In [ ]:

```python
# Import necessary libraries
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, LSTM, RepeatVector
from tensorflow.keras.datasets import cifar10
from sklearn.model_selection import train_test_split
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping

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

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

# Reshape the data for LSTM input (sequence length 32, features 32*3 for RGB channels)
# Each image is reshaped into a sequence of 32 rows and 96 features (32*3 for RGB)
x_train = x_train.reshape(-1, 32, 32 * 3)
x_test = x_test.reshape(-1, 32, 32 * 3)

# Split the training data into training and validation sets (80% training, 20% validation)
x_train, x_val, y_train, y_val = train_test_split(x_train, x_train, test_size=0.2, random_state=42)

# Define the LSTM autoencoder model with increased latent space size and added layers
input_img = Input(shape=(32, 32 * 3))  # Input shape: sequence length of 32, 32*3 features

# Encoder with two LSTM layers
encoded = LSTM(512, activation='tanh', return_sequences=True)(input_img)  # First LSTM layer
encoded = LSTM(256, activation='tanh')(encoded)  # Second LSTM layer

# Latent space representation
latent_space = RepeatVector(32)(encoded)

# Decoder with two LSTM layers
decoded = LSTM(256, activation='tanh', return_sequences=True)(latent_space)  # First LSTM layer in the decoder
decoded = LSTM(32 * 3, activation='sigmoid', return_sequences=True)(decoded)  # Second LSTM layer in the decode

# Create the autoencoder model
autoencoder = Model(input_img, decoded)

# Compile the model with Adam optimizer and a reduced learning rate
autoencoder.compile(optimizer=Adam(learning_rate=0.0001), loss='mse')

# Set up early stopping to monitor validation loss and stop training if no improvement after 10 epochs
early_stopping = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)

# Train the model for 200 epochs, using early stopping to stop when validation loss stops improving
history = autoencoder.fit(x_train, x_train, epochs=200, batch_size=128, validation_data=(x_val, x_val), callbac

# Plot the training and validation loss
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()

# Evaluate the model on the test set
test_loss = autoencoder.evaluate(x_test, x_test)
print(f"Test Loss: {test_loss}")

# Generate reconstructed images using the trained autoencoder
reconstructed = autoencoder.predict(x_test)

# Visualize the original and reconstructed images
n = 5  # Number of images to display
plt.figure(figsize=(15, 5))
for i in range(n):
    # Original
    ax = plt.subplot(2, n, i + 1)
    plt.imshow(x_test[i].reshape(32, 32, 3))
    plt.title("Original")
    plt.axis("off")

    # Reconstructed
    ax = plt.subplot(2, n, i + 1 + n)
    plt.imshow(reconstructed[i].reshape(32, 32, 3))
    plt.title("Reconstructed")
    plt.axis("off")
```

```
plt.show()
```

