Like squinting your eyes: The impact of different fusion modules on change detection with deep learning

Vasil Dakov[1]

Supervisor(s):
Jan van Gemert[1]**, Prof. Dessislava Petrova-Antonova**[2]

Affiliations:
[1]**EEMCS, Delft University of Technology, The Netherlands**
[2]**GATE, Sofia University "St. Kliment Ohridski", Bulgaria**

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 23, 2024

Name of the student: Vasil Dakov
Final project course: CSE3000 Research Project
Thesis committee: Jan van Gemert, Dessislava Petrova-Antonova, Klaus Hildebrandt

An electronic version of this thesis is available at http://repository.tudelft.nl/.

Abstract

Change detection with remote sensing data highlights semantic differences in an area between two or more time intervals. It involves the comparison of aerial photographs of the same location taken some time apart. This facilitates mass scale analysis of urban and rural data over time, including population trends, city expansion trends and illegal building detection. State-of-the-art methods for the task are predominantly deep learning networks, following an encoder-decoder architecture. These architectures all share the trait of having a ”fusion” point - a location in the network where inputs transition from being processed independently to becoming correlated. Fusions can be classified in three categories: early, middle and late, depending on how deep within the network they occur. This study aims to show how changing the fusion impacts the size, spread and number of changes detected. It is motivated by how the receptive field of feature maps in convolutional neural networks expands in deeper layers, extracting features with different complexities. For this, four fusion architectures on three different datasets are compared: LEVIR-CD, HiUCD and a new, fully-controled dataset, CSCD. In terms of test accuracy and the changes' size and spread, results are inconclusive. Which fusion achieves the highest performance varies per dataset. Possible reasons why include the complexity of remote sensing data and general differences between areas, but this is a subject of further study. The only conclusive category is the number of changes detected. On average, all architectures overestimate the number of changes in a scene. When the accuracy of architectures is comparable, however, early fusion overestimates the number of objects changed the most, while middle and late fusion give more realistic estimates. The case study has room for refinement in problem isolation, more data and extending the problem towards more architectures, but is a promising step towards understanding fusion.

1 Introduction

Change detection in remote sensing is the procedure of automatically detecting semantic, meaningful changes from a pair of images of the same location, captured at different times. On a given input, the goal is to predict which and where each object has changed. Changes are defined as the emergence, removal or size change of objects of interest - buildings, roads, green areas, etc. Having a catalogue of such data on a mass scale is of interest, as it facilitates the analysis of urban population trends [1], automatic updating of cadastre maps [2], illegal building detection [3], and more.

Deep learning has become a widely-adopted tool for change detection. This comes about by taking advantage of either (or both) the temporal or spatial information that can be drawn out from the pair of input images. Spatial information can be extracted through the use of segmentation networks like FCN and U-Net [4], [5], [6], while the temporal aspect has been previously tackled by recurrent neural networks [7], and more recently transformers and their attention mechanisms [8], [9].

Despite the abundance of ways to tackle change detection, the majority of deep learning methods share structural similarities. Architectures are often a variation of an encoder-decoder, typically in a Siamese configuration [9]. This goes for both supervised and unsupervised variants [6], [10]. Unsupervised deep learning methods often use autoencoders akin to anomaly detection, while supervised methods with all of the mentioned internal networks have been identified [8], [9]. A trait shared between all of these encoder-decoder architectures is the existence of a ”fusion” step [9], [10]. This refers to the location in a network where the two input images are coupled together, with there being multiple definitions of and types. There has been no identified research of how different fusion modules impact change detection results.

This paper is a systematic examination of the strengths and weaknesses of different fusion modules, with regards to general situations in remote sensing, such as object size or type. The key assumption is that the point of entangling inputs together is a crucial step for how semantic changes between images can be detected. Inspiration stems from the hierarchical model of the brain, and the increasing size of the receptive field at different cortex layers, similarly to the neuroscience basis of convolutional neural networks [11], [12]. Changing the location of the fusion module changes the size of the receptive field at which inputs are entangled. A hypothesis following is that the receptive field's size impacts the types of changes detected, particularly with regard to the changed objects' size and spread. Were there to be a shared trend or systematic difference identified, the benefit would be a better understanding of fusion, and in what change detection situation each architecture is most applicable. Such a conclusion is important, as different fusion configurations for change detection can all achieve decent performance, but some use cases necessitate different strengths. In one scenario, the changes of interest might be at a large-scale such as in the case of agricultural fields, while in others they might be small like houses.

The proposed experiment is to train different architecture configurations on different datasets, and vary the placement of the fusion. The locations tested are (1) - early fusion, prior to being input in the network, (2) - middle fusion, combining inputs within the network structure and (3) - late fusion, entangling the inputs as a post-processing step. The remote sensing datasets used are LEVIR-CD [8] due to its size and prominence for change detection and HiUCD [13] for its categorically labeled data. Additionally, a fully controlled synthetic dataset named CSCD has been generated to illustrate change, featuring varying object sizes and spreads.

The rest of the paper is structured as follows. Section 2 gives the prerequisite knowledge for the rest of the study. Section 3 discusses the experiment setup and the data used for it. Section 4 presents the experiment results and analyzes them. Section 5 discusses the scientific and ethical integrity of the study. Section 6 summarizes the findings and comments on how the study could be continued and improved upon.

2 Background

This section establishes the prerequisite knowledge required for understanding the experiment's setup and consequently the obtained results.

2.1 Encoder-Decoder Networks

Encoder-decoder networks (Autoencoders) are an artificial neural network architecture that learns a parameterized encoding function $E_\theta : \mathcal{X} \to \mathcal{Z}$, compressing the input $\mathcal{X}$, to a latent feature representation $\mathcal{Z}$ and a parameterized decoding function $D_{\theta'} : \mathcal{Z} \to \mathcal{Y}$ that tries to reconstruct the original input. Its primary use cases are dimensionality reduction, noise removal, and anomaly detection [14], but it can also be used to transform inputs to any domain output. Backpropagation spans both the encoder and decoder. The feedforward process of the architecture is shown in Figure 1.

