Reporting Deep Learning Experimentation: Group 12

Krzysztof Wiesniakowski s2124849
Chigozie Ifepe s2109365
Pedro Villadangos Benavides s1930079

Abstract

Earth Observation (EO) has emerged as a crucial tool for understanding various aspects of the Earth's surface and monitoring human activities. With the exponential growth in data availability, Artificial Intelligence (AI) techniques have proven instrumental in analyzing remote sensing data effectively. In particular, Synthetic Aperture Radar (SAR) imagery offers a valuable alternative to optical imagery, especially in scenarios characterized by cloud cover, such as weather-related disasters. However, the application of machine learning on SAR data remains challenging due to limited labeled datasets. This paper aims to contribute to this evolving field by leveraging the dataset introduced by Rambour et al. (2020). Despite encountering limitations in processing the entire dataset, our experimentation yielded promising results, achieving an accuracy rate of nearly 66%. These findings underscore the potential of convolutional neural networks (CNNs) and image analysis techniques in SAR data interpretation. As the importance of accurate predictions in the face of climate change becomes increasingly evident, further exploration and refinement of these techniques are warranted. This study outlines future research directions to advance the field and address existing challenges effectively.

Introduction

Over the past few decades, Earth Observation has provided valuable insights across geosciences and human activity monitoring. With the increasing availability of data, Artificial Intelligence (AI) techniques have proven highly effective in interpreting remote sensing data. Furthermore, alternative acquisition methods such as Synthetic Aperture Radar (SAR) offer solutions to challenges not adequately addressed by optical imagery alone, especially in scenarios like weather-related disasters where cloud cover is prevalent. Despite the promise, machine learning on Synthetic Aperture Radar (SAR) data remains challenging due to limited labeled datasets. Nevertheless, there is more research done in the field of Synthetic Aperture Radar (SAR) images and machine learning and this paper wants to contribute in this field. This project build upon utilizes the dataset presented by Rambour et. al. (2020) (8)

Background

Rambour et al. (2020) (8) introduced a novel dataset comprising Sentinel 1 sequences that cover the same areas and timeframes as the city-centered satellite sequences provided by the MediaEval 2019 Multimedia Satellite task Bischke et al. (2019) (3). These city-centered sequences grant access to series of multispectral Sentinel 2 images. Because Synthetic Aperture Radar (SAR) is unaffected by cloud cover, the authors were able to collect approximately twice as many SAR images as optical images for the same time period. To leverage both SAR and optical data effectively, they merged the MediaEval dataset with their own, resulting in the creation of the SEN12-FLOOD dataset. This SEN12-FLOOD dataset is going to be a data source for the model used in this research.

Dataset description

The dataset comprises 412 time series, each containing between 4 to 20 optical images and 10 to 58 SAR images. On average, there are 9 optical and 14 SAR images per sequence. The acquisition period spans from December 2018 to May 2019. Notably, a flood event is detected in 40% of the optical Sentinel-2 images and 47% of the SAR Sentinel-1 images.

Since the notation of the images will be used often throughout the report, below we present their short notation:
- S1 = Synthetic Aperture Radar (SAR) Sentinel-1 images
- S2 = optical Sentinel-2 images

Each image is accompanied by a binary label indicating the presence or absence of a flood event within the observed region. For one place there is a sequence of the images so it can be observed how the flooding event develops. These labels originate from the original MediaEval 2019 dataset and were sourced from the Copernicus Emergency Management Service (1). The Sentinel-1 images were acquired from the Scientific ESA hub website (2).

Neural networks in time-series and image classifications

Authors of the dataset which is utilized in this project, not only did present the novel dataset but also performed the experiments using state-of-the-art ResNet-50 network for a flood detection task on each image. These models performed

relatively well and the authors managed to achieve over 75% accuracy on that data.

In the world of deep learning and computer vision, Convolutional Neural Networks (CNNs) show great results. According to Mishra M, (6) Convolutional Neural Networks, often abbreviated as CNN or ConvNet, are a type of neural network designed to handle data with a grid-like structure, such as images. An image, in digital form, is essentially a binary representation of visual information. It comprises pixels organized in a grid-like pattern, where each pixel holds values representing its brightness and color.

