CNN Model OUTPUTS

October 5, 2025

Model Training

The Convolutional Neural Network (CNN) was trained on Synthetic Aperture Radar (SAR) data for flood detection. The training was configured to run for 50 epochs.

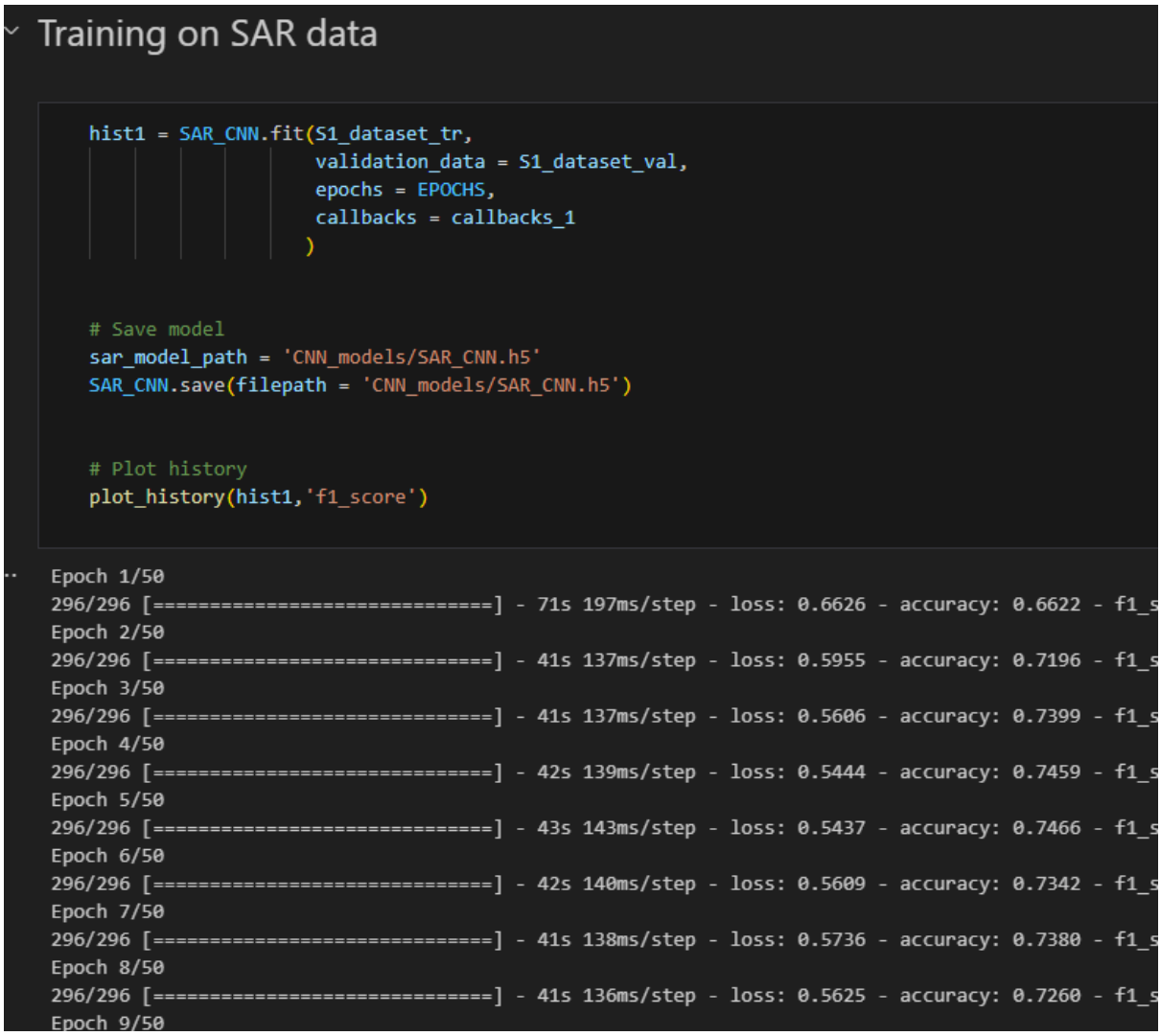

Figure 1: Training Process

Output

The performance of the model over the 50 epochs was plotted to visualize the learning process. The plots in Figure 2 show the training and validation metrics for loss, accuracy, and F1-score. A noticeable

gap between the training and validation curves, particularly for accuracy and F1-score, suggests a degree of overfitting. The validation loss decreases and then stabilizes, indicating that the model has learned relevant features from the data.

