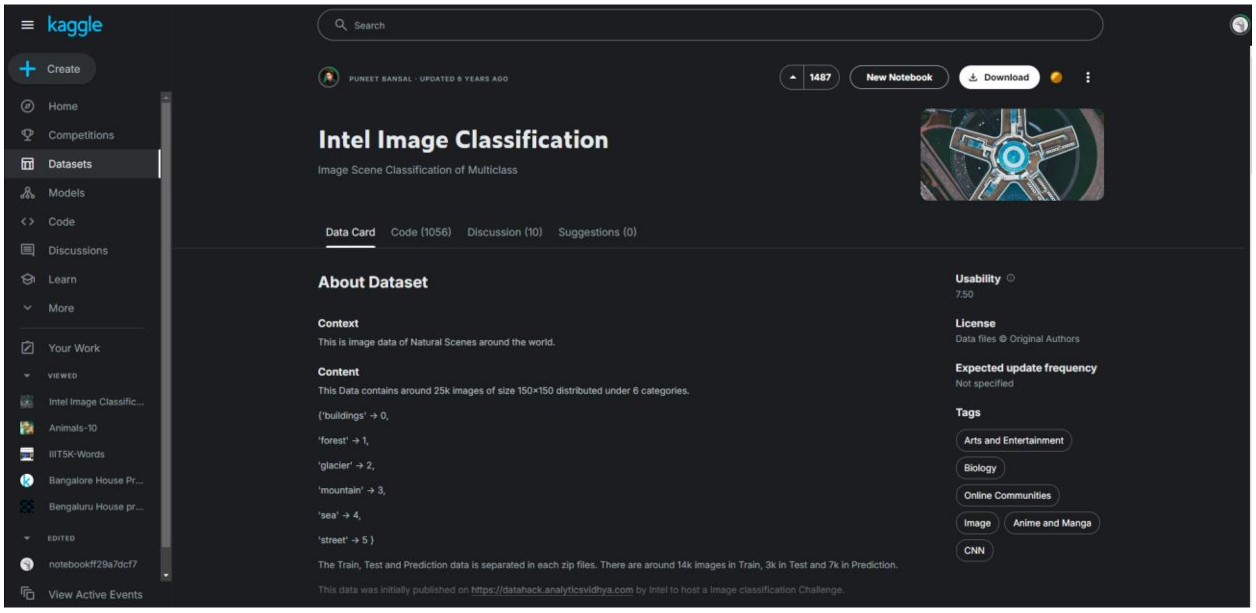
Project Report: Real World Image Classification on Intel Image Classification Dataset

Introduction:

Image classification is a critical area of study in computer vision that involves assigning a label to an input image based on its content. This technology has numerous applications, ranging from self-driving cars to medical image analysis. In this project, we aim to build a deep learning model that classifies images from the Intel Image Classification dataset. The goal is to achieve high accuracy and generalization on unseen data, providing a reliable tool for image classification tasks.

Dataset:



Intel Image Classification

Image Scene Classification of Multiclass

About Dataset

Context

This is image data of Natural Scenes around the world.

Content

This Data contains around 25k images of size 150×150 distributed under 6 categories.

{'buildings' -> 0,

'forest' -> 1,

'glacier' -> 2,

'mountain' -> 3,

'sea' -> 4,

'street' -> 5 }

The Train, Test and Prediction data is separated in each zip files. There are around 14k images in Train, 3k in Test and 7k in Prediction.

This data was initially published on https://datahack.analyticsvidhya.com by Intel to host a Image classification Challenge.

Usability
7.50

License
Data files © Original Authors

Expected update frequency
Not specified

Tags
Arts and Entertainment
Biology
Online Communities
Image
Anime and Manga
CNN

1. Overview

The Intel Image Classification dataset is a comprehensive collection of images designed to benchmark image classification algorithms. This dataset comprises a diverse range of images categorized into six distinct classes:

Buildings: Images featuring various architectural structures.

