

# Assignement 9

Shahriar Hasan

Id: 2010776115

July 11, 2025

## 1 Training an Autoencoder as a 2D Feature Generator and Visualizing CIFAR-10 Features

### 1.1 Dataset

- **Name:** CIFAR-10
- **Size:** 60,000 images ( $32 \times 32 \times 3$ )
  - 50,000 for training
  - 10,000 for testing
- **Classes (10 total):**  
Airplane, Automobile, Bird, Cat, Deer, Dog, Frog, Horse, Ship, Truck

### 1.2 Methodology

#### 1.2.1 Preprocessing

- Images are normalized to the range  $[0, 1]$ .
- All 60,000 images (train + test) are combined for unsupervised training.
- Corresponding class labels are concatenated to color points in the feature space.

#### 1.2.2 Model Architecture

##### Encoder:

- Conv2D(32)  $\rightarrow$  MaxPooling
- Conv2D(64)  $\rightarrow$  MaxPooling
- Flatten  $\rightarrow$  Dense(2)

##### Decoder:

- Dense  $\rightarrow$  Reshape
- Conv2DTranspose layers to reconstruct the input

### 1.2.3 Training

- **Loss Function:** Mean Squared Error (MSE)
- **Optimizer:** Adam
- **Epochs:** 10
- **Batch Size:** 256
- **Data:** All 60,000 images (unsupervised)

## 1.3 Visualization

A scatter plot is created from the 2D features extracted by the encoder. Points are color-coded according to their class labels using `matplotlib` and the `tab10` colormap.

## 1.4 Result

- The 2D encoded features show **clear clustering** based on semantic similarity.
- Classes such as *Airplane*, *Truck*, and *Ship* show visible separation, while visually similar classes like *Cat* and *Dog* may overlap.
- The autoencoder successfully learns compressed representations that **preserve class-level structure** in an unsupervised manner.

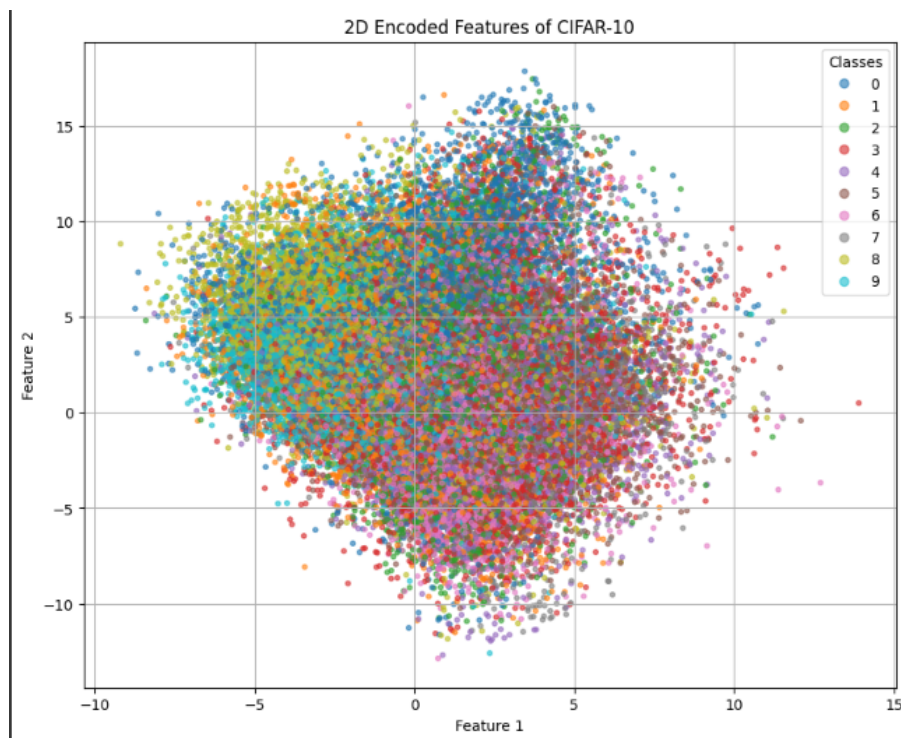


Figure 1: 2D Encoded Feature Visualization of CIFAR-10

## 2 Comparing autoencoder generated features with features extracted by a pre-trained CNN and reduced by dimension reduction techniques like PCA, t-SNE.

