Project Report

Monitoring war destruction from space using machine learning

Team: Arogya Koirala, Raghav Mittal

Github Repository: https://github.com/arogyakoirala/destruction

Background and motivation

This project aims to identify severely damaged buildings in the civil war affected region of Aleppo, Syria by the application of machine learning. War-related building destruction is a specific form violence, and in its most extreme form called “urbicide”, where it is strategically used to displace populations from a settlement area. Monitoring an active war zone is difficult: as there is a high chance of an information gap due to data being often incomplete. This is due to several factors. First, it is very difficult to go on the ground and collect real-time information about security concerns. Second, even in situations where information does exist, it is often incomplete and highly contested. Organizations that do report damage often cannot cover the full extent of the region. Additionally, there is no guarantee that reposting done by local or state owned media is not prone to bias and misinformation. This lack of information has consequences. It reduces the scope of humanitarian relief and reconstruction efforts, increases the likelihood of bias in media reporting, and limits potential academic advances in conflict studies.

Prior Work

Our goal is to use machine learning and computer vision techniques to provide a more ”objective” and ”complete” picture of war destruction. The final outcome will be heat-maps overlaid on top of satellite imagery of Aleppo for a given date, that show the probability of destruction in different parts of the city. See image below.

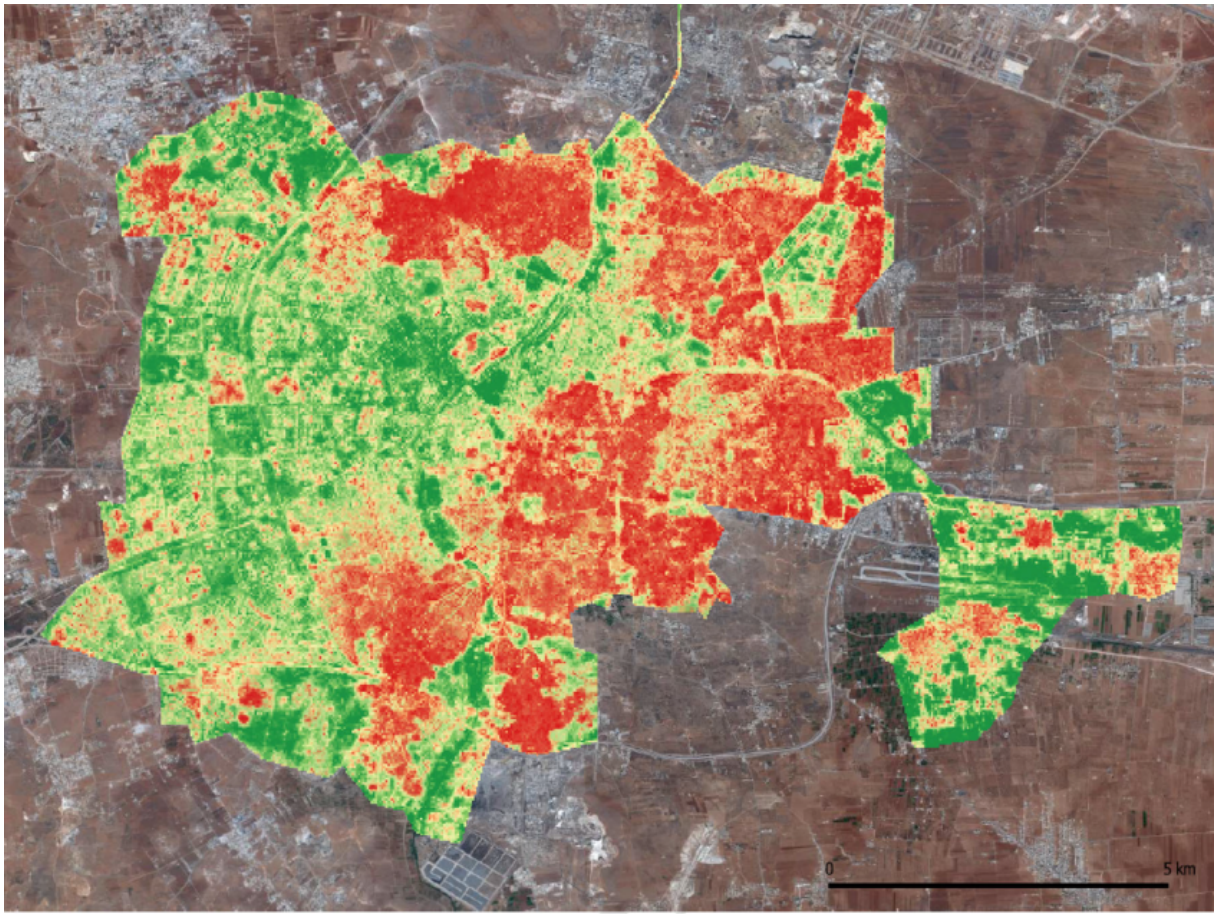

Figure: Dense predictions for the settlement region of Aleppo at the 128x128 pixel resolution. Red indicates high probability of damage, yellow indicates low.

Since Summer 2022, one of our team members has been working with a team of researchers (Hannes Mueller, Andre Groeger, Clement Gorin) in Barcelona School of Economics to explore the issues and challenges of using Satellite Imagery data from Google Earth to detect war destruction on buildings in 8 cities of Syria.

The team's current approach to this problem involves the training of neural network architectures (Convolutional Neural Networks and Siamese Neural Networks), which learn from the RGB channels of over 1.3 million image patches in these cities. Naturally, since the team has been working with deep networks, model training is very resource intensive, and often takes days. Additionally, since this is a "black-boxed" architecture they have no way of informing understanding of what the model is actually learning. The goal of this project is to understand if feature descriptors such as HOGs and Laplacians combined with dimensionality reduction can yield performance improvements to the model. We also want to carry out a benchmarking exercise where we compare

performance of these neural network models against a standard classifier to get a measure of how much gain these networks actually provide.

Update to existing proposal

Because of computational complexity, our original attempt at running an SVC classifier on the feature descriptors have not been successful. The data size seems to be too large for the model to converge. As a result, we do not have results on the SVC just yet. Instead, we have now focused our efforts on understanding the lift gained on Siamese networks when using feature descriptors. For this project, we will therefore, compare the performance of the following models:

1. Siamese Network with raw images
2. Siamese Network with HOG Descriptors of the Image
3. Siamese Network with Laplacians of the Image.

High level approach

An outline of the high level approach taken to identify war related destruction using Siamese Network is given below:

1. Make and label patches from satellite imagery of region of analysis (see below) cities in Syria from different points in time
2. Assign first two images (taken from a date before the war) as pre images:
    a. Divide pre-images imagery into patches of 64x64 128x128 pixels (roughly 64m x 64m)
3. Assign the rest as post images:
    a. Divide pre-images imagery into patches of 64x64 128x128 pixels (roughly 64m x 64m)
    b. Attribute a positive (1) to a patch if the 128 x 128 pixel window overlaps with a positive groundtruth tag
    c. If not it's a negative (0)
4. Create a coupled array that contains post and pre patches.
5. This gives us 6 layers (2*RGB) to feed into a SNN
6. Randomly assign patches to train and test train, test, and validation set.
7. Generate separate test, train, and validate datasets for extracted features (HOGs, Laplacians)
8. Sample positives up for training to reach 1:1
9. Train using the train and validation sets, and the report performance on the test set.

