Machine Learning based Ocean Eddy Detection using Cloud Services

Seraj AM Mostafa[1], Jinbo Wang[2], Jianwu Wang[1], Sanjay Purushotham[1]
{s172, jianwu, psanjay}@umbc.edu, jinbo.wang@jpl.nasa.gov
[1] Information Systems Department, UMBC
[2]JPL, NASA

Abstract: The main objective of this study is to make use of cloud services, such as amazon web services (AWS) to build machine learning models to identify ocean eddies from satellite images. There are two major approaches to conduct machine learning at AWS. The first approach is via AWS SageMaker, which is a machine learning platform that facilitates image labeling, jupyter notebooks, and various algorithms for model training. The second approach is via AWS EC2, which provides virtual instances as execution environments. Beside SageMaker and EC2, we also configured Google cloud as an alternative service to train our models to identify ocean eddies from satellite images. We chose a three layer CNN (convolutional neural network) model for binary image classification to see whether the images contain eddies or not, and YOLOv3 for eddy detection and localization (where the eddy is located if there are eddies in images). In terms of accuracy, our CNN model achieved 60% accuracy. Using the YOLOv3, the intersection over union (IoU) ranged from 40% to 90%. Comparing features we found that SageMaker is equipped with more functionalities compared to Google cloud platform (GCP), however, the services are not easy to merge to build end-to-end pipelines. GCP offers GPU (graphics processing unit) based services by default with TPU (Tensor Processing Unit) and CPU (Central Processing Unit) as well but in AWS the GPU base services need to be configured beforehand. In this work, we configured the scripts using Horovod to deploy on GPU based EC2 instances. Codes are open sourced for further references at https://github.com/big-data-lab-umbc/AWS-automation/tree/main/gpu-example/OceanEddy.
Between SageMaker and EC2, we think EC2 is a general computing resource and provides a lot of freedom for users to decide on what to do such as installing any software/packages, running Docker images, running Jupyter notebooks, utilizing multiple EC2 instances for parallel execution. Meanwhile, most of these capabilities need manual command line operations from users, which could be difficult for users who are not frequent command line users.

Introduction: This study uses cloud services to build Convolutional Neural Network (CNN) deep learning models to identify ocean eddies from SAR satellite images. Our first approach is to build a CNN for binary image classification. In this approach we identified whether the image is an eddy containing image or not. The second approach is to identify the eddy within a given image. To do so, we used YOLOv3 as an object identification and localization method which is able to identify an eddy and its location within that image. In both cases we used amazon web services (AWS) and Google as cloud platforms and their services such as, AWS SageMaker, S3 bucket, EC2, Google Colab and Google drive respectively which is discussed in the following section.

The main goal throughout this research was to identify how convenient the new tool (e.g., AWS SageMaker) and existing (e.g., Google cloud) tools in terms of their usability to build pipelines for such kinds of analysis (ocean eddy detection) to understand climate change. The AWS has got more features in terms of building machine learning (ML) pipelines, for example, it allows users to label images from AWS S3 storage, to choose ML algorithms to train models and store them for future use. It also provides GPUs to its user with additional cost. Moreover, SageMaker supports Edge services which are highly scalable and faster in processing. In our experiment, we utilized both GPU and CPU based SageMaker nodes incorporating S3, also single and multicore GPU based EC2 instances. GCP, on the other hand, is more convenient and free to use, though users can purchase high configuration GPUs for additional cost. The main problem with Google cloud we faced is 'time out' after a certain time of inactivity, whereas in SageMaker we did not face anything like this. In terms of environment and library compatibility, GCP is more user friendly compared to SageMaker, however we can configure it according to our need to meet those requirements. A more detailed description of each term and functions can be found in the following sections.

The next chapters talks about each term (ocean eddy, sar data, AWS SageMaker, S3 bucket, EC2, GPUs, Google services, CNN, YOLOv3), methodologies of this work, a detailed description on SageMaker as a tool perspective, results and discussion, and lastly a conclusion with future work direction.

