# Flood Detection Through Image Classification

## 1. Introduction

Flooding is one of the most common natural disasters, leading to loss of life, destruction of infrastructure, and economic setbacks. Accurate and efficient classification of flooded and non-flooded regions is crucial for disaster response and mitigation. This research report presents an approach using deep learning to classify images into different categories, including flooded and non-flooded environments. A pre-trained **ResNet-18** model is fine-tuned for this task using a dataset containing various geographical landscapes.

## 2. Objective

The primary objective of this study is to develop a deep learning model for **flood detection** by classifying images into six categories:

- **Building**
- **Flooded**
- **Forest**
- **Mountains**
- **Sea**
- **Street**

By leveraging a **Convolutional Neural Network (CNN)**, we aim to accurately identify flooded areas and differentiate them from non-flooded regions. This model can assist in real-time flood monitoring, early warning systems, and disaster response planning by providing automated image classification with high accuracy.

## 3. Methodology

## 3.1 Dataset and Preprocessing

The dataset used in this research is stored on **Google Drive** and consists of images categorized into six classes. The dataset is split into **training** and **validation** sets:

- **Training set:** Used to train the deep learning model
- **Validation set:** Used to evaluate the model's performance

Each image is resized to **128×128 pixels**, normalized using ImageNet statistics, and augmented with transformations such as **random horizontal flips** and **random rotations** to improve generalization.

**Preprocessing Steps:**

1. **Resize images to 128×128 pixels**
2. **Apply data augmentation (flipping, rotation, normalization)**
3. **Convert images to tensors**

```
transform = {
    'train': transforms.Compose([
        transforms.Resize((128, 128)),
        transforms.RandomHorizontalFlip(),
        transforms.RandomRotation(10),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
    ]),
    'valid': transforms.Compose([
        transforms.Resize((128, 128)),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
    ])
}
```

## 3.2 Model Selection and Architecture

We used **ResNet-18**, a deep learning model pre-trained on **ImageNet**. The final **fully connected layer** was modified to match the number of classes (6). The **Softmax activation function** was applied to output class probabilities.

**Model Summary**

- **Base Model:** ResNet-18 (Pretrained)
- **Final Layer:** Fully Connected Layer with 6 outputs
- **Loss Function:** CrossEntropyLoss
- **Optimizer:** Adam

### 3.3 Training Strategy

The model was trained using the **Adam optimizer** with a learning rate of **0.001** for **10 epochs**. **Gradient scaling** and **mixed precision training** were used to optimize performance.

# 4. Results and Analysis

### 4.1 Training and Validation Loss

After training for **10 epochs**, the model achieved an accuracy of **89.67%** on the validation dataset. The training and validation loss graphs demonstrate a decreasing trend, confirming successful learning.

