Name: Santiago, John Loyd C. Santiago
Course and Section: CPE 019 - CPE32S3
Date of Submission: April 27, 2024
Instructor: Engr. Roman Richard

link of the google colab: https://colab.research.google.com/drive/14tyP25rDIu4WHRroA2LUfTZn8ZqW1LqI?usp=sharing

link of the dataset:https://www.kaggle.com/datasets/gatewayadam/cars-and-tanks-image-classification

link of the google drive: https://drive.google.com/drive/folders/1_hD8_v0sCfAw0uJbfzah404TEkqy-9al?usp=sharing

1). Choose any dataset applicable to an image classification problem

The dataset that I pick Is about Cars and tanks it contains almost 1200 images divided into two classes, Cars and tanks class the link is above

2). Explain your datasets and the problem being addressed.

The problem I'm being address in this activity is the difference of car and tank

3). Show evidence that you can do the following:

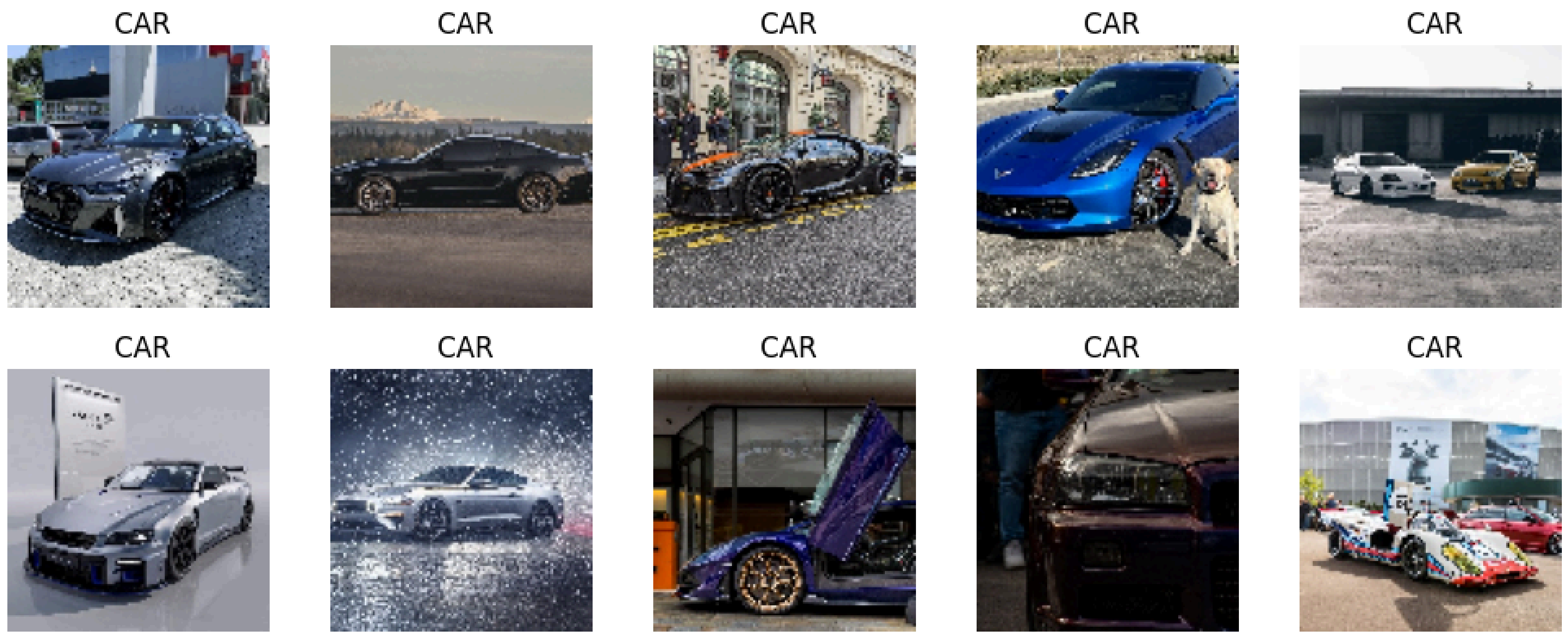
```
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.preprocessing.image import load_img

from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
def create_binary_model_data(train_dir, test_dir, batch_size):
    train_datagen = ImageDataGenerator(rescale=1./255)
    test_datagen = ImageDataGenerator(rescale=1./255)

    train_generator = train_datagen.flow_from_directory(
        directory=train_dir,
        target_size=(100, 100),
        batch_size=batch_size,
        class_mode='binary'
    )

    test_generator = test_datagen.flow_from_directory(
        directory=test_dir,
        target_size=(100, 100),
        batch_size=batch_size,
        class_mode='binary'
    )

    return train_generator, test_generator
```

```
def show_images(generator, num_images=10):
    plt.figure(figsize=(10, 10))
    rows = 5
    columns = 5
    for i in range(num_images):
        if i >= num_images:
            break
        image, label = next(generator)
        plt.subplot(rows, columns, i + 1)
        plt.imshow(image[0])
        plt.title(" {}".format("CAR" if label[0] == 0 else "TANK"))
        plt.axis('off')
    plt.tight_layout()
    plt.show()

train_dir = '/content/drive/MyDrive/Assignment 9.1 : Convolutional Neural Network/train'
test_dir = '/content/drive/MyDrive/Assignment 9.1 : Convolutional Neural Network/test'

batch_size = 1000
train_gen, test_gen = create_binary_model_data(train_dir, test_dir, batch_size)

show_images(train_gen, num_images=10)
```

```
Found 310 images belonging to 2 classes.
Found 200 images belonging to 2 classes.
```



