

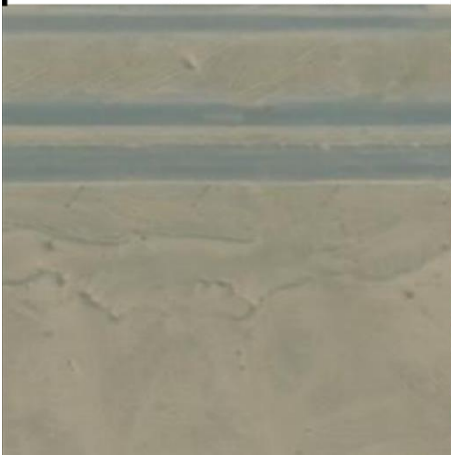
Team 5

Name	Section	BN
Youssef Said Ibrahim Rabie	2	40
Ahmed Alaa	1	6
Abdelrahman Mohamed Salem	2	1
Ahmed Fawzy Mohamed Ibrahim	1	7

Zero Prediction (Show how good is the dataset)

We noticed there are many black masks in the truth ground images so we made a model that will always produce black mask no matter what the image looks like. Then ran this model on the whole dataset. We got an average Jaccard index of **0.664**. this means that **66.4%** of the truth ground images in the whole dataset are just black images. That's why it will affect our deep learning in a way that it will favor the marking a pixel to be black rather than making it white.

A (before)



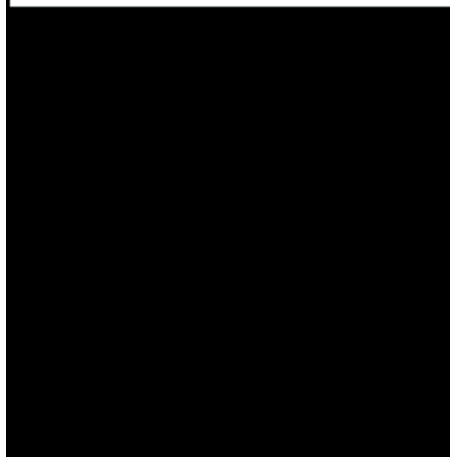
B (After)



Δ (truth ground)



Zero Prediction



PCA_Kmeans (Traditional Method)

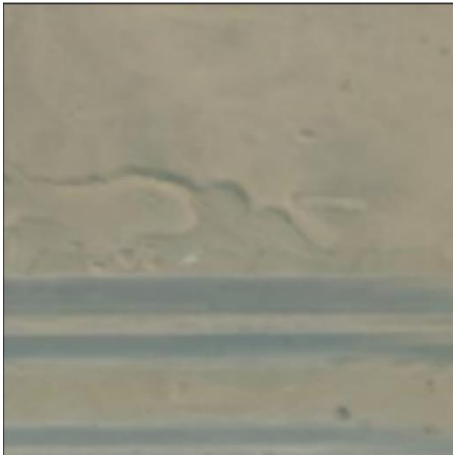
We followed these steps from the paper named “**Unsupervised Change Detection in Satellite Images Using Principal Component Analysis and k-Means Clustering**” by “**Turgay Celik**” in 2009. We added extra pre and post processing steps to the algorithm introduced to be able to not only detect changes but detect the buildings the appeared. This Algorithm is an unsupervised image processing algorithms, so it does not require any training, so we didn't have to split dataset (we used the whole dataset as validation and test dataset).

One of the main parameters of this algorithm is a variable called **h**. We will see its main usage later in the document.

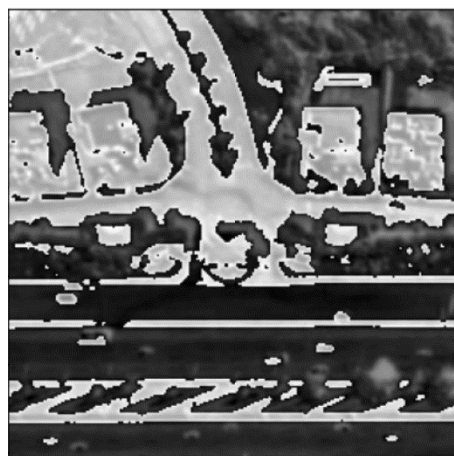
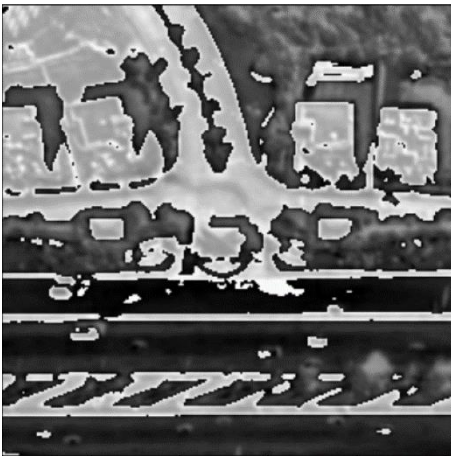
1. We have 2 RGB images A and B, where A represents the before image and B represents the after image.



