G5

PROJECT DOCUMENTATION

Architecture Notebook

Aerial Imagery Initiative

Index

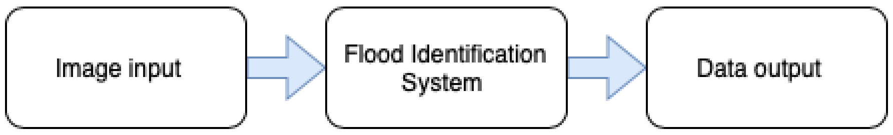
Purpose

The Department of Customer Services (DCS) is in the process of evaluating and creating a system to determine the extent of a flood extent from aerial imagery.

This document describes the system and process architecture that will be used to achieve such an outcome.

Architectural goals and philosophy

The desired outcome of the system is to be able to process an image and have the system analyse and determine the flood extent within the image.



The output will be used by Emergency Service Organisations (ESOs) to help plan and prepare for the rescue of people and assets during a flood event and/or however else they see fit to utilise the information.

Currently there is no quick and accurate method of performing this type of information extraction. DCS and ESOs are hoping to implement identifying the flooded region into their current image processing pipeline.

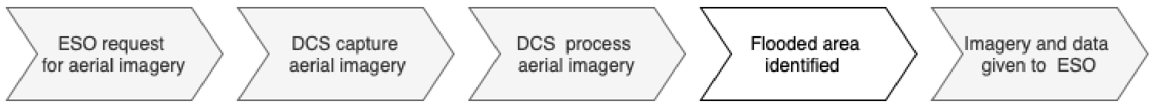
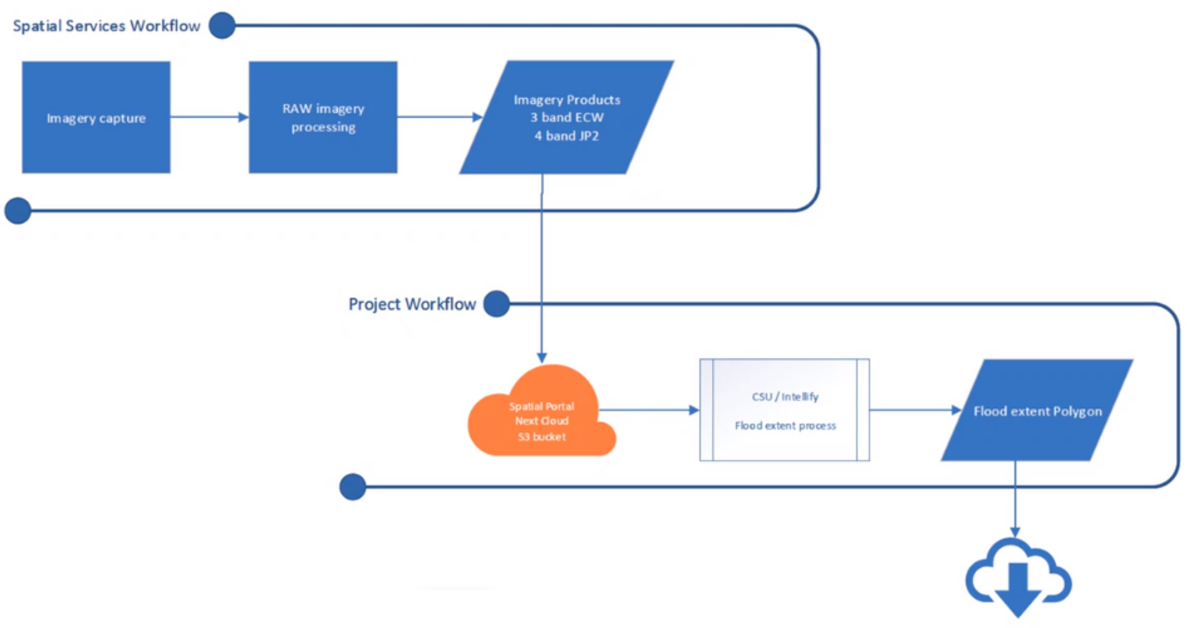

ESO request for aerial imagery
DCS capture aerial imagery
DCS process aerial imagery
Flooded area identified
Imagery and data given to ESO

Image processing pipeline

As the circumstances for the imagery request may be to help during a flood event, DCS attempts to have imagery processed and and returned to the requesting ESO within a 48hr time period. Because of this requirement manual identification of the flood area manually is not possible as it relies on human availability and may take many hours therefore an automated system is required.

The identification of the flooded area using an automated system is to not have a significant impact of the targeted 48hr turnaround time, therefore it is envisioned that the complete process will take 1-2 hours to complete. This includes the organisation and pre processing of data and the verification of output data. **Actual image processing time will need to be in the minutes.**

To Summarise DCS requires a system that accepts imagery captured from their aerial imagery systems processed to identify flood extents. The system will be able to identify the flood extent and express that in human readable format. The whole process will take 1-2 hrs to complete and the actual processing time on the image file to be in minutes.


Spatial Services Workflow
Imagery capture
RAW imagery processing
Imagery Products 3 band ECW 4 band JP2
Project Workflow
Spatial Portal Next Cloud S3 bucket
CSU / Intellify Flood extent process
Flood extent Polygon

DCS / SS Workflow (supplied of DCS)

Assumptions, constraints and dependencies

Type	Name	Reasoning
Constraint / Dependency	AWS infrastructure	The system is to be built within the DCS AWS environment. DCS currently utilises the AWS environment for other tasks within the business and it is desired that the flood extent identification system also resides within the AWS system. As we will be restricted to features and products with AWS we are constrained and also depend upon AWS to build and run the system.
Constraint	Image File Format (JP2, ECW)	Imagery files captured from aerial imagery will be in the JP2 and/or ECW format.
Assumption	Skills / Ability	The skill and ability to learn are a dependency for the project's success. The CSU team are not Machine Learning experts and have next to no experience. However, the team does have direct access to ML experts in the form of David Tein and Intellify consultants.
Assumption	Time	The CSU project students are available to work on the project till the end of the last session of the CSU calendar. It is being assumed that all work will be completed and handed over to DCS before the date of 29th Oct' 2021.
Constraint	Output format (ESRI SHP)	For the output to be of most use it is required to be in ESRI SHP file format. This format will be able to be given out and utilised in many mapping software tools.

		https://www.esri.com/content/dam/esrisites/sitecore-archive/Files/Pdfs/library/whitepapers/pdfs/shapefile.pdf
Assumption	Example/Training Data	For the development of accurate output from the system. Example data will be required to be able to train and test the eventual solution. This will be in the form of multiple images that provide a representation of the spectrum of scenarios that the system is expected to perform under.

Architecturally significant requirements

Requirement	Initial Requirements Model Reference	Architectural Significance
Image input	1.1.1	A mechanism must exist the enables the ability to upload or direct the system to images that need to be analysed
Image segmentation	1.1.3 1.2.2	For the purpose of training and testing a process is needed to segment images to identify flooded and non flooded areas. This will be performed in the AWS environment using SageMaker Ground Truth.
Image pre processing / Feature extraction	1.1.4 1.2.2	To increase the chances of accurately identifying flooded areas, images may need to be pre-processed to reduce the number of futures and noise on the image. To accomplish this tooling and processes will need to be available during preparation of the data for training, testing and production use. AWS SageMaker studio will be the working environment and hence the python programming language and its packages will be used to pre-process the image and perform feature extraction.
Image classification	1.1.2 1.2.1	An automated method must exist that can accept, process and identify the flooded area within an image. It is likely this will be performed via Machine Learning (ML) however traditional processes of using computer vision are also possible candidates. Once a suitable process is accepted the Image classification will be a standalone SageMaker EndPoint instance that is capable of processing new flood imagery.

Output format	1.1.5 1.2.4	The final output format will need to be compatible with ArcGIS software. DCS has indicated that a ESRI SHP fille format is desirable. Conversion from raw pixel data may need to transformed into a SHP file after classification is completed on the image.
Storage	1.4.2 1.4.3	Flood imagery files are very large (up to 10GB) and the system will need to be flexible to allow scalability in the future if the system proves successful. Therefore the storage needs to be flexible and efficient. AWS S3 is the likely candidate as it can be used to hold small to very large data sets.

