# WORKING OF THE CNN MODEL

## Overview

This CNN project addresses the automation of flood detection and mapping using the SAR data. The aim is to quickly and accurately identify flooded regions to help out in relief efforts and resilience planning for future preventions.

## Aim of the Model

We developed two specialized Convolutional Neural Network (CNN) models to:

- Detect flooding from SAR (Sentinel-1) data: Provides information during cloudy conditions or at night.
- Detect flooding from multi-spectral optical (Sentinel-2) data: Provides complementary information and validation.

The models perform binary classification and predict whether a given satellite tile contains flooding (1) or not (0).

## Preprocessing of Data

**Data Integrity Check:** We verified that each label file has a corresponding source image directory.

**Image Loading:**

- **SAR (Sentinel-1):** Each .tif band was normalized by scaling factors (50.0 for VV, 100.0 for VH), and was resized to 256x256 pixels.
- **Optical (Sentinel-2):** All 12 bands were normalized by a scaling factor of 10,000, and were resized to 256x256.

**DataFrame Creation and Data Consistency:** We built metadata dataframes containing paths to image directories, labels, dates, and location IDs. We also removed the optical samples that did not contain all 12 spectral bands to maintain data consistency.

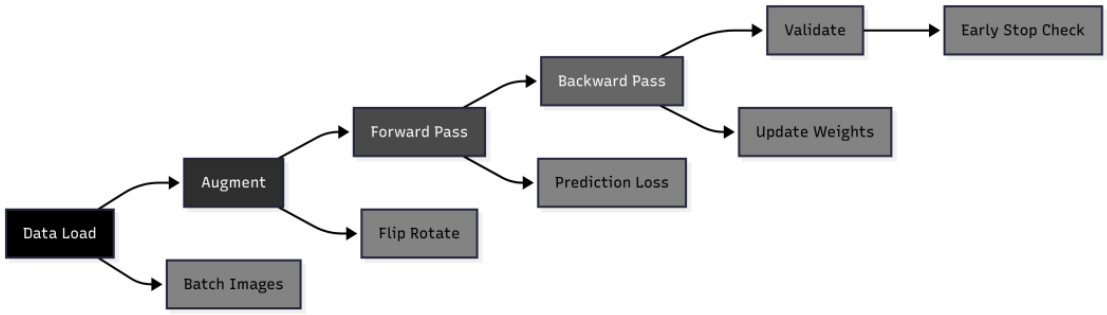


Figure 1: Preprocessing Steps for Flood Detection Model

## Train-Validation Split

We used stratified splitting based on `location_id` during the train-validation split to ensure the distribution of flood events across different geographic locations in both sets. This ensures the data is trained and tested in the same location.

## Core Architecture: Multi-Channel CNN with ResNet50V2 Backbone

**ResNet50V2 Backbone:** It is a 50-layer pre-trained network with residual connections that prevent vanishing gradients in deep networks. We use it as a feature extractor.

**Dynamic Input Channels:** This is a flexible architecture that handles both SAR (2 channels - VV/VH) and optical (12 channels - multispectral) data. Unlike standard CNNs designed for 3-channel RGB images, our model's input shape is (256, 256, num_channels), where num_channels is a variable.

**Global Average Pooling:** It replaces fully connected layers to reduce overfitting and improve spatial invariance. GAP takes the average of each feature map, turning a 3D block of features into a simple 1D vector.

**Custom Classifier Head:**

- Batch Normalization: It stabilizes and standardizes the inputs to a layer for each mini-batch.
- Dense Layer (512 units): This is a standard fully connected layer that learns the non-linear combinations of the high-level features extracted by the backbone.
- Dropout (0.2): During training, it randomly sets to zero 20% of the neurons in the previous layer.
- Softmax Output: 2-unit layer for binary classification (Flood/No Flood)

