In [1]:
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt
import numpy as np
import os
import tensorflow as tf
import path
from tensorflow.keras.preprocessing import image_dataset_from_directory
```

In [2]:
```
import pandas as pd
import numpy as np
import os
import csv
import requests
import urllib.request
```

In [3]:
```
batch_size = 32
IMG_SIZE = (224, 224)

from tensorflow.keras.preprocessing.image import ImageDataGenerator
train_datagen = ImageDataGenerator(rescale=1./255,
                                   shear_range=0.1,
                                   zoom_range=0.1,
                                   horizontal_flip=True)
# All images will be rescaled by 1./255
train_datagen = ImageDataGenerator(rescale=1/255)

# Flow training images in batches of 32 using train_datagen generator
train_set = train_datagen.flow_from_directory(
        'C:/Users/rpshc/Desktop/Assingment/Base_Folder/train/',  # This is the so
        target_size=(224, 224),  # All images will be resized to 200 x 200
        batch_size=batch_size,
        # Specify the classes explicitly
        classes = ['AreaGraph', 'BarGraph', 'LineGraph', 'Map', 'ParetoChart',
               'PieChart','RadarPlot','ScatterGraph','Table','VennDiagram'],
        # Since we use categorical_crossentropy loss, we need categorical labels
        class_mode='categorical')
```

```
Found 439 images belonging to 10 classes.
```

In [4]:
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# All images will be rescaled by 1./255
val_generator = ImageDataGenerator(rescale=1/255)

# Flow training images in batches of 32 using val_generator generator
val_set = train_datagen.flow_from_directory(
        'C:/Users/rpshc/Desktop/Assingment/Base_Folder/val/',  # This is the sour
        target_size=(224, 224),  # All images will be resized to 200 x 200
        batch_size=batch_size,
        # Specify the classes explicitly
        classes = ['AreaGraph', 'BarGraph', 'LineGraph', 'Map', 'ParetoChart',
               'PieChart','RadarPlot','ScatterGraph','Table','VennDiagram'],
        # Since we use categorical_crossentropy loss, we need categorical labels
        class_mode='categorical')
```

```
Found 190 images belonging to 10 classes.
```

In [5]:
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# All images will be rescaled by 1./255
test_datagen = ImageDataGenerator(rescale=1/255)

# Flow training images in batches of 32 using test_generator generator
test_set = train_datagen.flow_from_directory(
        'C:/Users/rpshc/Desktop/Assingment/Base_Folder/test/',  # This is the sou
        target_size=(224, 224),  # All images will be resized to 200 x 200
        batch_size=batch_size,
        # Specify the classes explicitly
        classes = ['AreaGraph', 'BarGraph', 'LineGraph', 'Map', 'ParetoChart',
               'PieChart','RadarPlot','ScatterGraph','Table','VennDiagram'],
        # Since we use categorical_crossentropy loss, we need categorical labels
        class_mode='categorical')
```

```
Found 90 images belonging to 10 classes.
```

In [6]:
```
preprocess_input = tf.keras.applications.mobilenet_v2.preprocess_input
```

In [7]:
```
import tensorflow as tf
```

In [8]:
```
data_augmentation = tf.keras.Sequential([
  tf.keras.layers.experimental.preprocessing.RandomFlip('horizontal'),
  tf.keras.layers.experimental.preprocessing.RandomRotation(0.2),
])
```

In [9]:
```
# Create the base model from the pre-trained model MobileNet V2
IMG_SHAPE = IMG_SIZE + (3,)
base_model = tf.keras.applications.MobileNetV2(input_shape=IMG_SHAPE,
                                               include_top=False,
                                               weights='imagenet')
```

In [10]:
```
image_batch, label_batch = next(iter(train_set))
feature_batch = base_model(image_batch)
print(feature_batch.shape)
```

```
(32, 7, 7, 1280)
```

In [11]:
```
base_model.trainable = False
```

In [12]:
```
# Let's take a look at the base model architecture
base_model.summary()
```

