```
# Import libraries

%tensorflow_version 2.x

import csv
import numpy as np
import numpy as np
import tensorflow as tf
from tensorflow import keras
from PIL import Image
import matplotlib.pyplot as plt
```

```
TensorFlow 2.x selected.
```

```
# Extract image data

!unzip -q "/content/IMG.zip"
```

```
# Read sample data from images csv file

with open("/content/driving_log.csv") as csv_file:
    csv_reader = csv.reader(csv_file, delimiter=',')
    line_count = 0

    print("\nColumns are: Center Image, Left Image, Right Image, Steering Angle, Throttle, Brake, Speed\n")

    for row in csv_reader:
        if line_count < 10:
            print(f'{", ".join(row)}')
            line_count += 1

    print(f'\nProcessed {line_count} lines.')
```

```
Columns are: Center Image, Left Image, Right Image, Steering Angle, Throttle, Brake, Speed

/Users/gerrymigwi/Desktop/data/IMG/center_2020_03_06_12_46_04_920.jpg,  /Users/gerrymigwi/Desktop/data/IMG/left_2020_0
/Users/gerrymigwi/Desktop/data/IMG/center_2020_03_06_12_46_05_025.jpg,  /Users/gerrymigwi/Desktop/data/IMG/left_2020_0
/Users/gerrymigwi/Desktop/data/IMG/center_2020_03_06_12_46_05_128.jpg,  /Users/gerrymigwi/Desktop/data/IMG/left_2020_0
/Users/gerrymigwi/Desktop/data/IMG/center_2020_03_06_12_46_05_231.jpg,  /Users/gerrymigwi/Desktop/data/IMG/left_2020_0
/Users/gerrymigwi/Desktop/data/IMG/center_2020_03_06_12_46_05_336.jpg,  /Users/gerrymigwi/Desktop/data/IMG/left_2020_0
/Users/gerrymigwi/Desktop/data/IMG/center_2020_03_06_12_46_05_444.jpg,  /Users/gerrymigwi/Desktop/data/IMG/left_2020_0
/Users/gerrymigwi/Desktop/data/IMG/center_2020_03_06_12_46_05_565.jpg,  /Users/gerrymigwi/Desktop/data/IMG/left_2020_0
/Users/gerrymigwi/Desktop/data/IMG/center_2020_03_06_12_46_05_666.jpg,  /Users/gerrymigwi/Desktop/data/IMG/left_2020_0
/Users/gerrymigwi/Desktop/data/IMG/center_2020_03_06_12_46_05_770.jpg,  /Users/gerrymigwi/Desktop/data/IMG/left_2020_0
/Users/gerrymigwi/Desktop/data/IMG/center_2020_03_06_12_46_05_871.jpg,  /Users/gerrymigwi/Desktop/data/IMG/left_2020_0

Processed 10 lines.
```

```
# Define final training data
X = []
y = []
```

```
# Initialize image data and labels - center

with open("/content/driving_log.csv") as csv_file:
    csv_reader = csv.reader(csv_file, delimiter=',')
    line_count = 0

    print("\nColumns are: Center Image, Left Image, Right Image, Steering Angle, Throttle, Brake, Speed\n")

    for row in csv_reader:
        # Center
        center_img_path = row[0].split("/")[-1]
        center_img = Image.open("/content/IMG/" + center_img_path)
        X.append(np.array(center_img))
        y.append(float(row[3]))
        center_flip = np.fliplr(np.array(center_img))
        X.append(center_flip)
        y.append(float(row[3]) * -1)

        line_count += 1

    print(f'\nProcessed {line_count} center images.')
```

```
Columns are: Center Image, Left Image, Right Image, Steering Angle, Throttle, Brake, Speed

Processed 12907 center images.
```

```
# Initialize image data and labels - left

with open("/content/driving_log.csv") as csv_file:
    csv_reader = csv.reader(csv_file, delimiter=',')
    line_count = 0

    for row in csv_reader:
        # Left
        left_img_path = row[1].split("/")[-1]
        left_img = Image.open("/content/IMG/" + left_img_path)
        X.append(np.array(left_img))
        y.append(float(row[3]) + 0.3)
        left_flip = np.fliplr(np.array(left_img))
        X.append(left_flip)
        y.append((float(row[3]) * -1) - 0.2)

        line_count += 1

    print(f'\nProcessed {line_count} left images.')
```

```
Processed 12907 left images.
```

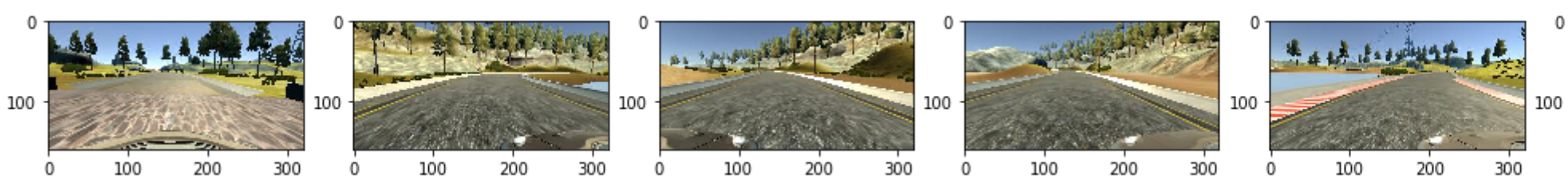
```
# Initialize image data and labels - right

with open("/content/driving_log.csv") as csv_file:
    csv_reader = csv.reader(csv_file, delimiter=',')
    line_count = 0

    for row in csv_reader:
        # Right
        right_img_path = row[2].split("/")[-1]
        right_img = Image.open("/content/IMG/" + right_img_path)
        X.append(np.array(right_img))
        y.append(float(row[3]) - 0.2)
        right_flip = np.fliplr(np.array(right_img))
        X.append(right_flip)
        y.append((float(row[3]) * -1) + 0.2)

        line_count += 1

    print(f'\nProcessed {line_count} right images.')
