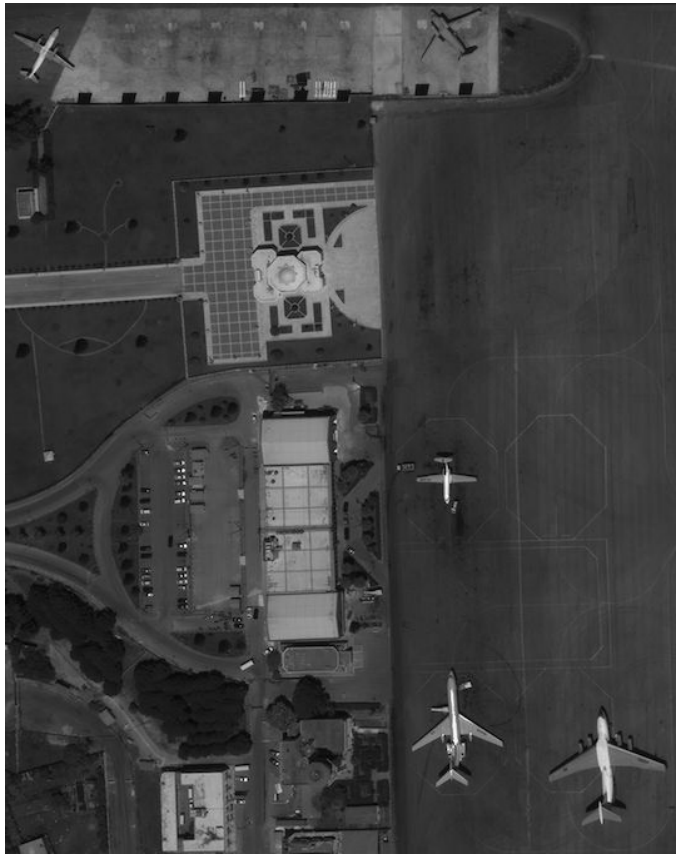
Berkeley
UNIVERSITY OF CALIFORNIA

Change Detection in Remote Sensing Images

Andrew Kiruluta

Examples of Change Detection Cases

Examples of Change Detection Cases

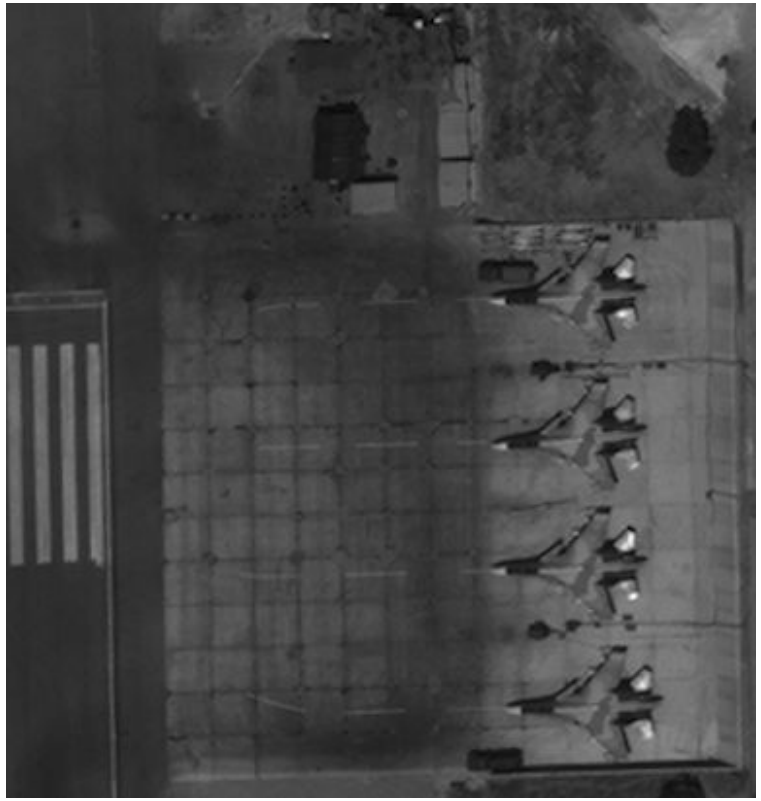


Examples of Change Detection Cases

Examples of Change Detection Cases



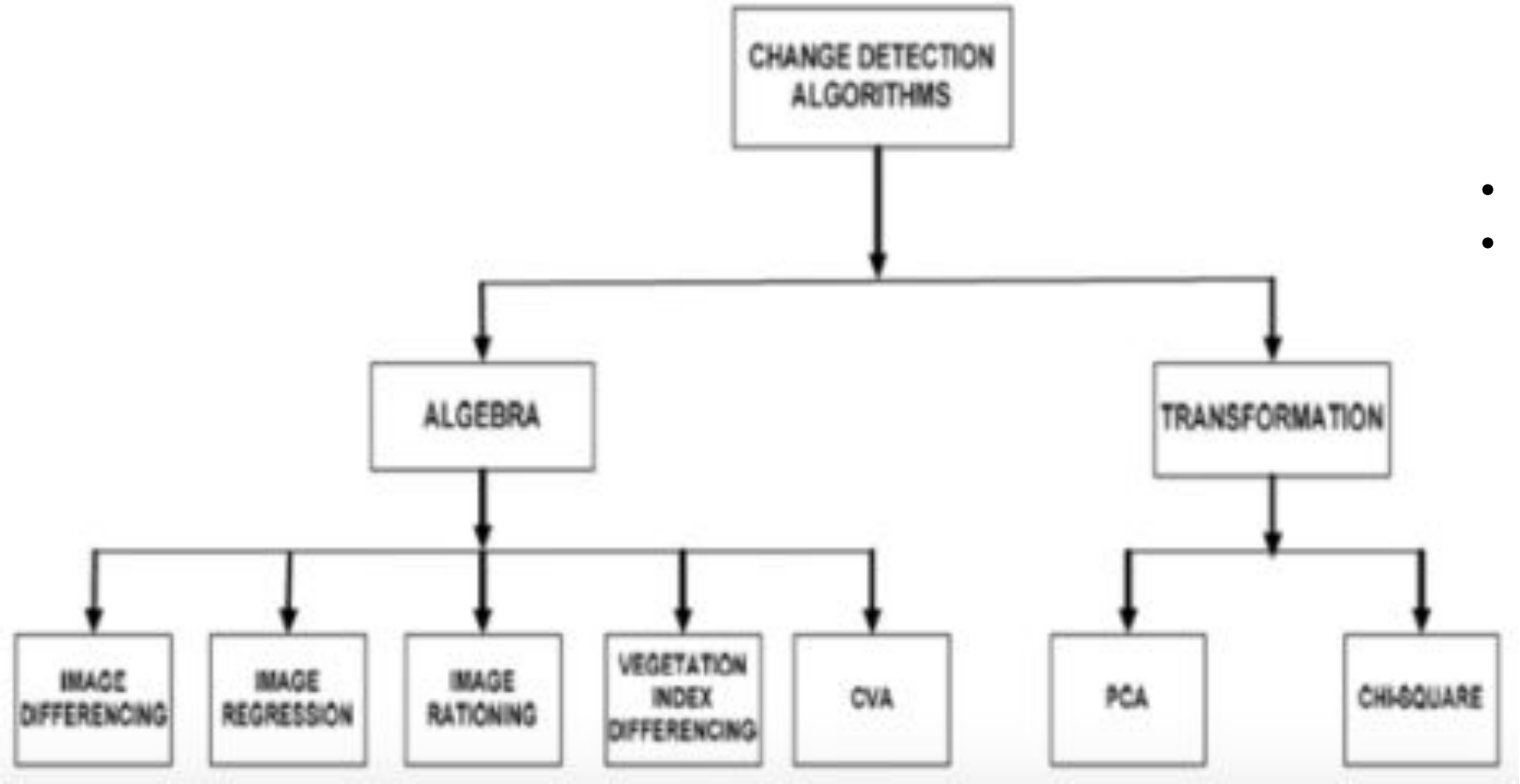
Examples of Change Detection Cases



Examples of Change Detection Cases



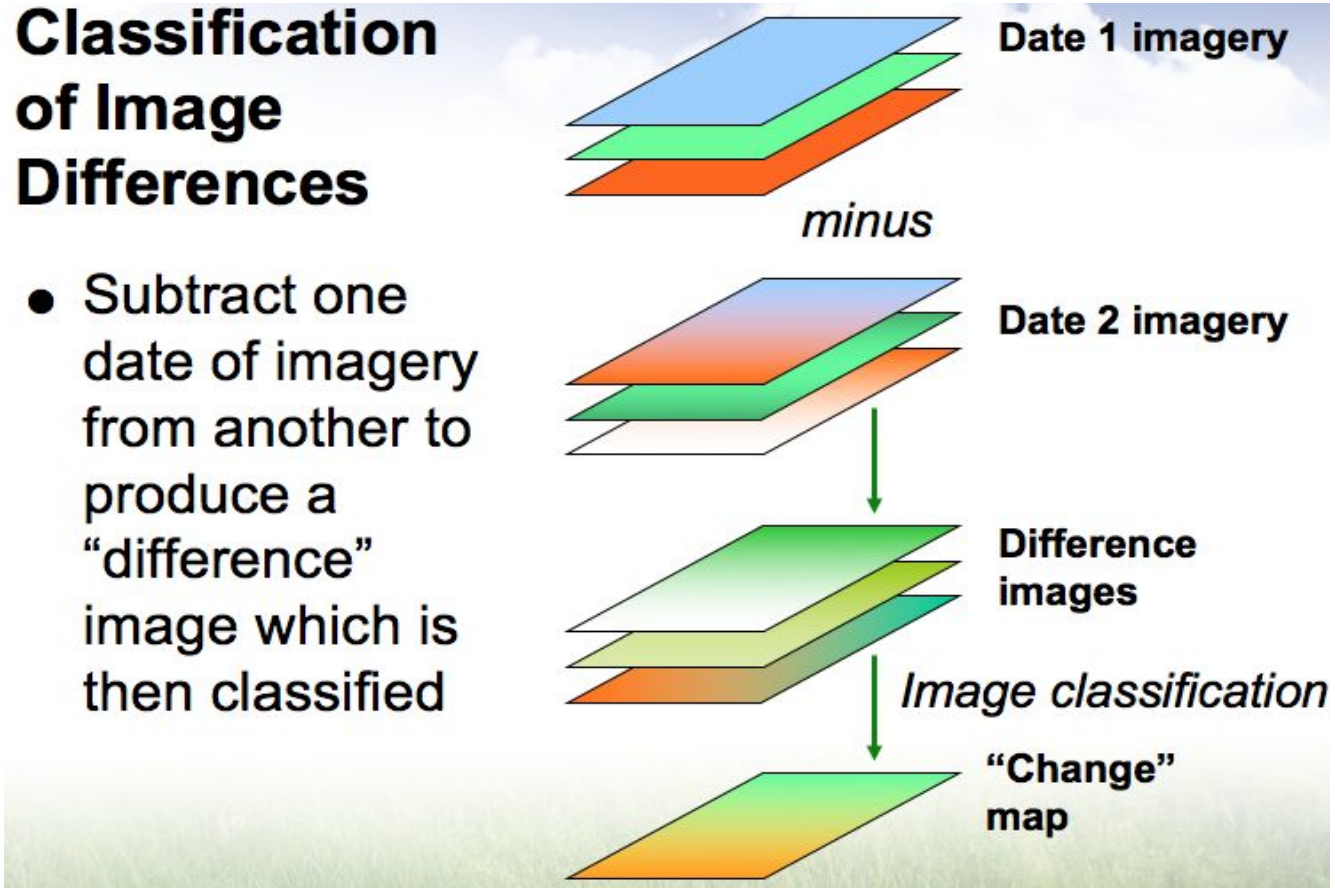
Traditional Algebraic/Transformational Methods of Change Detection



- pixel by pixel methods
- no classification of change

Initial Method of Attack:

- use algebraic or transformation methods



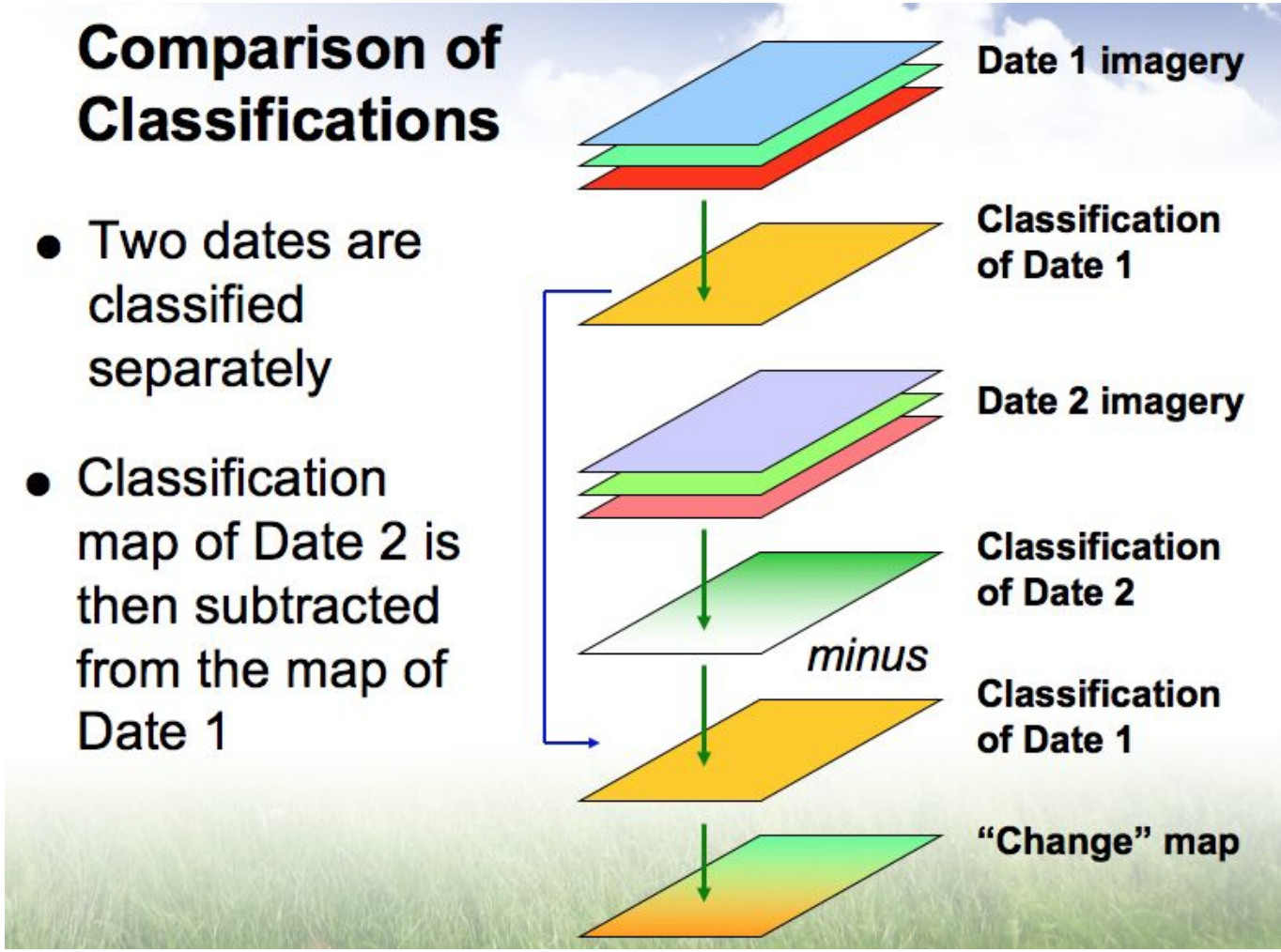
- **Unsupervised Classification**: no ground truth training data is required and it uses spectral properties of the image to classify the image into different classes.

- **Change detection** is done by first finding change using algebraic/transformation change detection algorithm and then performing **unsupervised classification** to separate changed and unchanged areas

Limitations: registration problems when carrying out the image difference operation

Deep Learning Change Detection and Identification

- run **supervised inference** on the change map to classify changed object (plane, building, truck ...) and look at quantities such as:
- - IofU - Jaccard index differences
- - Probability of detection of similar features between the two time sampled images



Weakly Supervised Change Detector Paper
Khan et al., arXiv:1606.02009v2

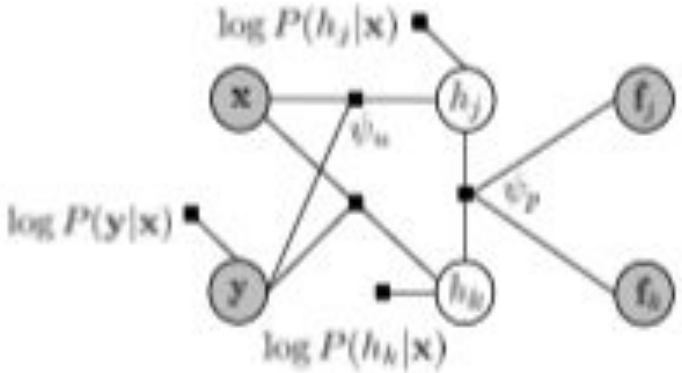


Figure 2: **Factor graph representation** of the weakly supervised change detection model. The shaded and non-shaded nodes represent the observable and hidden variables, respectively. The dense connections between hidden nodes are not shown here for clarity.

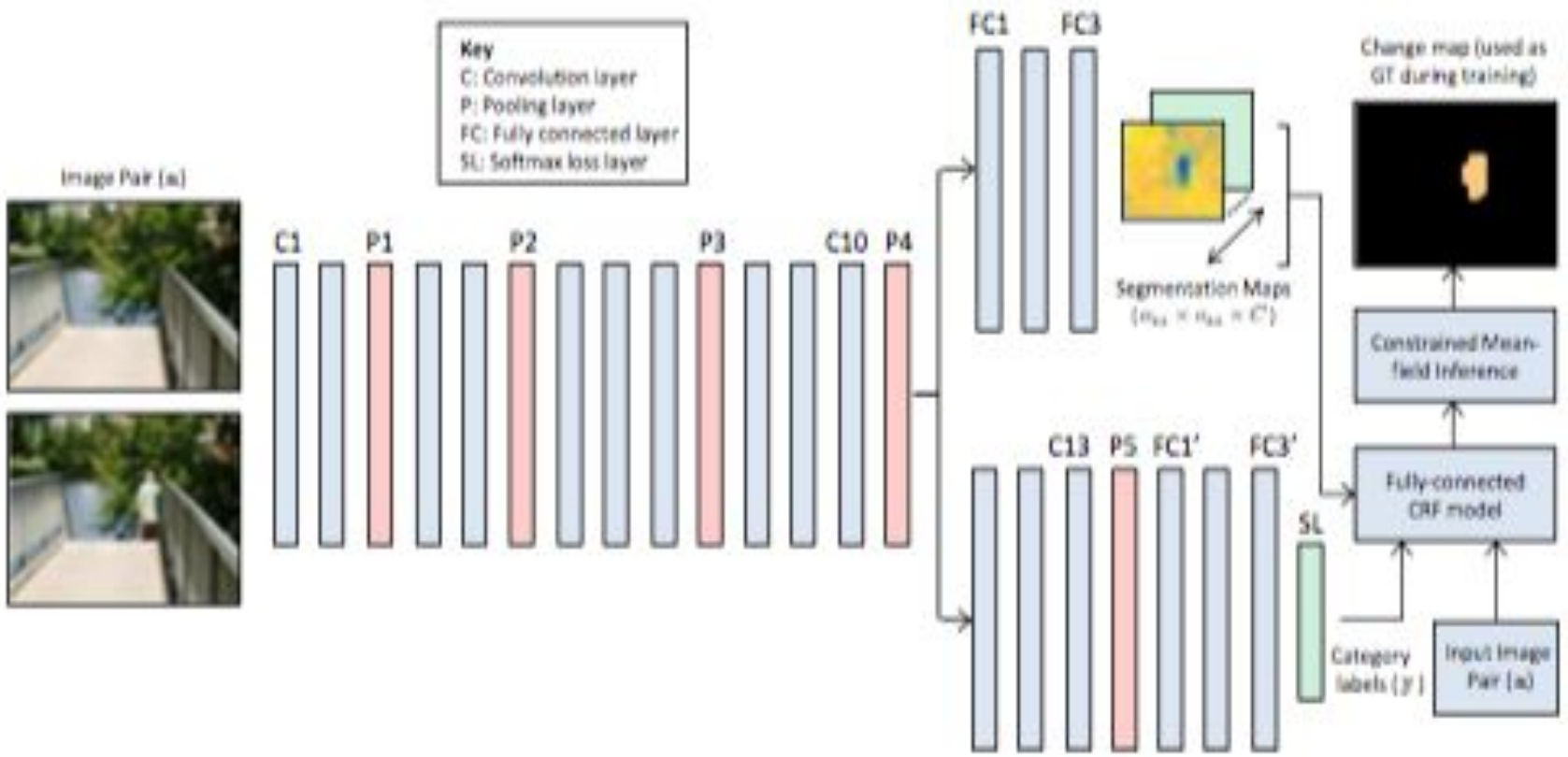
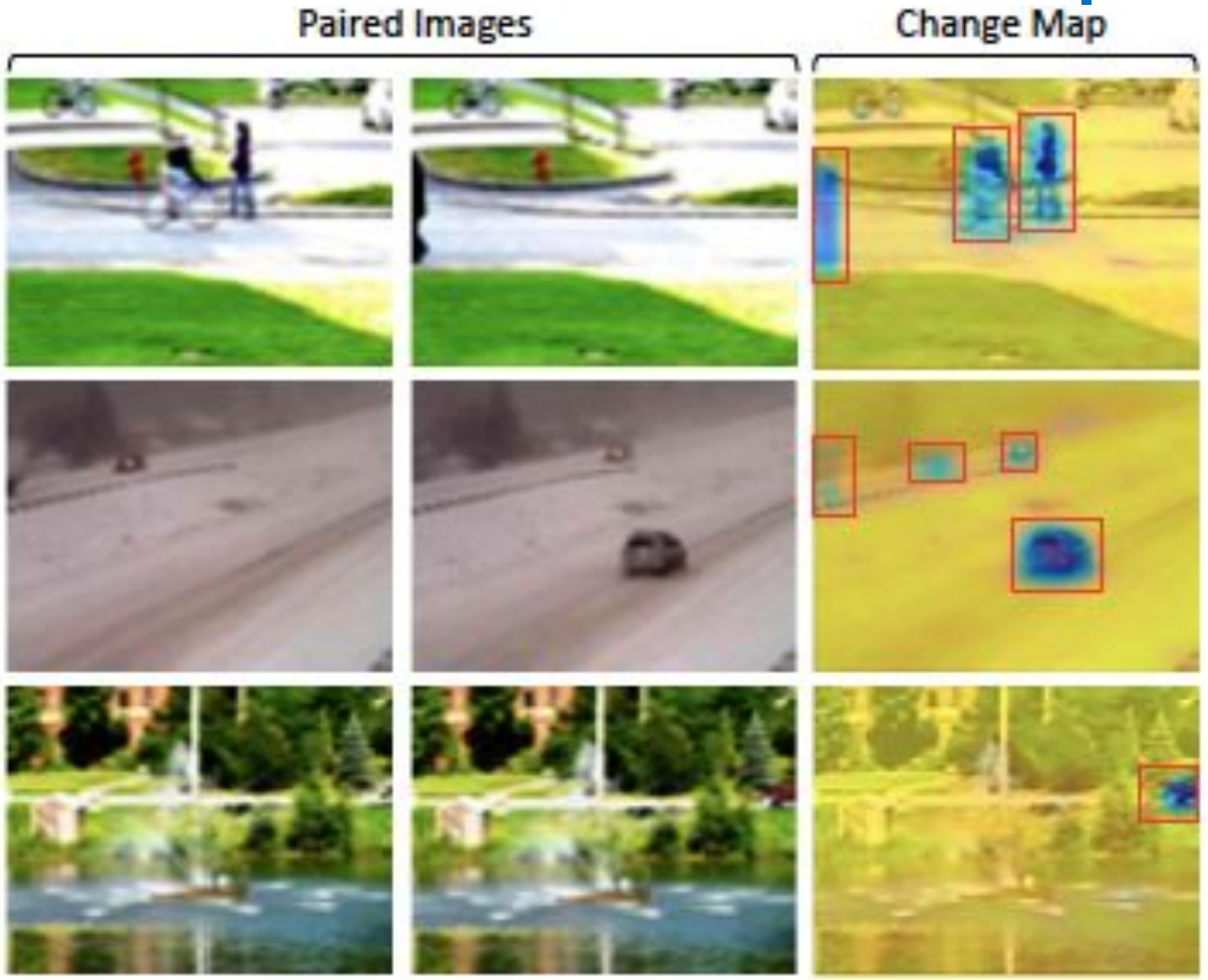


Figure 3: **CNN architecture:** The network operates on paired images and divides in two branches after the fourth pooling layer ($P4$). The classification branch (lower) is trained using the image-level labels (y). The hidden variables h_j are estimated using the constrained mean-field algorithm, which are iteratively used to update the CNN parameters.

