Technological Institute of the Philippines	**Quezon City - Computer Engineering**
Course Code:	CPE 019
Code Title:	Emerging Technologies in CpE 2
2nd Semester	AY 2024 - 2025

ASSIGNMENT 9.1	**Convolutional Neural Network**
Name	Calvadores, Kelly Joseph
Section	CPE32S3
Date Performed:	Arpil 24, 2024
Date Submitted:	April 27, 2024
Instructor:	Engr. Roman M. Richard

Choose any dataset applicable to an image classification problem

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

```
Mounted at /content/drive
```

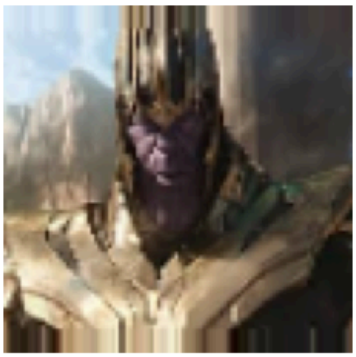
Load Dataset

Resource: https://www.kaggle.com/datasets/rogeriovaz/villains-image-classification?resource=download

```
import tensorflow as tf
from keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt

IMG = ImageDataGenerator(rescale = 1./255,
                         zoom_range = 0.2,
                         horizontal_flip = True,
                         validation_split = 0.2)

TDS = '/content/drive/MyDrive/CPE 019/Assignment 9.1/Villains'

Train_Dataset = IMG.flow_from_directory(batch_size = 1000,
                                        directory = TDS,
                                        shuffle = True,
                                        target_size = (100, 100),
                                        class_mode = 'categorical')

Validate_Dataset = IMG.flow_from_directory(batch_size = 1000,
                                           directory = TDS,
                                           shuffle = True,
                                           target_size = (100, 100),
                                           class_mode = "categorical")
Train_Dataset
```

```
Found 100 images belonging to 5 classes.
Found 100 images belonging to 5 classes.
<keras.src.preprocessing.image.DirectoryIterator at 0x79d2e74f5ed0>
```

```
TrainImages, TrainLabels = Train_Dataset.next()
plt.figure(figsize = (10, 10))
for i in range(9):
  ax = plt.subplot(3, 3, i + 1)
  plt.imshow(TrainImages[i])
  plt.axis("off")
plt.show()

ValidImages, ValidLabels = Validate_Dataset.next()
plt.figure(figsize = (10, 10))
for i in range(9):
  ax = plt.subplot(3, 3, i + 1)
  plt.imshow(ValidImages[i])
  plt.axis("off")
plt.show()
```

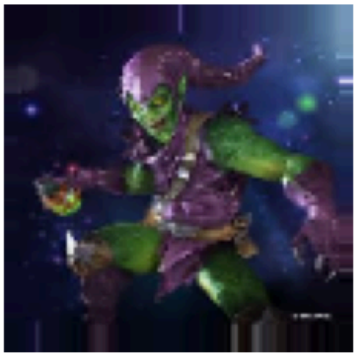


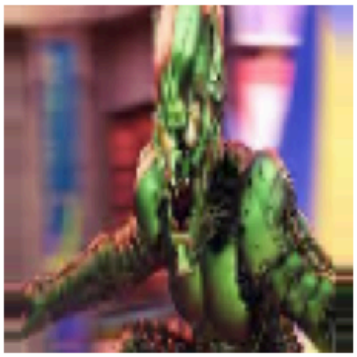










Explain your datasets and the problem being addressed.

- The Dataset contains of images of villains from different comics, show or movies. The problem being addressed for Image classification, the aim is to develop a model that capable of identifying and classifying images of the different villains, the task is to apply this kind of model on real-time such as security cameras and many more.

Show evidence that you can do the following:

Using your dataset, create a baseline model of the CNN

Create a baseline

Import Libraries

```
from fastai import *
from fastai.vision import *
from fastai.metrics import error_rate
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import cv2

from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D
from tensorflow.keras.layers import MaxPooling2D
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Flatten
from tensorflow.keras.optimizers import SGD


#Create Model
Model = Sequential()
Model.add(Conv2D(32, (3, 3), activation = "relu", kernel_initializer = 'he_uniform', input_shape = (100, 100, 3)))
Model.add(MaxPooling2D((2, 2)))
Model.add(Flatten())

Model.add(Dense(128, activation = "tanh", kernel_initializer = "he_uniform"))
Model.add(Dense(64, activation = "relu"))

Model.add(Dense(5, activation = "softmax"))

#Compile the model
Opt = SGD(learning_rate = 0.0001, momentum = 0.9)
Model.compile(optimizer = Opt, loss = 'categorical_crossentropy', metrics = ['accuracy'])
```

```
#Train the model
Model.fit_generator(Train_Dataset, epochs = 300, validation_data = Validate_Dataset)
```

```
<ipython-input-5-01baa14dd2e9>:17: UserWarning: `Model.fit_generator` is deprecated and will be removed in a f
  Model.fit_generator(Train_Dataset, epochs = 300, validation_data = Validate_Dataset)
Epoch 1/300
1/1 [==============================] - 4s 4s/step - loss: 1.8809 - accuracy: 0.2100 - val_loss: 1.7873 - val_a
Epoch 2/300
1/1 [==============================] - 3s 3s/step - loss: 1.7558 - accuracy: 0.2300 - val_loss: 1.6761 - val_a
Epoch 3/300
1/1 [==============================] - 3s 3s/step - loss: 1.7042 - accuracy: 0.2200 - val_loss: 1.6081 - val_a
Epoch 4/300
1/1 [==============================] - 3s 3s/step - loss: 1.6202 - accuracy: 0.2400 - val_loss: 1.6551 - val_a
Epoch 5/300
1/1 [==============================] - 3s 3s/step - loss: 1.6324 - accuracy: 0.2400 - val_loss: 1.6613 - val_a
Epoch 6/300
1/1 [==============================] - 3s 3s/step - loss: 1.6607 - accuracy: 0.2000 - val_loss: 1.6948 - val_a
Epoch 7/300
1/1 [==============================] - 6s 6s/step - loss: 1.6871 - accuracy: 0.2200 - val_loss: 1.6314 - val_a
Epoch 8/300
1/1 [==============================] - 3s 3s/step - loss: 1.6423 - accuracy: 0.2200 - val_loss: 1.6419 - val_a
Epoch 9/300
1/1 [==============================] - 2s 2s/step - loss: 1.6148 - accuracy: 0.2200 - val_loss: 1.5920 - val_a
Epoch 10/300
1/1 [==============================] - 3s 3s/step - loss: 1.5856 - accuracy: 0.2400 - val_loss: 1.5674 - val_a
Epoch 11/300
1/1 [==============================] - 5s 5s/step - loss: 1.5824 - accuracy: 0.2400 - val_loss: 1.5466 - val_a
Epoch 12/300
1/1 [==============================] - 3s 3s/step - loss: 1.5319 - accuracy: 0.2600 - val_loss: 1.5452 - val_a
Epoch 13/300
1/1 [==============================] - 2s 2s/step - loss: 1.5470 - accuracy: 0.2300 - val_loss: 1.5174 - val_a
Epoch 14/300
1/1 [==============================] - 3s 3s/step - loss: 1.5055 - accuracy: 0.3500 - val_loss: 1.5135 - val_a
Epoch 15/300
1/1 [==============================] - 3s 3s/step - loss: 1.4860 - accuracy: 0.3600 - val_loss: 1.5008 - val_a
Epoch 16/300
1/1 [==============================] - 3s 3s/step - loss: 1.4843 - accuracy: 0.4100 - val_loss: 1.4628 - val_a
Epoch 17/300
1/1 [==============================] - 2s 2s/step - loss: 1.4662 - accuracy: 0.4000 - val_loss: 1.4573 - val_a
Epoch 18/300
1/1 [==============================] - 3s 3s/step - loss: 1.4943 - accuracy: 0.3800 - val_loss: 1.4604 - val_a
Epoch 19/300
1/1 [==============================] - 4s 4s/step - loss: 1.4751 - accuracy: 0.3800 - val_loss: 1.4548 - val_a
Epoch 20/300
1/1 [==============================] - 7s 7s/step - loss: 1.4436 - accuracy: 0.4600 - val_loss: 1.4406 - val_a
Epoch 21/300
1/1 [==============================] - 3s 3s/step - loss: 1.4411 - accuracy: 0.4300 - val_loss: 1.4390 - val_a
Epoch 22/300
1/1 [==============================] - 3s 3s/step - loss: 1.4386 - accuracy: 0.4300 - val_loss: 1.4219 - val_a
Epoch 23/300
1/1 [==============================] - 3s 3s/step - loss: 1.4020 - accuracy: 0.4500 - val_loss: 1.4022 - val_a
Epoch 24/300
1/1 [==============================] - 3s 3s/step - loss: 1.4168 - accuracy: 0.5100 - val_loss: 1.3918 - val_a
Epoch 25/300
1/1 [==============================] - 3s 3s/step - loss: 1.3946 - accuracy: 0.5200 - val_loss: 1.3819 - val_a
Epoch 26/300
1/1 [==============================] - 3s 3s/step - loss: 1.3841 - accuracy: 0.4200 - val_loss: 1.3490 - val_a
Epoch 27/300
1/1 [==============================] - 3s 3s/step - loss: 1.3487 - accuracy: 0.5500 - val_loss: 1.3722 - val_a
Epoch 28/300
```

```
EvalData = Model.evaluate(Validate_Dataset, verbose = 0)
print("Validation Loss: "+str(round(EvalData[0]*100, 2))+"%"+"\nValidation Accuracy: "+str(round(EvalData[1]*100, 2)
```

