

# To GAN or Not To GAN: Segmentation Analysis on Mars DEM

Douglas D. Agbeve

agbeve01@ads.uni-passau.de

University of Passau

Salim Fares

fares01@ads.uni-passau.de

University of Passau

Aditya V. Handrale

handra01@ads.uni-passau.de

University of Passau

Seif E. Idani

idani01@ads.uni-passau.de

University of Passau

## ABSTRACT

*To better understand Martian Surface, which is needed to enable Rovers navigate Mars with ease, it is necessary to be able to determine the location of mounds. Detecting and studying these morphologies can also help us find evidence of extraterrestrial life, in this case, more specifically, water or signs of life conducive environments. Detection of mounds was done by manually mapping morphological parameters onto Digital Elevation Models. This paper solves the problem by automatically detecting and or predicting mounds on Mars using Neural Network based Semantic Segmentation methodologies. This is done by using supervised semantic segmentation model and generative adversarial approach. A comparison of the approaches shows that adding extra artificially generated data did not improve the result.*

## 1 INTRODUCTION

A picture, they say, is worth a thousand words. As a medium of communication, pictures have been used since the dawn of *Homo Sapiens*[1], which makes understanding the information embed in them a major part of research in the various fields of science ranging from Archaeology to the relatively more recent Computer Science.

In Computer Vision, a vital process in retrieving information in images and which form part of most systems for visual understanding is image segmentation[2]. It is the process of partitioning an image or a video frame into multiple segments[3]. Segments are fundamentally a set of pixels in a region that share a common property such as colour or texture, enabling the identification and locating of boundaries of objects in an image[4]. Digital image segmentation is applied in various domains, including but not limited to, medical (locating cancerous tumors and measuring tissue volumes)[5], facial and fingerprint recognition, self-driving vehicles (detecting pedestrian).

There are, and continue to be proposed, many algorithms for image segmentation. Currently, these approaches are either segregating pixels based on intensity changes (i.e. detecting discontinuity) such as edge detecting algorithms, or partitioning an image into regions with similar predefined criteria (i.e. Similarity detection) examples include thresholding approaches. The various algorithms can also be categorized based on the method used in solving the problem; namely (1) Edge detection based segmentation e.g. histogram-based[6] and gradient-based[7], (2) Thresholding

approaches such as those proposed by Abd Elaziz et al.[8], and Houssein et al.[9], (3) Region-based segmentation techniques include region growing[10, 11], region split and merge[12, 13], (4) techniques based on Partial Differential Equation are active contour[14], C-V model[15] etc., (5) Clustering methods such as K-means clustering algorithm[16] and relatively more recent approach is the Artificial Neural Network (ANN) algorithms which solves image segmentation problems with higher accuracy compared to other approaches. Examples of Deep learning or ANN approach to image segmentation include, but not limited to, Convolutional Network models such as VGG16, GoogLeNet, Fast R-CNN, Faster R-CNN[17], U-NET[18], V-NET[19] and Mask-RCNN[20], Recurrent Neural Network models include ReSeg[21], Generative and Adversarial (GAN) Models e.g. [22, 23, 24], a comprehensive list of GAN-based techniques can be found in [25].

The problem of image segmentation can be coined as that of classifying pixels with semantic labels i.e. semantic segmentation (fig. 1c) or segregating pixels into individual objects in the image i.e. instance segmentation (fig 1d)[26]. In semantic segmentation, pixels are assigned to the same segment if they are of the same object type in the image. This can be thought of as classification at pixel level. Instance segmentation involves assigning all the pixels that belong to the same single object to the same segment. It is a two-step process; extracting bounding boxes around each instance of an object through object detection and then classifying pixels that corresponding to each instance in the bounding box. This technique combines object detection (fig. 1b) and segmentation. A third, relatively new, formulation is to combine both instance and semantic segmentation termed Panoramic Segmentation (fig. 1e). It involves the detection and segmentation of all objects including background and labelling different instances in an image.

Digital Elevation Models (DEMs) are essential in analyzing erosion and drainage, hill-slope hydrology, studying groundwater flow, watersheds, and contaminant transportation as they are important tools for parameterizing topography. DEM is representation of planetary (earth, moon, mars etc.) terrain from elevation data in 3-D image. There are two types; raster Geographic Information Systems (GIS) layer representation and the vector-based triangular irregular network format. DEMs are obtained using techniques such as Surveying, Stereo Photogrammetry, Lidar, Radar, etc.. Flight and Train simulations, GIS and Satellite navigations are some of the systems in which DEMs are used. Martian surface has an abundance of geographical features such as volcanoes, layers and gullies, with phenomena like volcanoes bringing out microbial entities that were protected from solar radiation. Moreover, glaciers and mounds that have water underneath them might contain evidence of life.

Detecting and studying these morphologies can help us find ex-traterrestrial life on mars. The detection of mounds formed from phenomena such as volcanic eruption can be done automatically using ANN based image segmentation methods. Wuff-Jensen et al.

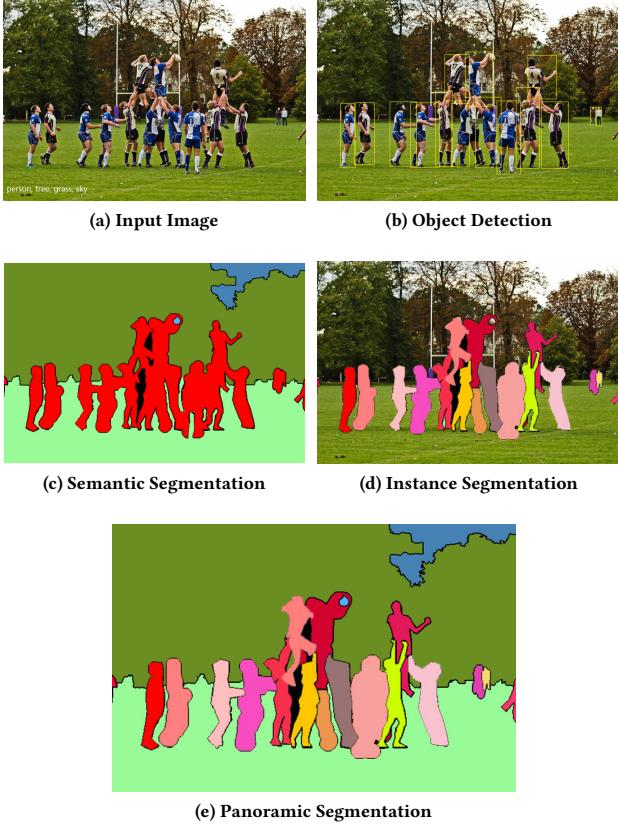


Figure 1: Variations of Segmentation [27]

proposed, to the best of our knowledge, the original idea of using GANs to generate terrain from DEMs. Their architecture was based on deep convolutional GAN (DCGAN)[28] – a type of GAN made up of a fractional-strided convolutions (generator) and a discriminator of strided convolutions.