Sample Results from Weakly Supervised Change Detection Paper



Limitations:

Identifies areas of change but in most cases, we cannot classify the change object

Examples of Change Detection Cases – specific to plane detection & classification

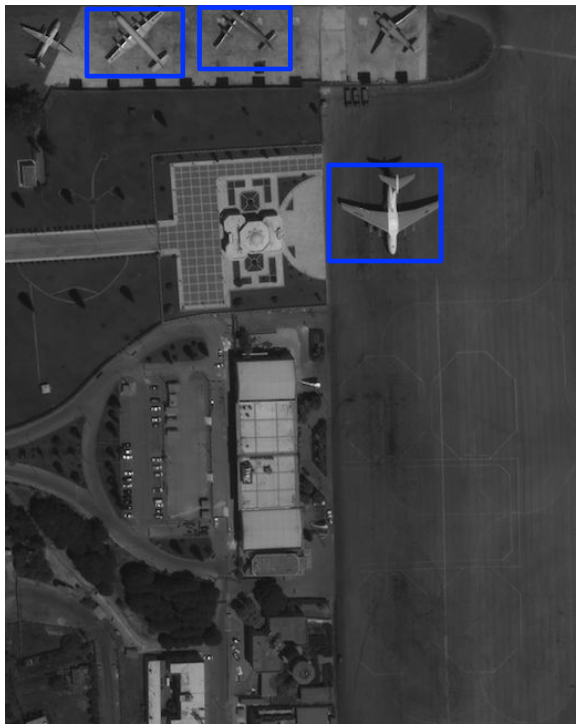
t = 0

t+

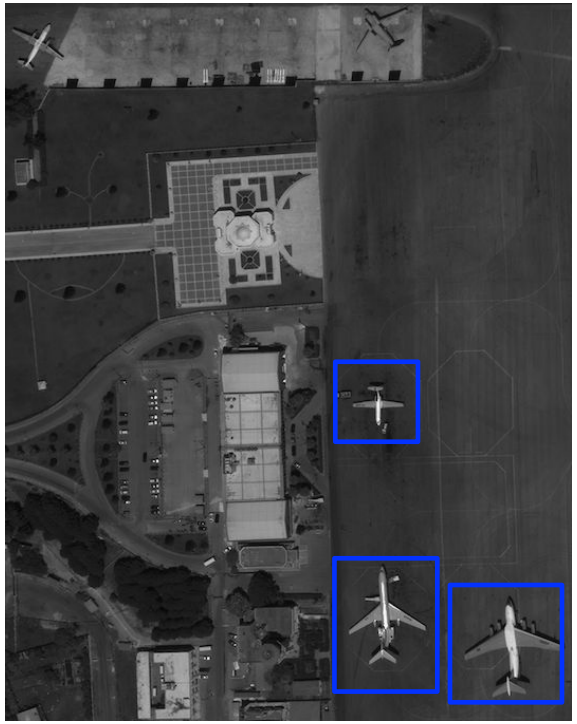
t++

Examples of Change Detection Cases with Plane Identification

t = 0



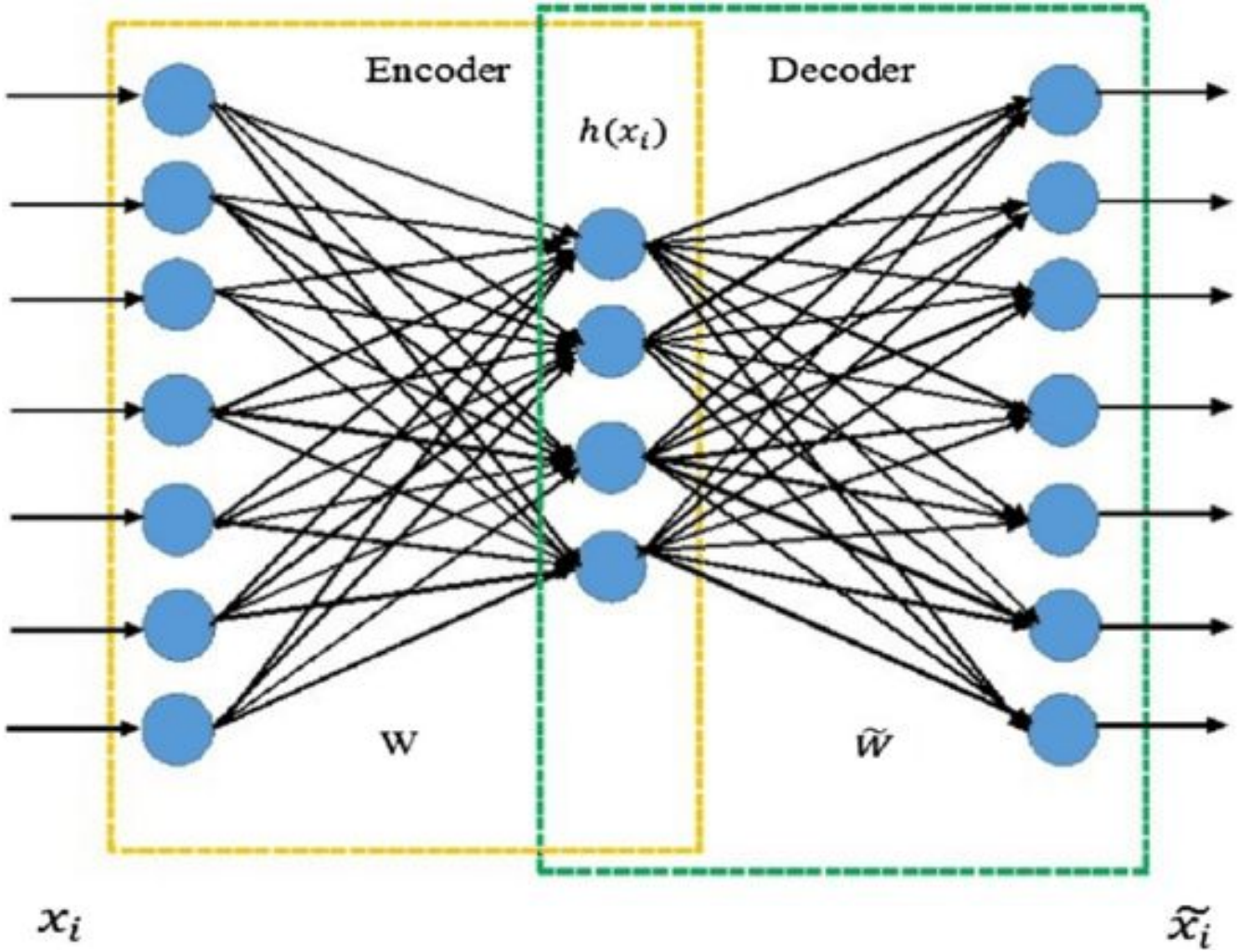
t+



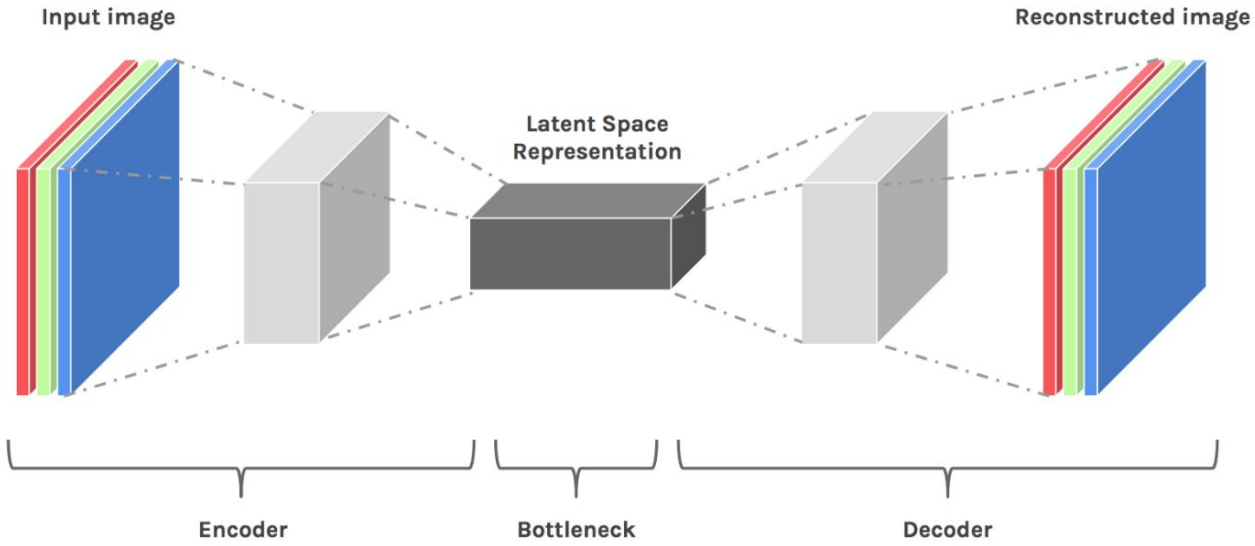
t++

Shallow Autoencoder

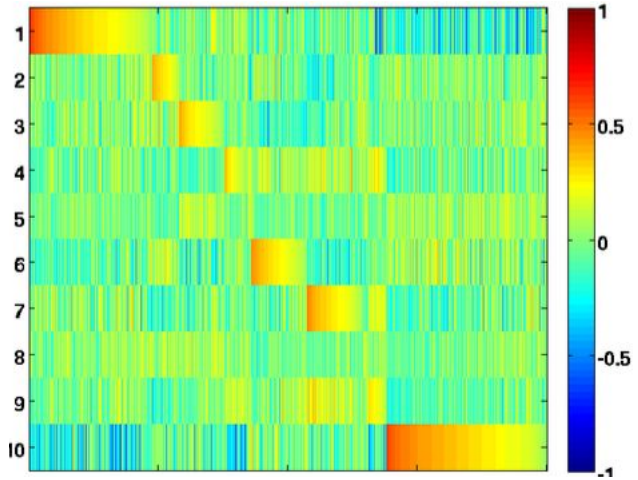
Conv2D(16, (3,3)) Conv2D(8, (3,3)) Conv2D(16, (3,3)



