Import Libraries

```
# Deep Learning Frameworks
import tensorflow as tf
from tensorflow.keras import datasets, models, layers
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.utils import to_categorical

# Data Manipulation
import numpy as np

# Visualization
import matplotlib.pyplot as plt
import seaborn as sns

# Evaluation and Utilities
from sklearn.metrics import classification_report, confusion_matrix
```

```
# Load the CIFAR-10 dataset
(x_train, y_train), (x_test, y_test) = datasets.cifar10.load_data()
```

```
# Print the shape of the dataset
print(f"Training Data: {x_train.shape}, Training Labels: {y_train.shape}")
print(f"Test Data: {x_test.shape}, Test Labels: {y_test.shape}")
```

```
Training Data: (50000, 32, 32, 3), Training Labels: (50000, 1)
Test Data: (10000, 32, 32, 3), Test Labels: (10000, 1)
```

```
#Normalize the pixel values to the range [0, 1]
x_train = x_train.astype('float32') / 255.0
x_test = x_test.astype('float32') / 255.0
```

```
# Convert the labels to one-hot encoded vectors
from tensorflow.keras.utils import to_categorical
y_train = to_categorical(y_train, 10)
y_test = to_categorical(y_test, 10)
```

Build The CNN Model

```
model = Sequential([
    # Convolutional layers
    Conv2D(32, (3, 3), activation='relu', padding='same', input_shape=(32, 32, 3)),
    MaxPooling2D((2, 2)),
    Dropout(0.2),

    Conv2D(64, (3, 3), activation='relu', padding='same'),
    MaxPooling2D((2, 2)),
    Dropout(0.3),

    Conv2D(128, (3, 3), activation='relu', padding='same'),
    MaxPooling2D((2, 2)),
    Dropout(0.4),

    # Flatten and fully connected layers
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(10, activation='softmax')  # Output layer
])
```

Complie the model

```
model = Sequential([
    # Convolutional layers
    Conv2D(32, (3, 3), activation='relu', padding='same', input_shape=(32, 32, 3)),
    MaxPooling2D((2, 2)),
    Dropout(0.2),
```

```
    Conv2D(64, (3, 3), activation='relu', padding='same'),
    MaxPooling2D((2, 2)),
    Dropout(0.3),

    Conv2D(128, (3, 3), activation='relu', padding='same'),