Data

This section provides details on the nature of the data that we are working with.

Satellite Imagery

War destruction images of the city of Aleppo, Syria, are obtained from Google Earth Satellite Imagery by selecting the specific dates of interest and defining the longitude and latitudes bounding box. All tiles for a city might not be available for a given date, and so Google Earth Engine selects the closest available tiles to the selected date. The images are at a submeter (~50cm resolution), and can be found here.

Images are cropped to specific raster dimensions, with the height and width multiples of 256. Doing this will allow the team to experiment with different patch sizes in future revisions.

Damage Annotations

We use georeferenced building damage labels from the United Nations Operational Satellite Applications Program (UNOSAT), available via the humanitarian data exchange website, to generate labels for our analysis. Some preprocessing was done to extract only the relevant columns, which corresponds to damage grades at different dates. It is important to mention that due to the high cost of carrying out this exercise, annotations are only available as point-coordinates and not pixel-wise format. The original image contained information about the following dates:

- 23 September 2013
- 23 May 2014
- 1 May 2015
- 26 April 2016
- 18 September 2016

These building destruction annotations were produced by the UN Satellite Office after manual inspection. We know, from our contacts at UNOSAT/UNITAR, that there is 100% coverage of all damaged buildings within the “analysis zones” of Aleppo. The “no analysis” zone, which involves non-civilian regions in the city, have to therefore be excluded from the analysis. Both damage annotations and the analysis zones are available in shapefile format.

The point wise nature of the data provides a challenge. Ideally, we would want the entire dataset to be pixel-wise annotated with the corresponding destruction labels between

2011 and 2017. Such a dataset would facilitate easy model training and, therefore, identification of destruction footprints in new, unobserved city images. However, annotating large volumes of data is expensive, and UNOSAT could only afford to provide point coordinates.

We take this nature of the data into account during the labeling stage.

Box 1: EDA of damage annotations

We performed an exploratory data analysis (EDA) on the damage annotation shapefiles. While performing EDA, we found an outlier date that did not correspond well with annotation from neighboring dates regarding the continuity of assigned labels.

For this particular date (2015-05-01), the labels differed from both neighbors (2015-04-26 and 2016-09-18), even though the neighboring dates were in agreement.

This finding was tested by calculating Cohen's Kappa coefficient score. The score gives us a quantitative measure of the agreement of two sets of labels / annotations / ratings. The score ranges between -1 and 1. A low score implies that the labels assigned are as good as random, while a score of 1 implies a perfect match.

Cohen's Kappa coefficient is computed using the following formula:

$$\kappa = (p_o - p_e)/(1 - p_e)$$

where p_o is the relative observed agreement among raters, and p_e is the hypothetical probability of chance agreement.

Dates	**2015-04-26**	**2015-05-01**	**2016-09-18**
2015-04-26	1	-0.01	0.215
2015-05-01	-0.01	1	-0.003
2016-09-18	0.215	-0.003	1

Table : Cohen's Kappa Coefficient of the annotations for different combinations of dates to identify outliers.

Cohen's Kappa for the center date (2015-05-01) and the neighboring dates, individually, was low while that for the neighboring dates in isolation was high. This suggests that the center date did not agree with the events' continuity.

This would lead to a lot of patches between two dates where labels would be uncertain, and patches would need to be dropped. For the purposes of our analysis, we have therefore decided to drop annotations from the date "2015-05-01", and assume that damage grades from 2015-04-26 will hold until the next annotation date is reached. To learn more about how labeling works refer to the labeling section.

The image below provides a quick overview of the nature and variety of the data we are working with.

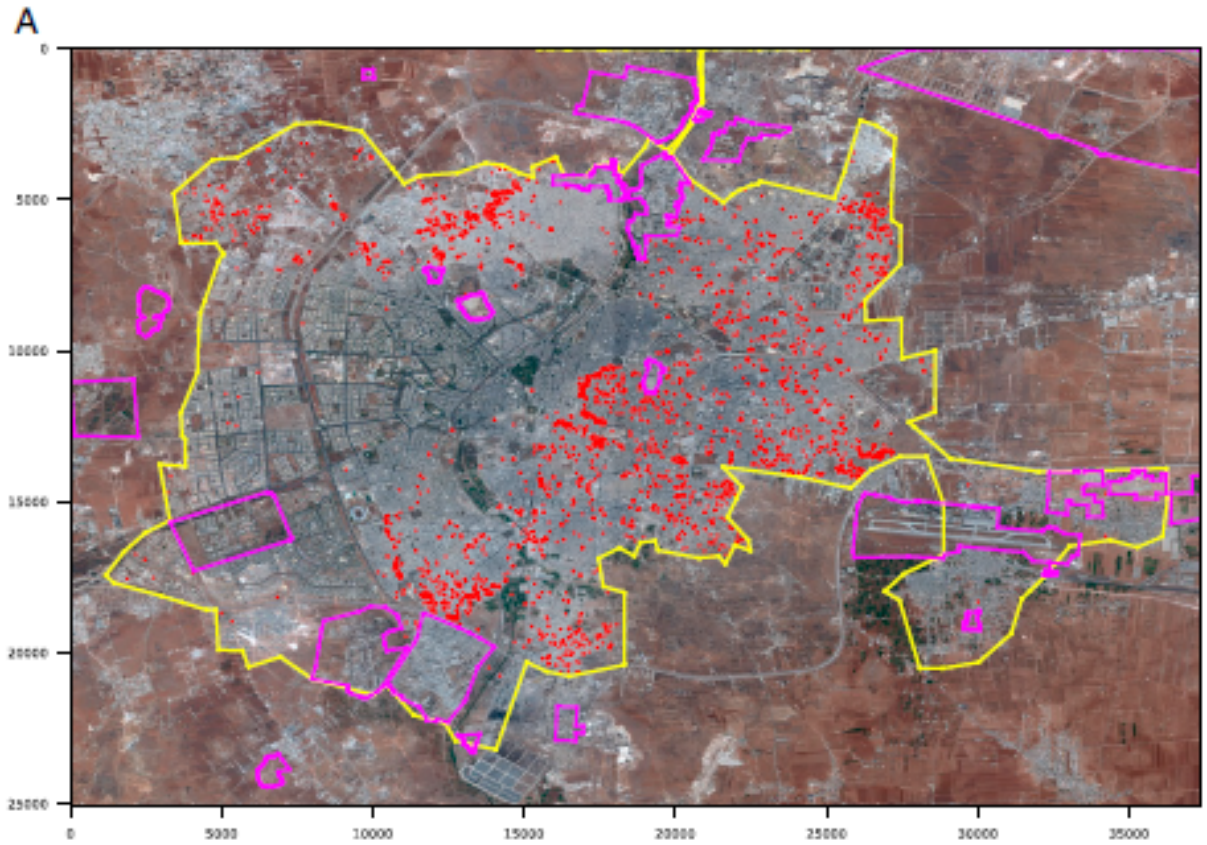

Figure: Imagery of Aleppo on September 18, 2016. Red dots indicate UNOSAT

annotations as destroyed. Areas enclosed by magenta lines are no analysis zones, excluded from the UNOSAT damage assessment due to being non-civilian. The yellow line encloses the populated areas of Aleppo under analysis. Sources: Google Earth/Maxar satellite imagery and UNITAR/UNOSAT damage annotations. A shows an overview of the urban area of Aleppo. B shows an area in central Aleppo close to the Citadel.