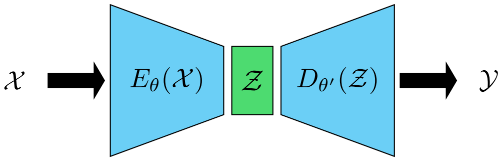


Figure 1: A sample encoder-decoder structure. An input $\mathcal{X}$ gets fed into the encoder, compressed to latent space representation $\mathcal{Z}$, before being reconstructed into $\mathcal{Y}$ from the decoder.

2.2 Semantic Segmentation and Fully Convolutional Networks

Image segmentation is the task of grouping all different objects in an image together. It consists of creating a function $f : \mathcal{X}^{\mathcal{W}\times\mathcal{H}\times\mathcal{D}} \to \mathcal{Y}^{\mathcal{W}\times\mathcal{H}}$ that takes an input image $\mathcal{X}$ of width $\mathcal{W}$, height $\mathcal{H}$ and $\mathcal{D}$ channels. The output image $\mathcal{Y}$ contains pixel-wise labels for k classes, and is once again with width and height $\mathcal{W}$, $\mathcal{H}$. Image segmentation in the context of deep learning makes the neural network approximating f.

FCN [4] and U-Net [5] are foundational semantic segmentation networks. FCN takes advantage of convolutional layers' feature extraction capabilities and the input size invariance they have. The former allows it to differentiate objects' features inside of the image and accordingly classify them. The network is applicable on any size input $\mathcal{X}$, due to there being no dense layers inside it, and the convolutional operation being size invariant. Consequently, the architecture is referred to as a "fully convolutional network". U-Net is a successor to FCN, extending it with skip connections and a more robust upsampling, creating a symmetric, "U-shaped", layer structure that requires less training and performs better.

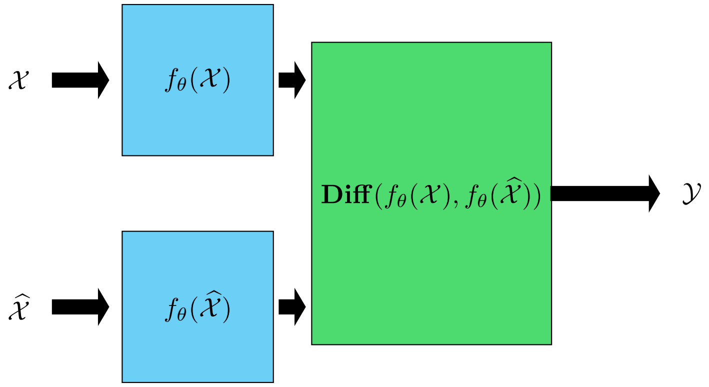
Both FCN and U-Net are an example of an encoder-decoder architecture due to their compression and reconstruction of the input.

2.3 Siamese Networks

Siamese networks are a neural network architecture used for measuring input similarity. Introduced at first for signature fraud detection [15], they compare two inputs, $\mathcal{X}, \widehat{\mathcal{X}}$ to get some output $\mathcal{Y}$, which could be a decision or have its own semantic meaning. A Siamese network consists of two identical multilayer perceptrons with their own function f_θ, typically with shared weights θ, and a differentiation module $\textbf{Diff}(f_\theta(\mathcal{X}), f_\theta(\widehat{\mathcal{X}}))$ [16]. This differentiation module can be as simple as a cosine distance function, or its own classifier. If it is a classifier, the differention module is included in the backpropagation of the network. The forward process is illustrated in Figure 2.

2.4 Morphology

Mathematical morphology is the study of geometrically altering spatial structures [17]. It is additionally a powerful image analysis and processing technique. Uses include noise removal, object extraction/grouping and blob removal.



$\mathcal{X}$
$f_\theta(\mathcal{X})$
$\textbf{Diff}(f_\theta(\mathcal{X}), f_\theta(\widehat{\mathcal{X}}))$
$\mathcal{Y}$
$\widehat{\mathcal{X}}$
$f_\theta(\widehat{\mathcal{X}})$

Figure 2: Diagram of an arbitrary Siamese network. The two input networks are the blue boxes, and inside of them is a representation of a multilayer perceptron. Both $\mathcal{X}$ and $\widehat{\mathcal{X}}$ get fed into their networks, which subsequently feed into the differentiation module to output $\mathcal{Y}$.

Morphological operations assume a binary image as a matrix $A \in \{0,1\}^{\mathcal{W}\times\mathcal{H}}$ for some width and height $\mathcal{W}, \mathcal{H}$, and an existing structuring element (kernel) B with an origin o . There are four main morphological operations listed below, and how each operation works visually is provided in Figure 3.:

- **Erosion:** A destructive morphological operation $E(A, B)$. Removes all pixels where the white pixels do not fully intersect the structuring element. See Equation 1.
- **Dilation:** An expanding morphological operation $D(A, B)$. Adds white pixels at all locations the origin of the structuring element intersects the white pixels in the image. See Equation 2.
- **Opening:** An opening consists of a number of erosions $E(A, B)$ followed by the same number of dilations $D(A, B)$. Intuitively, it is used for creating small openings in the input image.
- **Closing:** A closing consists of a number of dilations $D(A, B)$ followed by the same number of erosions $E(A, B)$. Intuitively, it is used for closing small gap artifacts in the input image.

$$E(A, B) = A \Theta B = \{o \mid (\widehat{B}_o) \subseteq A\} \tag{1}$$

$$D(A, B) = A \oplus B = \{o \mid (\widehat{B}_o) \cap A \neq \emptyset\} \tag{2}$$

where

A = Input image of some width $\mathcal{W}$ and height $\mathcal{H}$
B = Structuring element
$\widehat{B}$ = reflected B - $\{w|w = -b, \forall b \in B\}$
B_o = translation of B by o - $\{c|c = b + o\forall b \in B\}$

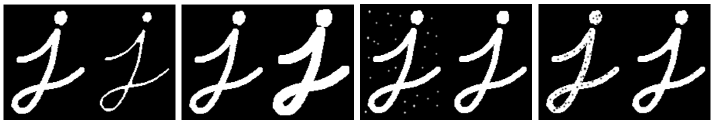

(a) Erosion (b) Dilation (c) Opening (d) Closing

Figure 3: An overview of the primary morphological operations. For all operations, the base image A is on the left, and the output of the operation is on the right. The structuring element B is a 5×5 rectangle. Examples sourced from [18].