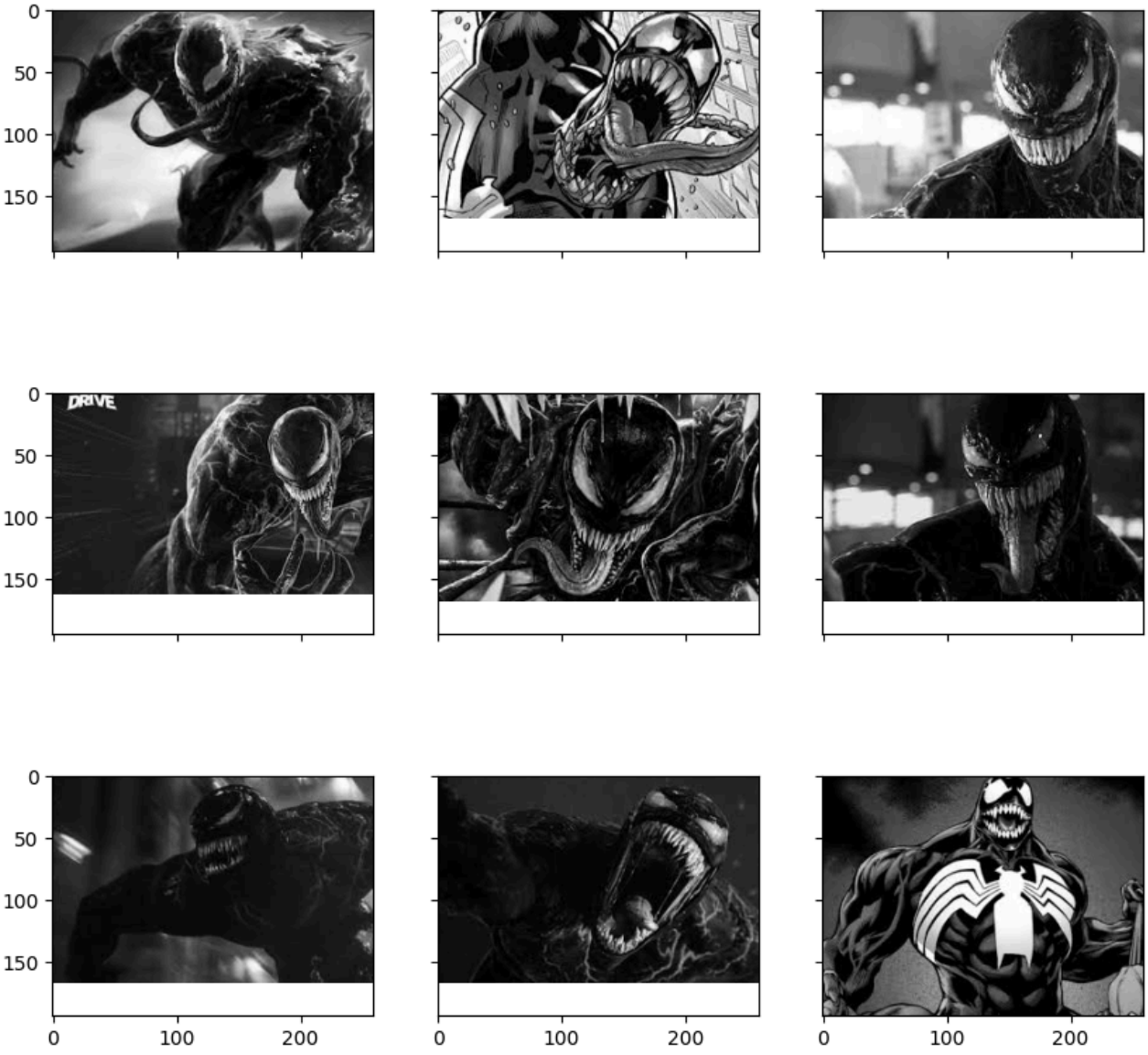
```
Validation Loss: 53.02%
Validation Accuracy: 92.0%
```

```
import tensorflow
from google.colab.patches import cv2_imshow
Testing = cv2.imread("/content/drive/MyDrive/CPE 019/Assignment 9.1/Villains/Darth Vader/Vader 20.jpg")
resize = tensorflow.image.resize(Testing, (100, 100))
cv2_imshow(Testing)
Val = Model.predict(np.expand_dims(resize/255, 0))
Index = np.argmax(Val, axis = None, out = None)

match Index:
  case 0:
    print("This is Darth vader")
  case 1:
    print("This is Green goblin")
  case 2:
    print("This is Joker")
  case 3:
    print("This is Thanos")
  case 4:
    print("This is Venom")
```

```
1/1 [==============================] - 0s 133ms/step
This is Darth vader
```