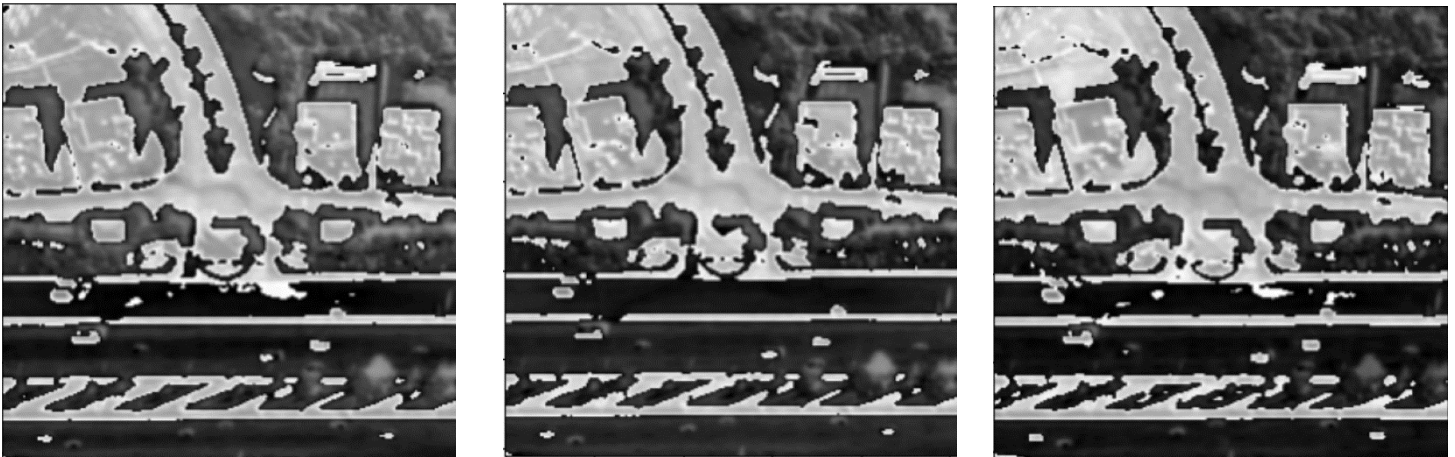
2. Applied **Gaussian Blur** with Kernal size 3x3 (Pre-processing to remove any noisy pixels in the image).



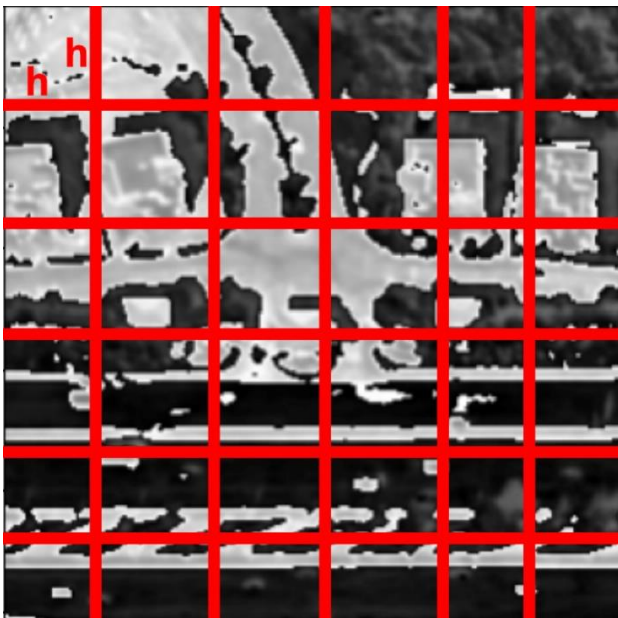
3. Computed the **absolute difference** between the channels of the 2 images.



4. **Resized** the image so that both width and length of the **absolute difference image** is divisible by **h**.

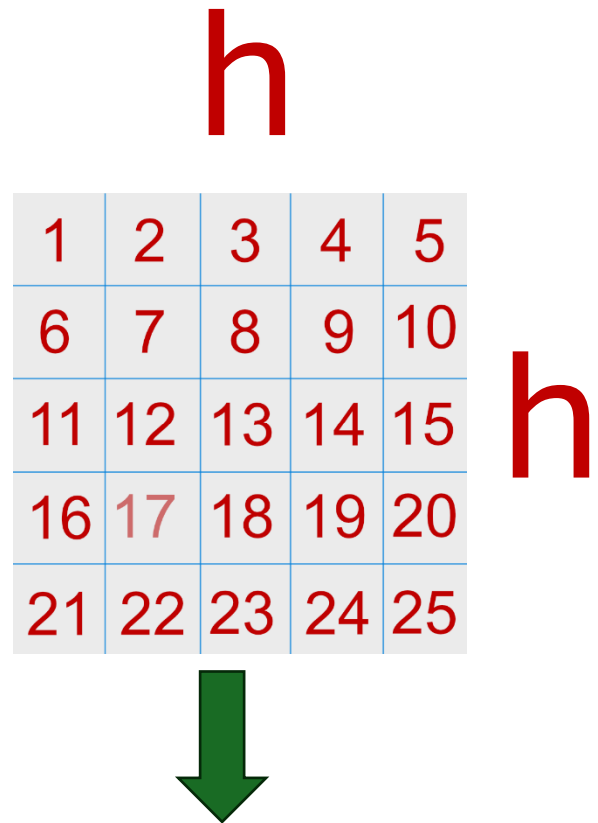
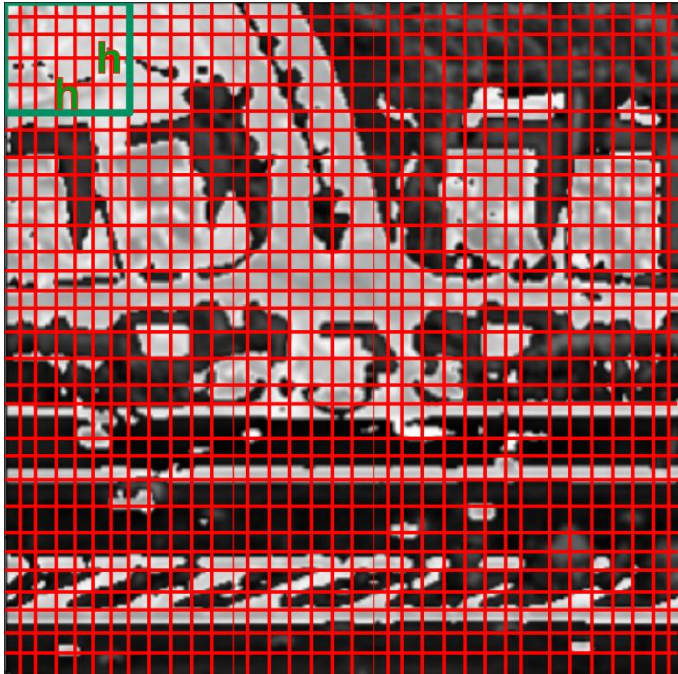


5. Represent each channel image by a set of vectors where vectors are flattened non-overlapping blocks of image and the size of each of these blocks is **h x h**. let's name these vectors “**image vectors**”.