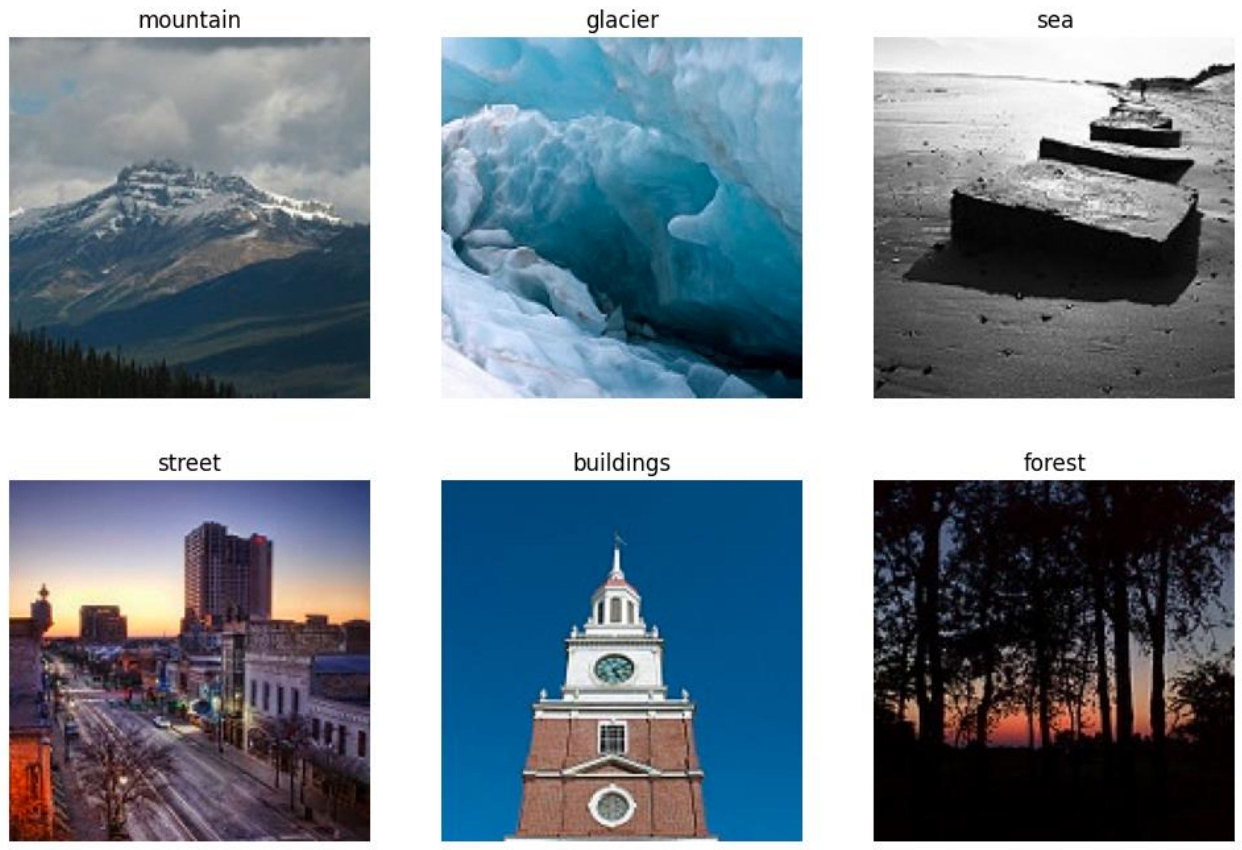
Forest: Images depicting dense woodland areas.

Glacier: Images showcasing ice formations and glacial landscapes.

Mountain: Images of mountainous terrain and peaks.

Sea: Images capturing oceanic views and seascapes.

Street: Images of urban environments, including roads and infrastructure.



2. Data Preprocessing

Before utilizing the dataset for training the model, several preprocessing steps were performed to ensure optimal model performance:

Resizing: Each image was resized to a consistent dimension (e.g., 150x150 pixels) to ensure uniformity in input size, which is essential for batch processing in neural networks.

Normalization: The pixel values of the images were normalized to a range of [0, 1] by dividing by 255. This step is crucial as it helps accelerate the training process and improve the model's convergence.