Term clarifications:

Ocean eddy: An eddy is a circular current of water or air that runs contrary to the main current (flow). Usually Eddies are smaller, temporary loops of swirling water that can travel long distances before dissipating. Ocean eddies help distribute heat to the lower sea level as well as can transfer proteins for the sea creatures and lives. Significant amounts of current flow may cause effective hurricanes. In a nutshell, ocean eddies play an important role in climate change.

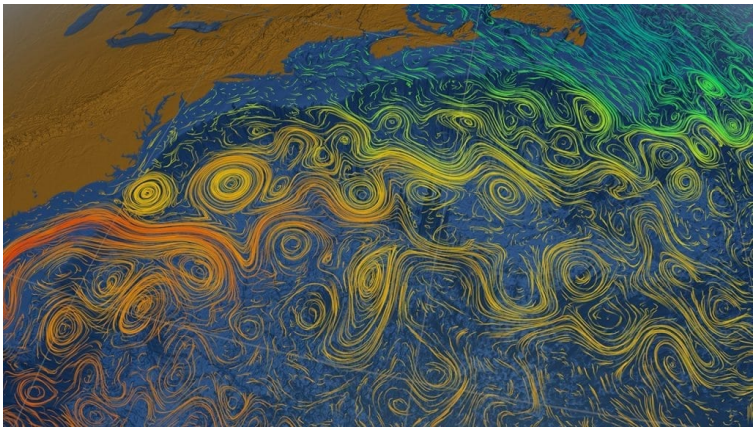

Figure 1: ocean eddies [2]

SAR-data: Synthetic aperture radar (SAR) data is collected from active sensors that transmit the microwave signals and then receive back the returned signals from the earth surface [9]. SAR data are basically high resolution satellite single or multi band images.

In this study we are provided the SAR image data from Jet propulsion laboratory (JPL), NASA [3] in tiff and png formats.

SageMaker: Amazon SageMaker is a fully managed cloud based machine learning service launched in November 2017. SageMaker not only enables data scientists and developers to quickly and easily build and train machine learning models, and then directly deploy them into a production-ready hosted environment. It also allows deployment of ML models on embedded systems and edge-devices. It provides an integrated Jupyter authoring notebook instance for easy access to your data sources for exploration and analysis, so there is no need for server management. It also provides common machine learning algorithms that are optimized to run efficiently against extremely large data in a distributed environment. With native support for bring-your-own-algorithms and frameworks, SageMaker offers flexible distributed training options that adjust to your specific workflows. Deploy a model into a secure and scalable environment by launching it with a few clicks from SageMaker Studio or the SageMaker console [4].

S3 bucket: Simple Storage Service (Amazon S3) is an object storage service that is offered by AWS services that ensures scalability, security with high performance. S3 is available for any size and sort of use cases such as, data lakes, websites, mobile applications, backup and restore, archive, enterprise applications, IoT devices, and big data analytics [5].

EC2 Instance: Amazon's Elastic Compute Cloud (EC2) is a cloud computing platform that allows users (with a subscription) to run their computing intensive applications. EC2 provides scalable deployment of applications in the cloud by initiating an instance and could turn off/terminate whenever the user wants [11]. EC2 allows us to choose from various operating systems including, linux (cent os, redhat, ubuntu), macos, raspberry pi and windows server with 32 and 64 bit architecture. Users can also choose from either CPU or GPU based instances with multiple cores (with added price).

GPUs: Graphics processing units (GPU) are specially designed electronic circuits that accelerate the image processing. GPUs are used within various systems including embedded systems, mobile phones, personal computers, workstations, game consoles and so on. Modern GPUs are very efficient with their highly parallel structure that differentiate the GPUs from the general-purpose central processing units (CPUs) for processing large blocks of data and algorithms in parallel. Cloud GPUs provide hardware acceleration without requiring that a GPU is deployed on the user's local device. Common use cases for cloud GPUs are big data processing, visualization workloads and computational workloads. In recent times GPU parallelism has been popular within Distributed Deep Learning (DDL) platforms. For large scale deep learning workloads GPU parallelism is one of the top most choices in the data scientist community. In this report we also introduced the DDL framework for the Ocean Eddy project.