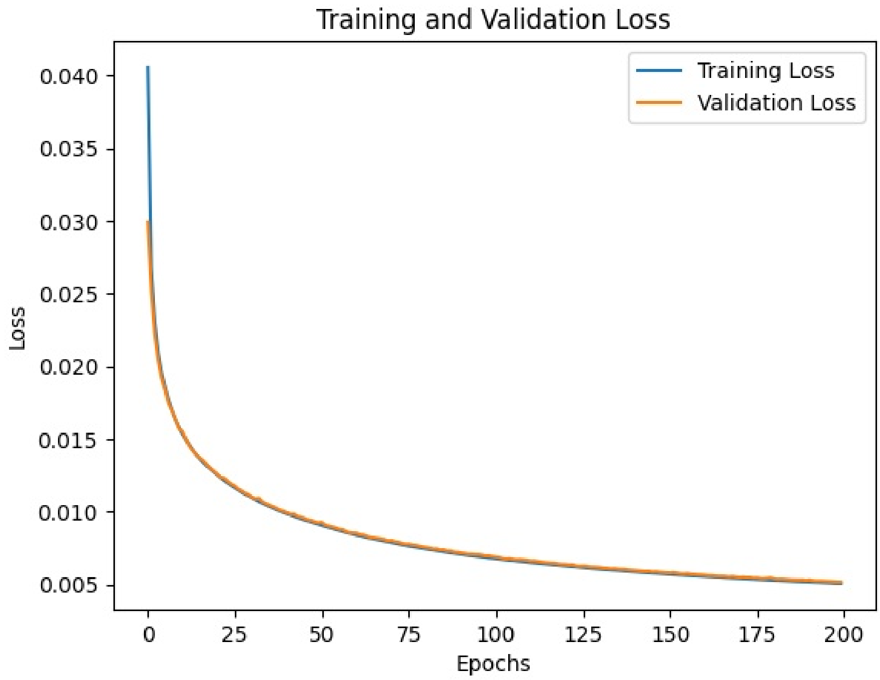
```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170498071/170498071 ━━━━━━━━━━━━━━━━━━━━ 4s 0us/step
Epoch 1/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 104ms/step - loss: 0.0525 - val_loss: 0.0299
Epoch 2/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 33s 91ms/step - loss: 0.0281 - val_loss: 0.0253
Epoch 3/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 25s 79ms/step - loss: 0.0238 - val_loss: 0.0222
Epoch 4/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 78ms/step - loss: 0.0214 - val_loss: 0.0205
Epoch 5/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 42s 80ms/step - loss: 0.0199 - val_loss: 0.0192
Epoch 6/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 79ms/step - loss: 0.0188 - val_loss: 0.0184
Epoch 7/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 42s 83ms/step - loss: 0.0179 - val_loss: 0.0174
Epoch 8/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 40s 81ms/step - loss: 0.0170 - val_loss: 0.0169
Epoch 9/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 40s 78ms/step - loss: 0.0165 - val_loss: 0.0163
Epoch 10/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 25s 79ms/step - loss: 0.0160 - val_loss: 0.0157
Epoch 11/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 80ms/step - loss: 0.0154 - val_loss: 0.0155
Epoch 12/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 80ms/step - loss: 0.0150 - val_loss: 0.0150
Epoch 13/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 79ms/step - loss: 0.0147 - val_loss: 0.0145
Epoch 14/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 43s 85ms/step - loss: 0.0142 - val_loss: 0.0142
Epoch 15/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 39s 80ms/step - loss: 0.0139 - val_loss: 0.0139
Epoch 16/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 40s 77ms/step - loss: 0.0136 - val_loss: 0.0137
Epoch 17/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 79ms/step - loss: 0.0135 - val_loss: 0.0135
Epoch 18/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 80ms/step - loss: 0.0132 - val_loss: 0.0132
Epoch 19/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 79ms/step - loss: 0.0130 - val_loss: 0.0130
Epoch 20/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 25s 79ms/step - loss: 0.0128 - val_loss: 0.0128
Epoch 21/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 42s 83ms/step - loss: 0.0125 - val_loss: 0.0126
Epoch 22/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 25s 80ms/step - loss: 0.0124 - val_loss: 0.0124
Epoch 23/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 25s 81ms/step - loss: 0.0122 - val_loss: 0.0123
Epoch 24/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 25s 81ms/step - loss: 0.0120 - val_loss: 0.0121
Epoch 25/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 81ms/step - loss: 0.0118 - val_loss: 0.0119
Epoch 26/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 82ms/step - loss: 0.0117 - val_loss: 0.0118
Epoch 27/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 83ms/step - loss: 0.0116 - val_loss: 0.0116
Epoch 28/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 82ms/step - loss: 0.0114 - val_loss: 0.0115
Epoch 29/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 81ms/step - loss: 0.0113 - val_loss: 0.0113
Epoch 30/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 81ms/step - loss: 0.0111 - val_loss: 0.0112
Epoch 31/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 40s 80ms/step - loss: 0.0110 - val_loss: 0.0110
Epoch 32/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 42s 81ms/step - loss: 0.0108 - val_loss: 0.0109
Epoch 33/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 25s 80ms/step - loss: 0.0107 - val_loss: 0.0109
Epoch 34/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 80ms/step - loss: 0.0106 - val_loss: 0.0107
Epoch 35/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 81ms/step - loss: 0.0105 - val_loss: 0.0106
Epoch 36/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 82ms/step - loss: 0.0104 - val_loss: 0.0104
Epoch 37/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 82ms/step - loss: 0.0103 - val_loss: 0.0104
Epoch 38/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 81ms/step - loss: 0.0102 - val_loss: 0.0102
Epoch 39/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 82ms/step - loss: 0.0101 - val_loss: 0.0101
Epoch 40/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 40s 80ms/step - loss: 0.0100 - val_loss: 0.0101