### 2.1 Dataset

- **Dataset:** CIFAR-10
- **Number of Images:** 5000 (random subset)
- **Image Size:** 32×32 RGB
- **Classes:** 10

### 2.2 Implementation Steps

#### 1. Autoencoder Training:

- Input shape: (32, 32, 3)
- Encoder compresses features to 2D (bottleneck layer)
- Decoder reconstructs images from the 2D features
- Trained using Adam optimizer and Mean Squared Error (MSE) loss for 10 epochs

#### 2. Feature Extraction using Autoencoder:

- Extracted 2D encoded features
- Directly visualized using scatter plot with class labels

#### 3. Feature Extraction using Pretrained VGG16:

- Images resized to 224×224×3
- Preprocessed using VGG16-specific normalization
- Features extracted from the global average pooling layer (last layer before classification)

#### 4. Dimensionality Reduction:

- PCA applied to reduce VGG16 features from 512-D to 2-D
- t-SNE used for nonlinear 2D projection

#### 5. Visualization:

- Used scatter plots for all 2D projections
- Points color-coded by CIFAR-10 class

## 2.3 Results (Visualizations)

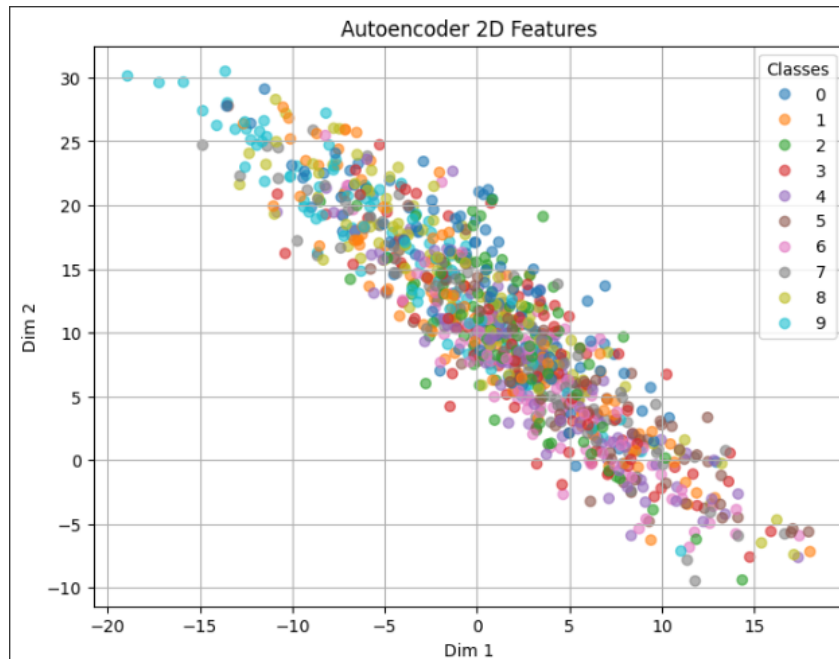


Figure 2: Autoencoder 2D Features

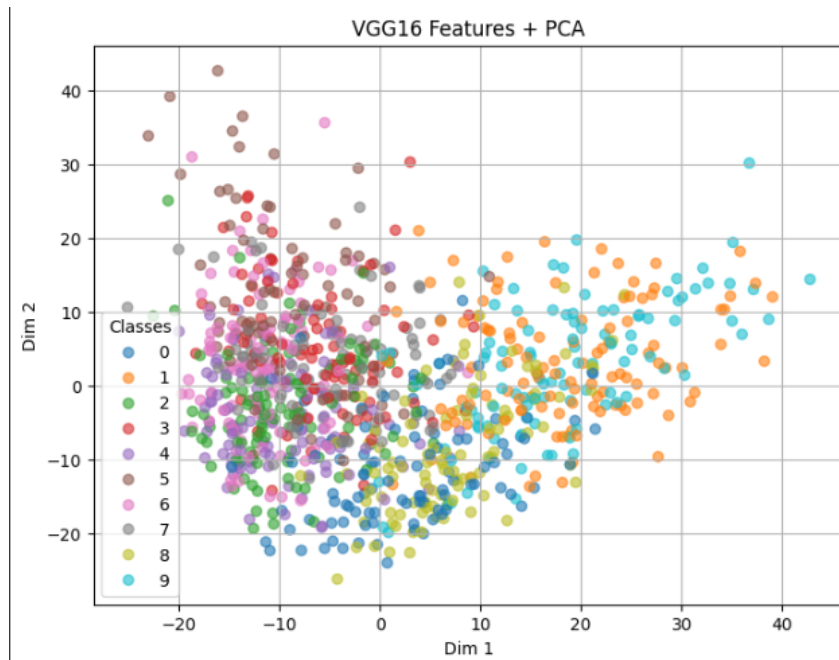


Figure 3: VGG16 Features + PCA

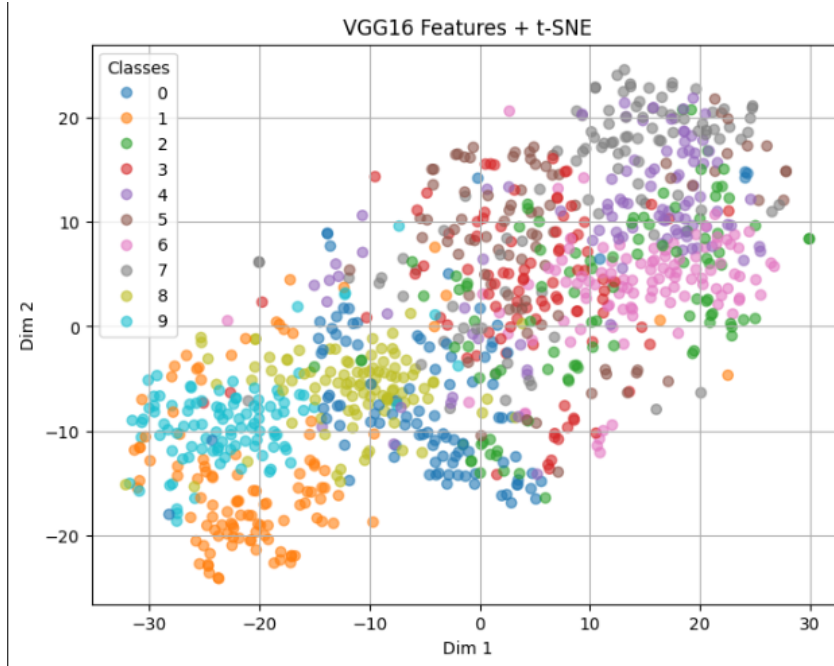


Figure 4: VGG16 Features + t-SNE

## 3 Training a Denoising Autoencoder on the CIFAR-10 Dataset

### 3.1 Dataset

- **Dataset Used:** CIFAR-10
- **Image Size:**  $32 \times 32 \times 3$
- **Training Samples:** 50,000
- **Testing Samples:** 10,000
- **Preprocessing:**
  - Pixel values normalized to the  $[0, 1]$  range
  - Gaussian noise (mean = 0, std = 1, factor = 0.2) added to training and test data
  - Noisy values clipped to  $[0, 1]$

### 3.2 Model Architecture

#### 3.2.1 Encoder

- Conv2D (32 filters,  $3 \times 3$ , ReLU, padding='same')
- MaxPooling2D ( $2 \times 2$ , padding='same')
- Conv2D (64 filters,  $3 \times 3$ , ReLU, padding='same')
- MaxPooling2D ( $2 \times 2$ , padding='same')

### 3.2.2 Decoder

- Conv2D (64 filters,  $3 \times 3$ , ReLU, padding='same')
- UpSampling2D ( $2 \times 2$ )
- Conv2D (32 filters,  $3 \times 3$ , ReLU, padding='same')
- UpSampling2D ( $2 \times 2$ )
- Conv2D (3 filters,  $3 \times 3$ , Sigmoid, padding='same') — output layer

## 3.3 Training Details

- **Loss Function:** Binary Crossentropy
- **Optimizer:** Adam
- **Epochs:** 10
- **Batch Size:** 128
- **Validation:** Performed on noisy test images with their original versions as targets