Google Cloud Platform (GCP): GCP offered by Google, is a suite of cloud computing services that runs on the same infrastructure. Google Cloud Platform provides infrastructure as a service, platform as a service, and serverless computing environments. It provides plenty of services including Storage, Databases, Networking, Operations, Developer Tools, Data Analytics, AI and Machine Learning. Google's Colab" for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education. More technically, Colab is a hosted Jupyter notebook service that requires no setup to use, while providing free access to computing resources including GPUs [6, 7]

CNN: A convolutional neural network (CNN, or ConvNet) is a class of artificial neural network (ANN), most commonly applied to analyze visual imagery. CNNs are composed of multiple layers of artificial neurons. Artificial neurons, a rough imitation of their biological counterparts, are mathematical functions that calculate the weighted sum of multiple inputs and output an activation value. When we input an image into a ConvNet, each of its layers generates several activation maps. Activation maps highlight the relevant features of the image. Each of the neurons takes a patch of pixels as input, multiplies their values by its weights, sums them up, and runs them through the activation function.

YOLOv3: YOLO is an abbreviation for the term 'You Only Look Once'. This is an algorithm that detects and recognizes various objects in a picture (in real-time). Object detection in YOLO is done as a regression problem and provides the class probabilities of the detected images. The YOLO algorithm employs convolutional neural networks (CNN) to detect objects in real-time. As the name suggests, the algorithm requires only a single forward propagation through a neural network to detect objects. This means that prediction in the entire image is done in a single algorithm run. The CNN is used to predict various class probabilities and bounding boxes simultaneously. The YOLO algorithm consists of various variants. Some of the common ones include tiny YOLO and YOLOv3 [10]. YOLO completes the processes in a few steps that includes residual blocks, bounding box regression and Intersection over Union (IOU).

At first, the **residual block** which is the network comes made up of convolutional layers that detect key features from images and process them by dividing the images into various grids (such as a dimension of SxS). Then, in the **bounding box regression** part, features from the convolution layers are used to make predictions on probabilities and bounding box coordinates that highlight an object within the image. Every bounding box in the image consists of width, height, Class (for example, person, car, eddy, etc.), and bounding box center. The final part is the **Intersection Over Union (IOU)** is the evaluation process that describes how boxes overlap. YOLO uses IOU to provide an output box that surrounds the objects perfectly. Each grid cell is responsible for predicting the bounding boxes and their confidence scores. The IOU is equal to 1 if the

predicted bounding box is the same as the real box. This mechanism eliminates bounding boxes that are not equal to the real box.

Dataset: We are given the satellite images which were in Float32 ‘tiff’ format. One major problem with those images is, the content cannot be seen in regular image viewer applications (e.g., photos in windows). We used QGIS (quantum geographic information system) software to view the image content. QGIS is also capable of converting images to ‘png’/’jpg’ formats. The tiff images were in RGB format, where one channel had the information we needed, and the other two channels contained latitude and longitude information. Thus, we needed to extract only the single band out of the image. To do so, we used the Geospatial Data Abstraction Library (GDAL) to extract the single band and convert the images (using gdal python) to Uint8 png formats. These Uint images were then viewable by the regular photo viewing applications. The main purpose of converting those images using GDAL library was to make them compatible in AWS and Google platform and reduce the image size to train them using the RestNet model as the tiff images were not able to be read by either of those platforms.