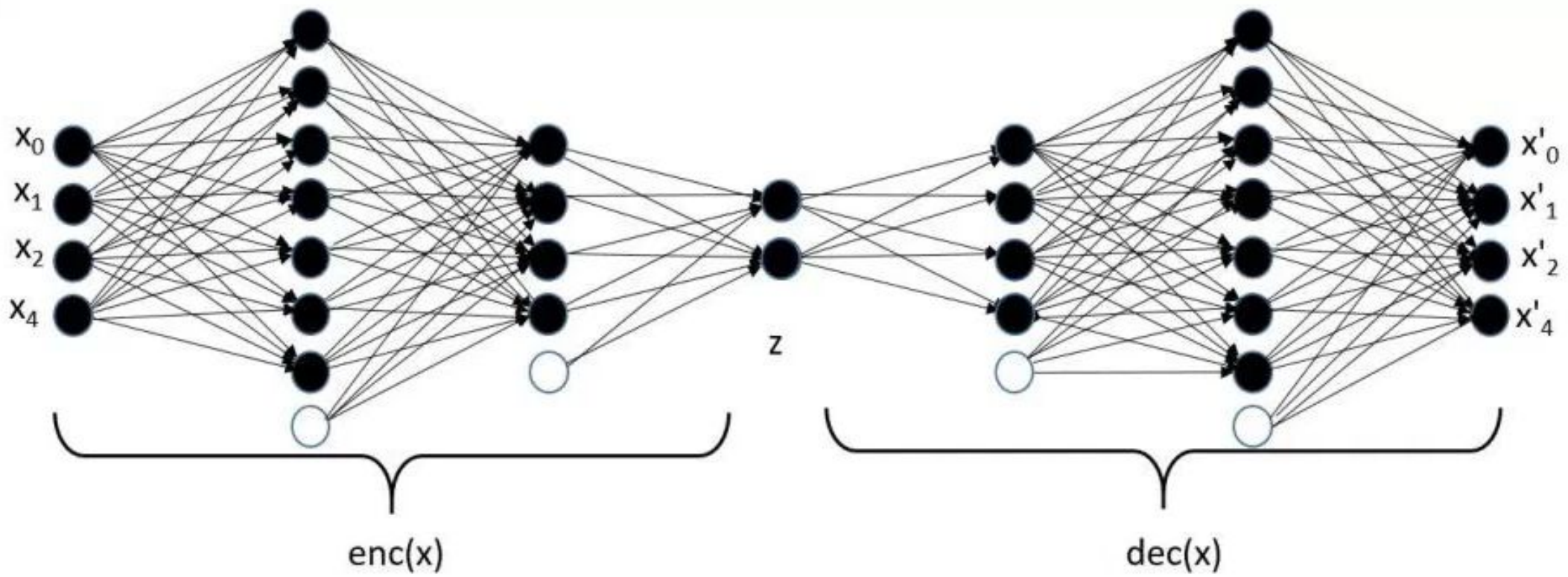
AutoEncoders

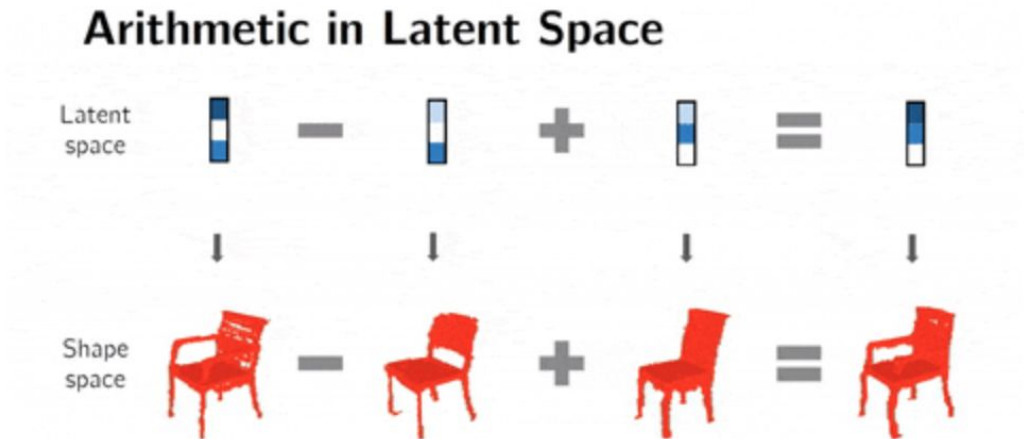


Convolutional Encoder-Decoder architecture



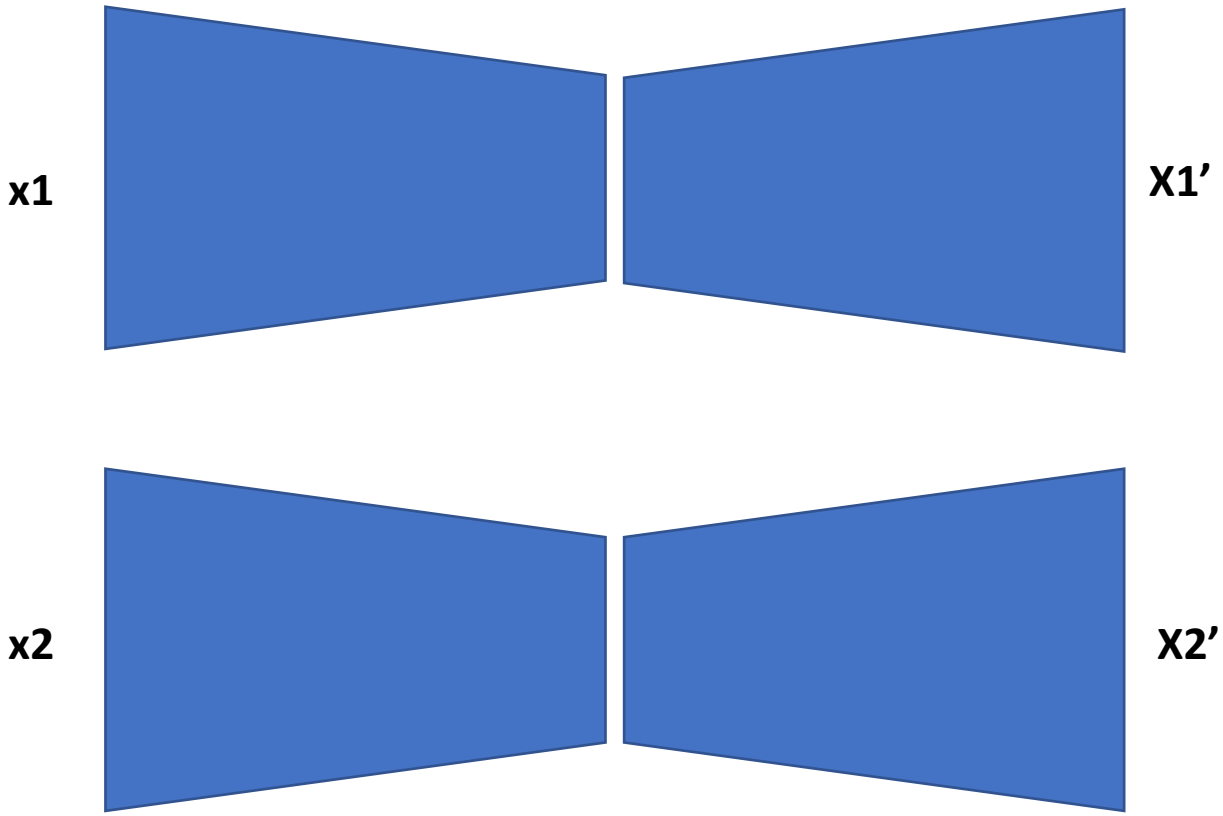
Propose to use Autoencoders for Change Detection:
Basic Concept is Depicted Here





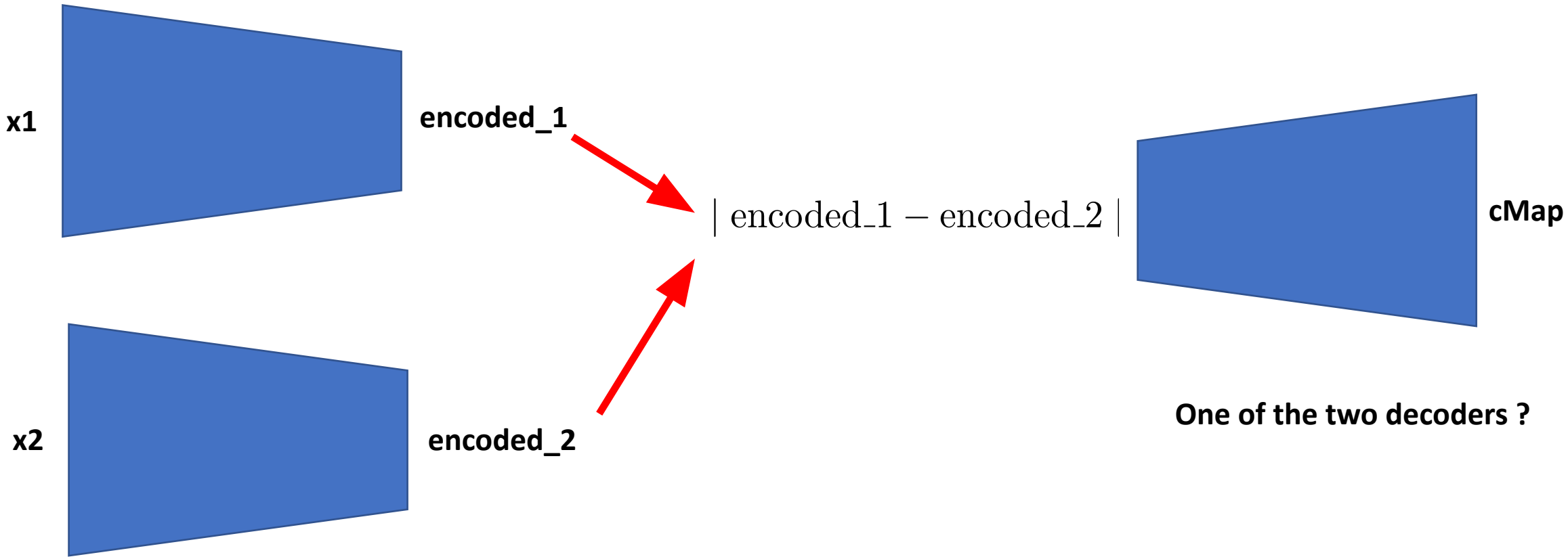
Two Autoencoder Approach to Change Map Generation

Training Phase:



Change Map from Difference in Latent Layers

Inference Phase I:



Shallow Encoder

Image # 1

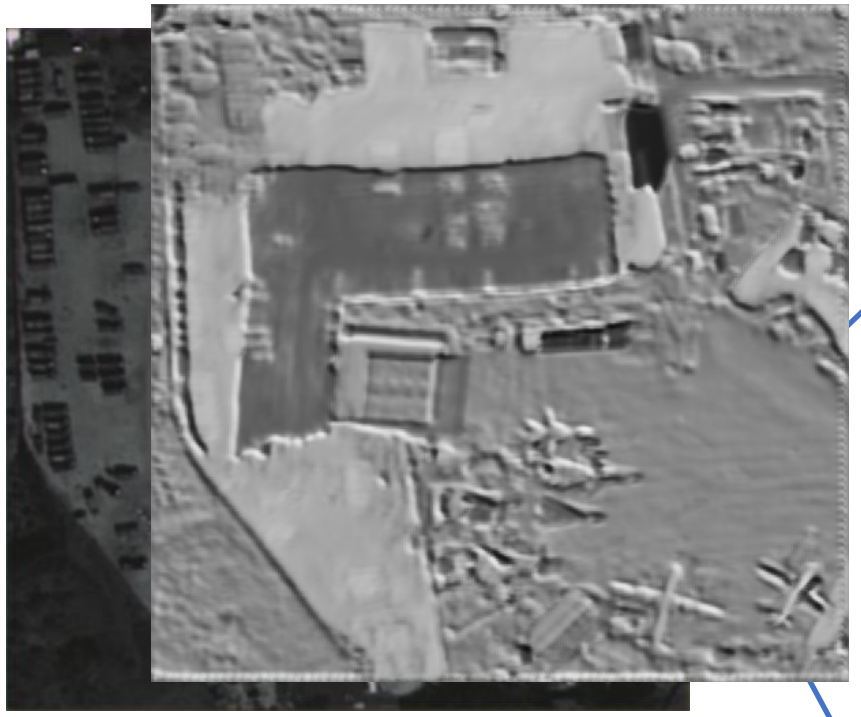


Image # 2

(image # 1 – image # 2)

Change Map (decoded_2 - decoded_1)

Change Map (encoded_2 - encoded_1)

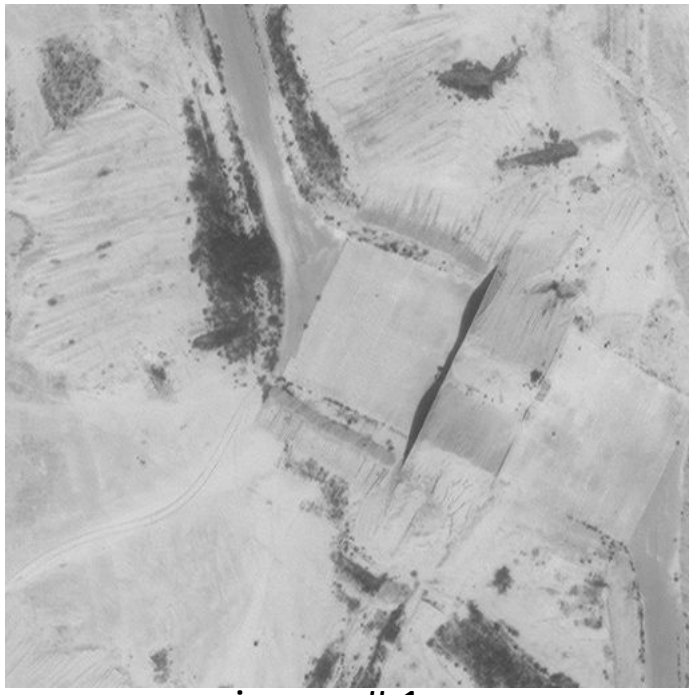


image # 1

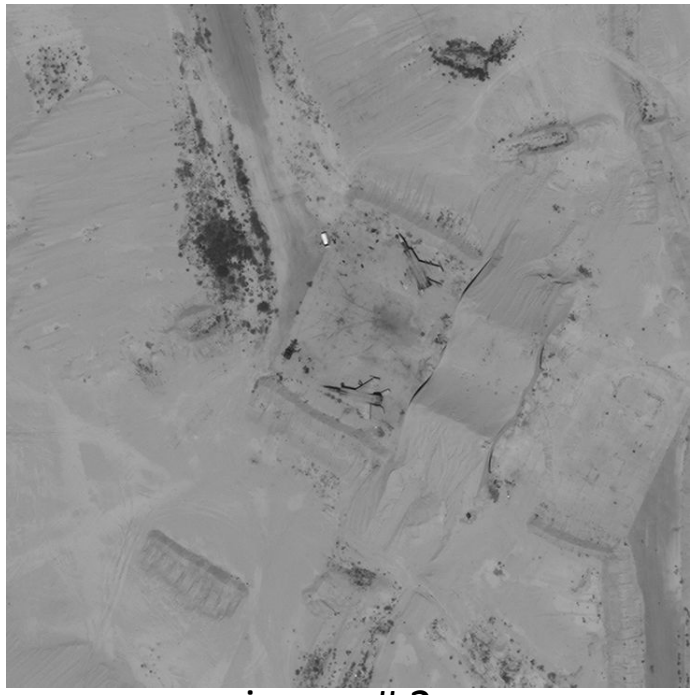
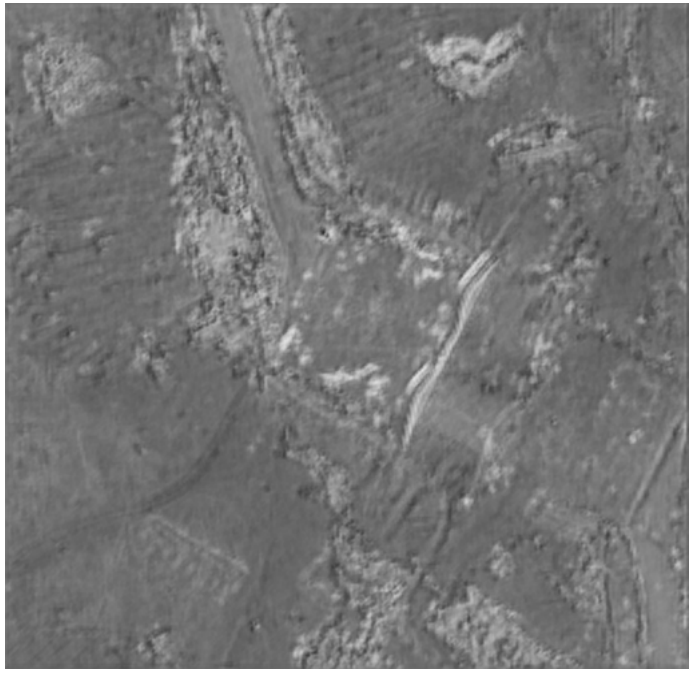


image # 2

(image # 1 – image # 2)

Change Map (decoded_2 - decoded_1)

Change Map (encoded_2 - encoded_1)



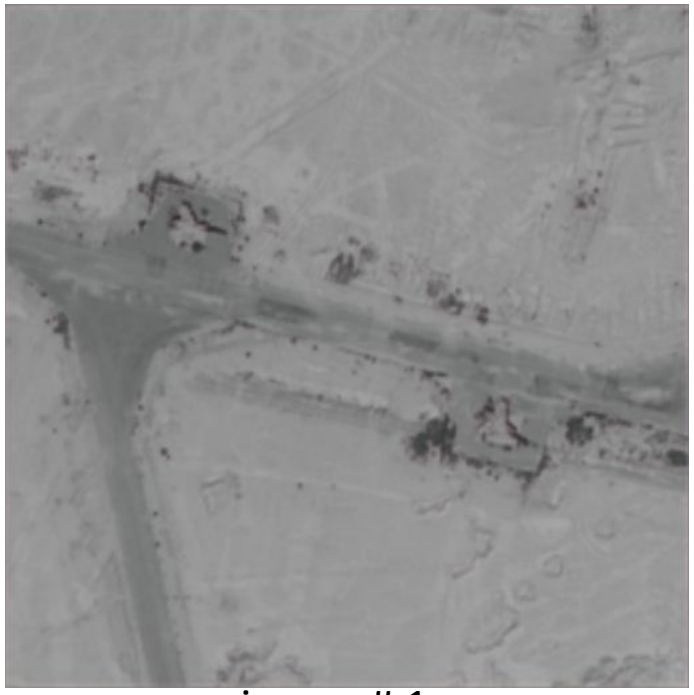


image # 1

image # 2

(image # 1 – image # 2)

Change Map (decoded_2 - decoded_1)

Change Map (encoded_2 - encoded_1)

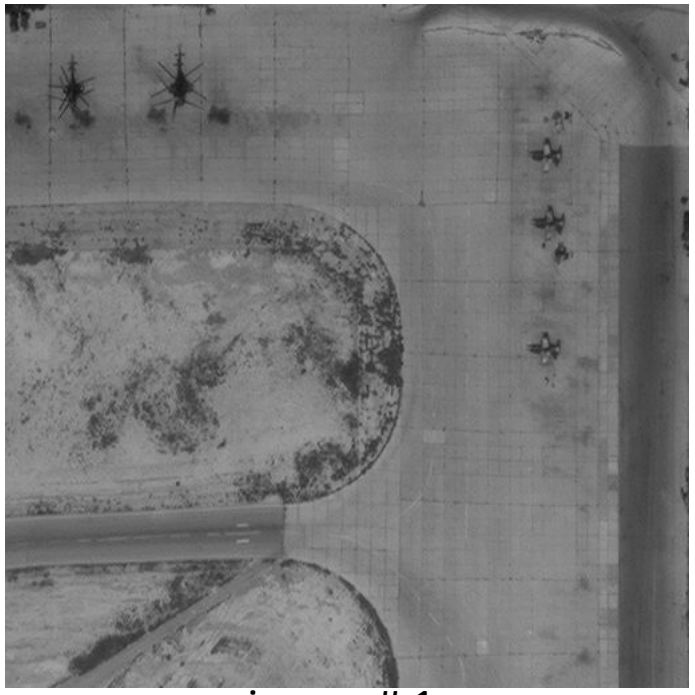


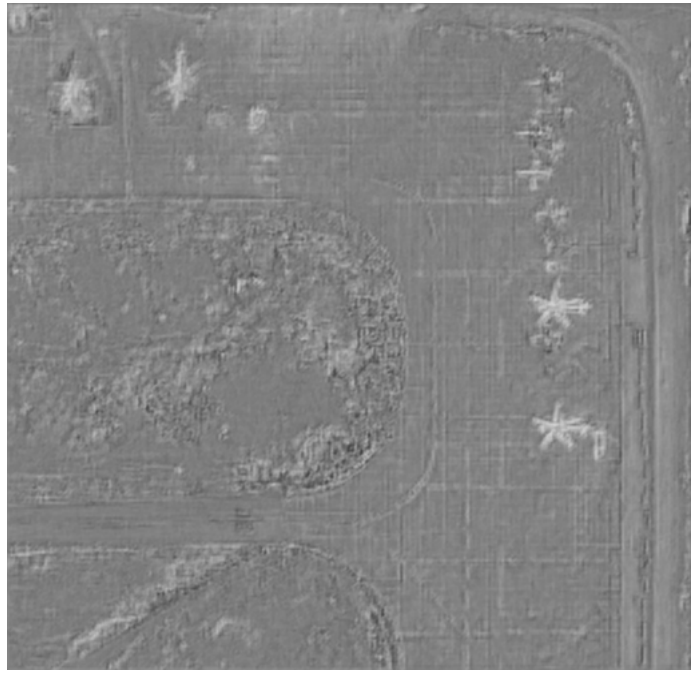
image # 1

image # 2

(image # 1 – image # 2)

Change Map (decoded_2 - decoded_1)

Change Map (encoded_2 - encoded_1)



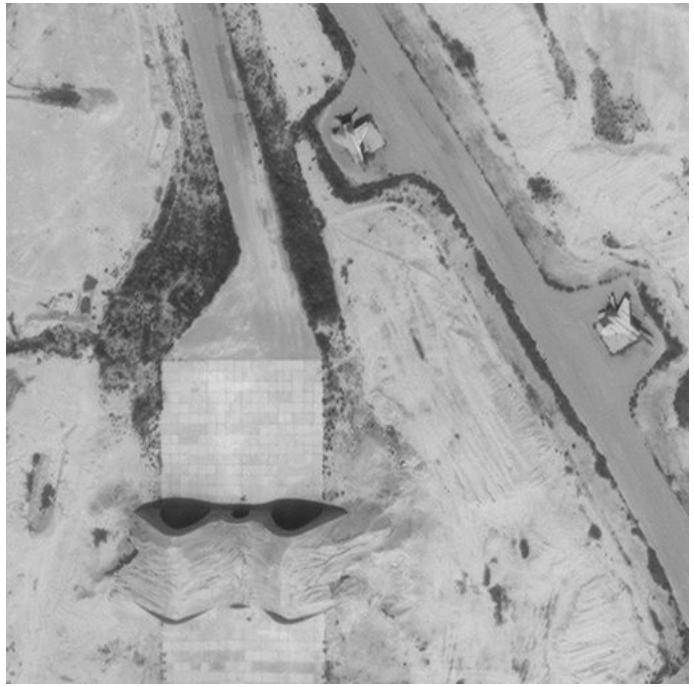
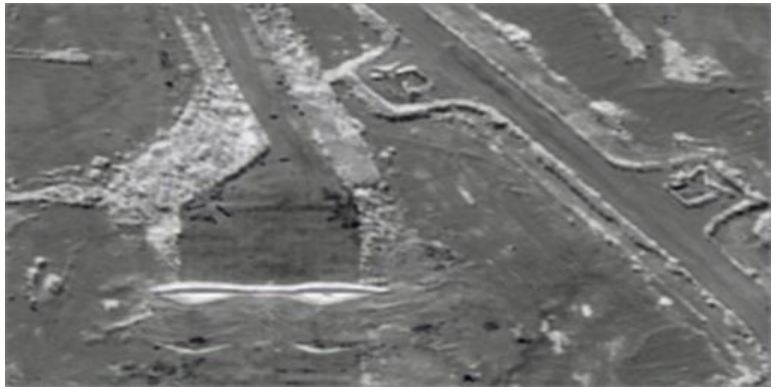