Decisions and justifications

Amazon AWS Environment

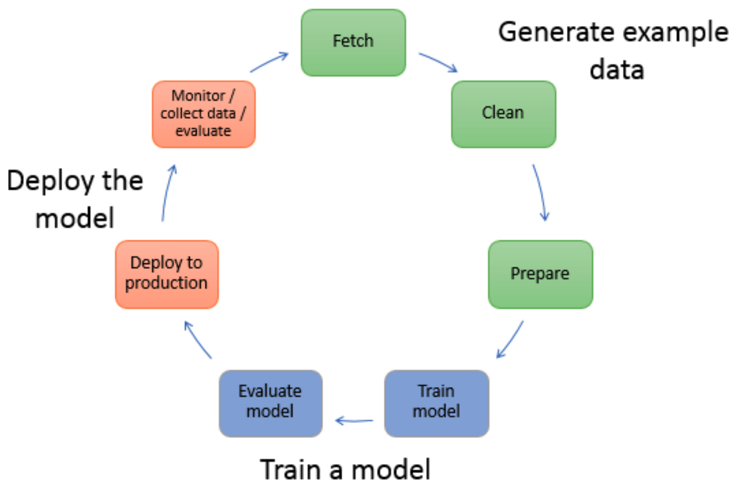
Based on the requirements, constraints and dependencies the system will be developed and hosted within the Amazon AWS environment. AWS already provides many of the tools and processes needed to accomplish the required outcome for the system. AWS was already specified as a requirement by the client however it is a suitable choice regardless of it being the preferred choice.

The AWS services used will be;

- S3 buckets for storage
- SageMaker for image pre-processing
- GroundTruth for Image Segmentation
- SageMaker to control the Training creation of ML models
- SageMaker to deploy the image classifier (SageMaker Endpoint)

Development / ML Lifecycle

As AWS is the preferred platform a life cycle similar to what is depicted below will be used for the development of the system. This life cycle is described by AWS for use in the SageMaker environment however this also reflects that of a typical ML lifecycle.



Development / ML lifecycle (Provided by AWS)

Generate Example Data: Images supplied by DCS are stored in S3 buckets ready for pre-processing. Preparing the data will include the cleaning, feature reduction and segmentation of images in GroundTruth and stored again in S3.
The segmented images can then be loaded within SageMaker ready for training.

Train a Model: SageMaker has the ability to perform experiments on data and conduct training on ML models. By using experiments you are able to determine the best method for image classification.

Deploy the Model: Once a model has been trained and tested it can be deployed as a SageMaker Endpoint. An Endpoint is a special EC2 instance that allows new images to be sent to the trained machine model for classification. Once classified the Endpoint returns a response that can be used to identify the flooded area within the supplied image.

SageMaker Endpoint

Once a ML model is trained and deployed as an SageMaker Endpoint it can be utilised by many other AWS services (AWS Lambda, EC2 instances). This means that it can be directly integrated into an automated pipeline or continuous integrated solution. The Sagemaker Endpoint can also be replaced by a better trained more accurate model or redeployed on more powerful hardware if quicker turnaround times are needed.

Architectural Mechanisms

AWS account and access

DCS will be supplying access to their AWS account using a role designed for use in the project. The account and role have been already configured and distributed to all developers working on the project. AWS hosts all the required tools to accomplish the project.

S3 Bucket

S3 is an object storage service provided within the AWS environment. It can be accessed from web browsers, AWS services (instances, SageMaker Studio, Lambda .etc). S3 will be the primary storage for all persistent project data:

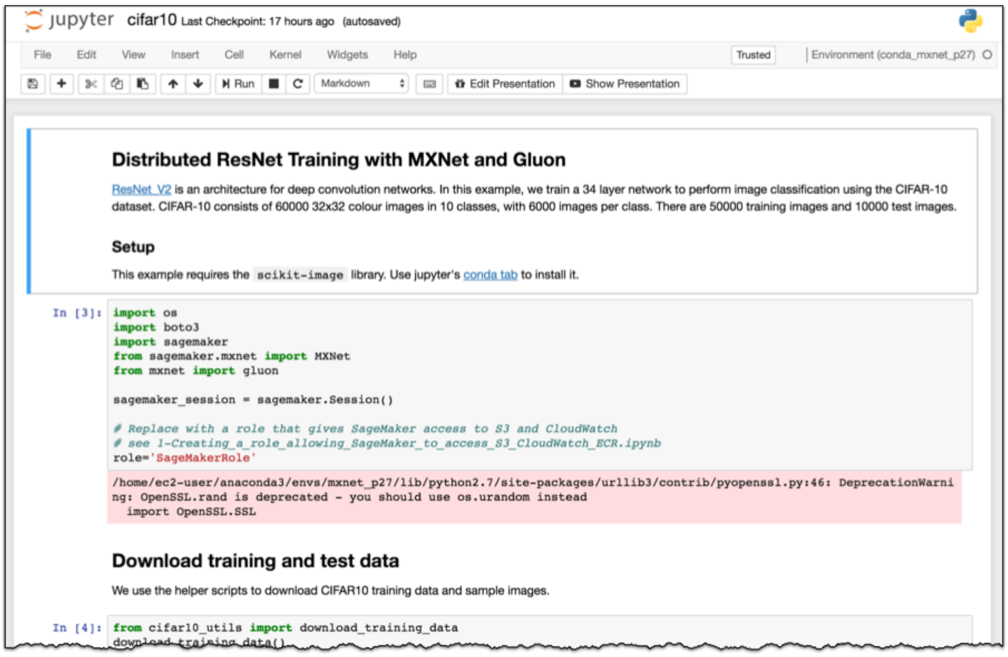
- Raw data / images
- Segmented images
- Pre-processed images
- Training data

S3 buckets have been created and access given to developers.

SageMaker Studio

SageMaker Studio will be the point of convergence for most tasks with the exception of uploading data to storage buckets and image segmentation within GroundTruth.

SageMaker Studio makes use of Jupyter Notebooks (JNB). JNB is a tool that allows developers to create and document their programs as they are made. Programs are typically written in Python and documented in Markdown. The program code and documentation live inside the same file (.pynb). Below is an example of a Jupyter Notebook used on an Amazon provided Sagemaker example.



Distributed ResNet Training with MXNet and Gluon

ResNet_V2 is an architecture for deep convolution networks. In this example, we train a 34 layer network to perform image classification using the CIFAR-10 dataset. CIFAR-10 consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

Setup

This example requires the `scikit-image` library. Use jupyter's conda tab to install it.

```
In [3]: import os
        import boto3
        import sagemaker
        from sagemaker.mxnet import MXNet
        from mxnet import gluon

        sagemaker_session = sagemaker.Session()

        # Replace with a role that gives SageMaker access to S3 and CloudWatch
        # see 1-Creating_a_role_allowing_SageMaker_to_access_S3_CloudWatch_ECR.ipynb
        role='SageMakerRole'
```

```
/home/ec2-user/anaconda3/envs/mxnet_p27/lib/python2.7/site-packages/urllib3/contrib/pyopenssl.py:46: DeprecationWarni
ng: OpenSSL.rand is deprecated - you should use os.urandom instead
  import OpenSSL.SSL
```

Download training and test data

We use the helper scripts to download CIFAR10 training data and sample images.

```
In [4]: from cifar10_utils import download_training_data
        download_training_data()
```

Example Jupyter Notebook (JNB)

JNB can also be used to interact with the underlying host system. Below is an example of installing a 3rd party python package using pip.

```
%pip install -qU 'sagemaker>=2.15.0' 's3fs==0.4.2'
```

SageMake example JNBs can be found here:
https://github.com/aws/amazon-sagemaker-examples

JNB will make use of python packages to complete the tasks required with the SageMaker Studio. Some of the packages that will be utilised are:

- Boto3 - SDK for python, allows use of many AWS services such as S3 and EC2
- Sagemaker - SDK for python, used for training and deploying sagemaker models

SageMaker and Jupyter NoteBooks are available within AWS. SageMaker studio will need to be deployed and setup once development begins

SageMaker GroundTruth

Image segmentation will be performed within the GroundTruth service on AWS. The service provides a set of tools and configuration options that allow raw images to be selected from an S3 bucket for image segmentation. Segmentation data is stored on the S3 bucket for use in training and testing of ML models within SageMaker.

Architectural Layers

The system will be viewed as a layered stack of components and processes as depicted in the image below. Each layer represents a stage in the machine learning life cycle whilst identifying which service and tools will be used at each stage.

Layer 1: Image processing and segmentation

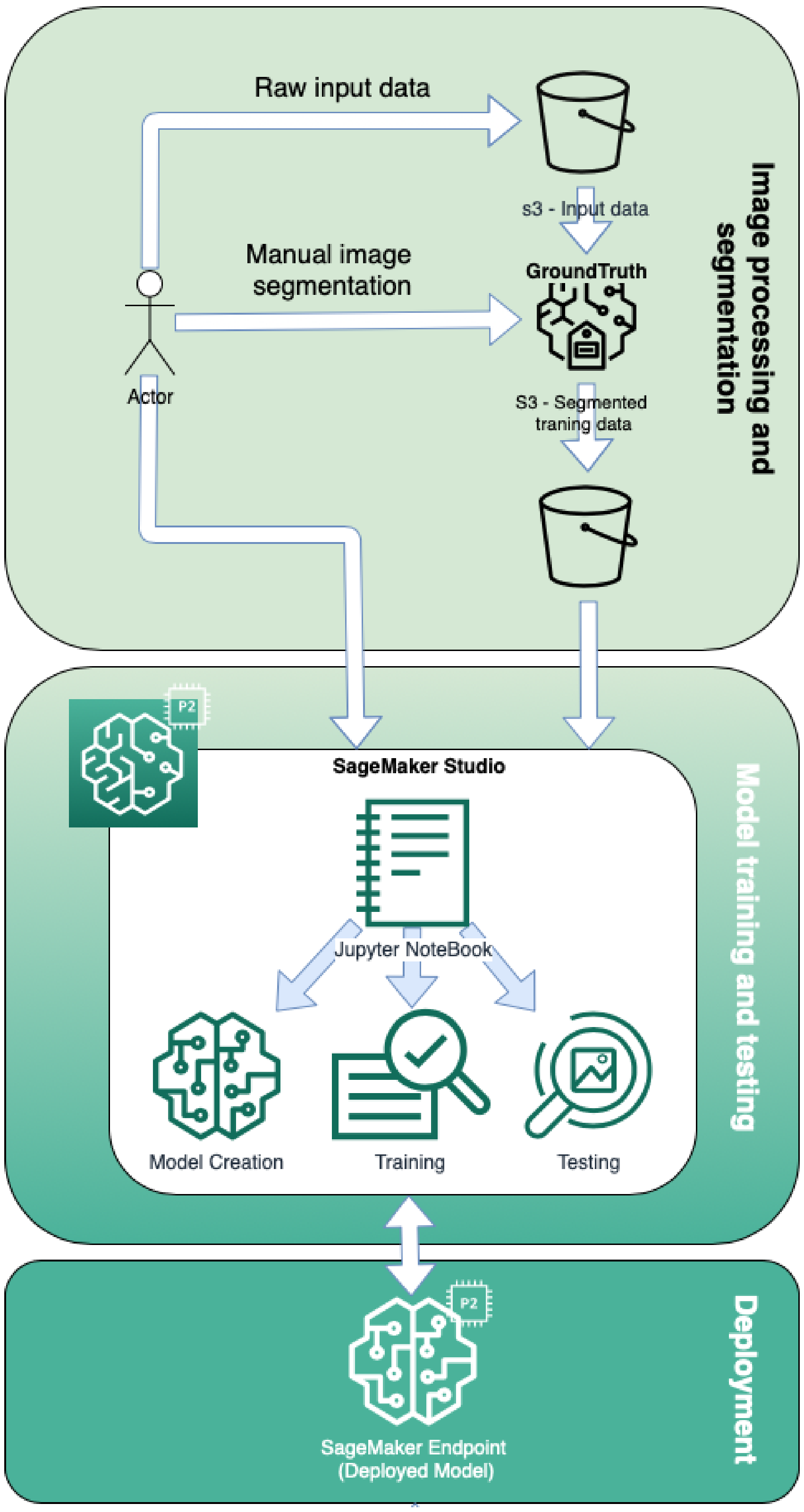
Data will be uploaded into S3 buckets. Once uploaded data can be retrieved and segmented using GroundTruth. The segmentation results will be saved back into an S3 bucket for later use. Much of this layer will initially be manual point and click actions or manual input of commands using the AWS CLI.