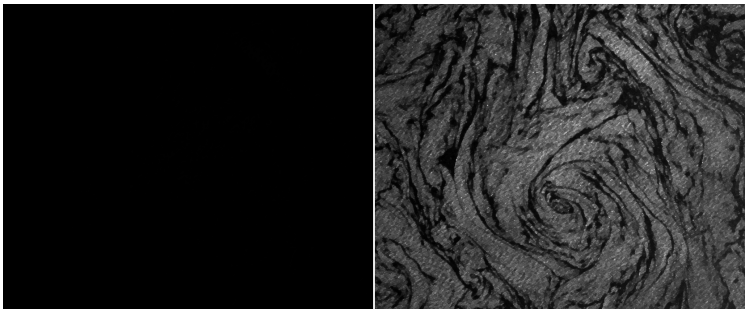

Figure 2: same image before(left) and after(right) conversion using ‘gdal’ library

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 254, 254, 32)      320

 max_pooling2d (MaxPooling2D  (None, 127, 127, 32)     0
 )

 conv2d_1 (Conv2D)           (None, 125, 125, 64)      18496

 max_pooling2d_1 (MaxPooling  (None, 62, 62, 64)       0
 2D)

 conv2d_2 (Conv2D)           (None, 60, 60, 32)        18464

 max_pooling2d_2 (MaxPooling  (None, 30, 30, 32)       0
 2D)

 flatten (Flatten)           (None, 28800)             0

 dense (Dense)               (None, 8)                 230408

 dense_1 (Dense)             (None, 1)                 9

=================================================================
Total params: 267,697
Trainable params: 267,697
Non-trainable params: 0
```

Figure 3: CNN model summary

Methodologies: There are 75 images in total where 70 of them are with eddies and 5 of them are with no eddies. We trained two different models for this study. Due to the imbalance data we first trained a K fold Cross validation CNN model (k=5) using the converted images. We used three consecutive convolutional and max pooling layers with the input shape of 256x256x1, relu activation in all three of them and later relu and sigmoid at the dense layer. Figure 4 is a model summary used in our case. The CNN model with cross validation achieved 60% accuracy using the CPU (reproducible using fixed seed). We also tried the same code with GPU based execution, but found the accuracy can vary the result, which might be because of their own randomness in the code [1].

We also used an object detection approach using YOLOv3. The main objective of this model is to identify and locate eddies in images. Before feeding the YOLO model, we prepared the labeling data in two ways. Our first approach was using the 'ground truth' feature from AWS's SageMaker and the second approach was to use the 'LabelImg', a graphical image annotation tool. Both of these tools provided the annotation data for the boudin box. Sagemaker has few extra features other than just bounding box options, which are image classification, semantic segmentation, label verification, anchor boxes etc. We only use the bounding box feature in this study. The annotated results produced from ground truth are stored into S3 bucket as a 'manifest' file which is directly accessible to the jupyter notebook in the AWS platform for further processing. LabelImg, on the other hand is more specific to YOLO (and pascal voc, which we did not use) that provided bounding box annotations results. We then used the image files and text files with the annotated bounding box information to the YOLO model to train. We trained a custom model using our own data and custom configuration (cfg) files with necessary requirements. Some core changes in the configuration file are shown in Figure 5.

```
# Training
batch=64
subdivisions=32
..........
..........
max_batches = 4000
..........

[convolutional]
size=1
stride=1
pad=1
#(no. of classes+5)x3
filters=21 # for two clasess
activation=linear
[yolo]
classes=2

[convolutional]
size=1
stride=1
pad=1
#(no. of classes+5)x3
filters=21 # for two clasess
activation=linear
[yolo]
classes=2

