**Deep Learning**

Course-End Project Problem Statement

**Course-End Project: Automating Port Operations**

**Project Statement:**

Marina Pier Inc. is leveraging technology to automate their operations on the San Francisco port.

The company’s management has set out to build a bias-free/ corruption-free automatic system that reports & avoids faulty situations caused by human error. Examples of human error include misclassifying the correct type of boat. The type of boat that enters the port region is as follows.

* Buoy
* Cruise\_ship
* Ferry\_boat
* Freight\_boar
* Gondola
* Inflatable\_boat
* Kayak
* Paper\_boat
* Sailboat

Marina Pier wants to use Deep Learning techniques to build an automatic reporting system that recognizes the boat. The company is also looking to use a transfer learning approach of any lightweight pre-trained model in order to deploy in mobile devices.

As a deep learning engineer, your task is to:

1. Build a CNN network to classify the boat.
2. Build a lightweight model with the aim of deploying the solution on a mobile device using transfer learning. You can use any lightweight pre-trained model as the initial (first) layer. MobileNetV2 is a popular lightweight pre-trained model built using Keras API.

**Dataset and Data Description:**

**boat\_type\_classification\_dataset.zip**

The dataset contains images of 9 types of boats. It contains a total of 1162 images. The training images are provided in the directory of the specific class itself.

Classes:

* ferry\_boat
* gondola
* sailboat
* cruise\_ship
* kayak
* inflatable\_boat
* paper\_boat
* buoy
* freight\_boat

**Perform the following steps:**

1. Build a CNN network to classify the boat.
   1. Split the dataset into train and test in the ratio 80:20, with shuffle and random state=43.
   2. Use Keras ImageDataGenerator to initialize the train generator with validation\_split=0.2 and test generator. Generators are required to avoid out of memory issues while training.
   3. Both generators will be initialized with data normalization.   
      *(Hint: rescale=1./255)*.
   4. Load train, validation and test dataset in batches of 32 using the generators initialized in the above step.
   5. Build a CNN network using Keras with the following layers
      * Cov2D with 32 filters, kernel size 3,3, and activation relu, followed by MaxPool2D
      * Cov2D with 32 filters, kernel size 3,3, and activation relu, followed by MaxPool2D
      * GLobalAveragePooling2D layer
      * Dense layer with 128 neurons and activation relu
      * Dense layer with 128 neurons and activation relu
      * Dense layer with 9 neurons and activation softmax.
   6. Compile the model with Adam optimizer, categorical\_crossentropy loss, and with metrics accuracy, precision, and recall.
   7. Train the model for 20 epochs and plot training loss and accuracy against epochs.
   8. Evaluate the model on test images and print the test loss and accuracy.
   9. Plot heatmap of the confusion matrix and print classification report.
2. Build a lightweight model with the aim of deploying the solution on a mobile device using transfer learning. You can use any lightweight pre-trained model as the initial (first) layer. MobileNetV2 is a popular lightweight pre-trained model built using Keras API.
   1. Split the dataset into train and test datasets in the ration 70:30, with shuffle and random state=1.
   2. Use Keras ImageDataGenerator to initialize the train generator with validation\_split=0.2 and test generator. Generators are required to avoid out-of-memory issues while training.
   3. Both generators will be initialized with data normalization.   
      *(Hint: rescale=1./255).*
   4. Load train, validation and test datasets in batches of 32 using the generators initialized in the above step.
   5. Build a CNN network using Keras with the following layers.
      * Load MobileNetV2 - Light Model as the first layer   
        *(Hint:* [*Keras API Doc*](https://keras.io/api/applications/mobilenet/)*)*
      * GLobalAveragePooling2D layer
      * Dropout(0.2)
      * Dense layer with 256 neurons and activation relu
      * BatchNormalization layer
      * Dropout(0.1)
      * Dense layer with 128 neurons and activation relu
      * BatchNormalization layer
      * Dropout(0.1)
      * Dense layer with 9 neurons and activation softmax
   6. Compile the model with Adam optimizer, categorical\_crossentropy loss, and metrics accuracy, Precision, and Recall.
   7. Train the model for 50 epochs and Early stopping while monitoring validation loss.
   8. Evaluate the model on test images and print the test loss and accuracy.
   9. Plot Train loss Vs Validation loss and Train accuracy Vs Validation accuracy.
3. Compare the results of both models built in steps 1 and 2 and state your observations.