- Using your dataset, create a baseline model of the CNN

```
model = define_model()
history = model.fit(train_gen, epochs=50, validation_data=test_gen)
```

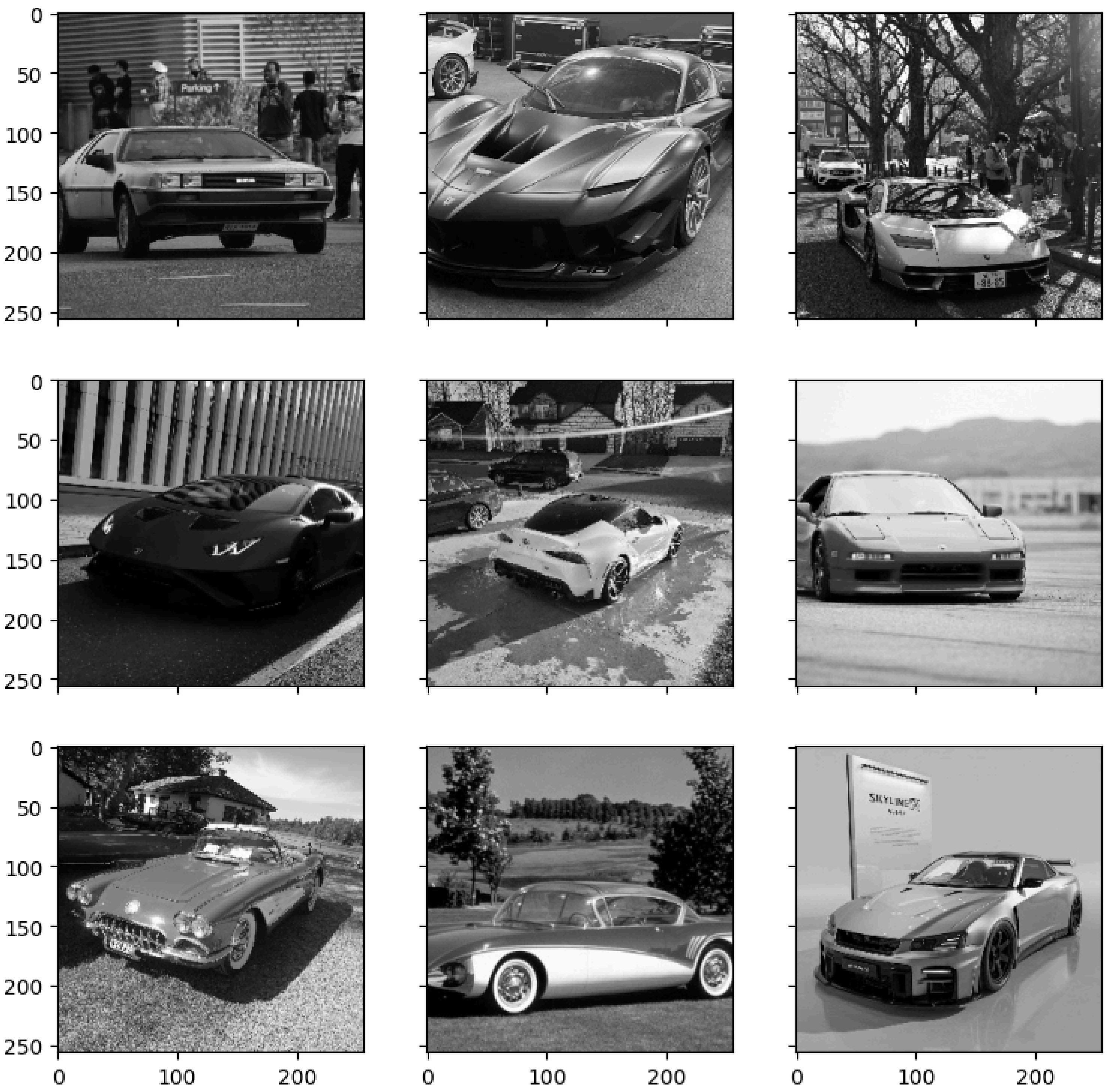
```
Epoch 1/50
1/1 [==============================] - 8s 8s/step - loss: 0.8770 - accuracy: 0.4422 - val_loss: 5.3410 - val_accuracy: 0.5567
Epoch 2/50
1/1 [==============================] - 8s 8s/step - loss: 5.6729 - accuracy: 0.5578 - val_loss: 1.2191 - val_accuracy: 0.5567
Epoch 3/50
1/1 [==============================] - 6s 6s/step - loss: 1.2886 - accuracy: 0.5578 - val_loss: 0.6964 - val_accuracy: 0.4227
Epoch 4/50
1/1 [==============================] - 8s 8s/step - loss: 0.6945 - accuracy: 0.4490 - val_loss: 0.6923 - val_accuracy: 0.5361
Epoch 5/50
1/1 [==============================] - 8s 8s/step - loss: 0.6912 - accuracy: 0.4796 - val_loss: 0.6913 - val_accuracy: 0.4948
Epoch 6/50
1/1 [==============================] - 6s 6s/step - loss: 0.6898 - accuracy: 0.4456 - val_loss: 0.6906 - val_accuracy: 0.4536
Epoch 7/50
1/1 [==============================] - 7s 7s/step - loss: 0.6881 - accuracy: 0.4490 - val_loss: 0.6909 - val_accuracy: 0.4742
Epoch 8/50
1/1 [==============================] - 6s 6s/step - loss: 0.6860 - accuracy: 0.4490 - val_loss: 0.6909 - val_accuracy: 0.4433
Epoch 9/50
1/1 [==============================] - 8s 8s/step - loss: 0.6832 - accuracy: 0.4864 - val_loss: 0.6899 - val_accuracy: 0.4742
Epoch 10/50
1/1 [==============================] - 6s 6s/step - loss: 0.6794 - accuracy: 0.5204 - val_loss: 0.6866 - val_accuracy: 0.5567
Epoch 11/50
1/1 [==============================] - 6s 6s/step - loss: 0.6744 - accuracy: 0.5918 - val_loss: 0.6828 - val_accuracy: 0.5979
Epoch 12/50
1/1 [==============================] - 6s 6s/step - loss: 0.6688 - accuracy: 0.6565 - val_loss: 0.6783 - val_accuracy: 0.6082
Epoch 13/50
1/1 [==============================] - 7s 7s/step - loss: 0.6628 - accuracy: 0.6769 - val_loss: 0.6731 - val_accuracy: 0.6289
Epoch 14/50
1/1 [==============================] - 8s 8s/step - loss: 0.6563 - accuracy: 0.6803 - val_loss: 0.6675 - val_accuracy: 0.6289
Epoch 15/50
1/1 [==============================] - 6s 6s/step - loss: 0.6489 - accuracy: 0.7109 - val_loss: 0.6614 - val_accuracy: 0.6186
