Flood Classification Using Explainable CNN and Uncertainty Quantification

Deven Biehler

December 2024

1 Introduction

Flood prediction is a vital tool with increasing societal impact, particularly in areas prone to natural disasters. As climate change increases, the frequency and severity of extreme weather events become more common. Accurate flood forecasting becomes essential for preparedness, resource allocation, and emergency response. Using machine learning, specifically convolutional neural networks (CNNs) trained on remote sensing imagery, we can create models that predict flood severity and provide insights into flood dynamics in near real-time, aiding government agencies, local communities, and emergency services.

This research is motivated by the increasing need for actionable, reliable, open-source, and interpretable flood predictions. This project seeks to address gaps in the literature by developing a CNN-based model that classifies flood severity while incorporating explainability and uncertainty quantification. These features ensure that the model achieves high predictive accuracy and communicates its confidence and rationale, making the predictions trustworthy and actionable for disaster management.

This project sets out to investigate the following questions:

- Can a CNN model accurately predict flooding from satellite imagery?
- Can explainable AI techniques be employed to make the model's predictions interpretable, providing context for its classification choices?
- Can uncertainty alongside interpretability provide trustworthiness in disaster response scenarios?

With a deep interest in applying machine learning to address pressing societal challenges, I chose flood prediction for its potential to mitigate human suffering during disasters. This project also aligns with my focus on making machine learning interpretable and trustworthy. My goal was to build a model that accurately predicts flooding and provides explainable outputs with quantified uncertainty. This added layer of interpretability and reliability could make machine learning predictions more actionable in critical applications like flood response planning.

One major challenge was the need for high-resolution, annotated flood data, as complex spatial and temporal variables create difficulty sourcing accurate data. Open-source satellite data is scarce, and flooding is relatively rare, leaving many flood events undocumented to satellite sensing. Due to the nature of floods, usually obfuscated by clouds, this model was tailored to work with synthetic aperture radar (SAR) imagery from Sentinel-1 satellite to ensure robustness even under cloud cover. SAR imagery can measure the properties of the Earth's surface by sending out pulses of energy that can collect data about the surface it hits and send it back to the sensor. SAR data contains two bands: VV (Vertical Transmit, Vertical Receive) Polarization and VH (Vertical Transmit, Horizontal Receive) Polarization. VV is particularly sensitive to calm water surfaces and is better for detecting man-made structures with vertical features. VH is sensitive to volume scattering caused by complex structures like trees and shrubs and is better at detecting variations in flooded areas than VV. A visualization of SAR next to a true-color image is provided in Figure 1, where the different energy bands are represented from purple to yellow. Given the human interpretability of SAR imagery, the human effort required to annotate this data is expensive.

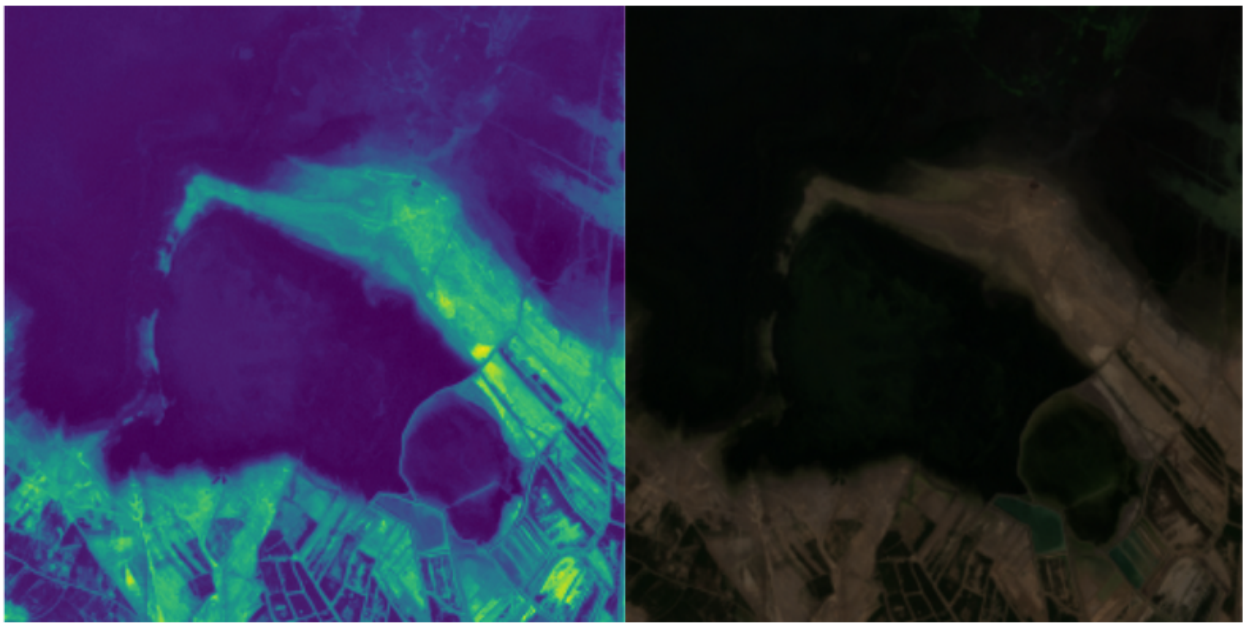

Figure 1: True Color vs SAR Image

Challenges in the model overfitting/underfitting the dataset. Another challenge was finding a good model for binary classification of the SAR imagery. Larger models were found to overfit, while smaller models underfit easily. This posed a problem during training, as much of the computation time was spent evaluating the best model instead of fine-tuning and evaluating the explainability/uncertainty.

