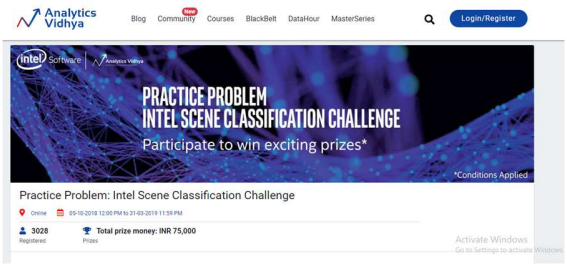
Intel Image Classification

ABSTRACT: Our project focuses on using machine learning techniques to develop an image classification system to categorize images into six classes such as Buildings, Forest, Glacier, Sea, Mountain, and Street .The primary goal is to develop an efficient model capable of identifying various landscapes. We try to classify the dataset using various classifiers and then choose the best model after analysing the performance of each classifier model.

I. INTRODUCTION:

This project is based on the Intel Scene Classification Challenge which was conducted by Analytics Vidhya in collaboration with Intel. You are provided with a dataset of 25,000 images from a wide range of natural scenes from all around the world.The images are of size 150 x 150 pixels. Your task is to identify which kind of scene the image contains. The categories were Buildings, Forest, Glacier, Sea, Mountain and Street.It consists of 6 classes.



This multi-class classification can be helpful in many ways such as

1) Enhancing navigation apps by providing information about the environment.

2) Analyzing urban landscapes to help in city planning and infrastructure development.

3) Monitoring wildlife habitats through the identification of forests, mountains, and other natural landscapes.

4) Enhancing camera features with automatic scene recognition to optimize camera settings for different environments.

5) Creating realistic virtual environments for different types of landscapes in Virtual Reality .

The following sections in this report explains about our project in the following manner. Section II consists of the information about the existing works related to this concept, sectionIII explains about our project work, sectionIV explains about the results that we received at the end of the project and the final section is the conclusion part.

II. RELATED WORKS:

Classification of this particular dataset was performed by many others and their works were published in kaggle.Some of these codes were used by us for reference.In this section, the different methods used by different people are discussed along with their drawbacks.

WORK BY	ACCURACY
VINCENZOOO	89.3%
UZAIR KHAN	82%
ESABELLE CHEN	88.4%
PTTRANG513	88.7%

1) WORK BY VINCENZOOO:

https://www.kaggle.com/code/vincee/intel-image-classification-cnn-keras/notebook

This code defines a Convolutional Neural Network (CNN) for image classification using TensorFlow and Keras. It begins by loading a dataset consisting of images from different categories (mountain, street, glacier, buildings, sea, forest). The dataset is divided into training and test sets. The CNN architecture consists of two convolutional layers followed by max-pooling layers for feature extraction, a flattening layer, and two fully connected layers for classification. The model is compiled with the Adam optimizer and sparse categorical cross-entropy loss. The training process is visualized by plotting accuracy and loss over epochs using the plot_accuracy_loss function.

2) WORK BY UZAIR KHAN:

https://www.kaggle.com/code/youngtaek/intel-image-classification-densenet-with-focalloss

In this code, a DenseNet model is constructed using the Keras framework for image classification. The model is configured for multi-class classification (6 classes) using

categorical focal loss as the loss function.The images are preprocessed by normalizing pixel values and converting labels to one-hot encoded format. Focal loss functions, both binary and categorical versions, are implemented for training, providing a way to handle class imbalance and focus on challenging examples.The model is compiled using the Adam optimizer with a specified learning rate. Training is performed for 35 epochs with a 30% validation split. The training progress is stored in the 'trained' variable, and metrics such as loss and accuracy are monitored during the training process.

3) WORK BY ESABELLE CHEN:

https://www.kaggle.com/code/esabellechen/intel-img-classification-by-esb

This code defines and trains two linear classifiers (model_Linear1 and model_Linear2) using PyTorch. The linear classifiers are implemented as simple fully connected neural networks. The training process includes optimizing the models' parameters with different optimizers (Adam and SGD) and learning rates. The training progress is printed every epoch, showing training and testing loss, accuracy, and learning rate.

4) WORK BY PTTRANG513:

https://www.kaggle.com/code/pttrang513/intel-image-classification-alexnet

This code defines and trains an AlexNet architecture for image classification using PyTorch. It prepares the dataset with data augmentation and normalizes the images. The AlexNet model is initialized and its parameters are set. The training loop includes optimizing the model using the Adam optimizer, and the accuracy and loss are calculated for both the training and validation sets. The best model is saved based on the lowest validation loss.

III. PROJECT WORK:

Used Dataset:

The dataset that we used for this project(Intel Image Classification dataset) was taken from Kaggle.The dataset consists of 25,000 images from a wide range of natural scenes from all around the world. The images are of size 150 x 150 pixels and divided into multiple folders for train data,test data and data that can be used for prediction respectively.It consists of 6 classes

Used Classifiers:

Various different types of models have been used for the classification of the dataset and the accuracy of each model is calculated.The classifiers are:

K-Nearest Neighbors (KNN) Classifier:

K-Nearest Neighbours is one of the most basic yet essential classification algorithms in Machine Learning. It belongs to the supervised learning domain and is used in various applications in pattern recognition, data mining,etc.The K-NN algorithm works by finding the K nearest neighbors to a given data point based on a distance metric, such as Euclidean distance. The class or value of the data point is then determined by the majority vote or average of the K neighbors.

We performed data augmentation by flattening image batches and storing them in X_train_augmented and X_test_augmented, while their corresponding labels are stored in y_train_augmented and y_test_augmented. These augmented datasets are then used to train a k-nearest neighbors classifier.