Spick et al.[29] presented a method of generating height maps from digital elevation of regions of earth. The authors approach was based on Spatial GANs (SGAN)[30]. Spatial GANs remove fully connected layers from DCGAN allowing outputs to be scaled to any size and mapping features onto output as translation-invariant.

Bowles et al.[31] investigated augmenting training data using GAN-derived synthetic images. Progressive Growing of GANs (PG-GAN)[32] network was used as a baseline architecture, and demonstrated that this can improve results across two segmentation tasks (1) - Computed Tomography (CT) images with manually delineated Cerebrospinal Fluid (CSF) labels, (2) - Fluid-Attenuated Inversion Recovery (FLAIR) images with manual White Matter Hyperintensity (WMH) segmentations.

## 1.1 Research Questions

The many algorithms available for solving image segmentation problem is an indication that no single approach can solve it all accurately and in a more efficient way. An affirmation of this assertion is the numerous methodologies proposed in literature by Deep Learning researchers in recent decade. Furthermore, the choice of algorithm and by extension the need for new approaches, is principally influenced by data pertaining to the problem, its representation and the type of segmentation problem. In view of these, we formulated the following questions with regard to the data at hand.

- Will data augmentation have any effect on the accuracy of our selected approach (es) for application, and if yes, how much extra data relative to the training data?
- Which Deep learning technique(s) segments the data with better accuracy?

## 2 PROBLEM STATEMENT

In this section, we formally present the problem of automatically detecting mounds in Mars' Arabia Terra. Specifically, classification of pixels in the digital elevation model of Mars in to segments of mounds or no-mounds using using Artificial Neural Network based techniques. We sought to predict where mounds given any DEM of Mars

More formally, given  $n$  annotated images  $\{x_1, x_2, \dots, x_n\}$  and each image, say  $x_i$ , has  $m_i$  objects that are categorize into  $C$  classes with objects labelled as  $y_i$ :

$$y^i = \{(c_1^i, v_1^i), (c_2^i, v_2^i), \dots, (c_{m_i}^i, v_{m_i}^i)\}, \quad (1)$$

where  $c_{m_i}^i \in C$  and  $v_{m_i}^i$  is the object's pixel mask.

The goal is to learn the values of some parameterized  $(\theta)$  function  $f$  such that:

$$y^i = f(x_i, \theta_i) \quad (2)$$

And be able to predict object (mound) locations  $\hat{y}$  for all new (unseen)  $x_i$

A loss function to optimize the prediction would be:

$$J(\theta) = \frac{1}{n} \sum_{i=1}^n l(\hat{y}^i, x_i, y^i; \theta) + \alpha \lambda(\theta), \quad (3)$$

where  $\alpha \in [0, \infty)$  controls the relative contribution of the penalty norm,  $\lambda$ , in other words, it controls the strength of the regularizer. No regularization when  $\alpha = 0$ , with larger values indicating more regularization.

$$\lambda(\theta) = \frac{1}{2} \|\theta\|_2^2 \quad \text{for } L_2 \text{ regularization and} \quad \lambda(\theta) = \|\theta\|_1 \quad \text{for } L_1 \text{ regularization.}$$

Due to the relatively small number of training samples, we proposed, as a starting point of our implementation, inspired by the work done by Souly et al.[23], an architecture with a generator to provide extra training samples and classifier as a discriminator in a Generative Adversarial Network. We evaluate the performance of each approach against the following metrics *Pixels Accuracy* i.e. ratio of properly classified pixels to total number of pixels, *Mean Pixel Accuracy (MPA)* is the average of the ratio of properly classified pixels to the number of pixels in a class and *Mean Intersection over Union (mIoU)* is the ratio of the intersection between the ground

truth and the predicted segmentation map to their union, averaged over all classes.

The paper follows the following structure: *section 3* discusses how the data was acquired and the preprocessing techniques such as feature engineering that were used followed with, in *section 4*, an explanation of the methods used to solve the problem at hand and finally in *section 5*, evaluation of our methodology and results are discussed.

### 3 DATA ACQUISITION & PRE-PROCESSING

#### 3.1 Data acquisition

In our paper, we are using data from HiRise (High Resolution Imaging Science Experiment) camera, onboard the Mars Reconnaissance Orbiter (MRO) spacecraft. We have the DEM (Digital elevation model) of the 'Firsoff' crater, which is an impact crater on mars. Two high resolution images of a specific area on the ground are taken from different camera angles, and then these stereo images are combined together to obtain a DTM (Digital Terrain Model). The photos were taken at an altitude of 272 kilometers above ground. The scale of the image is very high in resolution at 0.27 meter per pixel. This data has been made publicly available by NASA/JPL/University of Arizona. We decided to utilize a DEM format, since it is a three dimensional digital representation of a terrain with X,Y and Z coordinates. The Z coordinate (Elevation data) will help us find natural morphologies on the surface like mounds, craters and channels. We can use DEMs to generate additional features by performing slope and hillshade analysis. The use of DEMs also allows us to build 3D models of the surface. Moreover the distance between the sample points (spatial resolution) and the vertical resolution are very high in the HiRise DEM.

#### 3.2 Data preprocessing

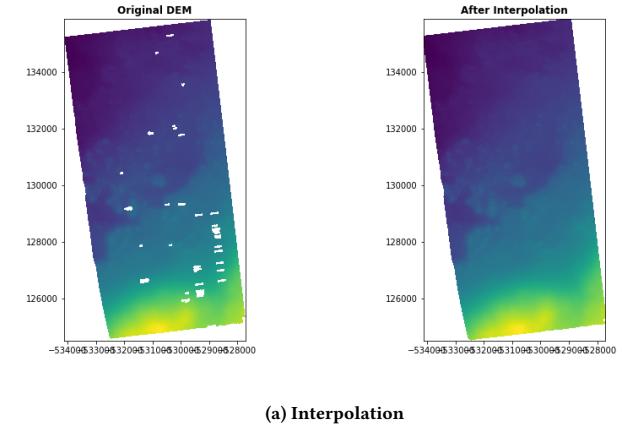
The first step in preprocessing was to fill in the No Data values. A no data value is present in the dataset when there is no depth reading available at that coordinate. This usually represents things such as backgrounds and borders, but also occurs when data is not available due to technical difficulties etc.

There are a few methods that can be used to treat the NoData values. We chose to perform automatic interpolation using the 'fillnodata' method from the python package 'rasterio'<sup>1</sup>. Manual interpolation is also possible, but requires expertise in domain. Sometimes interpolation can also cause geometric patterns and artifacts in the image which end up adding noise to our dataset.

The rasterio interpolation we used has two tunable parameters, Max\_search\_distance and smoothing\_iterations. Max\_search\_distance corresponds to the maximum number of pixels to search in all directions to find values to interpolate from (using inverse distance weighting). The smoothing\_iterations refers to the number of 3x3 average filters (passes to run on interpolated pixels) are applied to smooth out artifacts. We used a value of 60 for the max\_search\_distance and a value of 0 for the smoothing iterations. Those configurations allowed us to interpolate the area of interest. We did not want to interpolate the entire image because that would add a lot of noise to the dataset.

<sup>1</sup><https://rasterio.readthedocs.io/en/latest/api/rasterio.fill.html>

After interpolating, the DEM file is now ready for the tiling process.



(a) Interpolation

**Figure 2: The purpose of using interpolation is to fill the missing values in the original DEM, within the area of interest, as in the resulted image on the right.**

The DEM image we are using has a resolution of 6418 x 11339, and is a geoTIFF file. Each pixel corresponds to the depth measurement at a specific area on the ground. The maximum depth value is -1595.222, maximum value is -3050.866 and the No Data value is represented by -32767. The DEM has 75% valid values and a mean of -2594.830. A standard deviation value of 336.037 was observed.

#### 3.3 Feature engineering

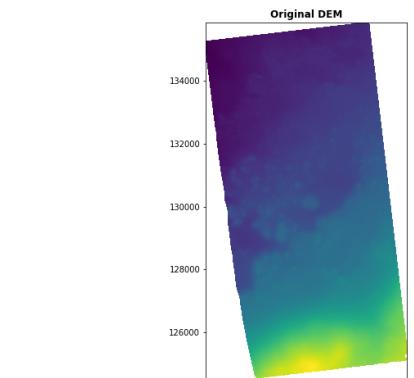
Adding channels or features to our data will help the segmentation neural network detect morphologies better. We can make use of additional features such as hillshades, slope and aspect analysis etc. We decided to generate hillshade and slope from our DEM file as additional features using gdal.DEMProcessing<sup>2</sup>.

Since the dataset is very small and consists of only one image, we decided to tile the image into XDIV x YDIV parts. Tiling will allow us to have multiple training samples, which in turn will improve the performance of our neural network. All of the morphologies to be detected have a polynomial, amoeba-like shape. There arise a few problems when using tiling, ex. A tile border can cut through a morphology, giving it a linear edge where there is none. Since we will also be tiling the testing data set, this should not be a problem. We created multiple sets with different (XDIV x YDIV) values to have more diversity in our dataset.

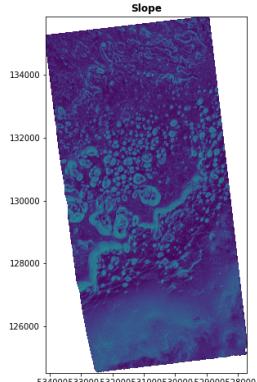
(XDIV x YDIV) = (2 x 4), (4 x 8), (8 x 16), (16 x 32), (32 x 64) Which gave us in total 2,728 tiles. We would split them into train, validation and test sets (We used some of the tiles for testing, since the actual test set does not have labels).

To tile the DEM, firstly, find the top-left corner of the image which has the lowest x-coordinate and the highest y-coordinate using gdal.GetGeoTransform.

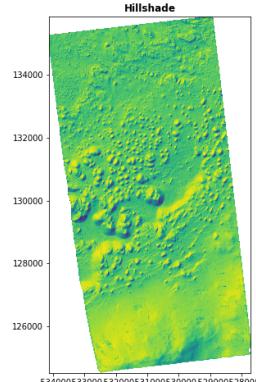
<sup>2</sup><https://gdal.org/python/osgeo.gdal-module.html#DEMProcessing>



(a) Origin Image



(b) Slope



(c) Hillshade

**Figure 3: Slope and Hillshade gave us a better visualization of the terrains, which is why we used them as additional channels for the segmentation model.**

$(xmin, ymax) = (gdal.GetGeoTransform()[0], gdal.GetGeoTransform()[3])$   
Then, get the size of the raster image which is the length of the coordination axis multiplied by the respected pixel size:

$xlen = xres * gdal.RasterXSize; xres = gdal.GetGeoTransform()[1]$   
 $ylen = yres * gdal.RasterYSize; yres = gdal.GetGeoTransform()[5]$   
After that, define the number of divisions on each axis. (xdiv, ydiv) which will make the size of the tile as follows:

$xsize = xlen/xdiv$   
 $ysize = ylen/ydiv$

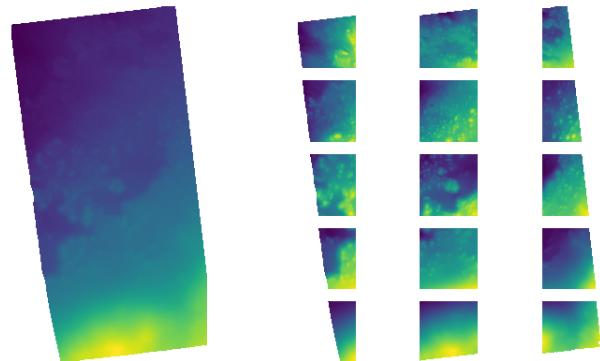
Next, define the strides (steps) of tiling as follows:

$xsteps = [xmin + xsize * i for i in range(xdiv+1)]$   
 $ysteps = [ymax - ysize * i for i in range(ydiv+1)]$

Finally, using `gdal.Warp` for each step in the steps vectors, passing these parameters:

- The wanted name of the output.

- The input DEM.
- `outputBounds=(xsteps[i], ysteps[i], xsteps[i+1], ysteps[i+1])`



(a) Original DEM

(b) Tiles

**Figure 4: Splitting the original DEM into a set of tiles to. In this example the original DEM is split into 12 equal-sized tiles using the tiling algorithm explained in this paper, with  $xdiv = 3$  (corresponds to the number of resulting columns) and  $ydiv = 4$  (corresponds to the number of resulting rows)**

### 3.4 Annotation

To annotate our dataset we need to extract the mounds from our DEM using the shape file that describe the mounds. We use the `geopandas.read_file()`<sup>3</sup> to read the shape file. We reproject the labels coordinate system to that of the original DEM which is stored in the ‘meta’ field of the DEM data.

We mask the labels using `rasterio.mask.mask()`<sup>4</sup>, which takes the dataset, the geometry of the mounds (stored in the shape file) and the noDataValue of the DEM. The annotations would be:

- ‘0’ For non-mounds points.
- ‘1’ For mounds points.

We also tried treating the invalid data as an independent class but the results were worsened so we chose only those two classes.

## 4 MODEL IMPLEMENTATION

### 4.1 Methodology

For our task, we first have to train an image segmentation network to extract the mounds from the DEM tiles. We then train the same image segmentation network with additional synthetic training data generated by a generative network, and observe if the additional augmentation data improves the accuracy of the segmentation. For the proposed technique, we need two neural network architectures, one for image segmentation and other for image generation. We chose to try two architectures for the segmentation of

<sup>3</sup>[https://geopandas.org/en/stable/docs/reference/api/geopandas.read\\_file.html](https://geopandas.org/en/stable/docs/reference/api/geopandas.read_file.html)

<sup>4</sup><https://rasterio.readthedocs.io/en/latest/api/rasterio.mask.html>

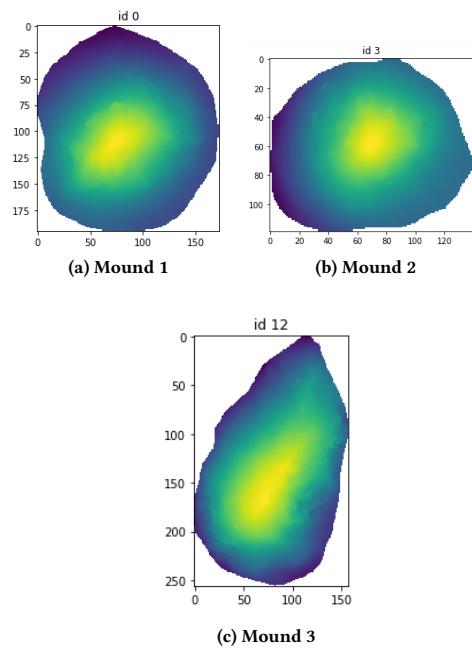


Figure 5: Mounds samples from the training dataset.

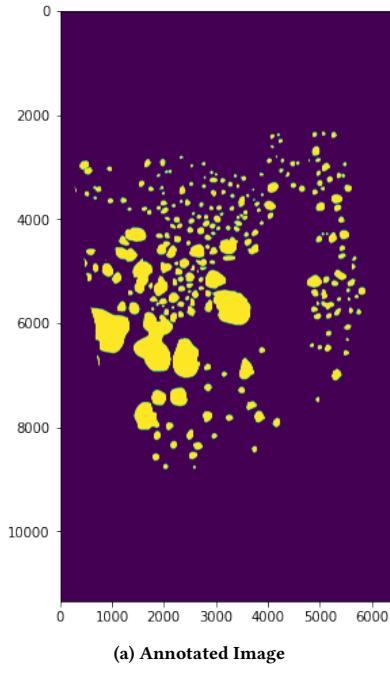


Figure 6: The annotated image (mask) is obtained by overlapping the coordination of the annotated mounds (provided in the shape file of the DEM) with the original DEM.

mounds, U-Net and FPN and later compare them. GAN was chosen for the image and label generation.

## 4.2 U-NET

U-NET [33] is deep convolutional network, developed for semantic segmentation of biomedical images. It is called a "U-NET" because its architecture has a "U" shape to it. It learns to segment images in an end to end setting, meaning it gets raw input image in, and outputs a segmentation map.

Image is fed into the input layer, then the data gets propagated through the network among all the possible paths. Most operations are 3x3 convolutions, followed by a non linear activation functions (ReLU). During the 3x3 convolutions, one pixel order is lost, this allows the network to process large image in individual tiles. Next operation is max pooling, it reduces the X,Y size of the feature map, illustrated by a downward arrow. Max pooling propagates max activation from each 2x2 window to the next feature map, preserving all the important features. Sequence of convolutions and max pooling results in 'spatial contraction', which means we gradually increase the "what" in the image at the same time decrease the "where". This allows for a kind of generalization of the image structures.

UNET also has an expansion path upwards, after the contraction. This creates a high resolution segmentation map. The expansion path has a sequence of up convolutions and concatenations, which correspond to high resolution features from the contracting path. The output we acquire is a segmentation map with two channels, one channel for foreground and one channel for the background. In our case, the foreground will be the labelled mound shapes.

We chose to use UNET for our segmentation architecture because it is observed that they outperform other segmentation architectures when the dataset size is low, i.e. few training images. Moreover, they are faster than other networks and result in faster convergence too. They also work well when two images of the same class are touching each other. The challenges for our dataset were fuzzy borders, low contrast and lack of training images.