Data Splitting: The dataset was divided into training and validation sets to evaluate the model's performance on unseen data. A common split ratio of 80% for training and 20% for validation was applied, ensuring a robust assessment of the model's accuracy.

3. Data Augmentation

To enhance the diversity of the training dataset and reduce the risk of overfitting, data augmentation techniques were employed. These techniques included:

Rotation: Randomly rotating images within a specified range to introduce variability in orientation.

Width and Height Shifting: Randomly shifting images along the width and height to simulate translations in the image.

Shearing and Zooming: Applying shear transformations and random zooming to create variations in the spatial distribution of the image content.

Horizontal Flipping: Randomly flipping images horizontally to introduce mirrored views, particularly useful for datasets where orientation may vary.

These preprocessing and augmentation strategies collectively contributed to creating a robust dataset that enhances the model's ability to generalize to new, unseen images.

Methodology:

1. Model Selection

For this project, a Convolutional Neural Network (CNN) was selected as the primary architecture for image classification tasks. CNNs are particularly effective in extracting spatial hierarchies from images, making them well-suited for tasks involving visual data. The chosen model comprises several convolutional layers, followed by pooling layers, and ultimately fully connected layers to perform classification. This architecture is known for its ability to learn features directly from the raw pixel data, thus minimizing the need for manual feature extraction.

2. Model Architecture

The architecture of the CNN used for this project includes the following layers:

Convolutional Layers: Multiple convolutional layers were employed to learn different feature representations from the images. Each layer used a set of learnable filters (kernels) to detect specific patterns, such as edges and textures.

Activation Function: The ReLU (Rectified Linear Unit) activation function was used in the convolutional layers to introduce non-linearity into the model, enabling it to learn complex mappings.

Pooling Layers: Max pooling layers were used to down-sample the feature maps, reducing dimensionality while retaining the most important features. This step helps to decrease computational load and mitigate overfitting.

Fully Connected Layers: The final layers of the model consist of fully connected layers that take the flattened output from the convolutional layers and produce class probabilities. A softmax activation function was used in the output layer to generate the final predictions for the six image classes.

3. Model Training

The model was compiled using the following parameters:

Loss Function: Categorical crossentropy was chosen as the loss function, suitable for multi-class classification tasks.

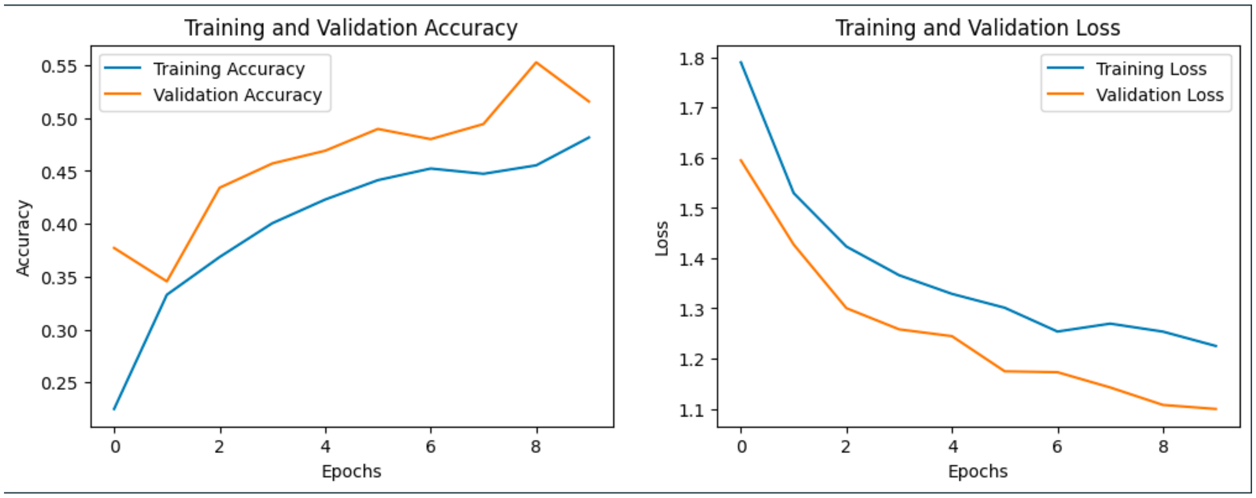
Optimizer: The Adam optimizer was utilized for its adaptive learning rate capabilities, which often leads to faster convergence.