When it comes to research in utilizing CNN for image recognition Zhao et al. (2017), (10) showed that Convolutional Neural Networks (CNNs) can automatically discover important patterns in time series data using special operations like convolution and pooling. Authors tested this idea using two types of data: simulations and real-world data from different areas like medicine or finance. Their experiments showed that our method is better than others at classifying time series data. It is more accurate and can handle noisy data better.

When working with images, CNN seems to be a powerful architecture, nevertheless, in this research, we work with images in time-series. Therefore, it is necessary to hold information from the previous images and pass it to the next ones.

For that, it is worth looking at ConvLSTM. As stated by Xavier on Medium Platform (9) ConvLSTM (Convolutional LSTM) is a variation of the Long Short-Term Memory (LSTM) network, which integrates convolutional operations within the LSTM architecture. While traditional LSTMs excel at capturing temporal dependencies in sequential data like time series or text, ConvLSTMs extend this capability to data with spatial structures, such as images or videos.

Livieris et. al. (5) proposed the model which capitalizes on the capabilities of convolutional layers to extract valuable insights and comprehend the underlying structure of time-series data, alongside harnessing the efficacy of long short-term memory (LSTM) layers to capture both short-term and long-term dependencies. Through a series of experiments on the dataset of the daily gold prices in USD from Jan 2014 to Apr 2018, they assessed the performance of the model compared to contemporary deep learning and machine learning approaches. Initial experimental results indicate that incorporating LSTM layers alongside additional convolutional layers leads to notable enhancements in forecasting accuracy.

Lazorenko, D (4) in her research assessed the performance of a Convolutional Long Short-Term Memory (ConvLSTM) model architecture learning spatial and temporal patterns from a sequence of satellite images for predicting cloud coverage. Upon training the machine learning model on a dataset covering a two-year period, it achieved an outstanding accuracy of 90 percent.

Therefore, it can be concluded that ConvLSTMs can leverage the strengths of both CNNs and LSTMs: CNNs for spatial feature extraction and LSTMs for capturing temporal dependencies. As a result, ConvLSTMs are well-suited for tasks that require understanding both spatial and temporal aspects of the data, such as video analysis or spatiotemporal forecasting.

Our study aims to see how well Convolutional Neural Networks (CNNs) can handle two kinds of satellite images: ones from SAR Sentinel-1 and ones from optical Sentinel-2. By using CNNs, we want to figure out how efficient they are at finding important details in both types of images. This research will help us understand how CNNs can be feasible for analyzing SAR and optical images from satellites.

Study Setup and Definition

Our research is driven by a keen interest in earth observation. However, accessing freely available satellite image datasets presents a significant challenge. Rambour et al. (2020) (8) made a substantial contribution to the field by sharing their research findings and providing a well-documented dataset for public use, which greatly aids our work.

Novel dataset introduced by Rambour et al. 2020

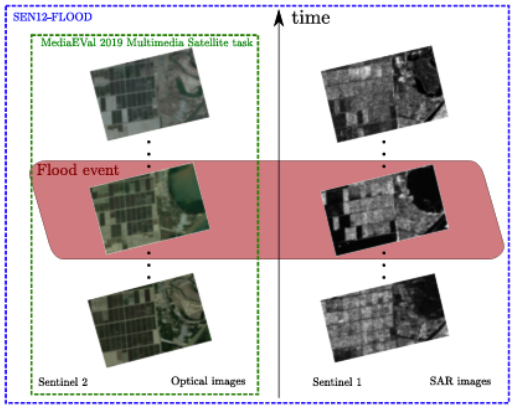

Figure 1: The SEN12-FLOOD dataset is composed of SAR and optical time series in which a flood event may occur. Picture source: from C. Rambour et al. 2020 (8)

We came across the work of C. Rambour et al., wherein they introduced a new dataset comprising SAR 1 images paired with SAR 2 images. Moreover, they conducted deep learning experiments utilizing ResNet models. This served as a valuable foundation for our own deep learning experiments.

Technical reproducibility of the experiment

The results presented in this document were obtained by executing the code available in the GitHub repository that is attached to that submission within the Azure Machine Learning environment.

The repository contains the following files: deeplearningflooding.ipynb (a Python notebook), environment.yaml (an environment specification file), and ReadMe.md (which provides detailed instructions for reproducing the results).

To set up the Azure environment for running the code, it is necessary to create an Azure workspace. Within the notebooks/files directory, the following three elements should be

placed: the Python notebook, the environment specification, and the complete SEN12FLOOD dataset.

