2. Build a CNN from scratch to classify images in the CIFAR-10 dataset. Focus on designing an effective CNN architecture and using pooling, dropout, and batch normalization.

Step 1: Import Libraries

```
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.utils import to_categorical
```

Step 2: Load & Normalize Dataset

```
(x_train, y_train), (x_test, y_test) =
tf.keras.datasets.cifar10.load_data()

x_train = x_train / 255.0
x_test = x_test / 255.0

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

Step 3: Design CNN Architecture

```
model = models.Sequential()

# Block 1
model.add(layers.Conv2D(32, (3,3), padding='same', activation='relu',
input_shape=(32,32,3)))
model.add(layers.BatchNormalization())
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Dropout(0.25))

# Block 2
model.add(layers.Conv2D(64, (3,3), padding='same', activation='relu'))
model.add(layers.BatchNormalization())
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Dropout(0.25))

# Block 3
model.add(layers.Conv2D(128, (3,3), padding='same', activation='relu'))
model.add(layers.BatchNormalization())
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Dropout(0.25))

# Fully Connected Layers
model.add(layers.Flatten())
model.add(layers.Dense(256, activation='relu'))
model.add(layers.BatchNormalization())
model.add(layers.Dropout(0.5))
model.add(layers.Dense(10, activation='softmax'))
```

Step 4: Compile Model

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

Step 5: Train Model

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

Step 6: Evaluate Model

```
test_loss, test_acc = model.evaluate(x_test, y_test)
print("Test Accuracy:", test_acc)
```

Expected accuracy: **78% – 85%** □

Step 7: Prediction Example

```
class_names =
['airplane','automobile','bird','cat','deer','dog','frog','horse','ship','t
ruck']

idx = 10
img = x_test[idx]
pred = model.predict(img.reshape(1,32,32,3))
print("Predicted:", class_names[np.argmax(pred)])
plt.imshow(img)
plt.axis('off')
```

Why We Used These Techniques

Technique	Purpose
Pooling	Reduces image size & computation
Batch Normalization	Stabilizes learning & speeds training
Dropout	Prevents overfitting
Multiple Conv Blocks	Learns low → high level features

Conclusion

We successfully built a **custom CNN model** that classifies CIFAR-10 images with high accuracy using:

- Convolution
- Pooling
- Batch Normalization
- Dropout
- Softmax classifier