Perform image augmentation

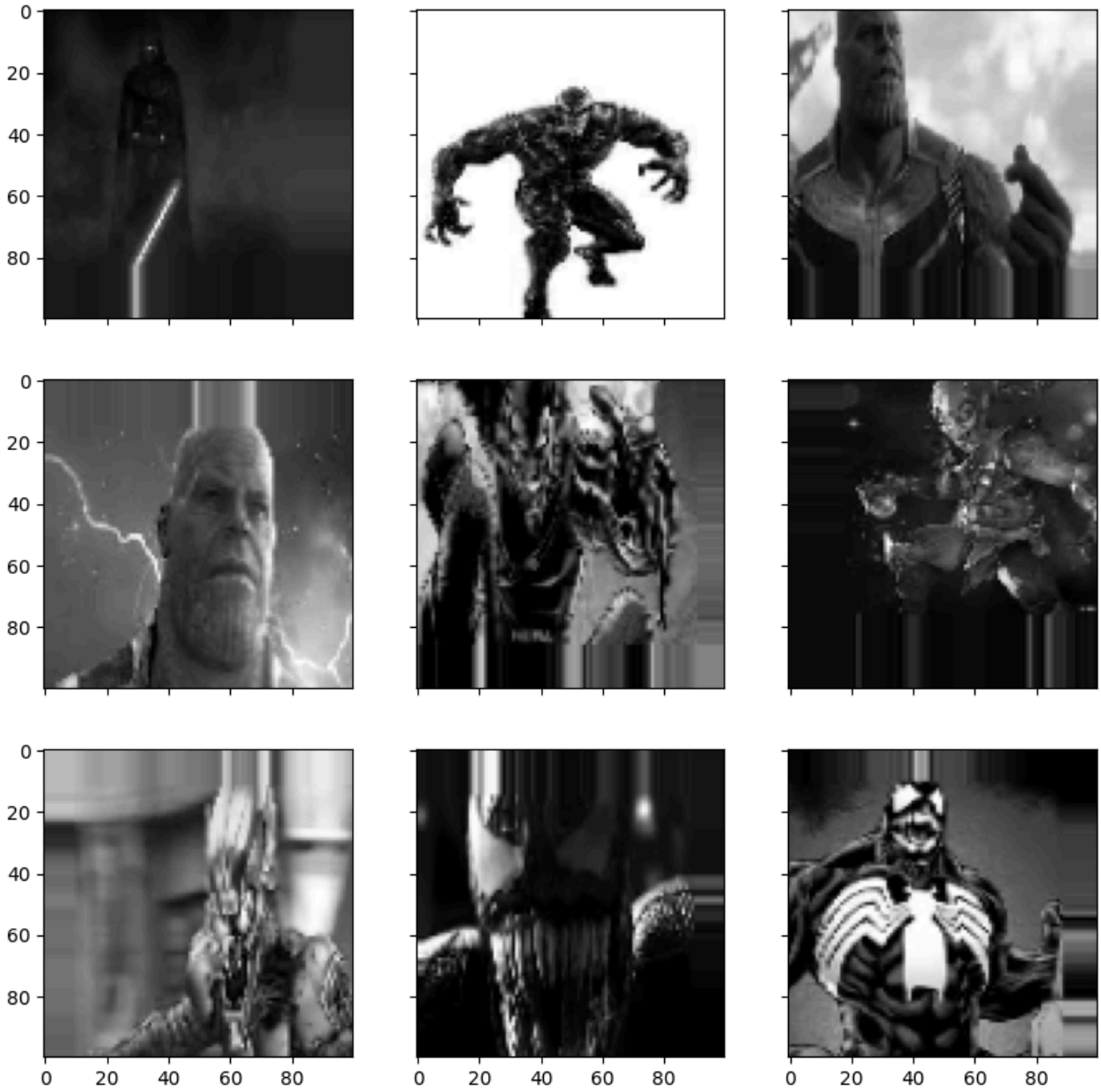
```
import os
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing.image import ImageDataGenerator, load_img

Dataset_2 = '/content/drive/MyDrive/CPE 019/Assignment 9.1/Villains'

def LoadData(directory):
  Images_2 = []
  Labels_2 = []
  for Labes in os.listdir(directory):
    Labels_dir = os.path.join(directory, Labes)
    for IMG_file in os.listdir(Labels_dir):
      IMG_path = os.path.join(Labels_dir, IMG_file)
      IMG = load_img(IMG_path, color_mode = 'grayscale')
      Images_2.append(IMG)
      Labels_2.append(Labes)
  return Images_2, Labels_2

X_train, y_train = LoadData(Dataset_2)

fig, ax = plt.subplots(3, 3, sharex = True, sharey = True, figsize = (10, 10))
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