```

```
Epoch 41/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 81ms/step - loss: 0.0099 - val_loss: 0.0099
Epoch 42/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 42s 84ms/step - loss: 0.0098 - val_loss: 0.0098
Epoch 43/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 25s 80ms/step - loss: 0.0097 - val_loss: 0.0099
Epoch 44/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 80ms/step - loss: 0.0096 - val_loss: 0.0097
Epoch 45/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 80ms/step - loss: 0.0095 - val_loss: 0.0096
Epoch 46/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 80ms/step - loss: 0.0095 - val_loss: 0.0096
Epoch 47/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 81ms/step - loss: 0.0094 - val_loss: 0.0094
Epoch 48/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 80ms/step - loss: 0.0093 - val_loss: 0.0094
Epoch 49/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 42s 83ms/step - loss: 0.0092 - val_loss: 0.0093
Epoch 50/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 40s 80ms/step - loss: 0.0092 - val_loss: 0.0092
Epoch 51/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 81ms/step - loss: 0.0091 - val_loss: 0.0093
Epoch 52/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 80ms/step - loss: 0.0090 - val_loss: 0.0091
Epoch 53/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 80ms/step - loss: 0.0090 - val_loss: 0.0090
Epoch 54/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 81ms/step - loss: 0.0089 - val_loss: 0.0090
Epoch 55/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 81ms/step - loss: 0.0088 - val_loss: 0.0089
Epoch 56/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 81ms/step - loss: 0.0087 - val_loss: 0.0088
Epoch 57/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 80ms/step - loss: 0.0087 - val_loss: 0.0088
Epoch 58/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 81ms/step - loss: 0.0086 - val_loss: 0.0086
Epoch 59/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 81ms/step - loss: 0.0085 - val_loss: 0.0086
Epoch 60/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 40s 79ms/step - loss: 0.0085 - val_loss: 0.0085
Epoch 61/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 81ms/step - loss: 0.0084 - val_loss: 0.0085
Epoch 62/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 42s 85ms/step - loss: 0.0084 - val_loss: 0.0084
Epoch 63/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 39s 80ms/step - loss: 0.0083 - val_loss: 0.0084
Epoch 64/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 81ms/step - loss: 0.0083 - val_loss: 0.0083
Epoch 65/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 80ms/step - loss: 0.0081 - val_loss: 0.0082
Epoch 66/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 25s 81ms/step - loss: 0.0082 - val_loss: 0.0082
Epoch 67/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 46s 96ms/step - loss: 0.0081 - val_loss: 0.0082
Epoch 68/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 36s 81ms/step - loss: 0.0081 - val_loss: 0.0081
Epoch 69/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 42s 85ms/step - loss: 0.0080 - val_loss: 0.0081
Epoch 70/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 39s 80ms/step - loss: 0.0080 - val_loss: 0.0080
Epoch 71/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 25s 80ms/step - loss: 0.0079 - val_loss: 0.0080
Epoch 72/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 25s 80ms/step - loss: 0.0079 - val_loss: 0.0079
Epoch 73/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 80ms/step - loss: 0.0078 - val_loss: 0.0079
Epoch 74/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 25s 81ms/step - loss: 0.0078 - val_loss: 0.0078
Epoch 75/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 25s 80ms/step - loss: 0.0077 - val_loss: 0.0078
Epoch 76/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 25s 80ms/step - loss: 0.0076 - val_loss: 0.0078
Epoch 77/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 42s 85ms/step - loss: 0.0076 - val_loss: 0.0077
Epoch 78/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 40s 81ms/step - loss: 0.0076 - val_loss: 0.0077
Epoch 79/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 80ms/step - loss: 0.0075 - val_loss: 0.0076
Epoch 80/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 25s 81ms/step - loss: 0.0075 - val_loss: 0.0076
Epoch 81/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 25s 80ms/step - loss: 0.0075 - val_loss: 0.0076
Epoch 82/200
```