    MaxPooling2D((2, 2)),
    Dropout(0.4),

    # Flatten and fully connected layers
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(10, activation='softmax')  # Output layer
])
```

Training Model

```
history = model.fit(x_train, y_train,
                    epochs=20,
                    batch_size=64,
                    validation_data=(x_test, y_test))
```

```
Epoch 1/20
782/782 ━━━━━━━━━━━━━━━━━━━━ 138s 173ms/step - accuracy: 0.2708 - loss: 1.9440 - val_accuracy: 0.5227 - val_loss: 1.3529
Epoch 2/20
782/782 ━━━━━━━━━━━━━━━━━━━━ 141s 172ms/step - accuracy: 0.4813 - loss: 1.4227 - val_accuracy: 0.5836 - val_loss: 1.1684
Epoch 3/20
782/782 ━━━━━━━━━━━━━━━━━━━━ 139s 178ms/step - accuracy: 0.5445 - loss: 1.2746 - val_accuracy: 0.6366 - val_loss: 1.0386
Epoch 4/20
782/782 ━━━━━━━━━━━━━━━━━━━━ 138s 173ms/step - accuracy: 0.5808 - loss: 1.1732 - val_accuracy: 0.6363 - val_loss: 1.0256
Epoch 5/20
782/782 ━━━━━━━━━━━━━━━━━━━━ 138s 177ms/step - accuracy: 0.6097 - loss: 1.1048 - val_accuracy: 0.6848 - val_loss: 0.9041
Epoch 6/20
782/782 ━━━━━━━━━━━━━━━━━━━━ 133s 170ms/step - accuracy: 0.6272 - loss: 1.0573 - val_accuracy: 0.7005 - val_loss: 0.8814
Epoch 7/20
782/782 ━━━━━━━━━━━━━━━━━━━━ 137s 175ms/step - accuracy: 0.6449 - loss: 1.0007 - val_accuracy: 0.7070 - val_loss: 0.8413
Epoch 8/20
782/782 ━━━━━━━━━━━━━━━━━━━━ 136s 174ms/step - accuracy: 0.6559 - loss: 0.9815 - val_accuracy: 0.7215 - val_loss: 0.8041
Epoch 9/20
782/782 ━━━━━━━━━━━━━━━━━━━━ 145s 177ms/step - accuracy: 0.6620 - loss: 0.9581 - val_accuracy: 0.7211 - val_loss: 0.8015
Epoch 10/20
782/782 ━━━━━━━━━━━━━━━━━━━━ 135s 173ms/step - accuracy: 0.6792 - loss: 0.9142 - val_accuracy: 0.6901 - val_loss: 0.8909
Epoch 11/20
782/782 ━━━━━━━━━━━━━━━━━━━━ 141s 172ms/step - accuracy: 0.6829 - loss: 0.9080 - val_accuracy: 0.7369 - val_loss: 0.7679
Epoch 12/20
782/782 ━━━━━━━━━━━━━━━━━━━━ 139s 169ms/step - accuracy: 0.6897 - loss: 0.8899 - val_accuracy: 0.7450 - val_loss: 0.7402
Epoch 13/20
782/782 ━━━━━━━━━━━━━━━━━━━━ 138s 177ms/step - accuracy: 0.6966 - loss: 0.8669 - val_accuracy: 0.7574 - val_loss: 0.7166
