Submitted by -:

Kushagra Pandey (22070127033)

Vivek Jadhav (22070127073)

Varad Desai (22070127074)

Submitted to-:

Dr. Sameer Sayad

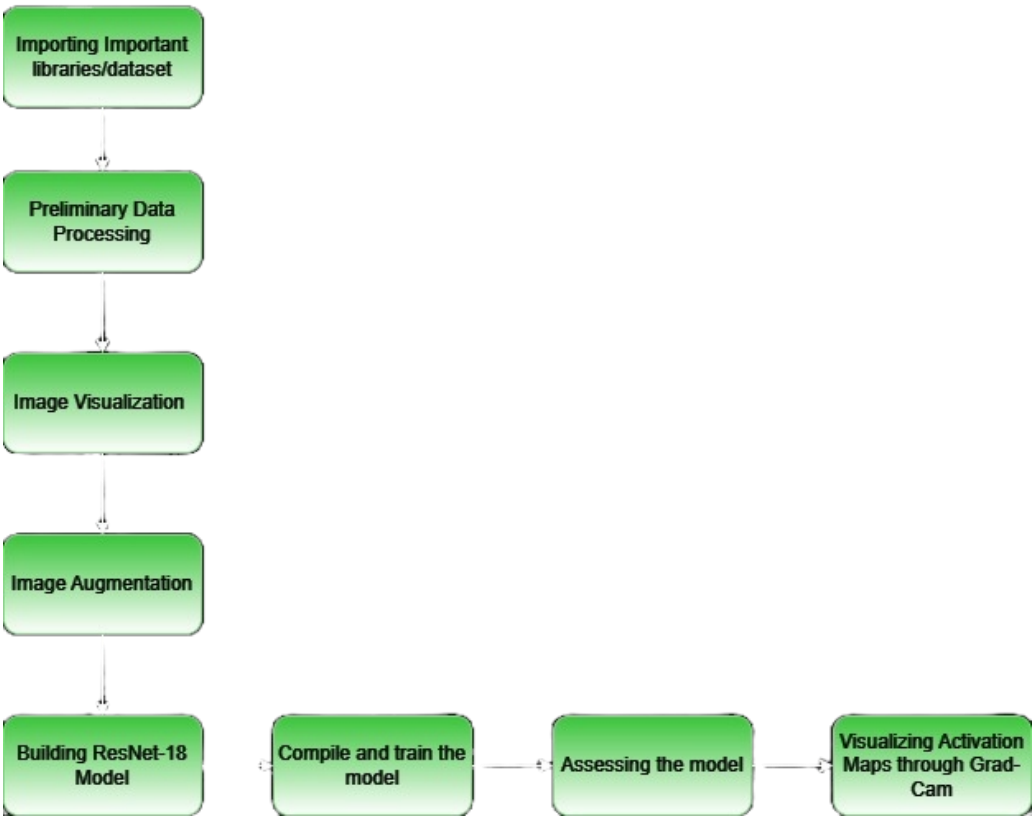
Robotics and Automation

SIT Pune

Explainable AI: Scene Classification and Grad Cam Visualization

- ## Methodology

  - ### Flowchart



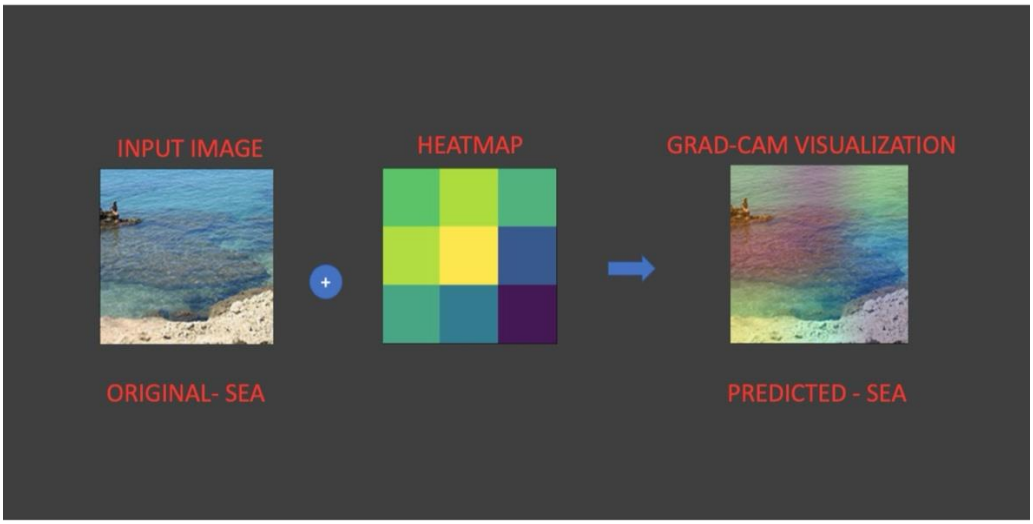
  - ### Explanation

In this project, we will train deep Convolutional Neural Networks(CNNs) and Residual Blocks to detect the type of scenery in an image. This project could be practically used for detecting the type of scenery from satellite image.