Region of Analysis

Ideally, we would want to carry out this analysis in the entire city of Aleppo, but due to compute constraints, we decided to conduct the analysis, only in a portion of the city, as shown in the image below:

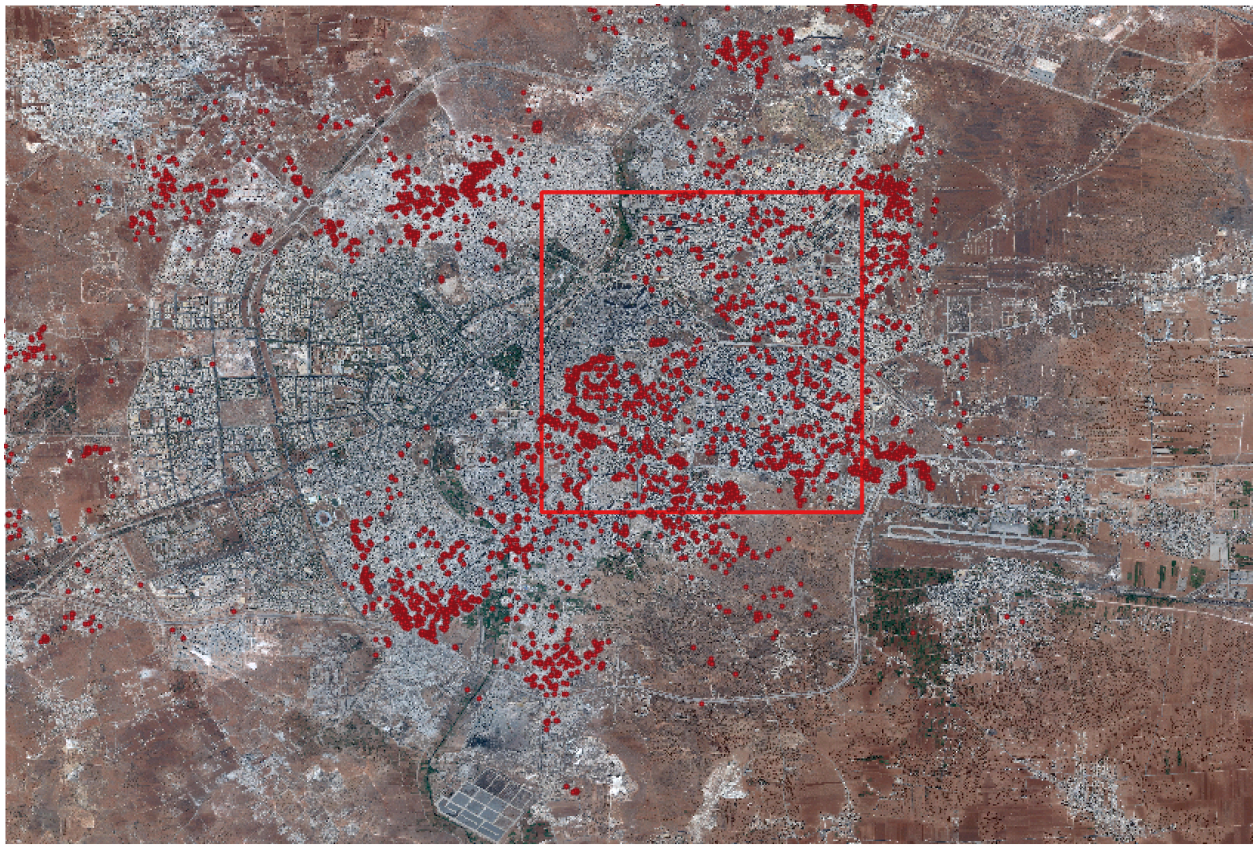

Figure: Area in the red box indicates our analysis region. Red dots indicate “severely damaged” buildings (see Sample Splitting and Labeling).

Moving forward, all numbers reported will be based on the region of analysis.

Data Preprocessing

Satellite image coregistration

Satellite images are not always perfectly aligned because of the fact that they are captured by satellites roaming the earth at different points in time. Considering our analysis will work on pixel-based values, small pixel-based shifts might lead to the introduction of a lot of noise. To account for this we have used the arosics package, to coregister the images. The image from date 26 June 2011 was used as a reference image to correct alignment for the rest of the images.

Patching

We divide every image corresponding to the analysis region into 128x128 pixel patches, generating hundreds of small zones within each satellite imagery.

Sample Splitting

We then create a master sample raster file, that allocates each of these patches to be in the training, test, or validation datasets.

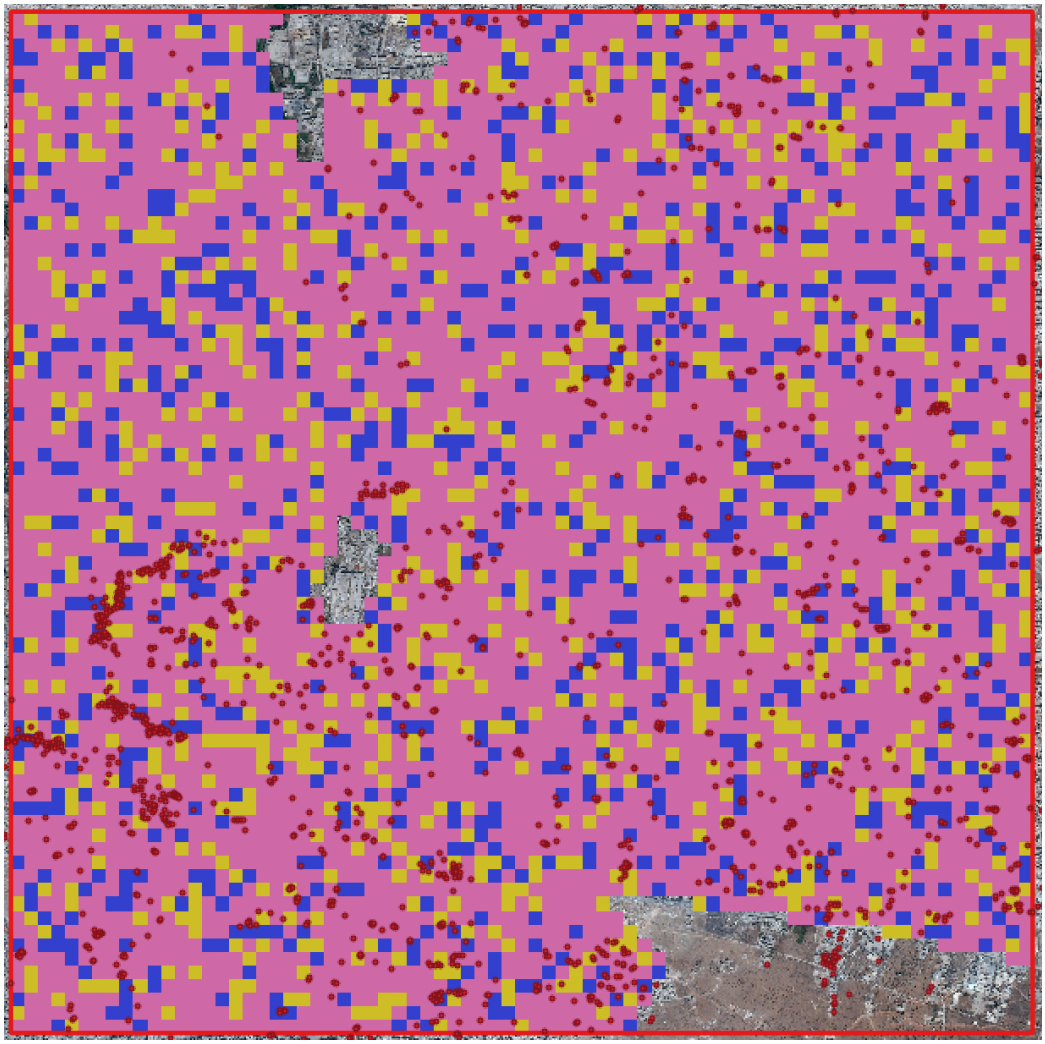

Figure: Analysis region divided into 128x128 patches and divided into train, test, validation, and no-analysis patches. Pink=Train; Orange=Validate; Yellow=Test

Patch Generation

For each satellite image, we now proceed to import the image, and generate a numpy array at the patch which we store it to disk on folders corresponding to the set in which they belong.

Details, including the code can be found here.

Labeling

The label values assigned to the civilian settlement areas during the annotation phase were based on the extent of destruction:

1 - low damage (low)
2 - moderate damage (medium)
3 - severe damage (high)
0 - no destruction

In our analysis, we only considered ‘severe damage’ as the positive class and re-labeled the others as having no destruction or hailing from the negative class. The other labels were dropped because upon inspection, it was observed that at the satellite imagery resolution level, significant differences could not be identified in the patches labeled as having severe damage, moderate damage, or no destruction.

Label Augmentation

Satellite images don’t always coincide with the dates for annotation. This presents a problem, as each patch needs to be given a label (“damaged” or “not damaged”) To get around this, we propose a label augmentation technique based on the following assumptions:

1. There was no reconstruction in Aleppo between 2011 and 2018
2. If a patch is labeled as damaged in an earlier date, it will be damaged in the later date as well.
3. In case a patch’s damage grade changes between two different dates, all patches in between those two dates will need to be dropped from the analysis, as it is uncertain when the changes occurred.

Given this logic, we now go ahead and assign damage labels to every patch in the training, testing, and validation datasets and store them for later use.

This approach helps solve two problems - we can expand the size of the training set by boosting the number of data labels and including additional time periods, thereby improving the model’s availability to handle domain shift. It is important to note that overall, our label augmentation approach is conservative in that the labels are assigned only for the uncertain class.

Balancing the data

Following contextual label augmentation, exploratory data analysis (EDA) revealed a significant class imbalance in the dataset. This is understandable, given that a majority of the images either do not feature destruction or were not completely destroyed. As a

result, we now have an over-representation of the negative class in the dataset. In the empirical context, the level of imbalance is extreme.

Dataset	Total Number of patches	Patches Labeled as Damage	Patches labeled as not damaged
Training Set (70%)	96642	7208	89434
Validation Set (15%)	20774	1454	19008
Test Set (15%)	20462	1510	19264

Even though our region of analysis suffered vast destruction, only 8% of the patches of civilian settlement zones contained a building classified as destroyed by the UNOSAT.

Moving forward with an imbalanced dataset to train an automated building-damage classifier will likely lead to a high false positive rate (FPR). Even if FPR is low, a high absolute number of false positives will yield misleading destruction predictions. We overcome this issue by sampling up from the positive class.

To deal with this class imbalance, we decided to upsample the positive class so that there is a 50:50 representation between the damaged and undamaged patches. We then shuffled the training, testing, and validation sets. The dataset is now ready for analysis.

Box 2: The balancing problem

Traditional measures of model evaluation don't work really well with unbalanced data. The following example outlines the problem.

- **Implication:** Traditional metrics of model performance (Accuracy, TPR, FPR) will not be indicative of how well the model is doing.
- **Example**: 100,000 samples, 1,000 destroyed
  - Recall/True positive rate

$$TPR = \frac{TP}{TP + FN} = 90\% \rightarrow 900\ TP$$

  - False positive rate (share of 0's *not* predicted correctly)

$$FPR = \frac{FP}{FP + TN} = 15\% \rightarrow 14,850\ FP$$

  - Precision in unbalanced sample only 6%, but in balanced sample would be 86%!

For our analysis, we will therefore report average precision in unbalanced samples along with the AUC score.

Feature Extraction

The dataset under assessment comprises satellite images of the settlement areas in Aleppo in Syria, with a population of 1.8 million people. They are all very high resolution images as each pixel represents approximately 50 cm x 50 cm of area. These images have several elements, such as buildings, roads, houses, roundabouts, open spaces, junctions, shadows, vehicles in transit, etc. Across the city, thousands of instances of these elements vary spatially and temporally in terms of size, orientation, and lighting intensity (brightness) in the images. For the purposes of identifying building destruction, many of these elements are likely to add a lot of noise to the model and affect its overall learning ability. They contribute to reducing the signal-to-noise ratio for the dataset.

Histogram of Gradients and Laplacian

We address the noise contributions from these extraneous elements by using techniques that extract and enhance features corresponding to the signal's key elements. We deployed two methods to extract and enhance features from the images: Histogram of Gradients (HoGs) and Laplacians.

Histograms of Gradients (HoGs) capture both the gradient and the orientation of the highest frequency change. In our image dataset, they highlight the edges as well as the direction of gradient change, thereby reducing noise.

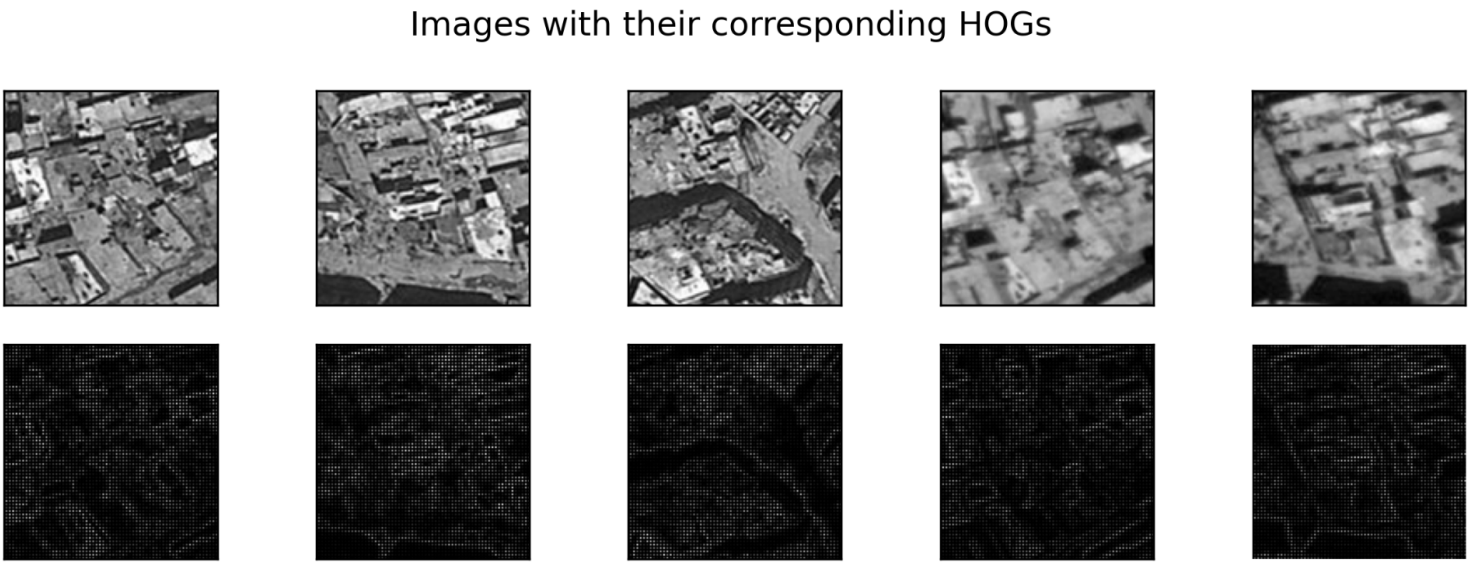

Figure : (top row) Sample patches of civilian settlement from Aleppo (grayscale). (bottom row) HOGs of the sample patches highlighting the edges and orientations of change.

Laplacians isolate features corresponding to specific frequency ranges. For our model, we are interested in enhancing the features corresponding to the high-frequency range - edges and other sharp transients. Feeding sharpened images with features corresponding to the highest set of frequencies from the Laplacian stack to the models facilitates accelerated pickup of the relevant patterns behind destruction.

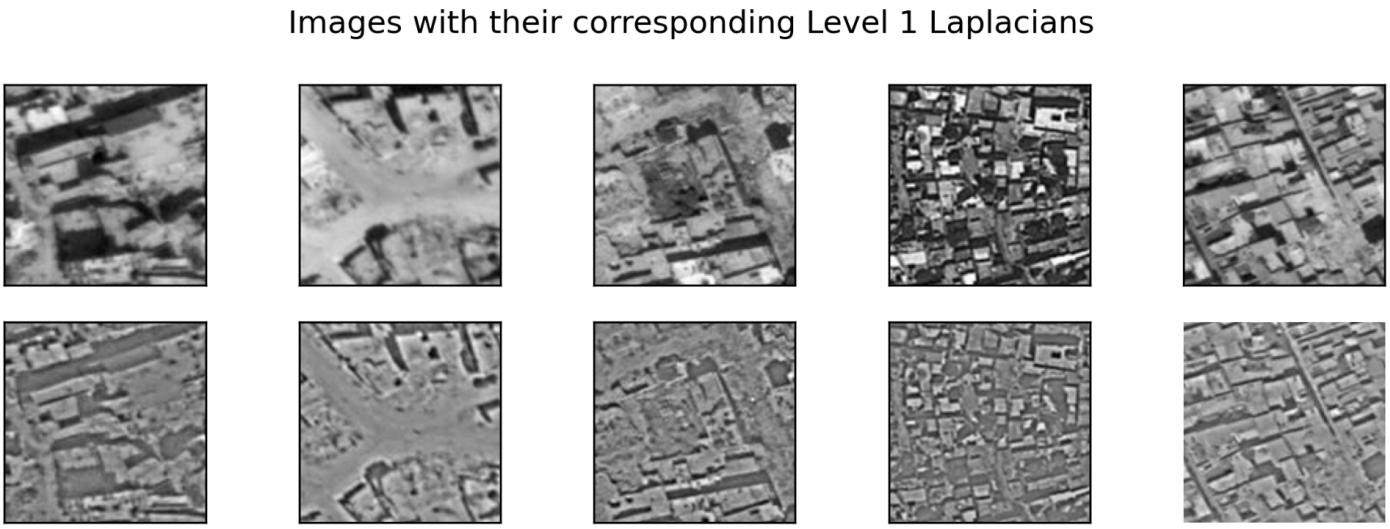

Figure : (top row) Sample patches of civilian settlement from Aleppo (grayscale). (bottom row) Level 1 Laplacians of the sample patches highlighting the sharp features corresponding to higher frequencies.

Dimensionality reduction using Principal Component Analysis (PCA)

Though feature extraction enhances and highlights features of interest, the images continue to harbor many dispensable dimensions that do not capture the essence of the dataset. We use PCA to identify new dimensions that best capture the variation in the dataset. In the reduced dimension hyperspace, PCA gives us a set of new basis vectors that take up less space but retain most of the information from the original image.

PCA of Laplacians

Satellite imagery is inherently very noisy, and this is reflected in the dimensionality reduction exercise we carried out for the laplacians. We found that despite even the first 500 principal components was only able to explain 46% of the variance. See image below.

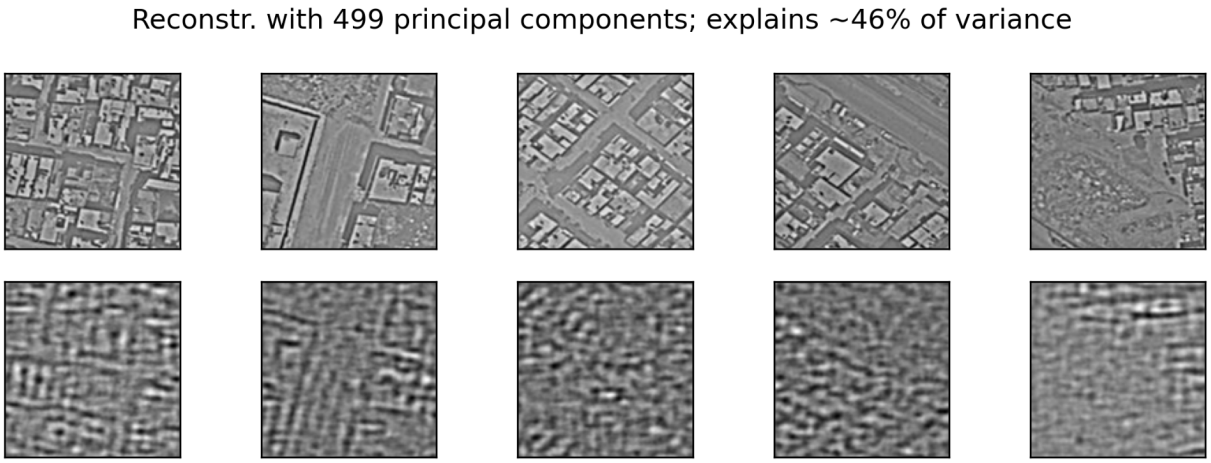

Fig _ : (top row) Laplacians of sample patches of civilian settlement from Aleppo. (bottom row) Laplacians reconstructed using first 499 principal components out of an initial 16,384 dimensions from the sample patches above.

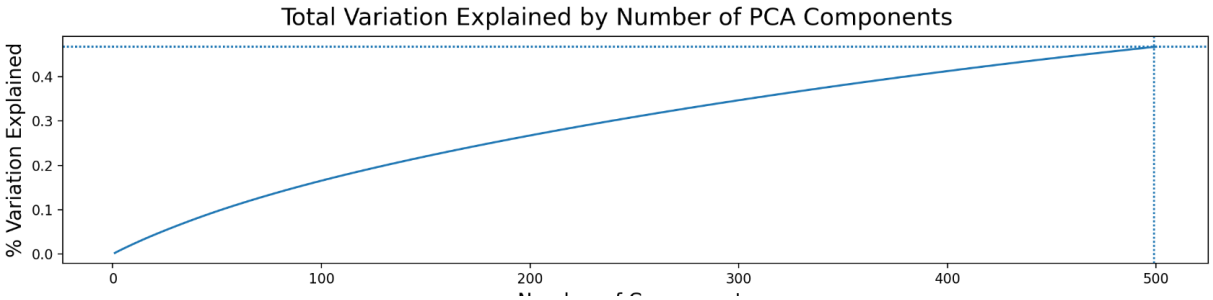

***Figure : Trend in the % of variation explained by increasing the number of principal components while running PCA on the dataset*.**

PCA of HOGs

We are still running the code for getting PCAs of HOGs. This section will be ready by 12PM midnight. However, we do assume that the PCA exercise will yield similar results as those for laplacians.

t-SNE visualization of features

The t-SNE algorithm aids in visualizing higher dimensional data by prioritizing the retention of local structure from the high dimensional space in its projection to the two-dimensional space. It does this by keeping data points close in the low-dimensional space that were close in the high-dimensional space.

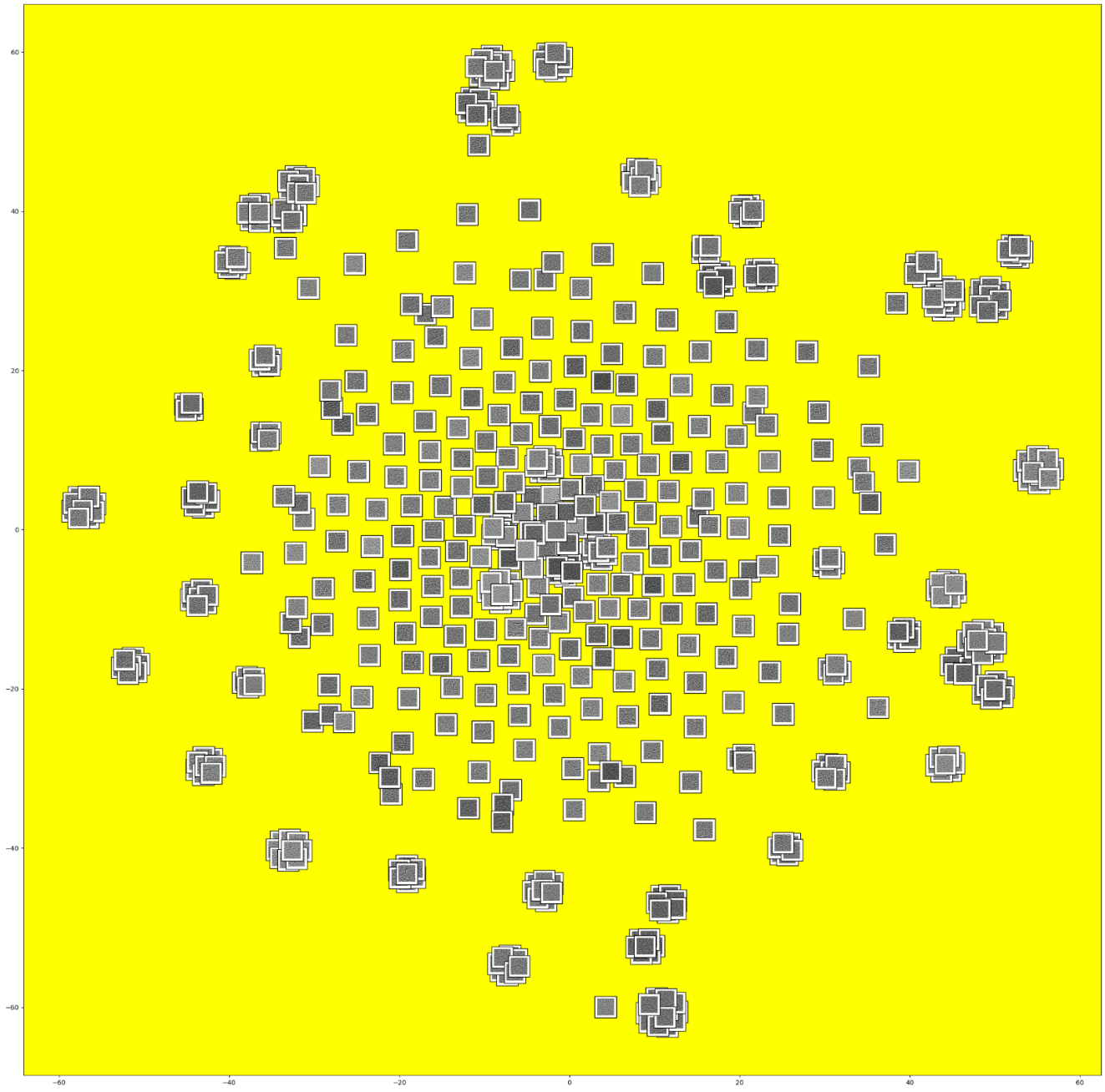

Figure : t-SNE 2D projections for laplacians of images

Close examination of the t-SNE visualization reveals how the method has clustered image patches having similar features. For eg. Roads in similar orientation, density of settlement areas, presence of roundabouts, etc. Many of these clusters also comprised

images of the same region from different years given how they are likely to be close to each other in the hyper-dimensional space. The remaining scatter of the

Conclusion
While ideally being able to use principal components would greatly speed up model training, we haven't found enough evidence that suggests that taking advantage of principal components will isolate the signal from the noise. Therefore in our classification step, we will be using the raw images, their HOGs, and their Laplacians, and refrain from using the Principal Components.

Classification

Following pre-processing, sample splitting, label augmentation, balancing, feature extraction. and dimensionality reduction, the resultant city-level images patches of 128x128 pixels (representing roughly 64m x 64m on ground) are now ready to fe bed into the classification model. Here, we train, validate, and test two computer vision algorithms to classify destruction for image patches. These include three variants of the Siamese Neural Network.

1. Siamese Neural Network between the Raw Image: Raw unprocessed patches from before the war are paired with raw unprocessed images from during the wars. Labels for each pair correspond to the label in the post image (as pre-war damage is assumed to be 0)
2. Siamese Neural Network between the Laplacians of Raw Image: Laplacians patches from before the war are paired with laplacians of patches from during the wars. Labels for each pair correspond to the label in the post image (as pre-war damage is assumed to be 0)
3. Siamese Neural Network using HOGs of Raw Image: HOGs of patches from before the war are paired with HOGs of patches from during the wars. Labels for each pair correspond to the label in the post image (as pre-war damage is assumed to be 0)

Siamese Neural Network (SNN)

An extension of the convolutional neural net based approach, the siamese neural network model takes two images as inputs, where the parameters on the two convolutional sub-networks are constrained to be the same. For each optimization iteration, the parameter updates are computed using the average gradients across the two sub-networks. As mentioned above, For our problem space, we constrained the first image to always be a "pre" image, i.e., an image from a time when we knew there was

no destruction. This allowed us to reimagine the model training step as a change detection model, and the goal for the Siamese network is to learn how to detect changes – or differences – between image patches for a given location taken from two different points in time.

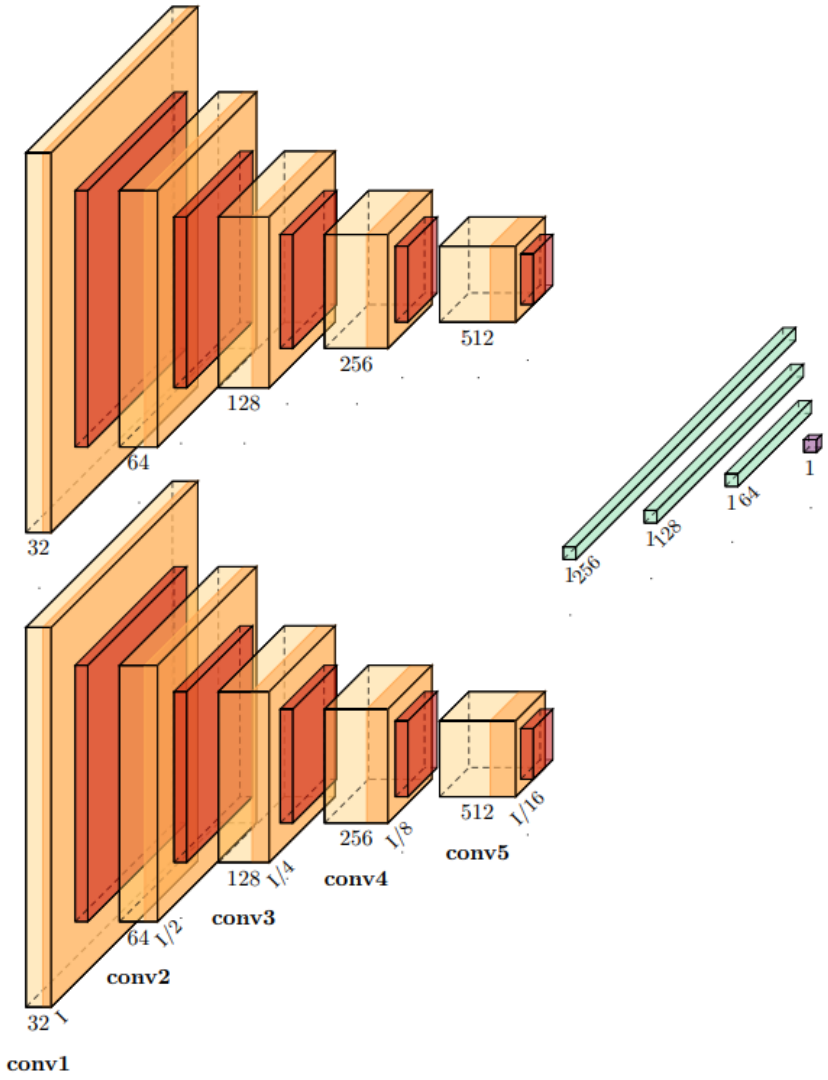

Fig: Model Architecture of a Siamese Neural Network that takes in two vectors, feeds them to two separate neural networks and works to reduce the euclidean distance between the resultant encodings.

The architecture in a little more detail

The convolutional blocks contain two convolution filters, 1 ReLU activation, and 2 × 2 maximum pooling. At the end of the conpolution blocks, two encodings are generated for which the model then computes the euclidean distance between the encodings, takes the sigmoid to be the final prediction. It then iterating over epochs to reduce the

associated loss. All convolutions are padded, each block uses batch normalization (Loffe and Szegedy 2015) and spatial dropout (Tompson et al. 2014). Parameters are estimated by minimizing the focal binary cross-entropy (Lin et al. 2020) using the Adam algorithm (Kingma and Ba 2015).

Using validation AUC as our measure of model performance, we employ an early stopping criterion into our training step to reduce training time. If the model fails to observe an increase in validation AUC for 4 subsequent epochs, we stop the training, and retain the weights from the epoch with the best validation AUC score is retained.

Hyperparameters

The SNN was fed images of patch size 128x128 with each batch comprising 32 samples. Binary Cross-entropy was selected as the loss function A dropout rate was randomly selected from a linear uniformly spread range between 0.1 and 0.15 to ensure that the model does not overfit the training set. Three different learning rates were used to determine the optimum step size for every iteration of the model. While training the model, we maintained a record of the validation set AUC score to decide when to terminate the epochs. The values outlined in the table have been arrived at after repeatedly experimenting in a smaller subset of the data. However, because of the inherently high variability in the data, this approach does not guarantee that the values chosen are in fact the most optimal. There is more scope for hyperparameter tuning in future revisions..

Hyperparameter	Range/Value	Notes
Batch Size	32 samples	# of samples in a batch
Learning Rate	[0.002, 0.003, 0.004]	Experimenting with different values for optimizing speed
Droupout	[0.1, 0.15]	Random number from this range to avoid overfitting
Epoch Size	[70, 100]	Max # of epochs to be chosen from between the mentioned range
# of convolution layers	5	Reached at after experimentation
# of dense neural network	5	Reached at after experimentation

layers		
Filters	8	

Results

Three different variants of Siamese Neural Networks were trained, validated and tested. They differed in the input vectors that were fed to them - raw images, HoGs of the images and Laplacians of the images. The key metric we considered to assess the model performance in our case was the Reciever Operator Characteristic - Area Under the Curve (AUC) score. In addition we have also evaluated precision, recall and accuracy to assess and compare different models.

Results - SNN with Raw Images

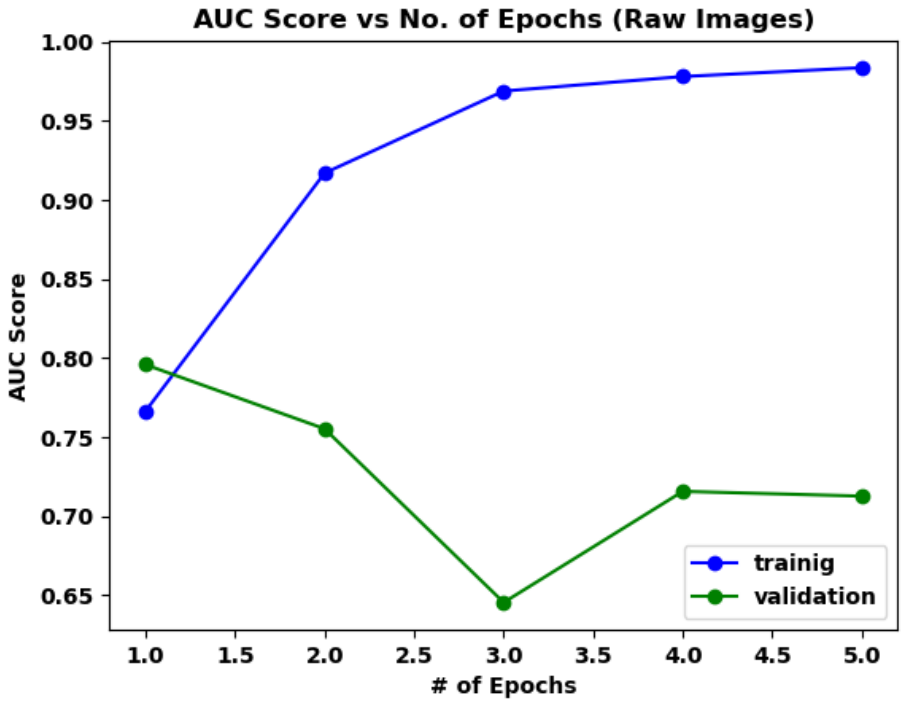

Figure : Trend in Training and Validation AUC Scores with Number of Epochs for SNN trained on raw images.

Raw Images: Across the five epochs, the AUC score for the training set continuously improved while the validation score decreased. The behaviour observed suggests that the model was heading towards the overfitting territory which is why subsequent

epochs were not run. We finally settled with the SNN model corresponding to a training set AUC score of 0.76 and a validation AUC score of 0.797

Results - SNN with Laplacians

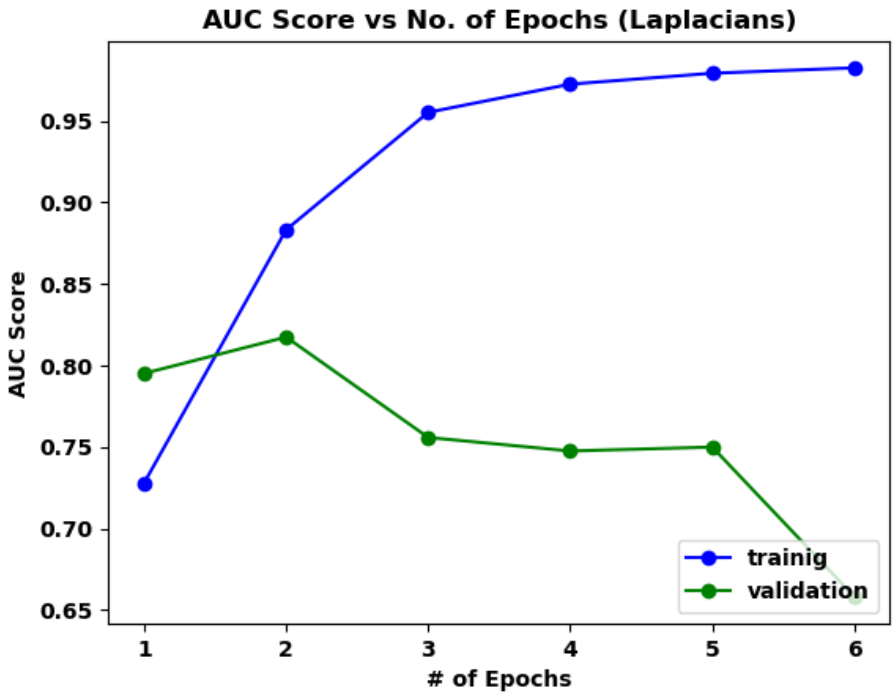

Figure : Trend in Training and Validation AUC Scores with Number of Epochs for SNN trained on Laplacians of the raw images.

Model results for Laplacians of the raw images showed a similar trend in the training and validation AUC scores. However, the performance on the Laplacians lacked behind the raw images. The best training set and validation AUC scores obtained were 0.73 and 0.79. Here again, the model started drifting towards the overfitting territory as it gave lower validation AUC scores with increasing epochs.

Results - SNN with HOGs

We are still running the code for getting results for SNN with HOGs. This section will be ready by 12PM midnight. Our prior here is that SNNs with HOGs will perform better than Laplacians.

Overall Results

Model	Precision* (0/1)	Recall* (0/1)	F1-Score* (0/1)	AUC	Average Precision
SNN (Raw Image)	0.98 / 0.13	0.54 / 0.88	0.7 / 0.23	0.81	0.16
SNN (Laplacian)	0.97 / 0.18	0.76 / 0.67	0.85 / 0.28	0.79	0.15
SNN (HoGs)	TBA	TBA	TBA	TBA	TBA

* Assuming a probability threshold of 0.5 for classifying images as destroyed.

Discussion

One of the biggest challenges we faced during the exercise has been the massive size of the data that we are working in. The computational requirements are too heavy, and work needs to be carried out to make feature extraction less costly. Early results seem to indicate that keeping all things constant, Laplacians fare no better than the original image when it comes to increasing the predictive power of our model. However, based on our own experimentation on a smaller subset of the data, we do expect HOGs to do better.

Due to memory constraints, we considered only one city - Aleppo - out of eight cities that suffered major war destruction in Syria. This has a direct hit on generalizability of the model, as other cities might not have similar geographical features, maybe be more densely or sparsely populate, and may contain other man made or natural artifacts that were not present in our dataset. Ideally, we want to test the predictions on a city which the model has not learned from and achieve generalizability. However, our models learned the key destruction features from Aleppo alone which may not be similar for other cities .

Given enough memory and compute power, we hope to build high-performing models that can spot destruction patterns in regions that the model has not yet been exposed by training it with a larger and more diverse dataset of satellite images. Out-of-sample validation would therefore help increase the robustness of the results, thereby achieving enhanced generalizability.

In our approach we arbitrarily chose a threshold of 0.5 and could have experimented with a range of values to determine the optimum assignment for best model performance. Due to memory and compute power constraints we had to limit ourselves to a singular consideration of the probability threshold.

Experimentation with dropout rates to avoid overfitting. We considered a conservative range of dropout rates while training the SNNs. In the future, we can consider higher dropout rates to achieve higher overall performance

References:

1. Van Der Maaten L, Hinton G. Visualizing data using t-SNE. Journal of Machine Learning Research. 2008; 9(Nov):2579–2605.

2. Soner Yıldırım. Hyperparameter Tuning for Support Vector Machines — C and Gamma Parameters. Medium Article [link]

3. Hannes Mueller et al. Monitoring war destruction from space using machine learning. 2021, Proceedings of the National Academy of Sciences, 10.1073/pnas.2025400118 [doi], https://www.pnas.org/doi/abs/10.1073/pnas.2025400118