In [ ]:
```
import numpy as np
import tensorflow as tf
import keras
from keras import utils
from keras import layers
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Activation, Dropout, Flatten, Dense, Reshape
import matplotlib.pyplot as plt
import pandas as pd

from astroNN.datasets import galaxy10
from astroNN.datasets.galaxy10 import galaxy10cls_lookup, galaxy10_confusion
```

In [ ]:
```
images, labels = galaxy10.load_data() #loading images
labels = utils.to_categorical(labels, 10) #convering labels to categorical values
```

In [ ]:
```
image_count = len(list(images))
print(image_count)
print(images)
print(labels)
print(type(images))
print(type(labels))
```

In [ ]:
```
batch_size = 64
img_height = 69
img_width = 69
shuffle_buffer_size = 100
```

In [ ]:
```
train_ds = tf.data.Dataset.from_tensor_slices((images[0:15428], labels[0:15428]))
train_ds = train_ds.shuffle(shuffle_buffer_size).batch(batch_size)
#train_images = images[:15428]
#train_labels = labels[:15428]
val_ds = tf.data.Dataset.from_tensor_slices((images[15429:17428], labels[15429:17428]))
val_ds = val_ds.shuffle(shuffle_buffer_size).batch(batch_size)
test_ds = tf.data.Dataset.from_tensor_slices((images[17429:], labels[17429:]))
test_ds = test_ds.shuffle(shuffle_buffer_size).batch(batch_size)
```

In [ ]:
```
input_shape = (69, 69, 3)
num_classes = 10

model = Sequential()
model.add(layers.experimental.preprocessing.Rescaling(1./255, input_shape = input_shape))
model.add(layers.Conv2D(16, kernel_size = (3,3), padding = 'same', activation = 'relu'))
model.add(layers.MaxPooling2D())
model.add(layers.Conv2D(32, (3,3), padding = 'same', activation = 'relu'))
model.add(layers.MaxPooling2D())
model.add(layers.Conv2D(64, (3,3), padding = 'same', activation = 'relu'))
model.add(layers.MaxPooling2D())
model.add(layers.Flatten())
model.add(layers.Dense(128, activation = 'relu'))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(num_classes, activation = 'softmax'))
```

In [ ]:
```
model.compile(optimizer = tf.keras.optimizers.Adam(learning_rate=0.0005), loss = tf.keras.losses.CategoricalCrossentropy(), metri
model.summary()
```

In [ ]:
```
epochs = 25
classifier = model.fit(
    train_ds,
    #train_images,
    #train_labels,
    batch_size = batch_size,
    validation_data = val_ds,
    epochs = epochs
)
```

In [ ]:
```
evaluation = model.evaluate(test_ds)
```

In [ ]:
```
plt.figure(figsize=(15,10))
ax = plt.subplot(1, 2, 1)
plt.plot(classifier.history["accuracy"])
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.title("Training Accuracy")
ax = plt.subplot(1, 2, 2)
plt.plot(classifier.history["val_accuracy"])
plt.ylabel('Val_Accuracy')
plt.xlabel('Epoch')
plt.title("Validation Accuracy")
```

In [ ]:
```
plt.figure(figsize=(15,10))
ax = plt.subplot(1, 2, 1)
plt.plot(classifier.history["loss"])
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.title("Training Loss")
ax = plt.subplot(1, 2, 2)
plt.plot(classifier.history["val_loss"])
plt.ylabel('Val_Loss')
plt.xlabel('Epoch')
plt.title("Validation Loss")
```

In [ ]:
```
check = model.predict(test_ds)
check = pd.DataFrame(check)
check.columns = ['Disk, Face-on, No Spiral', 'Smooth, Completely round', 'Smooth, in-between round',
                 'Smooth, Cigar shaped', 'Disk, Edge-on, Rounded Bulge', 'Disk, Edge-on, Boxy Bulge',
                 'Disk, Edge-on, No Bulge', 'Disk, Face-on, Tight Spiral', 'Disk, Face-on, Medium Spiral', 'Disk, Face-on, Loose
```

In [ ]:
```
check
```

In [ ]:
```
# Please notice predicted_labels are labels predicted from neural network. test_labels are ground truth from the dataset
predicted_labels = model.predict(images[17429:])

# Convert predicted_labels to class
prediction_class = np.argmax(predicted_labels, axis=1)

# Convert test_labels to class
test_class = np.argmax(labels[17429:], axis=1)

# Prepare a confusion matrix
confusion_matrix = np.zeros((10,10))

# create the confusion matrix
for counter, i in enumerate(prediction_class):
    confusion_matrix[i, test_class[counter]] += 1

# Plot the confusion matrix
galaxy10_confusion(confusion_matrix)
```

In [ ]:
```
```