These morphological techniques are used for the automatic creation of the images in the CSCD dataset, proposed and used in this study.

2.5 Remote Sensing Considerations

This study examines change detection in the context of remote sensing. All data used is either from remote sensing datasets, or is trying to emulate such. This data differs from standard images, and must be accordingly treated as such. The following should be taken into account:

- All images used are in the orthogonal perspective.
- Spatial resolution refers to the amount of area interpolated into a single pixel from the satellite image. For example, a spatial resolution of 0.1m means that there are 0.1m^2 of information per pixel.

2.6 Cognitive Inspirations

The experiment motivation is the intuitive importance of the fusion's location. It has a neuroscience basis, similar to the one of the original convolutional neural networks. The biological inspirations of CNNs stem from the hierarchical model of the visual system [11]. It talks about how the visual cortices are structured in a sequence, such that initial layers are easily excited and extract simple features (points, lines), while deeper layers inhibit these stimuli and extract more complex features. This is also referred to as having an increasing receptive field [12]. An illustration of this hierarchical structure of the cortices from V1 to IT, along with the corresponding change in the receptive field's size per layer is given in Figure 4.

Relating it back to the experiment, changing the position of fusion changes the size of the receptive field at which the two images stop being compared independently and become entangled/correlated (either through concatenation or skip connections). This ranges from no independent feature extraction (early fusion) to an intermediate (middle fusion) and fully independent (late fusion).

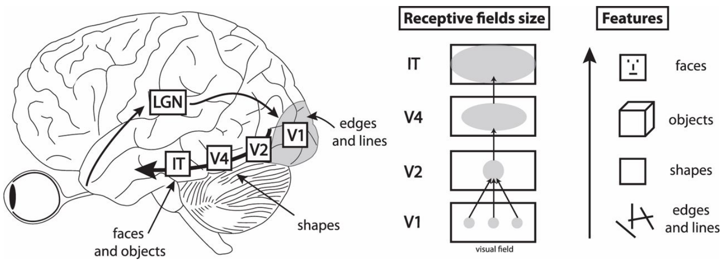


Figure 4: An overview of the hierarchical model of the brain (left) and how the receptive field expands with regards to the features detected (right). Figure borrowed from [12].

3 Methodology

This section presents an explanation and justification of the methodology of the study. It describes the experiment setup, datasets used, and how they were evaluated.

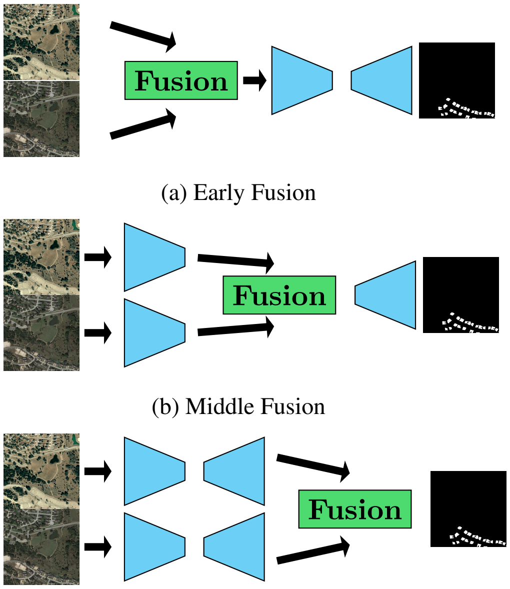
3.1 Experiment

The experiment consists of varying the location of different fusion modules along multiple architectures, and evaluating each model on the same datasets.

Fusion in change detection is something different studies define inconsistently, seen by definitions in [9] and [10]. To ensure a reproducible and transparent experiment, this study relies on the definition proposed in [9]. It consists of three different parts:

- **Early Fusion**: Any image correlation prior to input in the change detection network.
- **Middle Fusion**: Any entanglement of the inputs inside of the neural network. This could be some concatenation of inputs inside of different layers, prior to the final layer, could be residual connections within the network.
- **Late Fusion**: Any form of image correlation done as a post-processing step. This could be a simple pixel difference from two object detection networks or a secondary classification/segmentation network.

How the three different fusion points of fusion look, abstracted from the encoder-decoder implementations of the networks is shown in Figure 5.



(a) Early Fusion

(b) Middle Fusion

(c) Late Fusion

Figure 5: Diagram of an arbitrary Siamese network with different fusion strategies. The blue boxes are encoder-decoder networks. The point of fusion may or may not be a part of them, depending on implementation.

The experiments have been standardized on three Siamese fully convolutional architectures proposed and tested in [6]: one using early fusion and two different takes on middle fusion. All of them follow a Siamese encoder-decoder structure, using U-Net as its backbone, and have been standardized to receive two input images

$\mathcal{T}_1, \mathcal{T}_2$, outputting a two-channel image $\mathcal{Y}$, where its channel difference is the binary change mask desired. To compare all fusion types, this study adds a fourth configuration, following the same backbone, but using a late fusion. All architectures are described as follows, and illustration is provided in Figure 6.

1. **FC-EF:** *Early Fusion.* $\mathcal{X}_1, \mathcal{X}_2$ are fused by concatenating their channels together along the same axis, prior to any processing by the network. See Figure 6 a).
2. **FC-Siam-Conc:** *Middle Fusion.* There are separate encoder streams for both $\mathcal{X}_1, \mathcal{X}_2$. The fusion is done by combining the U-Net skip connections at all layers of the architecture, essentially letting the network do a concatenation over the feature maps' channels at any intermediate step. See Figure 6 b).
3. **FC-Siam-Diff:** *Middle Fusion.* Similar to FC-Siam-Conc, this configuration once again concatenates the results from the input streams. However, the value concatenated is the absolute value of the image difference. See Figure 6 c).
4. **FC-LF:** *Late Fusion.* Both images are processed and segmented separately by a U-Net encoder-decoder. These segmentations are then concatenated together and input into FC-EF. See Figure 6 d).

