Satellite Imagery Project

Team #13

BN	Sec	الاسم
35	**1**	عبد الرحمن أشرف محمد
36	**2**	وليد محمد هشام
20	**1**	خالد مصطفى السيد
20	**2**	محمد وليد فتحي

Dataset

Firstly, we divided the dataset into 80% training set and 20% validation set for both classic and deep learning training, then for deep learning model, we divided the dataset into 5 parts and did cross validation with 4 folds for training and a fold for validation and repeated the training 5 times with different validation fold each time.

Traditional Approach

Image Differencing:

Subtracted the two input images as mentioned in lecture slides, for each image determined a threshold for change.

The choice of threshold according to **Tim Ellis** and **Paul L. Rosin** in the paper ***Image Difference Threshold Strategies and Shadow Detection,*** was parameter-less depending on robust statistics of the difference image, and not assuming any distribution.

$$D_{x,y} = |I_{x,y} - B_{x,y}| \quad (Difference\ image)$$
$$MED = median_{x,y \in I}\, D_{x,y} \quad (Median)$$
$$MAD = median_{x,y \in I}\, |D_{x,y} - MED| \quad (Median\ Absolute\ Deviation)$$

Assuming less than half the image is in motion the median should correspond to typical noise values

$$T = MED + 3 \times 1.4826 \times MAD \quad (Threshold)$$

where 1.4826 is a normalization factor wrt. a Gaussian distribution. And the scalar 3 is based on the assumption that in normal distributions 99.7% of the data lies inside that threshold

This approach yielded, 18% Jaccard score on the Training division, and yielded 16% Jaccard score on the Validation division.

Limitations of Traditional Approaches:

The dataset isn't for detecting changes in general, it just captures changes in buildings, this made classic approaches that treats all pixels equally result in bad scores as they capture changes in general.

Other Tested Approaches

PCA K-means

This approach makes use PCA followed by K-Means. The image is traversed to extract image pixel values, for these pixel values make them into feature vectors (A vector for each block) Then a normalization step follows to normalize the feature vectors. Dimensionality of the feature vectors is reduced by using PCA that keeps 90% of the information represented by the feature vector.

Then K-means with 2 clusters follows to assign 0s and 1s based on the change values. Based on the dataset, higher number of pixels in a cluster should indicate that this cluster is unchanged. Thus, values are assigned to these pixels accordingly.

We also tried following that PCA K-means approach by edge detection to detect building like figures, but that didn't boost the Jaccard score.

Limitations of PCA K-means:

The choice of 2 clusters leads to the change map always having the 2 values 0 & 1 (255) which is mostly not the case, as most images (66% of the whole dataset) had completely dark labels. So, this has affected the accuracy (Jaccard score) of this approach.

CVA

Implemented but had the same threshold issue as image differencing, hence abandoned since it led to similar results initially and was easier to find a threshold for the image differencing approach.

Edge Detection

We intended to detect edges of building like structures and keep them, while labeling the rest of the image as unchanged, but this was difficult to test, since most of the outputs were completely dark, and those that had actual changes, had them mixed with other unnecessary changes. And thus, that approach was abandoned as well.

We were able to test it later on but it didn't enhance the score.

Registering Images by GCP

Attempted to register the latter image to the former one prior to image differencing, but the lack of GCPs made it nearly impossible to apply that registering.

Deep Learning Approach

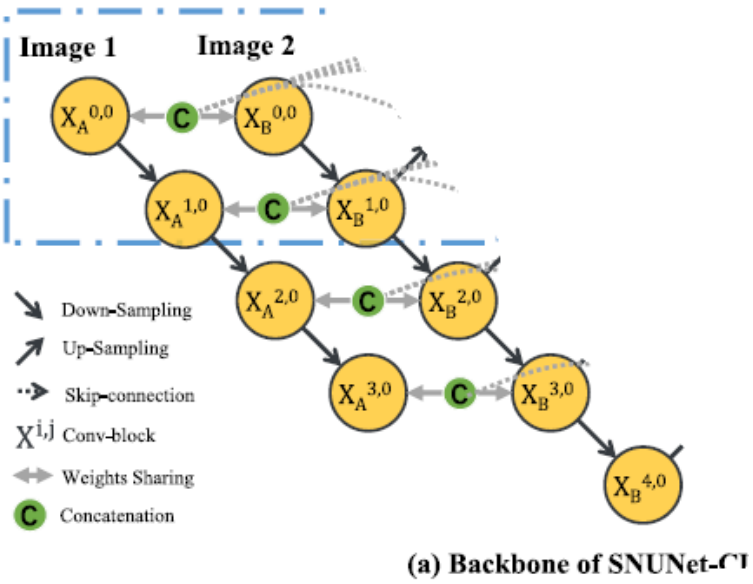
Used Siamese UNet-ECAM model.