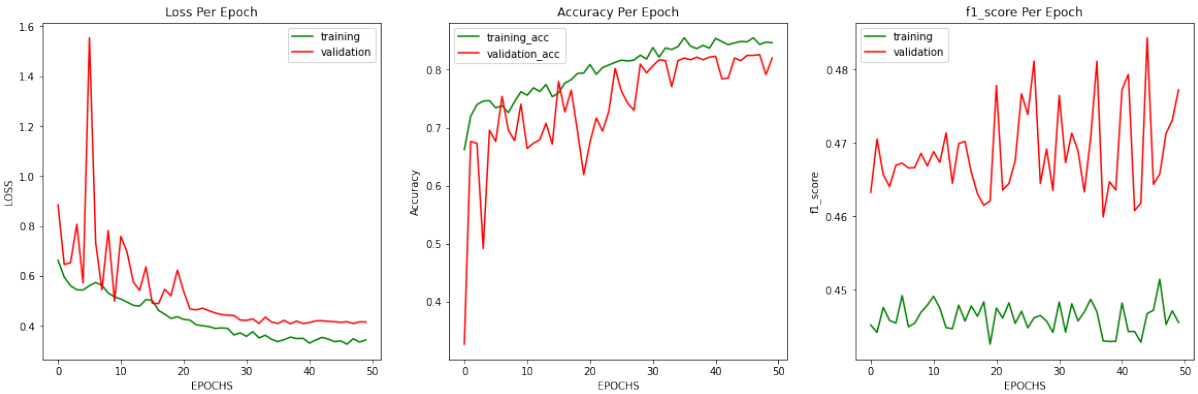

Figure 2: Output Plots

Model Prediction

After training, the saved model loaded to perform predictions on new, unseen SAR images. A single GeoTIFF image was passed to the `predict_flood` function. The model processed the image and returned a prediction. As shown in Figure 4, the model classified the input image with the label "Flooding" with a high confidence score of 0.7943.

Prediction and testing

```python
# Define the mapping between model output classes and their labels
label_map = {0: "No Flooding", 1: "Flooding"}

# Configuration class - must match the settings used during training
class CFG:
    img_size = (256, 256)


print("Loading the trained model...")
SAR_CNN = tf.keras.models.load_model(
    'CNN_models/SAR_CNN.h5',
    custom_objects={'f1_score': None, 'recall_m': None, 'precision_m': None}
)
print("Model loaded successfully!")


def preprocess_sar_image(geotiff_path):

    # Open the GeoTIFF file and read all its bands
    with rasterio.open(geotiff_path) as dataset:
        # Ensure the file has both VV and VH bands
        if dataset.count < 2:
            raise ValueError("The input GeoTIFF must have at least 2 bands (VV and VH).")

        # Read all bands into a NumPy array
        all_bands = dataset.read()

    # Reorder dimensions to match the model's expected input format
    img = all_bands.transpose((1, 2, 0))

    # Apply the same preprocessing steps used during training
    img = img.astype(np.float32) / 50.0
    img = cv2.resize(img, CFG.img_size, interpolation=cv2.INTER_AREA)

    # Add an extra dimension to represent a single image batch
    img = np.expand_dims(img, axis=0)

    return img
```

Figure 3: Testing on a GeoTIFF

```python
def predict_flood(geotiff_path):
    (variable) model_input: Any
    model_input = preprocess_sar_image(geotiff_path)

    # Run the prediction
    pred_probs = SAR_CNN.predict(model_input)

    # Extract the most likely class and its confidence
    class_idx = np.argmax(pred_probs, axis=1)[0]
    confidence = pred_probs[0][class_idx]

    return label_map[class_idx], confidence


# Provide the path to a single SAR GeoTIFF image
geotiff_image_path = "/kaggle/input/test-data/subset_2_of_subset_0_of_S1A_IW_GRDH_1SDV_20220628T115711_20220628T115736_043863_

# Run the prediction and display results
try:
    predicted_label, confidence_score = predict_flood(geotiff_image_path)

    print("\n--- Prediction Result ---")
    print(f"Label: {predicted_label}")
    print(f"Confidence: {confidence_score:.4f}")

except Exception as e:
    print(f"\nAn error occurred during prediction: {e}")
```

```
Loading trained model...
Model loaded successfully.

--- Prediction Result ---
Label: Flooding
Confidence: 0.7943
```

Figure 4: Test Output