```
Model: "mobilenetv2_1.00_224"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_1 (InputLayer)         [(None, 224, 224, 3)      0         
_________________________________________________________________
Conv1 (Conv2D)               (None, 112, 112, 32)      864       
_________________________________________________________________
bn_Conv1 (BatchNormalization (None, 112, 112, 32)      128       
_________________________________________________________________
Conv1_relu (ReLU)            (None, 112, 112, 32)      0         
_________________________________________________________________
```

In [13]:
```
global_average_layer = tf.keras.layers.GlobalAveragePooling2D()
feature_batch_average = global_average_layer(feature_batch)
print(feature_batch_average.shape)
```

```
(32, 1280)
```

In [14]:
```
prediction_layer = tf.keras.layers.Dense(1)
prediction_batch = prediction_layer(feature_batch_average)
print(prediction_batch.shape)
```

```
(32, 1)
```

In [16]:
```
inputs = tf.keras.Input(shape=(224, 224, 3))
x = data_augmentation(inputs)
x = preprocess_input(x)
x = base_model(x, training=False)
x = global_average_layer(x)
x = tf.keras.layers.Dropout(0.2)(x)
outputs = prediction_layer(x)
model = tf.keras.Model(inputs, outputs)
```

In [17]:
```
base_learning_rate = 0.0001
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

In [18]:
```
model.summary()
```

```
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_3 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
sequential (Sequential)      (None, 224, 224, 3)       0
_________________________________________________________________
tf.math.truediv_1 (TFOpLambd (None, 224, 224, 3)       0
_________________________________________________________________
tf.math.subtract_1 (TFOpLamb (None, 224, 224, 3)       0
_________________________________________________________________
mobilenetv2_1.00_224 (Functi (None, 7, 7, 1280)        2257984
_________________________________________________________________
global_average_pooling2d (Gl (None, 1280)              0
_________________________________________________________________
dropout_1 (Dropout)          (None, 1280)              0
_________________________________________________________________
dense (Dense)                (None, 1)                 1281
=================================================================
Total params: 2,259,265
Trainable params: 1,281
Non-trainable params: 2,257,984
_________________________________________________________________
```

In [19]:
```
initial_epochs = 10

loss0, accuracy0 = model.evaluate(val_set)
```

```
6/6 [==============================] - 8s 1s/step - loss: 1.1921e-07 - accurac
y: 0.9000
```

In [20]:
```
print("initial loss: {:.2f}".format(loss0))
print("initial accuracy: {:.2f}".format(accuracy0))
```

```
initial loss: 0.00
initial accuracy: 0.90
```

In [25]:
```
history = model.fit(train_set,
                    epochs=initial_epochs,
                    validation_data=val_set,
                    verbose=1)
```