Encoder

This model uses a **Siamese** network branch in the encoder of the **UNet**, a Siamese network takes two input images and fits models to them, with the updates in a path mirrored in the other path, Siamese networks in general are utilized in capturing the similarity between two images (as mentioned by **Andrew NG** in Siamese network explanation)

In change detection, Siamese networks have proved to have better interpretability than other methods that just concatenate or subtract the two input images. And since it shares the same weights in two branches, it uses the same method to extract features of two images. The image pairs are generally obtained with the same device and in the same area, so there is a certain similarity between the two images. Therefore, it is reasonable to extract features using the same method. In addition, the weight sharing makes the network have fewer parameters and converge faster.

The concatenation of the two feature maps is used to fuse the features between two Siamese branches to ensure the integrity of the information.

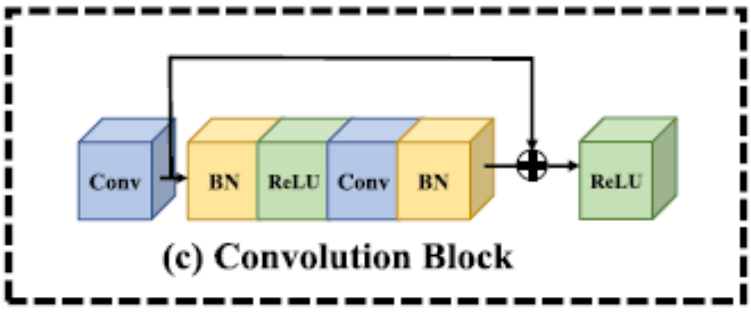


Encoder Half of the Network

Convolution Block

This convolution block has several elements, first **convolution**, it applies convolution to the input images followed by **Batch Normalization** which normalizes batches so that training doesn't get much affected by an anomalous batch and provides a form of regularization to the network.

The residual branch in the block helps the network converge faster.



Max Pooling

The convolution block is naturally followed by a pooling layer to reduce the dimensionality of the feature maps.

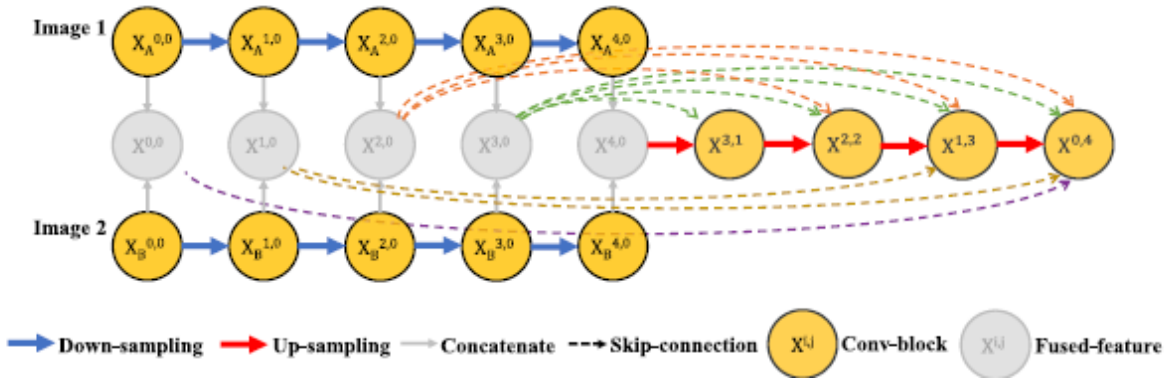
UNet++

UNet++ is utilized which differs from the ordinary UNet in the way it concatenates outputs from encoder in several decoder subnetworks in order to output several outputs, one corresponding to each decoder subnetwork.

Shallow subdecoders often capture fine grained features whereas deeper capture coarse features, we can make use of both.

Upsampling

Outputs from the second encoder branch are upsampled, these upsampled outputs are concatenated with the encoder outputs of the previous layer and contribute to the outputs of the corresponding subdecoders.