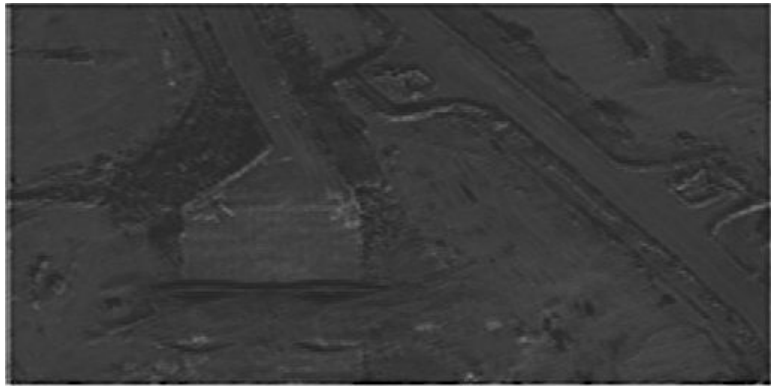
image # 1

image # 2

Change Map (decoded_2 - decoded_1)



Change Map (encoded_2 - encoded_1)



Results Analysis

- The results are a marginal compared to a simple difference image with shallow network (**2 layers**)
- Try a deeper autoencoder (up to **8 layers** per encoder & decoder module)

Deep Autoencoder

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 512, 512, 3)	0
conv2d_1 (Conv2D)	(None, 512, 512, 16)	448
max_pooling2d_1 (MaxPooling2	(None, 256, 256, 16)	0
conv2d_2 (Conv2D)	(None, 256, 256, 8)	1160
encoder (MaxPooling2D)	(None, 128, 128, 8)	0
conv1_dec (Conv2D)	(None, 128, 128, 8)	584
up_sampling2d_1 (UpSampling2	(None, 256, 256, 8)	0
conv11_dec (Conv2D)	(None, 256, 256, 16)	1168
up_sampling2d_2 (UpSampling2	(None, 512, 512, 16)	0
conv2_dec (Conv2D)	(None, 512, 512, 3)	435

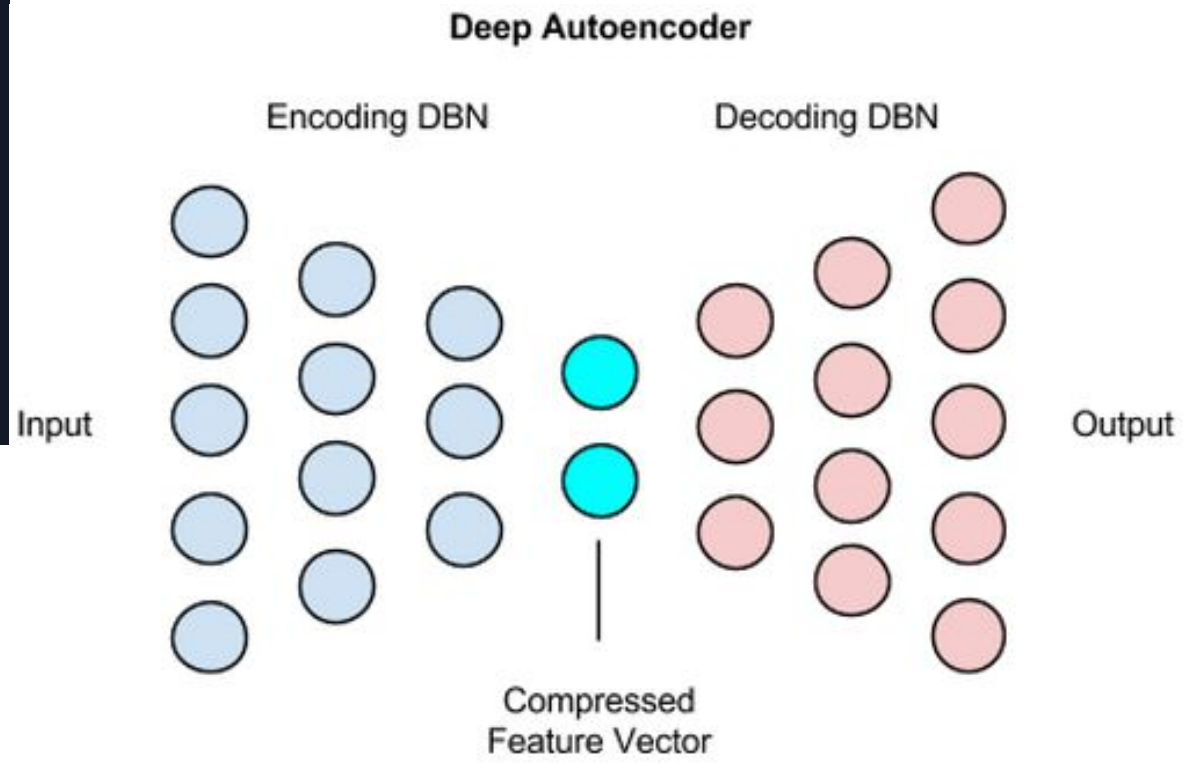


image # 1

image # 2

Change Map (decoded_2 - decoded_1)

Change Map (encoded_2 - encoded_1)

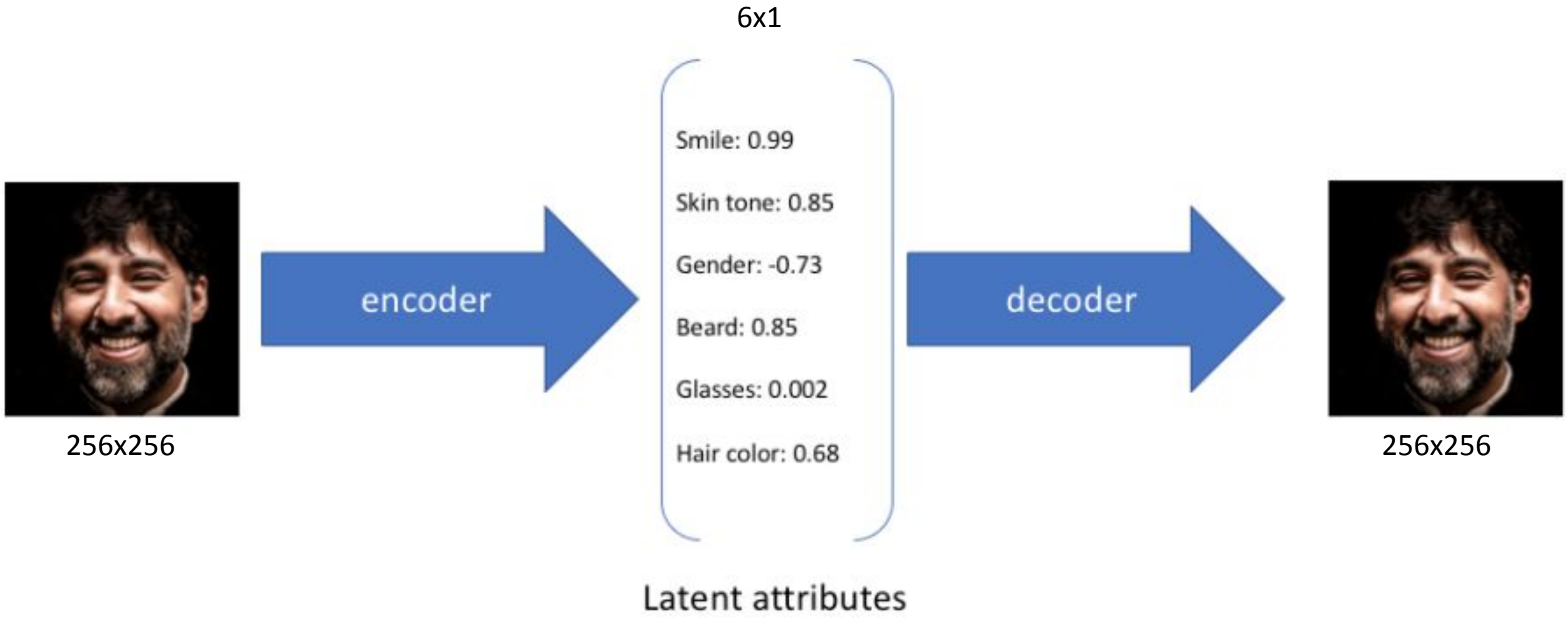
Summary of Work Status with Simple Autoencoders:

- Used two Autoencoders to Reduce an Image Pairs to Smaller Dimensional form - *Latent Space*
- Subtraction of the Paired Reduced Dimensional Space before Decoding to generate Change Map
- Results showed ***minimal improvements over a simple difference*** map
- Switched to ***Variational Autoencoders*** for subtle encoding of features in Latent Space (Reduced Dimensional Space)

Results Analysis

- The results are a marginal compared to a simple difference image or even a shallow network. There are several problems identified in this approach:
  1. The latent layer of the encode is a single value per feature. It does not encode subtle details in the image
  2. Normalizing of the image pair to achieve same intensity per corresponding pixel.
- Introduce concept of variational autoencoder to the change detection problem to address the first issue

Latent Attributes Coding Limitations

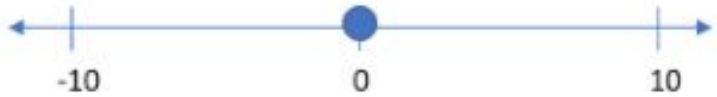


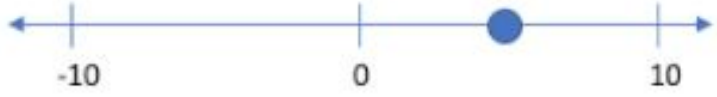
- Standard autoencoder outputs a single value for each encoding dimension
- The decoder takes these values and attempts to recreate the original input
- Instead, we will represent each **latent attribute** for a given input **as a probability distribution**

Variational Autoencoders

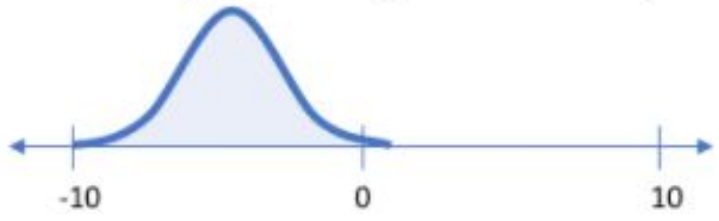
Variational Autoencoders

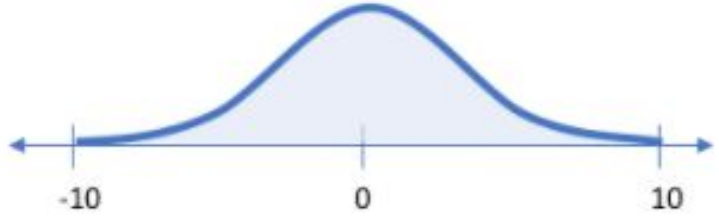
Smile (discrete value)



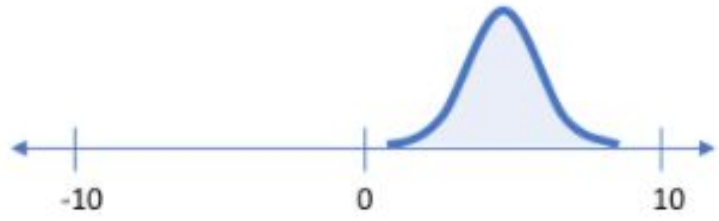


Smile (probability distribution)