```
import os
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing.image import ImageDataGenerator, load_img

Dataset_3 = '/content/drive/MyDrive/CPE 019/Assignment 9.1/Villains'

def LoadData3(directory):
  Images_3 = []
  Labels_3 = []
  for Label3 in os.listdir(directory):
    Lables_dir3 = os.path.join(directory, Label3)
    for IMG_file3 in os.listdir(Lables_dir3):
      IMG_path3 = os.path.join(Lables_dir3, IMG_file3)
      IMG3 = load_img(IMG_path3, color_mode = 'grayscale', target_size = (100, 100))
      Images_3.append(np.array(IMG3))
      Labels_3.append(Label3)
  return np.array(Images_3), np.array(Labels_3)

X_train_3, Y_train_3 = LoadData3(Dataset_3)
X_train_3 = X_train_3.astype('float32') / 255.0
X_train_3 = np.expand_dims(X_train_3, axis = -1)

IDG_3 = ImageDataGenerator(rescale = 1./255,
                           zoom_range = 0.2,
                           horizontal_flip = True,
                           height_shift_range = 0.2,
                           width_shift_range = 0.2,
                           featurewise_center=True,
                           featurewise_std_normalization=True)

IDG_3.fit(X_train_3)
IDG_mean = X_train_3.mean(axis = 0)
IDG_std = X_train_3.std(axis = 0)

for Bx, By in IDG_3.flow(X_train_3, Y_train_3, batch_size = 200, shuffle = True):
  print("Min: ", Bx.min())
  print("Mean: ", Bx.mean())
  print("Max: ", Bx.max())