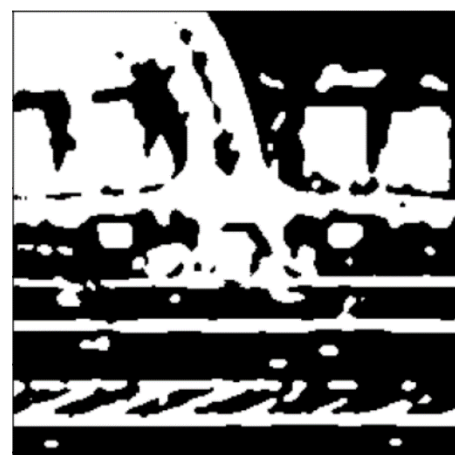
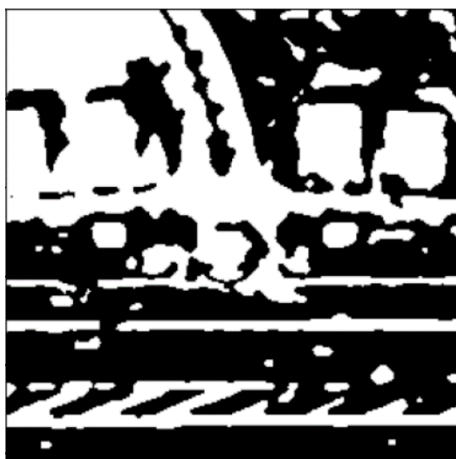
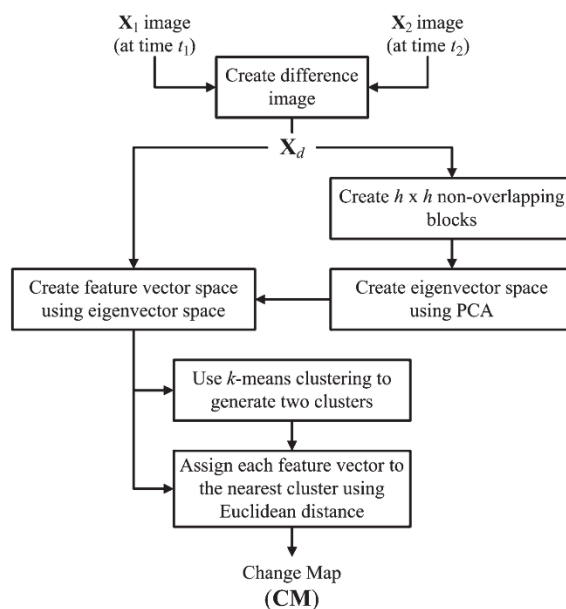
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25

6. Subtract “image vectors” from their mean so that all the vectors have a mean of 0.
7. Apply PCA on the resulting vectors from the preceding step.
8. Segment each channel image into overlapping blocks of size $h \times h$ where each block will represent a pixel then flatten these blocks to find vectors representing pixels. let's name these vectors “pixel vectors”.



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

9. Subtract “pixel vectors” from their mean so that all the vectors have a mean of 0.
10. Project the “pixel vectors” onto PCA eigen vectors.
11. Apply Kmeans with number of clusters = 2. Where the 2 classes will be (pixels changed cluster, pixels didn't change cluster).
12. The author assumed that that cluster with the smallest size shall be the cluster representing the changed pixels.



13. After finding pixels that changed in the 3 channels, we applied **morphological operation OPENING** to remove any small changed pixels with kernel size 5x5. (assuming building size will be big)



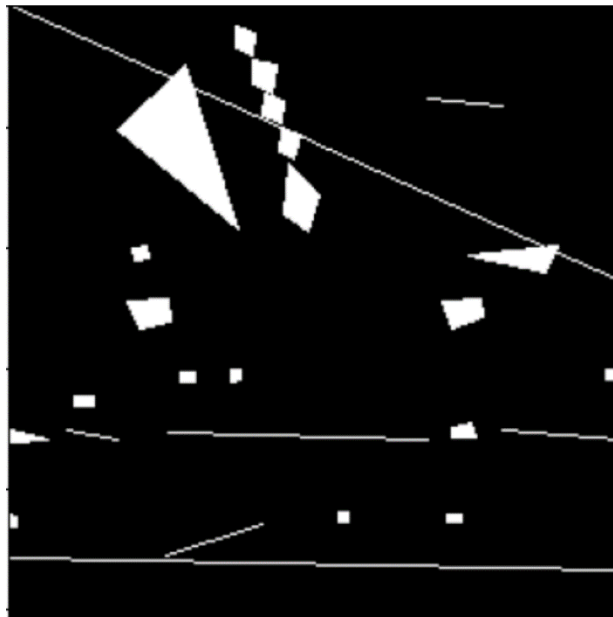
14. Applied **morphological operation CLOSING** afterwards to fill any small gaps inside big building with kernel size 5x5. (assuming building size will be big)



15. We find the **intersection** between the **3 channels** of step 14 assuming that the building pixel has changed all of its 3 channel values.



16. We find all the **polygons** in the image.



17. find the **area** of each polygon.
18. find the **area of the minimum area rectangle** that fits each polygon.
19. **Divide** both areas by each other and if the division factor is less than a threshold, we can consider this shape a rectangle and thus a **building**.



We achieved average Jaccard score on the whole dataset around **40.9%**.

Unet Change Detection (Deep learning Method)

We followed these steps from the paper named “**Unsupervised Change Detection in Satellite Images Using Convolutional Neural Networks**” by “**Kevin Louis de Jong**” and “**Anna Sergeevna Bosman**” in 2019. The basic idea is a small modification on the normal UNET that was introduced in 2015 by concatenation of difference image to the decoder path instead of the feature vector. We split that dataset into **80%** training and **20%** testing.

