IMAGE CLASSIFICATION

Business Problem:

Description:

Ship or vessel detection has a wide range of applications, in the areas of maritime safety, fisheries management, marine pollution, defence and maritime security, protection from piracy, illegal migration, etc.

Keeping this in mind, a Governmental Maritime and Coastguard Agency is planning to deploy a computer vision based automated system to identify ship type only from the images taken by the survey boats. You have been hired as a consultant to build an efficient model for this project.

Dataset Description

There are 6252 images in train and 2680 images in test data. The categories of ships and their corresponding codes in the dataset are as follows -

There are 5 classes of ships to be detected which are as follows:

Cargo---------->>> Class label 1

Military---------->>> Class label 2

Carrier---------->>> Class label 3

Cruise---------->>> Class label 4

Tankers---------->>> Class label 5

This dataset was in the Analytics Vidhya contest i.e GAME OF DEEP LEARNING https://datahack.analyticsvidhya.com/contest/game-of-deep-learning/ (https://datahack.analyticsvidhya.com/contest/game-of-deep-learning/) which was computer vision hackathon from 25th/may/19 to 10th/June/19.

Problem Statemtent:

To predict the class label y i.e predect catagory of image(Class label) is Cargo,Military,Carrier,Cruise or Tankers .

Evaluation Metric:

Weighted F1 score:---------->>> The F1 Scores are calculated for each label and then their average is weighted by support - which is the number of true instances for each label. It can result in an F-score that is not between precision and recall.

https://en.wikipedia.org/wiki/F1_score (https://en.wikipedia.org/wiki/F1_score) https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html (https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html)

Real World / Business Objectives and Constraints:

1---->> Predect the class label of images with high Weighted F1 score.

2---->> No strict latency constraints.

HERE WE HAVE MADE THIS CASE CASE STUDY ON ONLY 6252 images which is further spilit into train and test.

Importing libraries:

In [57]:

```
import keras
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras.utils import to_categorical
from keras.preprocessing import image
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from tqdm import tqdm
%matplotlib inline
from keras.models import Sequential
from scipy.misc import imread
get_ipython().magic('matplotlib inline')
import matplotlib.pyplot as plt
import numpy as np
import keras
from keras.layers import Dense
import pandas as pd

#from keras.applications.vgg16 import VGG16
from keras.preprocessing import image
from keras.applications.vgg16 import preprocess_input
import numpy as np
#from keras.applications.vgg16 import decode_predictions
from keras.utils.np_utils import to_categorical

from sklearn.preprocessing import LabelEncoder
from keras.models import Sequential
from keras.optimizers import SGD
from keras.layers import Input, Dense, Convolution2D, MaxPooling2D, AveragePooling2D, ZeroP
from keras.layers.normalization import BatchNormalization
from sklearn.metrics import log_loss
import matplotlib.pyplot as plt
import seaborn as sns

# Python program to read
# image using matplotlib

# importing matplotlib modules
import matplotlib.image as mpimg
import matplotlib.pyplot as plt
```

In [5]:

```
# this will do preprocessing and realtime data augmentation
'''from keras.preprocessing.image import ImageDataGenerator
from keras.layers.normalization import BatchNormalization
datagen = ImageDataGenerator(
    featurewise_center=False,  # set input mean to 0 over the dataset
    samplewise_center=False,  # set each sample mean to 0
    featurewise_std_normalization=False,  # divide inputs by std of the dataset
    samplewise_std_normalization=False,  # divide each input by its std
    zca_whitening=False,  # apply ZCA whitening
    rotation_range=25,  # randomly rotate images in the range (degrees, 0 to 180)
    width_shift_range=0.1,  # randomly shift images horizontally (fraction of total width)
    height_shift_range=0.1,  # randomly shift images vertically (fraction of total height)
    horizontal_flip=True,  # randomly flip images
    vertical_flip=False)  # randomly flip images

datagen.fit(train_img)'''
```

SOME EDA ON DATA:

In [4]:

```
X_train[2].shape
```

Out[4]:

```
(139, 210, 3)
```

In [49]:

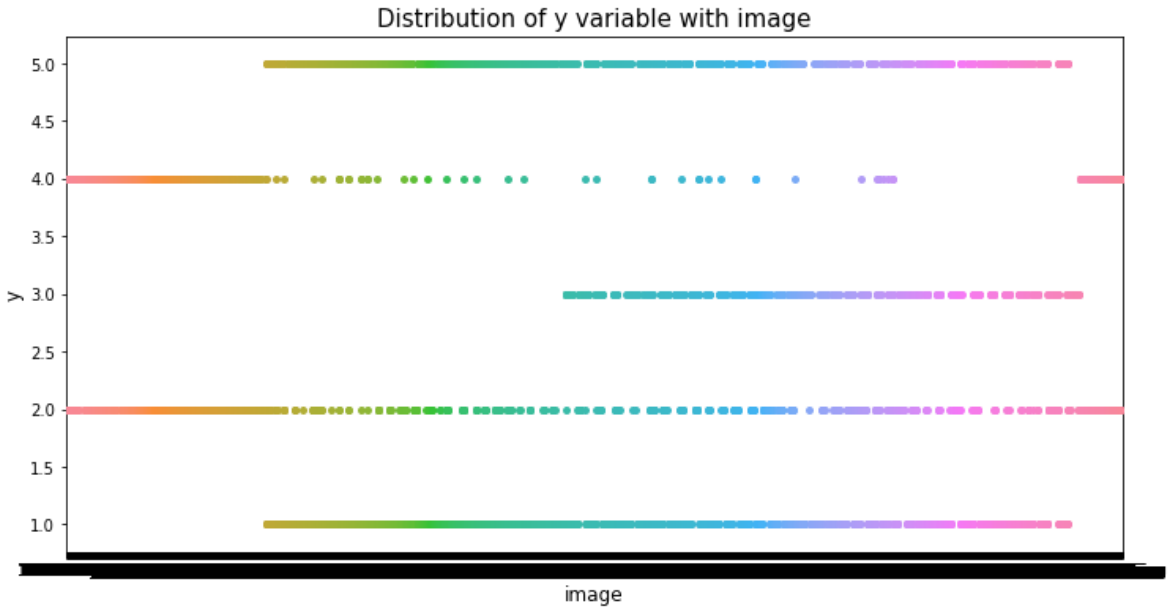
```
train.head(10)
```

Out[49]:

	image	category
0	2823080.jpg	1
1	2870024.jpg	1
2	2662125.jpg	2
3	2900420.jpg	3
4	2804883.jpg	2
5	621252.jpg	4
6	2833467.jpg	1
7	2843780.jpg	5
8	2859567.jpg	3
9	2896557.jpg	1

In [59]:

```
var_name = "image"
col_order = np.sort(train[var_name].unique()).tolist()
plt.figure(figsize=(12,6))
sns.stripplot(x=var_name, y='category', data=train, order=col_order)
plt.xlabel(var_name, fontsize=12)
plt.ylabel('y', fontsize=12)
plt.title("Distribution of y variable with "+var_name, fontsize=15)
plt.show()
```