Preliminary results demonstrate the model's potential for robust flood detection with an accuracy of 80.8%. The model has a high degree of explainability through saliency maps, which show that the model recognizes key features in the landscape, such as rivers, valleys, cities, etc. Alongside the saliency maps, Monte-Carlo dropout was utilized to give the user uncertainty, with each prediction showing that the model could be trusted in a disaster response scenario.

2 Machine Learning Task

The primary goal is to determine areas at the highest risk of flooding by analyzing satellite images and providing a confident and explainable classification of flooding areas. This involves segmenting images of flooded cities and then using ResNet [2] to classify flooded vs non-flooded images.

The input data for this model will be tiff files of 256 x 256 pixels that are collected from the Sentinel-1 satellite. These images are synthetic-aperture radar (SAR) images, as seen in Figure 1. It's important that the input data is SAR images because flooding tends to happen under heavy cloud cover, which is inaccessible to true-color imagery. These tiff files are paired with binary labels that classify either an image of flooding or not flooding. Flooding usually consists of a river or lake that has overflown into urban environments.

The output of the ML approach will be a binary classification from a CNN model, classifying whether the image contains flooding or not. The model will have explainability techniques to describe important features when classifying flooding. Also, a layer of uncertainty is provided to show how confident the model is about its predictions. Due to the sensitive nature of the problem, it is important to provide these features for the model's trustworthiness.

In the end, I hope to answer the following questions:

- Can a CNN-based model classify flooding environments with high accuracy?
- Can the model provide explainable outputs, enabling stakeholders to interpret the rationale behind its predictions?

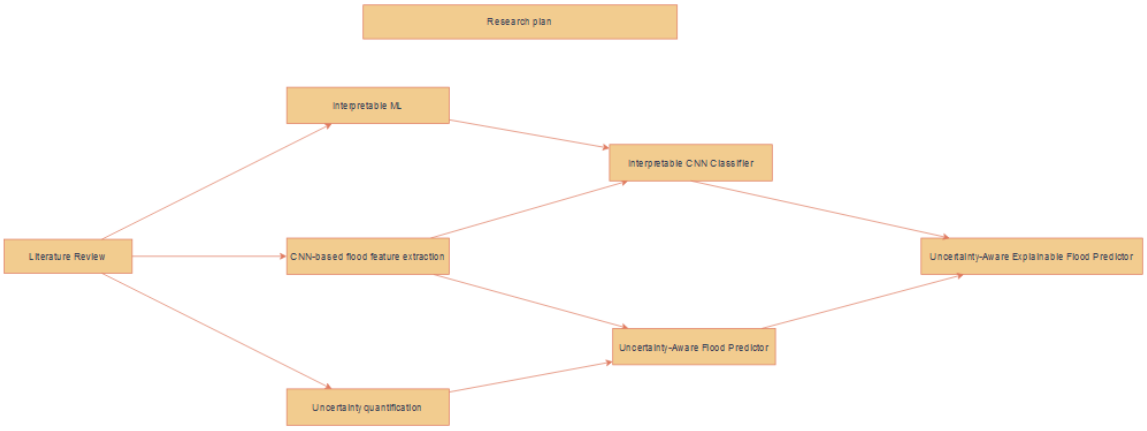

Figure 2: Research Plan

- Can the model provide confidence in its outputs, enabling stakeholders to trust its predictions?

The key challenges in this project are:

- Obtaining open-source, high-resolution, annotated satellite images is crucial but challenging due to limited availability and labeling complexity.
- Creating a model with high accuracy is challenging because there is a time-consuming process of evaluating and fine-tuning.
- Creating a safe/explainable model is challenging because the academic literature is relatively new in this application, providing little direction on good explainability and uncertainty techniques.

3 Technical Approach

As seen in Figure 2, my plan is to do a literature review on three topics to culminate into a single project. Interpretable ML, uncertainty quantification, and CNN-based flood feature extraction.

3.1 Dataset Pre-Processing

The dataset underwent normalization to standardize feature ranges, ensuring uniformity and improving model convergence. Additionally, Data augmentation was used to ensure the model had enough data to train on. Training involved splitting the dataset into training, validation, and testing subsets to ensure rigorous evaluation.

3.2 Model Selection

The final model was a ResNet-34 architecture with MC dropout which was implemented using Pytorch, which provided robust tools for streamlined model training. Model Training consisted of tracking metrics such as training loss, validation loss, and validation accuracy. Additionally, saving the best model allowed for long training without worrying about the model over-fitting.

