Data Visualization

In [ ]:
```
fig,ax=plt.subplots(5,2)
fig.set_size_inches(15,15)
for i in range(5):
    for j in range (2):
        l=rn.randint(0,len(Z))
        ax[i,j].imshow(X[l])
        ax[i,j].set_title('Dog: '+Z[l])

plt.tight_layout()
```

Model Building

In [ ]:
```
base_model = InceptionV3(include_top=False,
                  input_shape = (IMG_SIZE,IMG_SIZE,3),
                  weights = 'imagenet')

# Freezing Layers
for layer in base_model.layers:
    layer.trainable = False

model = Sequential()
model.add(base_model)
model.add(GlobalAveragePooling2D())
model.add(Dense(512,activation='relu'))
model.add(Dense(512,activation='relu'))
model.add(Dense(7,activation='softmax'))
model.summary()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/inception_v3/incept
ion_v3_weights_tf_dim_ordering_tf_kernels_notop.h5
87916544/87910968 [==============================] - 1s 0us/step
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
inception_v3 (Functional)    (None, 3, 3, 2048)        21802784
_________________________________________________________________
global_average_pooling2d (Gl (None, 2048)              0
```


```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/inception_v3/incept
ion_v3_weights_tf_dim_ordering_tf_kernels_notop.h5
87916544/87910968 [==============================] - 1s 0us/step
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
inception_v3 (Functional)    (None, 3, 3, 2048)        21802784
_________________________________________________________________
global_average_pooling2d (Gl (None, 2048)              0
_________________________________________________________________
dense (Dense)                (None, 512)               1049088
_________________________________________________________________
dense_1 (Dense)              (None, 512)               262656
_________________________________________________________________
dense_2 (Dense)              (None, 7)                 3591
=================================================================
Total params: 23,118,119
Trainable params: 1,315,335
Non-trainable params: 21,802,784
_________________________________________________________________
```

Compiling and Training model

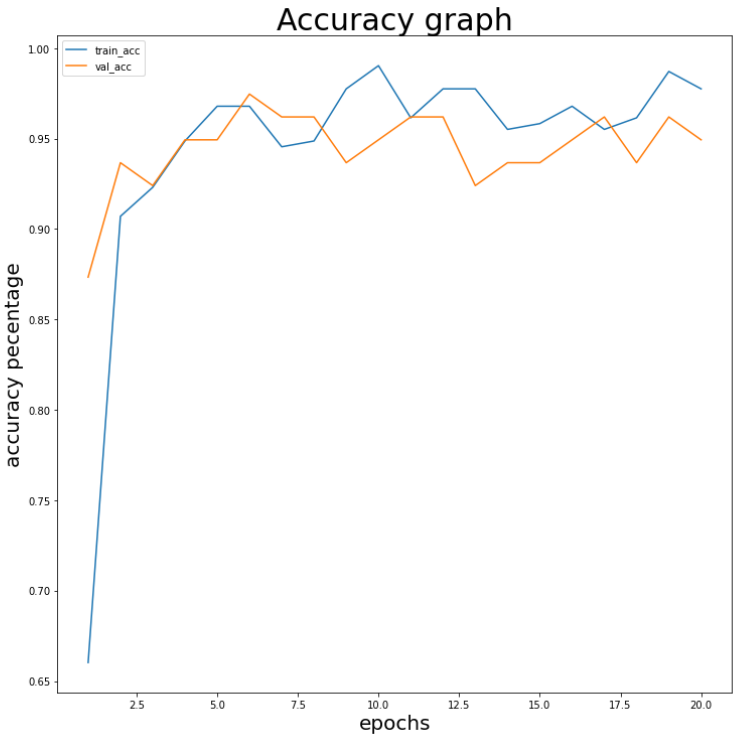
In [ ]:
```
#----------Compile---------------#
model.compile(
    loss='categorical_crossentropy',
    optimizer='adam',
    metrics=['accuracy']
)
```