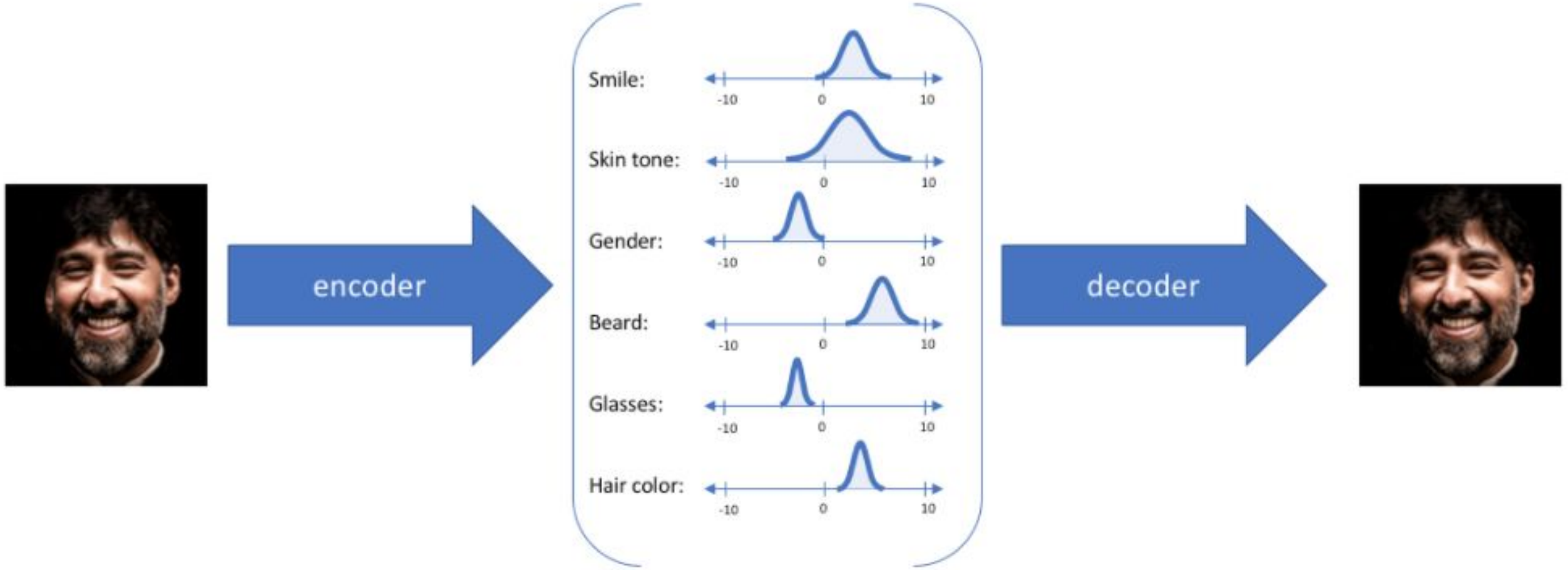




vs.

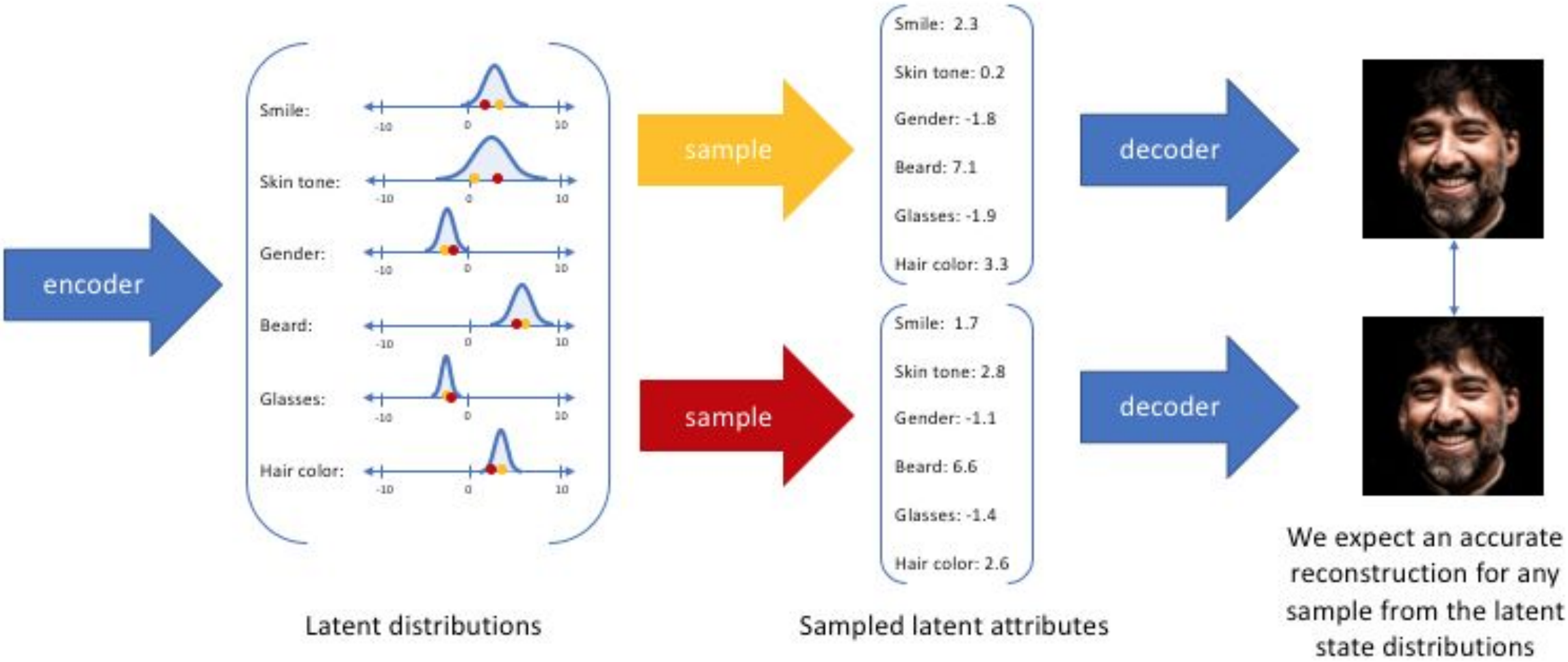


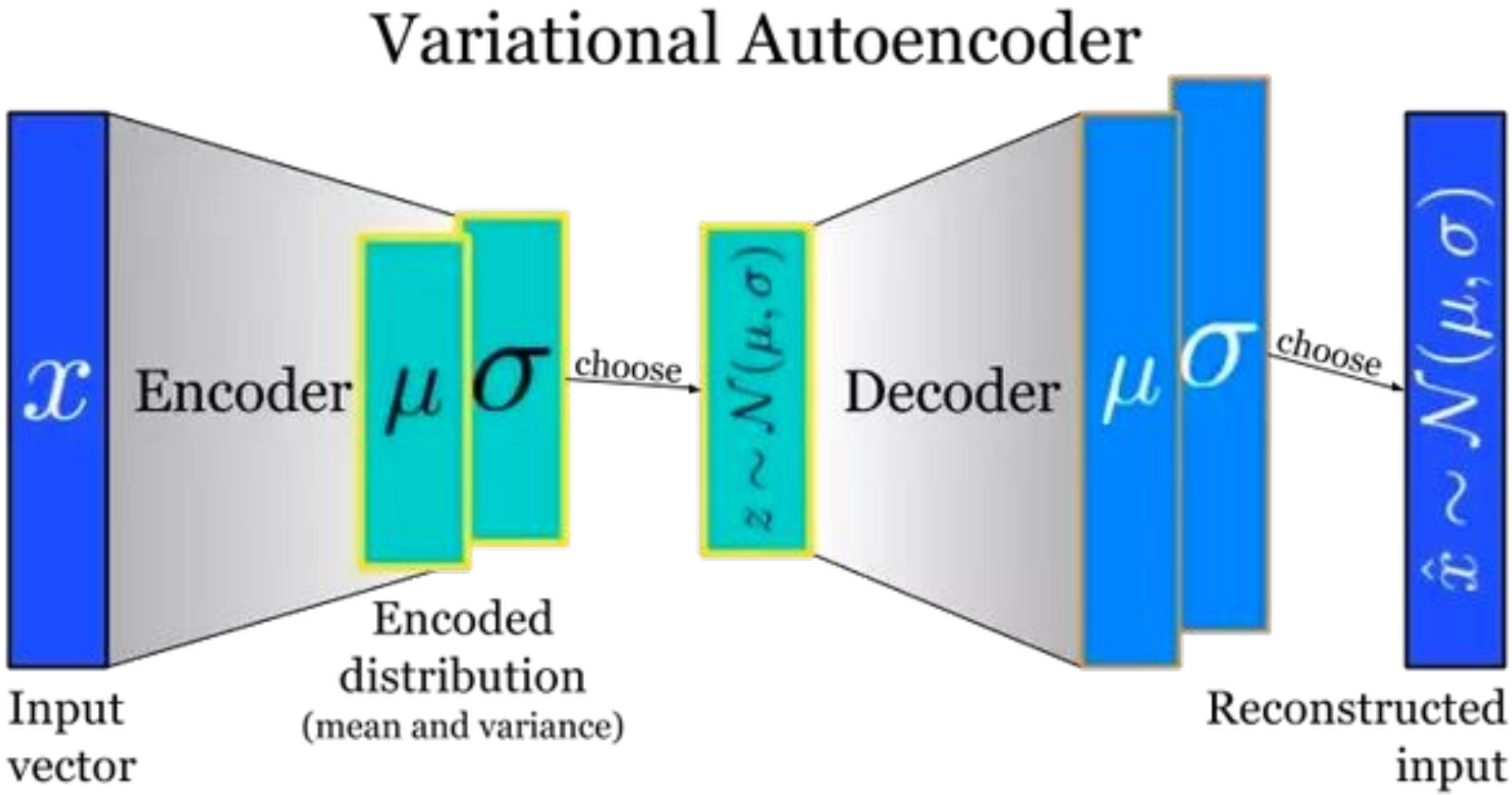
Latent Attributes



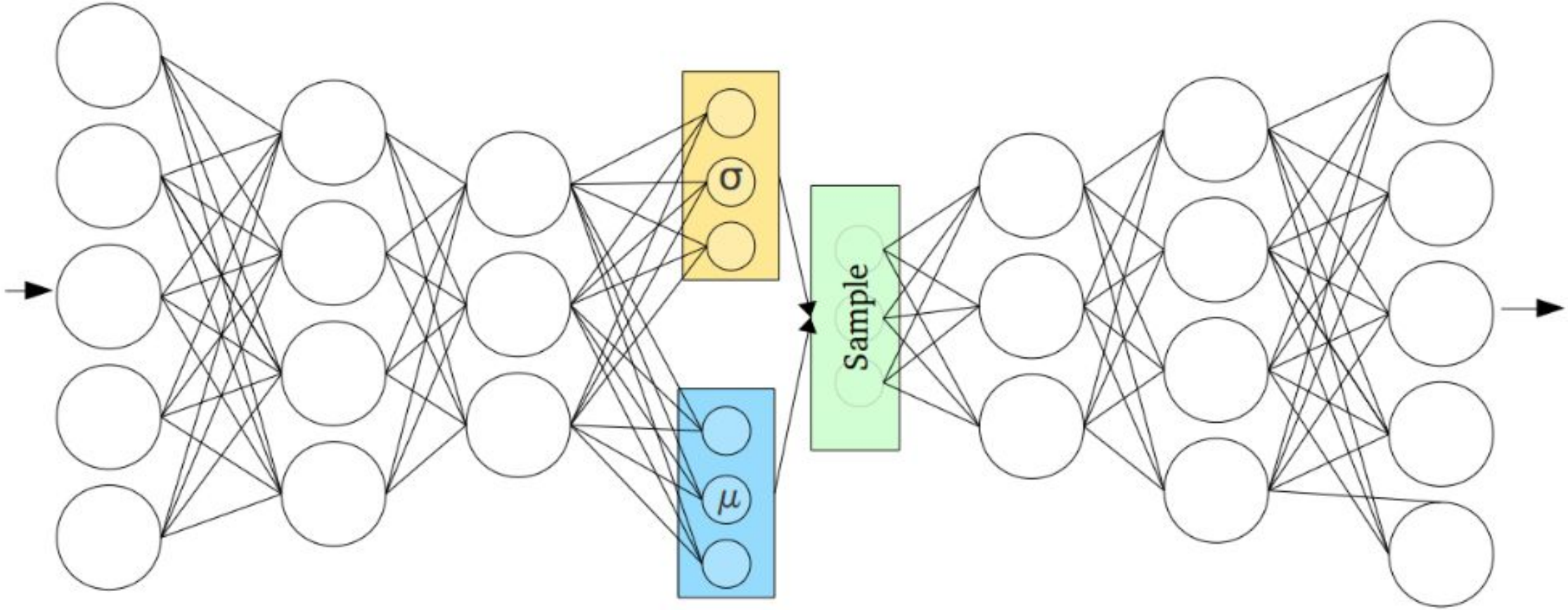
Decoding from the latent state, we will randomly sample from each latent state a distribution to generate a vector as input to the decoder model

