Remote Sensing and Satellite Imagery

Mainstream - Spring 2024

Project Report

Team 9

Name	Sec	B.N
Ahmed Hosny	1	2
Ahmed Sabry	1	4
Basma Hatem Farid	1	15
Zeinab Moawad Fayez	1	28

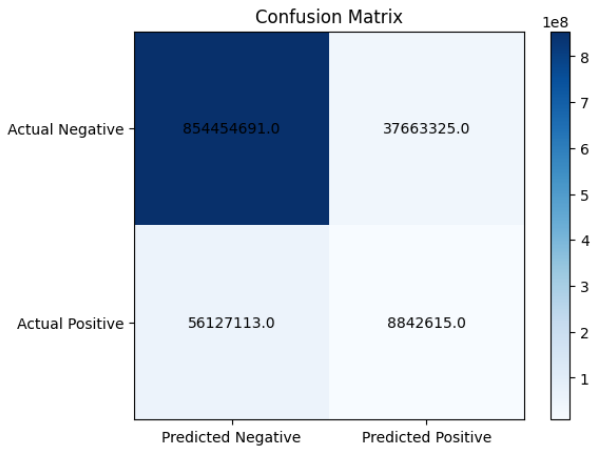
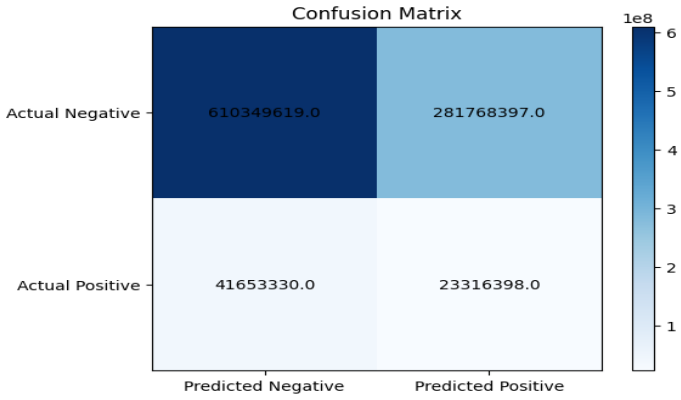
Supervised by:

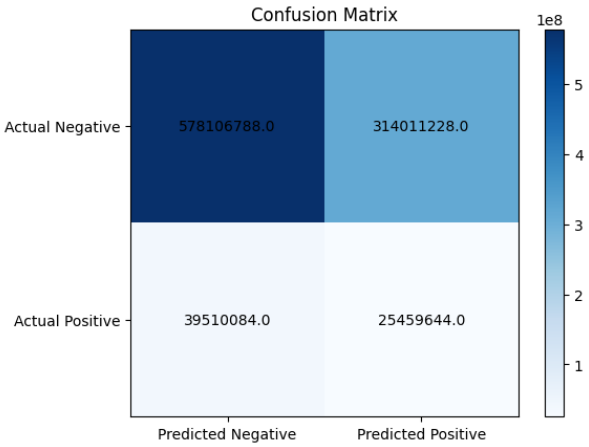
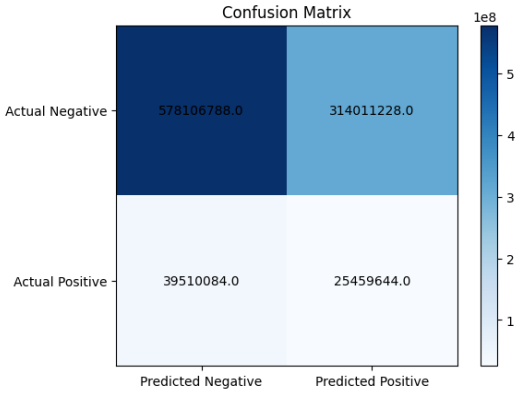
Eng: Muhammad Sayed

Data Set devised by 70 train and 30 test

Classical approach:

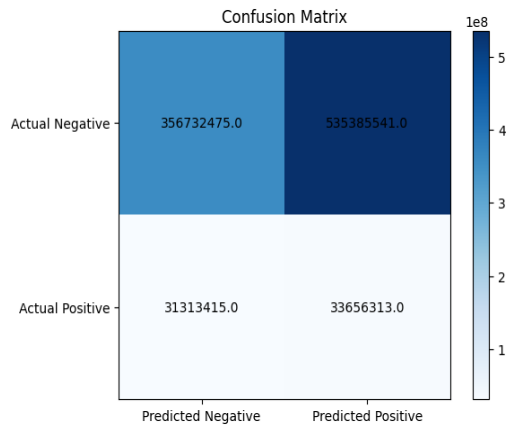
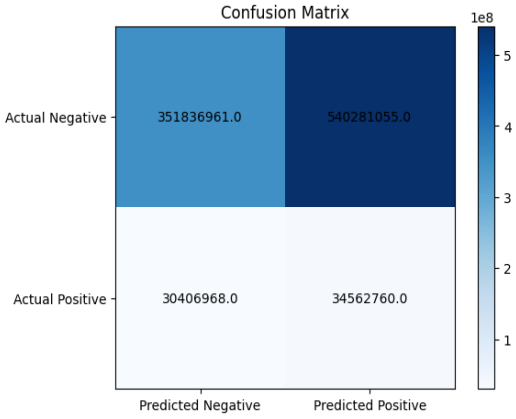
- Image differencing:
  - Simple one:

Threshold	preprocessing	results
78	NO	Confusion Matrix:  Accuracy: 0.9020043474718239 Precision: 0.19013947465635572 Recall: 0.1361036173646902 F1 Score: 0.15864654876972792 Mean Jaccard Index: 0.1028530606521822
45	Yes	Confusion Matrix:  Accuracy: 0.6620772452394919 Precision: 0.07642595888792164 Recall: 0.3588809545270068 F1 Score: 0.12601601413205804 Mean Jaccard Index: 0.07652192968089663

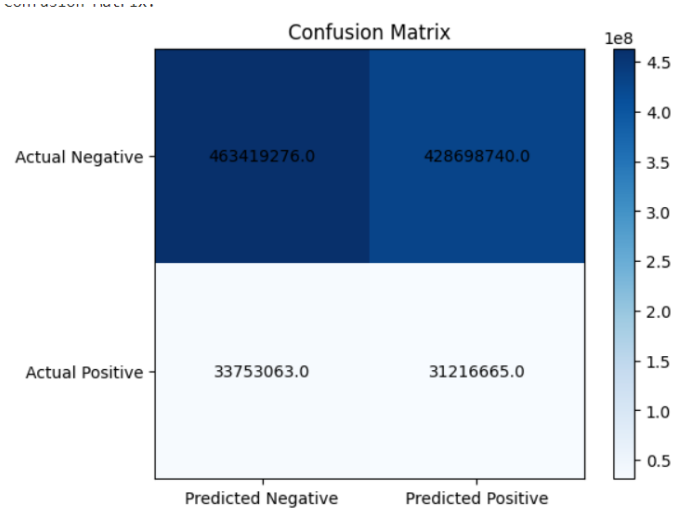
35	Yes	Confusion Matrix:  Accuracy: 0.6306281067579944 Precision: 0.07499802221617412 Recall: 0.3918693333609154 F1 Score: 0.12590053520838412 Mean Jaccard Index: 0.07768617112462077
25	Yes	Confusion Matrix:  Accuracy: 0.6306281067579944 Precision: 0.07499802221617412 Recall: 0.3918693333609154 F1 Score: 0.12590053520838412 Mean Jaccard Index: 0.07768617112462077

- Another Way:

Threshold	Preprocessing	results
400	No	Accuracy: 0.4078923697930041 Precision: 0.05914558439492221 Recall: 0.5180306896159392 F1 Score: 0.10616939486761616 Mean Jaccard Index: 0.08084878138885547

		Confusion Matrix: 
350	No	Accuracy: 0.4037244478600282 Precision: 0.06012547947480308 Recall: 0.53198252576954 F1 Score: 0.10804010130182568 Mean Jaccard Index: 0.08077700860072906 

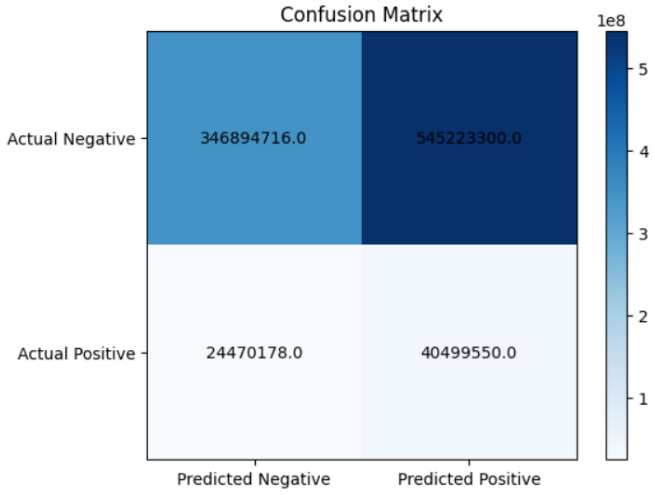
- Post Classification (number of clusters **2**):
  - Total: 957087744.0
  - Accuracy: 0.5168135775438369
  - Precision: 0.06787479754021286
  - Recall: 0.48048015531171684
  - F1 Score: 0.1189466534195016
  - Mean Jaccard Index: 0.05359536905158464



- Pixel wise classification:
  - Using Random Forest Classifier data split ratio = (0.1,0.05)

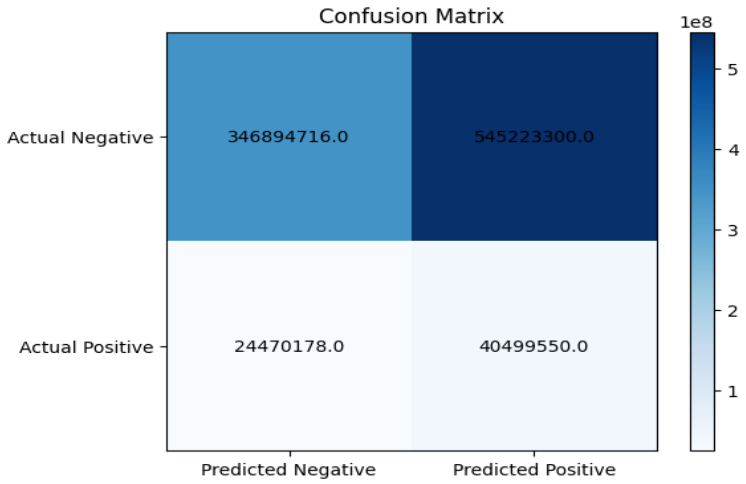
    ```
    0.11058043292055691
    ```

- CVA (Change Vector Analysis):
  - By first channel:
    - Accuracy: 0.40476358456012157
    - Precision: 0.06914456214231697
    - Recall: 0.6233603132831339
    - F1 Score: 0.12448136453156225
    - Mean Jaccard Index: 0.060722311740071026

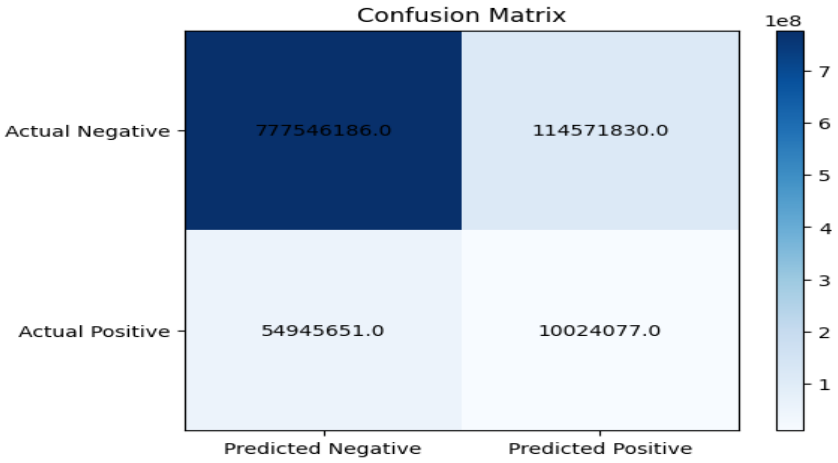
    

  - Multi-channel:

- Total: 957087744.0
  Accuracy: 0.40476358456012157
  Precision: 0.06914456214231697
  Recall: 0.6233603132831339
  F1 Score: 0.12448136453156225
  Mean Jaccard Index: 0.060722311740071026
  Confusion Matrix:



- Masking:
  - Accuracy: 0.8228819854159578
    Precision: 0.08045269897991111
    Recall: 0.1542884249107523
    F1 Score: 0.10575837756669346
    Mean Jaccard Index: 0.12861187774206656
    Confusion Matrix:



Deep Learning approach:

- Preprocessing
  - In the transfor mer we make many types of augmentation like
    - Horizontal flip
    - Vertical flip
    - Convert to tensor
    - Rotation with random numbers
    - Resize
    - Normalization
- ARCH
  - We impalement paper which use Unet and train it using GANs
    the Arch

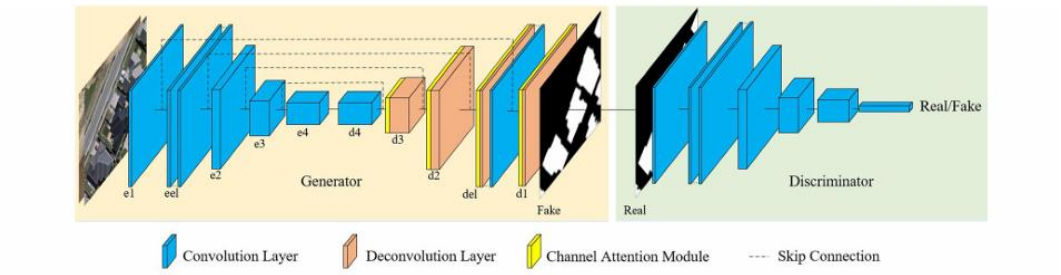


Fig. 2 Illustration of the proposed CSA-CDGAN. This model contains two parts, a generator (left) and discriminator (right)

  -
    - LINKE FOR THE PAPER: https://link.springer.com/article/10.1007/s00521-022-07637-z
  - First the Generator
    - We use Unet take input as the two RGB image concatenated so the input channel is 6
    - We apply 4 convolution layers and 3extra layers then at the bottom layer we convert the features to 100 then apply up sampling until get the output as 1 channel
    - Then apply threshold at 0.5 as the output is gray scale and we need to convert the output to binary
    - We use self-attention layer instead of copy and crop to take features from the encoder to decoder
  - Second the Discriminator
    - We use 3 convolution layer and 3 extra layer and get features of the output from the generator the calculate the losses to tune the generator and discriminator
- TUNNING
  - We use BCE Loss to calculate the losses from the discriminator and L1loss + DicLoss to tune the generator
  - We multiply the losses from the generator with 200 as it is the target from the problem to generate images and instead of train the generator first many times then the discriminator
- FAILED TRIES

- we try to make the architecture extract more features and make it until 512 in the bottom instead of 100 but sadly it stuck in 78% as Jaccard metric score
- We add another layer in the generator and make the bottom features 512 but sadly it stuck in 77%
- We try the paper implementation in the generator and make the bottom features 100 it gets 81% but we see that it biased to black image as it gets 91 on only black images and 50 on images with changes
- We add another layer in the generator and make the bottom features 100 but sadly it stuck in 79%

Model name	test	One	zero
onserver_100_copy	81.62632677165297	0.5008637989078955	0.9519230769230769
output_80_on_75_best	80	-	-
78 test	78	0.5216761444479356	0.9519230769230769
output_80_without_removing_75_25	80	-	-
output_best_model_change_schedular	80	-	-
Netg 2	78	0.57736530996149	0.9194711538461539

We try another paper implementation using tomogan which take the image as gray scale and try to train the generator for 4 epochs then train the discriminator for 2 epochs but sadly it gets 66.5% Jaccard metric the link for the paper https://arxiv.org/pdf/1902.07582v5