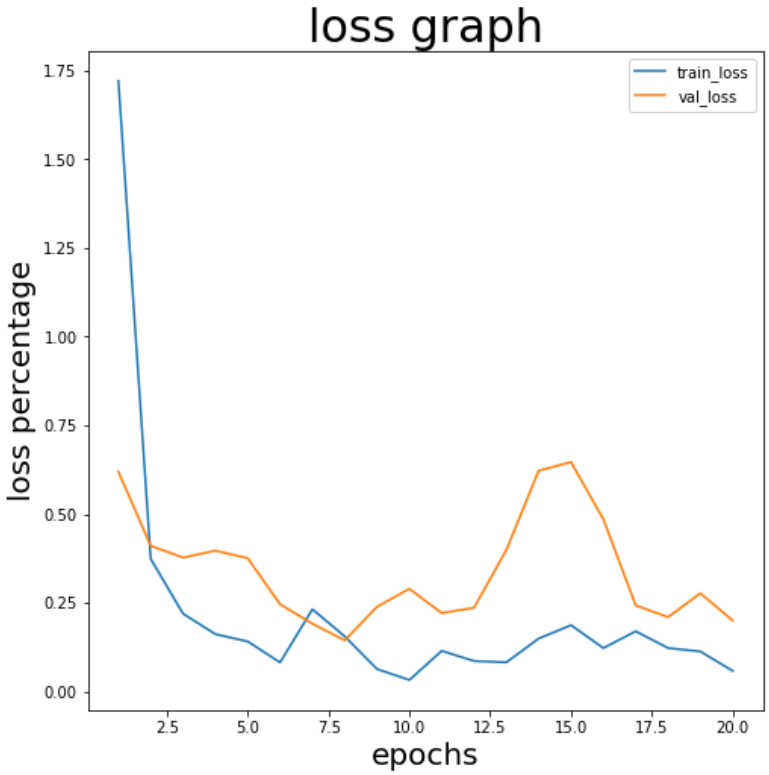
In [ ]:
```
#-----------Training------------#
history = model.fit_generator(
    augs_gen.flow(x_train,y_train,batch_size=16),
    validation_data  = (x_test,y_test),

    epochs = 20,
    verbose = 1,
)
```

```
WARNING:tensorflow:From <ipython-input-20-629181e044a8>:8: Model.fit_generator (from tensorflow.pytho
n.keras.engine.training) is deprecated and will be removed in a future version.
Instructions for updating:
Please use Model.fit, which supports generators.
Epoch 1/20
20/20 [==============================] - 3s 172ms/step - loss: 1.7202 - accuracy: 0.6603 - val_loss:
0.6194 - val_accuracy: 0.8734
Epoch 2/20
20/20 [==============================] - 2s 85ms/step - loss: 0.3737 - accuracy: 0.9071 - val_loss: 0.
```

```
Epoch 14/20
20/20 [==============================] - 2s 84ms/step - loss: 0.1495 - accuracy: 0.9551 - val_loss: 0.
6224 - val_accuracy: 0.9367
Epoch 15/20
20/20 [==============================] - 2s 83ms/step - loss: 0.1871 - accuracy: 0.9583 - val_loss: 0.
6465 - val_accuracy: 0.9367
Epoch 16/20
20/20 [==============================] - 2s 83ms/step - loss: 0.1225 - accuracy: 0.9679 - val_loss: 0.
4855 - val_accuracy: 0.9494
Epoch 17/20
20/20 [==============================] - 2s 83ms/step - loss: 0.1700 - accuracy: 0.9551 - val_loss: 0.
2430 - val_accuracy: 0.9620
Epoch 18/20
20/20 [==============================] - 2s 84ms/step - loss: 0.1227 - accuracy: 0.9615 - val_loss: 0.
2097 - val_accuracy: 0.9367
Epoch 19/20
20/20 [==============================] - 2s 83ms/step - loss: 0.1131 - accuracy: 0.9872 - val_loss: 0.
2769 - val_accuracy: 0.9620
Epoch 20/20
20/20 [==============================] - 2s 83ms/step - loss: 0.0584 - accuracy: 0.9776 - val_loss: 0.
2004 - val_accuracy: 0.9494
```

By training the model, the model give 92% validation accuracy. Here the training accuracy and validation accuracy is not having huge variation. So, the model does not go for overfitting. Hence, model is good fit.


Accuracy graph
train_acc
val_acc
accuracy pecentage
epochs
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0


loss graph
train_loss
val_loss
loss percentage
epochs
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0

Prediction Probabilities

In [ ]:
```
model.predict(new_image)
```

Out[ ]:
```
array([[5.2547269e-19, 2.9561605e-18, 2.4333582e-12, 5.5266704e-18,
        1.0000000e+00, 1.0304492e-15, 7.0007334e-14]], dtype=float32)
```

In [ ]:
```
array_num = model.predict(new_image)
num_list = array_num.tolist()
x=num_list[0]
item = max(x)
#search for the item
index = x.index(item)
breed_classes=list(zip(breed,breed_name))
q=0
while q<7:
  if index in breed_classes[q]:
    print('Dog Breed: '+ breed_classes[q][1])
    break
  else:
    q=q+1
img = mpimg.imread(img_url)
imgplot = plt.imshow(img)
plt.show()
```

```
Dog Breed: Maltese
```