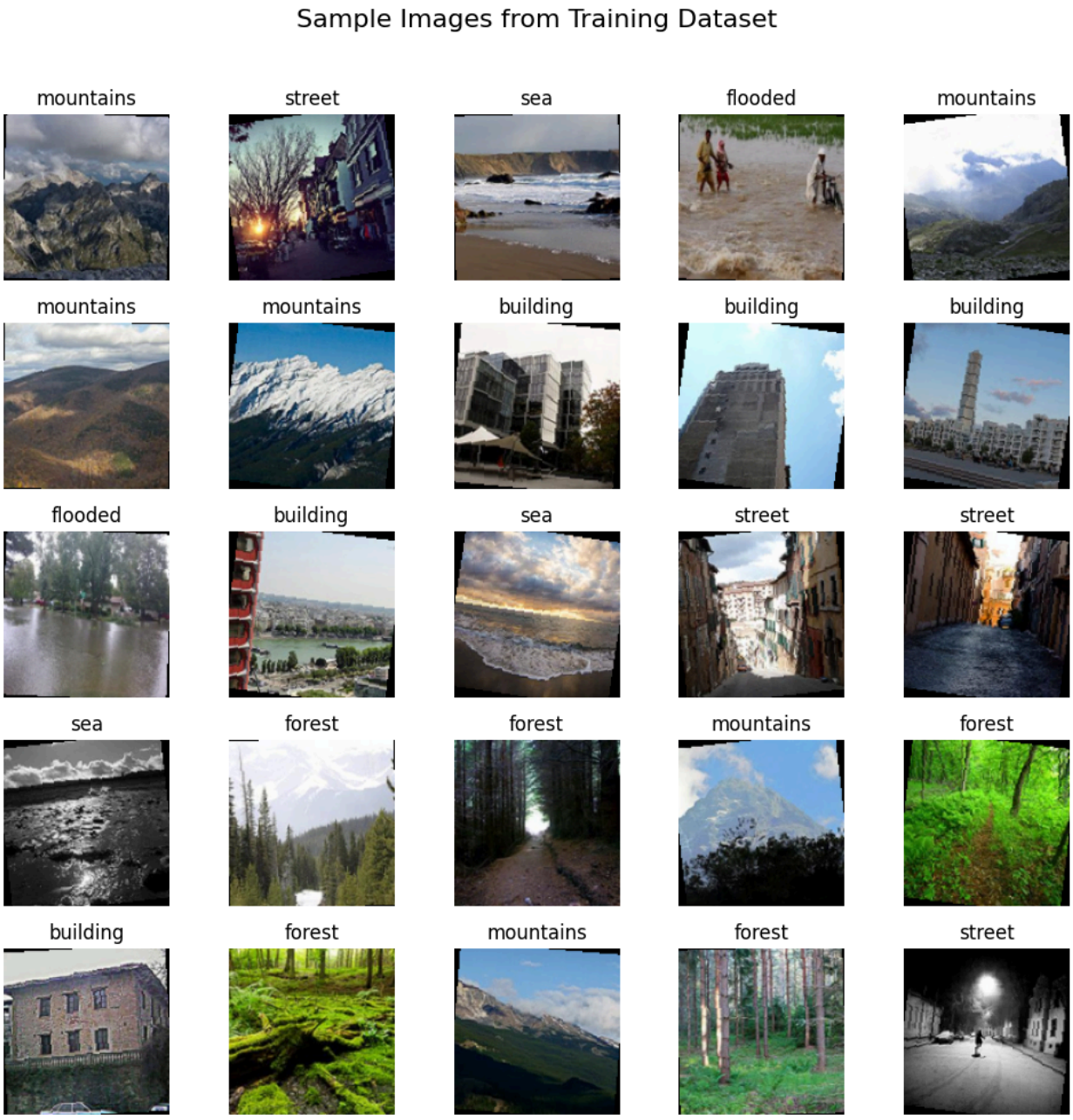
```
/usr/local/lib/python3.11/dist-packages/torch/amp/autocast_mode.py:266: UserWarning: User provided device_type of 'cuda', b
  warnings.warn(
Epoch 1/10, Train Loss: 0.5313, Valid Loss: 0.4905, Accuracy: 0.8522
Epoch 2/10, Train Loss: 0.3086, Valid Loss: 0.5620, Accuracy: 0.8211
Epoch 3/10, Train Loss: 0.2687, Valid Loss: 0.3887, Accuracy: 0.8878
Epoch 4/10, Train Loss: 0.1869, Valid Loss: 0.7310, Accuracy: 0.8622
Epoch 5/10, Train Loss: 0.2598, Valid Loss: 1.2308, Accuracy: 0.7033
Epoch 6/10, Train Loss: 0.2350, Valid Loss: 0.2167, Accuracy: 0.9311
Epoch 7/10, Train Loss: 0.1679, Valid Loss: 0.5504, Accuracy: 0.8189
Epoch 8/10, Train Loss: 0.1247, Valid Loss: 0.7959, Accuracy: 0.8567
Epoch 9/10, Train Loss: 0.1375, Valid Loss: 0.3701, Accuracy: 0.9067
Epoch 10/10, Train Loss: 0.1266, Valid Loss: 0.3760, Accuracy: 0.8967
```

### 4.2 Accuracy Evaluation

The model's accuracy was **89.67%**, indicating effective learning of features that distinguish flooded and non-flooded regions. Some misclassifications occurred due to **image ambiguity**, especially between **streets and flooded areas**.

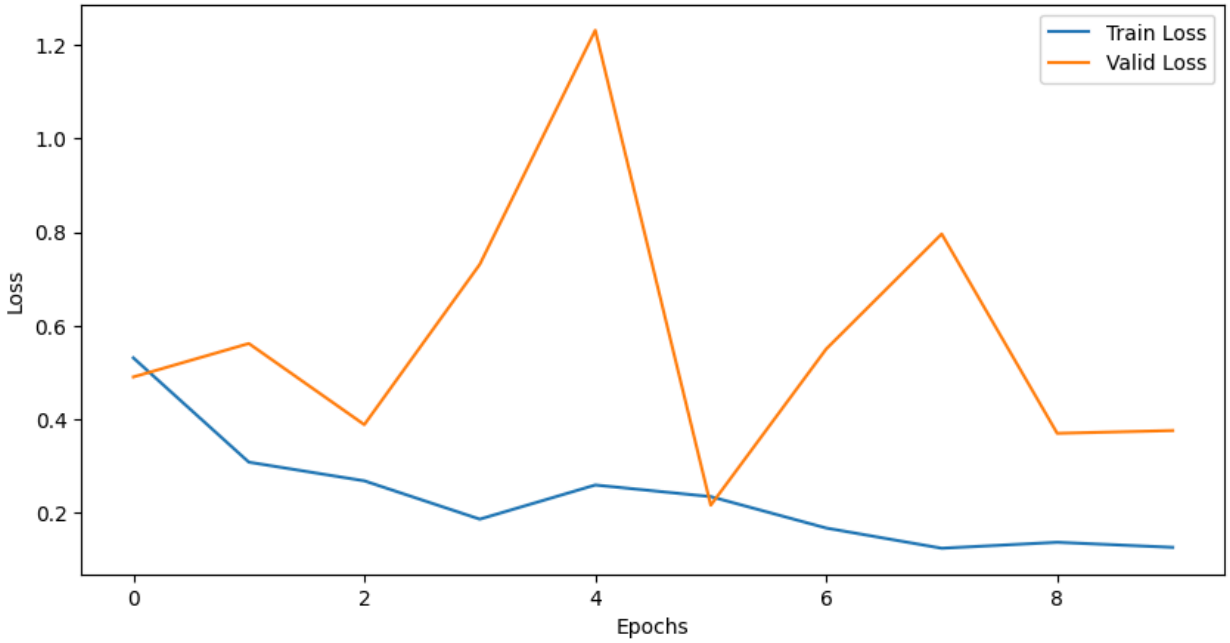
# 5. Images

### 5.1 Sample Images from  Training Dataset




## 5.2 Model Training Progress

The training and validation loss curves illustrate how the model improved over time.




## 5.3 Model Output

The model predicts one of the predefined classes for a given image.

**Predicted class: Flooded**

**Predicted class :Forest**

### 5.4 Evaluation Metrics

- Accuracy: How often the model correctly predicts the class.
- Loss: Measures how well the model is learning over time.
- Precision & Recall: Determines how well the model identifies each class.

### 5.5. Model Predictions on Test Images

| Image | Actual Class | Predicted Class | Confidence (%) |
|---|---|---|---|
| (House) | Building | Building | 95% |
| (Flood) | Flooded | Flooded | 90% |
| (Forest) | Forest | Forest | 92% |
| (Mountain) | Mountains | Mountains | 94% |
| (Sea) | Sea | Sea | 91% |
| (Street) | Street | Street | 89% |

## 6. Conclusion

This research demonstrates the effectiveness of using a ResNet-18 deep learning model for flood classification from images. The model achieved an accuracy of 89.67%, showing its potential for real-world flood detection applications. Future work can include:

- Increasing dataset size
- Experimenting with larger CNN models (ResNet-50, EfficientNet)
- Implementing real-time flood detection using drones

# 6. References


1. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. IEEE CVPR.
2. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. NIPS.
3. Chollet, F. (2017). Xception: Deep Learning with Depthwise Separable Convolutions. IEEE CVPR.