To execute the final notebook, a compute instance must be created. Given the constraints on computing power and storage for basic Azure accounts, the chosen instance type for running the code was the Standard-E16s-v3 (16 cores, 128 GB RAM, 256 GB disk). This is the largest machine available under our student accounts. Due to time and computing power constraints, the experiment was conducted on a all the readable scenarios in the dataset. For training the model on additional scenarios, it is advisable to use a more powerful machine to reduce compute times and achieve a more generalized model, with the possibility of training it over more epochs or test different model architectures.

In the initial cells of the Python notebook, the name of the selected machine must be correctly formatted to appropriately apply the environment and relevant Python dependencies. It is also noteworthy that the plain Python 3.8 AzureML kernel was used. Hard-coded directories for accessed image data and its respective metadata is worth a consideration.

DL Modelling Technique Overview

Data preprocessing

The SEN12FLOOD dataset comprises metadata associated with 421 time-series sequences, and these are detailed in the files "S1list.json" and "S2list.json". These files provide crucial contextual information, including the date when each satellite image was captured and a binary indicator (the "flooding label") signifying the presence of flooding in the sequence.

Each S1 and S2 image was processed using the Python TIFF library (7) However, due to the extensive size of certain images, issues with the TIFF library necessitated the removal of a limited number of scenarios from the set. Key information relevant to the model (such as vectorized images, dates, flooding labels, and scenario labels) was organized into a dictionary to maintain consistency and manageability for the CNN. Given the varying sizes of the images throughout the dataset, they were resized to a 522×544 (height $\times$ width) dimension. Moreover, the images were sorted by date within each scenario to account for the time-series nature of the data.

To align with the required input dimensions of the convolutional network, additional dimensions were added to the S1 and S2 images. The desired shape for the model was set as follows: Dim(a, b, c, d, e). The parameter 'a' represents the batch size, which, as discussed in the subsequent section, corresponds to a batch size of 1 for this study, as the model is trained on a scenario-by-scenario basis using a for-loop, learning about new flooding sequences with each iteration. The parameter 'b' denotes the time-series parameter, indicating the number of images belonging to the time series within a batch. For example, if a scenario contains 10 S1 images, the batch would start at the first analyzed image and end after 10 images. This dimension is crucial as it retains the temporal nature of our data. The parameters 'c' and 'd' correspond to the height and width of the images, respectively, which are fixed at 522×544 as previously mentioned. The last parameter 'e' represents the number of layers in the image. For the S1 SAR images, the input dimension has been set to 1, focusing exclusively on the "VH" (Horizontal) polarized images. An alternative approach would be to include a second dimension representing the "VV" (Vertical) polarized component. In the case of the multispectral images, the twelve channels, each corresponding to different wavelengths, have been combined to form a three-channel RGB image, resulting in an input dimension of 3. Similarly, the flooding label, which serves as the 'y' parameter in the input, is structured to match the dimensionality of the input 'x'. An extra dimension 'a' is added to the length of the array containing the flooding labels to indicate that the batch size is also 1. The other dimensional parameter 'b' corresponds to the length of the boolean array, hence 'y' has the dimensionality (a, b). This ensures that for each scenario of images ingested by the model, there is a corresponding array of flooding labels.

As highlighted in the original research on the dataset, the combined utilization of S1 and S2 images is motivated by the distinctive insights each type can provide. Specifically, synthetic aperture radar (SAR) images (S1) are independent of weather conditions, in contrast to optical images (S2). As a result, both image types are stored in separate data structures to be fed into and processed by the model in separate batches. The final layers of the model then concatenate the features learned from both image types.

However, a challenge arises due to the varying lengths of the S1 and S2 images across scenarios, which inhibits direct concatenation. To address this, padding is utilized to align the dimensions. Padding is a technique where sequences of images are extended with extra values, typically zeros, to ensure that all sequences have the same length. This allows the model to process all the sequences uniformly.

Following this, another issue arises with the length of the flooding labels "y". These labels are also padded and extended to match the input dimensions. This ensures that the length of the flooding labels corresponds with the number of images in the sequence, maintaining consistency with the input dimensions.

Being the the exhaustive data processing part done and having the correct formatted input is time to decide which architecture and how are we going to train the model.