3.3 Model Evaluations/Fine-Tuning

Hyperparameter optimization was carried out using Bayesian Optimization to fine-tune the model and achieve optimal performance. Bayesian Optimization uses a surrogate model, in this case, ResNet-18, to optimize the hyper parameters before deploying them on the real model. This is accomplished using Python package Optuna. The final paramaters were lr: 0.001, batchsize: 3, dropout: 0.5, optimizer: adam, along with a learning rate scheduler to decay the learning rate by exponential(-0.1) every 10 epochs.

The model was iteratively refined through evaluation on a validation set, using metrics such as accuracy, precision, recall, and F1-score. This iterative process ensured that the final model was robust and performed well across various evaluation criteria.

3.4 Model Trustworthiness

To improve model trustworthiness, Monte-Carlo Dropout was employed during inference to quantify the model's uncertainty. This approach helps identify ambiguous or high-risk areas in flood classification, enhancing decision-making.

Saliency maps were generated to understand the model's learned feature extraction. As seen in Figure 7, the map brightens features that the model deems as important.

4 Related Work

The following papers use SAR to classify flooding accurately or have a novel way of assessing the explainability of its model.

"Convolutional Neural Network-Based Deep Learning Approach for Automatic Flood Mapping Using NovaSAR-1 and Sentinel-1 Data" [1]. This model uses U-net architecture, particularly U-Net with pre-trained ResNet-152 backbone, a pixel-wise classification architecture. U-Net is noted for its ability to retain spatial features and generate more interpretable outputs, such as clear flood boundaries and detailed pixel-level classifications. While I was focusing on binary patch classification, I took note of the effectiveness of ResNet-152 as a backbone.

The paper "Flood Detection in Time Series of Optical and SAR Images" [3] introduces the SEN12-FLOOD dataset, which combines time series of Sentinel-1 Synthetic Aperture Radar (SAR) and Sentinel-2 optical imagery for flood event detection. The model is a ResNet-50 combined with a GRU for spatial and temporal analysis, taking multimodal Sentinel-1 SAR and Sentinel-2 optical time series images as input and outputting binary flood detection labels for each frame in the sequence. This paper was the main inspiration for the base model.

5 Evaluation Methodology

5.1 Dataset and Source

The dataset employed for this study is the SEN12-FLOOD Dataset [4], a curated collection of co-registered optical and Synthetic Aperture Radar (SAR) time-series images. It encompasses a wide geographic range, including regions in West and Southeast Africa, the Middle East, and Australia. Designed for flood detection, the dataset provides labeled data, including polarimetry channels and imaging coverage, which is valuable for binary classification tasks.

Challenges in Using the Dataset

- **Cloud and Atmospheric Interference:** Despite using SAR imagery capable of penetrating clouds, certain regions in the dataset exhibit noise or artifacts that complicate classification.
- **Limited Label Resolution:** Some samples have missing data that could possibly throw off the model's performance.

These problems with the dataset were addressed with image filtering using the color-variance and color-histogram to determine which images had little to no features.

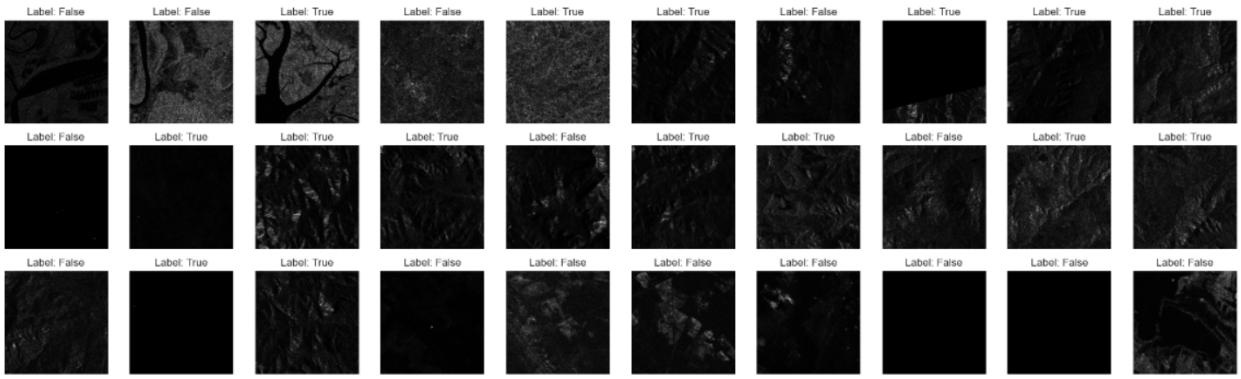

Figure 3: True labels of SAR imagery in SEN12-FLOOD dataset

5.2 Evaluation Metrics

5.3 Model Explainability

Saliency maps are useful tools when interpreting the model's results. A saliency map can show where the model looks in the image to understand it the best. It works well for flood detection because it highlights the most critical features in satellite imagery, such as water bodies or inundated areas, that influence the model's prediction. By focusing on the regions with significant gradients, it's easy to tell if the model understands how to look for floods the way that a human would. This is useful for disaster response specialists to confirm a model's trustworthiness and reasoning in its prediction.