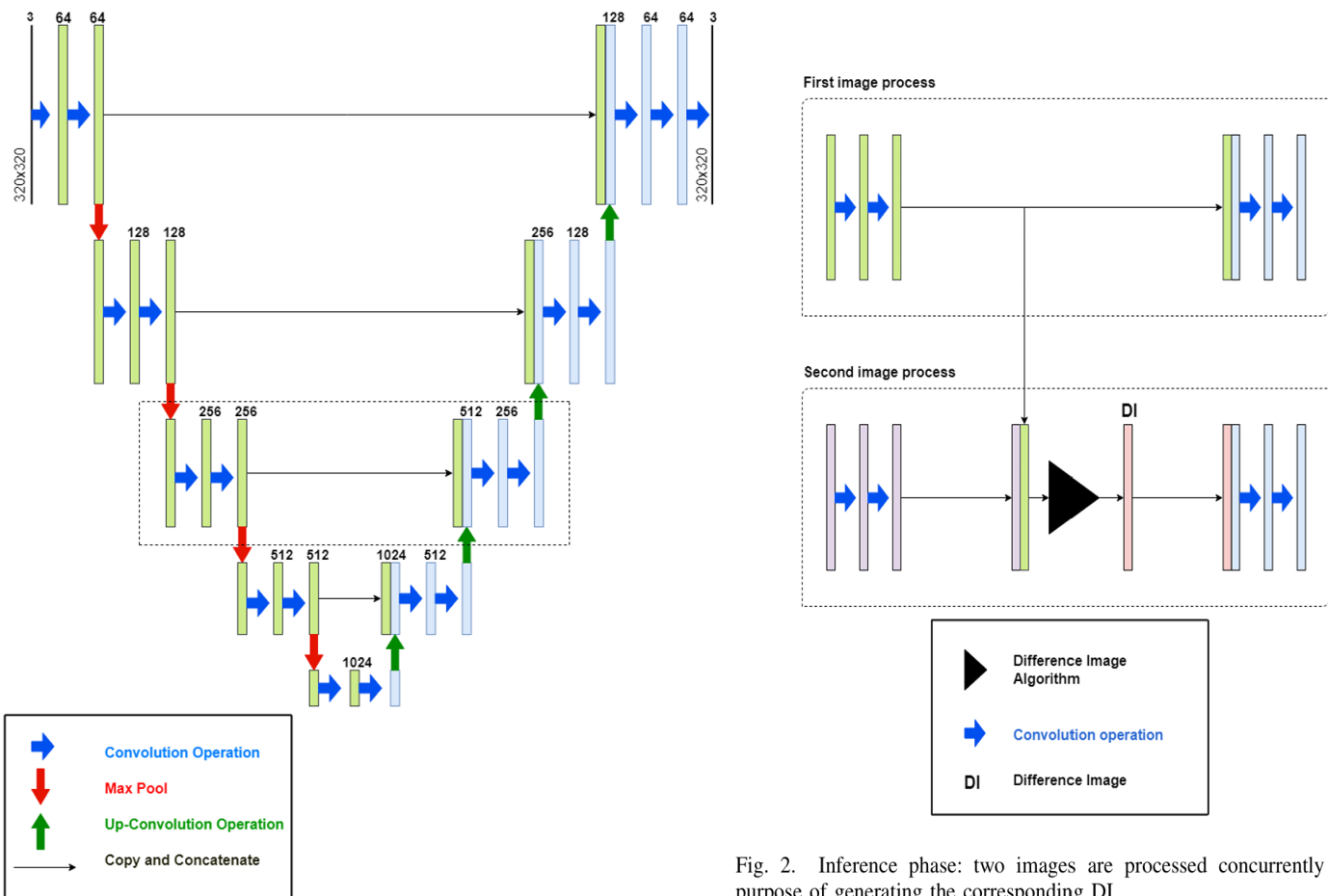


Fig. 1. U-net architecture used in this study.

Fig. 2. Inference phase: two images are processed concurrently for the purpose of generating the corresponding DI.

The input to our UNET is **2 images** and the output is **1 mask** representing pixels where a change in buildings happened. The number of Trainable parameters in our model is **53,378,689** and Average size of model is around **135 MB**. Where all the details about architecture implementations are taken from the paper.

The Full Pipeline of the UNET is as follows:

Layer (type)	Output Shape	Param #
=====		
Conv2d-1	[-1, 64, 256, 256]	1,792
BatchNorm2d-2	[-1, 64, 256, 256]	128
LeakyReLU-3	[-1, 64, 256, 256]	0
Conv2d-4	[-1, 64, 256, 256]	36,928
BatchNorm2d-5	[-1, 64, 256, 256]	128
LeakyReLU-6	[-1, 64, 256, 256]	0
MaxPool2d-7	[-1, 64, 128, 128]	0
Conv2d-8	[-1, 128, 128, 128]	73,856
BatchNorm2d-9	[-1, 128, 128, 128]	256
LeakyReLU-10	[-1, 128, 128, 128]	0
Conv2d-11	[-1, 128, 128, 128]	147,584
BatchNorm2d-12	[-1, 128, 128, 128]	256
LeakyReLU-13	[-1, 128, 128, 128]	0
MaxPool2d-14	[-1, 128, 64, 64]	0
Conv2d-15	[-1, 256, 64, 64]	295,168
BatchNorm2d-16	[-1, 256, 64, 64]	512
LeakyReLU-17	[-1, 256, 64, 64]	0
Conv2d-18	[-1, 256, 64, 64]	590,080
BatchNorm2d-19	[-1, 256, 64, 64]	512
LeakyReLU-20	[-1, 256, 64, 64]	0
MaxPool2d-21	[-1, 256, 32, 32]	0
Conv2d-22	[-1, 512, 32, 32]	1,180,160
BatchNorm2d-23	[-1, 512, 32, 32]	1,024
LeakyReLU-24	[-1, 512, 32, 32]	0
Conv2d-25	[-1, 512, 32, 32]	2,359,808
BatchNorm2d-26	[-1, 512, 32, 32]	1,024
LeakyReLU-27	[-1, 512, 32, 32]	0
MaxPool2d-28	[-1, 512, 16, 16]	0
Conv2d-29	[-1, 1024, 16, 16]	4,719,616
BatchNorm2d-30	[-1, 1024, 16, 16]	2,048
LeakyReLU-31	[-1, 1024, 16, 16]	0
Conv2d-32	[-1, 1024, 16, 16]	9,438,208
BatchNorm2d-33	[-1, 1024, 16, 16]	2,048
LeakyReLU-34	[-1, 1024, 16, 16]	0
Conv2d-35	[-1, 64, 256, 256]	1,792
BatchNorm2d-36	[-1, 64, 256, 256]	128
LeakyReLU-37	[-1, 64, 256, 256]	0
Conv2d-38	[-1, 64, 256, 256]	36,928
BatchNorm2d-39	[-1, 64, 256, 256]	128
LeakyReLU-40	[-1, 64, 256, 256]	0
MaxPool2d-41	[-1, 64, 128, 128]	0
Conv2d-42	[-1, 128, 128, 128]	73,856
BatchNorm2d-43	[-1, 128, 128, 128]	256
LeakyReLU-44	[-1, 128, 128, 128]	0
Conv2d-45	[-1, 128, 128, 128]	147,584
BatchNorm2d-46	[-1, 128, 128, 128]	256
LeakyReLU-47	[-1, 128, 128, 128]	0
MaxPool2d-48	[-1, 128, 64, 64]	0
Conv2d-49	[-1, 256, 64, 64]	295,168
BatchNorm2d-50	[-1, 256, 64, 64]	512
LeakyReLU-51	[-1, 256, 64, 64]	0
Conv2d-52	[-1, 256, 64, 64]	590,080
BatchNorm2d-53	[-1, 256, 64, 64]	512