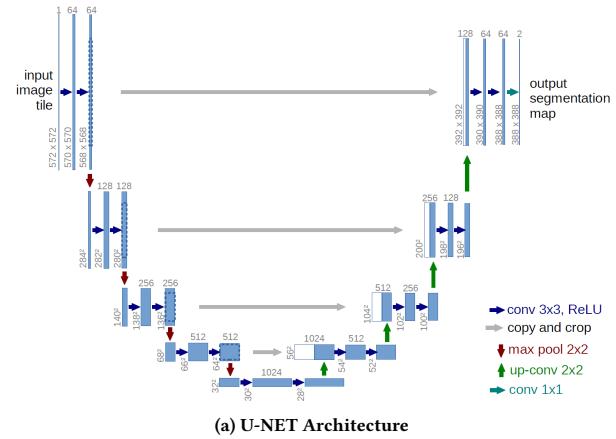


Figure 7: U-NET architecture (<https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/>)

### 4.3 FPN

FPN [34] stands for Feature Pyramid Network. It is a feature extraction neural network, commonly used for object detection. In image segmentation tasks, sometimes a model fails to detect objects that are too small or large compared to the training data sizes. FPN was introduced to tackle the problem of detecting objects in different scales.

FPN has two pathways, a bottom up and a top down pathway. The bottom up pathway uses CNNs for feature extraction. As we keep going up, the spatial resolution of the image decreases, i.e. only high level structures are preserved, that is why the semantic value or the information increases and the resolution decreases as we go up in the bottom up path. The bottom up path makes use of the ResNext architecture. Each layer we move up, the spatial dimension of feature maps is reduced by half.

After the bottom up path, FPNs have a top down path that takes the low resolution, high semantic feature maps and passes it down, increasing the image resolution. This results in a semantically rich layer at the end of the top down path. The reconstructed layers might contain high semantic information but the location of the objects is poor. This happens due to the upsampling and downsampling applied in the bottom up and top down path. Therefore, we add lateral connections between each of the pathways, this will help the detector predict locations of the objects more precisely. These lateral connections between the two paths are often referred to as skip connections.

For segmentation, a 2D sliding window is passed over the feature maps and generates a object segment, this is done for all the feature maps. Finally we combine all the object segments at different scale and create our final mask prediction.

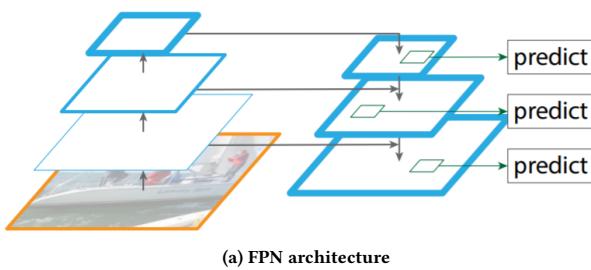


Figure 8: FPN architecture (FPN architecture)

### 4.4 Pix2Pix

GAN [35] stands for Generative Adversarial network. They are unsupervised neural networks used for image generation and manipulation. GANs can capture and copy variation in a dataset, and use it to produce new synthetic images.

Pix2Pix is a version of conditional GANs or cGANs. This is different from normal GAN because they provide control over what image is being generated i.e. they allow us to generate an image of a given class. We need to use cGANs because we need to generate additional image data and label data as well. cGANs are generally used for image to image translation.

The architecture consists of two models, a generator network and a discriminator network. Both of these models act in an adversarial fashion, i.e against each other. The generator given an input image and is tasked with generating a translated version of the image, whereas the discriminator is given an input image and real or generated paired image and is tasked with determining if the paired image is real or generated. The generator learns from distribution of classes and is supposed to fool the discriminator into thinking that the image is produced was real. As the generator gets better, discriminator also gets better. The whole network converges when discriminator can no longer differentiate between real images and the ones generated by the generator network. We also use a U-NET as a generator in our Pix2Pix model.

A cGAN can be defined by the mathematical equation

$$\min_G \max_D V(D, G) = E_x p_{data(x)} [\log D(x|y)] + E_z p_z(z) [\log(1 - D(G(z|y)))] \quad (4)$$

where,

- G = Generator
- D = Discriminator
- x = sample from real data
- z = sample from generator
- y = Auxiliary information
- $p_{data}(x)$  = distribution of real data
- $D(x)$  = Discriminator network
- $G(z)$  = Generator network

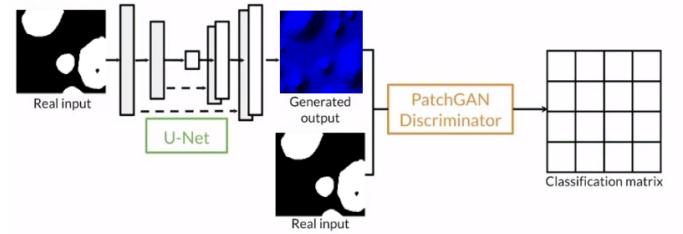


Figure 9: Pix2Pix architecture

### 4.5 Training

For training the U-Net and the FPN we used 2728 images in total. Of which, 1632 were used for the training dataset and 546 for the testing and validation set each. These images were obtained after the feature engineering step where we combine channels of original DEM, slope, hillshade. The dimension of the images is 224 x 192 x 3. All the images have 3 channels.

For the FPN network we used a pretrained model with se\_resnext\_50 [36] encoder weights pretrained on ImageNet dataset with 25M parameters from the segmentation-models-pytorch [37] library. And for the U-Net we used a pretrained resnet34[38] encoder with

weights trained on ImageNet dataset. We trained both of these models on our dataset.

We treated our segmentation problem as a binary classification problem since the mask we predict and the ground truth mask are both binary. The predicted mask had values ranging from 0 to 1. So we chose sigmoid as our activation function and 0.4 as the threshold. This value was chosen because it provided the best results compared to other values. For the training segmentation masks, we used images of 224 x 192 where white pixels represent mound shape and black pixels represent the background. The results were obtained after training the network for 10 epochs, with a batch size of 8. The learning rate used was 1e-4.

For the GAN, we used pix2pix as our base architecture, to produce the synthesized images. The training set consisted 512 images. Out of which the test set contained 112 samples and the training set consisted of 400 samples. All images were 256 x 256 x 3 with three channels corresponding to DEM, slope and hillshade feature respectively. The model takes a lot of time to converge, even with the small dataset. The generator model had over 41M tunable parameters. The model was trained for 120 epochs and generated 128 synthetic images with labels as the output. The generation was done on the labels of the test dataset.