5.4 Model Uncertainty

5.4.1 Model Uncertainty in Machine Learning

Model uncertainty refers to the degree of confidence a model has in its predictions. It can be broadly categorized into two types: aleatoric uncertainty, which arises from inherent noise in the data, and epistemic uncertainty, which reflects uncertainty in the model due to insufficient training data or model complexity. In the context of deep learning, particularly Convolutional Neural Networks (CNNs), various methods have been developed to estimate these uncertainties. Bayesian Neural Networks (BNNs) are a prominent example, incorporating distributions that are overweight to capture epistemic uncertainty. Techniques like Monte Carlo (MC) Dropout and Deep Ensembles are also widely used, offering scalable and practical alternatives to full Bayesian approaches. These methods provide insights into both the model's confidence and areas where additional data could improve performance. In this case, the model uses MC Dropout to provide a quantitative uncertainty about the predictions.

To evaluate the quantitative performance of the ML approach and address the questions investigated, the following metrics were employed:

- **Accuracy:** Measures the proportion of correctly predicted instances across all classes. This is good for understanding the model performance but must be complemented with other metrics in imbalanced scenarios.
- **Precision, Recall, and F1-Score:** These metrics were used to evaluate the model's performance on each class

6 Results and Discussion

The results of this project are presented and discussed in a structured manner, highlighting what worked, what did not work, and the reasoning behind these outcomes. Each step of the process is explained with reference to the corresponding models, methodologies, and challenges encountered.