3.1.1 Training

All models have been trained on the same datasets, using the following setup. The loss function is PyTorch's negative log likelihood (NLL) - see Equation 3, that assumes a logarithm softmax final layer [19]. The optimization method used is Adam along with an exponential learning rate scheduler with $\gamma = 0.95$. To calculate the class weights, it is desirable that the black pixels (no change) have a lower priority. The reason is that white pixels (change labels) are sparser. As such, a balancing term `FP_Modifier` is used for both classes, based on the amount of white and black pixels in the labels of the dataset.

$$l(\widehat{y}, y) = \sum_{n=1}^{N} \frac{1}{\sum_{n=1}^{N} w_{y_n}} \cdot l_n \tag{3}$$

where $\widehat{y}$ = predicted image
y = ground truth image
$l_n = -w_{y_n} \cdot \widehat{y}_{n,y_n}$ = pixel prediction, weighed by class
w_c := weight of class $c \in \{0, 1\}$
$w_0 = \frac{2 \times \texttt{FP\_MODIFIER} \times (\texttt{total\_pixels} - \texttt{1\_pixels})}{\texttt{total\_pixels}}$,
$w_1 = \frac{2 \times \texttt{1\_pixels}}{\texttt{total\_pixels}}$
N = number of samples in the dataset
`FP_Modifier` - false positve modifer - parameter to adjust black pixel weight
`total_pixels` - sum of pixels in dataset
`1_pixels` - sum of pixels with value 1 in dataset

For LEVIR-CD and HiUCD, an `FP_Modifier` $= 1$ was used, while for CSCD, it was set to 10. Batch sizes for LEVIR-CD and HiUCD were 4, while for CSCD it was 64, due to GPU memory constraints.

All of the models configurations are trained until the validation loss was keeping pace with the training loss to avoid overfitting. This is achieved through the implementation of an early stopping criterion. After the 10th epoch, there was a patience criterion of 5 epochs, necessitating an increase in the best validation loss. The best validation set parameters are restored after training finishes. All architectures and dataset combinations were trained for a maximum of 100 epochs.

This setup is almost identical to the one in [6], with the differences that images are not dynamically cut up into patches to create extra data.

All training has been performed on Google Colab using the Tesla T4 GPU configuration.

3.2 Datasets

The study uses three datasets, two of which come from remote sensing areas, the third one is synthetically generated. Two of them, HiUCD and CSCD, are categorical to test the the hypothesis of the impact of fusion on the size and spread of changes detected.

3.2.1 LEVIR-CD Dataset

The dataset consists of 637 Very-High-Resolution (VHR) 0.5m images of 1024×1024 pixels each. Their time differences vary between 5 and 14 years. Introduced in [8], it has since been used as a benchmark dataset for change detection due to its quality labels and large size. The changes in it are predominantly urban: apartment buildings, garages, etc.

3.2.2 HiUCD (Mini) dataset

A VHR (0.1 m) dataset of Tallinn contains including both binary masks and semantic, categorical maps [13]. There are 745 samples total. The type of data annotated are of refined urban changes. The categorical labels are on a pixel-level, and can be seen in Table 1. It should be mentioned that HiUCD as a dataset is still being refined, and what was used in this study was a smaller version, provided upon request.

ID	Description
0	Water
1	Grass
2	Building
3	Greenhouse
4	Road
5	Bridge
6	Bare land
7	Woodland
8	Others
9	Unlabeled

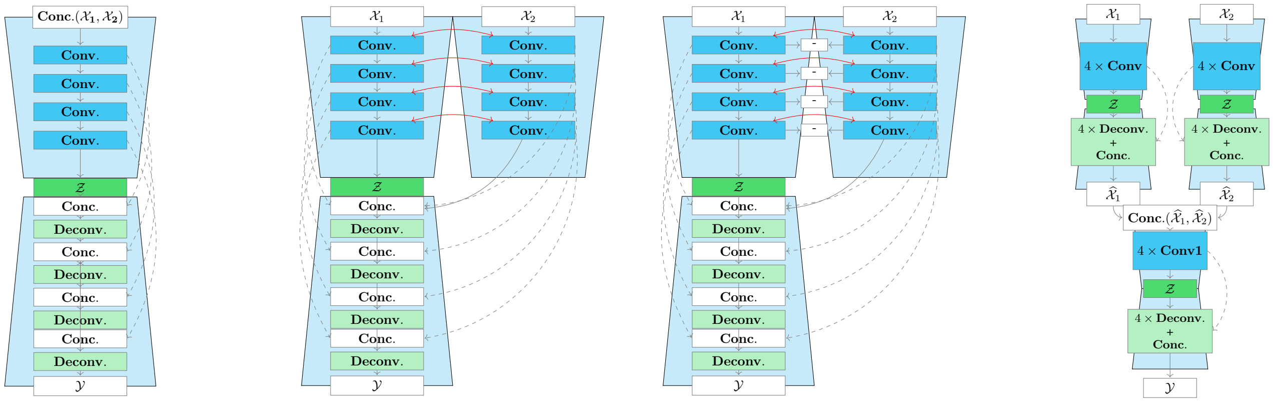
Table 1: HiUCD Dataset Classes. The ID column represents what pixel value the category is stored under.

3.2.3 CSCD - Custom Semantic Change Detection

CSCD is meant to facilitate training and change analysis with regards to the hypothesis on the impact of fusion on changes of different sizes and spreads. It has been created specifically for the study through an automatic, morphological procedure. It has the advantage of its samples belonging to categories exactly per their definition.

The dataset used in the fusion architectures' training consists of 6144 pairs of 128×128 images, combined with a categorical and binary mask label. The images are two-colored, but with an arbitrary RGB color for both the foreground and background in every pair. The aim of this data augmentation is to make the dataset robust, so the model does not learn a plain XOR function during training.

There are four cases (categories) considered in generation of the CSCD:

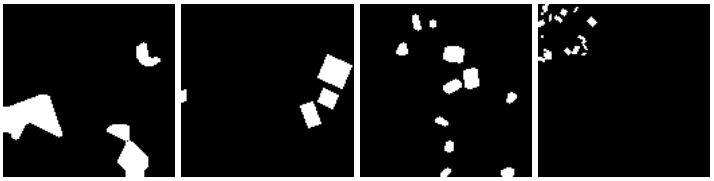


(a) (b) (c) (d)

Figure 6: A showcase of all four of the architectures compared: (a) **FC-EF**; (b) **FC-Siam-Conc**; (c) **FC-Siam-Diff**; (d) **FC-LF**. Pooling layers are omitted. Figure adapted from [6].

- **LCU**: *Large Changes - Uniform*
- **SCU**: *Small Changes - Uniform*
- **LCNU**: *Large Changes - Nonuniform*
- **SCNU**: *Small Changes - Nonuniform*

Additional metadata is added related to number of buildings per image, used to draw better conclusions on how each fusion performs. The differences in the spread of the buildings and their sizes is provided in Figure 7.



(a) (b) (c) (d)

Figure 7: How the change labels in CSCD look. (a) - **LCU**: *Large Changes - Uniform*; (b) - **LCNU**: *Large Changes - Nonuniform*; (c) **SCU**: *Small Changes - Uniform*; (d) - **SCNU**: *Small Changes - Nonuniform*

3.2.4 CSCD Generation Procedure

The dataset generation is based on morphology, comprising of the following four steps:

1. ***Creation of base image - $\mathcal{T}_1$.*** A black image with custom width and height $\mathcal{W} \times H$ is generated. Then $N \in [0, 12]$ rectangles are spread uniformly within $\mathcal{T}_1$. The rectangles have a minimum width and height of $\frac{1}{20}$ of $\mathcal{W}$ and $\mathcal{H}$, and a maximum of $\frac{1}{2}$.
2. ***Creation of binary changes on base image - $\mathcal{T}_2$.*** Depending on the category specified, N' rectangles are once again spread around with constraints with regards to size and distribution.
   - Large changes are $N' \in [0, 5]$, while small ones are $N'' \in [5, 25]$.
   - Uniform changes are spread at random image coordinates, while non-uniform changes have the image divided in a 3×3 grid and spread randomly within $k \in [1, 3]$ of those grids.

   To mimic changes of buildings growing in size and being removed, morphological operations are applied to each image, varying per category as follows:

   - **LCU**, **LCNU**: Opening → Closing → Erosion using a $B = 7 \times 7$ ellipse-shaped structuring element.
   - **SCU**, **SCNU**: Opening with a $B = 5 \times 5$ ellipse-shaped structuring element
3. ***Creation of change label $\mathcal{Y}$ between $\mathcal{T}_1$ and $\mathcal{T}_2$.*** An XOR is performed between the two images to obtain a binary mask. Due to the random spread of the rectangles, shapes end up unnatural. To make the changes more similar to remote sensing datasets, an opening operation is performed with a $B \in (5, 5)$ ellipse structuring element.
4. ***Post-Processing.*** The $\mathcal{T}_1, \mathcal{T}_2$ images get a random RGB color assigned for foreground and background. The number of changes within $\mathcal{Y}$ is calculated by contour counting for metadata. Gaussian blur is applied to $\mathcal{T}_1$ and $\mathcal{T}_2$.

All of the numbers listed in the procedure are currently arbitrary, and are meant to bring the data closer to remote sensing images.

3.3 Evaluation

All fusion architectures are evaluated in terms of accuracy. For LEVIR-CD, HiUCD and CSCD, the change detection predictions have been evaluated in overall accuracy, while HiUCD and CSCD have additionally been evaluated categorically. The predictions get labeled as a True Positive (TP), False Positive (FP), True Negative (TN) and False Negative (FN), based on their intersection over union (IoU) with the ground truth labels. For all the mentioned formulas, see Equation 4. The IoU threshold chosen to classify a prediction as a TP, FP, TN or FN is a standard 0.5.

On all training data, this allows the measurement of the metrics precision, recall, F-1, and accuracy.

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}} \qquad \text{Recall} = \frac{\text{TP}}{\text{TP+FN}}$$

$$\text{F}_1 = \frac{2\cdot\text{Precision}\cdot\text{Recall}}{\text{Precision+Recall}} \qquad \text{Accuracy} = \frac{\text{TP+TN}}{\text{TP+TN+FP+FN}}$$

$$\text{IoU} = \frac{\text{TP}}{\text{TP+FP+FN}} \tag{4}$$

An additional metric used is the ground truth number of changes $|\mathcal{Y}|$ vs. the number of changes predicted $|\widehat{\mathcal{Y}}|$, as well as their mean difference μ_{diff} for N samples - see Equation 5.

$$\mu_{\text{diff}} = \frac{1}{N}\sum_{i=1}^{N}\left||\mathcal{Y}_i| - |\widehat{\mathcal{Y}}_i|\right| \tag{5}$$

The way the predicted number is estimated is via counting the connected components of the output of OpenCV's contour algorithm [20], [21]. The procedure was selected due to it being an automatic way to measure the number of changes in the labels. It is, however, imperfect and produces outliers, such as in situations where single pixels were predicted. To maintain an accurate reasoning on the number of changes, outliers have been floored to 0 via Median Average Deviation (MAD) from all predictions for the given fusion architecture.

4 Results and Discussion

This section presents the qualitative and quantitative differences between the different fusion architectures. All evaluation metrics can be seen in Figure 8.

Complex datasets like LEVIR-CD show variability in performance between fusion architectures. As seen in Table 2, in terms of accuracy FC-Siam-Conc. (82.03) and FC-LF (81.24) highly outperform the early fusion and the other middle fusion architecture. While FC-EF is comparable (70.31), FC-Siam-Diff's accuracy is twice as low (39.06). Analyzing the difference between the number of predicted changes and the ground truth ones, shown in Figure 8d, all architectures tend to overestimate the amount in a given image, also evident in their lower precision metrics. The best performers are FC-LF with $\mu_{\text{diff}} = 53.6$ and FC-Siam-Conc with $\mu_{\text{diff}} = 80.21$. This contradicts how one outperforms the other in accuracy, with a possible explanation being FC-Siam-Conc segmenting its output into more connected components than FC-LF for the same ground truth object - this may still achieve a higher IoU, while being inaccurate.