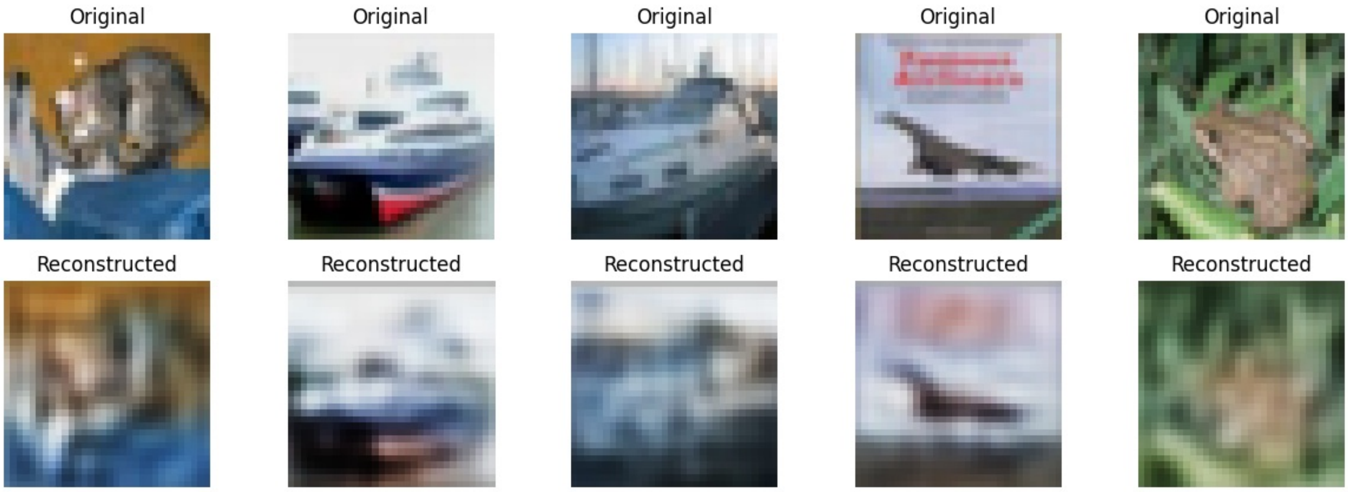
```
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 80ms/step - loss: 0.0074 - val_loss: 0.0075
Epoch 83/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 81ms/step - loss: 0.0074 - val_loss: 0.0074
Epoch 84/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 25s 80ms/step - loss: 0.0073 - val_loss: 0.0075
Epoch 85/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 42s 84ms/step - loss: 0.0073 - val_loss: 0.0074
Epoch 86/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 40s 81ms/step - loss: 0.0073 - val_loss: 0.0074
Epoch 87/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 80ms/step - loss: 0.0072 - val_loss: 0.0073
Epoch 88/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 25s 80ms/step - loss: 0.0072 - val_loss: 0.0073
Epoch 89/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 25s 80ms/step - loss: 0.0071 - val_loss: 0.0072
Epoch 90/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 81ms/step - loss: 0.0071 - val_loss: 0.0072
Epoch 91/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 25s 81ms/step - loss: 0.0071 - val_loss: 0.0072
Epoch 92/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 80ms/step - loss: 0.0070 - val_loss: 0.0071
Epoch 93/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 81ms/step - loss: 0.0070 - val_loss: 0.0071
Epoch 94/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 81ms/step - loss: 0.0070 - val_loss: 0.0071
Epoch 95/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 40s 78ms/step - loss: 0.0069 - val_loss: 0.0071
Epoch 96/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 80ms/step - loss: 0.0069 - val_loss: 0.0071
Epoch 97/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 81ms/step - loss: 0.0069 - val_loss: 0.0070
Epoch 98/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 82ms/step - loss: 0.0068 - val_loss: 0.0070
Epoch 99/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 27s 85ms/step - loss: 0.0068 - val_loss: 0.0070
Epoch 100/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 40s 81ms/step - loss: 0.0068 - val_loss: 0.0069
Epoch 101/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 82ms/step - loss: 0.0067 - val_loss: 0.0069
Epoch 102/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 80ms/step - loss: 0.0068 - val_loss: 0.0069
Epoch 103/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 81ms/step - loss: 0.0067 - val_loss: 0.0068
Epoch 104/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 82ms/step - loss: 0.0067 - val_loss: 0.0068
Epoch 105/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 25s 80ms/step - loss: 0.0067 - val_loss: 0.0068
Epoch 106/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 81ms/step - loss: 0.0067 - val_loss: 0.0067
Epoch 107/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 80ms/step - loss: 0.0066 - val_loss: 0.0067
Epoch 108/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 80ms/step - loss: 0.0066 - val_loss: 0.0067
Epoch 109/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 80ms/step - loss: 0.0066 - val_loss: 0.0067
Epoch 110/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 80ms/step - loss: 0.0065 - val_loss: 0.0067
Epoch 111/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 25s 81ms/step - loss: 0.0065 - val_loss: 0.0066
Epoch 112/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 42s 83ms/step - loss: 0.0065 - val_loss: 0.0066
Epoch 113/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 40s 81ms/step - loss: 0.0065 - val_loss: 0.0066
Epoch 114/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 80ms/step - loss: 0.0064 - val_loss: 0.0065
Epoch 115/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 80ms/step - loss: 0.0064 - val_loss: 0.0065
Epoch 116/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 25s 79ms/step - loss: 0.0064 - val_loss: 0.0065
Epoch 117/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 80ms/step - loss: 0.0063 - val_loss: 0.0065
Epoch 118/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 25s 80ms/step - loss: 0.0063 - val_loss: 0.0065
Epoch 119/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 25s 79ms/step - loss: 0.0063 - val_loss: 0.0064
Epoch 120/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 80ms/step - loss: 0.0063 - val_loss: 0.0064
Epoch 121/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 80ms/step - loss: 0.0063 - val_loss: 0.0064
Epoch 122/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 80ms/step - loss: 0.0062 - val_loss: 0.0063
Epoch 123/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 26s 84ms/step - loss: 0.0062 - val_loss: 0.0064
```

```
Epoch 124/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 25s 80ms/step - loss: 0.0062 - val_loss: 0.0063
Epoch 125/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 78ms/step - loss: 0.0061 - val_loss: 0.0063
Epoch 126/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 80ms/step - loss: 0.0062 - val_loss: 0.0063
Epoch 127/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 25s 79ms/step - loss: 0.0061 - val_loss: 0.0063
Epoch 128/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 80ms/step - loss: 0.0061 - val_loss: 0.0062
Epoch 129/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 25s 79ms/step - loss: 0.0061 - val_loss: 0.0062
Epoch 130/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 42s 83ms/step - loss: 0.0061 - val_loss: 0.0062
Epoch 131/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 25s 79ms/step - loss: 0.0061 - val_loss: 0.0062
Epoch 132/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 79ms/step - loss: 0.0061 - val_loss: 0.0061
Epoch 133/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 78ms/step - loss: 0.0060 - val_loss: 0.0061
Epoch 134/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 25s 78ms/step - loss: 0.0060 - val_loss: 0.0061
Epoch 135/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 25s 78ms/step - loss: 0.0060 - val_loss: 0.0061
Epoch 136/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 80ms/step - loss: 0.0060 - val_loss: 0.0061
Epoch 137/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 79ms/step - loss: 0.0060 - val_loss: 0.0060
Epoch 138/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 40s 77ms/step - loss: 0.0059 - val_loss: 0.0060
Epoch 139/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 42s 80ms/step - loss: 0.0059 - val_loss: 0.0060
Epoch 140/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 80ms/step - loss: 0.0059 - val_loss: 0.0060
Epoch 141/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 25s 79ms/step - loss: 0.0059 - val_loss: 0.0060