Model specificatitions

It has been decided that the best approach to train the model is using the Keras train on batch pre-build function. This approach allows us to feed the model with each of the scenarios sequentially. By iterating through all the scenarios this way the model learns how the flooding sequence develops. A pivotal point in our stud us time-series awareness is key to the development of a basic model.

To grasp the process of learning the model outer the loop to train on batch on the different training scenarios another loop to run different epochs has been implemented. By running different epochs we can see how the model updates its loss function and improves in accuracy with each iteration as we can see in Figure 3 in the results section. It is worth to noted how the values of the loss function did not converge

yet. Which means that more epochs would make the model learn even better at the features of the training set.

The chosen architecture for the model has been influenced significantly by the persistent issues of overfitting and gradient explosions encountered in numerous preliminary runs. As a result, a simplified and less complex architecture was adopted to achieve a functional model.

Due to the distinct characteristics of S1 and S2 images, they are processed through separate branches and subsequently concatenated. These branches each consist of two layers of ConvLSTM2D, each with two neurons. As previously discussed, Convolutional LSTM neural networks are adept at extracting spatial features and capturing temporal dependencies, making them suitable for this experiment.

Batch normalization is applied after the first ConvLSTM2D layer to stabilize the learning process. Dropout and L2 regularization are employed after each layer to mitigate overfitting, particularly in the non-flooding class. In addition, masking is incorporated to correct the padding introduced earlier. Padding is used to ensure that all input sequences have the same length, which is crucial when working with sequences of varying lengths, such as time-series satellite images. However, the padding introduces unnecessary information that can interfere with learning, so Masking is used to instruct the model to ignore these padded values.

Once the model learns about the features of both types of images, the features learned are concatenated, and after that, a Dense layer with two neurons is used in conjunction with a TimeDistributed wrapper, enabling operations across the time step length of each batch during training and testing. The model concludes with a sigmoid-activated TimeDistributed output, which is appropriate for the binary classification task of predicting flooding events.

Following the precedent set by the dataset's authors, Stochastic Gradient Descent was chosen as the optimization algorithm, with a low learning rate of 7x10-7 to prevent overfitting and gradient explosions. Gradient clipping was also implemented to further prevent gradient explosions.

The model architecture can be observed in the Figure 2.

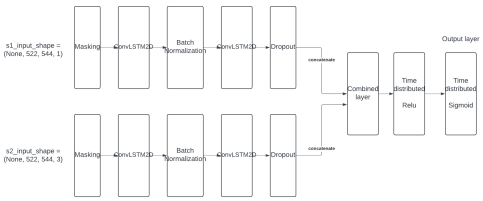

Figure 2: Model architecture

The selected loss function is binary cross-entropy, suitable for binary classification problems. The weights for the binary cross-entropy loss function are adjusted based on the class distribution of flooding and non-flooding events.

Train, Test, and Validation approach

The validation approach involves a popular train-test-validation split. Being the training set 3X times the size of the validation set. The model's performance is evaluated on unseen data, with metrics such as accuracy and loss tracked across multiple epochs to monitor the learning process and identify potential issues like overfitting.

Classification Metrics

For evaluating the model metrics for classification problem are going to be presented.

Let:

$$\begin{aligned} \text{TP} &= \text{True Positive} \\ \text{TN} &= \text{True Negative} \\ \text{FP} &= \text{False Positive} \\ \text{FN} &= \text{False Negative} \end{aligned}$$

Accuracy (ACC)

$$\text{ACC} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

Precision (P)

$$\text{P} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Recall (R) or Sensitivity or True Positive Rate (TPR)

$$\text{R} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

F1 Score

$$\text{F1} = 2 \times \frac{\text{P} \times \text{R}}{\text{P} + \text{R}}$$

Results

Below it could be observed how the training and validation loss function behaves during the training process during the training of the model for 10 epochs with a learning rate of 7X10-7

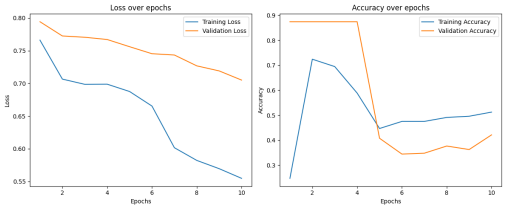

Figure 3: Train and validation loss together with accuracy

It could be observed that the loss function drops during the training process for both the training and validation sets, indicating that the model is effectively learning the features and patterns present in the data. This reduction in loss suggests that the model is converging towards an optimal solution, as expected in a well-trained model. However, it is important to note that while the loss decreases, it may not necessarily guarantee an improvement in accuracy, as the

loss function represents a measure of how well the model is performing internally, rather than its performance on unseen data.