LeakyReLU-54	[-1, 256, 64, 64]	0
MaxPool2d-55	[-1, 256, 32, 32]	0
Conv2d-56	[-1, 512, 32, 32]	1,180,160
BatchNorm2d-57	[-1, 512, 32, 32]	1,024
LeakyReLU-58	[-1, 512, 32, 32]	0
Conv2d-59	[-1, 512, 32, 32]	2,359,808
BatchNorm2d-60	[-1, 512, 32, 32]	1,024
LeakyReLU-61	[-1, 512, 32, 32]	0
MaxPool2d-62	[-1, 512, 16, 16]	0
Conv2d-63	[-1, 1024, 16, 16]	4,719,616
BatchNorm2d-64	[-1, 1024, 16, 16]	2,048
LeakyReLU-65	[-1, 1024, 16, 16]	0
Conv2d-66	[-1, 1024, 16, 16]	9,438,208
BatchNorm2d-67	[-1, 1024, 16, 16]	2,048
LeakyReLU-68	[-1, 1024, 16, 16]	0
ConvTranspose2d-69	[-1, 512, 32, 32]	4,719,104
BatchNorm2d-70	[-1, 512, 32, 32]	1,024
Conv2d-71	[-1, 512, 32, 32]	4,719,104
BatchNorm2d-72	[-1, 512, 32, 32]	1,024
LeakyReLU-73	[-1, 512, 32, 32]	0
Conv2d-74	[-1, 512, 32, 32]	2,359,808
BatchNorm2d-75	[-1, 512, 32, 32]	1,024
LeakyReLU-76	[-1, 512, 32, 32]	0
ConvTranspose2d-77	[-1, 256, 64, 64]	1,179,904
BatchNorm2d-78	[-1, 256, 64, 64]	512
Conv2d-79	[-1, 256, 64, 64]	1,179,904
BatchNorm2d-80	[-1, 256, 64, 64]	512
LeakyReLU-81	[-1, 256, 64, 64]	0
Conv2d-82	[-1, 256, 64, 64]	590,080
BatchNorm2d-83	[-1, 256, 64, 64]	512
LeakyReLU-84	[-1, 256, 64, 64]	0
ConvTranspose2d-85	[-1, 128, 128, 128]	295,040
BatchNorm2d-86	[-1, 128, 128, 128]	256
Conv2d-87	[-1, 128, 128, 128]	295,040
BatchNorm2d-88	[-1, 128, 128, 128]	256
LeakyReLU-89	[-1, 128, 128, 128]	0
Conv2d-90	[-1, 128, 128, 128]	147,584
BatchNorm2d-91	[-1, 128, 128, 128]	256
LeakyReLU-92	[-1, 128, 128, 128]	0
ConvTranspose2d-93	[-1, 64, 256, 256]	73,792
BatchNorm2d-94	[-1, 64, 256, 256]	128
Conv2d-95	[-1, 64, 256, 256]	73,792
BatchNorm2d-96	[-1, 64, 256, 256]	128
LeakyReLU-97	[-1, 64, 256, 256]	0
Conv2d-98	[-1, 64, 256, 256]	36,928
BatchNorm2d-99	[-1, 64, 256, 256]	128
LeakyReLU-100	[-1, 64, 256, 256]	0
Conv2d-101	[-1, 1, 256, 256]	577

```

=====
Total params: 53,378,689
Trainable params: 53,378,689
Non-trainable params: 0

```

```

-----
Input size (MB): 0.00
Forward/backward pass size (MB): 1254.50
Params size (MB): 203.62
Estimated Total Size (MB): 1458.12
-----

```


We used a **batch size of 4** where the size of training data is **3894 images**. Each epoch took around **25 minutes** to complete. The best loss we could get is **1.3%**. The best accuracy we could get on the testing data is **79.8%** (at epoch No. **110** and loss on training data was **1.4%**) using average Jaccard index where the size of test data is **974 images**. The Epoch number at which we stopped at is **epoch number 129**. Where the training of such UNET took around **54.17 hours**. We only begin calculating accuracies after a point where the loss was acceptable, but we don't use test data in any type of optimization/training the model. The optimizer used is **Adam optimizer** and the loss function used is **Binary Cross Entropy** with **sigmoid activation** function. After some time (after epoch 110), we noticed that the loss is stuck at around **1.3%** for around 20 epochs, that's when we assumed that we reached the global minimum.