Epoch 142/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 79ms/step - loss: 0.0058 - val_loss: 0.0059
Epoch 143/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 80ms/step - loss: 0.0058 - val_loss: 0.0059
Epoch 144/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 43s 85ms/step - loss: 0.0058 - val_loss: 0.0059
Epoch 145/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 39s 80ms/step - loss: 0.0058 - val_loss: 0.0059
Epoch 146/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 80ms/step - loss: 0.0058 - val_loss: 0.0059
Epoch 147/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 25s 79ms/step - loss: 0.0058 - val_loss: 0.0059
Epoch 148/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 79ms/step - loss: 0.0058 - val_loss: 0.0058
Epoch 149/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 25s 80ms/step - loss: 0.0057 - val_loss: 0.0058
Epoch 150/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 80ms/step - loss: 0.0057 - val_loss: 0.0058
Epoch 151/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 42s 84ms/step - loss: 0.0057 - val_loss: 0.0058
Epoch 152/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 40s 81ms/step - loss: 0.0057 - val_loss: 0.0058
Epoch 153/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 81ms/step - loss: 0.0057 - val_loss: 0.0058
Epoch 154/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 81ms/step - loss: 0.0057 - val_loss: 0.0057
Epoch 155/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 40s 79ms/step - loss: 0.0057 - val_loss: 0.0057
Epoch 156/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 25s 79ms/step - loss: 0.0056 - val_loss: 0.0058
Epoch 157/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 81ms/step - loss: 0.0056 - val_loss: 0.0057
Epoch 158/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 82ms/step - loss: 0.0056 - val_loss: 0.0057
Epoch 159/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 80ms/step - loss: 0.0056 - val_loss: 0.0057
Epoch 160/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 80ms/step - loss: 0.0056 - val_loss: 0.0057
Epoch 161/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 81ms/step - loss: 0.0055 - val_loss: 0.0057
Epoch 162/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 25s 81ms/step - loss: 0.0055 - val_loss: 0.0056
Epoch 163/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 80ms/step - loss: 0.0055 - val_loss: 0.0056
Epoch 164/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 26s 83ms/step - loss: 0.0055 - val_loss: 0.0056
Epoch 165/200
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 40s 80ms/step - loss: 0.0055 - val_loss: 0.0056
Epoch 166/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 80ms/step - loss: 0.0055 - val_loss: 0.0056
Epoch 167/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 80ms/step - loss: 0.0055 - val_loss: 0.0056
Epoch 168/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 80ms/step - loss: 0.0054 - val_loss: 0.0055
Epoch 169/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 80ms/step - loss: 0.0054 - val_loss: 0.0056
Epoch 170/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 42s 84ms/step - loss: 0.0054 - val_loss: 0.0055
Epoch 171/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 40s 79ms/step - loss: 0.0054 - val_loss: 0.0055
Epoch 172/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 25s 79ms/step - loss: 0.0054 - val_loss: 0.0055
Epoch 173/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 25s 80ms/step - loss: 0.0054 - val_loss: 0.0055
Epoch 174/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 25s 80ms/step - loss: 0.0054 - val_loss: 0.0055
Epoch 175/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 25s 79ms/step - loss: 0.0054 - val_loss: 0.0054
Epoch 176/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 80ms/step - loss: 0.0054 - val_loss: 0.0055
Epoch 177/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 25s 80ms/step - loss: 0.0053 - val_loss: 0.0054
Epoch 178/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 26s 82ms/step - loss: 0.0053 - val_loss: 0.0054
Epoch 179/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 81ms/step - loss: 0.0053 - val_loss: 0.0054
Epoch 180/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 81ms/step - loss: 0.0053 - val_loss: 0.0055
Epoch 181/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 26s 83ms/step - loss: 0.0053 - val_loss: 0.0054
Epoch 182/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 83ms/step - loss: 0.0053 - val_loss: 0.0054
Epoch 183/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 40s 81ms/step - loss: 0.0053 - val_loss: 0.0054
Epoch 184/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 25s 80ms/step - loss: 0.0052 - val_loss: 0.0054
Epoch 185/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 42s 84ms/step - loss: 0.0052 - val_loss: 0.0053
Epoch 186/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 40s 81ms/step - loss: 0.0052 - val_loss: 0.0053
Epoch 187/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 81ms/step - loss: 0.0052 - val_loss: 0.0053
Epoch 188/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 81ms/step - loss: 0.0052 - val_loss: 0.0053
Epoch 189/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 25s 80ms/step - loss: 0.0052 - val_loss: 0.0053
Epoch 190/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 40s 78ms/step - loss: 0.0052 - val_loss: 0.0053
Epoch 191/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 42s 81ms/step - loss: 0.0052 - val_loss: 0.0053
Epoch 192/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 80ms/step - loss: 0.0051 - val_loss: 0.0053
Epoch 193/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 80ms/step - loss: 0.0051 - val_loss: 0.0052
Epoch 194/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 81ms/step - loss: 0.0051 - val_loss: 0.0052
Epoch 195/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 80ms/step - loss: 0.0051 - val_loss: 0.0052
Epoch 196/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 80ms/step - loss: 0.0051 - val_loss: 0.0052
Epoch 197/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 81ms/step - loss: 0.0051 - val_loss: 0.0052
Epoch 198/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 80ms/step - loss: 0.0051 - val_loss: 0.0052
Epoch 199/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 81ms/step - loss: 0.0051 - val_loss: 0.0052
Epoch 200/200
313/313 ━━━━━━━━━━━━━━━━━━━━ 41s 80ms/step - loss: 0.0051 - val_loss: 0.0052
```




```
313/313 ━━━━━━━━━━━━━━━━━━━━ 6s 18ms/step - loss: 0.0052
Test Loss: 0.005131924524903297
313/313 ━━━━━━━━━━━━━━━━━━━━ 7s 22ms/step
```