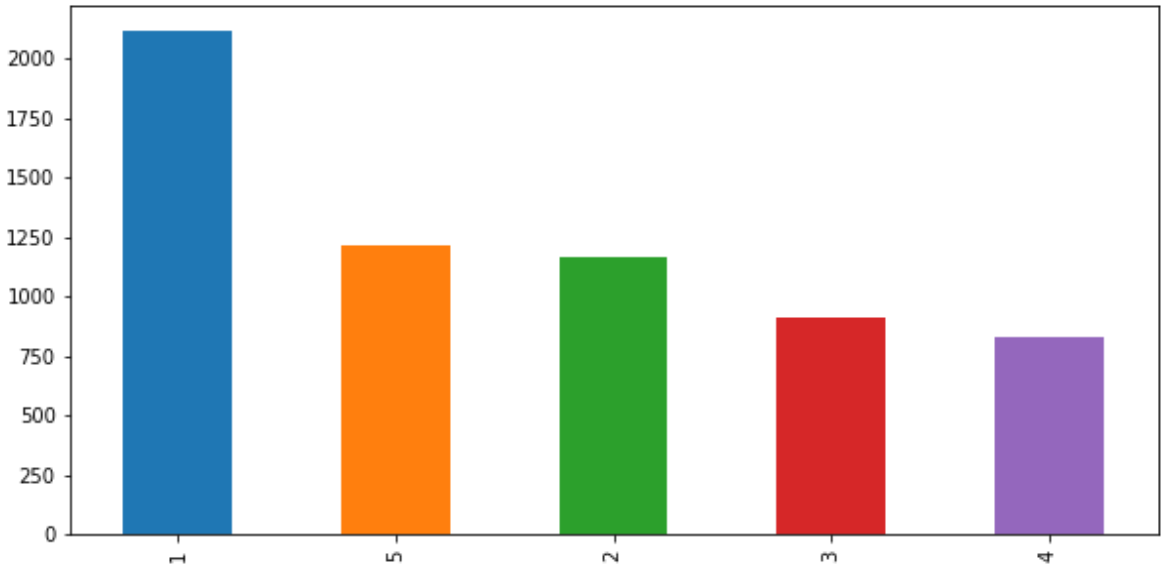


In [23]:

```
fig, ax1 = plt.subplots(1, 1, figsize= (10, 5))
train['category'].value_counts().plot(kind='bar', ax=ax1)
```

Out[23]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x297dd22e400>
```



OBSERVATION: HERE WE CAN OBSERVE THAT IMAGES WITH CLASS 1 IS MAJORITY I.E HAVING MORE NO. OF POINTS AS COMPAIR TO OTHER CLASSES, REST ALL OTHER CLASS HAVING BALANCED NO OF LABELS

Cargo Class Label 1

In [50]:

```
image.load_img(train_path+train['image'][1])
```

Out[50]:

Military Class Label 2

In [51]:

```
image.load_img(train_path+train['image'][2])
```

Out[51]:

Carrier Class Label 3

In [52]:

```
image.load_img(train_path+train['image'][3])
```

Out[52]:

Cruise Class Label 4

In [53]:

```
image.load_img(train_path+train['image'][5])
```

Out[53]:

Tankers Class Label 5

In [54]:

```
image.load_img(train_path+train['image'][7])
```

Out[54]:

DATA PREPROCESSING i.e CONVERTING IMAGES INTO ARRAYES:

In [2]:

```
train=pd.read_csv("train.csv")
test=pd.read_csv("test_ApKoW4T.csv")
train_path="images/"
test_path="images/"

from scipy.misc import imresize

train_img=[]
for i in range(len(train)):

    temp_img=image.load_img(train_path+train['image'][i],target_size=(139, 210))

    temp_img=image.img_to_array(temp_img)

    train_img.append(temp_img)

#converting train images to array and applying mean subtraction processing

train_img=np.array(train_img)
train_img=preprocess_input(train_img)
# applying the same procedure with the test dataset

test_img=[]
for i in range(len(test)):

    temp_img=image.load_img(test_path+test['image'][i],target_size=(139, 210))

    temp_img=image.img_to_array(temp_img)

    test_img.append(temp_img)

test_img=np.array(test_img)
test_img=preprocess_input(test_img)



train_y=np.asarray(train['category'])
# performing one-hot encoding for the target variable

train_y=pd.get_dummies(train_y)
train_y=np.array(train_y)
# creating training and validation set
```

Splitting data into Train(85%) and Test(15%):

In [62]:

```
from sklearn.model_selection import train_test_split
X_train, X_valid, Y_train, Y_valid=train_test_split(train_img,train_y,test_size=0.15, randc
```

Defining the architecture of C.N.N model:

In [60]:

```
model = Sequential()


model.add(Convolution2D(64, 2,2, input_shape = (139, 210, 3), border_mode='same'))
model.add(Conv2D(64, (2,2), activation='relu'))

model.add(MaxPooling2D(pool_size=(3,3)))
model.add(Dropout(0.25))

model.add(Conv2D(64, (3,3), padding='same', activation='relu'))
model.add(Conv2D(64, (3,3), activation='relu'))


model.add(BatchNormalization())


model.add(MaxPooling2D(pool_size=(3,3)))
model.add(Dropout(0.25))

model.add(Conv2D(64, (3,3), padding='same', activation='relu'))
model.add(Conv2D(64, (3,3), activation='relu'))

model.add(BatchNormalization())

model.add(MaxPooling2D(pool_size=(3,3)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(5, activation='softmax'))
model.summary()
```