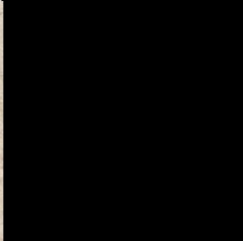
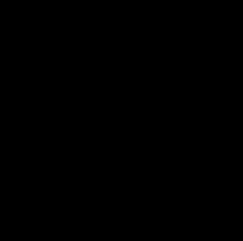
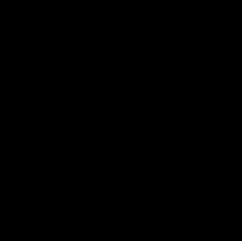
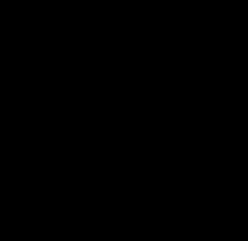
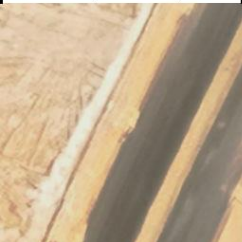

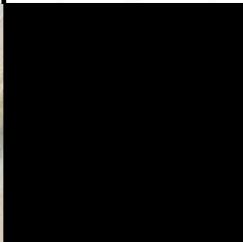
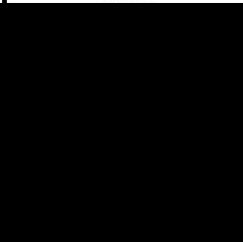

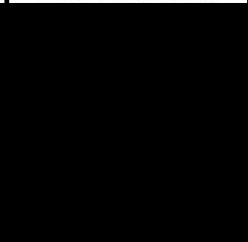


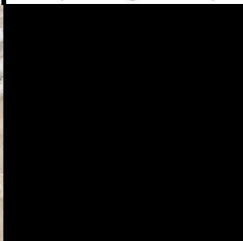
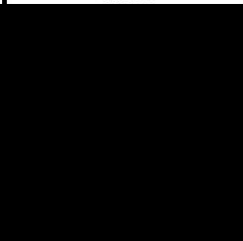
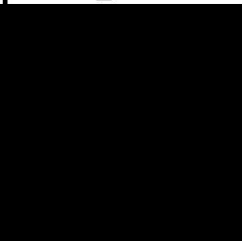
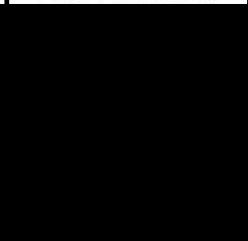

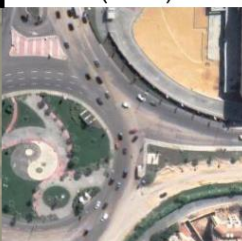


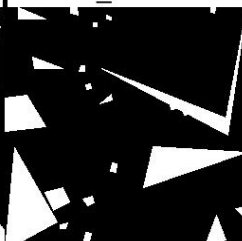









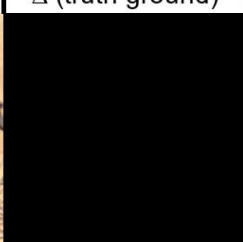
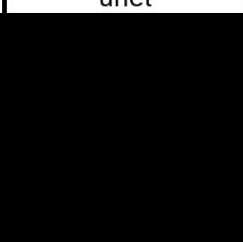

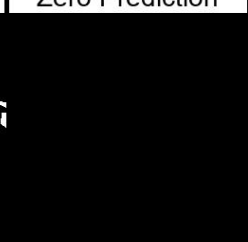
The Average Jaccard index on the whole dataset (including training and Testing) is **90.3%**.




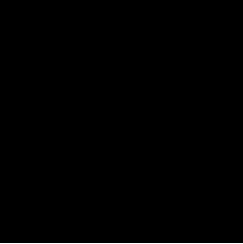

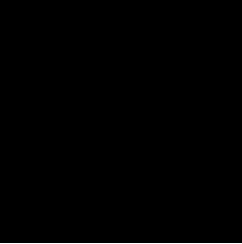




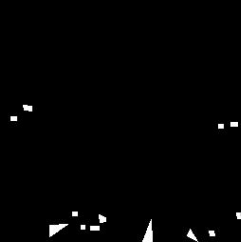
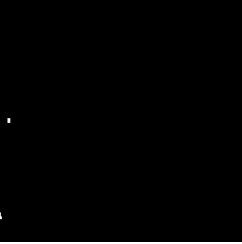




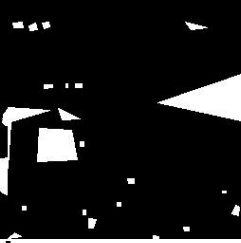




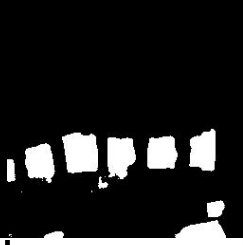
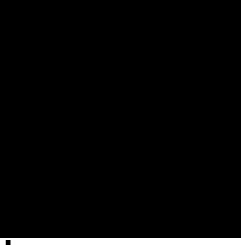
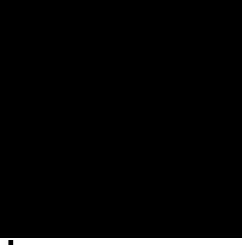