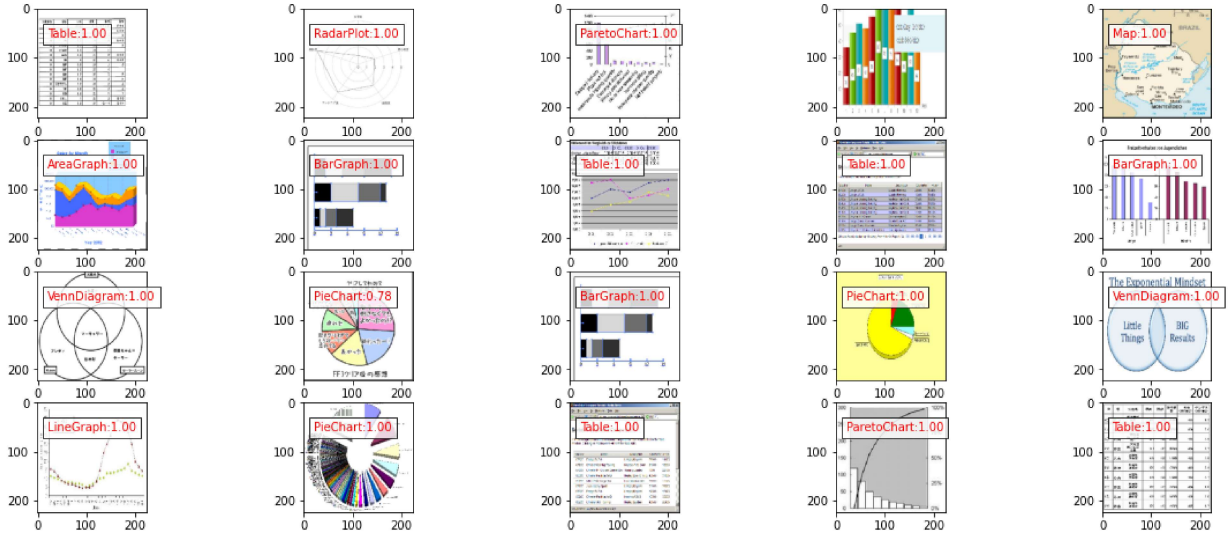
```
Epoch 1/10
14/14 [==============================] - 22s 2s/step - loss: 4.5740e-04 - acc:
1.0000 - val_loss: 1.2228 - val_acc: 0.9000
Epoch 2/10
14/14 [==============================] - 21s 2s/step - loss: 2.5190e-04 - acc:
1.0000 - val_loss: 1.3428 - val_acc: 0.8895
Epoch 3/10
14/14 [==============================] - 21s 2s/step - loss: 9.2844e-05 - acc:
1.0000 - val_loss: 1.4532 - val_acc: 0.8947
Epoch 4/10
14/14 [==============================] - 21s 2s/step - loss: 2.0060 - acc: 0.89
29 - val_loss: 1.2681 - val_acc: 0.8737
Epoch 5/10
14/14 [==============================] - 21s 2s/step - loss: 0.0309 - acc: 0.99
54 - val_loss: 1.1671 - val_acc: 0.8895
Epoch 6/10
14/14 [==============================] - 21s 2s/step - loss: 0.0208 - acc: 0.99
09 - val_loss: 1.0158 - val_acc: 0.9105
Epoch 7/10
14/14 [==============================] - 21s 2s/step - loss: 0.0020 - acc: 1.00
00 - val_loss: 1.0660 - val_acc: 0.9053
Epoch 8/10
14/14 [==============================] - 21s 2s/step - loss: 6.2979e-04 - acc:
1.0000 - val_loss: 1.1099 - val_acc: 0.9053
Epoch 9/10
14/14 [==============================] - 21s 2s/step - loss: 0.0552 - acc: 0.99
09 - val_loss: 7.6209 - val_acc: 0.3316
Epoch 10/10
14/14 [==============================] - 21s 2s/step - loss: 0.6396 - acc: 0.89
98 - val_loss: 0.8953 - val_acc: 0.9105
```

In [35]:
```
model.save('My_model.h5')
```

In [36]:
```
from tensorflow.keras.models import load_model
model_1 = load_model('My_model.h5')
model =load_model('model.h5')
```

