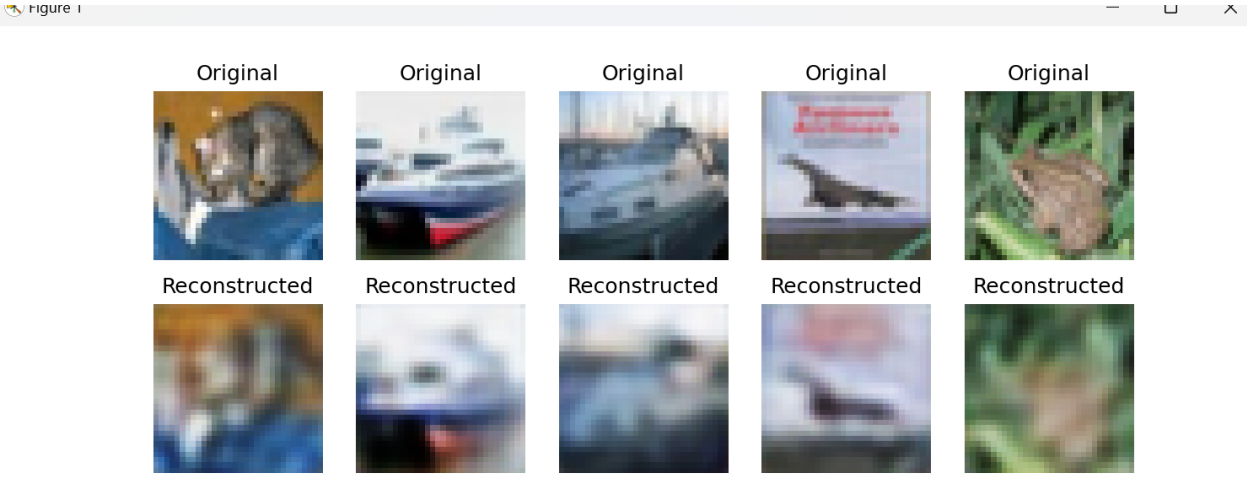
Charlie Misbach
CSCI-3470
Assignment #3
Due 4/7/2025

```
# Question #1: Implement a convolutional Autoencoder and discuss your
# discovery/thinking during the implementation.

# Needed imports
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Conv2D, MaxPooling2D,
UpSampling2D
from tensorflow.keras.optimizers import Adam

# Creation of the convolutional autoencoder
# 1. Load the CIFAR-10 dataset
(x_train, _), (x_test, _) = cifar10.load_data()
x_train = x_train.astype('float32') / 255.
x_test = x_test.astype('float32') / 255.

# Using smaller subset for quicker training
x_train_smaller = x_train[:10000]
x_test_smaller = x_test[:1000]

# Autoencoder structure
input_img = Input(shape=(32, 32, 3))

# Encoder: compresses the image into a smaller, representative
encoding
x = Conv2D(32, (3, 3), activation='relu', padding='same')(input_img)
x = MaxPooling2D((2, 2), padding='same')(x)
x = Conv2D(16, (3, 3), activation='relu', padding='same')(x)
x = MaxPooling2D((2, 2), padding='same')(x)

# Decoder: reconstructs the image from the encoded representation
x = Conv2D(16, (3, 3), activation='relu', padding='same')(x)
```

```
x = UpSampling2D((2, 2))(x)
x = Conv2D(32, (3, 3), activation='relu', padding='same')(x)
x = UpSampling2D((2, 2))(x)
decoded = Conv2D(3, (3, 3), activation='sigmoid', padding='same')(x)

# Compile model
autoencoder = Model(input_img, decoded)
autoencoder.compile(optimizer=Adam(), loss='mae')

# Train model
autoencoder.fit(x_train_smaller, x_train_smaller,
                epochs=20,
                batch_size=128,
                shuffle=True,
                validation_data=(x_test_smaller, x_test_smaller))

# Predict reconstructed images
decoded_imgs = autoencoder.predict(x_test_smaller)

# 2. Visualization of original vs. the reconstructed images
n = 5
plt.figure(figsize=(10, 4))
for i in range(n):
    # Original images
    ax = plt.subplot(2, n, i + 1)
    plt.imshow(x_test_smaller[i])
    plt.title("Original")
    plt.axis("off")

    # Reconstructed images
    ax = plt.subplot(2, n, i + 1 + n)
    plt.imshow(decoded_imgs[i])
    plt.title("Reconstructed")
    plt.axis("off")

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

2. Required Displays:



This image shows the original images from the cifar test data and then the images below are the new images after the encoder encoding the images and then decoding them from the compacted (encoded) version using many convolutional layers to learn the features.

3. Discussion:

This assignment really helped me to learn more about autoencoders with the actual application of one within this assignment after learning the basics to their functionality in class.

The structure for my autoencoder is very simple, since the images are very small and simple and not many layers are needed to encode / decode these images. Mean absolute error was chosen as the assignment recommended but also because it's very effective in evaluating pixel-level accuracy for image reconstruction tasks.

Looking at the images plotted before and after you can see that the autoencoder reconstructed the general shapes and color patterns of the images very accurately, although the finer details still appear to be slightly blurry, this could be fixed by adding additional layers to the encoder and decoder (making the architecture just generally more complex). For this assignment I didn't see it necessary to add many layers though when just a couple for both the encoder and decoder seemed to work very well as is.