Technological Institute of the Philippines	Quezon City - Computer Engineering
Course Code:	CPE 019
Code Title:	Emerging Technologies in CpE 2 - 2nd Semester
ACTIVITY NO.	**Assignment 9.1 : Convolutional Neural Network**
Name	Dela Cruz, Irish
Section	CPE32S3
Date Performed:	04/24/2024
Date Submitted:	04/27/2024
Instructor:	Engr. Roman M. Richard

Introduction of Dataset

The "Sportball" dataset is a collection of images curated for multiclass image classification tasks, specifically focusing on recognizing various types of sports balls. The dataset comprises images labeled with different categories corresponding to distinct sports equipment, including soccer balls, basketballs, footballs, tennis balls, and more.

Link: https://www.kaggle.com/datasets/samuelcortinhas/sports-balls-multiclass-image-classification

Dataset Characteristics:

- **Multivariate:** Images with multiple features.
- **Subject Area:** Sports and Image Classification.

Features: (categorical and integer type)

1. Type of sports ball (e.g., soccer, basketball, football).
2. Image attributes (e.g., size, color, texture).

Instances:

- 48,842 labeled images.

Applications: The Sportball dataset is valuable for various applications in computer vision and machine learning, including:

- Sports equipment recognition and classification.

- Image-based sports analytics and tracking.
- Augmented reality applications in sports training and broadcasting.
- Training and evaluating deep learning models for image classification tasks.
- Understanding visual perception and categorization of sports-related objects.

With its labeled images and diverse range of sports ball categories, the Sportball dataset offers a rich resource for research and development in image classification and sports analytics.

Installing of Requirement Packages

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

```
Drive already mounted at /content/drive; to attempt to forcibly remount,
```

```
import os
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.model_selection import train_test_split
```

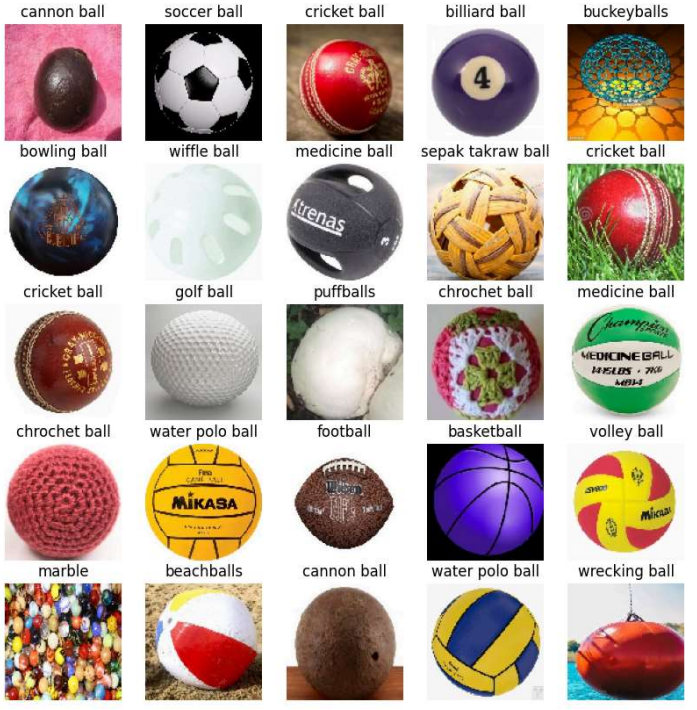
Reading Dataset

```
def split_data(train_dir, test_dir, validation_split=0.1, shuffle=True, seed
    train_data = []
    test_data = []
    for label in os.listdir(train_dir):
        label_dir = os.path.join(train_dir, label)
        filenames = os.listdir(label_dir)
        np.random.shuffle(filenames)
        split_index = int(len(filenames) * (1 - validation_split))
        train_data.extend([{'filename': os.path.join(label, f), 'label': lab
        test_data.extend([{'filename': os.path.join(label, f), 'label': labe

    train_df = pd.DataFrame(train_data)
    test_df = pd.DataFrame(test_data)

    valid_data = []
    for label in os.listdir(test_dir):
        label_dir = os.path.join(test_dir, label)
        filenames = os.listdir(label_dir)
        valid_data.extend([{'filename': os.path.join(label, f), 'label': lab

    valid_df = pd.DataFrame(valid_data)

    if shuffle:
        train_df = train_df.sample(frac=1, random_state=seed).reset_index(dr
        valid_df = valid_df.sample(frac=1, random_state=seed).reset_index(dr
        test_df = test_df.sample(frac=1, random_state=seed).reset_index(drop

    return train_df, valid_df, test_df
```

```
def create_model_data(train_df, valid_df, test_df, batch_size):
    train_datagen = ImageDataGenerator(rescale=1./255)
    valid_datagen = ImageDataGenerator(rescale=1./255)
    test_datagen = ImageDataGenerator(rescale=1./255)

    train_generator = train_datagen.flow_from_dataframe(
        dataframe=train_df,
        directory=train_dir,
        x_col='filename',
        y_col='label',
        target_size=(100, 100),
        batch_size=batch_size,
        class_mode='categorical'
    )

    valid_generator = valid_datagen.flow_from_dataframe(
        dataframe=valid_df,
        directory=test_dir,
        x_col='filename',
        y_col='label',
        target_size=(100, 100),
        batch_size=batch_size,
        class_mode='categorical'
    )

    test_generator = test_datagen.flow_from_dataframe(
        dataframe=test_df,
        directory=test_dir,
        x_col='filename',
        y_col='label',
        target_size=(100, 100),
        batch_size=batch_size,
        class_mode='categorical'
    )

    return train_generator, valid_generator, test_generator
```

```
def show_images(generator):
    classes = list(generator.class_indices.keys())
    images, labels = next(generator)
    plt.figure(figsize=(10, 10))
    for i in range(25):
        plt.subplot(5, 5, i + 1)
        plt.imshow(images[i])
        plt.title(classes[np.argmax(labels[i])])
        plt.axis('off')
    plt.show()


train_dir = '/content/drive/MyDrive/Sports balls - multiclass image classifi
test_dir = '/content/drive/MyDrive/Sports balls - multiclass image classific
validation_dir = '/content/drive/MyDrive/Sports balls - multiclass image cla
train_df, valid_df, test_df = split_data(train_dir, test_dir)


batch_size = 50
train_gen, valid_gen, test_gen = create_model_data(train_df, valid_df, test_


show_images(train_gen)
```

```
Found 3222 validated image filenames belongin
Found 150 validated image filenames belonging
Found 0 validated image filenames belonging t
/usr/local/lib/python3.10/dist-packages/keras
  warnings.warn(
```



Remarks for Reading the Dataset

The dataset consist if 3222 validated image filename belonging to 30 classes for training and 150 validated image filenames belonging to 30 classes. However, there are some invalid images filenames detected in the dataset, that totaling of 373 and will be ignored during processing.

T1. Create a Baseline Model of the CNN

```
#Create cnn model
def define_model():
    model = Sequential()
    model.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_u
    model.add(MaxPooling2D((2, 2)))
    model.add(Flatten())
    model.add(Dense(100, activation='relu', kernel_initializer='he_uniform')
    model.add(Dense(10, activation='softmax'))
    # compile model
    opt = SGD(learning_rate=0.01, momentum=0.9)
    model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['
    return model
```

Remarks for Creating a baseline model for CNN

Creating a baseline CNN involves setting up a simple architecture with convolutional and pooling layers for feature extraction, followed by dense layers for classification. Key steps include choosing activation functions like ReLU, mitigating overfitting with dropout layers and batch normalization, and optimizing hyperparameters like learning rate and batch size. Evaluating performance through visualization and comparison with other models helps establish a benchmark for further enhancements.

T2. Perform image augmentation

One-hot encode labels

```
from tensorflow.keras.utils import to_categorical

def encode_labels(df):
    labels = pd.Categorical(df['label']).codes
    encoded_labels = to_categorical(labels)
    return encoded_labels

def preprocess_data(train_df, test_df):
    train_data = encode_labels(train_df)
    test_data = encode_labels(test_df)
    return train_data, test_data

# Usage
train_dir = '/content/drive/MyDrive/Sports balls - multiclass image classifi
test_dir = '/content/drive/MyDrive/Sports balls - multiclass image classific
train_df, valid_df, test_df = split_data(train_dir, test_dir)

train_data, test_data = preprocess_data(train_df, test_df)


train_data.shape, test_data.shape
```

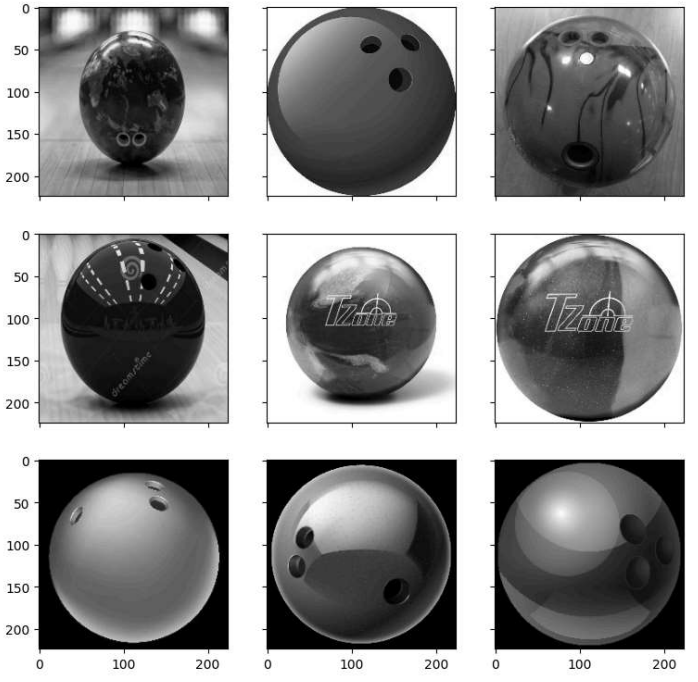
```
((3222, 30), (373, 30))
```

Remarks for One-hot encode labels for train and test of Sports Balls

One-hot encoding labels in the context of the Sports Balls dataset means converting categorical labels into a binary format that machine learning algorithms can better understand. Each label is represented by a vector where all elements are zero except for the index corresponding to the class label, which is set to one. This encoding allows the model to effectively distinguish between different classes without implying any ordinal relationship between them. It's like assigning a unique code to each type of sports ball, making it easier for the model to identify and classify them accurately.

```
import os
from tensorflow.keras.preprocessing.image import load_img
import matplotlib.pyplot as plt
```

```
train_dir = '/content/drive/MyDrive/Sports balls - multiclass image classifica
test_dir = '/content/drive/MyDrive/Sports balls - multiclass image classificat

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()
```



Remarks for Image Augmentation

As you can see above, the sport ball dataset involves applying various transformation to images to increase the diversity of training data without

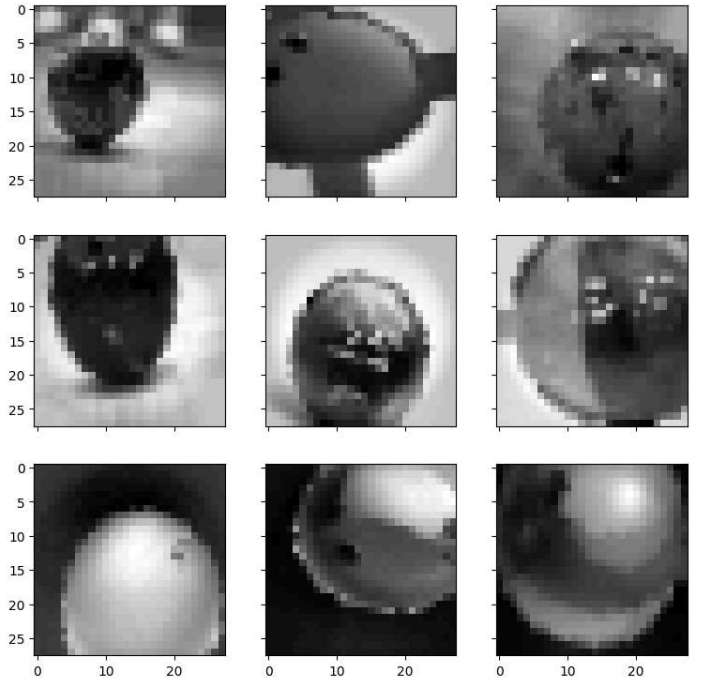
collecting a new samples. By augmenting the dataset, it provides the model with different perspective and angles of sport ball, helping it to learn more effectively.

T3. Perform feature standardization

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

dataset_dir = '/content/drive/MyDrive/Sports balls - multiclass image classifi

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,
            images.append(np.array(img))
            labels.append(label)
    return np.array(images), np.array(labels)
```

```
X_train, y_train = load_data(dataset_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)
datagen.mean = X_train.mean(axis=0)
datagen.std = X_train.std(axis=0)

for X_batch, y_batch in datagen.flow(X_train, y_train, batch_size=9, shuffle
    print(X_batch.min(), X_batch.mean(), X_batch.max())
    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_cma

    plt.show()
    break
```

```
-2.040658 -0.4475686 2.0382812
```



Remark for Feature Standardization

the minimum, mean, and maximum values from the batch printed above are:

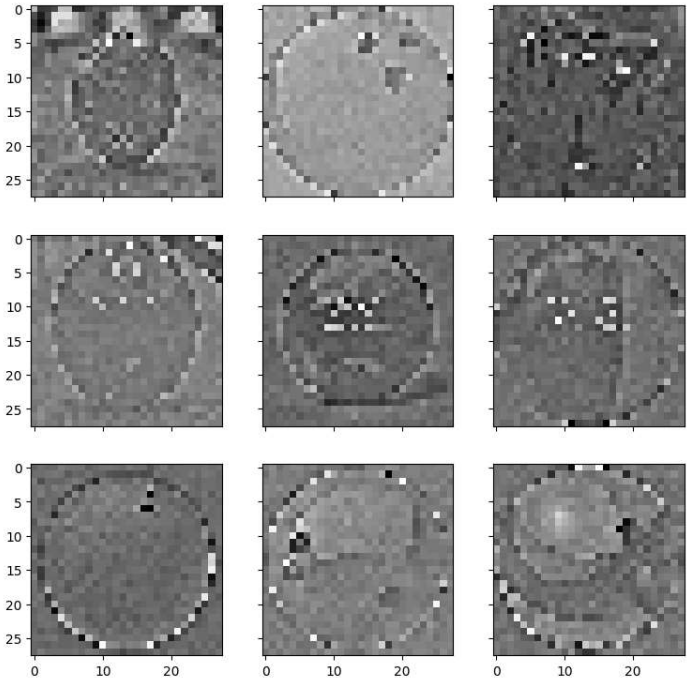
```
-2.040658 -0.4475686 2.0382812
```

In the context it involves transforming the values to mean of 0 and std of 1. It helps to make features comparable and prevents larger scale from dominating the learning process. By standardizing the features, I ensure that each feature contributes equally the model's training, which can improve the performance, particularly to gradient descent-based optimization algorithms

T4. 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
    print(X_batch.min(), X_batch.mean(), X_batch.max())
    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_cma

    plt.show()
    break
```

```
/usr/local/lib/python3.10/dist-packages/keras
  warnings.warn(
-5.545361 -0.035489492 5.5707655
```



Remarks ZCA whitening of your images

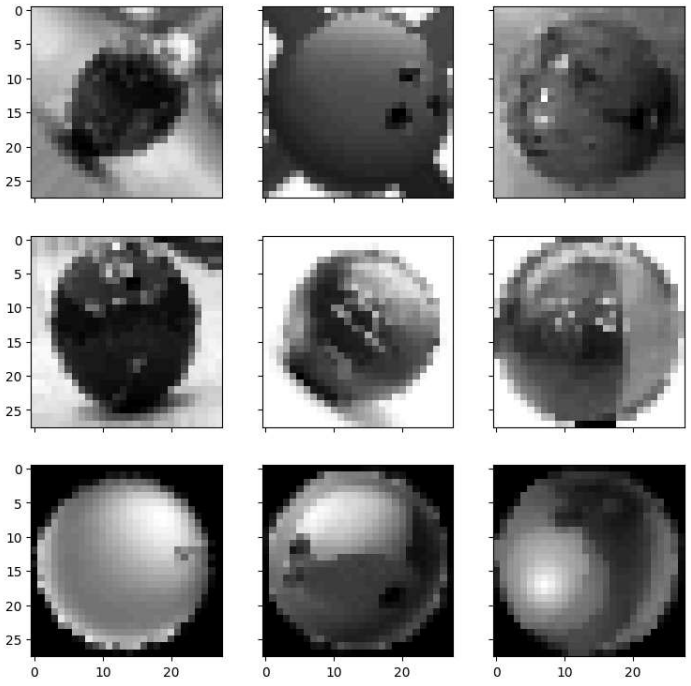
the minimum, mean, and maximum values from the batch printed above are:

```
-5.545361 -0.035489492 5.5707655
```

Applying ZCA whitening involves transforming the pixel values of the images to reduc the correlation between adjacent pixels. It helps the model to learn more by making less redundant and enhancing the discriminative features present in images. It aim to improve the convergence of learning algorithm, especially where the input data is highly correlated or strong spatial dependencies.

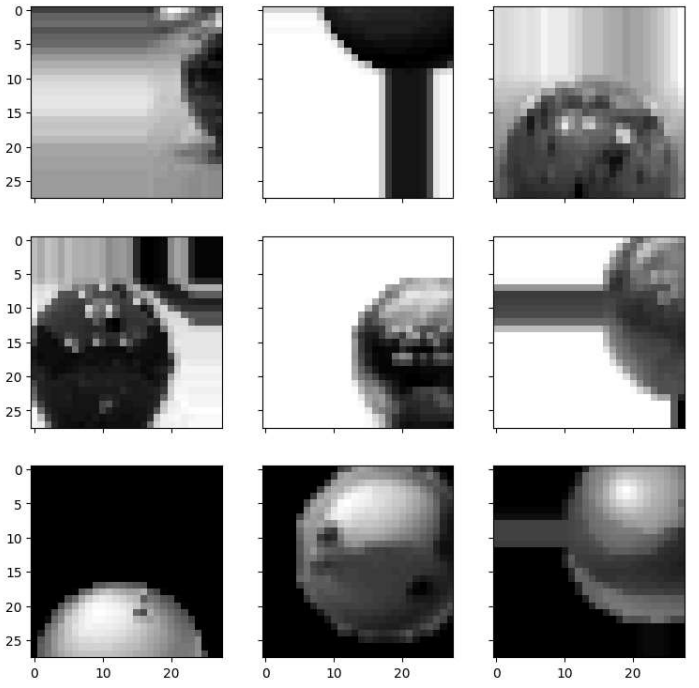
T5. Augment data with random rotations, shifts, and flips

```
# Random Rotations
rotation_range = 90
datagen = ImageDataGenerator(rotation_range=rotation_range)

for X_batch, y_batch in datagen.flow(X_train, y_train, batch_size=9, shuffle=F
    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(

    plt.show()
    break
```

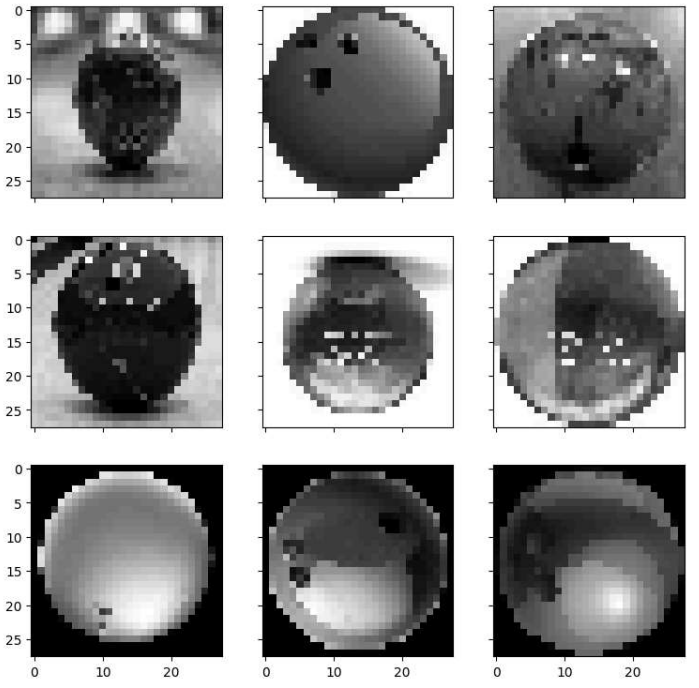


```
#Random Shifts
shift = 0.7
datagen = ImageDataGenerator(
    width_shift_range=shift,
    height_shift_range=shift
)

for X_batch, y_batch in datagen.flow(X_train, y_train, batch_size=9, shuffle=F
```

```
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(

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
    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_cma

    plt.show()
    break
```



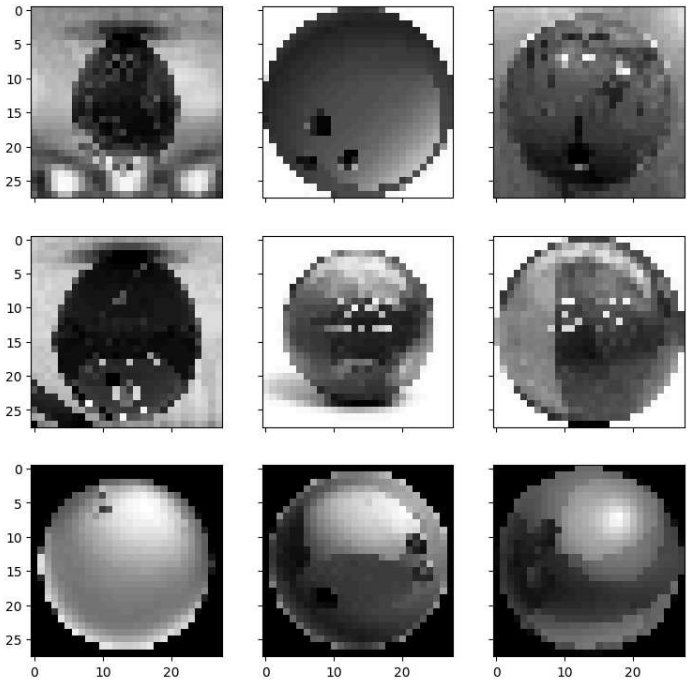
Remarks for Random Rotation, Shift, and Flips

These transformations can include horizontal or vertical flips, rotation, zoom, or changes in brightness or contrast. Since Image Augmentation helps the model to becomde more robust and better able to generalize to unseen data. It's providing

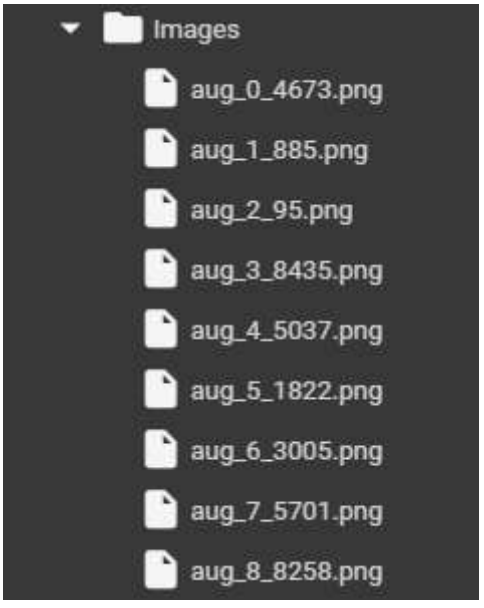
the model with different perspectives and angle of sport ball, helping it to learn more effectively and improve the performance on real-world images.

T6. Save augmented image data to disk

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

dataset_dir = '/content/drive/MyDrive/Sports balls - multiclass image classi

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
            images.append(img)
            labels.append(label)
    return images, labels

images, labels = load_data(dataset_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=

    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_cma

    plt.show()
    break
```



Verifying the save augmented image



Remarks for Saving Augmented Images

Saving Augmented Images allow us to inspect and verify the effectiveness of the augmentation technique applied. It enable us to reuse the augmented dataset for the future experimenys without needing to regenerate the augmented images each time.

Introduction of Dataset

The "Fashion MNIST" dataset is a well-known benchmark dataset used for image classification tasks, particularly in the context of fashion and clothing recognition. It consists of grayscale images depicting various fashion items such as shirts, pants, dresses, shoes, and accessories. The dataset serves as a substitute for the original MNIST dataset, offering a more challenging problem domain while retaining the same structure and dimensions.

Link: https://www.tensorflow.org/datasets/catalog/fashion_mnist

Dataset Characteristics:

- **Multivariate:** Images with multiple pixel values.
- **Subject Area:** Fashion and Image Classification.

Features:

1. Pixel values representing grayscale intensities.
2. Image labels indicating the type of fashion item.

Instances:

- 70,000 labeled images (60,000 for training and 10,000 for testing).

Applications: The Fashion MNIST dataset finds applications in various domains, including:

- Fashion recommendation systems and e-commerce platforms.
- Clothing detection and segmentation in image editing software.
- Fashion trend analysis and forecasting.
- Benchmarking and evaluating machine learning algorithms for image classification tasks.
- Education and research in computer vision and deep learning.

Due to its standardized format and relevance to real-world fashion scenarios, the Fashion MNIST dataset serves as a fundamental resource for developing and evaluating image classification models in the domain of fashion and apparel.

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt

print(tf.__version__)
```

```
     2.15.0
```

```
fashion_mnist = tf.keras.datasets.fashion_mnist

(train_images, train_labels), (test_images, test_labels) = fashion_mnist.loa
```

```
class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
               'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
```

```
train_images.shape
```

```
     (60000, 28, 28)
```

```
len(train_labels)
```

```
     60000
```

```
train_labels
```

```
array([9, 0, 0, ..., 3, 0, 5], dtype=uint8)
```

```
test_images.shape
```

```
(10000, 28, 28)
```

```
len(test_labels)
```

```
10000
```