Epoch 14/20
782/782 ━━━━━━━━━━━━━━━━━━━━ 137s 171ms/step - accuracy: 0.7010 - loss: 0.8569 - val_accuracy: 0.7409 - val_loss: 0.7600
Epoch 15/20
782/782 ━━━━━━━━━━━━━━━━━━━━ 149s 180ms/step - accuracy: 0.7032 - loss: 0.8531 - val_accuracy: 0.7529 - val_loss: 0.7151
Epoch 16/20
782/782 ━━━━━━━━━━━━━━━━━━━━ 142s 180ms/step - accuracy: 0.7114 - loss: 0.8282 - val_accuracy: 0.7554 - val_loss: 0.7022
Epoch 17/20
782/782 ━━━━━━━━━━━━━━━━━━━━ 138s 176ms/step - accuracy: 0.7078 - loss: 0.8363 - val_accuracy: 0.7520 - val_loss: 0.7110
Epoch 18/20
782/782 ━━━━━━━━━━━━━━━━━━━━ 139s 178ms/step - accuracy: 0.7146 - loss: 0.8250 - val_accuracy: 0.7556 - val_loss: 0.6965
Epoch 19/20
782/782 ━━━━━━━━━━━━━━━━━━━━ 141s 176ms/step - accuracy: 0.7212 - loss: 0.8015 - val_accuracy: 0.7555 - val_loss: 0.7076
Epoch 20/20
782/782 ━━━━━━━━━━━━━━━━━━━━ 138s 171ms/step - accuracy: 0.7227 - loss: 0.8003 - val_accuracy: 0.7623 - val_loss: 0.6930
```

Evaluate the Model

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

```
import numpy as np
y_train = np.argmax(y_train, axis=1)  # Convert to integers
y_test = np.argmax(y_test, axis=1)    # Convert to integers
```

```
print(f"x_train shape: {x_train.shape}")  # Should be (50000, 32, 32, 3)
print(f"x_test shape: {x_test.shape}")    # Should be (10000, 32, 32, 3)
print(f"y_train shape: {y_train.shape}")  # Should match your loss function
```

```
print(f"y_test shape: {y_test.shape}")    # Should match your loss function
```

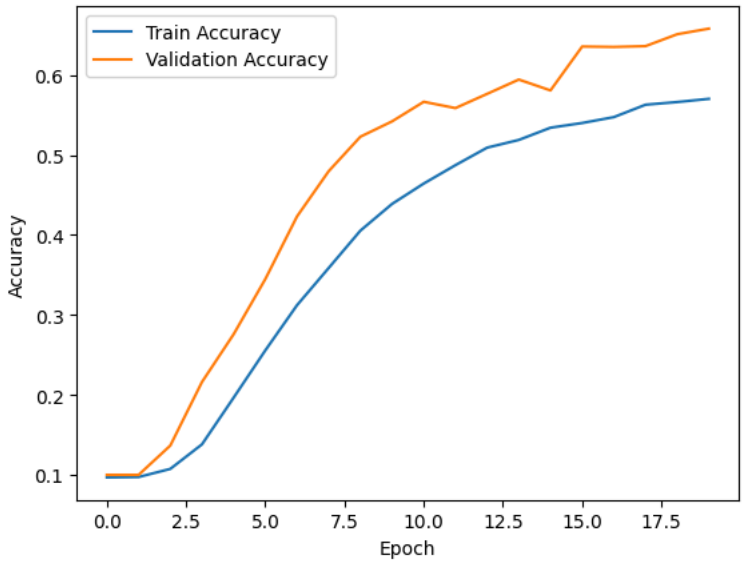
```
x_train shape: (50000, 32, 32, 3)
x_test shape: (10000, 32, 32, 3)
y_train shape: (50000,)
y_test shape: (10000,)
```

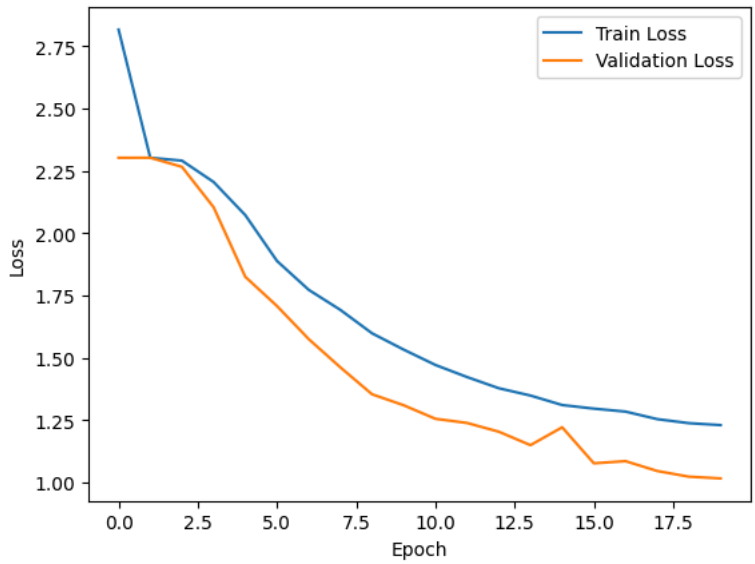
```
test_loss, test_accuracy = model.evaluate(x_test, y_test, verbose=2)
print(f"Test Accuracy: {test_accuracy:.2f}")
```

```
313/313 - 8s - 26ms/step - accuracy: 0.0996 - loss: 42.8872
Test Accuracy: 0.10
```

```
# ... (previous code) ...

# Training Model
history = model.fit(x_train, y_train,
                    epochs=20,
                    batch_size=64,
                    validation_data=(x_test, y_test))