The Original images to understand what the classes of images look like

In [3]:

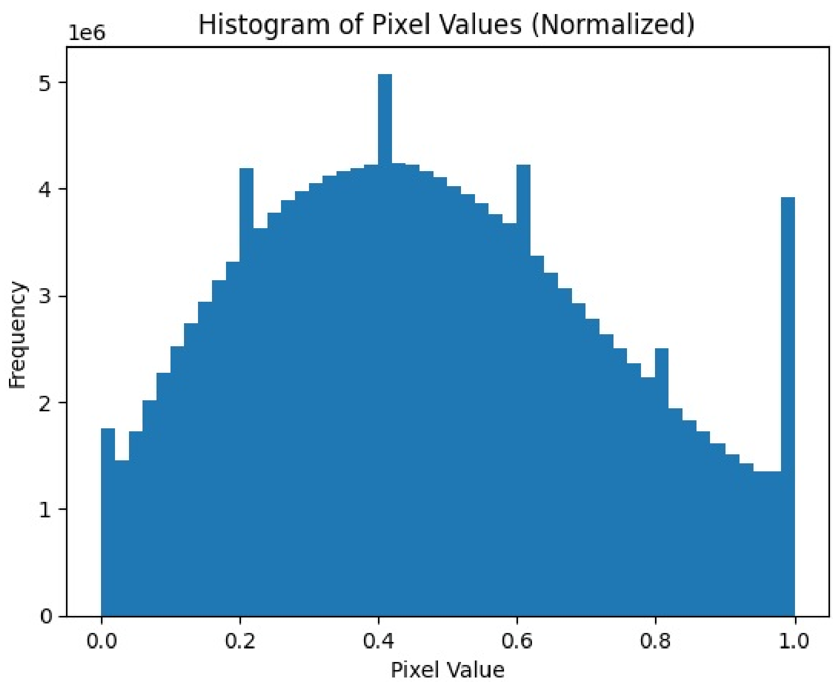
```
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.datasets import cifar10

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

# Get unique labels (classes)
unique_labels = np.unique(y_train)

# Create a figure and axes
plt.figure(figsize=(15, 3))

# Iterate through unique labels
```

```python
for i in range(10):
  # Find an image with the current label
  index = np.where(y_train == i)[0][0]
  image = x_train[index]

  # Create a subplot for each image
  ax = plt.subplot(1, 10, i + 1)
  plt.imshow(image)
  plt.title(f"Class {i}")
  plt.axis("off")

# Show the plot
plt.show()
```

Pixel representation of the images after normalization

In [5]:

```python
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.datasets import cifar10

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

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

# Flatten the normalized image data
flattened_images = x_train.reshape(-1)

# Create a histogram of the pixel values
plt.hist(flattened_images, bins=50)  # Adjust the number of bins for better visualization
plt.xlabel('Pixel Value')
plt.ylabel('Frequency')
plt.title('Histogram of Pixel Values (Normalized)')
plt.show()
```



Mean variances of images to access the brightness

In [8]:

```python
import numpy as np
import pandas as pd

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

# Normalize the data: scale pixel values to the range [0, 1]
```

```
x_train = x_train.astype('float32') / 255.0

# Calculate mean and variance for each image
means = []
variances = []

for image in x_train:
  means.append(np.mean(image))
  variances.append(np.var(image))

# Create a Pandas DataFrame to store and display the data
data = {'Mean Pixel Intensity (Normalized)': means, 'Variance': variances}
df = pd.DataFrame(data)

# Display the DataFrame
print(df)
# Optionally, you can round the values for better readability
# df = df.round(2)

# Display the rounded DataFrame
print(df.round(2))
```

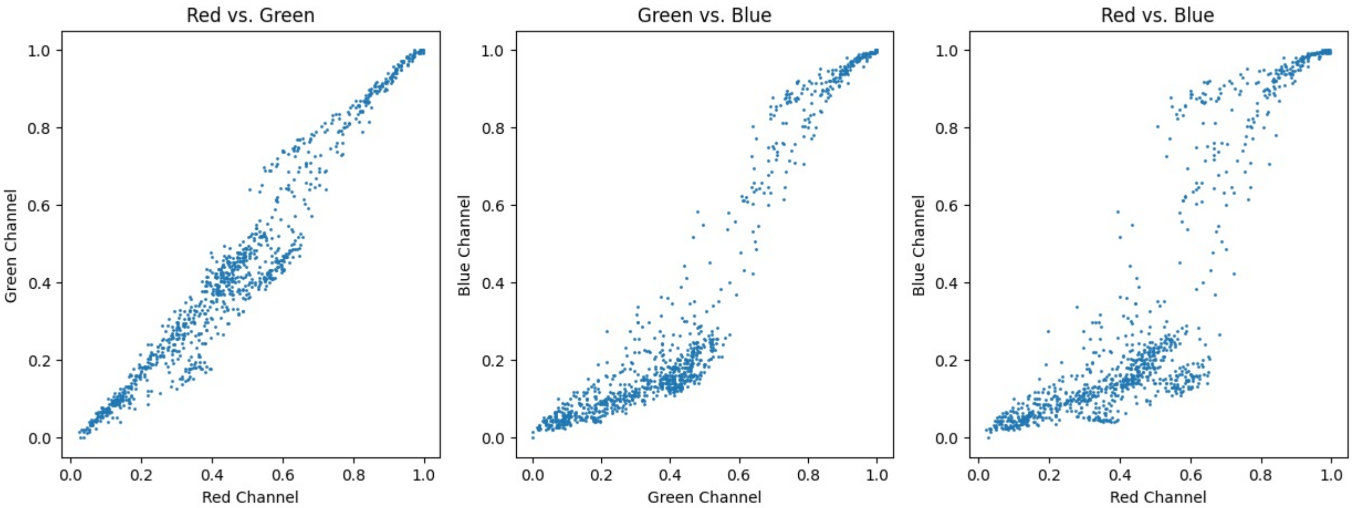
```
       Mean Pixel Intensity (Normalized)  Variance
0                               0.405676  0.041542
1                               0.511166  0.056251
2                               0.524473  0.104832
3                               0.314605  0.020457
4                               0.405784  0.056382
...                                  ...       ...
49995                           0.594171  0.058287
49996                           0.572502  0.096438
49997                           0.411914  0.087331
49998                           0.664484  0.050202
49999                           0.517177  0.051731

[50000 rows x 2 columns]
       Mean Pixel Intensity (Normalized)  Variance
0                                   0.41      0.04
1                                   0.51      0.06
2                                   0.52      0.10
3                                   0.31      0.02
4                                   0.41      0.06
...                                  ...       ...
49995                               0.59      0.06
49996                               0.57      0.10
49997                               0.41      0.09
49998                               0.66      0.05
49999                               0.52      0.05

[50000 rows x 2 columns]
```

Relationships between pixel values accross the RGB colour channels

In [9]:

```
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.datasets import cifar10

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

# Normalize the data (optional, but helps with visualization)
x_train = x_train.astype('float32') / 255.0

# Choose a random image from the training set
image_index = np.random.randint(0, len(x_train))
image = x_train[image_index]

# Extract the red, green, and blue channels
red_channel = image[:, :, 0]
green_channel = image[:, :, 1]
blue_channel = image[:, :, 2]

# Create a scatter plot for Red vs Green, Green vs Blue, and Red vs Blue
plt.figure(figsize=(15, 5))

plt.subplot(1, 3, 1)
plt.scatter(red_channel.flatten(), green_channel.flatten(), s=1)  # s=1 for smaller marker size
plt.xlabel('Red Channel')
plt.ylabel('Green Channel')
plt.title('Red vs. Green')
```