The output of the GAN and ground truth had a Bhattacharyya distance of 0.4610 and a mutual information score of 0.32 but the predicted image visually appeared very similar to the real input image.

The images had very natural looking textures and shadows. Some outputs had a grainy appearance or presence of some artifacts such as repeating patterns. It was observed that the artifacts and artificial looking patterns started slowly disappearing as the training went on. In some cases, it was virtually impossible to distinguish if the prediction was a real image or fake image.

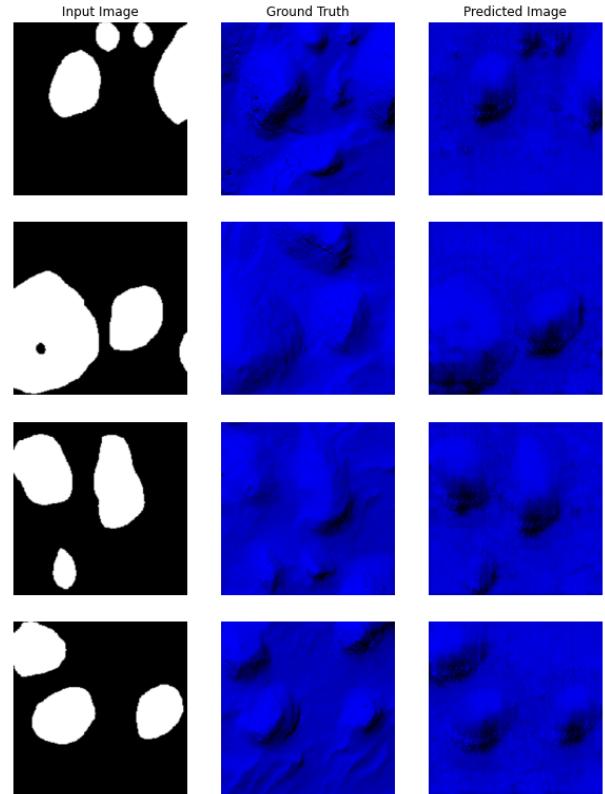


Figure 10: GAN generated output results

## 5 EVALUATION

### 5.1 Results

First of all, the evaluation of the generated synthetic data by GAN, was based on the same strategies used by the authors in the pix2pix article [39]

- The first strategy we used imply using human scoring. We iterated over the generated mounds images and interpret the quality of the generated data based on human perception. Fig.11 shows a sample of 3 images generated by our GAN model.

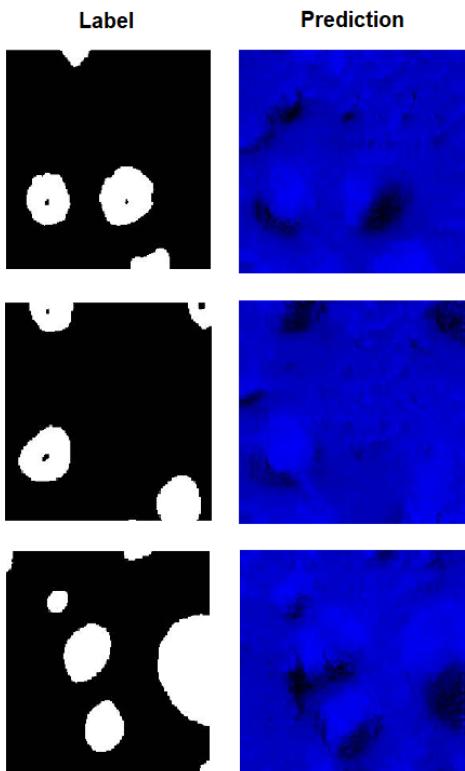


Figure 11: Examples of generated augmented data

- The second strategy consist of rating the performances of a semantic segmentation network model while segmenting the real data and while segmenting the synthetic generated data. The model used in our case is U-Net pre-trained model.

In this context, we will investigate whether the U-net model and the FPN are able to segment data, and whether its performance improves by applying GAN for data augmentation. The segmentation quality have been evaluated using Intersection over Union (IoU) with Dice Loss function. This loss primarily measures the overlap between samples. the measure could range from 0 to 1 where the coefficient 1 indicates perfect and complete overlap.

FPN model and U-net model trained with the real data, showed good some results.

However, in the case of training these two architectures with both real and synthetic data set, showed significantly worse quality of segmentation compared to the use of real data only As shown in Fig.12.

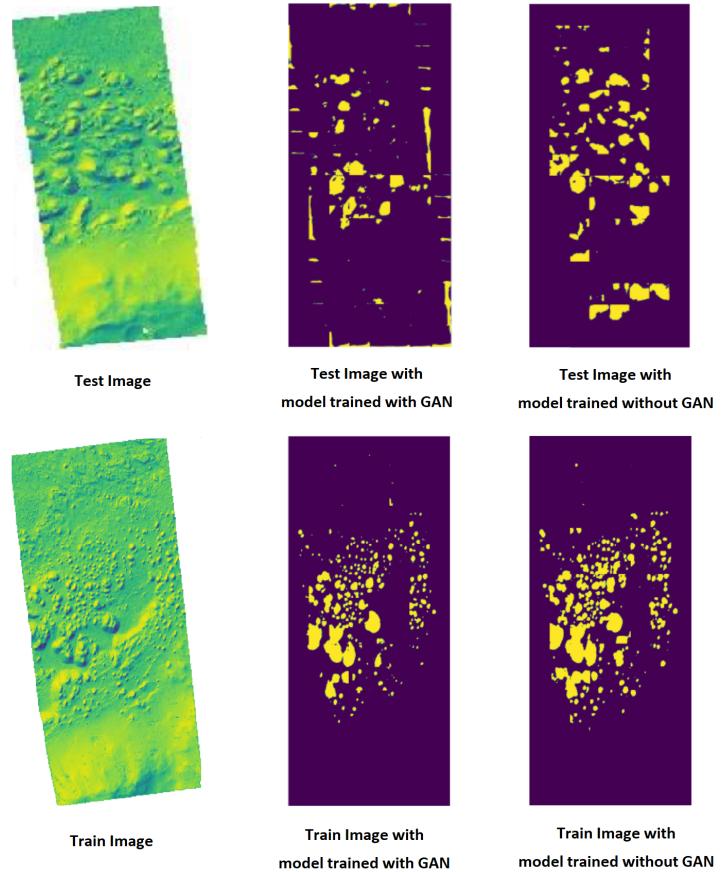


Figure 12: Comparison on predicted mask

To compare both U-Net and FPN models performance, we tested the trained models with augmented data and real data using F1 score as a metric. Results are shown in the table 1.

	U-Net without GAN	U-Net with GAN	FPN without GAN	FPN with GAN
Accuracy	0.95	0.97	0.97	0.94
Precision	0.77	0.80	0.75	0.78
Recall	0.77	0.68	0.83	0.73
FDR	0.23	0.20	0.25	0.23
FOR	0.02	0.03	0.02	0.03
F1-Score	0.77	0.74	0.79	0.75
IOU-Score	0.81	0.81	0.84	0.80

Table 1: Test metrics results

## 5.2 Discussion

we note that, The geographical morphologies are easily learned by GANs, and they are successfully able to reproduce very natural looking patterns features. Making them an excellent way to generate more data close to real mounds images. Besides, the usage of two GAN models of pix2pix architecture to generate the synthetic images gave us more variance in our dataset, as both the models produced slightly different images. The final GAN synthesized additional data comprised of both these models' output.

However, despite not having a testing mask, we can realize by observing the testing results that the segmentation with real and synthesized data sets achieved better segmentation quality than segmentation on the real data set.

In fact, GAN confuses the network, we can discern for example that the model trained with GAN predicts several mounds as a line shaped especially for the case of no data areas which not correct in reality.

Finally, although we got better precision, the recall gets worse drastically which illustrate the fact that the model doesn't know how to recognize the mounds any more, in other words GAN ruin what the U-net and FPN has learned from the actual tiles.

## 6 CONCLUSION

In this study, we investigate whether artificially generated images of the mounds on Mars can improve the performance of a segmentation model.

In this regard, we proposed and developed an FPN based network a U-Net based network for solving mound's segmentation problem. Hence, there is still the issue of training our segmentation model itself based on a limited data set. To facilitate this, we performed some transformations to the images in the original data set. Then, we leverage an existing conditional generative adversarial network (cGAN) model pretrained on large-scale data set to generate realistic synthesized data set, following the concept of transfer learning. For this purpose, we used pix2pix pre-trained to translate labels to facades.

Afterword, we compared the segmentation quality of the U-Net trained model with and without augmentation.

The results showed that models trained with real data sets achieved better segmentation quality than the one trained on both the real and synthesized data set.

In the end, We found out that adding GAN augmented data doesn't actually help in segmentation. As a matter of fact, The performance is significantly better without using GAN images, hence it will be better to rely on the real data to make better predictions.

## REFERENCES

- [1] Yuval Noah Harari. 2015. *Sapiens: A Brief History of Mankind*. Harper, 195 Broadway New York, NY 10007 USA.
- [2] David A. Forsyth and Jean Ponce. 2012. *Computer Vision - A Modern Approach, Second Edition*. Pitman, Hoboken, New Jersey, 1–91. ISBN: 978-0-273-76414-4.
- [3] Ying Tan. 2016. Chapter 11 - applications. In *Gpu-Based Parallel Implementation of Swarm Intelligence Algorithms*. Ying Tan, editor. Morgan Kaufmann, San Francisco, CA, 167–177. ISBN: 978-0-12-809362-7. doi: <https://doi.org/10.1016/B978-0-12-809362-7.5.0011-X>.
- [4] Rajeshwar Dass and Swapna Devi. 2012. Image segmentation techniques. *International Journal of Electronics & Communication Technology*, 3, 1. ISSN: 2230-7109 (Online).
- [5] Dzung L. Pham, Chenyang Xu, and Jerry L. Prince. 2000. Current methods in medical image segmentation. *Annual Review of Biomedical Engineering*, 2, 1, 315–337. PMID: 11701515. doi: 10.1146/annurev.bioeng.2.1.315. <https://doi.org/10.1146/annurev.bioeng.2.1.315>.
- [6] Ming Zeng, Youfu Li, Qinghao Meng, Ting Yang, and Jian Liu. 2012. Improving histogram-based image contrast enhancement using gray-level information histogram with application to x-ray images. *Optik*, 123, 6, 511–520. ISSN: 0030-4026. doi: <https://doi.org/10.1016/j.ijleo.2011.05.017>.
- [7] Jamil A. M. Saif, Mahgoub H. Hammad, and Ibrahim A. A. Alqubati. 2016. Gradient based image edge detection. *IACSIT International Journal of Engineering and Technology*, 8, 3. doi: 10.7763/IJET.2016.V8.876.
- [8] Mohamed Abd Elaziz, Siddhartha Bhattacharyya, and Songfeng Lu. 2019. Swarm selection method for multi-level thresholding image segmentation. *Expert Systems with Applications*, 138, 112818. ISSN: 0957-4174. doi: <https://doi.org/10.1016/j.eswa.2019.07.035>.
- [9] Essam H. Houssein, Marwa M. Emam, and Abdelmgeid A. Ali. 2021. An efficient multilevel thresholding segmentation method for thermography breast cancer imaging based on improved chimp optimization algorithm. *Expert Systems with Applications*, 185, 115651. ISSN: 0957-4174. doi: <https://doi.org/10.1016/j.eswa.2021.115651>.
- [10] Zhuang Cheng and Jianfeng Wang. 2020. Improved region growing method for image segmentation of three-phase materials. *Powder Technology*, 368, 80–89. ISSN: 0032-5910. doi: <https://doi.org/10.1016/j.powtec.2020.04.032>.
- [11] Nagaraj Jothiaruna, Joseph K. Abraham Sundar, and Balasubramanian Karthikeyan. 2019. A segmentation method for disease spot images incorporating chrominance in comprehensive color feature and region growing. *Computers and Electronics in Agriculture*, 165, 104934. ISSN: 0168-1699. doi: <https://doi.org/10.1016/j.compag.2019.104934>.
- [12] Marie Lachaize, Sylvie Le Hégarat-Mascle, Emanuel Aldea, Aude Maitrot, and Roger Reynaud. 2018. Evidential split-and-merge: application to object-based image analysis. *International Journal of Approximate Reasoning*, 103, 303–319. ISSN: 0888-613X. doi: <https://doi.org/10.1016/j.ijar.2018.10.008>.
- [13] Lifeng Liu and Stan Sclaroff. 2004. Deformable model-guided region split and merge of image regions. *Image and Vision*

- [14] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. 1988. Snakes: active contour models. *international Journal of Computer Vision*, 1, 321–331. doi: <https://doi.org/10.1007/BF00133570>.

[15] Xianghai Wang, Yu Wan, Rui Li, Jinling Wang, and Lingling Fang. 2016. A multi-object image segmentation c-v model based on region division and gradient guide. *Journal of Visual Communication and Image Representation*, 39, 100–106. ISSN: 1047-3203. doi: <https://doi.org/10.1016/j.jvcir.2016.05.011>.

[16] Man Yan, Jianyong Cai, Jiebing Gao, and Lili Luo. 2012. K-means cluster algorithm based on color image enhancement for cell segmentation. In *2012 5th International Conference on BioMedical Engineering and Informatics*, 295–299. doi: [10.1109/BMEI.2012.6513157](https://doi.org/10.1109/BMEI.2012.6513157).

[17] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2016. Faster r-cnn: towards real-time object detection with region proposal networks. (2016). arXiv: 1506.01497 [cs.CV].

[18] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: convolutional networks for biomedical image segmentation. (2015). arXiv: 1505.04597 [cs.CV].

[19] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. 2016. V-net: fully convolutional neural networks for volumetric medical image segmentation. (2016). arXiv: 1606.04797 [cs.CV].

[20] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. 2018. Mask r-cnn. (2018). arXiv: 1703.06870 [cs.CV].

[21] Francesco Visin, Marco Ciccone, Adriana Romero, Kyle Kastner, Kyunghyun Cho, Yoshua Bengio, Matteo Matteucci, and Aaron Courville. 2016. Reseg: a recurrent neural network-based model for semantic segmentation. (2016). arXiv: 1511.07053 [cs.CV].

[22] Pauline Luc, Camille Couprie, Soumith Chintala, and Jakob Verbeek. 2016. Semantic segmentation using adversarial networks. (2016). arXiv: 1611.08408 [cs.CV].

[23] Nasim Souly, Concetto Spampinato, and Mubarak Shah. 2017. Semi supervised semantic segmentation using generative adversarial network. In *2017 IEEE International Conference on Computer Vision (ICCV)*, 5689–5697. doi: [10.1109/ICCV.2017.606](https://doi.org/10.1109/ICCV.2017.606).

[24] Wei-Chih Hung, Yi-Hsuan Tsai, Yan-Ting Liou, Yen-Yu Lin, and Ming-Hsuan Yang. 2018. Adversarial learning for semi-supervised semantic segmentation. (2018). arXiv: 1802.07934 [cs.CV].

[25] [n. d.] <https://github.com/hindupuravinash/the-gan-zoo>. accessed: 18-11-2021.

[26] Suman Paneru and Idris Jeelani. 2021. Computer vision applications in construction: current state, opportunities & challenges. *Automation in Construction*, 132, 103940. ISSN: 0926-5805. doi: <https://doi.org/10.1016/j.autcon.2021.103940>.

[27] BigdataAILab. 2021. What is semantic segmentation, instance segmentation, panoramic segmentation? (April 2021). <https://becominghuman.ai/what-is-semantic-segmentation-instance-segmentation-panoramic-segmentation-3bb03856c12>. accessed: 18-11-2021.

[28] Alec Radford, Luke Metz, and Soumith Chintala. 2016. Unsupervised representation learning with deep convolutional generative adversarial networks. (2016). arXiv: 1511.06434 [cs.LG].

[29] Ryan J. Spick, Peter Cowling, and James Alfred Walker. 2019. Procedural generation using spatial gans for region-specific learning of elevation data. In *2019 IEEE Conference on Games (CoG)*, 1–8. doi: [10.1109/CIG.2019.8848120](https://doi.org/10.1109/CIG.2019.8848120).

[30] Nikolay Jetchev, Urs Bergmann, and Roland Vollgraf. 2017. Texture synthesis with spatial generative adversarial networks. (2017). arXiv: 1611.08207 [cs.CV].

[31] Christopher Bowles, Liang Chen, Ricardo Guerrero, Paul Bentley, Roger Gunn, Alexander Hammers, David Alexander Dickie, Maria Valdés Hernández, Joanna Wardlaw, and Daniel Rueckert. 2018. Gan augmentation: augmenting training data using generative adversarial networks. (2018). arXiv: 1810.10863 [cs.CV].

[32] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. 2018. Progressive growing of gans for improved quality, stability, and variation. (2018). arXiv: 1710.10196 [cs.NE].

[33] O. Ronneberger, P. Fischer, and T. Brox. 2015. U-net: convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)* (LNCS). Volume 9351. (available on arXiv:1505.04597 [cs.CV]). Springer, 234–241. <http://lmb.informatik.uni-freiburg.de/Publications/2015/RFB15a>.

[34] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. 2016. Feature pyramid networks for object detection. *CoRR*, abs/1612.03144. arXiv: 1612.03144. <http://arxiv.org/abs/1612.03144>.

[35] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial networks. (2014). arXiv: 1406.2661 [stat.ML].

[36] Saining Xie, Ross B. Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. 2016. Aggregated residual transformations for deep neural networks. *CoRR*, abs/1611.05431. arXiv: 1611.05431. <http://arxiv.org/abs/1611.05431>.

[37] Pavel Yakubovskiy. 2020. Segmentation models pytorch. [https://github.com/qubvel/segmentation\\_models.pytorch](https://github.com/qubvel/segmentation_models.pytorch). (2020).

[38] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep residual learning for image recognition. *CoRR*, abs/1512.03385. arXiv: 1512.03385. <http://arxiv.org/abs/1512.03385>.

[39] Tinghui Zhou Phillip Isola Jun-Yan Zhu and Alexei A Efros. 2017. Image-toimage translation with conditional adversarial networks. In *Computer Vision and Pattern Recognition (CVPR)*.

## A PHASE CONTRIBUTION

- Phase One Douglas D. Agbeve
  - Phase Two Salim Fares
  - Phase Three Aditya Handrale
  - Phase Four Seif Eddine Idani