```
C:\Users\hp\Anaconda3\lib\site-packages\ipykernel_launcher.py:4: UserWarnin
g: Update your `Conv2D` call to the Keras 2 API: `Conv2D(64, (2, 2), input_s
hape=(139, 210,..., padding="same")`
  after removing the cwd from sys.path.
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 139, 210, 64)      832
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 138, 209, 64)      16448
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 46, 69, 64)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 46, 69, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 46, 69, 64)        36928
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 44, 67, 64)        36928
_________________________________________________________________
batch_normalization_1 (Batch (None, 44, 67, 64)        256
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 14, 22, 64)        0
_________________________________________________________________
dropout_2 (Dropout)          (None, 14, 22, 64)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 14, 22, 64)        36928
```

```
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 12, 20, 64)        36928
_________________________________________________________________
batch_normalization_2 (Batch (None, 12, 20, 64)        256
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 4, 6, 64)          0
_________________________________________________________________
dropout_3 (Dropout)          (None, 4, 6, 64)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 1536)              0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               196736
_________________________________________________________________
dropout_4 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 5)                 645
=================================================================
Total params: 362,885
Trainable params: 362,629
Non-trainable params: 256
_________________________________________________________________
```

Here we are training model with above architecture for 50 and 100 epocs:

In [73]:

```
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
history_1=model.fit(X_train, Y_train, batch_size=16, nb_epoch=100,verbose=1,validation_data
# Make predictions
```

```
ETA: 29s - loss: 0.0464 - acc: 0.98 - ETA: 28s - loss: 0.0463 - acc: 0.98
- ETA: 27s - loss: 0.0462 - acc: 0.98 - ETA: 26s - loss: 0.0466 - acc: 0.9
8 - ETA: 25s - loss: 0.0464 - acc: 0.98 - ETA: 24s - loss: 0.0463 - acc:
0.98 - ETA: 23s - loss: 0.0462 - acc: 0.98 - ETA: 23s - loss: 0.0462 - ac
c: 0.98 - ETA: 22s - loss: 0.0465 - acc: 0.98 - ETA: 21s - loss: 0.0464 -
acc: 0.98 - ETA: 20s - loss: 0.0463 - acc: 0.98 - ETA: 19s - loss: 0.0461
- acc: 0.98 - ETA: 18s - loss: 0.0460 - acc: 0.98 - ETA: 17s - loss: 0.045
9 - acc: 0.98 - ETA: 16s - loss: 0.0458 - acc: 0.98 - ETA: 15s - loss: 0.0
456 - acc: 0.98 - ETA: 14s - loss: 0.0455 - acc: 0.98 - ETA: 13s - loss:
0.0455 - acc: 0.98 - ETA: 12s - loss: 0.0454 - acc: 0.98 - ETA: 12s - los
s: 0.0453 - acc: 0.98 - ETA: 11s - loss: 0.0465 - acc: 0.98 - ETA: 10s - l
oss: 0.0464 - acc: 0.98 - ETA: 9s - loss: 0.0464 - acc: 0.9854 - ETA: 8s -
loss: 0.0469 - acc: 0.985 - ETA: 7s - loss: 0.0468 - acc: 0.985 - ETA: 6s
- loss: 0.0466 - acc: 0.985 - ETA: 5s - loss: 0.0469 - acc: 0.984 - ETA: 4
s - loss: 0.0470 - acc: 0.984 - ETA: 3s - loss: 0.0468 - acc: 0.984 - ETA:
2s - loss: 0.0467 - acc: 0.985 - ETA: 1s - loss: 0.0466 - acc: 0.985 - ET
A: 1s - loss: 0.0465 - acc: 0.985 - ETA: 0s - loss: 0.0465 - acc: 0.985 -
324s 61ms/step - loss: 0.0465 - acc: 0.9851 - val_loss: 0.9731 - val_acc:
0.8561
```

In [67]:

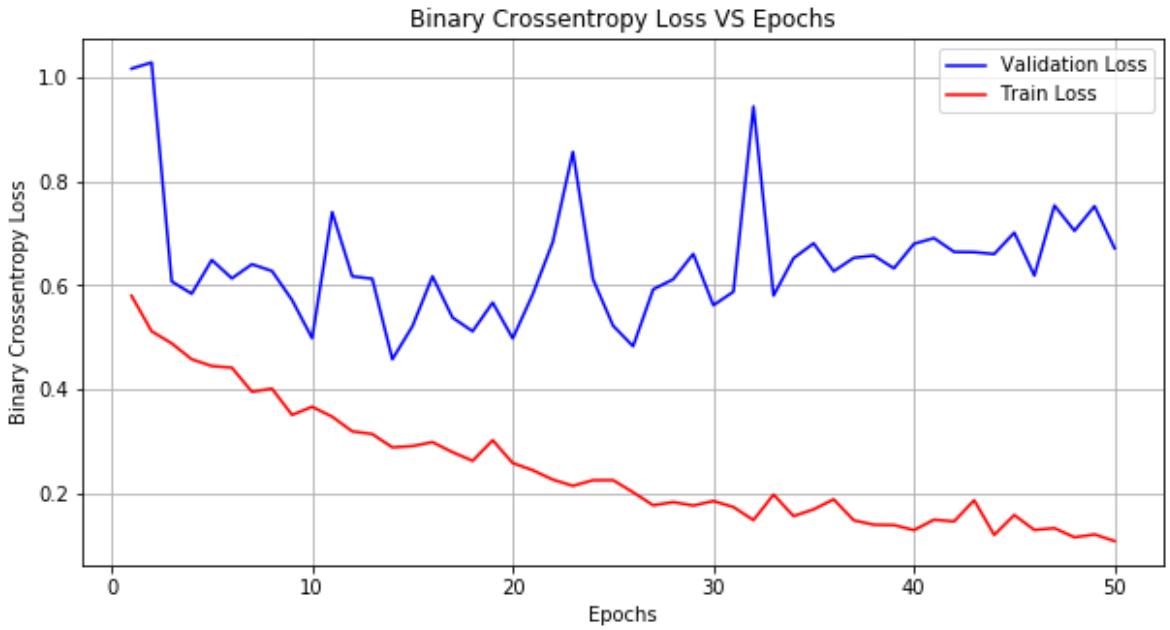
```
def plt_dynamic(x, vy, ty):
  plt.figure(figsize=(10,5))
  plt.plot(x, vy, 'b', label="Validation Loss")
  plt.plot(x, ty, 'r', label="Train Loss")
  plt.xlabel('Epochs')
  plt.ylabel('Binary Crossentropy Loss')
  plt.title('\nBinary Crossentropy Loss VS Epochs')
  plt.legend()
  plt.grid()
  plt.show()
```

Evaluating and Saving model for 50 epocs:

In [68]:

```
# Final evaluation of the model
scores = model.evaluate(X_valid, Y_valid, verbose=0)
print("Accuracy: %.2f%%" % (scores[1]*100))

# Test and train accuracy of the model
model_1_test = scores[1]
model_1_train = max(history_1.history['acc'])

# Plotting Train and Test Loss VS no. of epochs
# list of epoch numbers
x = list(range(1,51))

# Validation loss
vy = history_1.history['val_loss']
# Training loss
ty = history_1.history['loss']

# Calling the function to draw the plot
plt_dynamic(x, vy, ty)
```

```
Accuracy: 85.82%
```



Predicting the class with highest probability:

In [69]:

```
# make predictions on test
convnet3_test_preds = model.predict(x_train)

# predict as the class with highest probability
convnet3_test_preds = np.argmax(convnet3_test_preds, axis = 1)+1

# put predictions in pandas Series
convnet3_test_preds = pd.Series(convnet3_test_preds, name='category')

# Add 'ImageId' column
convnet3_for_csv = pd.concat([pd.Series(range(1,939), name='image'),
                              convnet3_test_preds], axis=1)

# write to csv for submission
convnet3_for_csv.to_csv('convnet3_keras.csv', index=False)
```

In [72]:

```
from keras.models import model_from_json
import numpy
import os
# serialize model to JSON
model_json = model.to_json()
with open("model.json", "w") as json_file:
    json_file.write(model_json)
# serialize weights to HDF5
model.save_weights("game_of_dl_1.h5")
print("Saved model to disk")
```

```
Saved model to disk
```

In [70]:

```
Answer = convnet3_test_preds

submission = pd.read_csv('sample_submission_ns2btKE.csv')
submission['category'] = Answer
submission.to_csv('sample_submission_aaic1.csv', index=False)
submission.head(5)
```

Out[70]:

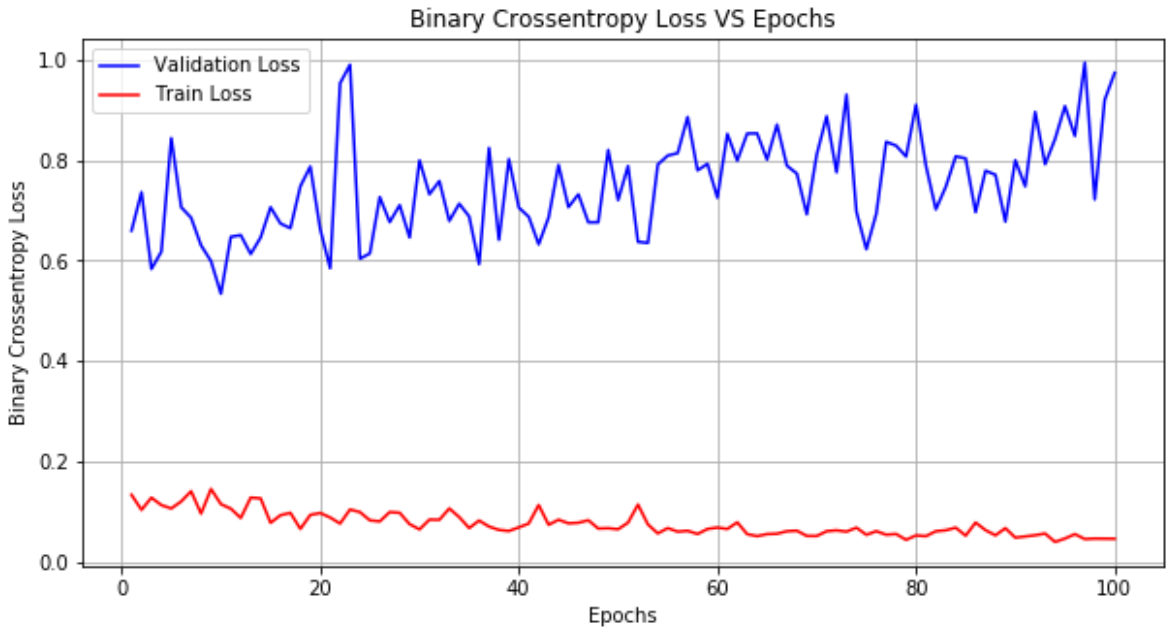
	image	category
0	1007700.jpg	4.0
1	1011369.jpg	5.0
2	1051155.jpg	5.0
3	1062001.jpg	3.0
4	1069397.jpg	4.0

Evaluating and Saving model for 100 epocs:

In [74]:

```
# Final evaluation of the model
scores = model.evaluate(X_valid, Y_valid, verbose=0)
print("Accuracy: %.2f%%" % (scores[1]*100))

# Test and train accuracy of the model
model_1_test = scores[1]
model_1_train = max(history_1.history['acc'])

# Plotting Train and Test Loss VS no. of epochs
# list of epoch numbers
x = list(range(1,101))

# Validation loss
vy = history_1.history['val_loss']
# Training loss
ty = history_1.history['loss']

# Calling the function to draw the plot
plt_dynamic(x, vy, ty)
```

```
Accuracy: 85.61%
```



In [75]:

```
model_json = model.to_json()
with open("model.json", "w") as json_file:
    json_file.write(model_json)
# serialize weights to HDF5
model.save_weights("game_of_dl_2.h5")
print("Saved model to disk")
```

```
Saved model to disk
```

Predicting the class with highest probability:

In [116]:

```
# make predictions on test
convnet3_test_preds = model.predict(X_valid)

# predict as the class with highest probability
convnet3_test_preds = np.argmax(convnet3_test_preds, axis = 1)

# put predictions in pandas Series
convnet3_test_preds = pd.Series(convnet3_test_preds, name='category')

# Add 'ImageId' column
convnet3_for_csv = pd.concat([pd.Series(range(1,939), name='image'),
                              convnet3_test_preds], axis=1)

# write to csv for submission
#convnet3_for_csv.to_csv('convnet3_keras.csv', index=False)
```

In [112]:

```
actual_category=np.argmax(Y_valid, axis = 1)
actual_category = pd.Series( actual_category, name='category')
```

Compairing predected_category actual_category of images

In [121]:

```
Answer = convnet3_test_preds

submission = pd.read_csv('sample_submission_ns2btKE.csv')
submission['predected_category'] = Answer+1
submission['actual_category'] =  actual_category+1
submission.to_csv('sample_submission_aaic_compair.csv', index=False)
```

In [122]:

```
df=pd.read_csv('sample_submission_aaic_compair.csv')
df1 = df.drop(['category'],axis=1)
df1 .head(20)
```

Out[122]:

	image	predected_category	actual_category
0	1007700.jpg	4.0	4.0
1	1011369.jpg	5.0	1.0
2	1051155.jpg	5.0	1.0
3	1062001.jpg	3.0	3.0
4	1069397.jpg	4.0	4.0
5	1072861.jpg	4.0	2.0
6	1097264.jpg	3.0	1.0
7	1098763.jpg	3.0	3.0
8	1098766.jpg	5.0	5.0
9	1101145.jpg	3.0	3.0
10	1114371.jpg	3.0	3.0
11	1114781.jpg	2.0	2.0
12	1115492.jpg	5.0	5.0
13	1121306.jpg	2.0	2.0
14	1123736.jpg	1.0	1.0
15	1124946.jpg	2.0	2.0
16	1125700.jpg	3.0	3.0
17	1127236.jpg	5.0	5.0
18	1127336.jpg	2.0	2.0
19	1130300.jpg	4.0	4.0

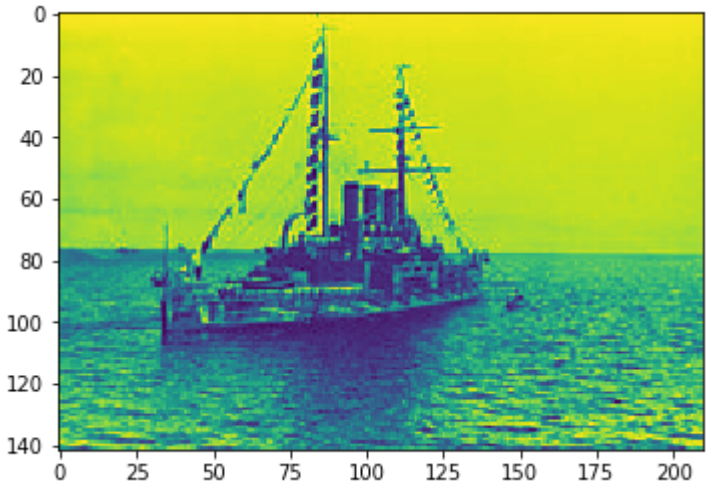
As we can see in above table 1123736.jpg is of class 1 i.e Cargo

In [129]:

```
img = mpimg.imread('images/1123736.jpg')
plt.imshow(img)
```

Out[129]:

```
<matplotlib.image.AxesImage at 0x297de909ba8>
```



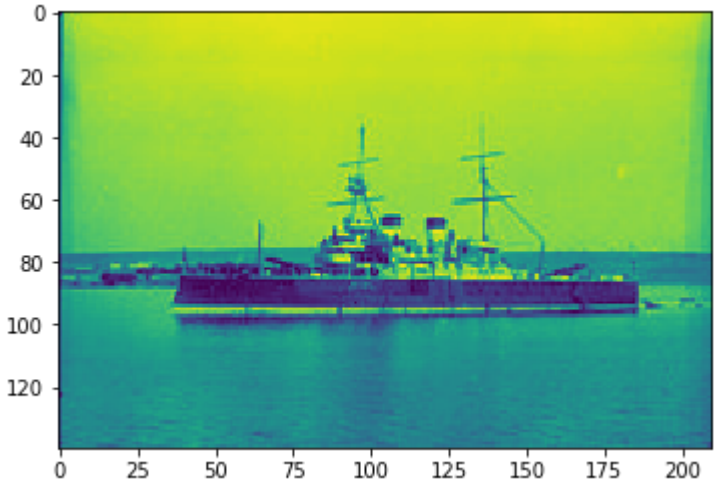
As we can see in above table 1114781.jpg is of class 2 i.e Military

In [135]:

```
img = mpimg.imread('images/1124946.jpg')
plt.imshow(img)
```

Out[135]:

```
<matplotlib.image.AxesImage at 0x297e09480b8>
```



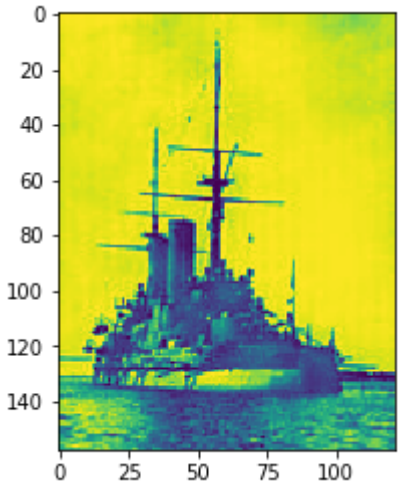
As we can see in above table 1062001.jpg is of class 3 i.e Carrier

In [133]:

```
img = mpimg.imread('images/1062001.jpg')
plt.imshow(img)
```

Out[133]:

```
<matplotlib.image.AxesImage at 0x297e060b1d0>
```



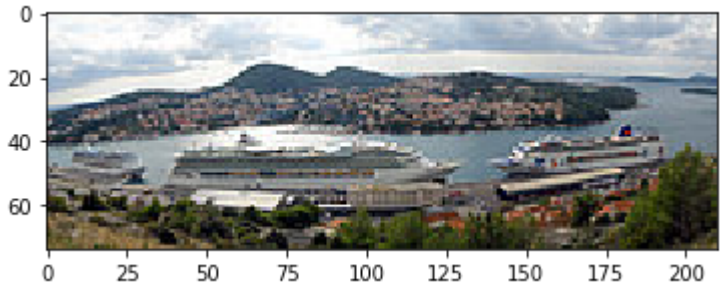
As we can see in above table 1007700.jpg is of class 4 i.e cruise

In [127]:

```
# Read Images
img = mpimg.imread('images/1007700.jpg')

# Output Images
plt.imshow(img)
```

Out[127]:

```
<matplotlib.image.AxesImage at 0x297de397c88>
```



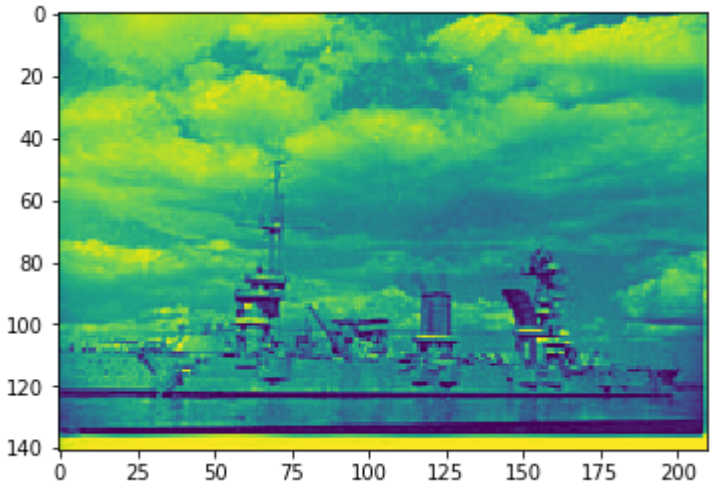
As we can see in above table 1098766.jpg is of class 5 i.e Tankers

In [128]:

```
img = mpimg.imread('images/1098766.jpg')
plt.imshow(img)
```

Out[128]:

```
<matplotlib.image.AxesImage at 0x297de8a92e8>
```



Observation:

---------->>>> Model with 50 epocs having 85.61 accuracy.

---------->>>> Model with 50 epocs having 85.82 accuracy.

NOTE: We can also use transfer learning or can implement more deep C.N.N like VGG 16 or our own deep layer C.N.N and increase the accuracy upto99%. We did not train deep C.C.C due to lack of resources.

My score in above contest is 90 with user_name: himanshu577 and rank is around 215 which is not finalised.

please visit below link to see the leaderboard position with user_name: himanshu577 rank 215 around

https://datahack.analyticsvidhya.com/contest/game-of-deep-learning/pvt_lb
(https://datahack.analyticsvidhya.com/contest/game-of-deep-learning/pvt_lb)

In [ ]: