ST-Project Report

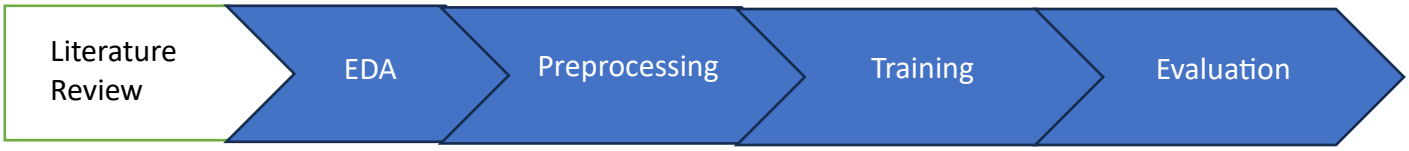
Submitted by: Team 8

Name	Sec	BN
Ahmed Abdelatty	1	10
Malek Hossam	2	4
Mohamed Taher	2	8
Ahmed Mohamed Omer	1	11

Submitted to: Eng. Noran Hany

1. Project Pipeline

The project pipeline is as follows: -



2. Literature Review

Cloud masking is about finding and removing cloudy pixels from satellite images. Essential for weather, climate, land, and solar applications.

Old-school methods = simple thresholding on spectral bands (like reflectance, temperature). Popular algorithms include FMask[1] for Landsat images.

Good for clear skies. Fail on clouds over snow, thin cirrus, or bright surfaces.

Machine Learning came next:

- Random Forests.
- SVMs.
- Shallow Neural Networks.

ML learns features instead of manually setting thresholds. Better, but problems stay: small clouds are missed, heavy dependence on labeled data, sensor-specific tuning needed.

Deep Learning changed the game:

- **U-Net[2]** — classic for segmentation.
- **CloudFCN[3]** — optimized for cloud masking.
- **Cloud-Net[4]** — specialized architecture for clouds.

DL models learn both spectral and spatial features end-to-end. No manual feature engineering. They beat everything before — especially on hard cases like thin clouds.

Transfer learning (using pre-trained models) and sensor-independent networks like **SEnSeI[5]** help handle data scarcity and different satellites.

Big problems that still remain:

- Annotation noise — datasets disagree on what counts as "cloud."
- Evaluation inconsistency — metrics like IoU, F1, OA change with datasets.

- Labeling cost — training good models needs lots of pixel-perfect masks.

New datasets like **CloudSEN12**[6] try to fix standardization issues.

Future needs:

- Smaller, greener models (less compute, less carbon).
- Transfer learning, domain adaptation, weak supervision to avoid heavy labeling.
- More generalization across sensors, times, regions.

References

1. Zhe Zhu, Shixiong Wang, Curtis E. Woodcock, Improvement and expansion of the Fmask algorithm: cloud, cloud shadow, and snow detection for Landsats 4–7, 8, and Sentinel 2 images, Remote Sensing of Environment, Volume 159, 2015, Pages 269-277, ISSN 0034-4257, https://doi.org/10.1016/j.rse.2014.12.014.
2. Ronneberger, O., Fischer, P., Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. In: Navab, N., Hornegger, J., Wells, W., Frangi, A. (eds) Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015. MICCAI 2015. Lecture Notes in Computer Science(), vol 9351. Springer, Cham. https://doi.org/10.1007/978-3-319-24574-4_28
3. Francis, Alistair & Sidiropoulos, Panagiotis & Muller, J.-P. (2019). CloudFCN: Accurate and Robust Cloud Detection for Satellite Imagery with Deep Learning. Remote Sensing. 11. 2312. 10.3390/rs11192312.
4. Mohajerani, Sorour, and Parvaneh Saeedi. "Cloud-Net: An end-to-end cloud detection algorithm for Landsat 8 imagery." *IGARSS 2019-2019 IEEE international geoscience and remote sensing symposium*. IEEE, 2019.
5. A. Francis, J. Mrziglod, P. Sidiropoulos and J. -P. Muller, "SEnSeI: A Deep Learning Module for Creating Sensor Independent Cloud Masks," in IEEE Transactions on Geoscience and Remote Sensing, vol. 60, pp. 1-21, 2022, Art no. 5406121, doi: 10.1109/TGRS.2021.3128280. keywords: {Satellites;Data models;Deep learning;Remote sensing;Cloud computing;Training;Earth;Cloud masking;deep learning;Earth observation;interoperability;multispectral;sensor independence}
6. Aybar, C., Ysuhuaylas, L., Loja, J. *et al.* CloudSEN12, a global dataset for semantic understanding of cloud and cloud shadow in Sentinel-2. *Sci Data* **9**, 782 (2022). https://doi.org/10.1038/s41597-022-01878-2

3. Exploratory Data Analysis

It was found that the labelling of the data was not perfect. The masks overestimated the boundaries of the clouds and several images were mislabeled. Through manual inspection, we found 3174 images to be mislabeled out of a total of 10573 images. This posed a great challenge, but we managed to filter out the noisy masks and only retain the masks that were satisfactory.

4. Preprocessing

We have three different preprocessing schemes depending on the used model.

a. Threshold-based model

Here, our preprocessing consists of normalizing each band of the RGB and IR images as well as constructing the *diff* image which is the difference between the IR and RGB images.

b. Machine Learning Model

Here, we extract the features from our images which are the 4 bands, the images' textures in each band (local variance in shape of size 3x3), and the NDVI (Normalized Difference Vegetation Index) for a total of 9 features. We also resize the images to 128x128 for training to be feasible on our large dataset.

c. Deep Learning Model

Here, a number of augmentations are performed in order to increase the variability of our dataset and hence improve generalizability. These include: horizontal flips, vertical flips, random rotations, elastic transforms, distortions, contrast and blur. Additionally, we resize to 192x192 to decrease training time and increase the batch size which helps batch norm calculate more accurate statistics. However, with the UNet we do not resize the images for accuracy.

5. Model Training

a. Thresholding

After constructing the *diff* image mentioned above, we apply a threshold on this image and on the IR image: clouds have mean IR values that are **higher** than a certain value and diff values that are **lower** than a certain value. We found the best two thresholds to be 0.4 and 0.3.

b. Random Forest

We train the random forest by first extracting 9 features for each pixel as mentioned above. We then use scikit-learn's RandomForestClassifier with num_classifiers=8 to train on each pixel in our dataset.

We use a train/val split of 60/40 in order to have a better estimate of the performance on unseen data, random forests have a tendency to overfit, and to reduce training time. We also downsample images to 128x128 for the same reason.

c. UNet

We trained the UNet for 45 epochs where each epoch took approximately 14 minutes training on a batch size of 8 with Adam optimizer and an initial learning rate of 1e-3 which is continuously reduced by a factor of 0.5 whenever improvement does not happen in 3 epochs. We used a loss function that was the mean of BCE loss and Dice loss. The hyperparameter values were arrived at via experimentation and reviewing the literature. BCE (Binary Cross-Entropy) loss is the go-to loss for binary semantic segmentation, but it does not inherently deal with imbalanced classes, and it is not the actual goal but rather a proxy, which is why we used dice loss as well.

d. CloudNet

CloudNet is a neural network architecture based on the UNet, with modified skip connections that account for all upper layer activations and not just the current layer. It also has more complex convolution blocks. The model was developed specifically for cloud masking, hence the name.

We trained it for 100 epochs where each epoch took about 5 minutes (because of the reduced image size) on a batch size of 16 with Adam optimizer and an initial learning rate of 5e-4 which was continuously reduced by a factor of 0.7 whenever improvement did not occur for 5 epochs. We used the same mixed loss as above but slightly skewed towards BCE loss, as we thought it was more important to reduce BCE loss in order to improve classification accuracy.

6. Model Selection

Given that the CloudNet model achieved the highest Dice Coefficient and Jaccard Index scores on the validation set (0.9231 and 0.8419 respectively), we decided to select this model for inference on the test set.

7. Model Evaluation

Random Forest was able to achieve a Jaccard Index (mIoU) of 0.7887.

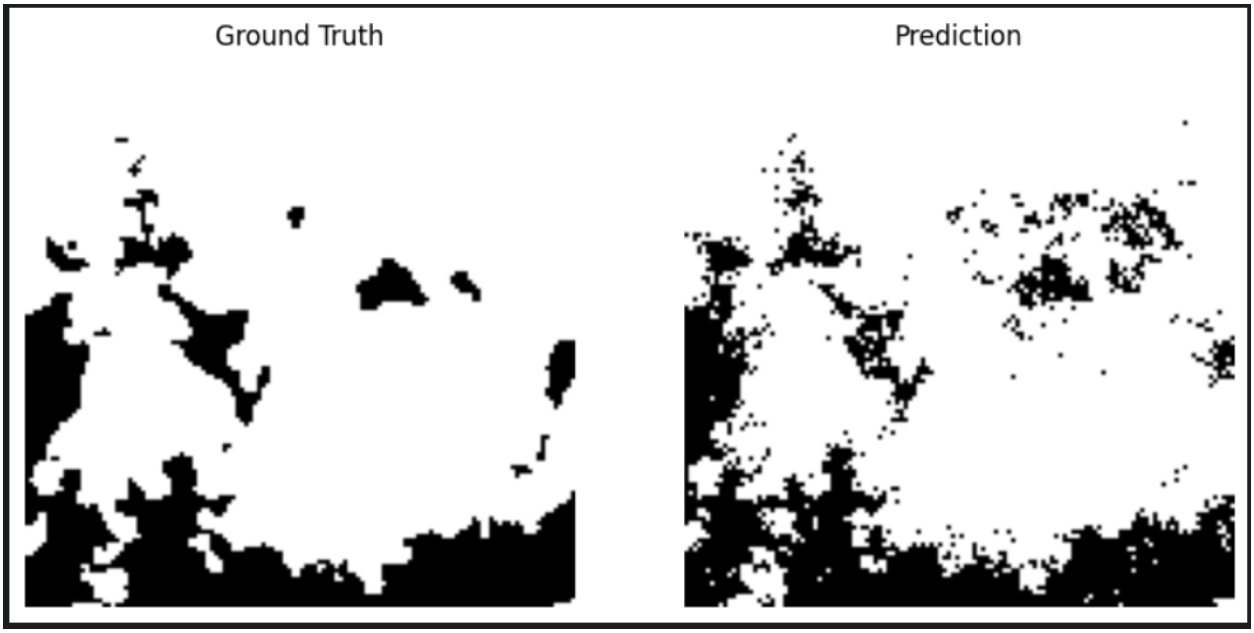


Figure 1: Sample prediction of random forest

CloudNet achieved a Jaccard Index (mIoU) of 0.8419 and a Dice Coefficient of 0.9231, while the BCE loss was 0.1462.

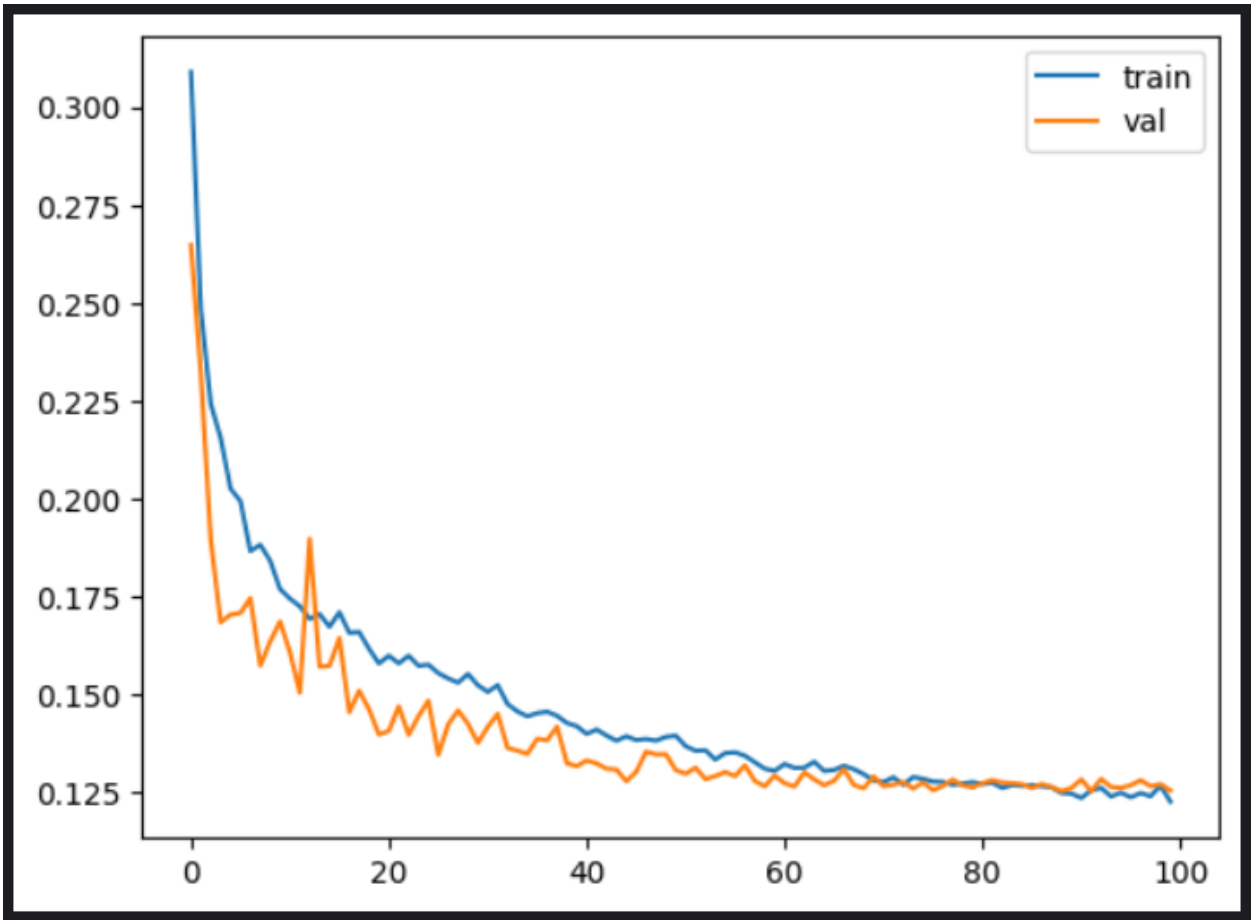


Figure 2: Train/Val loss curve of CloudNet

UNet achieved a Jaccard Index of 0.8199 and a Dice Coefficient of 0.9126, while the BCE loss was 0.1532.

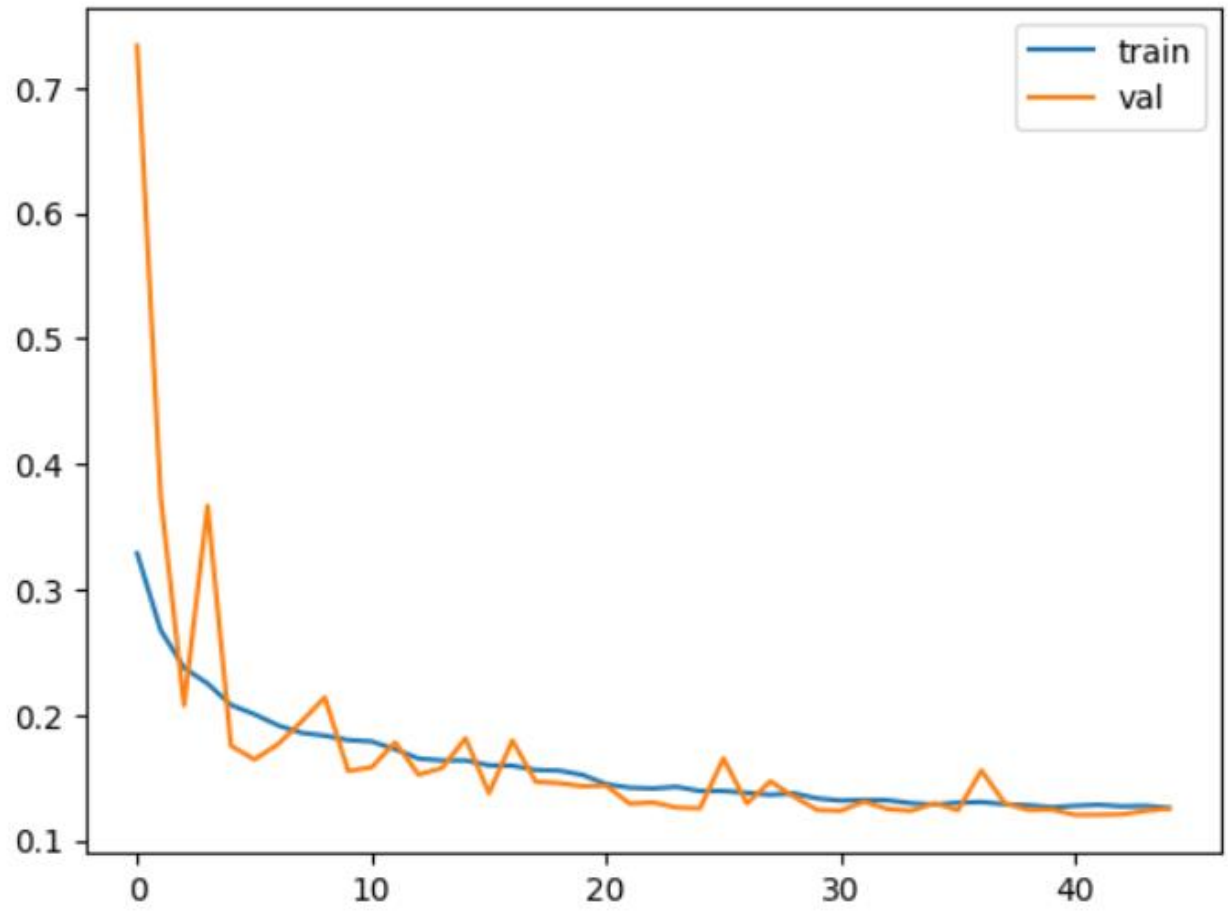


Figure 3: Train/Val loss curve for UNet

8. Future Work

1. Use ensemble methods for stronger, more stable predictions
2. Postprocessing using CRF (Conditional Random Fields) or morphological operations
3. Multi-task learning: predicting clouds and their types and not just binary cloud mask
4. Using tiny models for deployment on edge devices (like ENet)

9. Workload Distribution

Name	Workload
Ahmed Abdelatty	Training
Malek Hossam	Evaluation
Mohamed Taher	Preprocessing
Ahmed Mohamed Omer	EDA

10. Appendix

a. Profiler

The profiling script was renamed to `profile_script.csv` to avoid an ambiguity between it and a builtin python library.