# ... (rest of the code) ...
import matplotlib.pyplot as plt

# Plot accuracy
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.show()

# Plot loss (optional)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

```
Epoch 1/20
782/782 ━━━━━━━━━━━━━━━━━━━━ 134s 168ms/step - accuracy: 0.0977 - loss: 5.4734 - val_accuracy: 0.1000 - val_loss: 2.3026
Epoch 2/20
782/782 ━━━━━━━━━━━━━━━━━━━━ 130s 167ms/step - accuracy: 0.0971 - loss: 2.3030 - val_accuracy: 0.1000 - val_loss: 2.3026
Epoch 3/20
782/782 ━━━━━━━━━━━━━━━━━━━━ 142s 167ms/step - accuracy: 0.1008 - loss: 2.3012 - val_accuracy: 0.1366 - val_loss: 2.2662
Epoch 4/20
782/782 ━━━━━━━━━━━━━━━━━━━━ 142s 167ms/step - accuracy: 0.1358 - loss: 2.2194 - val_accuracy: 0.2161 - val_loss: 2.1040
Epoch 5/20
782/782 ━━━━━━━━━━━━━━━━━━━━ 142s 168ms/step - accuracy: 0.1762 - loss: 2.1259 - val_accuracy: 0.2759 - val_loss: 1.8249
Epoch 6/20
782/782 ━━━━━━━━━━━━━━━━━━━━ 140s 166ms/step - accuracy: 0.2454 - loss: 1.9169 - val_accuracy: 0.3449 - val_loss: 1.7078
Epoch 7/20
782/782 ━━━━━━━━━━━━━━━━━━━━ 142s 166ms/step - accuracy: 0.2994 - loss: 1.8002 - val_accuracy: 0.4232 - val_loss: 1.5753
Epoch 8/20
782/782 ━━━━━━━━━━━━━━━━━━━━ 141s 165ms/step - accuracy: 0.3541 - loss: 1.7038 - val_accuracy: 0.4798 - val_loss: 1.4618
Epoch 9/20
782/782 ━━━━━━━━━━━━━━━━━━━━ 141s 164ms/step - accuracy: 0.3972 - loss: 1.6159 - val_accuracy: 0.5229 - val_loss: 1.3538
Epoch 10/20
782/782 ━━━━━━━━━━━━━━━━━━━━ 143s 165ms/step - accuracy: 0.4341 - loss: 1.5357 - val_accuracy: 0.5421 - val_loss: 1.3097
Epoch 11/20
782/782 ━━━━━━━━━━━━━━━━━━━━ 142s 165ms/step - accuracy: 0.4583 - loss: 1.4814 - val_accuracy: 0.5666 - val_loss: 1.2554
Epoch 12/20
782/782 ━━━━━━━━━━━━━━━━━━━━ 129s 165ms/step - accuracy: 0.4842 - loss: 1.4311 - val_accuracy: 0.5587 - val_loss: 1.2387
Epoch 13/20
782/782 ━━━━━━━━━━━━━━━━━━━━ 142s 166ms/step - accuracy: 0.5045 - loss: 1.3963 - val_accuracy: 0.5765 - val_loss: 1.2036
Epoch 14/20
782/782 ━━━━━━━━━━━━━━━━━━━━ 129s 165ms/step - accuracy: 0.5140 - loss: 1.3552 - val_accuracy: 0.5944 - val_loss: 1.1497
Epoch 15/20
782/782 ━━━━━━━━━━━━━━━━━━━━ 142s 166ms/step - accuracy: 0.5339 - loss: 1.3134 - val_accuracy: 0.5808 - val_loss: 1.2214
Epoch 16/20
782/782 ━━━━━━━━━━━━━━━━━━━━ 140s 163ms/step - accuracy: 0.5377 - loss: 1.3030 - val_accuracy: 0.6357 - val_loss: 1.0769
Epoch 17/20
782/782 ━━━━━━━━━━━━━━━━━━━━ 143s 164ms/step - accuracy: 0.5419 - loss: 1.2993 - val_accuracy: 0.6352 - val_loss: 1.0855
Epoch 18/20
782/782 ━━━━━━━━━━━━━━━━━━━━ 128s 164ms/step - accuracy: 0.5605 - loss: 1.2586 - val_accuracy: 0.6362 - val_loss: 1.0461
Epoch 19/20
782/782 ━━━━━━━━━━━━━━━━━━━━ 142s 164ms/step - accuracy: 0.5685 - loss: 1.2335 - val_accuracy: 0.6511 - val_loss: 1.0232
Epoch 20/20
782/782 ━━━━━━━━━━━━━━━━━━━━ 129s 164ms/step - accuracy: 0.5724 - loss: 1.2189 - val_accuracy: 0.6581 - val_loss: 1.0163
```





Data Augmentation

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

datagen = ImageDataGenerator(