In [37]:
```
import numpy as np
from keras.preprocessing import image
fig=plt.figure(figsize=(20, 8))
columns = 5
rows = 4
for i in range(columns*rows):
    fig.add_subplot(rows, columns, i+1)
    img1 = image.load_img('C:\\Users\\rpshc\\Desktop\\Assingment\\Base_Folder\\te
    img = image.img_to_array(img1)
    img = img/255
    img = np.expand_dims(img, axis=0)
    result = model_1.predict(img, batch_size=None,steps=1) #gives all class prob.
    if result[0][0]>0.5:
        value = 'AreaGraph:%1.2f'%result[0][0]
        plt.text(20, 58,value,color='red',fontsize=10,bbox=dict(facecolor='white'

    elif result[0][1]>0.5:
        value = 'BarGraph:%1.2f'%result[0][1]
        plt.text(20, 58,value,color='red',fontsize=10,bbox=dict(facecolor='white'

    elif result[0][2]>0.5:
        value ='LineGraph:%1.2f'%result[0][2]
        plt.text(20, 58,value,color='red',fontsize=10,bbox=dict(facecolor='white'

    elif result[0][3]>0.5:
        value ='Map:%1.2f'%result[0][3]
        plt.text(20, 58,value,color='red',fontsize=10,bbox=dict(facecolor='white'

    elif result[0][4]>0.5:
        value ='ParetoChart:%1.2f'%result[0][4]
        plt.text(20, 58,value,color='red',fontsize=10,bbox=dict(facecolor='white'

    elif result[0][5]>0.5:
        value ='PieChart:%1.2f'%result[0][5]
        plt.text(20, 58,value,color='red',fontsize=10,bbox=dict(facecolor='white'

    elif result[0][6]>0.5:
        value ='RadarPlot:%1.2f'%result[0][6]
        plt.text(20, 58,value,color='red',fontsize=10,bbox=dict(facecolor='white'

    elif result[0][7]>0.5:
        value ='ScatterGraph:%1.2f'%result[0][7]
        plt.text(20, 58,value,color='red',fontsize=10,bbox=dict(facecolor='white'

    elif result[0][8]>0.5:
        value ='Table:%1.2f'%result[0][8]
        plt.text(20, 58,value,color='red',fontsize=10,bbox=dict(facecolor='white'

    elif result[0][9]>0.5:
        value ='VennDiagram:%1.2f'%result[0][9]
        plt.text(20, 58,value,color='red',fontsize=10,bbox=dict(facecolor='white'

    plt.imshow(img1)
print(result)
```

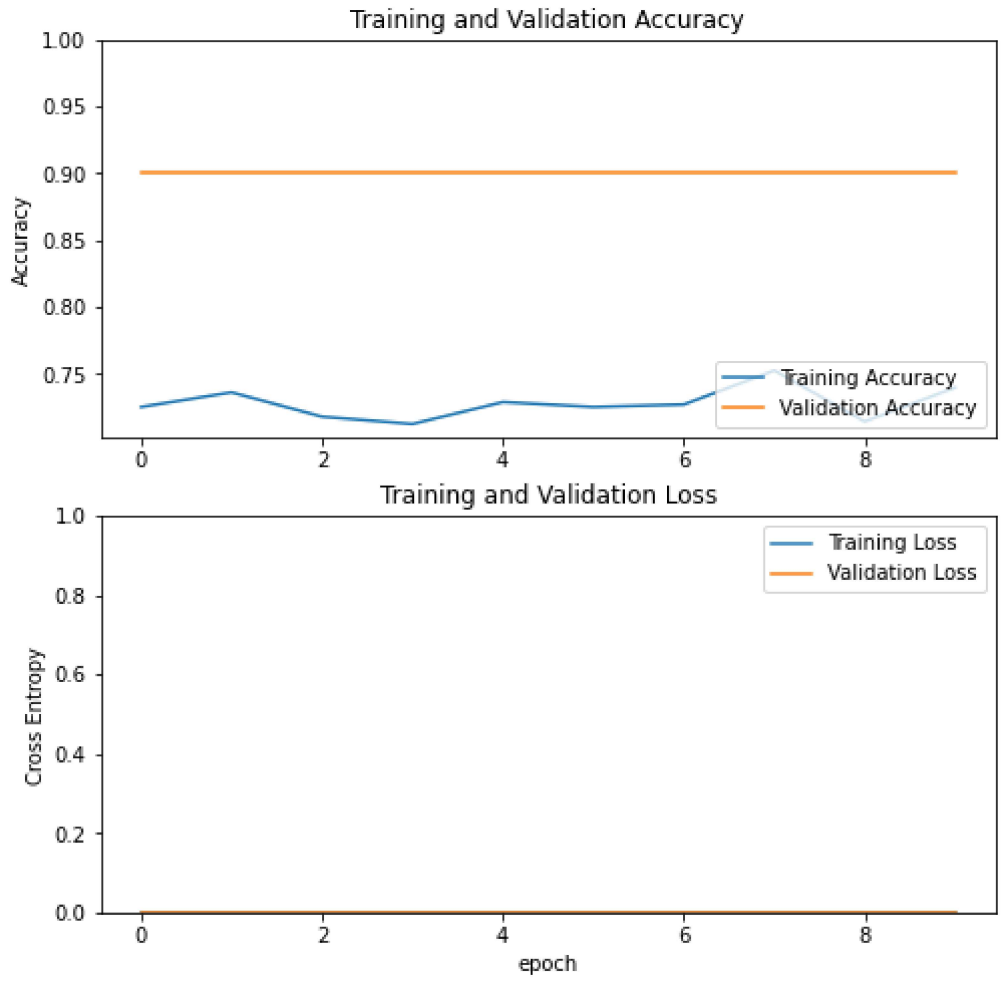
```
[[2.9512560e-13 1.4992901e-05 2.3104337e-07 1.8289147e-04 3.4184898e-11
  1.8455636e-13 2.4935706e-15 1.5139937e-12 9.9980193e-01 9.0172300e-11]]
```