When analyzing the accuracy during the training process, significant fluctuations were observed. These fluctuations may be attributed to various factors, such as the complexity of the dataset, the learning rate of the optimizer, and the presence of outliers or noisy data points. Additionally, the fluctuations in accuracy could also indicate instances of overfitting or underfitting, where the model either learns to memorize the training data too closely or fails to capture the underlying patterns effectively. Therefore, while fluctuations in accuracy are common during the training process, careful monitoring and tuning of hyperparameters are necessary to ensure the stability and generalization performance of the model.

After running the above-mentioned model with the following specifications to predict for the test set we obtained an average accuracy of: 0.6590

Exact metrics that were obtained on the test set are presented below in Figure 3.

```
Average Metrics:
Average Accuracy: 0.6590
Average Precision: 0.0553
Average Recall: 0.0772
Average F1 Score: 0.0550
```

Figure 4: Metrics obtained on the test set

Conclusion

In conclusion, while our experimentation yielded relatively favorable outcomes, we encountered limitations in fully processing the entire dataset. Despite this challenge, we achieved commendable results, achieving an accuracy rate nearing 66%. The promising performance underscores the potential of this technique for future exploration. Notably, convolutional neural networks (CNNs) and image analysis boast a robust academic foundation, warranting continued investigation into their application. As the implications of climate change intensify, the need for more precise predictions grows increasingly vital. Looking ahead, our findings suggest avenues for future research, as outlined in the forthcoming sections on future work.

Limitations

The study's limitations include potential overfitting due to the model's complexity and the varied nature of the scenarios in the dataset. Additionally, the replication problem statement revolves around the challenge of developing a model that can generalize across diverse flooding events, leveraging the strengths of both radar and optical imagery.

The acquisition of trained models on larger datasets and the employment of more intricate neural network structures were constrained by the limitations and associated costs of the Azure Machine Learning environment. The inability to utilize the majority of the dataset significantly impeded the development of enhanced models based on the breadth of our dataset.

Our study was hindered by limitations associated with the Tiff library, particularly its inability to accurately read all image files. This constraint posed challenges in our data processing pipeline, impacting the comprehensiveness of our analysis.

Future Work

Moving forward, there are a couple of areas we could focus on for future work. Firstly, investing in more powerful virtual machines (VMs) would allow us to work with the entire dataset without running into limitations. This means we could analyze more data and get a better understanding of our results. Additionally, we could consider running more experiments with longer training times (more epochs). This could help us explore how our models improve over time and potentially increase their accuracy. By addressing these points, we can strengthen the reliability and depth of our findings in deep learning experimentation.

References

[1] Copernicus emergency management service. Retrieved from https://emergency.copernicus.eu/.

[2] Open access hub. Retrieved from https://scihub.copernicus.eu/.

[3] Benjamin Bischke, Patrick Helber, Christoph Schulze, Vishal Srinivasan, Andreas Dengel, and Damian Borth. The multimedia satellite task at mediaeval 2019. In *Proc. of the MediaEval 2019 Workshop*, 2019.

[4] D. Lazorenko. Satellite nowcasting of cloud coverage via machine learning, February 2023.

[5] Ioannis E. Livieris, Eleni Pintelas, and Panagiotis Pintelas. A CNN-LSTM model for gold price time-series forecasting. *Neural Computing and Applications*, 32:17351–17360, 2020.

[6] Manisha Mishra. Convolutional neural networks, explained - towards data science. *Medium*, December 2021.

[7] PyPI. tifffile. https://pypi.org/project/tifffile/, May 2024. Accessed on 4th May, 2024.

[8] 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.

[9] Alex Xavier. An introduction to convlstm - neuronio. *Medium*, December 2021.

[10] Bendong Zhao, Huanzhang Lu, Shangfeng Chen, Junliang Liu, and Dongya Wu. Convolutional neural networks for time series classification. *Journal of Systems Engineering and Electronics*, 28(1):162–169, 2017.