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()

plt.show()
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.7200 - loss: 0.9020
Test Loss: 0.9199
Test Accuracy: 0.7157
```