In [402]:
```
model.metrics_names
```

Out[402]:
```
['loss', 'acc']
```

In [403]:
```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

plt.figure(figsize=(8, 8))
plt.subplot(2, 1, 1)
plt.plot(acc, label='Training Accuracy')
plt.plot(val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.ylabel('Accuracy')
plt.ylim([min(plt.ylim()),1])
plt.title('Training and Validation Accuracy')

plt.subplot(2, 1, 2)
plt.plot(loss, label='Training Loss')
plt.plot(val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.ylabel('Cross Entropy')
plt.ylim([0,1.0])
plt.title('Training and Validation Loss')
plt.xlabel('epoch')
plt.show()
```



In [404]:
```
#Fine tuning
base_model.trainable = True
```

In [405]:
```
# Let's take a look to see how many layers are in the base model
print("Number of layers in the base model: ", len(base_model.layers))

# Fine-tune from this layer onwards
fine_tune_at = 100

# Freeze all the layers before the `fine_tune_at` layer
for layer in base_model.layers[:fine_tune_at]:
    layer.trainable =  False
```

```
Number of layers in the base model:  154
```

In [406]:
```
model.compile(loss='categorical_crossentropy',
              optimizer = tf.keras.optimizers.RMSprop(learning_rate=base_learning
              metrics=['accuracy'])
```