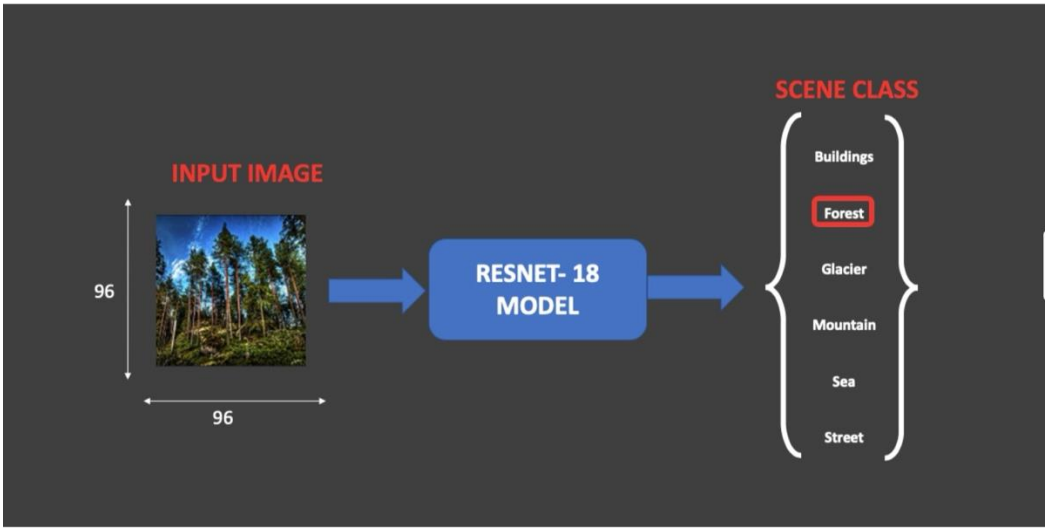
MICROSOFT AI FOR EARTH, has created the most detailed United States forest map using satellite imagery and AI which would essentially be a game changer in reducing deforestation, pests and wildfires.

https://www.microsoft.com/en-us/ai/ai-for-earth

Explainable AI: Gradient-Weighted Class Activation Map (Grad-CAM) helps visualize the regions of the input that contributed towards making prediction by the model.



The dataset consists of **17034 images** belonging to 6 categories. Categories that are present in the data are **buildings, forest, glacier, mountain, sea and street**.



Step 1 : Importing Important libraries and dataset

In this step, we will start by importing important libraries and dependencies which are required for this project which includes *pandas, NumPy, matplotlib, seaborn, TensorFlow, OS, PIL, ImageDataGenerator, Keras, ResNet50, InceptionResNetV2, SGD optimizer, LearningRateScheduler etc.*

After Importing our dependencies, we will check the number of images in our training and testing dataset using the **os.listdir** function.

Step 2: Preliminary Data Processing

In this step we will be exploring our dataset through different ways like plotting graphs, charts etc. We will also check the number of images in each class in the training and testing dataset.

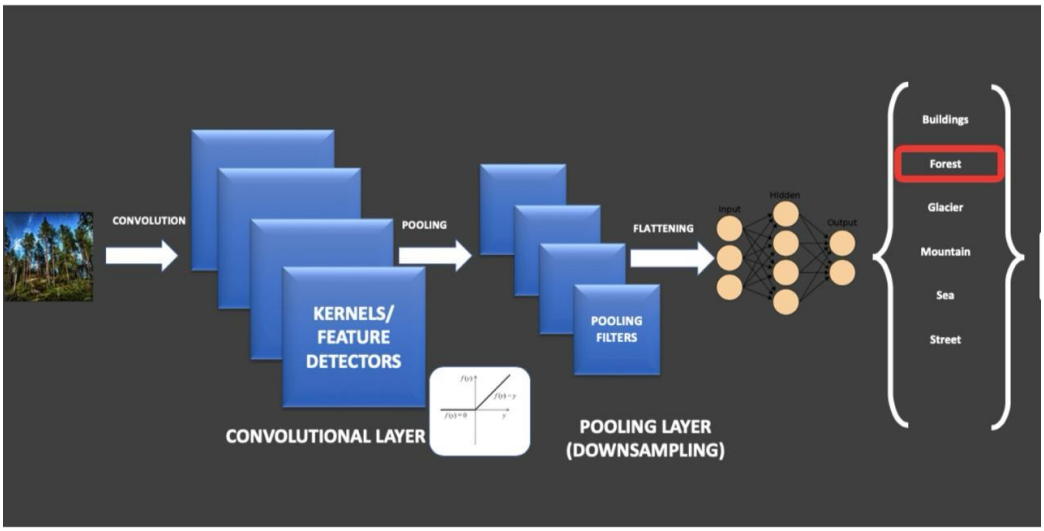
Step 3: Image Visualization

We will be visualizing the images in the dataset using the subplot function, we will get the list of images in the particular class and plot 5-10 images per class for both the testing and training dataset.