Epoch 16/50
1/1 [==============================] - 6s 6s/step - loss: 0.6404 - accuracy: 0.7041 - val_loss: 0.6554 - val_accuracy: 0.6289
Epoch 17/50
1/1 [==============================] - 6s 6s/step - loss: 0.6309 - accuracy: 0.6973 - val_loss: 0.6493 - val_accuracy: 0.6392
Epoch 18/50
1/1 [==============================] - 6s 6s/step - loss: 0.6203 - accuracy: 0.7007 - val_loss: 0.6432 - val_accuracy: 0.6495
Epoch 19/50
1/1 [==============================] - 6s 6s/step - loss: 0.6090 - accuracy: 0.7041 - val_loss: 0.6381 - val_accuracy: 0.6392
Epoch 20/50
1/1 [==============================] - 7s 7s/step - loss: 0.5972 - accuracy: 0.7041 - val_loss: 0.6339 - val_accuracy: 0.6392
Epoch 21/50
1/1 [==============================] - 6s 6s/step - loss: 0.5855 - accuracy: 0.7143 - val_loss: 0.6305 - val_accuracy: 0.6392
Epoch 22/50
1/1 [==============================] - 8s 8s/step - loss: 0.5748 - accuracy: 0.7075 - val_loss: 0.6287 - val_accuracy: 0.6495
Epoch 23/50
1/1 [==============================] - 6s 6s/step - loss: 0.5657 - accuracy: 0.7075 - val_loss: 0.6290 - val_accuracy: 0.6495
Epoch 24/50
1/1 [==============================] - 8s 8s/step - loss: 0.5583 - accuracy: 0.7109 - val_loss: 0.6317 - val_accuracy: 0.6598
Epoch 25/50
1/1 [==============================] - 6s 6s/step - loss: 0.5521 - accuracy: 0.7143 - val_loss: 0.6351 - val_accuracy: 0.6598
Epoch 26/50
1/1 [==============================] - 8s 8s/step - loss: 0.5468 - accuracy: 0.7211 - val_loss: 0.6377 - val_accuracy: 0.6598
Epoch 27/50
1/1 [==============================] - 6s 6s/step - loss: 0.5419 - accuracy: 0.7245 - val_loss: 0.6363 - val_accuracy: 0.6598
Epoch 28/50
1/1 [==============================] - 6s 6s/step - loss: 0.5363 - accuracy: 0.7245 - val_loss: 0.6307 - val_accuracy: 0.6495
Epoch 29/50
1/1 [==============================] - 8s 8s/step - loss: 0.5293 - accuracy: 0.7245 - val_loss: 0.6226 - val_accuracy: 0.6392
```

```
loss, accuracy = model.evaluate(test_gen)
print("Test Loss:", loss)
print("Test Accuracy:", accuracy)
```

```
1/1 [==============================] - 1s 836ms/step - loss: 0.5561 - accuracy: 0.7113
Test Loss: 0.5560869574546814
Test Accuracy: 0.7113401889801025
```

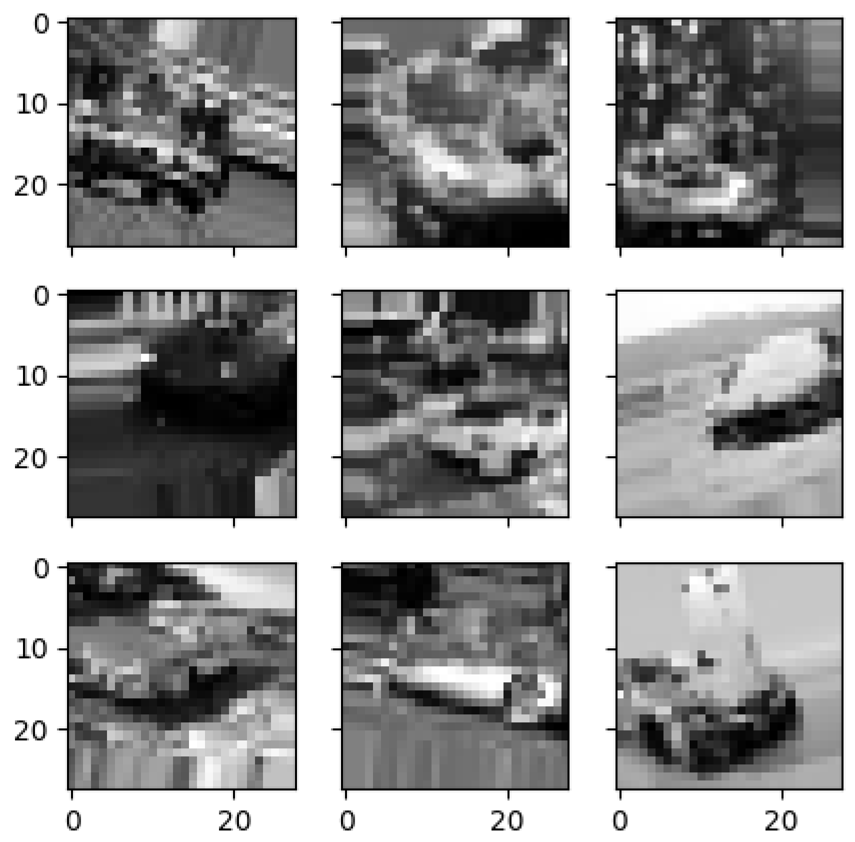
- Perform image augmentation

```
from tensorflow.keras.preprocessing.image import load_img

def load_data(directory):
    images = []
    labels = []
    for label in os.listdir(directory):
        label_dir = os.path.join(directory, label)
        for img_file in os.listdir(label_dir):
            img_path = os.path.join(label_dir, img_file)
            img = load_img(img_path, color_mode='grayscale')
            images.append(img)
            labels.append(label)
    return images, labels

X_train, y_train = load_data(train_dir)

fig, ax = plt.subplots(3, 3, sharex=True, sharey=True, figsize=(9,9))
for i in range(3):
    for j in range(3):
        ax[i][j].imshow(X_train[i*3+j], cmap=plt.get_cmap("gray"))

plt.show()
```



- Perform feature standardization

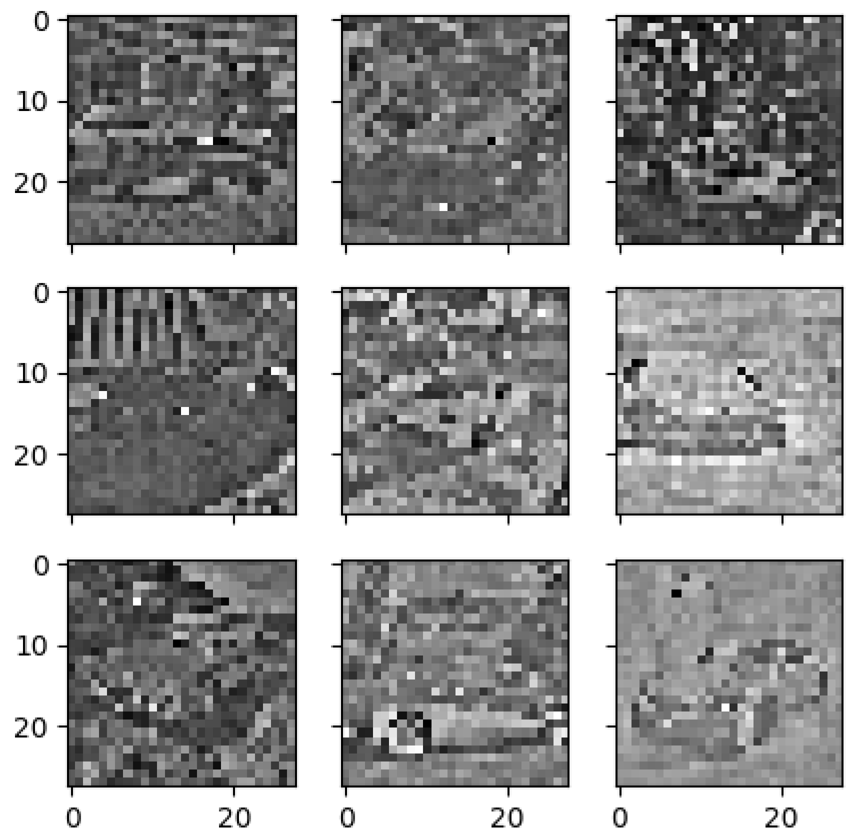
```
def load_data(directory):
    images = []
    labels = []
    for label in os.listdir(directory):
        label_dir = os.path.join(directory, label)
        for img_file in os.listdir(label_dir):
            img_path = os.path.join(label_dir, img_file)
            img = load_img(img_path, color_mode='grayscale', target_size=(28, 28))
            images.append(np.array(img))
            labels.append(label)
    return np.array(images), np.array(labels)

train_dir = '/content/drive/MyDrive/Assignment 9.1 : Convolutional Neural Network/train'
X_train, y_train = load_data(train_dir)

X_train = X_train.astype('float32') / 255.0
X_train = np.expand_dims(X_train, axis=-1)

datagen = ImageDataGenerator(
    featurewise_center=True,
    featurewise_std_normalization=True,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)

datagen.fit(X_train)

for X_batch, y_batch in datagen.flow(X_train, y_train, batch_size=9, shuffle=False):
    fig, ax = plt.subplots(3, 3, sharex=True, sharey=True, figsize=(5, 5))
    for i in range(3):
        for j in range(3):
            ax[i][j].imshow(X_batch[i*3+j].reshape(28, 28), cmap=plt.get_cmap("gray"))
    plt.show()
    break
```



- Perform ZCA whitening of your images

```
datagen = ImageDataGenerator(featurewise_center=True,
                             featurewise_std_normalization=True,
                             zca_whitening=True)

datagen.fit(X_train)

for X_batch, y_batch in datagen.flow(X_train, y_train, batch_size=9, shuffle=False):
    print(X_batch.min(), X_batch.mean(), X_batch.max())
    fig, ax = plt.subplots(3, 3, sharex=True, sharey=True, figsize=(5, 5))
    for i in range(3):
        for j in range(3):
            ax[i][j].imshow(X_batch[i*3+j].reshape(28, 28), cmap=plt.get_cmap("gray"))

    plt.show()
    break
```

```
-3.3785515 0.0026001758 2.9525511
```



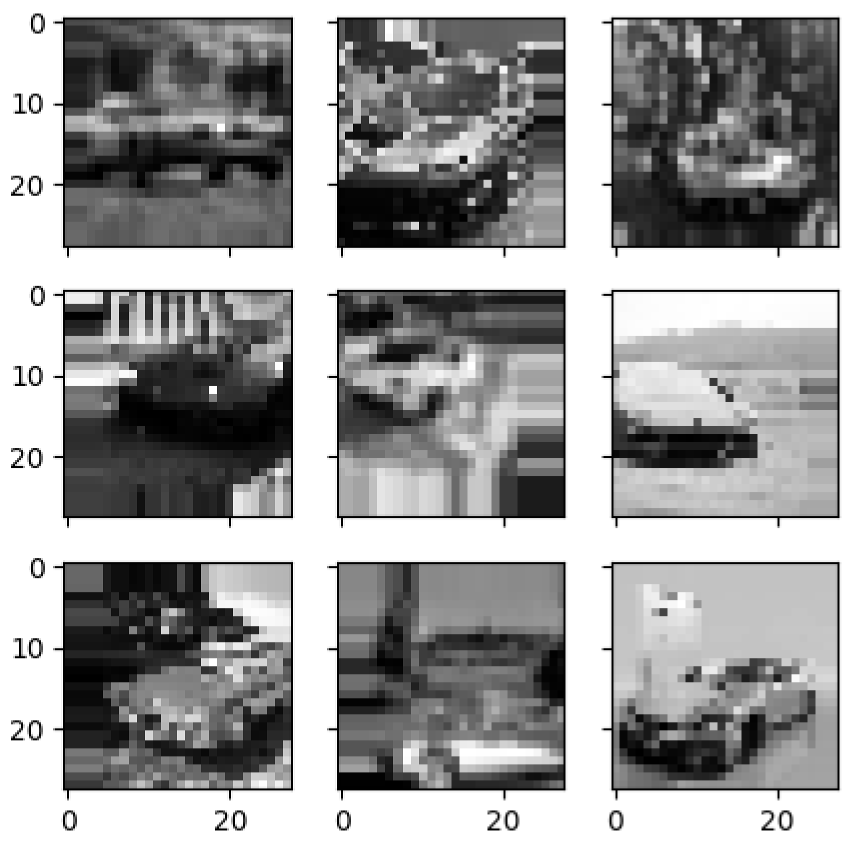
- Data Augmentation (random rotations, shifts, flips)

- Random Rotations

```
datagen = ImageDataGenerator(rotation_range=90)

for X_batch, y_batch in datagen.flow(X_train, y_train, batch_size=9, shuffle=False):
    fig, ax = plt.subplots(3, 3, sharex=True, sharey=True, figsize=(5, 5))
    for i in range(3):
        for j in range(3):
            ax[i][j].imshow(X_batch[i*3+j].reshape(28, 28), cmap=plt.get_cmap("gray"))

    plt.show()
    break
```

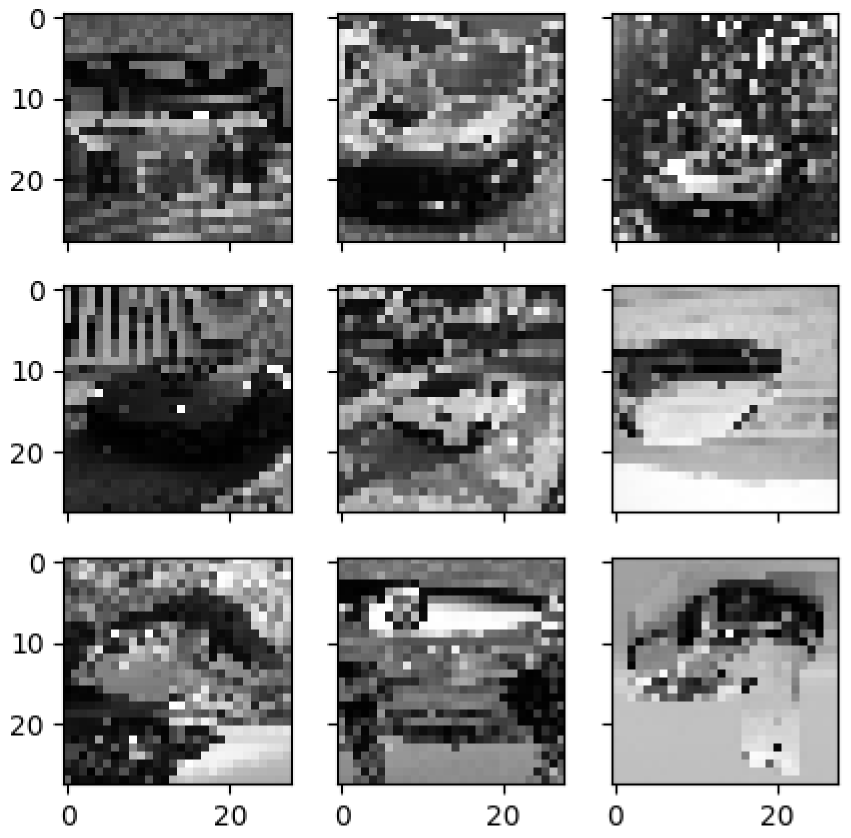
- Random Shifts

```
shift = 0.2
datagen = ImageDataGenerator(width_shift_range=shift, height_shift_range=shift)

for X_batch, y_batch in datagen.flow(X_train, y_train, batch_size=9, shuffle=False):
    fig, ax = plt.subplots(3, 3, sharex=True, sharey=True, figsize=(5,5))
    for i in range(3):
        for j in range(3):
            ax[i][j].imshow(X_batch[i*3+j].reshape(28,28), cmap=plt.get_cmap("gray"))
    # show the plot
    plt.show()
    break
```



