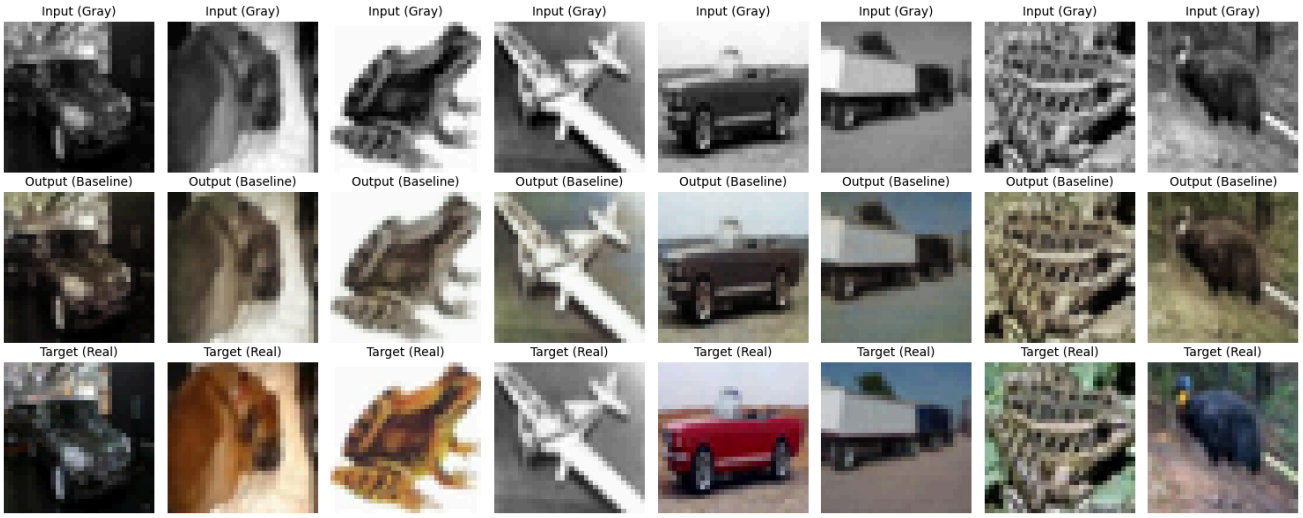
```
import torch
import torch.nn as nn
import torch.optim as optim
import torchvision
import torchvision.transforms as transforms
from torch.utils.data import DataLoader
import matplotlib.pyplot as plt
import numpy as np

# Configuration
BATCH_SIZE = 64
EPOCHS = 10  # Reduced for lab demonstration speed
LEARNING_RATE = 0.001
DEVICE = 'cuda' if torch.cuda.is_available() else 'cpu'

print(f"Running on: {DEVICE}")


transform = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5)) # Normalize to [-1, 1]
])

# Load CIFAR10
train_dataset = torchvision.datasets.CIFAR10(root='./data', train=True,
                                             download=True, transform=transform)
train_loader = DataLoader(train_dataset, batch_size=BATCH_SIZE, shuffle=True)

test_dataset = torchvision.datasets.CIFAR10(root='./data', train=False,
                                            download=True, transform=transform)

test_loader = DataLoader(test_dataset, batch_size=10, shuffle=True)

# Helper function to generate "Paired" data (Gray -> Color)
def generate_pairs(rgb_images):

    gray_images = (0.299 * rgb_images[:, 0, :, :] +
                   0.587 * rgb_images[:, 1, :, :] +
                   0.114 * rgb_images[:, 2, :, :])
    gray_images = gray_images.unsqueeze(1) # Add channel dim back: [B, 1, 32, 32]
    return gray_images, rgb_images


class EncoderDecoder(nn.Module):
    def __init__(self):
        super(EncoderDecoder, self).__init__()


        self.encoder = nn.Sequential(
            nn.Conv2d(1, 64, kernel_size=3, stride=1, padding=1),
            nn.ReLU(True),
            nn.Conv2d(64, 128, kernel_size=3, stride=2, padding=1), # 32x32 -> 16x16
            nn.ReLU(True),
            nn.Conv2d(128, 256, kernel_size=3, stride=2, padding=1), # 16x16 -> 8x8
            nn.ReLU(True)
        )


        self.decoder = nn.Sequential(
            nn.ConvTranspose2d(256, 128, kernel_size=3, stride=2, padding=1, output_padding=1),
            nn.ReLU(True),
            nn.ConvTranspose2d(128, 64, kernel_size=3, stride=2, padding=1, output_padding=1),
            nn.ReLU(True),
            nn.Conv2d(64, 3, kernel_size=3, stride=1, padding=1),
            nn.Tanh()
        )

    def forward(self, x):
        x = self.encoder(x)
        x = self.decoder(x)
        return x

model = EncoderDecoder().to(DEVICE)
criterion = nn.MSELoss() # Reconstruction Loss [cite: 93]
optimizer = optim.Adam(model.parameters(), lr=LEARNING_RATE)
```

```
print("\n--- Starting Training ---")
loss_history = []

for epoch in range(EPOCHS):
    running_loss = 0.0
    for i, (images, _) in enumerate(train_loader):
        images = images.to(DEVICE)


        gray, color = generate_pairs(images)

        optimizer.zero_grad()


        outputs = model(gray)


        loss = criterion(outputs, color)


        loss.backward()
        optimizer.step()

        running_loss += loss.item()

    avg_loss = running_loss / len(train_loader)
    loss_history.append(avg_loss)
    print(f"Epoch [{epoch+1}/{EPOCHS}], Loss (MSE): {avg_loss:.4f}")


def imshow(img, ax, title=None):

    img = img / 2 + 0.5
    npimg = img.cpu().numpy()
    ax.imshow(np.transpose(npimg, (1, 2, 0)), cmap='gray' if npimg.shape[0]==1 else None)
    if title: ax.set_title(title, fontsize=10)
    ax.axis('off')

dataiter = iter(test_loader)
images, _ = next(dataiter)
images = images.to(DEVICE)
gray_inputs, real_targets = generate_pairs(images)


model.eval()
with torch.no_grad():
    predicted_outputs = model(gray_inputs)


fig, axes = plt.subplots(3, 8, figsize=(15, 6))

for i in range(8):

    imshow(gray_inputs[i], axes[0, i], "Input (Gray)")


    imshow(predicted_outputs[i], axes[1, i], "Output (Baseline)")


    imshow(real_targets[i], axes[2, i], "Target (Real)")

plt.tight_layout()
plt.show()

print("\nObservation: Notice that the 'Output' images are blurry and desaturated compared to 'Target'.")
print("This is because MSE loss averages all possible colors, failing to capture high-frequency details.")
```

```
Running on: cuda
100%|██████████| 170M/170M [00:13<00:00, 12.9MB/s]

--- Starting Training ---
Epoch [1/10], Loss (MSE): 0.0336
Epoch [2/10], Loss (MSE): 0.0241
Epoch [3/10], Loss (MSE): 0.0228
Epoch [4/10], Loss (MSE): 0.0224
Epoch [5/10], Loss (MSE): 0.0219
Epoch [6/10], Loss (MSE): 0.0217
Epoch [7/10], Loss (MSE): 0.0215
Epoch [8/10], Loss (MSE): 0.0214
Epoch [9/10], Loss (MSE): 0.0212
Epoch [10/10], Loss (MSE): 0.0211
```




```
Observation: Notice that the 'Output' images are blurry and desaturated compared to 'Target'.
This is because MSE loss averages all possible colors, failing to capture high-frequency details.
```