Both in the categorical and performance evaluations, HiUCD's results are inconclusive due to the complexity of the data. All architectures struggle with a dominant amount of FPs, as is visible by the low precision in Table 3. This aligns with results, obtained on the original HiUCD paper with the early and middle fusions [13], where they claimed an average IoU of below 40%, where the criterion for a true positive in this study is 50%. This also impacts the categorical evaluation in Figure 8h, where all histograms are very comparable to each other. There are small differences when evaluating smaller object like buildings or grasslands, but nothing conclusive. Due to the discrepancy between performance on HiUCD and the remaining different datasets, it can only be concluded, that adjustment to the general model architecture and optimization is needed, both for overall and categorical performance.

The results on CSCD show differences in fusion modules for performance, but are inconclusive on a categorical basis. Accuracy-wise on the test sets in Table 4, FC-EF (98.97) and FC-LF (97.31) outperform both FC-Siam-Conc (91.79) and FC-Siam-Diff (97.31). No fusion module showed a predisposition towards any CSCD category that did not coincide with their overall performance - see Figure 8g. A trend was observed when examining the number of buildings per fusion architecture - see Figure 8f. FC-EF, while achieving the best IoU classification results, was also the one to overestimate the total amount of objects the most. The middle architectures fusion, while less accurate, did best on estimating the number of objects. FL-LF achieved a balance between its accuracy and estimating the number of objects. These results should be taken with the simplicity of CSCD into account. Despite the data augmentations from the Gaussian blurring, re-coloring and the post-processing morphology, early and late fusion achieved near-perfect results, suggesting it is still not robust enough. Especially in the case of FC-LF, the pre-segmentation via U-Net may be making late fusion particularly suitable to a simple change detection scenario, denoising the image beforehand.

Results differed per dataset in terms of training stability and training time. It cannot be said which fusion converges in the least epochs. For LEVIR-CD, as seen in Figure 8a late fusion converges the fastest, followed by FC-Siam-Diff and with the remaining architectures similarly converging around 60 epochs. For HiUCD's training - see Figure 8b - all architectures converged early. On CSCD - Figure 8c, FC-EF and FC-LF converge the fastest, with the latter having periodic negative spikes in performance. Both middle fusion architectures converged with more epochs and were unstable as doing so. FC-Siam-Conc was even trained up to the 100th epoch, suggesting more training is possible for it. As far as training stability, LEVIR-CD and CSCD had a smooth training process compared to HiUCD, backing up the data complexity claims. The actual loss values achieved are not comparable due to the pixel class imbalances in each dataset, and their different sizes.

Analyzing the results, it can be stated that which fusion performs best depends on the dataset's complexity. In terms of general performance, either FC-EF, FC-Siam-Conc and FC-LF could perform best, as seen by the discrepancies between LEVIR-CD and CSCD. For change counting purposes, a middle or late fusion method seems to stick closest to the ground truth data, with the least overestimation. For general recognition, having pixel subtraction as a differentiation metric seems to confuse the network, achieving lower performance, as seen by the less stable and lower performance of FC-Siam-Diff.

5 Responsible Research

In addition to the practical results obtained, concerns are raised in relation to reproducibility of the study and its ethical implications.

To ensure authenticity and verifiability, all details about the experiment are public, both in the paper and in an online GitHub repository[1]. Replication studies are recommended on different datasets (due to the specificity nature of satellite images), and with different training configurations, to ensure both the validity and correctness of the chosen methods.

Some of the remote sensing data used for categorical evaluation is not in the public domain (HiUCD), and has to be requested from its

[1] https://github.com/vdakov/encoder-decoder-change-detection

Architecture	Acc.	Prec.	Rec.	F-1	μ_{diff}
FC-EF	70.31	0.70	0.96	0.81	97.55
FC-Siam-Conc	82.03	0.83	0.96	0.89	80.21
FC-Siam-Diff	39.06	0.36	0.87	0.51	276.5
FC-LF	81.24	0.81	0.99	0.89	53.6

Table 2: Evaluation on LEVIR-CD per architecture.

Architecture	Acc.	Prec.	Rec.	F-1	μ_{diff}
FC-EF	4.92	0.04	1	0.09	726
FC-Siam-Conc	7.51	0.07	1	0.13	776.9
FC-Siam-Diff	15.54	0.15	0.96	0.26	1336.6
FC-LF	17.09	0.17	0.85	0.29	975.5

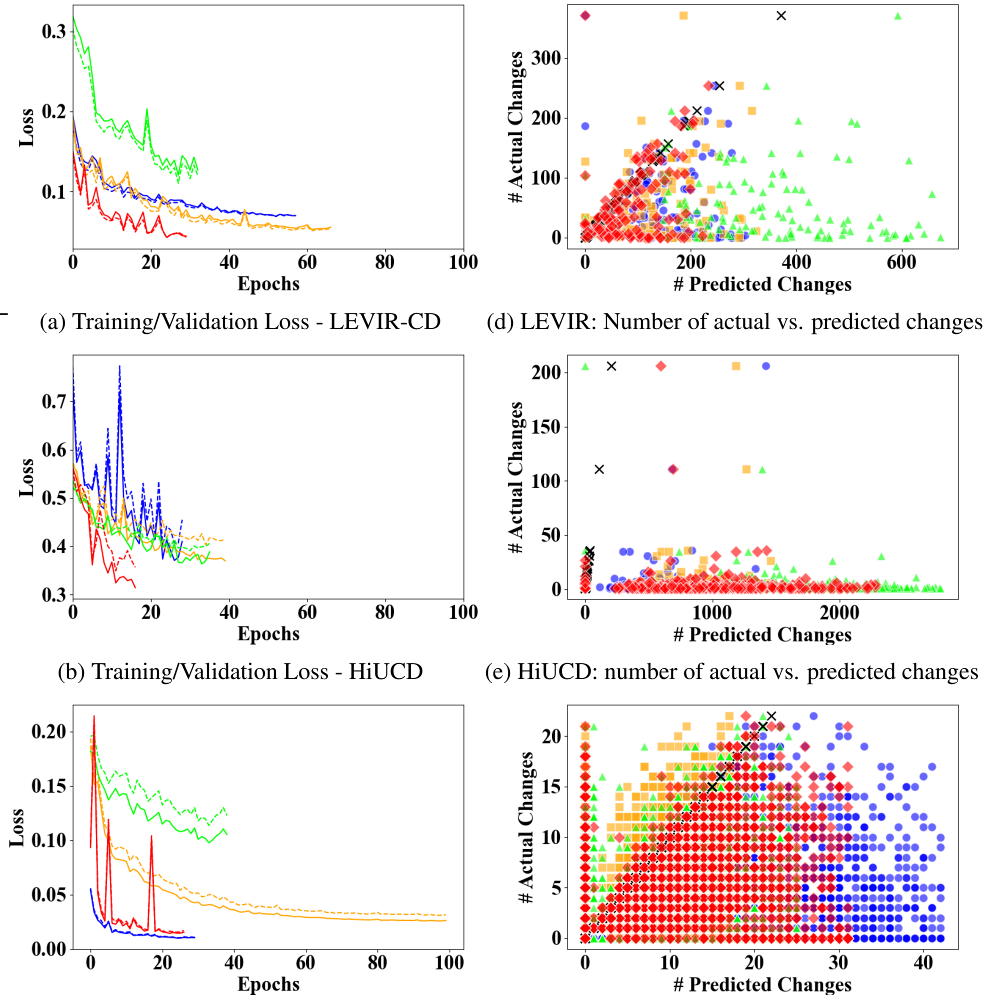
Table 3: Evaluation on HiUCD per architecture.