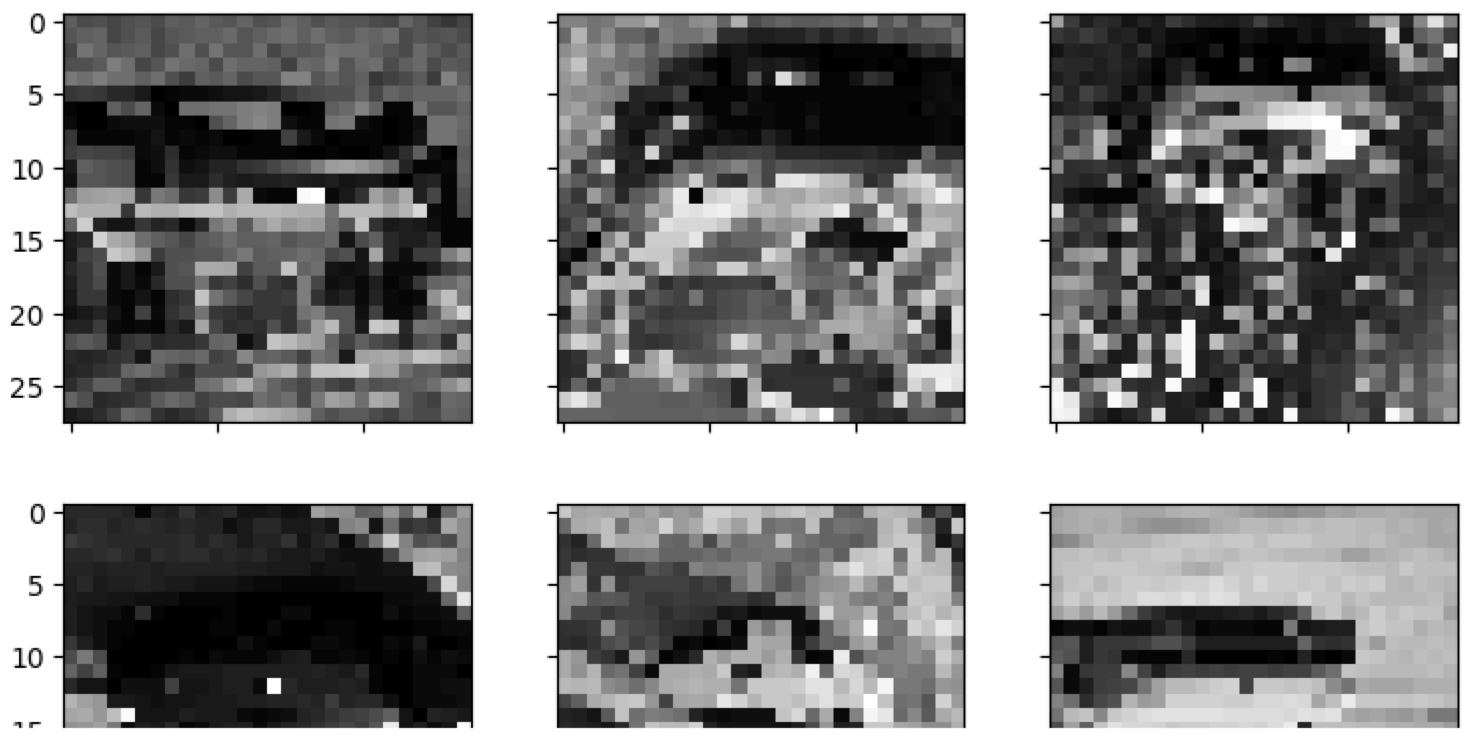
- Random Flips

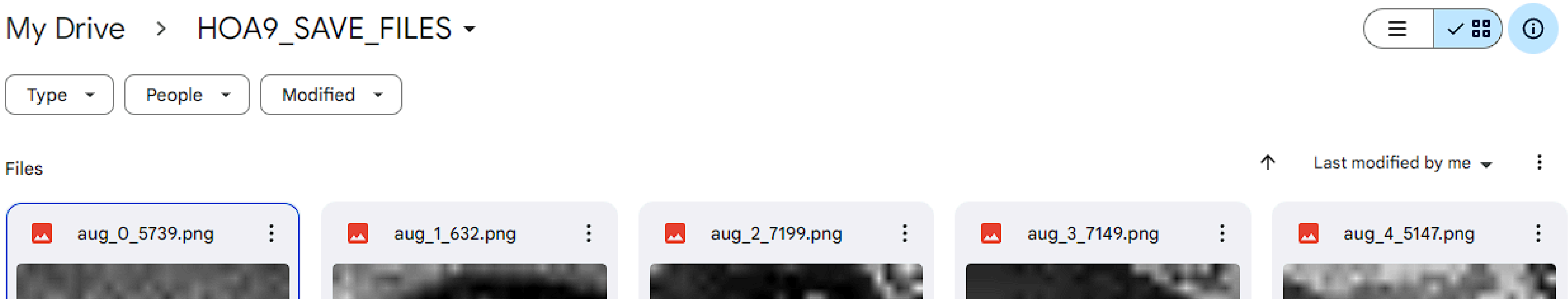
```
datagen = ImageDataGenerator(horizontal_flip=True, vertical_flip=True)

for X_batch, y_batch in datagen.flow(X_train, y_train, batch_size=9, shuffle=False):
    fig, ax = plt.subplots(3, 3, sharex=True, sharey=True, figsize=(5,5))
    for i in range(3):
        for j in range(3):
            ax[i][j].imshow(X_batch[i*3+j].reshape(28,28), cmap=plt.get_cmap("gray"))
    # show the plot
    plt.show()
    break
```



- Save augmented image data to disk

```
from tensorflow.keras.preprocessing.image import img_to_array

def load_data(directory):
    images = []
    labels = []
    for label in os.listdir(directory):
        label_dir = os.path.join(directory, label)
        for img_file in os.listdir(label_dir):
            img_path = os.path.join(label_dir, img_file)
            img = load_img(img_path, color_mode='grayscale', target_size=(28, 28))
            images.append(img)
            labels.append(label)
    return images, labels

images, labels = load_data(train_dir)
X_train = np.array([img_to_array(img) for img in images])
X_train = X_train.astype('float32') / 255.0

datagen = ImageDataGenerator(horizontal_flip=True, vertical_flip=True)

for X_batch, y_batch in datagen.flow(X_train, labels, batch_size=9, shuffle=False, save_to_dir='/content/drive/MyDrive/HOA9_SAVE_FILES', save_prefix='aug', save_format='png'):

    fig, ax = plt.subplots(3, 3, sharex=True, sharey=True, figsize=(9, 9))
    for i in range(3):
        for j in range(3):
            ax[i][j].imshow(X_batch[i*3+j].reshape(28, 28), cmap=plt.get_cmap("gray"))

    plt.show()
    break
```





My Drive > HOA9_SAVE_FILES

Files

Last modified by me

aug_0_5739.png

aug_1_632.png

aug_2_7199.png

aug_3_7149.png

aug_4_5147.png