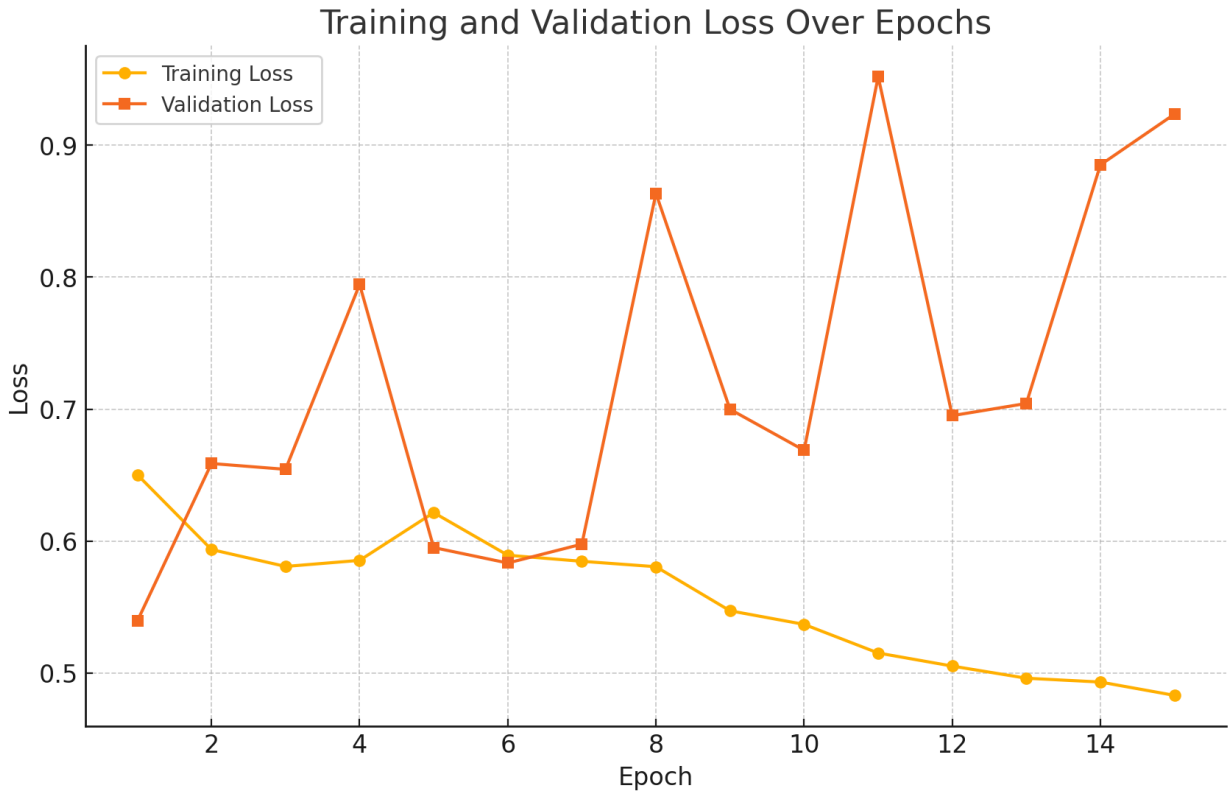

Figure 4: Resnet-18 loss with MC dropout

Figure 5: Resnet-34 loss with MC dropout

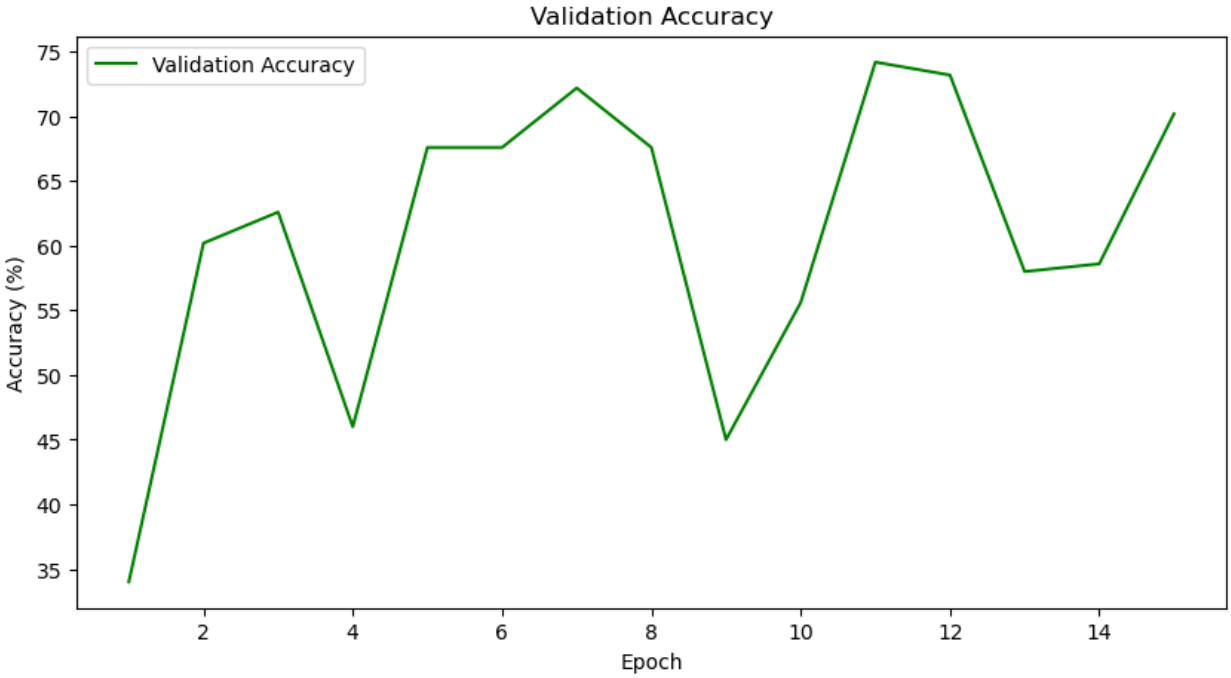

Figure 6: Resnet-34 accuracy with MC dropout

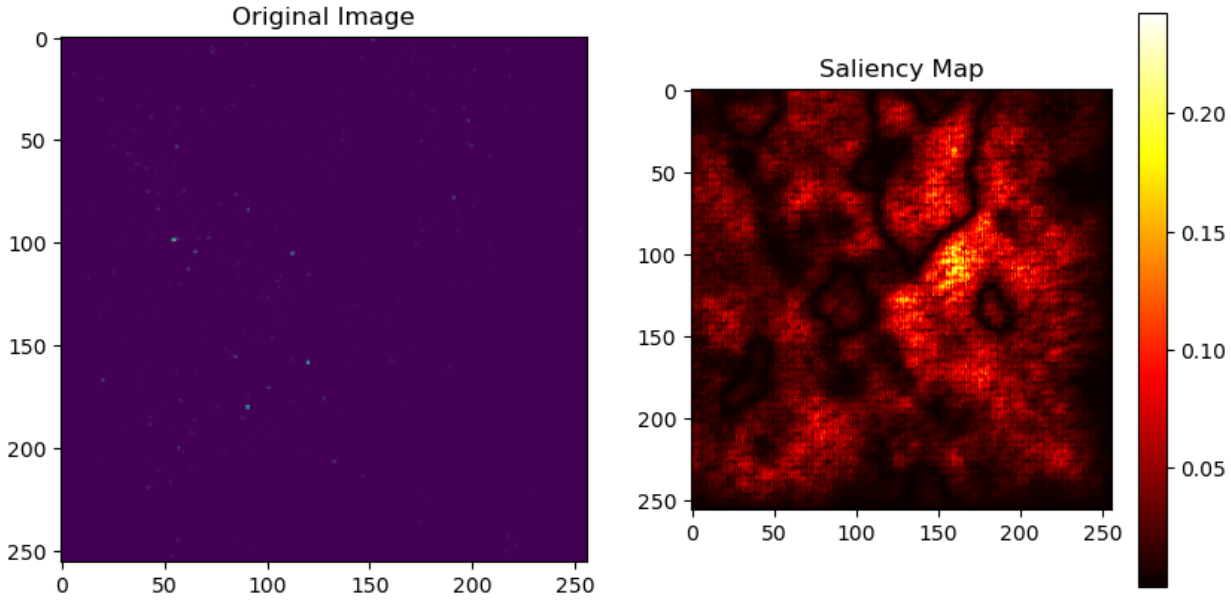

Figure 7: Resnet-34 Salience Example 1

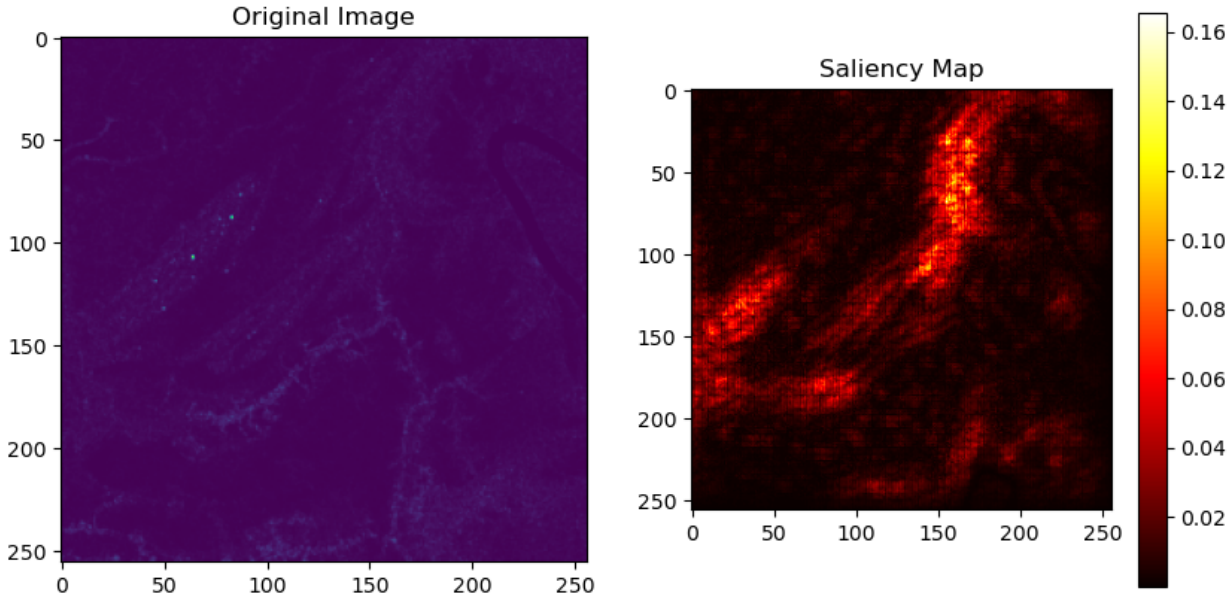

Figure 8: Resnet-34 Salience Example 2

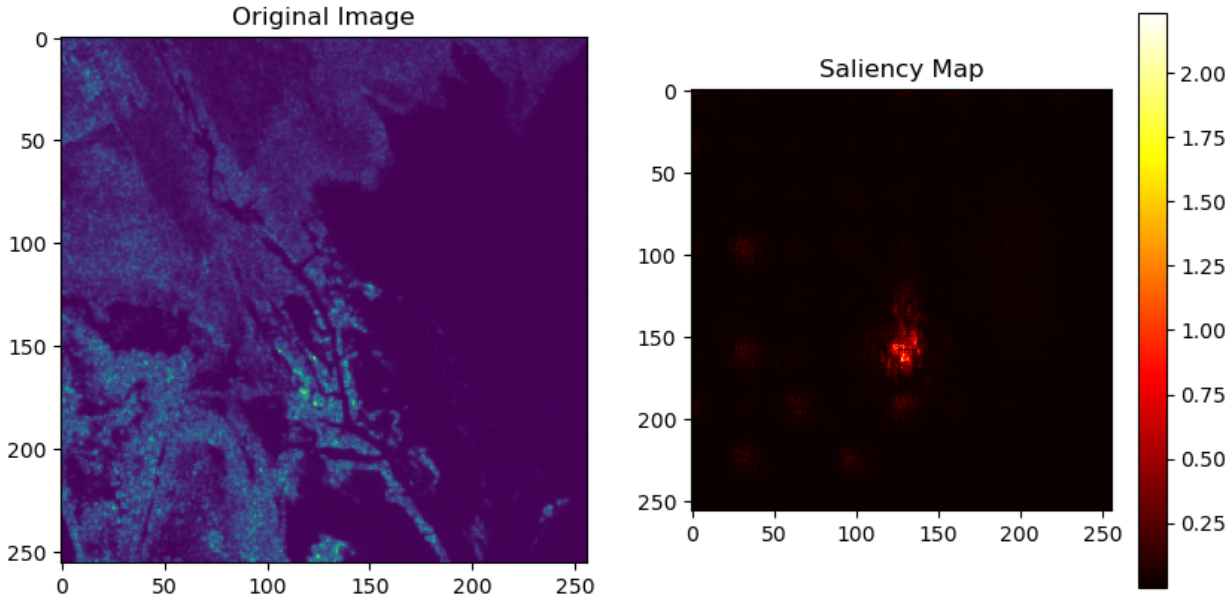

Figure 9: Resnet-18 Salience Example 1

Table 1: Results of Experiments with Different ResNet Models

Model	Accuracy (%)	Precision	Recall	F1-Score
ResNet-152	51.23	0.4512	0.8510	0.5912
ResNet-34	28.23	0.2314	0.7321	0.3518
ResNet-50	55.20	0.3842	0.9577	0.5484

Table 2: Results of ResNet Models with Monte-Carlo Dropout

Model	Accuracy (%)	Average Uncertainty	Precision	Recall	F1 Score
ResNet-18	80.8	0.0455	0.6346	0.6972	0.6644
ResNet-34	75.8	0.0230	0.6346	0.6972	0.6644

6.1 Dataset

I initially started this project with the "SN8: Flood Detection Challenge Using Multiclass Segmentation" [5]. At the same time, this dataset was clean and had been established in the literature. It was not exactly what I was looking for. I intended to find a way to apply machine learning to emergency response and not post-emergency management. Due to the applicability of this dataset and the complexity of the task of multiclass classification, I settled for a simpler dataset.