Architecture	Acc.	Prec.	Rec.	F-1	μ_{diff}
FC-EF	98.97	0.98	1	0.99	10.08
FC-Siam-Conc	91.79	0.91	0.99	0.95	3.7
FC-Siam-Diff	67.91	0.76	0.80	0.78	5.34
FC-LF	97.31	0.97	0.99	0.98	6.71

Table 4: Evaluation on CSCD per architecture.

Legend

FC-EF	FC-EF-Val.
FC-Siam-Conc.	FC-Siam-Conc.-Val.
FC-Siam-Diff.	FC-Siam-Diff.-Val.
FC-LF	FC-LF-Val.
Ground Truth	



(a) Training/Validation Loss - LEVIR-CD

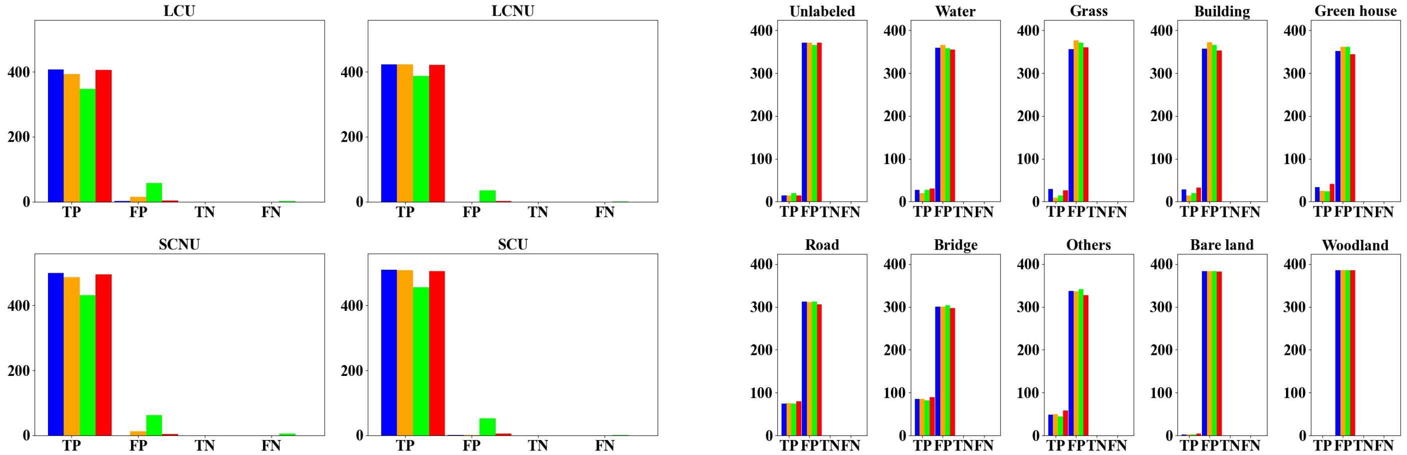
(d) LEVIR: Number of actual vs. predicted changes

(b) Training/Validation Loss - HiUCD

(e) HiUCD: number of actual vs. predicted changes

(c) Training/Validation Loss - CSCD

(f) CSCD: number of actual vs. predicted changes



(g) CSCD: Categorical Evaluation. From top left to bottom right - **LCU**, **LCNU**, **SCNU**, **SCU**.

(h) HiUCD: Categorical Evaluation. From top left to bottom right - **Unlabeled**, **Water**, **Grass**, **Building**, **Green house**, **Road**, **Bridge**, **Others**, **Bare land**, **Woodland**.

Figure 8: Aggregated results from all three datasets: CSCD, HiUCD, LEVIR-CD. On the top left are tables with results on the test sets after training. Abbreviations are for accuracy, precision, recall, F-1 and μ_{diff} is the mean absolute difference between the number of buildings predicted and ground truth. In the middle are the training and validation losses (using the early stopping criterion). Right side showcases each dataset's predicted number of buildings vs. the ground truth. The bottom of the figures shows accuracy histograms per category. A legend with consistent labels per plot is on the left.

corresponding authors. Concerning CSCD, the procedure is transparent and descriptive, and the data used for training is also published.

Change detection analyses are usually performed for predicting urban trends, or making economic decisions. That is why making wrong conclusions or letting inaccuracies into the studies can cause financial consequences on economic development of the cities and communities. It is why this study outlines all known issues with the results presented, and is conservative in its conclusions.

To the best knowledge of the authors of the study, all work presented here is as described.

6 Conclusions and Future Work

Changing the fusion configuration for change detection impacts results from a performance standpoint, but the only category conclusively impacted is the number of changes detected. Moreover, different dataset complexities impact which fusion is best, and can vary.

The study can be expanded upon with replication studies on different datasets, more architectures and different loss functions.

All architectures tested have the same U-Net backbone. This does not fully reflect the current state of change detection. Recent methods are based on transformers, due to the images and tasks' temporal nature. It is worth seeing if extending them to different types of fusion keeps the trends identified with U-Net. RNN variants, or even unsupervised approaches with autoencoders, should also be examined.