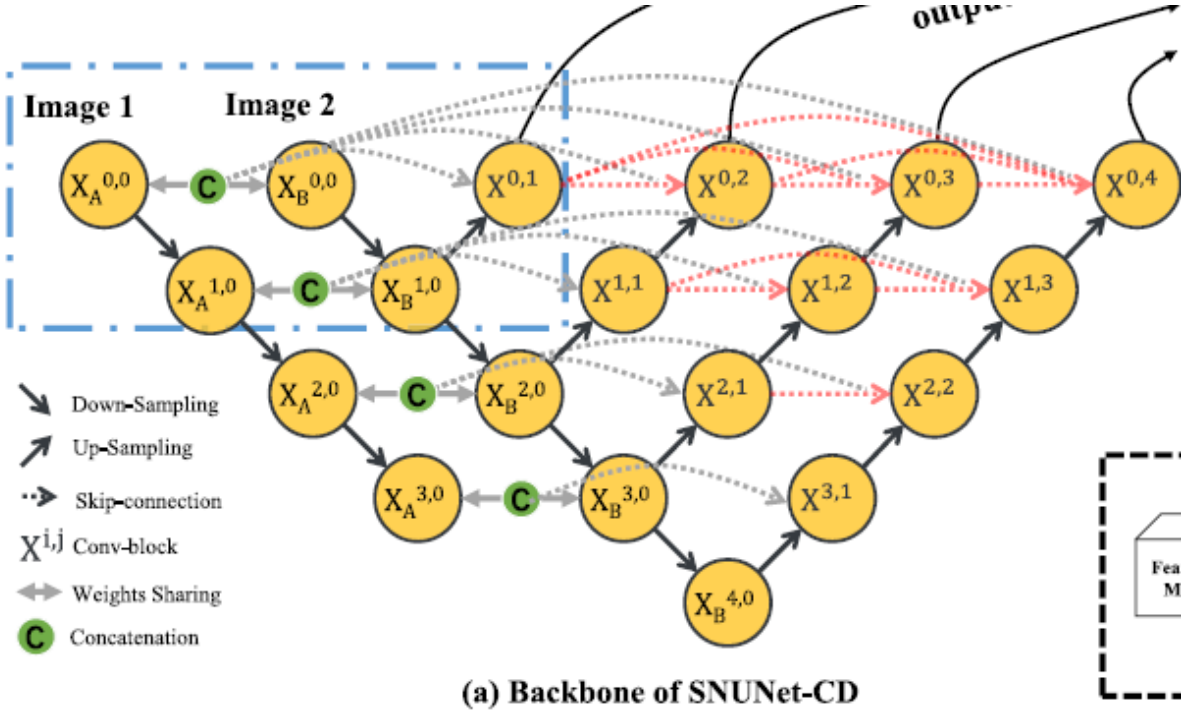


Structure of a Subdecoder

Equations Involved thus far

$$x^{i,j} = \begin{cases} \mathcal{P}(\mathcal{H}(x^{i-1,j})) & j = 0 \\ \mathcal{H}([x_A^{i,0}, x_B^{i,0}, \mathcal{U}(x^{i+1,j-1})]) & j = 1 \\ \mathcal{H}([x_A^{i,0}, x_B^{i,0}, [x^{i,k}]_{k=1}^{j-1}, \mathcal{U}(x^{i+1,j-1})]) & j > 1 \end{cases} \quad (1)$$

Where H represents a convolution block, P represents a 2*2 max pooling. U represents upsampling, and square brackets([ ]) represents concatenation on the channel dimentsion.



Complete UNet++ architecture

ECAM

Stands for Ensemble Channel Attention Module.

As shown in the figure above, 4 outputs are produced from the decoder part each has the same size as the original images.

The outputs of shallow sub-decoder have finer grained features and more precise localization, and the outputs of the deep sub-decoder have more coarse-grained features and richer semantics.

When fusing these features automatic channel strategy is needed (**ECAM**).

ECAM produces two outputs, M_intra and M_inter:

- The term "minimum intra-channel attention" refers to the attention scores computed within individual channels of a set of feature maps. It represents the degree to which different channels within each feature map are attended to or emphasized based on their importance for the task. (Importance of features within each feature map)
- The term "minimum inter-channel attention" refers to the attention scores computed across different channels within a set of feature maps. It represents the degree to which different channels within the feature maps are attended to or emphasized based on their relevance to the task. (Importance of each feature map among different maps)

Each ECAM ensembles a few feature maps, and inputs them to a CAM module.