  fig, ax = plt.subplots(3, 3, sharex = True, sharey = True, figsize = (10, 10))
  for i in range(3):
    for j in range(3):
      ax[i][j].imshow(Bx[i * 3 + j].reshape(100, 100), cmap = plt.get_cmap("gray"))
  plt.show()
  break
```

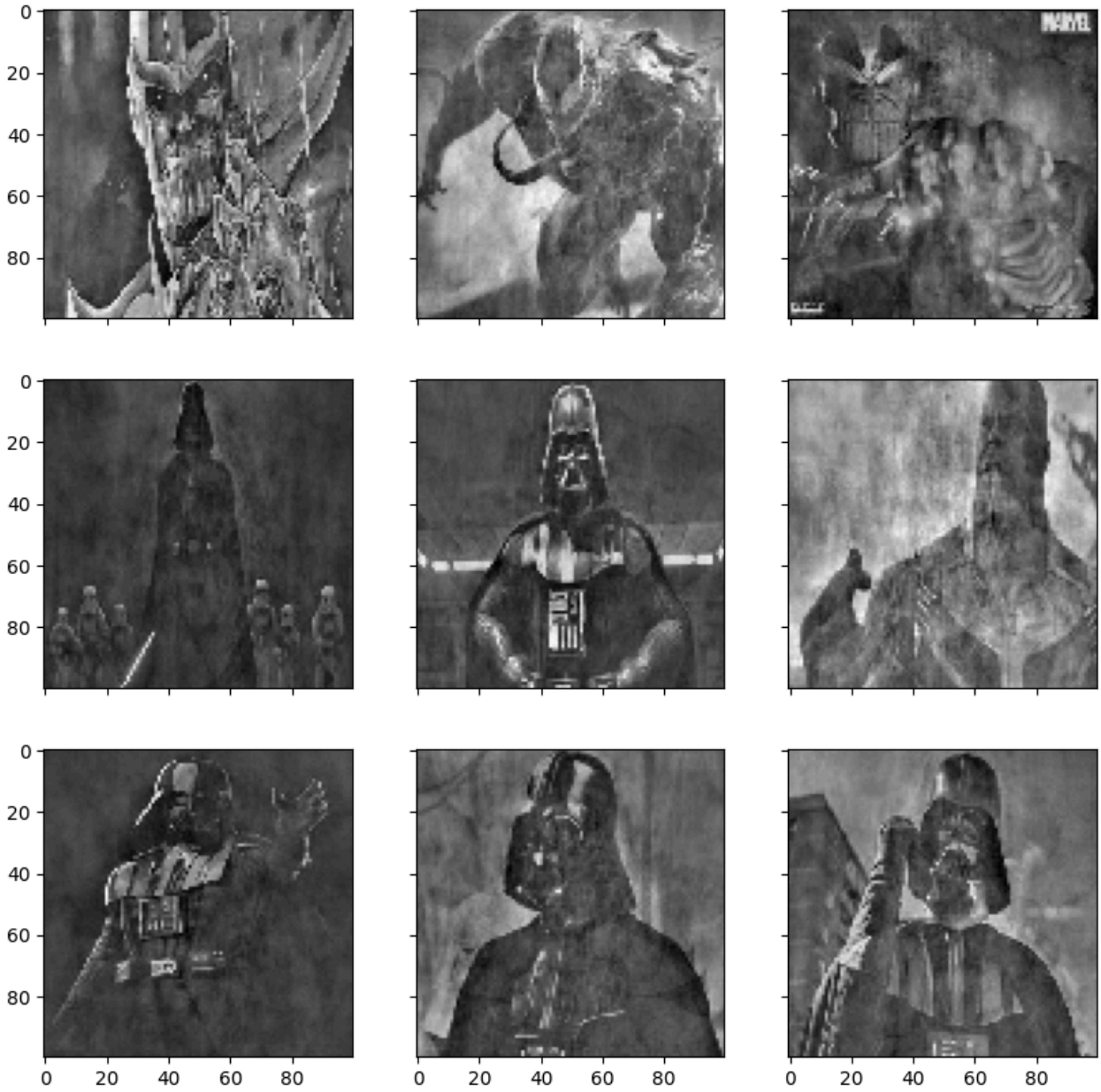
```
Min:  -1.1818268
Mean:  -0.01299767
Max:  2.3549316
```



Perform ZCA whitening of your images

```
IDG_4 = ImageDataGenerator(zca_whitening = True,
                           featurewise_center = True,
                           featurewise_std_normalization = True)
IDG_4.fit(X_train_3)

for Bx4, By4 in IDG_4.flow(X_train_3, Y_train_3, batch_size = 200, shuffle = True):
  print("Min: ", Bx4.min())
  print("Mean: ", Bx4.mean())
  print("Max: ", Bx4.max())

  fig, ax = plt.subplots(3, 3, sharex = True, sharey = True, figsize = (10, 10))
  for i in range(3):
    for j in range(3):
      ax[i][j].imshow(Bx4[i * 3 + j].reshape(100, 100), cmap = plt.get_cmap("gray"))
  plt.show()
  break
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/preprocessing/image.py:1451: UserWarning: This ImageDataGenerat
  warnings.warn(
Min:  -0.5320778
Mean:  -0.0010841907
Max:  1.3302853
```



Augment data with random rotations, shifts, and flips

Random Rotations

```
import os
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing.image import ImageDataGenerator, load_img

Dataset_5 = '/content/drive/MyDrive/CPE 019/Assignment 9.1/Villains'

def LoadData5(directory):
  Images_5 = []
  Labels_5 = []
  for Label5 in os.listdir(directory):
    Lables_dir5 = os.path.join(directory, Label5)
    for IMG_file5 in os.listdir(Lables_dir5):
      IMG_path5 = os.path.join(Lables_dir5, IMG_file5)
      IMG5 = load_img(IMG_path5, color_mode = 'grayscale', target_size = (100, 100))
      Images_5.append(np.array(IMG5))
      Labels_5.append(Label5)
  return np.array(Images_5), np.array(Labels_5)

X_train_5, Y_train_5 = LoadData5(Dataset_5)
X_train_5 = X_train_5.astype('float32') / 255.0
X_train_5 = np.expand_dims(X_train_5, axis = -1)

IDG_5 = ImageDataGenerator(rotation_range = 90)

for Bx5, By5 in IDG_5.flow(X_train_5, Y_train_5, batch_size = 200, shuffle = False):
  fig, ax = plt.subplots(3, 3, sharex = True, sharey = True, figsize = (10, 10))
  for i in range(3):
    for j in range(3):
      ax[i][j].imshow(Bx5[i * 3 + j].reshape(100, 100), cmap = plt.get_cmap("gray"))
  plt.show()
  break
```

Random Shifts

```
import os
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing.image import ImageDataGenerator, load_img

Dataset_6 = '/content/drive/MyDrive/CPE 019/Assignment 9.1/Villains'

def LoadData6(directory):
  Images_6 = []
  Labels_6 = []
  for Label6 in os.listdir(directory):
    Lables_dir6 = os.path.join(directory, Label6)
    for IMG_file6 in os.listdir(Lables_dir6):
      IMG_path6 = os.path.join(Lables_dir6, IMG_file6)
      IMG6 = load_img(IMG_path6, color_mode = 'grayscale', target_size = (100, 100))
      Images_6.append(np.array(IMG6))
      Labels_6.append(Label6)
  return np.array(Images_6), np.array(Labels_6)

X_train_6, Y_train_6 = LoadData6(Dataset_6)
X_train_6 = X_train_6.astype('float32') / 255.0
X_train_6 = np.expand_dims(X_train_6, axis = -1)

IDG_6 = ImageDataGenerator(height_shift_range = 0.2, width_shift_range = 0.2)

for Bx6, By6 in IDG_6.flow(X_train_6, Y_train_6, batch_size = 200, shuffle = False):
  fig, ax = plt.subplots(3, 3, sharex = True, sharey = True, figsize = (10, 10))
  for i in range(3):
    for j in range(3):
      ax[i][j].imshow(Bx6[i * 3 + j].reshape(100, 100), cmap = plt.get_cmap("gray"))
  plt.show()
  break
```

Random Flips

```
import os
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing.image import ImageDataGenerator, load_img

Dataset_7 = '/content/drive/MyDrive/CPE 019/Assignment 9.1/Villains'

def LoadData7(directory):
  Images_7 = []
  Labels_7 = []
  for Label7 in os.listdir(directory):
    Lables_dir7 = os.path.join(directory, Label7)
    for IMG_file7 in os.listdir(Lables_dir7):
      IMG_path7 = os.path.join(Lables_dir7, IMG_file7)
      IMG7 = load_img(IMG_path7, color_mode = 'grayscale', target_size = (100, 100))
      Images_7.append(np.array(IMG7))
      Labels_7.append(Label7)
  return np.array(Images_7), np.array(Labels_7)

X_train_7, Y_train_7 = LoadData7(Dataset_7)
X_train_7 = X_train_7.astype('float32') / 255.0
X_train_7 = np.expand_dims(X_train_7, axis = -1)

IDG_7 = ImageDataGenerator(height_shift_range = True, vertical_flip = True)

for Bx7, By7 in IDG_7.flow(X_train_7, Y_train_7, batch_size = 200, shuffle = False):
  fig, ax = plt.subplots(3, 3, sharex = True, sharey = True, figsize = (10, 10))
  for i in range(3):
    for j in range(3):
      ax[i][j].imshow(Bx7[i * 3 + j].reshape(100, 100), cmap = plt.get_cmap("gray"))
  plt.show()
  break
```

Save augmented image data to disk