## 3.4 Results(Visualization)

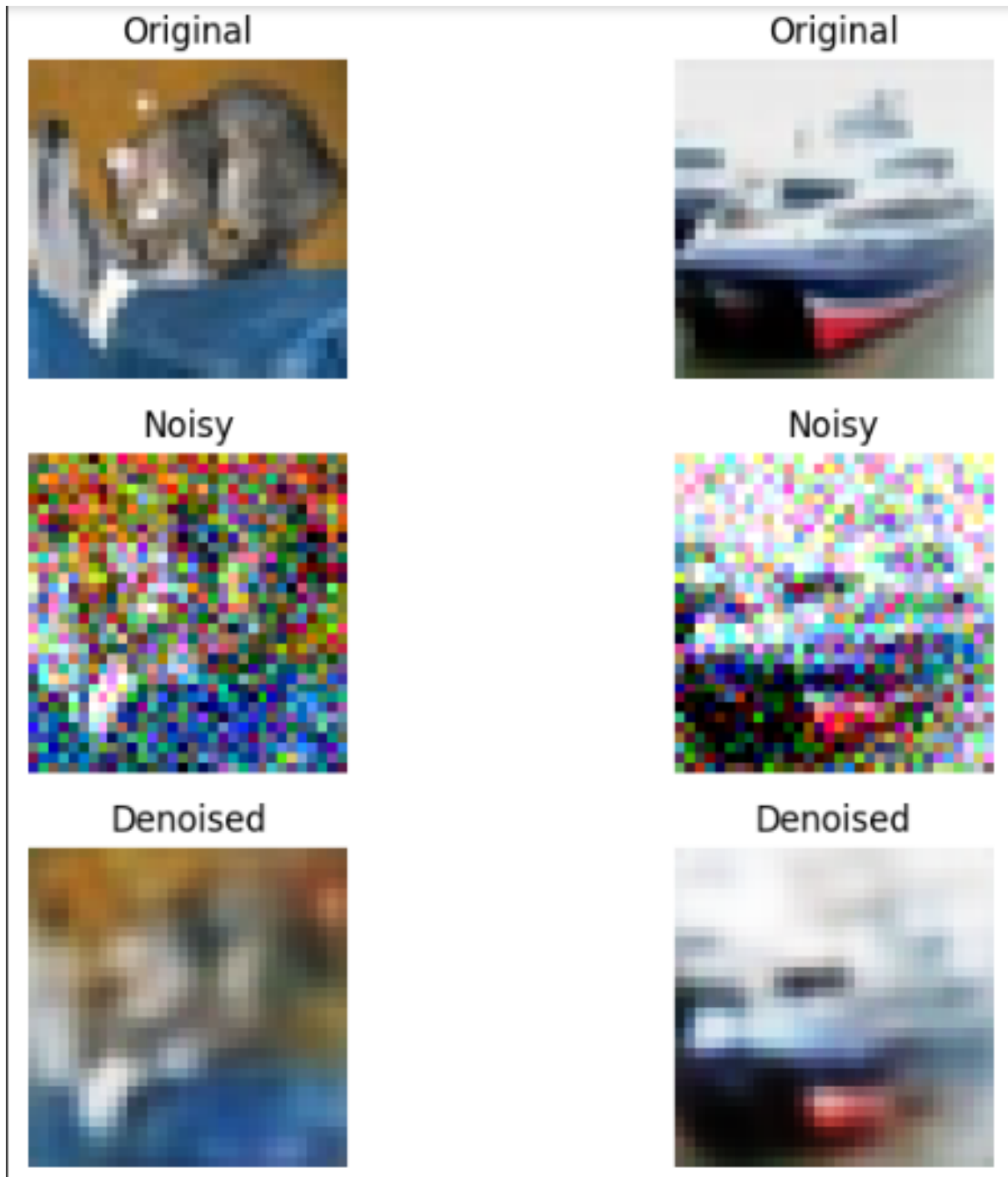


Figure 5: Example of Original, Noisy, and Denoised images from the CIFAR-10 dataset.

## 4 Training a CNN based CIFAR-10 classifier without any single-image data augmentation techniques

### 4.1 Dataset

- **CIFAR-10:** Contains 60,000  $32 \times 32$  color images in 10 classes, with 6,000 images per class.

