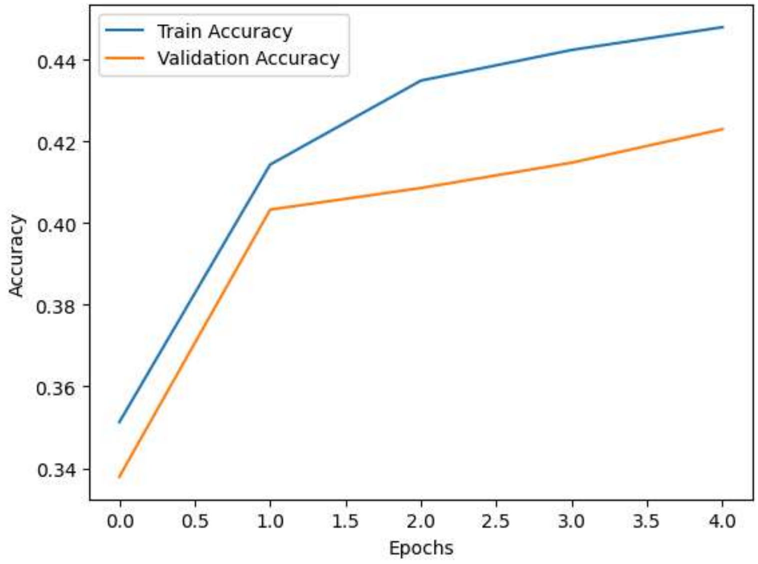
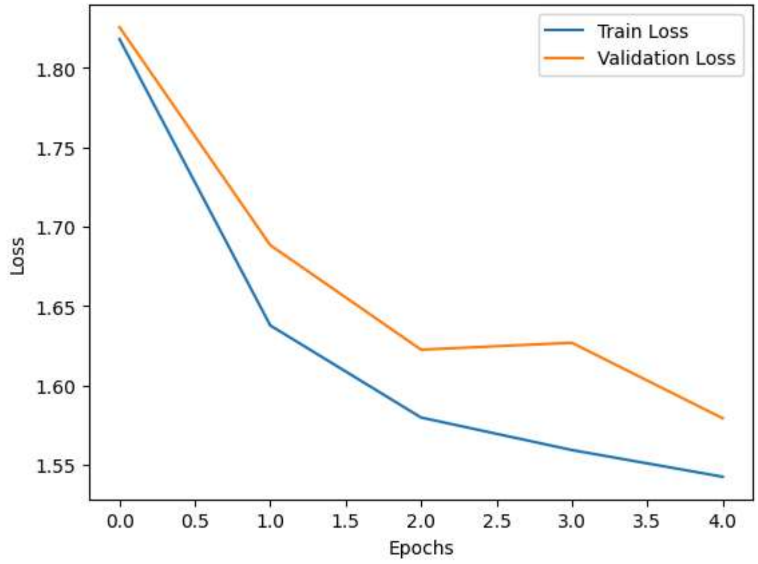
```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
import numpy as np
import matplotlib.pyplot as plt

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

# 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

# Flatten the images for a fully connected network
x_train = x_train.reshape(x_train.shape[0], -1)
x_test = x_test.reshape(x_test.shape[0], -1)

# Convert class labels to one-hot encoding
y_train = keras.utils.to_categorical(y_train, 10)
y_test = keras.utils.to_categorical(y_test, 10)

# Define the fully connected neural network model with optimizations
model = keras.Sequential([
    layers.Dense(1024, activation='relu', input_shape=(3072,)),
    layers.BatchNormalization(),
    layers.Dropout(0.3),
    layers.Dense(512, activation='relu'),
    layers.BatchNormalization(),
    layers.Dropout(0.3),
    layers.Dense(256, activation='relu'),
    layers.BatchNormalization(),
    layers.Dense(128, activation='relu'),
    layers.Dense(10, activation='softmax')  # Softmax activation for multi-class classification
])

# Compile the model with optimizations
model.compile(optimizer=keras.optimizers.Adam(learning_rate=0.001),
              loss='categorical_crossentropy',  # Multi-class classification loss
              metrics=['accuracy'])

# Train the model with early stopping and learning rate reduction
callback_list = [
    keras.callbacks.EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True),
    keras.callbacks.ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=3, verbose=1)
]

history = model.fit(x_train, y_train, epochs=5, batch_size=64, validation_data=(x_test, y_test), callbacks=callback_list)

# Evaluate the model
test_loss, test_acc = model.evaluate(x_test, y_test, verbose=2)
print("Test accuracy:", test_acc)

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

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

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170498071/170498071 ━━━━━━━━━━━━━━━━━━━━ 3s 0us/step
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` argumen
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Epoch 1/5
782/782 ━━━━━━━━━━━━━━━━━━━━ 59s 69ms/step - accuracy: 0.3070 - loss: 1.9574 - val_accuracy: 0.3379 - val_loss: 1.8257 - learning_rate:
Epoch 2/5
782/782 ━━━━━━━━━━━━━━━━━━━━ 54s 68ms/step - accuracy: 0.4109 - loss: 1.6516 - val_accuracy: 0.4033 - val_loss: 1.6884 - learning_rate:
Epoch 3/5
782/782 ━━━━━━━━━━━━━━━━━━━━ 84s 71ms/step - accuracy: 0.4305 - loss: 1.5933 - val_accuracy: 0.4086 - val_loss: 1.6225 - learning_rate:
Epoch 4/5
782/782 ━━━━━━━━━━━━━━━━━━━━ 82s 71ms/step - accuracy: 0.4448 - loss: 1.5542 - val_accuracy: 0.4148 - val_loss: 1.6269 - learning_rate:
Epoch 5/5
782/782 ━━━━━━━━━━━━━━━━━━━━ 53s 67ms/step - accuracy: 0.4531 - loss: 1.5288 - val_accuracy: 0.4230 - val_loss: 1.5793 - learning_rate:
313/313 - 3s - 9ms/step - accuracy: 0.4230 - loss: 1.5793
Test accuracy: 0.4230000078678131
```


Train Accuracy
Validation Accuracy
Accuracy
Epochs


Train Loss
Validation Loss
Loss
Epochs