    rotation_range=15,
    width_shift_range=0.1,
    height_shift_range=0.1,
    horizontal_flip=True
)
datagen.fit(x_train)

# Train with augmented data
history = model.fit(datagen.flow(x_train, y_train, batch_size=64),
                    epochs=30,
                    validation_data=(x_test, y_test))
```

```
Epoch 1/30
/usr/local/lib/python3.10/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:122: UserWarning: Your `PyDataset`
  self._warn_if_super_not_called()
782/782 ━━━━━━━━━━━━━━━━━━━━ 160s 203ms/step - accuracy: 0.5150 - loss: 1.3821 - val_accuracy: 0.6207 - val_loss: 1.0815
Epoch 2/30
782/782 ━━━━━━━━━━━━━━━━━━━━ 204s 205ms/step - accuracy: 0.5260 - loss: 1.3596 - val_accuracy: 0.6275 - val_loss: 1.0836
Epoch 3/30
782/782 ━━━━━━━━━━━━━━━━━━━━ 159s 202ms/step - accuracy: 0.5236 - loss: 1.3601 - val_accuracy: 0.6413 - val_loss: 1.0466
Epoch 4/30
782/782 ━━━━━━━━━━━━━━━━━━━━ 162s 206ms/step - accuracy: 0.5309 - loss: 1.3399 - val_accuracy: 0.6321 - val_loss: 1.0790
Epoch 5/30
782/782 ━━━━━━━━━━━━━━━━━━━━ 199s 203ms/step - accuracy: 0.5323 - loss: 1.3440 - val_accuracy: 0.6336 - val_loss: 1.0571
Epoch 6/30
782/782 ━━━━━━━━━━━━━━━━━━━━ 159s 203ms/step - accuracy: 0.5409 - loss: 1.3091 - val_accuracy: 0.6446 - val_loss: 1.0210
Epoch 7/30
782/782 ━━━━━━━━━━━━━━━━━━━━ 159s 203ms/step - accuracy: 0.5358 - loss: 1.3162 - val_accuracy: 0.6416 - val_loss: 1.0570
Epoch 8/30
782/782 ━━━━━━━━━━━━━━━━━━━━ 201s 202ms/step - accuracy: 0.5427 - loss: 1.3158 - val_accuracy: 0.6276 - val_loss: 1.0636
Epoch 9/30
782/782 ━━━━━━━━━━━━━━━━━━━━ 167s 213ms/step - accuracy: 0.5420 - loss: 1.3129 - val_accuracy: 0.6538 - val_loss: 1.0248
Epoch 10/30
782/782 ━━━━━━━━━━━━━━━━━━━━ 159s 203ms/step - accuracy: 0.5421 - loss: 1.3190 - val_accuracy: 0.6295 - val_loss: 1.0528
Epoch 11/30
782/782 ━━━━━━━━━━━━━━━━━━━━ 161s 206ms/step - accuracy: 0.5479 - loss: 1.2983 - val_accuracy: 0.6233 - val_loss: 1.0747
Epoch 12/30
782/782 ━━━━━━━━━━━━━━━━━━━━ 194s 196ms/step - accuracy: 0.5489 - loss: 1.2933 - val_accuracy: 0.6647 - val_loss: 0.9871
Epoch 13/30
782/782 ━━━━━━━━━━━━━━━━━━━━ 155s 197ms/step - accuracy: 0.5350 - loss: 1.3247 - val_accuracy: 0.6546 - val_loss: 1.0606
Epoch 14/30
782/782 ━━━━━━━━━━━━━━━━━━━━ 154s 197ms/step - accuracy: 0.5482 - loss: 1.3063 - val_accuracy: 0.6435 - val_loss: 1.0673
Epoch 15/30
782/782 ━━━━━━━━━━━━━━━━━━━━ 154s 197ms/step - accuracy: 0.5513 - loss: 1.2912 - val_accuracy: 0.6492 - val_loss: 1.0904
Epoch 16/30
782/782 ━━━━━━━━━━━━━━━━━━━━ 218s 217ms/step - accuracy: 0.5511 - loss: 1.2901 - val_accuracy: 0.6542 - val_loss: 1.0067
Epoch 17/30
782/782 ━━━━━━━━━━━━━━━━━━━━ 207s 222ms/step - accuracy: 0.5532 - loss: 1.2946 - val_accuracy: 0.6284 - val_loss: 1.0904
Epoch 18/30
782/782 ━━━━━━━━━━━━━━━━━━━━ 185s 201ms/step - accuracy: 0.5583 - loss: 1.2822 - val_accuracy: 0.6543 - val_loss: 1.0015
Epoch 19/30
782/782 ━━━━━━━━━━━━━━━━━━━━ 156s 199ms/step - accuracy: 0.5565 - loss: 1.2877 - val_accuracy: 0.6635 - val_loss: 1.0067
```