Metrics: Accuracy was selected as the evaluation metric to monitor the model's performance during training and validation.

Results

Model Performance:

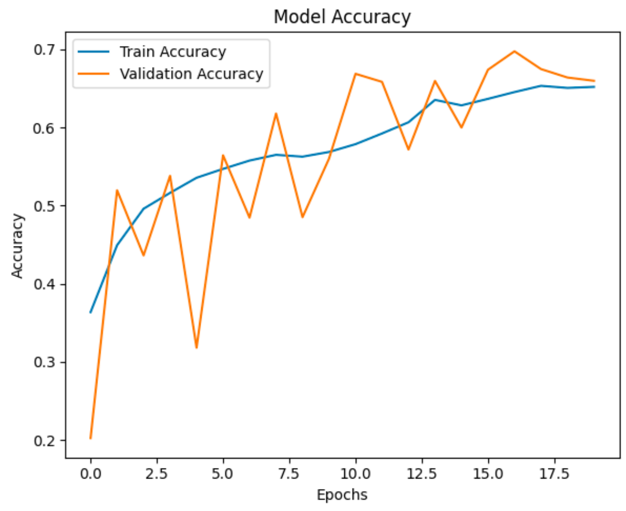
First time Model Fitting result:

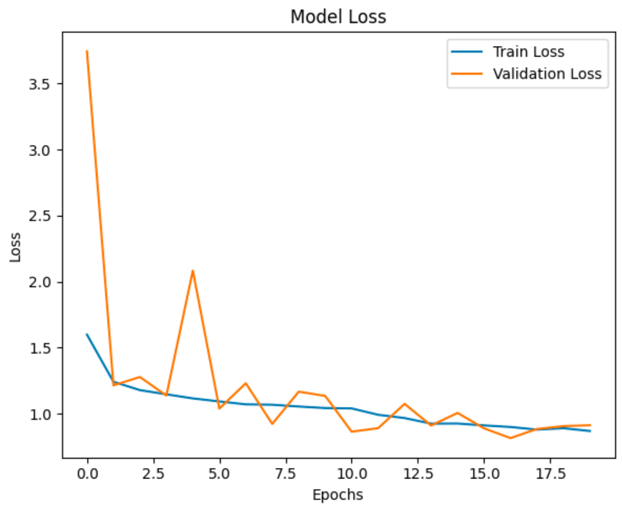


Max Training Accuracy: 0.48, Max Validation Accuracy: 0.55

Min Training Loss: 1.25, Min Validation Loss: 1.1

Second time Model Fitting result:



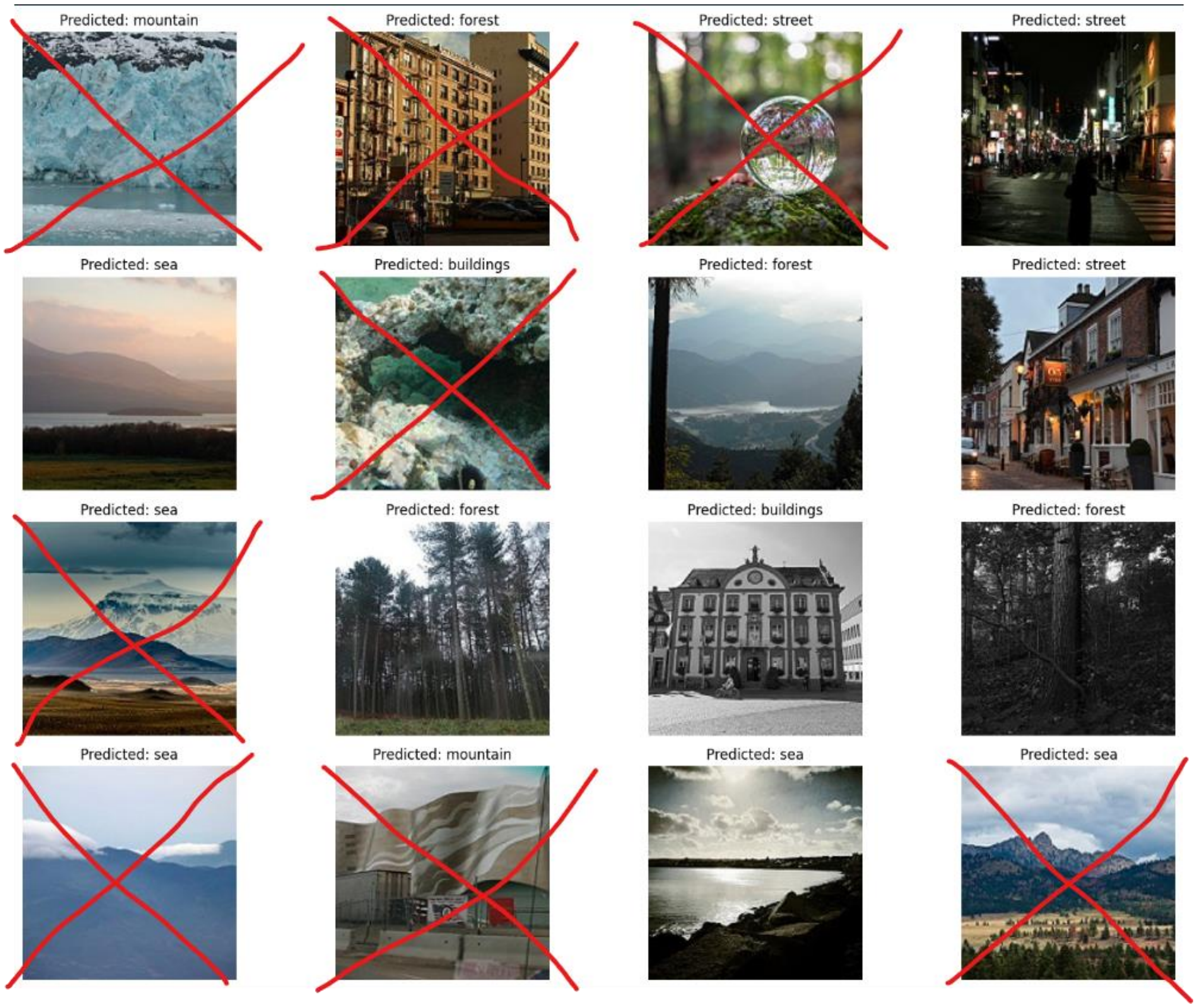


Max Training Accuracy: 0.64, Max Validation Accuracy: 0.68

Min Training Loss: 1.0, Min Validation Loss: 1.1

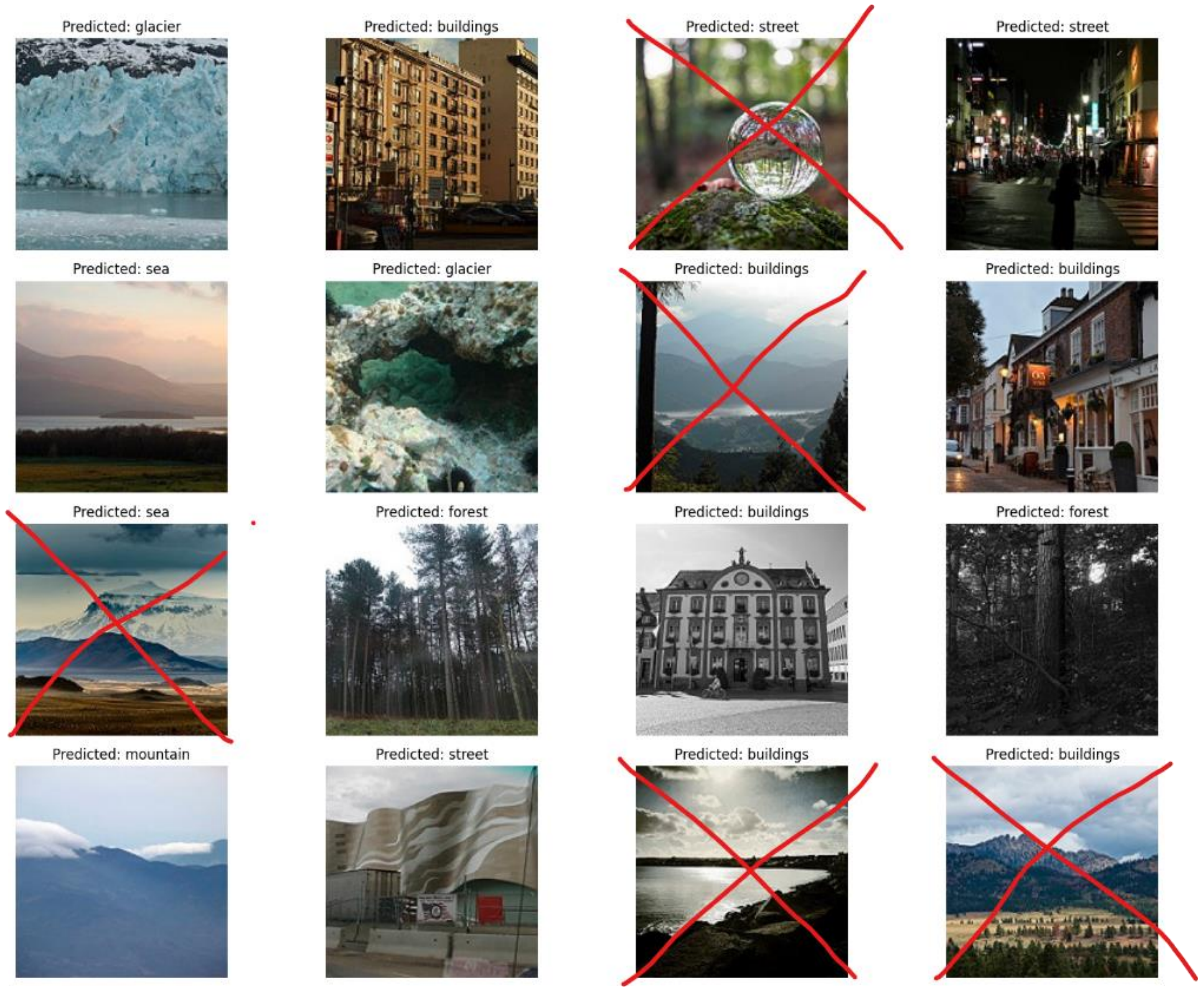
Predictions:

Predictions done by initial model training on the prediction images provided by Intel Dataset:



True:8, False:8, Accuracy: 50%

Predictions done by second model training on the same prediction images provided by Intel Dataset:



True:11, False:5, Accuracy: 68.75%

Conclusion

In this project, we implemented an image classification model using the Intel Image Classification dataset. By applying a Convolutional Neural Network (CNN) approach, we aimed to classify images accurately into their respective categories. Our model's performance was evaluated using key metrics, including accuracy and loss for both training and validation datasets. Additionally, prediction accuracy was assessed on a set of unseen test images.

During the initial model fitting, the CNN achieved a maximum training accuracy of 48% and a validation accuracy of 55%, with a minimum training loss of 1.25 and validation loss of 1.1. The model's prediction accuracy on the test set was 50%, with 8 correct and 8 incorrect classifications.

After refining the model and applying strategies like data augmentation and learning rate adjustments, the second training session showed notable improvement. The model's accuracy increased to 64% for training and 68% for validation, with a minimum training loss of 1.0 and validation loss still at 1.1. For the test predictions, the accuracy increased to 68.75%, with 11 correct predictions and 5 incorrect ones.

These results demonstrate the impact of iterative training and tuning on model performance. While the improved accuracy is promising, further enhancements, such as deeper architectures or alternative regularization techniques, could be explored to achieve higher accuracy. Overall, this project provides a solid foundation for understanding image classification using CNNs and highlights the importance of hyperparameter tuning and model optimization in achieving reliable classification results.