Support Vector Machines (SVM):

Support Vector Machine (SVM) is a supervised machine learning algorithm used for both classification and regression. The main objective of the SVM algorithm is to find the optimal hyperplane in an N-dimensional space that can separate the data points in different classes in the feature space. The hyperplane tries that the margin between the closest points of different classes should be as maximum as possible. We perform data augmentation and then train the SVM Classifier.

Naïve Bayes classifier :

Naive Bayes classifiers are a collection of classification algorithms based on Bayes' Theorem. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other.They are

fast and easy to implement but their biggest disadvantage is that the requirement of predictors to be independent.Given a set of features, the classifier calculates the probability of each class for the given features using Bayes' theorem. The class with the highest probability is assigned as the predicted class. We augmented the data using TensorFlow's ImageDataGenerator with horizontal and vertical flips, and prepares the data for a Naive Bayes classifier.

Decision Tree Classifier :

Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.The decisions or the test are performed on the basis of features of the given dataset.It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.We created augmented training and testing datasets by reshaping image batches and extracting labels. It is then used to train a Decision Tree classifier.

CNN Classifier:

A Convolutional Neural Network (CNN) is a type of deep learning model designed for processing and analyzing visual data, particularly images. CNNs have proven highly effective in tasks such as image recognition, object detection, and classification. They are composed of layers that automatically learn hierarchical patterns and features directly from the raw pixel data of images.

Key components of a CNN include:

Convolutional Layers: These layers apply convolutional operations to the input images, enabling the network to learn local patterns and features.

Pooling Layers: Pooling layers downsample the spatial dimensions of the input data, reducing its complexity and computational requirements. Common pooling operations include max pooling and average pooling.

Activation Functions: Activation functions, such as ReLU (Rectified Linear Unit), introduce non-linearity to the model, allowing it to learn complex relationships in the data.

Fully Connected Layers: Fully connected layers connect every neuron from one layer to every neuron in the next layer, enabling the model to make decisions based on learned features.

Image generators (train_generator and test_generator) are established to load and preprocess images from specified directories. The images are resized to 200x200 pixels, and the generators are set up to work with categorical labels. The summary reports the total number of images and classes found in the training and testing sets (14034 images belonging to 6 classes in training and 3000 images belonging to 6 classes in testing).

IV. RESULTS

GENERATING PREDICTIONS FOR THE TEST DATA :

We present the model prediction for some of the sample images of our test set.

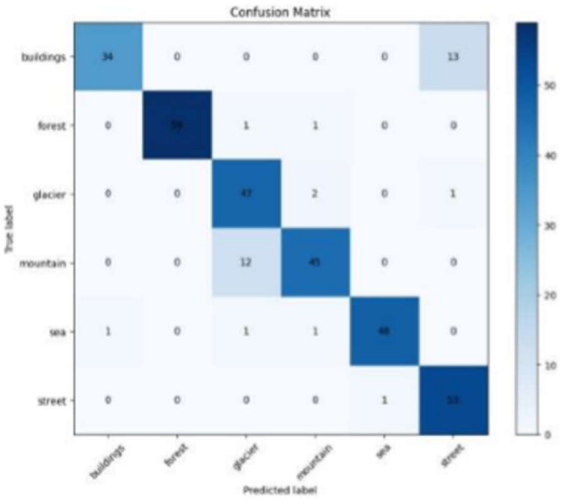
EVALUATION METRICS:

COMPARISON OF ACCURACY OF DIFFERENT MODELS:

MODEL	ACCURACY
CNN	89%
NAÏVE BAYES	42%
SVM	42%
KNN	35%
DECISION TREE	35%

CONFUSION MATRIX:

A Confusion Matrix is a table used in machine learning to evaluate the performance of a classification model. It helps you understand how well your model is classifying instances by showing the count of true positives, true negatives, false positives, and false negatives.



CLASSIFICATION REPORT:

Classification Report is the report which explains everything about the classification. This is the summary of the quality of classification made by the constructed ML model. It comprises mainly 5 columns and (N+3) rows. The first column is the class label's name and followed by Precision, Recall, F1-score, and Support. N rows are for N class labels and other three rows are for accuracy, macro average, and weighted average.

	precision	recall	f1-score	support
buildings	0.97	0.72	0.83	47
forest	1.00	0.97	0.98	61
glacier	0.77	0.94	0.85	50
mountain	0.92	0.79	0.85	57
sea	0.98	0.94	0.96	51
street	0.79	0.98	0.88	54
accuracy			0.89	320
macro avg	0.91	0.89	0.89	320
weighted avg	0.91	0.89	0.89	320

V. CONCLUSION:

This classification project aimed to effectively categorize images into 6 different classes using various machine learning models, including Convolutional Neural Networks (CNNs), as well as traditional classifiers like Naive Bayes and Decision Trees. After trying to classify our dataset using various models ,we got a maximum accuracy of 89% while we used CNN classifier.
This model can be used to identify the landscape of the surrounding environment just by using its images.

REFERENCES:

[1] **WORK BY VINCENZOOO:**
https://www.kaggle.com/code/vincee/intel-image-classification-cnn-keras/notebook

[2] **WORK BY UZAIR KHAN:**
https://www.kaggle.com/code/youngtaek/intel-image-classification-densenet-with-focalloss

[3] **WORK BY ESABELLE CHEN:**
https://www.kaggle.com/code/esabellechen/intel-img-classification-by-esb

[4]**WORK BY PTTRANG513:**
https://www.kaggle.com/code/pttrang513/intel-image-classification-alexnet