Preprocess the Data

```
train_images = train_images / 255.0
test_images = test_images / 255.0


plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(train_images[i], cmap=plt.cm.binary)
    plt.xlabel(class_names[train_labels[i]])
plt.show()
```


Ankle boot
T-shirt/top
T-shirt/top
Dress
T-shirt/top
Pullover
Sneaker
Pullover
Sandal
Sandal
T-shirt/top
Ankle boot
Sandal
Sandal
Sneaker
Ankle boot
Trouser
T-shirt/top
Shirt
Coat
Dress
Trouser
Coat
Bag
Coat

T7. Develop a test harness to develop a robust evaluation of a model and establish a baseline of performance for a classification task

Baseline: 1 VGG Blocks

```
import time
start_time = time.time()

import sys
from matplotlib import pyplot
from keras.datasets import fashion_mnist
from keras.utils import to_categorical
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Dense, Flatten, Dropout
from keras.optimizers import SGD


def load_dataset():
  (trainX, trainY), (testX, testY) = fashion_mnist.load_data()
  trainY = to_categorical(trainY)
  testY = to_categorical(testY)
  return trainX, trainY, testX, testY


def prep_pixels(train, test):
  train_norm = train.astype('float32')
  test_norm = test.astype('float32')
  train_norm = train_norm / 255.0
  test_norm = test_norm / 255.0
  return train_norm, test_norm
```

```
def define_model():
    model = Sequential()
    model.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_u
    model.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_u
    model.add(MaxPooling2D((2, 2)))
    model.add(Flatten())
    model.add(Dense(128, activation='relu', kernel_initializer='he_uniform')
    model.add(Dense(10, activation='softmax'))
    # compile model
    opt = SGD(lr=0.001, momentum=0.9)
    model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['
    return model


def summarize_diagnostics(history):
    # plot loss
    pyplot.subplot(211)
    pyplot.title('Cross Entropy Loss')
    pyplot.plot(history.history['loss'], color='blue', label='train')
    pyplot.plot(history.history['val_loss'], color='orange', label='test')
    # plot accuracy
    pyplot.subplot(212)
    pyplot.title('Classification Accuracy')
    pyplot.plot(history.history['accuracy'], color='blue', label='train')
    pyplot.plot(history.history['val_accuracy'], color='orange', label='test
    # save plot to file
    filename = sys.argv[0].split('/')[-1]
    pyplot.savefig(filename + '_plot.png')
    pyplot.close()

def run_test_harness():
    start_time = time.time()

    trainX, trainY, testX, testY = load_dataset()
    trainX, testX = prep_pixels(trainX, testX)
    model = define_model()
    history = model.fit(trainX, trainY, epochs=10, batch_size=500, validatio
    _, acc = model.evaluate(testX, testY, verbose=0)
    print('> Test Accuracy: %.3f' % (acc * 100.0))
    summarize_diagnostics(history)

run_test_harness()
```

```
WARNING:absl:`lr` is deprecated in Keras optimizer, please use `learning
> Test Accuracy: 90.380
```

```
end_time = time.time()
total_time = end_time - start_time
print("Total time taken:", total_time / 60, "minutes")
```

```
Total time taken: 7.864359366893768 minutes
```

```
from PIL import Image

def load_image(file_path):
    try:
        image = Image.open(file_path)
        return image
    except IOError:
        print("Unable to load image")
        return None

load_image('/content/drive/MyDrive/Plotting.plot.png')
```

```
Unable to load image
```

Baseline: 2 VGG Blocks

```
def define_model():
    model = Sequential()
    model.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_u
    model.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_u
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_u
    model.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_u
    model.add(MaxPooling2D((2, 2)))
    model.add(Flatten())
    model.add(Dense(128, activation='relu', kernel_initializer='he_uniform')
    model.add(Dense(10, activation='softmax'))
    # compile model
    opt = SGD(lr=0.001, momentum=0.9)
    model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['
    return model
```

```
def summarize_diagnostics(history):
    # plot loss
    pyplot.subplot(211)
    pyplot.title('Cross Entropy Loss')
    pyplot.plot(history.history['loss'], color='blue', label='train')
    pyplot.plot(history.history['val_loss'], color='orange', label='test')
    # plot accuracy
    pyplot.subplot(212)
    pyplot.title('Classification Accuracy')
    pyplot.plot(history.history['accuracy'], color='blue', label='train')
    pyplot.plot(history.history['val_accuracy'], color='orange', label='test
    # save plot to file
    filename = sys.argv[0].split('/')[-1]
    pyplot.savefig(filename + '_plot.png')
    pyplot.close()


def run_test_harness():
    start_time = time.time()

    trainX, trainY, testX, testY = load_dataset()
    trainX, testX = prep_pixels(trainX, testX)
    model = define_model()
    history = model.fit(trainX, trainY, epochs=10, batch_size=500, validatio
    _, acc = model.evaluate(testX, testY, verbose=0)
    print('> Test Accuracy: %.3f' % (acc * 100.0))
    summarize_diagnostics(history)


end_time = time.time()
total_time = end_time - start_time
print("Total time taken:", total_time / 60, "minutes")
```

```
 Total time taken: 13.80118642648061 minutes
```

```
from PIL import Image

def load_image(file_path):
    try:
        image = Image.open(file_path)
        return image
    except IOError:
        print("Unable to load image")
        return None
load_image('/content/drive/MyDrive/Plotting.py_plot.png')
```

```
 Unable to load image
```

Baseline: 3 VGG Blocks

```
def define_model():
    model = Sequential()
    model.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_u
    model.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_u
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_u
    model.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_u
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_
    model.add(Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_
    model.add(MaxPooling2D((2, 2)))
    model.add(Flatten())
    model.add(Dense(128, activation='relu', kernel_initializer='he_uniform')
    model.add(Dense(10, activation='softmax'))
    # compile model
    opt = SGD(lr=0.001, momentum=0.9)
    model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['
    return model


def summarize_diagnostics(history):
  # plot loss
  pyplot.subplot(211)
  pyplot.title('Cross Entropy Loss')
  pyplot.plot(history.history['loss'], color='blue', label='train')
  pyplot.plot(history.history['val_loss'], color='orange', label='test')
  # plot accuracy
  pyplot.subplot(212)
  pyplot.title('Classification Accuracy')
  pyplot.plot(history.history['accuracy'], color='blue', label='train')
  pyplot.plot(history.history['val_accuracy'], color='orange', label='test')
  # save plot to file
  filename = sys.argv[0].split('/')[-1]
  pyplot.savefig(filename + '_plot.png')
  pyplot.close()
```

```
def run_test_harness():
    start_time = time.time()

    trainX, trainY, testX, testY = load_dataset()
    trainX, testX = prep_pixels(trainX, testX)
    model = define_model()
    history = model.fit(trainX, trainY, epochs=10, batch_size=500, validatio
    _, acc = model.evaluate(testX, testY, verbose=0)
    print('> Test Accuracy: %.3f' % (acc * 100.0))
    summarize_diagnostics(history)


end_time = time.time()
total_time = end_time - start_time
print("Total time taken:", total_time / 60, "minutes")
```

```
 Total time taken: 16.609914898872375 minutes
```

```
from PIL import Image

def load_image(file_path):
    try:
        image = Image.open(file_path)
        return image
    except IOError:
        print("Unable to load image")
        return None
load_image('/content/drive/MyDrive/Plotting.py_plot.png')
```

```
 Unable to load image
```

Remarks for testing harness to develop a robust evaluation of a model and establish a baseline of performance for a classification task

The results can be summarized below, although we must assume some variance in these results given the stochastic nature of the algorithm:

```
VGG 1: 10.775%
VGG 2: 13.801%
VGG 3: 16.620%
```

The results of the model on test dataset showed an improvement in classification accuracy with each increase in depth of the model. All 3 models showed the same pattern of dramatic overfitting at around 5 epoch

These results suggest that the model with 3 VGG blocks is good starting point or baseline model for experiments.

T8. Explore extensions to a baseline model to improve learning and model capacity.

Baseline model with Increasing Dropout

```
def define_model():
    model = Sequential()
    model.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_u
    model.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_u
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_u
    model.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_u
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_
    model.add(Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_
    model.add(MaxPooling2D((2, 2)))
    model.add(Flatten())
    model.add(Dense(128, activation='relu', kernel_initializer='he_uniform')
    model.add(Dense(10, activation='softmax'))
    # compile model
    opt = SGD(lr=0.001, momentum=0.9)
    model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['
    return model
```

```
def summarize_diagnostics(history):
    # plot loss
    pyplot.subplot(211)
    pyplot.title('Cross Entropy Loss')
    pyplot.plot(history.history['loss'], color='blue', label='train')
    pyplot.plot(history.history['val_loss'], color='orange', label='test')
    # plot accuracy
    pyplot.subplot(212)
    pyplot.title('Classification Accuracy')
    pyplot.plot(history.history['accuracy'], color='blue', label='train')
    pyplot.plot(history.history['val_accuracy'], color='orange', label='test
    # save plot to file
    filename = sys.argv[0].split('/')[-1]
    pyplot.savefig(filename + '_plot.png')
    pyplot.close()


def run_test_harness():
    start_time = time.time()

    trainX, trainY, testX, testY = load_dataset()
    trainX, testX = prep_pixels(trainX, testX)
    model = define_model()
    datagen = ImageDataGenerator(width_shift_range=0.1, height_shift_range=0
    it_train = datagen.flow(trainX, trainY, batch_size=500)
    steps = int(trainX.shape[0] / 64)
    history = model.fit_generator(it_train, steps_per_epoch=steps, epochs=10
    _, acc = model.evaluate(testX, testY, verbose=0)
    print('> %.3f' % (acc * 100.0))
    summarize_diagnostics(history)


end_time = time.time()

total_time = end_time - start_time
print("Total time taken:", total_time/60, "minutes")
```

```
Total time taken: 24.45235433578491 minutes
```

```
from PIL import Image

def load_image(file_path):
    try:
        image = Image.open(file_path)
        return image
    except IOError:
        print("Unable to load image")
        return None

load_image('/content/drive/MyDrive/Plotting.py_plot.png')
```

```
 Unable to load image
```

Baseline model with Dropout and Data Augmentation

```
def define_model():
    model = Sequential()
    model.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_u
    model.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_u
    model.add(MaxPooling2D((2, 2)))
    model.add(Dropout(0.2))
    model.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_u
    model.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_u
    model.add(MaxPooling2D((2, 2)))
    model.add(Dropout(0.2))
    model.add(Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_
    model.add(Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_
    model.add(MaxPooling2D((2, 2)))
    model.add(Dropout(0.2))
    model.add(Flatten())
    model.add(Dense(128, activation='relu', kernel_initializer='he_uniform')
    model.add(Dropout(0.2))
    model.add(Dense(10, activation='softmax'))
    # compile model
    opt = SGD(lr=0.001, momentum=0.9)
    model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['
    return model
```

```
def summarize_diagnostics(history):
    # plot loss
    pyplot.subplot(211)
    pyplot.title('Cross Entropy Loss')
    pyplot.plot(history.history['loss'], color='blue', label='train')
    pyplot.plot(history.history['val_loss'], color='orange', label='test')
    # plot accuracy
    pyplot.subplot(212)
    pyplot.title('Classification Accuracy')
    pyplot.plot(history.history['accuracy'], color='blue', label='train')
    pyplot.plot(history.history['val_accuracy'], color='orange', label='test
    # save plot to file
    filename = sys.argv[0].split('/')[-1]
    pyplot.savefig(filename + '_plot.png')
    pyplot.close()


def run_test_harness():
    start_time = time.time()

    trainX, trainY, testX, testY = load_dataset()
    trainX, testX = prep_pixels(trainX, testX)
    model = define_model()
    datagen = ImageDataGenerator(width_shift_range=0.1, height_shift_range=0
    it_train = datagen.flow(trainX, trainY, batch_size=500)
    steps = int(trainX.shape[0] / 64)
    history = model.fit_generator(it_train, steps_per_epoch=steps, epochs=10
    _, acc = model.evaluate(testX, testY, verbose=0)
    print('> %.3f' % (acc * 100.0))
    summarize_diagnostics(history)


end_time = time.time()

total_time = end_time - start_time
print("Total time taken:", total_time/60, "minutes")
```

```
Total time taken: 30.348976735273997 minutes
```

```
from PIL import Image

def load_image(file_path):
    try:
        image = Image.open(file_path)
        return image
    except IOError:
        print("Unable to load image")
        return None

load_image('/content/drive/MyDrive/Plotting.py_plot.png')
```

```
Unable to load image
```

Baseline + Increasing Dropout + Data Augmentation + Batch Normalization:

```
def define_model():
    model = Sequential()
    model.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_u
    model.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_u
    model.add(MaxPooling2D((2, 2)))
    model.add(Dropout(0.2))
    model.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_u
    model.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_u
    model.add(MaxPooling2D((2, 2)))
    model.add(Dropout(0.3))
    model.add(Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_
    model.add(Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_
    model.add(MaxPooling2D((2, 2)))
    model.add(Dropout(0.4))
    model.add(Flatten())
    model.add(Dense(128, activation='relu', kernel_initializer='he_uniform')
    model.add(Dropout(0.5))
    model.add(Dense(10, activation='softmax'))

    opt = SGD(learning_rate=0.001, momentum=0.9)
    model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['
    return model
```

```
def summarize_diagnostics(history):
    pyplot.subplot(211)
    pyplot.title('Cross Entropy Loss')
    pyplot.plot(history.history['loss'], color='blue', label='train')
    pyplot.plot(history.history['val_loss'], color='orange', label='test')

    pyplot.subplot(212)
    pyplot.title('Classification Accuracy')
    pyplot.plot(history.history['accuracy'], color='blue', label='train')
    pyplot.plot(history.history['val_accuracy'], color='orange', label='test
```

```
def run_test_harness():
    start_time = time.time()

    trainX, trainY, testX, testY = load_dataset()
    trainX, testX = prep_pixels(trainX, testX)
    model = define_model()
    datagen = ImageDataGenerator(width_shift_range=0.1, height_shift_range=0
    it_train = datagen.flow(trainX, trainY, batch_size=500)
    steps = int(trainX.shape[0] / 64)
    history = model.fit_generator(it_train, steps_per_epoch=steps, epochs=10
    _, acc = model.evaluate(testX, testY, verbose=0)
    print('> %.3f' % (acc * 100.0))
    summarize_diagnostics(history)


end_time = time.time()

total_time = end_time - start_time
print("Total time taken:", total_time/60, "minutes")
```

```
     Total time taken: 30.695006457964578 minutes
```

```
summarize_diagnostics(history)
```

```
from PIL import Image
```