- **Split:** 50,000 training images and 10,000 test images.

## 4.2 Preprocessing

- Image pixel values normalized to the range  $[0, 1]$ .
- Labels were one-hot encoded for multi-class classification.

## 4.3 Model Architecture

- **Input:**  $32 \times 32 \times 3$
- **Convolutional Block 1:**
  - Conv2D(32,  $3 \times 3$ , ReLU, same padding)
  - Conv2D(32,  $3 \times 3$ , ReLU)
  - MaxPooling2D( $2 \times 2$ )
  - Dropout(0.25)
- **Convolutional Block 2:**
  - Conv2D(64,  $3 \times 3$ , ReLU, same padding)
  - Conv2D(64,  $3 \times 3$ , ReLU)
  - MaxPooling2D( $2 \times 2$ )
  - Dropout(0.25)
- **Fully Connected:**
  - Flatten
  - Dense(512, ReLU)
  - Dropout(0.5)
  - Dense(10, Softmax)

## 4.4 Training Configuration

- **Loss Function:** Categorical Crossentropy
- **Optimizer:** Adam
- **Metrics:** Accuracy
- **Epochs:** 20
- **Batch Size:** 64
- **Validation:** Test set used for validation

## 4.5 Results

The model was trained and validated over 20 epochs. Accuracy and loss curves were plotted to monitor performance.



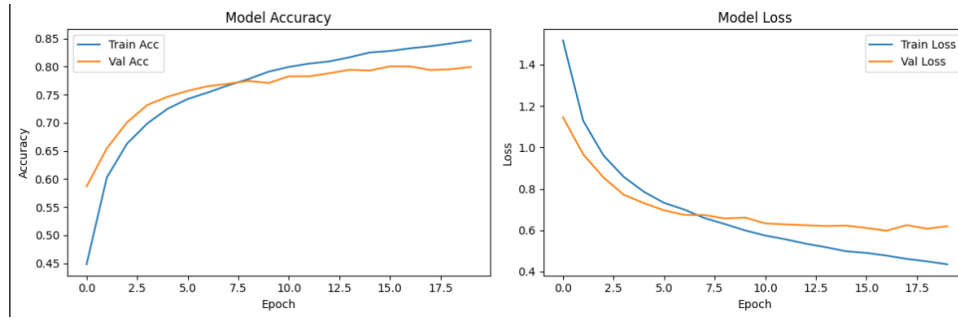


Figure 6: Training and validation accuracy (left) and loss (right) over epochs.

## 5 training a CNN based CIFAR-10 classifier with a single/multiple single-image data augmentation techniques

### 5.1 Dataset

- **Name:** CIFAR-10
- **Images:** 60,000 (32x32 RGB)
  - 50,000 training images
  - 10,000 test images
- **Classes:** 10 (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck)

### 5.2 Data Preprocessing

- Normalized pixel values to the range  $[0, 1]$ .
- One-hot encoding applied to the labels.

### 5.3 Data Augmentation Techniques Applied

- Horizontal Flip
- Random Rotation ( $\pm 10\%$ )
- Random Zoom ( $\pm 10\%$ )
- Random Translation ( $\pm 10\%$ )
- Random Contrast ( $\pm 10\%$ )

### 5.4 Model Architecture

A CNN consisting of:

- Two convolutional blocks:

- Conv2D → Conv2D → MaxPooling2D → Dropout
- Fully connected layers:
  - Flatten → Dense(512) → Dropout → Dense(10)

## 5.5 Training Details

- **Epochs:** 20
- **Batch size:** 64
- **Loss function:** Categorical Crossentropy
- **Optimizer:** Adam
- **Metrics:** Accuracy