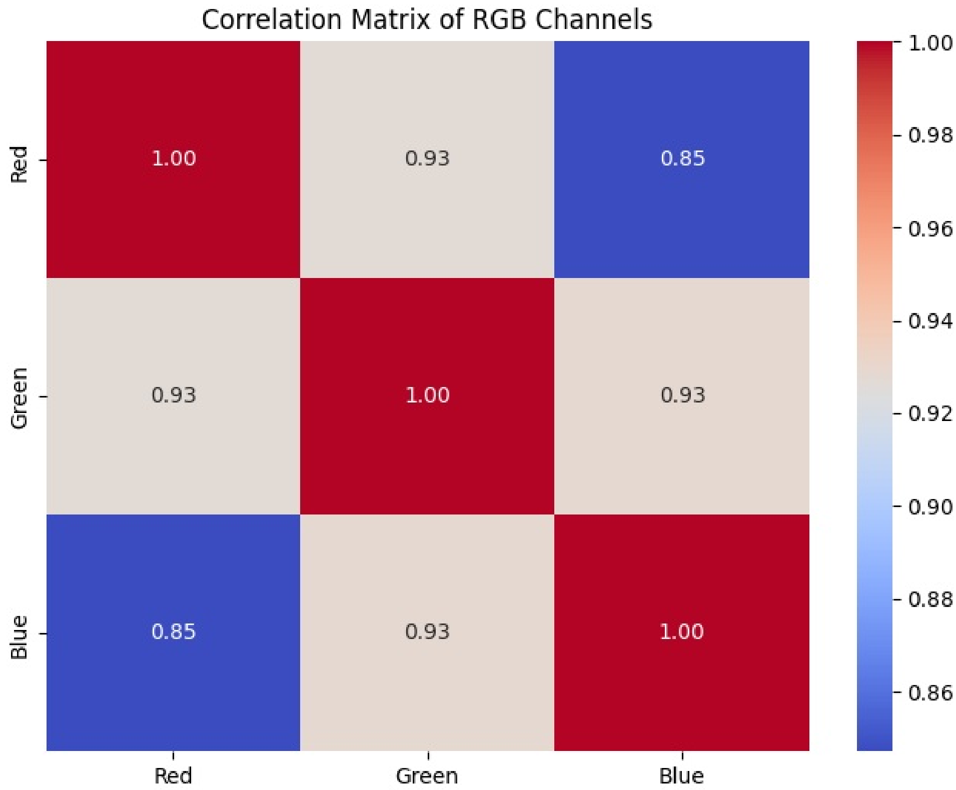
```
plt.subplot(1, 3, 2)
plt.scatter(green_channel.flatten(), blue_channel.flatten(), s=1)
plt.xlabel('Green Channel')
plt.ylabel('Blue Channel')
plt.title('Green vs. Blue')

plt.subplot(1, 3, 3)
plt.scatter(red_channel.flatten(), blue_channel.flatten(), s=1)
plt.xlabel('Red Channel')
plt.ylabel('Blue Channel')
plt.title('Red vs. Blue')

plt.show()

# You can also calculate correlation coefficients between the channels to quantify the relationships.
correlation_rg = np.corrcoef(red_channel.flatten(), green_channel.flatten())[0, 1]
correlation_gb = np.corrcoef(green_channel.flatten(), blue_channel.flatten())[0, 1]
correlation_rb = np.corrcoef(red_channel.flatten(), blue_channel.flatten())[0, 1]

print(f"Correlation between Red and Green: {correlation_rg}")
print(f"Correlation between Green and Blue: {correlation_gb}")
print(f"Correlation between Red and Blue: {correlation_rb}")
```




```
Correlation between Red and Green: 0.9741234363831361
Correlation between Green and Blue: 0.9332290393269796
Correlation between Red and Blue: 0.8661948968530371
```

In [10]:

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
from tensorflow.keras.datasets import cifar10

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

# Normalize the data (optional, but helps with visualization)
x_train = x_train.astype('float32') / 255.0

# Calculate correlation matrix for RGB channels
correlation_matrix = np.zeros((3, 3))
for i in range(len(x_train)):
    image = x_train[i]
    red_channel = image[:, :, 0].flatten()
    green_channel = image[:, :, 1].flatten()
    blue_channel = image[:, :, 2].flatten()
    channels = np.stack((red_channel, green_channel, blue_channel))
    correlation_matrix += np.corrcoef(channels)

correlation_matrix /= len(x_train)

# Create a DataFrame for the correlation matrix
df_correlation = pd.DataFrame(correlation_matrix, columns=['Red', 'Green', 'Blue'], index=['Red', 'Green', 'Blu

# Plot the correlation matrix using Seaborn
plt.figure(figsize=(8, 6))
sns.heatmap(df_correlation, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Matrix of RGB Channels')
plt.show()
```



Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js