I used the SEN12-FLOOD Dataset [4] for this project. This dataset comprises 336 sequences of co-registered optical and Synthetic Aperture Radar (SAR), providing a unique resource for advancing machine learning applications in flood classification.

6.2 Cleaning the Dataset

Many images in this dataset were either blank or did not provide many features. I was able to learn this after looking into the data myself. Using a color-variance analysis on the dataset seemed to work decent, as many of the image features had high variance. Using a color histogram worked decently, as many of the images proved to have noise that the histogram would leave in.

6.3 Baseline Model Results

The initial experiment utilized a ResNet-152 model due to previous experiments showing that it could work for flooding SAR images. The model was trained for five epochs to get a baseline, yielding the following results as shown in Table 1: Accuracy: 44.23%, Precision: 0.3393, Recall: 1.0000, F1-Score: 0.5066.

Why It Didn't Work: The baseline model was able to correctly classify all flood images (100% recall), but it over-predicted the presence of floods, leading to many false positives. This result highlights the model's sensitivity to flooding features but also indicates its lack of specificity. The limited training epochs and the large number of parameters in the model likely caused its low accuracy. A model too large can easily overfit the training set's features.

6.4 Data Augmentation and Extended Training

Next, data augmentation techniques such as flipping, rotation, and scaling were applied to diversify the training dataset. The training was extended to 10 epochs to allow the model to learn more robust features. Results after this stage were: Accuracy: 45.88%, Precision: 0.3390, Recall: 0.9372, F1-Score: 0.4979 as seen in Table 1.

Why It Didn't Work: Despite extended training and augmented data, the model still struggled with overfitting. The saliency map showed the model focused on irrelevant regions, suggesting inadequate feature learning.

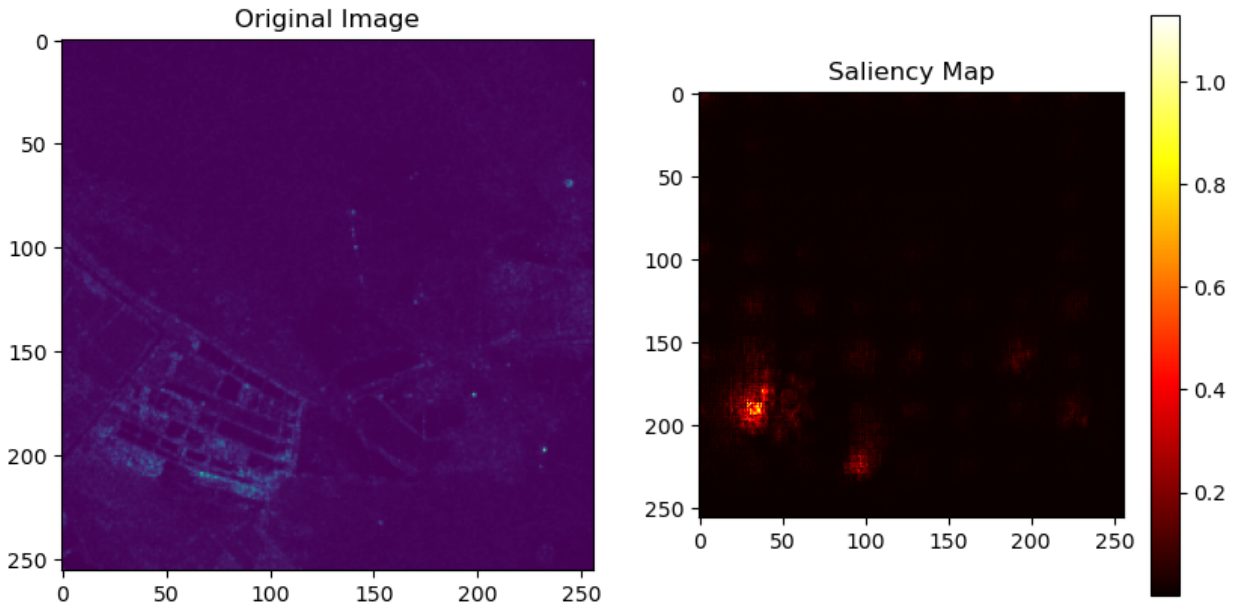

Figure 10: Resnet-18 Salience Example 2

6.5 Surrogate Model for Hyperparameter Optimization

Given the computational expense of training ResNet-152, a surrogate ResNet-18 model was used for Bayesian optimization via Optuna. Key hyperparameters such as learning rate, batch size, and optimizer type were tuned, yielding the following optimal settings:

- Learning Rate: 9.38×10^{-5}
- Batch Size: 16
- Optimizer: Adam

These settings were applied to ResNet-152, resulting in improved performance: Accuracy: 51.23%, Precision: 0.4512, Recall: 0.8510, F1-Score: 0.5912

Why It Worked: Bayesian optimization allowed efficient hyperparameter tuning, helping the model converge better and achieve a balanced trade-off between precision and recall.