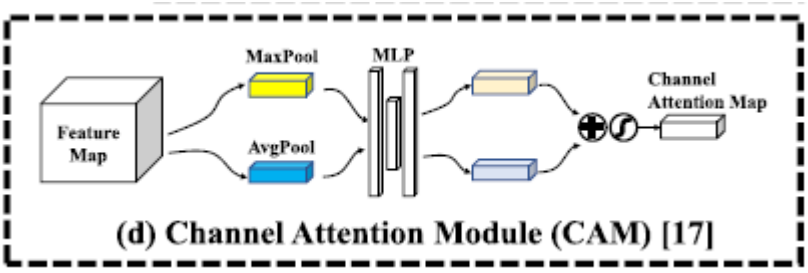
Equation for the CAM module is:

$$CAM(F) = \sigma(MLP(AvgPool(F)) + MLP(MaxPool(F)))$$

It applies AvgPooling and Maxpooling to the input feature map, inputs each pooling to a multilayer perception, then sums the two outputs and inputs them to a sigmoid function.

The **Avg pooling** is sensitive to the frequent presence of features across the feature map.

The **Max pooling** is sensitive to peak values of the features, and thus is robust against small shifts in the features.



(d) Channel Attention Module (CAM) [17]

ECAM Equations

$$M_{intra} = CAM(x^{0,1} + x^{0,2} + x^{0,3} + x^{0,4})$$

This equation captures the important features in each feature map.

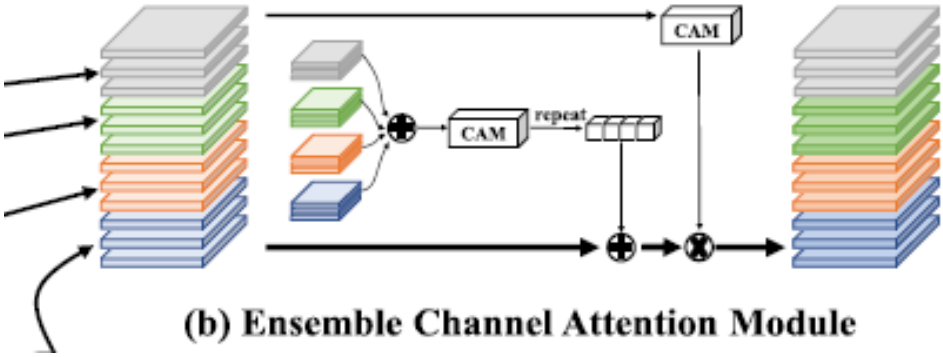
$$F_{Ensemble} = [x^{0,1}, x^{0,2}, x^{0,3}, x^{0,4}]$$

$$M_{inter} = CAM(F_{Ensemble})$$

This captures the important features across the 4 feature maps.

$$ECAM(F_{Ensemble}) = (F_{Ensemble} + repeat(4)(M_{intra})) \otimes M_{inter}$$

Then the M_intra output is repeated 4 times to match the dimensions of the ensembled map and added to each other. All that is then multiplied element-wise with the M_inter feature map. Now, we have represented the intra feature map features as well as the inter feature map features.



ECAM

Loss Function:

Used a hybrid loss function that utilized both Jaccard Loss and Cross Entropy loss, The use of Jaccard loss came due to the following reasons:

1. It handles class Imbalance
2. In line with the principles of empirical risk minimization in statistical learning theory, the metric used for evaluation should also be optimized during training, and our metric is Jaccard Index, thus we used the Jaccard loss.

The use of and Cross Entropy loss came due to the fact that it's more stable in training, as the Jaccard Loss (and similarly the Dice Loss can get very high values suddenly if the denominator is small).

As we can see, the hybrid loss function combines the strengths of both losses.

Weight initialization

Used He initialization to initialize the weights of the convolution nodes, He initialization counters fading gradient, and is compatible with ReLU as an activation function.

Extra Notes on implementation

Data **Augmentation** is applied on the training set, but it doesn't increase the size of the training set, it just applies a series of transformations to the input images (as well as the label image).

- Horizontal Flip by a 50% probability
- Vertical Flip by a 50% probability
- Rotation by a 75% probability

These three transformations aim at adding more diversity to the training set.

We plotted the Jaccard index of each batch of training and validation to be able to decide on some changes to the model.