```

```
Processed 12907 right images.
```

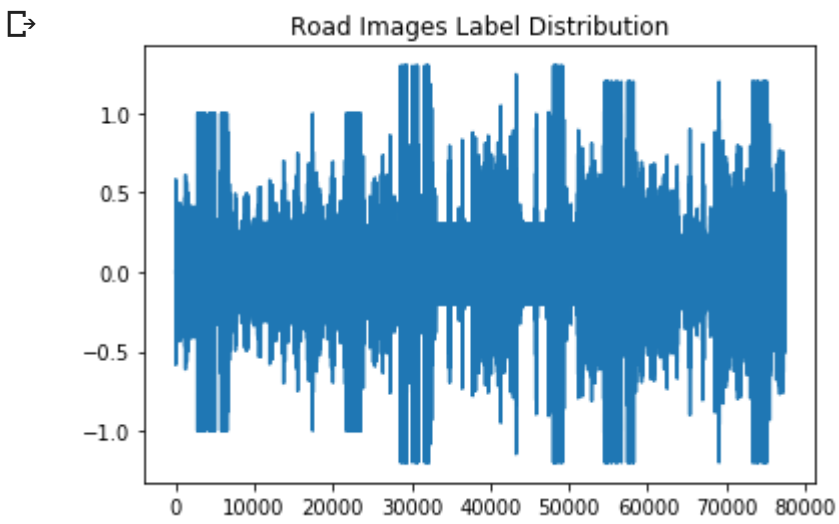
```
# Shape and inspect data

X = np.asarray(X)
y = np.asarray(y)

print("Total images shape:", X.shape)
print("Total labels shape:", y.shape)
```

```
Total images shape: (77442, 160, 320, 3)
Total labels shape: (77442,)
```

```
# Visualize random samples of road images

print("\tSample Road Images\n")
fig, ax = plt.subplots(1, 6, figsize=(20, 20))
for row in ax:
    rand_img = np.random.randint(0, len(X))
    row.imshow(np.squeeze(X[rand_img]))
plt.show()
```




```
# Visualize image label distribution

plt.title("Road Images Label Distribution")
plt.plot(list(range(len(X))), y)
plt.show()
```




```
# Create the model

model = tf.keras.Sequential([
          tf.keras.layers.Lambda(lambda x: (x / 255.0) - 0.5, input_shape=(160, 320, 3)),
          tf.keras.layers.Cropping2D(cropping=((50, 20), (0, 0))),
          tf.keras.layers.Conv2D(128, (5, 5), activation="relu"),
          tf.keras.layers.MaxPooling2D(3, 3),
          tf.keras.layers.Conv2D(64, (5, 5), activation="relu"),
          tf.keras.layers.MaxPooling2D(3, 3),
          tf.keras.layers.Conv2D(64, (5, 5), activation="relu"),
          tf.keras.layers.MaxPooling2D(3, 3),
          tf.keras.layers.Flatten(),
          tf.keras.layers.Dense(1),
        ])
```

```
# Model summary

model.summary()
```

```
Model: "sequential_19"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lambda_19 (Lambda)           (None, 160, 320, 3)       0
_________________________________________________________________
cropping2d_19 (Cropping2D)   (None, 90, 320, 3)        0
_________________________________________________________________
conv2d_33 (Conv2D)           (None, 86, 316, 128)      9728
_________________________________________________________________
max_pooling2d_31 (MaxPooling (None, 28, 105, 128)      0
_________________________________________________________________
conv2d_34 (Conv2D)           (None, 24, 101, 64)       204864
_________________________________________________________________
max_pooling2d_32 (MaxPooling (None, 8, 33, 64)         0
_________________________________________________________________
conv2d_35 (Conv2D)           (None, 4, 29, 64)         102464
_________________________________________________________________
max_pooling2d_33 (MaxPooling (None, 1, 9, 64)          0
_________________________________________________________________
flatten_19 (Flatten)         (None, 576)               0
_________________________________________________________________
dense_19 (Dense)             (None, 1)                 577
=================================================================
Total params: 317,633
Trainable params: 317,633
Non-trainable params: 0
_________________________________________________________________
```

```
# Epoch callback class
```

```
class epochCallback(tf.keras.callbacks.Callback):
    def on_epoch_end(self, epoch, logs = {}):
        if(logs.get('val_loss') < 0.024):
            epoch_val_loss = logs.get('val_loss')
            print(f'\n\nModel validation loss currently at {epoch_val_loss}. Stopping training...\n\n')
            self.model.stop_training = True
```

```
# Compile the model

model.compile(loss="mean_squared_error", optimizer="adam")
```

```
# Fit the model

history = model.fit(X, y, epochs=1, validation_split=0.2, shuffle=True, callbacks = [epochCallback()])
```

```
Train on 61953 samples, validate on 15489 samples
61953/61953 [==============================] - 62s 1ms/sample - loss: 0.0370 - val_loss: 0.0324
```

```
# Save the model

model.save("driver.h5")
```