Why It Didn't Work: The surrogate model (ResNet-18) introduced limitations due to its shallower architecture, leading to suboptimal hyperparameters for the deeper ResNet-152 model. Additionally, the complexity of ResNet-152 likely led to overfitting, given the relatively small dataset size.

6.6 ResNet-50 Experiment

To explore whether a simpler architecture could improve generalization, I tried ResNet-50. This model worked the best out of any of my previous attempts with the following metrics: Accuracy: 55.2%, Precision: 0.3842, Recall: 0.9577, F1 Score: 0.5484.

Why It Worked: The ResNet-50 was able to get a better accuracy than my previous attempts by existing in between my previous attempts of ResNet-18 and ResNet-152. It also worked well with the hyper-parameters defined by the surrogate model due to being relatively much closer in complexity.

Why It Didn't Work: The model is clearly still not improving its validation accuracy, still proving to over-fit the data.

6.7 Resnet-18 with Monte-Carlo Dropout

Once I had gotten the best baseline model, I attempted to introduce Monte-Carlo Dropout. This allowed me to improve the model's accuracy and quantify its uncertainty. My results are as follows: Accuracy: 80.8%, Average uncertainty: 0.0455, Precision: 0.6346, Recall: 0.6972, F1 Score: 0.6644 as seen in Table 2.

While the accuracy showed the model was exceptional, the explainability proved the model was clearly lacking, as seen in Figure 9 and Figure 10, where the salience map shows gradient on little to no important features.

6.8 Resnet-34 with Mone-Carlo Dropout

The next step was to test a slightly larger model with the same parameters as before. This resulted in slightly lower accuracy while showing better confidence in its predictions. This was corroborated by the salience map in Figure 7 and Figure 8, showing that it was able to learn distinct features in the images. The training loss history is seen in Figure 5 and the validation accuracy in Figure 6. The quantitative results are as follows: Accuracy: 75.8Average uncertainty: 0.0230, Precision: 0.6346, Recall: 0.6972, F1 Score: 0.6644, as seen in 2

The lesson here is that even though Resnet-18 recieved a much higher accuracy, given the model uncertainty and the salience map, its clear that Resnet-34 out-performs through its reliability and trustworthiness.

6.9 Explainability

The best-performing model (ResNet-50) was further analyzed using saliency maps. These maps highlighted critical areas in the satellite imagery, such as water bodies and urban regions, confirming the model's focus on flood-relevant features. However, certain irrelevant areas (e.g., vegetation) were occasionally emphasized, suggesting areas for further improvement.

Why It Worked: Explainability techniques validated the model's reasoning towards features in each image. I was able to look at images and ensure the model understood important features.

6.10 Summary of Findings

- **What Worked:**
  - Deep architectures like ResNet-50 effectively captured features indicative of flooding while also hallucinating other features not indicative of flooding due to messy data.
  - Bayesian optimization and data augmentation improved performance through a surrogate model.
  - Saliency maps improved the trustworthiness of the model through interpretability.
- **What Didn't Work:**
  - ResNet-152 over-fit to the data due to a high number of parameters.
  - ResNet-34 lacked the capacity for this complex task.
  - The surrogate model approach for hyperparameter tuning had limitations when transferring to much larger models.

These findings underscore the need for high-quality datasets and appropriate model complexity for flood prediction tasks. Future work will focus on addressing these challenges to further enhance model performance and safety through uncertainty quantification.

I wanted to introduce uncertainty quantification along side explainability. Unfortunately, ensuring good performance on the dataset was more important, and with unforeseen complications, the model took longer to be functional than expected.

7 Lessons Learned

Start with a simpler task. Starting with a task such as building a ResNet-152 model for regression analysis of flood risk ended up being an extremely more difficult task than initially thought and wasted a lot of time. Working on a simpler task that can be further scaled up proved to be a much better strategy. If I were to start over, I would start with a reduced scope.

Due to difficulties finding a appropriately sized model, I was unable to expand on explainability and uncertainty techniques. If I were to start over, I would focus more on comparing uncertainty techniques such as Bayesian Neural Networks, Gaussian Processes, Laplace Approximation, etc. I would have also tried comparing the salience maps with Grad-cam maps.

I discovered that Bayesian Optimization is not guaranteed to work, especially for a much smaller surrogate model such as ResNet-18.

References

[1] Ogbaje Andrew, Armando Apan, Dev Raj Paudyal, and Kithsiri Perera. Convolutional neural network-based deep learning approach for automatic flood mapping using novasar-1 and sentinel-1 data. *ISPRS International Journal of Geo-Information*, 12(5), 2023.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[3] C. Rambour, N. Audebert, E. Koeniguer, B. Le Saux, M. Crucianu, and M. Datcu. Flood detection in time series of optical and sar images. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLIII-B2-2020:1343–1346, 2020.

[4] Clément Rambour, Nicolas Audebert, Elise Koeniguer, Bertrand Le Saux, Michel Crucianu, and Mihai Datcu. Sen12-flood : a sar and multispectral dataset for flood detection, 2020.

[5] SpaceNet. Spacenet dataset. https://registry.opendata.aws/spacenet/, n.d. Accessed: 2024-12-12.