## 5.6 Results

### 5.6.1 Model Accuracy and Loss Curves

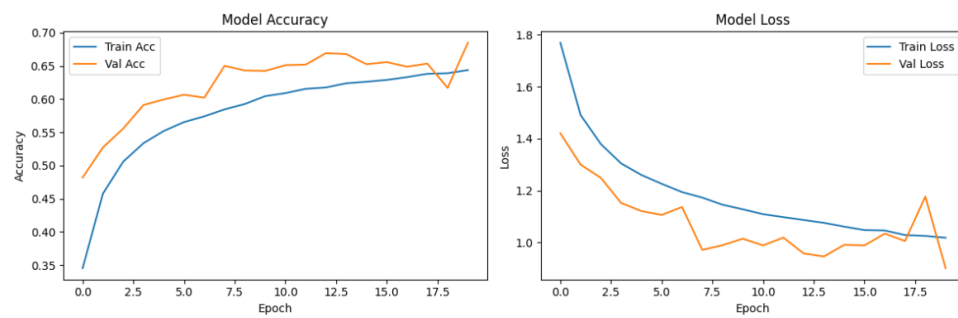


Figure 7: Training and Validation Accuracy and Loss

### 5.6.2 Visualization of Data Augmentation

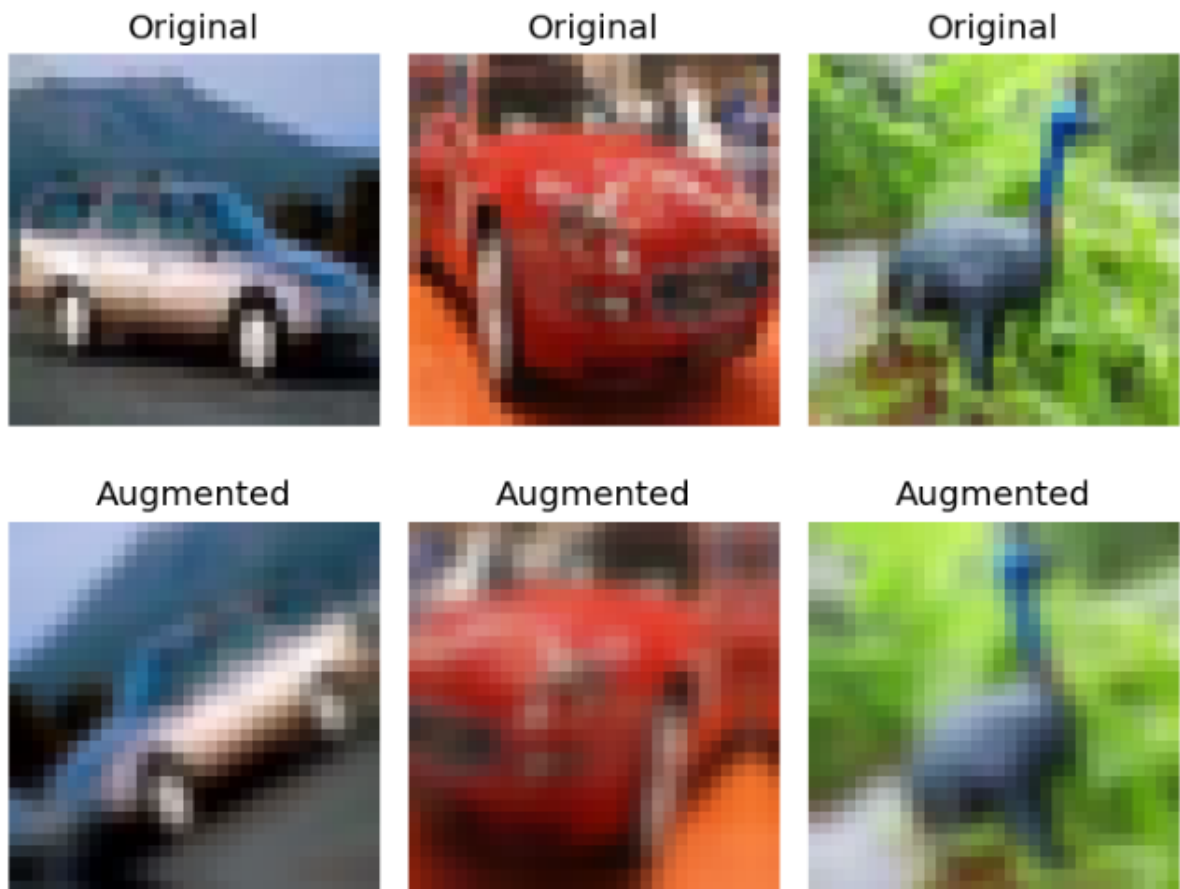


Figure 8: Examples of Augmented Images

You can access the code and visuals from the following link:  
**Github Link**