[convolutional]
size=1
stride=1
pad=1
#(no. of classes+5)x3
filters=21 # for two clasess
activation=linear
[yolo]
classes=2
```

Figure 4: Yolo v3 custom configuration file
(key changes from default configurations are shown)

Detecting the eddies from these images was quite challenging, however we got a variable success rate starting from 40% to 90% while detecting and locating the eddies. The probable cause of less accuracy could be, *i)* the circular motion of eddies sometimes expanded with variable tail size (long/short and/or thin/thick tails) in the images, and *ii)* the deep learning model requires a large number of images which we do not have in our case.

However, we faced some limitations with the YOLO for our model prediction. We observed that the results are quite variable in terms of identifying and confidence level. As we can see from the above images. In portion 1, we can see ot detected the eddy correctly but the confidence is as low as 44%, in portion 2, it detected one of the two eddies with the same confidence. In portion 3, it detected a less important eddy by leaving the center eddy. In block 4 we have pretty good results both for eddies and non-eddies.

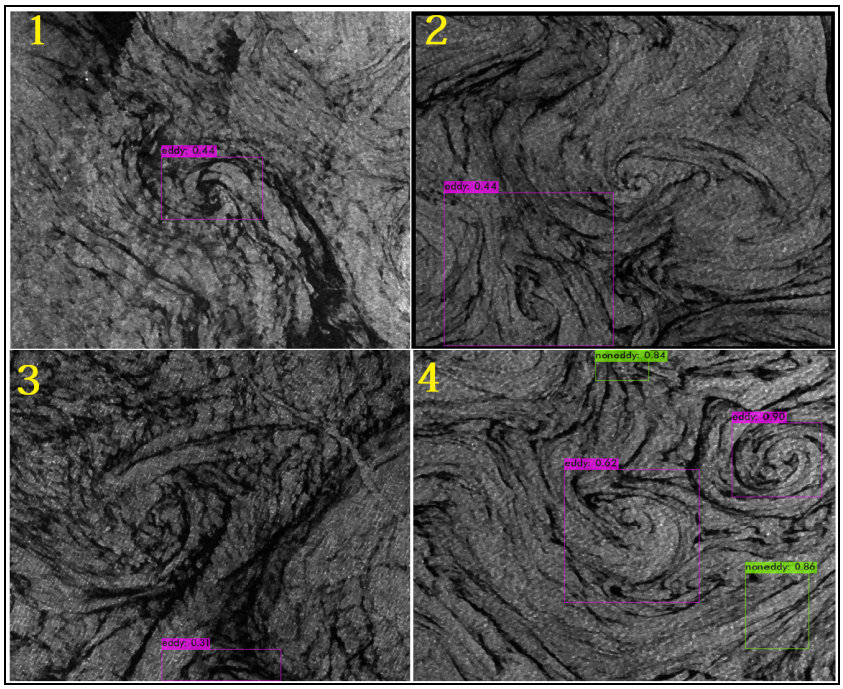

Figure 5: object detection and identified using YOLOv3

Ocean Eddy Detection via AWS SageMaker: Amazon SageMaker is a great tool for developers and data scientists to quickly and easily build, train, and deploy machine learning models at any scale. The SageMaker removes the barriers of knowing programming languages to train and build models for non computing personnel. However, it is not as user friendly yet that anyone can use without prior knowledge. The AWS documentation for SageMaker also lacks more defined steps towards building an end-to-end model.

SageMaker provides a ground truth tool that enables a user to label their images. Users can either choose from single class to multi class problem, or use bounding box annotations to primarily indicate objects. It also allows additional workers to do the job. For example, if you have 1000 images to label you can assign a few workers to label the images. After labeling all images, it is stored in the S3 bucket as a manifest file which contains a json type structure. Basically, this manifest file provides the bounding box values along with the image path. This manifest file can directly be used in model training.

Another important thing about SageMaker is, it provides a jupyter notebook which is a convenient option for developers to deploy their customized codes. The jupyter notebook can directly access files from S3 bucket within the AWS platform which is faster and convenient in terms of accessibility. Even any produced results from this notebook can be stored back to the S3 directories.

The next important feature of SageMaker is the ‘Training’ feature that allows users to train their model. The main purpose of this feature is to enable any user without no programming knowledge to still train models according to their needs with a few simple steps like choosing

directories from S3, selecting which model to run, etc. In this study we did not go through this automatic method selection process.

Beside these features we found some significant shortcomings with amazon SageMaker. First, if there is no worker assigned, the labeling process is never completed which means, the original initiator cannot finish the labeling job alone as of now. Second important drawback is that you can assign more than one worker but when any one worker has completed his/her labeling task, it does not allow other workers to do their jobs. When the next worker logs in to complete their tasks, they get a notification saying that, 'you have no work to do!'. We are identifying this as an important problem as various workers might have more labeled data that could help in accuracy. Another important con we found is we could select a model to train using AWS SageMaker using our annotated data from the ground truth to build an end-to-end pipeline, however the platform does not support that yet.

Ocean Eddy Detection via AWS EC2: In this work we also utilized the EC2 instance for ocean eddy detection. EC2 offers a range of OS platforms with either CPU or GPU based computation. We are more interested in single core and multi core GPU instances which are used throughout this work. In terms of image processing GPU is an ideal hardware as it offers comparatively faster processing than CPU. Moreover, multicore GPUs in EC2 allow parallel processing which is even better for large data sets.

The EC2 instances can easily be configured from the terminal in terms of setting up the environment. Once the environment is prepared, the data and source code can be uploaded to the instance. We cannot execute any notebook files (i.e. jupyter notebook in SageMaker) other than scripts (i.e., bash, python etc.) To use the GPU, we need to confirm that the GPU is enabled and can be accessed. To ensure that we use CUDA that manages the GPU instances and to ensure the distribution of workload we use Horovod [12]. Horovod is a free and open-source software framework for distributed deep learning training using Keras and other popular libraries. The main goal of Horovod is to make distributed Deep Learning faster and easier which we deployed in the OceanEddy project. The OceanEddy training script is configured using Horovod to scale up the training process between GPUs. The OceanEddy task is now fully parallelized using Horovod. Sample notebooks and scripts configured with Horovod can be found in our github repository.

Results and discussions: the results we have achieved so far are satisfactory in terms of identifying ocean eddies from satellite images. However, there is room for improvement by incorporating more images of similar type. When we have more annotated images we can improve our results toward better prediction. This project was a very good learning experience on how to work with imbalanced image data in terms of CNN and object localization in cloud platforms. The SageMaker is especially new and exciting in terms of its purpose. Furthermore, we would like to have more high resolution images to train and test our model to compare how the models behave if we have different kinds of eddy images. In order to improve CNN and YOLO, we need a balanced dataset with increased numbers. Also we would like to try with better quality images where the characteristics of eddies are identifiable. For example, in most

cases the eddy tails overlapped other eddies with their long tails (from a visual observation) and it is hard to define a single perfect eddy. This is one of the observations from this study. The other possible reason could be image contents, for example, the image has some dark spots around eddies (some images) which is misleading toward non-eddies.

Conclusion and future work: This study helped us to understand the ocean eddy detection of specific regions by training models from satellite data using cloud services and. From this result we can determine whether there is an eddy and where the eddy is located within the image. However, it is possible to be specific with the size of eddis, their movements, directions and probable cause of variation and possible impacts in the sea life thus observing the atmospheric changes. Further analysis on the ocean eddy tracking and analyzing the variations by building suitable models using available cloud services would help us better understand climate change in a quicker manner.

References

[1] https://machinelearningmastery.com/reproducible-results-neural-networks-keras/
[2] https://www.whoi.edu/know-your-ocean/ocean-topics/ocean-circulation/currents-gyres-eddies/
[3] https://www.jpl.nasa.gov/
[4] https://docs.aws.amazon.com/sagemaker/latest/dg/whatis.html
[5] https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html
[6] https://cloud.google.com/docs/overview
[7] https://research.google.com/colaboratory/faq.html
[8] https://bdtechtalks.com/2020/01/06/convolutional-neural-networks-cnn-convnets/
[9] https://asf.alaska.edu/information/sar-information/what-is-sar/
[10] https://medium.com/analytics-vidhya/yolo-explained-5b6f4564f31
[11] https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html
[12] https://horovod.ai/