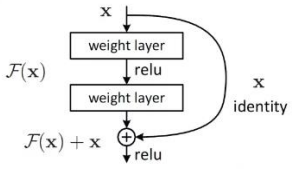
Step 4: Image Augmentation

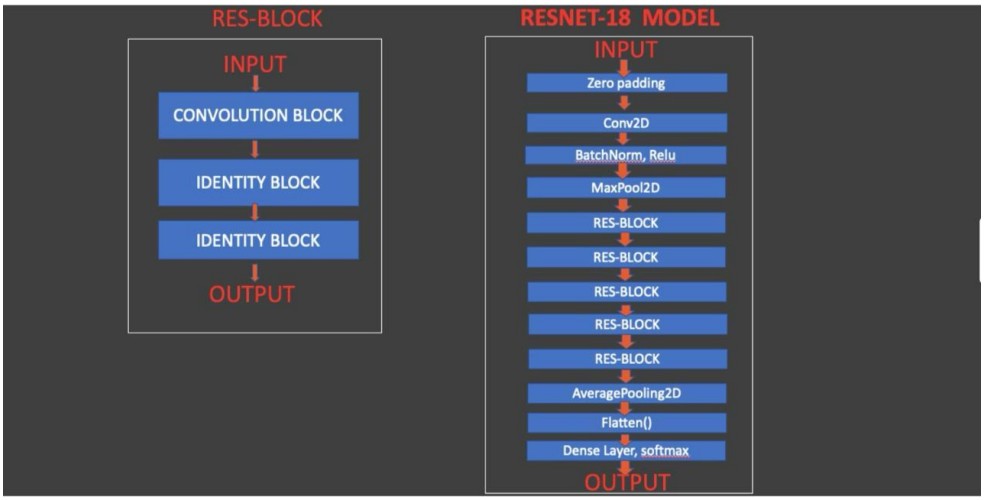
In this step we will perform data augmentation and create data generators. We will start by creating a run-time augmentation on training and test dataset.
For training datagenerator, we will add normalization, shear angle, zooming range and horizontal flip.
For testing datagenerator, we will only normalize the data.

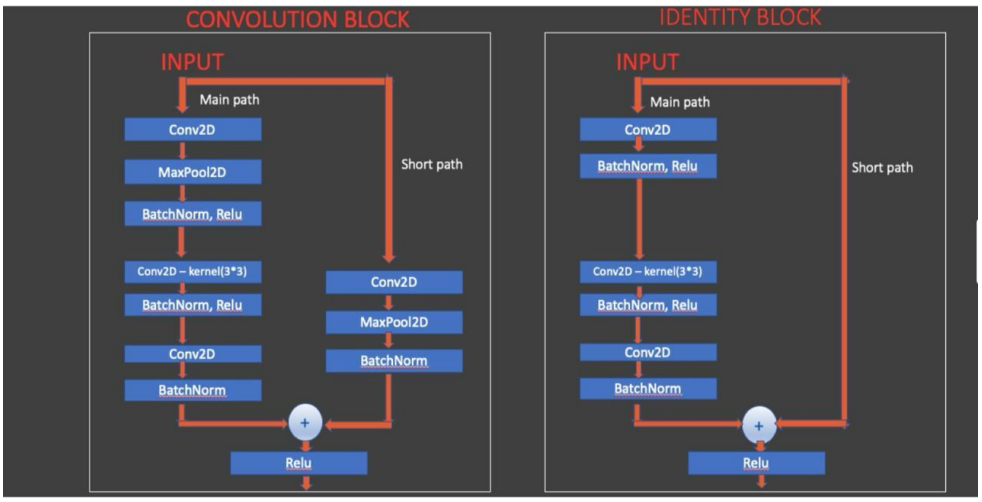
Step 5: Building the ResNet-18 Model



As CNNs grow deeper, vanishing gradient tend to occur which negatively impact network performance.
Vanishing gradient problem occurs when the gradient is back propagated to earlier layers which results in a very small gradient.
Residual Neural Network includes “skip connection” feature which enables training of 152 layers without vanishing gradient issues.
ResNet works by adding “identity mappings” on top of the CNN.







Step 6: Compile and train the model

We will be using early stopping to exit training if validation loss is not decreasing even after certain epochs. Then we will save the best model with lower validation loss.

Step 7: Assessing the model prediction

We will evaluate the performance of the model by using the model.evaluate() function and we will generate test accuracy, and print out the classification report and confusion matrix and comment on the model performance on the basis of that.

Step 8: Visualizing Activation Maps through Grad-CAM

Gradient-Weighted Class Activation Mapping (Grad-CAM) helps visualize the regions of the input that contributed towards making prediction by the model.

It does so by using the class specific gradient information flowing into the final convolutional layer of CNN to localize the important regions in the image that resulted in predicting that particular class.

To visualize the activation maps, we need to pass the image through the model to make predictions.

We will use argmax to find the index corresponding to the maximum value in the prediction, which gives us the predicted class. Now, we take the predicted value for that class by the model.