In [407]:
```
model_1.summary()
len(model_1.trainable_variables)
```

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_10 (Conv2D)           (None, 222, 222, 16)      448
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 111, 111, 16)      0
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 109, 109, 32)      4640
_________________________________________________________________
max_pooling2d_11 (MaxPooling (None, 54, 54, 32)        0
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 52, 52, 64)        18496
_________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 26, 26, 64)        0
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 24, 24, 64)        36928
_________________________________________________________________
max_pooling2d_13 (MaxPooling (None, 12, 12, 64)        0
_________________________________________________________________
conv2d_14 (Conv2D)           (None, 10, 10, 64)        36928
_________________________________________________________________
max_pooling2d_14 (MaxPooling (None, 5, 5, 64)          0
_________________________________________________________________
flatten_2 (Flatten)          (None, 1600)              0
_________________________________________________________________
dense_5 (Dense)              (None, 128)               204928
_________________________________________________________________
dense_6 (Dense)              (None, 10)                1290
=================================================================
Total params: 303,658
Trainable params: 303,658
Non-trainable params: 0
_________________________________________________________________
```

Out[407]: 14

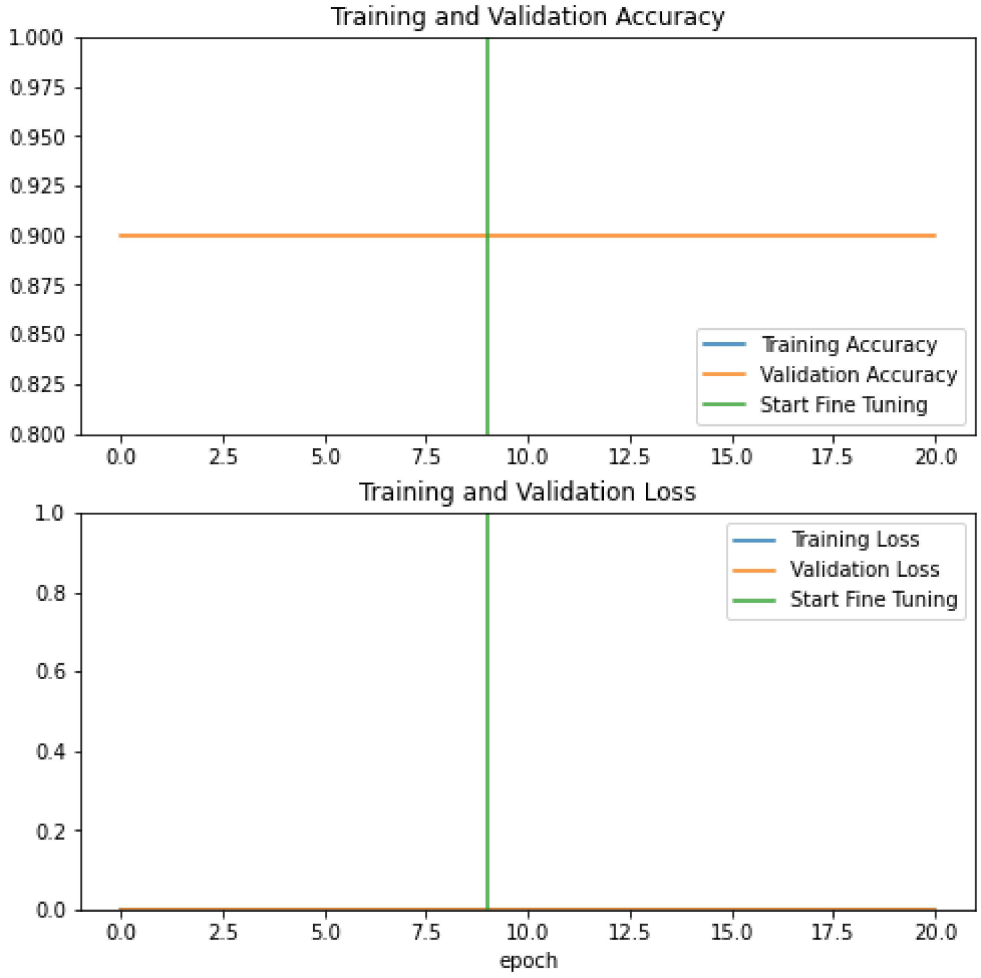
In [349]:
```
fine_tune_epochs = 10
total_epochs =  initial_epochs + fine_tune_epochs

history_fine = model_1.fit(train_set,
                           epochs=total_epochs,
                           initial_epoch=history.epoch[-1],
                           validation_data=val_set)
```

```
Epoch 10/20
14/14 [==============================] - 29s 2s/step - loss: 1.1921e-07 - accur
acy: 0.7178 - val_loss: 1.1921e-07 - val_accuracy: 0.9000
Epoch 11/20
14/14 [==============================] - 27s 2s/step - loss: 1.1921e-07 - accur
acy: 0.7560 - val_loss: 1.1921e-07 - val_accuracy: 0.9000
Epoch 12/20
14/14 [==============================] - 27s 2s/step - loss: 1.1921e-07 - accur
acy: 0.7178 - val_loss: 1.1921e-07 - val_accuracy: 0.9000
Epoch 13/20
14/14 [==============================] - 27s 2s/step - loss: 1.1921e-07 - accur
acy: 0.7196 - val_loss: 1.1921e-07 - val_accuracy: 0.9000
Epoch 14/20
14/14 [==============================] - 27s 2s/step - loss: 1.1921e-07 - accur
acy: 0.7269 - val_loss: 1.1921e-07 - val_accuracy: 0.9000
Epoch 15/20
14/14 [==============================] - 27s 2s/step - loss: 1.1921e-07 - accur
acy: 0.7433 - val_loss: 1.1921e-07 - val_accuracy: 0.9000
Epoch 16/20
14/14 [==============================] - 27s 2s/step - loss: 1.1921e-07 - accur
acy: 0.7378 - val_loss: 1.1921e-07 - val_accuracy: 0.9000
Epoch 17/20
14/14 [==============================] - 25s 2s/step - loss: 1.1921e-07 - accur
acy: 0.7342 - val_loss: 1.1921e-07 - val_accuracy: 0.9000
Epoch 18/20
14/14 [==============================] - 25s 2s/step - loss: 1.1921e-07 - accur
acy: 0.7251 - val_loss: 1.1921e-07 - val_accuracy: 0.9000
Epoch 19/20
14/14 [==============================] - 25s 2s/step - loss: 1.1921e-07 - accur
acy: 0.7378 - val_loss: 1.1921e-07 - val_accuracy: 0.9000
Epoch 20/20
14/14 [==============================] - 26s 2s/step - loss: 1.1921e-07 - accur
acy: 0.7305 - val_loss: 1.1921e-07 - val_accuracy: 0.9000
```

In [38]:
```
model.save('My_model_1.h5')
from tensorflow.keras.models import load_model
model_2 = load_model('My_model_1.h5')
```

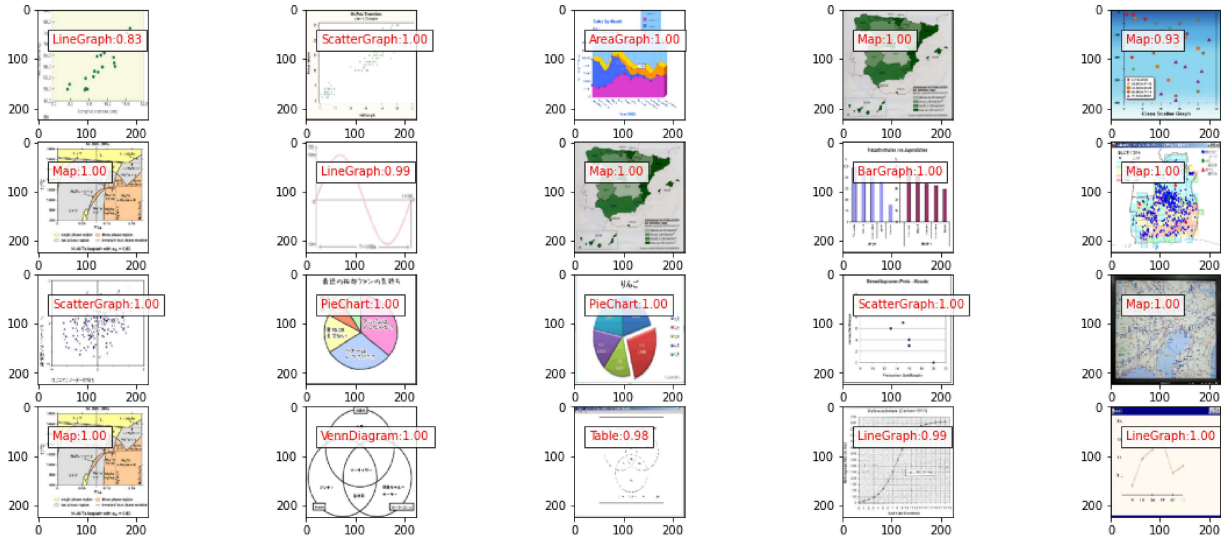
In [409]:
```
acc += history_fine.history['accuracy']
val_acc += history_fine.history['val_accuracy']

loss += history_fine.history['loss']
val_loss += history_fine.history['val_loss']
```

In [410]:
```
plt.figure(figsize=(8, 8))
plt.subplot(2, 1, 1)
plt.plot(acc, label='Training Accuracy')
plt.plot(val_acc, label='Validation Accuracy')
plt.ylim([0.8, 1])
plt.plot([initial_epochs-1,initial_epochs-1],
          plt.ylim(), label='Start Fine Tuning')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(2, 1, 2)
plt.plot(loss, label='Training Loss')
plt.plot(val_loss, label='Validation Loss')
plt.ylim([0, 1.0])
plt.plot([initial_epochs-1,initial_epochs-1],
         plt.ylim(), label='Start Fine Tuning')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.xlabel('epoch')
plt.show()
```



In [40]:
```
# for generator image set u can use
# ypred = classifier.predict_generator(test_set)

import numpy as np
from keras.preprocessing import image
fig=plt.figure(figsize=(20, 8))
columns = 5
rows = 4
for i in range(columns*rows):
    fig.add_subplot(rows, columns, i+1)
    img1 = image.load_img('C:\\Users\\rpshc\\Desktop\\Assingment\\Base_Folder\\te
    img = image.img_to_array(img1)
    img = img/255
    img = np.expand_dims(img, axis=0)
    result = model_2.predict(img, batch_size=None,steps=1) #gives all class prob.
    if result[0][0]>0.5:
        value = 'AreaGraph:%1.2f'%result[0][0]
        plt.text(30, 68,value,color='red',fontsize=10,bbox=dict(facecolor='white'
    elif result[0][1]>0.5:
        value = 'BarGraph:%1.2f'%result[0][1]
        plt.text(30, 68,value,color='red',fontsize=10,bbox=dict(facecolor='white'
    elif result[0][2]>0.5:
        value ='LineGraph:%1.2f'%result[0][2]
        plt.text(30, 68,value,color='red',fontsize=10,bbox=dict(facecolor='white'
    elif result[0][3]>0.5:
        value ='Map:%1.2f'%result[0][3]
        plt.text(30, 68,value,color='red',fontsize=10,bbox=dict(facecolor='white'
    elif result[0][4]>0.5:
        value ='ParetoChart:%1.2f'%result[0][4]
        plt.text(30, 68,value,color='red',fontsize=10,bbox=dict(facecolor='white'

    elif result[0][5]>0.5:
        value ='PieChart:%1.2f'%result[0][5]
        plt.text(30, 68,value,color='red',fontsize=10,bbox=dict(facecolor='white'

    elif result[0][6]>0.5:
        value ='RadarPlot:%1.2f'%result[0][6]
        plt.text(30, 68,value,color='red',fontsize=10,bbox=dict(facecolor='white'

    elif result[0][7]>0.5:
        value ='ScatterGraph:%1.2f'%result[0][7]
        plt.text(30, 68,value,color='red',fontsize=10,bbox=dict(facecolor='white'

    elif result[0][8]>0.5:
        value ='Table:%1.2f'%result[0][8]
        plt.text(30, 68,value,color='red',fontsize=10,bbox=dict(facecolor='white'

    elif result[0][9]>0.5:
        value ='VennDiagram:%1.2f'%result[0][9]
        plt.text(30, 68,value,color='red',fontsize=10,bbox=dict(facecolor='white'

    plt.imshow(img1)

print(result)
```

```
[[3.01893760e-10 6.43620115e-06 9.99836087e-01 1.14771666e-17
  1.51766974e-10 1.43810113e-22 2.83070622e-10 1.57505085e-04
  2.39974492e-15 2.10784294e-18]]
```



In [41]:
```
loss, accuracy = model_2.evaluate(test_set)
print('Test accuracy :', accuracy)
```

```
3/3 [==============================] - 1s 376ms/step - loss: 2.0510 - acc: 0.82
22
Test accuracy : 0.8222222328186035
```

In [42]:
```
loss, accuracy = model_2.evaluate(val_set)
print('Test accuracy :', accuracy)
```

```
6/6 [==============================] - 3s 564ms/step - loss: 1.1778 - acc: 0.90
00
Test accuracy : 0.8999999761581421
```

In [ ]:
```

```