Layer 2: Model Training and Testing

When data is ready testing and training of a model will begin. Here SageMaker studio and JNB will be heavily used to retrieve data from the S3 buckets and given to the ML model. Once a model is trained it will be tested to ensure it has the ability to perform to an acceptable degree of accuracy.

Layer 3: Deployment

When a model has been trained and deemed suitable it will be deployed as a sageMaker endpoint. Once deployed it can be used by many AWS services (lambda, EC2, SageMaker Studio .etc). From this point DCS will be able to utilise the Endpoint in their workflow when delivering

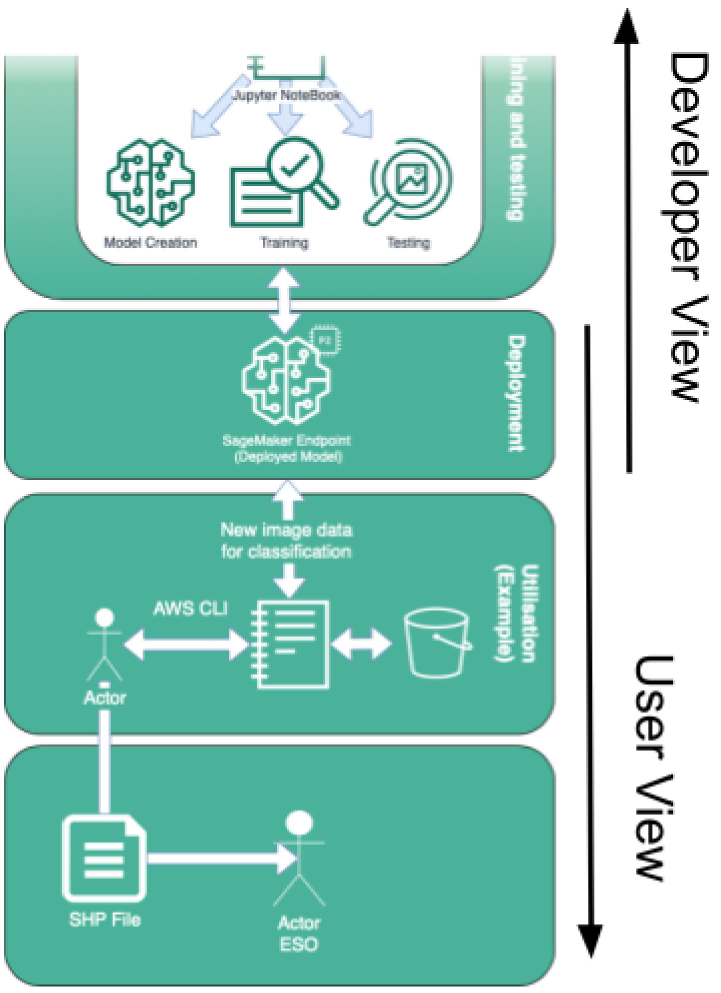

Raw input data
s3 - Input data
Manual image segmentation
GroundTruth
Actor
S3 - Segmented traning data
Image processing and segmentation
SageMaker Studio
P2
Jupyter NoteBook
Model Creation
Training
Testing
Model training and testing
P2
SageMaker Endpoint (Deployed Model)
Deployment

Architectural Layers

Architectural views

Below is an extended model of the architecture. As described in the Layers section the system is composed of 3 layers. This may be considered the **logical view** or **Development View.** It describes how all the key components control and interact with each other.

The new layer, layer 4 is how the end user might utilise the system once deployed. This could be considered the **User View.** In this view the user is interacting with a JNB to retrieve data from a S3 bucket and have it processed by the SageMaker Endpoint. The endpoint will return classification data that will be used to identify flood extents in the image.



References

Title page cover photo.
Yeronga flooded. (2011). Retrieved from https://www.abc.net.au/news/2011-06-03/brisbane-floods3a-before-and-after-article/2742794?nw=0