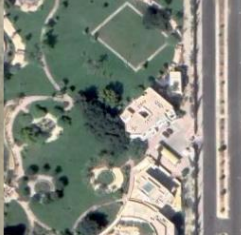



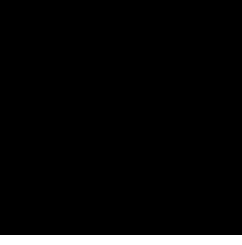
```
Epoch Number: 0, loss: 0.17471466610280822
Epoch Number: 1, loss: 0.14199091476202777
Epoch Number: 2, loss: 0.1307049605438918
Epoch Number: 3, loss: 0.12042065357475769
Epoch Number: 4, loss: 0.11350404719623222
Epoch Number: 5, loss: 0.10678567868816137
Epoch Number: 6, loss: 0.10418169898195173
Epoch Number: 7, loss: 0.10154102344063036
Epoch Number: 8, loss: 0.09619461639520847
Epoch Number: 9, loss: 0.09681554608074738
Epoch Number: 10, loss: 0.09043137947610842
Epoch Number: 11, loss: 0.09122554887611514
Epoch Number: 12, loss: 0.08778136230073869
Epoch Number: 13, loss: 0.08531742462100318
Epoch Number: 14, loss: 0.08513728124602961
Epoch Number: 15, loss: 0.08396289597396762
Epoch Number: 16, loss: 0.08204886507624999
Epoch Number: 17, loss: 0.08123266641002817
Epoch Number: 18, loss: 0.07788623177774776
Epoch Number: 19, loss: 0.07695879778371026
Epoch Number: 20, loss: 0.0773150671296753
Epoch Number: 21, loss: 0.07461446291725271
Epoch Number: 22, loss: 0.07397567353265073
Epoch Number: 23, loss: 0.07110195147118364
Epoch Number: 24, loss: 0.07282069058223174
Epoch Number: 25, loss: 0.07203668586381425
Epoch Number: 26, loss: 0.06936057655755669
Epoch Number: 27, loss: 0.06749881396326833
Epoch Number: 28, loss: 0.06644962163367428
Epoch Number: 29, loss: 0.06421788896651914
```




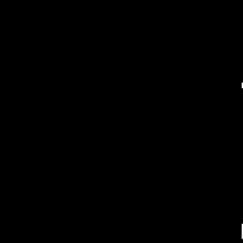

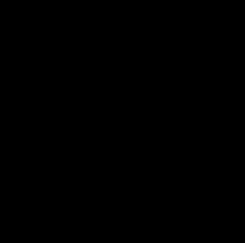




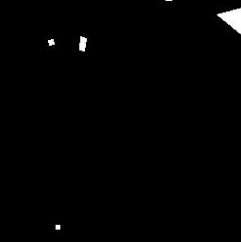
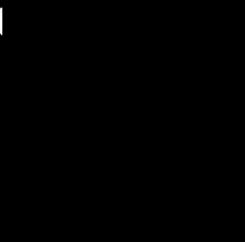


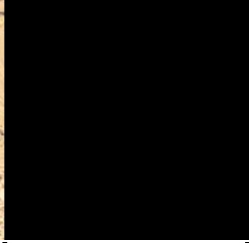
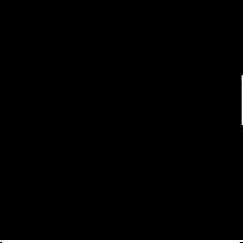

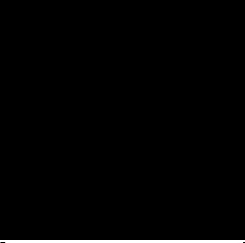
Epoch Number: 30, loss: 0.06119485259121762
Epoch Number: 31, loss: 0.0635630344072333
Epoch Number: 32, loss: 0.06183925170000666
Epoch Number: 33, loss: 0.05941628801672218
Epoch Number: 34, loss: 0.0579487271534088
Epoch Number: 35, loss: 0.05860779281007126
Epoch Number: 36, loss: 0.055897489813013145
Epoch Number: 37, loss: 0.0532455685517888
Epoch Number: 38, loss: 0.05475971405648101
Epoch Number: 39, loss: 0.05167587826432935
Epoch Number: 40, loss: 0.05259273568438426
Epoch Number: 41, loss: 0.04742643369125943
Epoch Number: 42, loss: 0.0495780134108216
Epoch Number: 43, loss: 0.04696909153545932
Epoch Number: 44, loss: 0.04727337025368634
Epoch Number: 45, loss: 0.04640071866825635
Epoch Number: 46, loss: 0.044624641416758565
Epoch Number: 47, loss: 0.04636625606106976
Epoch Number: 48, loss: 0.0395877977635395
Epoch Number: 49, loss: 0.04267916032548946
Epoch Number: 50, loss: 0.04036075148665054
Epoch Number: 51, loss: 0.038599751777200694
Epoch Number: 52, loss: 0.039527586821783096
Epoch Number: 53, loss: 0.036022510089507055
Epoch Number: 54, loss: 0.03860302367203612
Epoch Number: 55, loss: 0.03348042369882863
Epoch Number: 56, loss: 0.03476635820545758
Epoch Number: 57, loss: 0.033827092255180616
Epoch Number: 58, loss: 0.03547216569396625
Epoch Number: 59, loss: 0.032490302818358685
Epoch Number: 60, loss: 0.03181324988419268
Epoch Number: 61, loss: 0.03036412198159199
Epoch Number: 62, loss: 0.03203518028242681
Epoch Number: 63, loss: 0.03025976282579508
Epoch Number: 64, loss: 0.029523484432179545
Epoch Number: 65, loss: 0.031800010666554905
Epoch Number: 66, loss: 0.031178366149132414
Epoch Number: 67, loss: 0.02727725324516098
Epoch Number: 68, loss: 0.02928224215034928
Epoch Number: 69, loss: 0.025545205192055363
Epoch Number: 70, loss: 0.026400011608847983
Epoch Number: 71, loss: 0.02656357577568717
Epoch Number: 72, loss: 0.02846442469823113
Epoch Number: 73, loss: 0.024731862069018416
Epoch Number: 74, loss: 0.02802028005623866
Epoch Number: 75, loss: 0.02254212385992548
Epoch Number: 76, loss: 0.02273202372093249
Epoch Number: 77, loss: 0.025751342572440102
Epoch Number: 78, loss: 0.022477793754459036
Epoch Number: 79, loss: 0.022053216487945927, accuracy: 0.7869910427949316
Epoch Number: 80, loss: 0.024805048190197947, accuracy: 0.6907955541043014
Epoch Number: 81, loss: 0.020462346137466784, accuracy: 0.7436567584159638
Epoch Number: 82, loss: 0.026825547118296992, accuracy: 0.6537994087079059
Epoch Number: 83, loss: 0.022409917038482188, accuracy: 0.6840106076513264
Epoch Number: 84, loss: 0.019118333191589232, accuracy: 0.7672067348193526
Epoch Number: 85, loss: 0.020251649564919576, accuracy: 0.7617529957171347
Epoch Number: 86, loss: 0.019782174667803375, accuracy: 0.7219969644730446
Epoch Number: 87, loss: 0.02722662753700626, accuracy: 0.7567429156590346

Epoch Number: 88, loss: 0.019142074202814583, accuracy: 0.774129668128886
Epoch Number: 89, loss: 0.019226005152195285, accuracy: 0.7046181426563597
Epoch Number: 90, loss: 0.02144231983046689, accuracy: 0.7672622234385005
Epoch Number: 91, loss: 0.01812967308697131, accuracy: 0.7785455402332744
Epoch Number: 92, loss: 0.018764672111809398, accuracy: 0.7466231905208605
Epoch Number: 93, loss: 0.020013407093323714, accuracy: 0.7651935727007493
Epoch Number: 94, loss: 0.021485326976842956, accuracy: 0.7517801207040218
Epoch Number: 95, loss: 0.01763455447283027, accuracy: 0.7833356918543678
Epoch Number: 96, loss: 0.016019435809745724, accuracy: 0.7920859274596624
Epoch Number: 97, loss: 0.017902836229308702, accuracy: 0.7559186967999327
Epoch Number: 98, loss: 0.016589173903125462, accuracy: 0.7411334146869094
Epoch Number: 99, loss: 0.018863177647579457, accuracy: 0.7761943500161986
Epoch Number: 100, loss: 0.015854322590047768, accuracy: 0.7271847571124899
Epoch Number: 101, loss: 0.01855422941511372, accuracy: 0.769073589107627
Epoch Number: 102, loss: 0.016244728196780227, accuracy: 0.7787071870368447
Epoch Number: 103, loss: 0.01925076852685752, accuracy: 0.7370219883984259
Epoch Number: 104, loss: 0.017273170100691994, accuracy: 0.785366661200586
Epoch Number: 105, loss: 0.016334658692242394, accuracy: 0.747247270718485
Epoch Number: 106, loss: 0.01476667620485573, accuracy: 0.7683090719323502
Epoch Number: 107, loss: 0.015077765135101333, accuracy: 0.7266526493216285
Epoch Number: 108, loss: 0.014437564531791186, accuracy: 0.759088149836224
Epoch Number: 109, loss: 0.01446279214243567, accuracy: 0.7983807751551625
Epoch Number: 110, loss: 0.01331067952809136, accuracy: 0.7662518140769952
Epoch Number: 111, loss: 0.013293619061328988, accuracy: 0.7868450819217254
Epoch Number: 112, loss: 0.013248386818644903, accuracy: 0.7700389718219534
Epoch Number: 113, loss: 0.013306120334469037, accuracy: 0.739332749647138
Epoch Number: 114, loss: 0.013282486230157538, accuracy: 0.7773041188669914
Epoch Number: 115, loss: 0.013285523185285199, accuracy: 0.7383100974837852
Epoch Number: 116, loss: 0.013308286097464869, accuracy: 0.778420286732594
Epoch Number: 117, loss: 0.013252660052372588, accuracy: 0.7646779380758932
Epoch Number: 118, loss: 0.013279916829056814, accuracy: 0.7732039066478488
Epoch Number: 119, loss: 0.01324253241493119, accuracy: 0.7592502010050094
Epoch Number: 120, loss: 0.01329216821672162, accuracy: 0.7646928063582472
Epoch Number: 121, loss: 0.013258468697189621, accuracy: 0.7464283438402518
Epoch Number: 122, loss: 0.013256829387661267, accuracy: 0.7712372280553632
Epoch Number: 123, loss: 0.013229919829855368, accuracy: 0.7726134172994895
Epoch Number: 124, loss: 0.013220536708694197, accuracy: 0.7944397577941694
Epoch Number: 125, loss: 0.0133429440978216, accuracy: 0.7817902949890775
Epoch Number: 126, loss: 0.013311653138790484, accuracy: 0.7441484583440446
Epoch Number: 127, loss: 0.013336575514767902, accuracy: 0.7790645996373085
Epoch Number: 128, loss: 0.0132588277182661, accuracy: 0.7778985279360849
Epoch Number: 129, loss: 0.013352043190783055, accuracy: 0.750840650250314

Comparison Examples

A (before)	B (After)	Δ (truth ground)	unet	PCA_Kmeans	Zero Prediction
					
A (before)	B (After)	Δ (truth ground)	unet	PCA_Kmeans	Zero Prediction
					
A (before)	B (After)	Δ (truth ground)	unet	PCA_Kmeans	Zero Prediction
					
A (before)	B (After)	Δ (truth ground)	unet	PCA_Kmeans	Zero Prediction
					
A (before)	B (After)	Δ (truth ground)	unet	PCA_Kmeans	Zero Prediction
					
A (before)	B (After)	Δ (truth ground)	unet	PCA_Kmeans	Zero Prediction
					

A (before)	B (After)	Δ (truth ground)	unet	PCA_Kmeans	Zero Prediction
					
A (before)	B (After)	Δ (truth ground)	unet	PCA_Kmeans	Zero Prediction
					
A (before)	B (After)	Δ (truth ground)	unet	PCA_Kmeans	Zero Prediction
					
A (before)	B (After)	Δ (truth ground)	unet	PCA_Kmeans	Zero Prediction
					
A (before)	B (After)	Δ (truth ground)	unet	PCA_Kmeans	Zero Prediction
					
A (before)	B (After)	Δ (truth ground)	unet	PCA_Kmeans	Zero Prediction
					

A (before)	B (After)	Δ (truth ground)	unet	PCA_Kmeans	Zero Prediction
					
A (before)	B (After)	Δ (truth ground)	unet	PCA_Kmeans	Zero Prediction
					
A (before)	B (After)	Δ (truth ground)	unet	PCA_Kmeans	Zero Prediction
					
A (before)	B (After)	Δ (truth ground)	unet	PCA_Kmeans	Zero Prediction