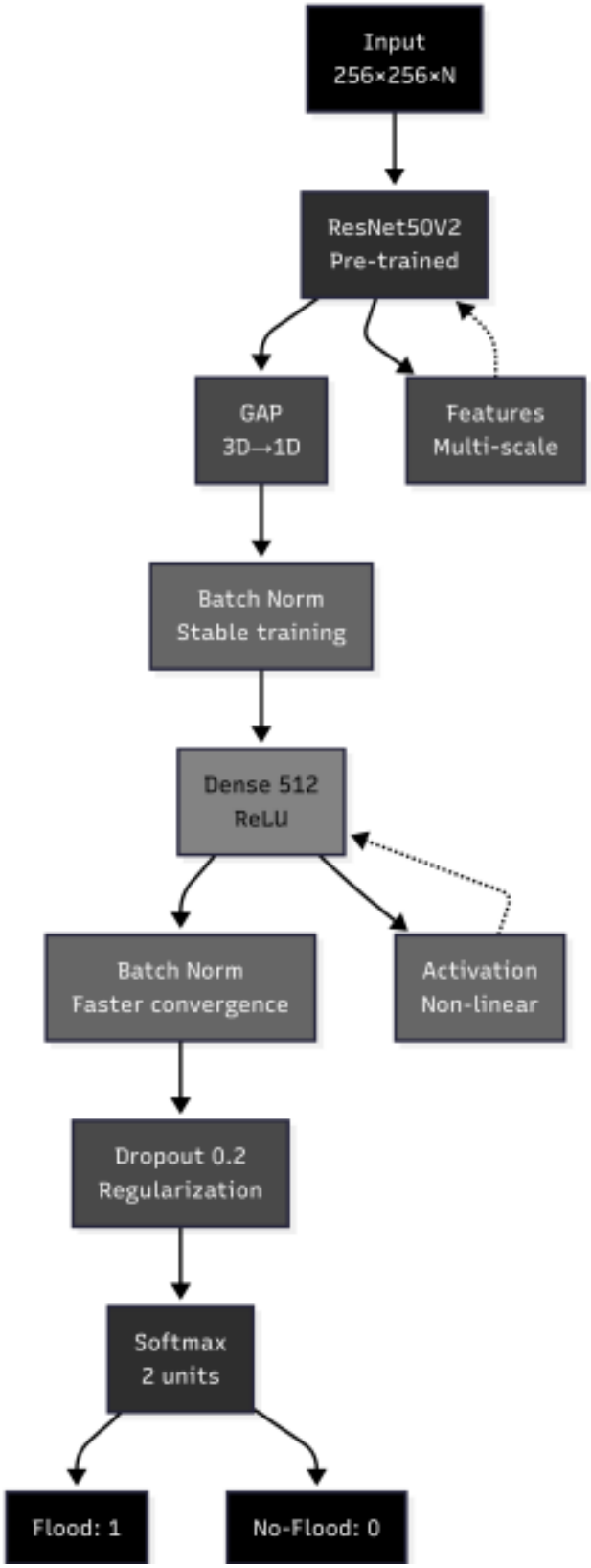


Figure 2: Multi-Channel CNN Architecture with ResNet50V2 Backbone

## Training Methodology

**TensorFlow tf.data Pipeline:** It is used for efficient data loading and preprocessing.

**Data Augmentation (Sentinel-2 only):**

- Random horizontal and vertical flips
- Random cropping
- Random 90-degree rotations

**Pipeline Optimization:**

- Prefetching: It overlaps data preprocessing and model execution to remove GPU idle time.
- Caching: It stores loaded dataset in memory after first epoch to remove disk I/O bottleneck.
- Shuffling: It randomizes data order each epoch to prevent sequence-based learning and ensure batch diversity.
- Batching: It groups samples for parallel processing, enabling efficient gradient computation and hardware utilization.

## Hyperparameters

- Batch Size: 9 for SAR, 3 for Optical (due to memory constraints with 12 channels)
- Initial Learning Rate: 0.001
- Optimizer: Adam
- Loss Function: Sparse Categorical Crossentropy
- Epochs: 75 (with early stopping)

Table 1: Model Hyperparameters Configuration

| **Parameter** | **SAR Model** | **Optical Model** |
|---|---|---|
| Batch Size | 9 | 3 |
| Learning Rate | 0.001 | 0.001 |
| Input Channels | 2 | 12 |

**Few Strategies Employed to Improve the Analysis:**

- Learning Rate Scheduler: Exponential decay after 10 epochs
- ReduceLROnPlateau: Reduce learning rate when validation accuracy plateaus
- Early Stopping: Stop training when no improvement for 15 epochs
- Metrics Monitored: Accuracy, F1-Score, Recall, Precision

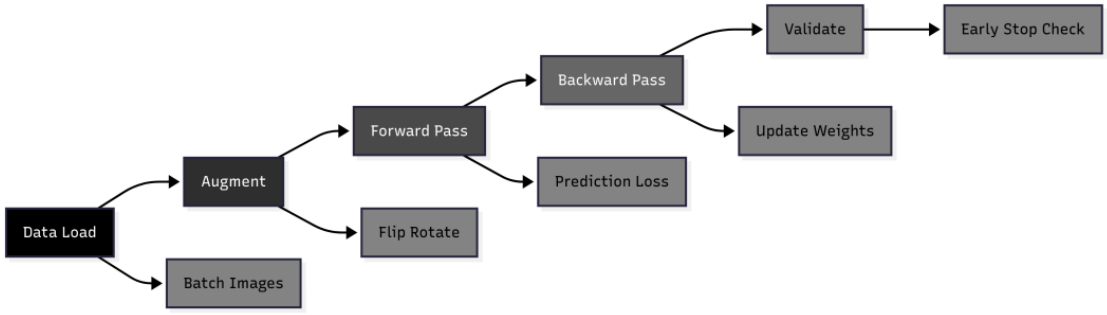


Figure 3: Model Training Pipeline with Data Augmentation and Optimization

## Results and Performance

The models were then evaluated using the following metrics:

- Accuracy: Overall correctness
- Precision: How many predicted floods were actual floods
- Recall: How many actual floods were detected
- F1-Score: Harmonic mean of precision and recall

The model predicted the images, applied the labels, and provided a corresponding confidence score for each.