Negative log likelihood is imperfect as a loss function, due to the class imbalance between black and white pixels. The current method of introducing a balancing term leads to changes in results between hyperparameter values and datasets. More modern object detection losses like focal loss [22] are made to deal with sparser objects, like the ones in change detection. It was not tested in the study due to time constraints, and discovery late into experiments.

The categorical evaluation on real remote sensing datasets is limited. High-quality remote sensing data for change detection is hard to collect. High-quality and quantity datasets like LEVIR-CD are few and far between. This is even more the case for categorical data. HiUCD as a dataset is still being created and was provided for this study upon request. Its ground truth images are harder to interpret by the network, due to the challenges of labeling. Outlining which object has changed by hand is difficult, and labeling a site as water, green area or similar is even harder, so it is prone to errors. Even if there were more categorical data, remote sensing images come from inherently different locations (e.g. one is more urban, while the other rural). If areas with similar characteristics could be identified, and have their ground truth categories standardized, this would give more credit to any future replication studies. Consequently, transfer learning is an area, where future studies can expand on to ensure model robustness.

Intersection over union and the parameter chosen for it could be refined. The reason a high threshold of 0.5 was chosen was to ensure an acceptable accuracy. However, not all change detection datasets are large enough to allow such accuracy, as seen in HiUCD. Additionally, the discrepancy between early fusion's overestimation and its high accuracy mean this evaluation metric could be expanded.

The custom dataset, CSCD, is limited in both its generation procedure and semantics. This procedure must be better formalized, since the method outlined is partially based on intuition to bring the data close to other remote sensing datasets. It needs to be specified what all objects of interests are geometrically, how this may change over time, and not limit the definition to rectangles and modifying sizes with morphology. One possible direction is utilizing data from cadastre maps, bringing the image closer to real remote sensing objects. Next, the current implementation maps the base images to random foreground and background colours. While this, in combination with the data augmentation, does prevent the models from learning a simple XOR function, it also may make them more biased towards certain colors per dataset generation due to the randomness of the process. Extensions, such as dynamic brightness adjustments, a more equalized color distribution, and possibly more noising operations should be applied.

These problems come from problem isolation - a constraint to a specific deep learning architecture, a dataset, a loss function. To better understand how fusion works, a standardized procedure needs to be created, which is abstracted from the neural network specifics, and be on an equalized dataset. If one were to solve this, change detection architectures would be better defined moving forward, and the decision of where to employ what system could be better defended.

References

[1] I. S. Serasinghe Pathiranage, L. N. Kantakumar, and S. Sundaramoorthy, "Remote Sensing Data and SLEUTH Urban Growth Model: As Decision Support Tools for Urban Planning," *Chinese Geographical Science*, vol. 28, pp. 274–286, Apr. 2018.

[2] F. Dahle, K. Arroyo Ohori, G. Agugiaro, and S. Briels, "Automatic change detection of digital maps using aerial images and point clouds," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XLIII-B2-2021, pp. 457–464, 2021.

[3] S. Xu, G. Vosselman, and S. Oude Elberink, "Detection and classification of changes in buildings from airborne laser scanning data," *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. II-5/W2, pp. 343–348, 2013.

[4] E. Shelhamer, J. Long, and T. Darrell, "Fully Convolutional Networks for Semantic Segmentation," May 2016. arXiv:1605.06211 [cs] version: 1.

[5] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," May 2015. arXiv:1505.04597 [cs].

[6] R. Caye Daudt, B. Le Saux, and A. Boulch, "Fully Convolutional Siamese Networks for Change Detection," in *2018 25th IEEE International Conference on Image Processing (ICIP)*, pp. 4063–4067, Oct. 2018. ISSN: 2381-8549.

[7] "Change Detection in Multisource VHR Images via Deep Siamese Convolutional Multiple-Layers Recurrent Neural Network | IEEE Journals & Magazine | IEEE Xplore."

[8] H. Chen and Z. Shi, "A Spatial-Temporal Attention-Based Method and a New Dataset for Remote Sensing Image Change Detection," *Remote Sensing*, vol. 12, p. 1662, Jan. 2020. Number: 10 Publisher: Multidisciplinary Digital Publishing Institute.

[9] G. Cheng, Y. Huang, X. Li, S. Lyu, Z. Xu, Q. Zhao, and S. Xiang, "Change Detection Methods for Remote Sensing

in the Last Decade: A Comprehensive Review," May 2023. arXiv:2305.05813 [cs, eess].

[10] W. Shi, M. Zhang, R. Zhang, S. Chen, and Z. Zhan, "Change Detection Based on Artificial Intelligence: State-of-the-Art and Challenges," *Remote Sensing*, vol. 12, p. 1688, Jan. 2020. Number: 10 Publisher: Multidisciplinary Digital Publishing Institute.

[11] D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *The Journal of Physiology*, vol. 160, pp. 106–154.2, Jan. 1962.

[12] M. H. Herzog and A. M. Clarke, "Why vision is not both hierarchical and feedforward," *Frontiers in Computational Neuroscience*, vol. 8, Oct. 2014. Publisher: Frontiers.

[13] S. Tian, A. Ma, Z. Zheng, and Y. Zhong, "Hi-ucd: A large-scale dataset for urban semantic change detection in remote sensing imagery," 2020.

[14] U. Michelucci, "An introduction to autoencoders," 1 2022.

[15] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature Verification using a "Siamese" Time Delay Neural Network," in *Advances in Neural Information Processing Systems*, vol. 6, Morgan-Kaufmann, 1993.

[16] D. Chicco, "Siamese Neural Networks: An Overview," in *Artificial Neural Networks* (H. Cartwright, ed.), pp. 73–94, New York, NY: Springer US, 2021.

[17] P. Soille, *Morphological Image Analysis*. Berlin, Heidelberg: Springer, 2004.

[18] "OpenCV: Morphological Transformations."

[19] "NLLLoss — PyTorch 2.3 documentation."

[20] "OpenCV: Contour Features."

[21] S. Suzuki and K. be, "Topological structural analysis of digitized binary images by border following," *Computer Vision, Graphics, and Image Processing*, vol. 30, pp. 32–46, Apr. 1985.

[22] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," pp. 2999–3007, 2017.