Sampling from Latent Space Distributions




Variational Autoencoder
x
Encoder
μ
σ
choose
$z \sim \mathcal{N}(\mu, \sigma)$
Decoder
μ
σ
choose
$\hat{x} \sim \mathcal{N}(\mu, \sigma)$
Input vector
Encoded distribution (mean and variance)
Reconstructed input

Variational Autoencoder



Encoder/Recognition model

Decoder/Generative model

Variational Autoencoder (VAE)

Variational Autoencoder (2013) work prior to GANs (2014)

- Explicit Modelling of $P(X|z; \theta)$, we will drop the θ in the notation.
- $z \sim P(z)$, which we can sample from, such as a Gaussian distribution.

$$P(X) = \int P(X|z;\theta)P(z)dz$$

- Maximum Likelihood --- Find θ to maximize $P(X)$, where X is the data.
- Approximate with samples of z

$$P(X) \approx \frac{1}{n}\sum_{i=0}^{n} P(X|z_i)$$

Variational Autoencoder (VAE)

Variational Autoencoder (2013) work prior to GANs (2014)

- Explicit Modelling of $P(X|z; \theta)$, we will drop the θ in the notation.
- $z \sim P(z)$, which we can sample from, such as a Gaussian distribution.

$$P(X) = \int P(X|z;\theta)P(z)dz$$

- Maximum Likelihood --- Find θ to maximize $P(X)$, where X is the data.
- Approximate with samples of z

$$P(X) \approx \frac{1}{n}\sum_{i=0}^{n} P(X|z_i)$$

Variational Autoencoder (VAE)

Variational Autoencoder (2013) work prior to GANs (2014)

- Explicit Modelling of $P(X|z; \theta)$, we will drop the θ in the notation.
- $z \sim P(z)$, which we can sample from, such as a Gaussian distribution.

$$P(X) = \int P(X|z;\theta)P(z)dz$$

- Maximum Likelihood --- Find θ to maximize $P(X)$, where X is the data.
- Approximate with samples of z

$$P(X) \approx \frac{1}{n}\sum_{i=0}^{n} P(X|z_i)$$

Variational Autoencoder (VAE)

- Approximate with samples of *z*

$$P(X) \approx \frac{1}{n} \sum_{i=0}^{n} P(X|z_i)$$

- Need a lot of samples of z and most of the **P(X|z) ≈ 0.**
- Not practical computationally.
- **Question:** Is it possible to know which z will generate P(X|z) >> 0?
  - Learn a distribution Q(z), where z ~ Q(z) generates P(X|z) >> 0.

Variational Autoencoder (VAE)

- Approximate with samples of *z*

$$P(X) \approx \frac{1}{n} \sum_{i=0}^{n} P(X|z_i)$$

- Need a lot of samples of z and most of the **P(X|z) ≈ 0.**
- Not practical computationally.
- **Question:** Is it possible to know which z will generate P(X|z) >> 0?
  - Learn a distribution Q(z), where z ~ Q(z) generates P(X|z) >> 0.

VAE's Loss function

Convert the lower bound to a loss function:

$$\log P(X) - \mathcal{D}\left[Q(z) \| P(z|X)\right] = E_{z\sim Q}\left[\log P(X|z)\right] - \mathcal{D}\left[Q(z) \| P(z)\right]$$

Assume P(z) ~ N(0,I) then D[Q(z|X) || P(z)] has a closed form solution.

Putting it all together: $E_{z\sim Q(z|X)} \log P(X|z) \propto ||X-f(z)||^2$

$$L = ||X - f(z)||^2 - \lambda \cdot D[Q(z) \| P(z)]$$

, given a (X, z) pair.

Pixel difference

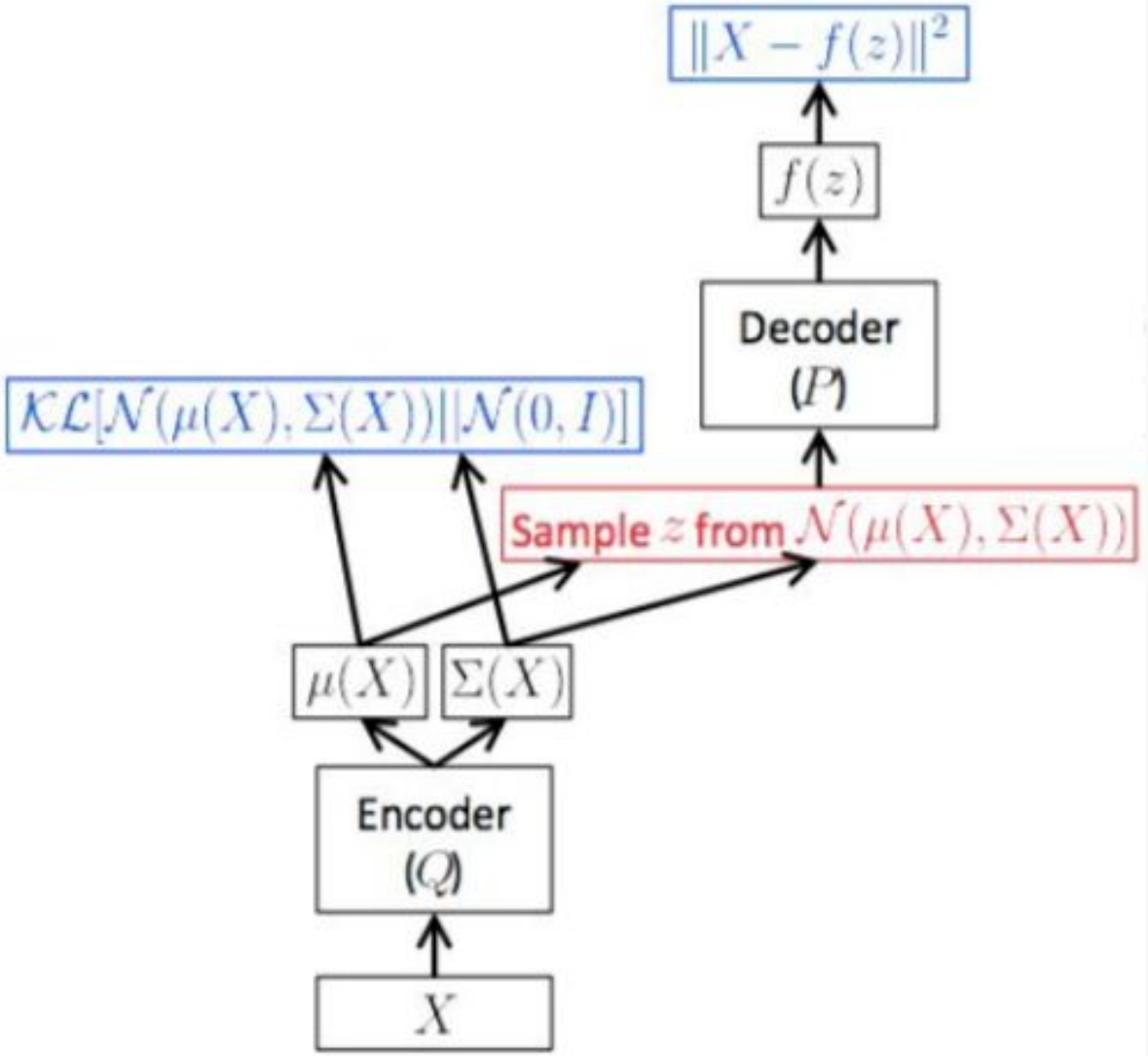
Regularization

Variational Autoencoder

Training the Decoder is easy, just standard backpropagation.

How to train the Encoder?

- Not obvious how to apply gradient descent through samples.

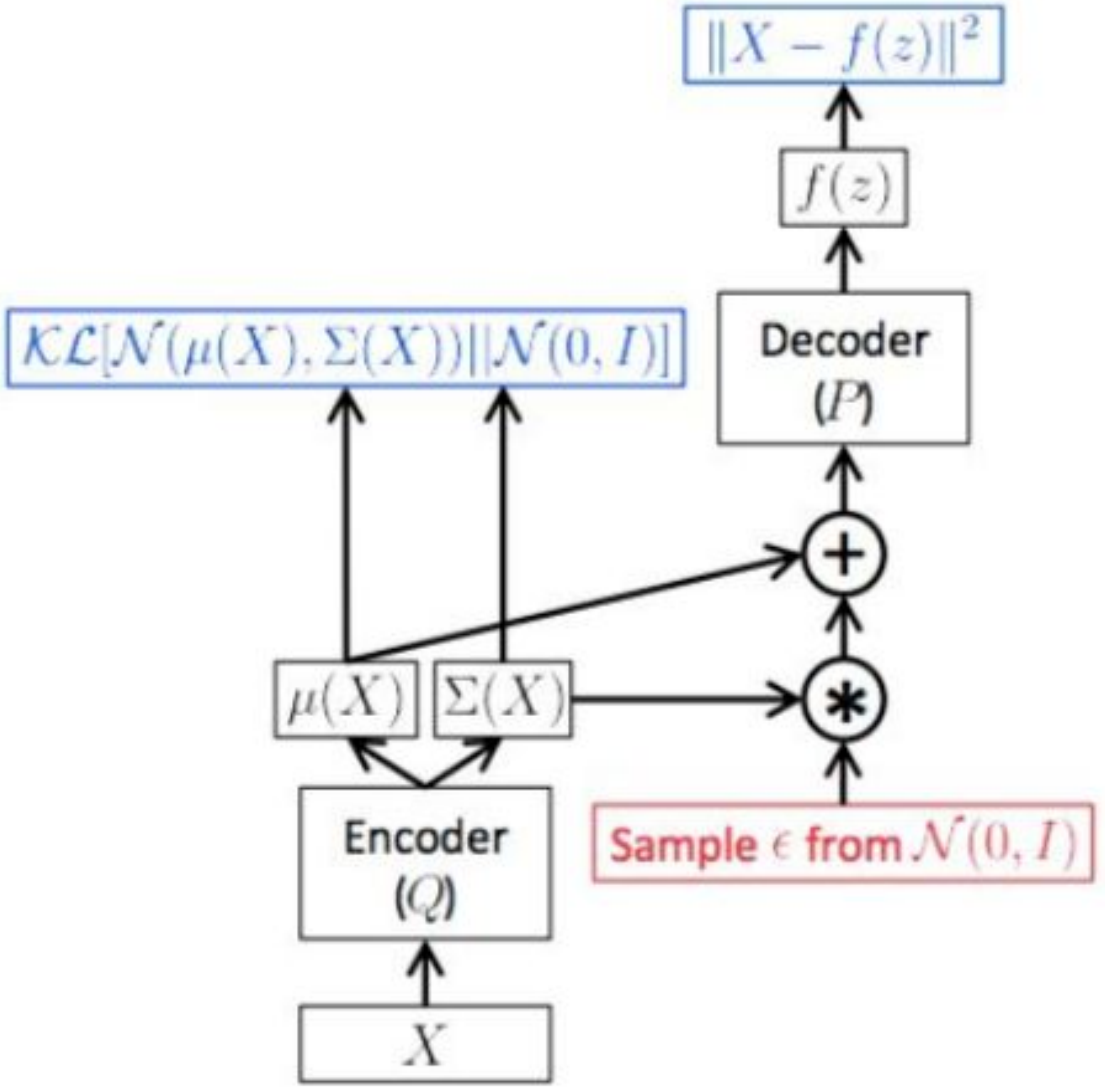


Reparameterization Trick

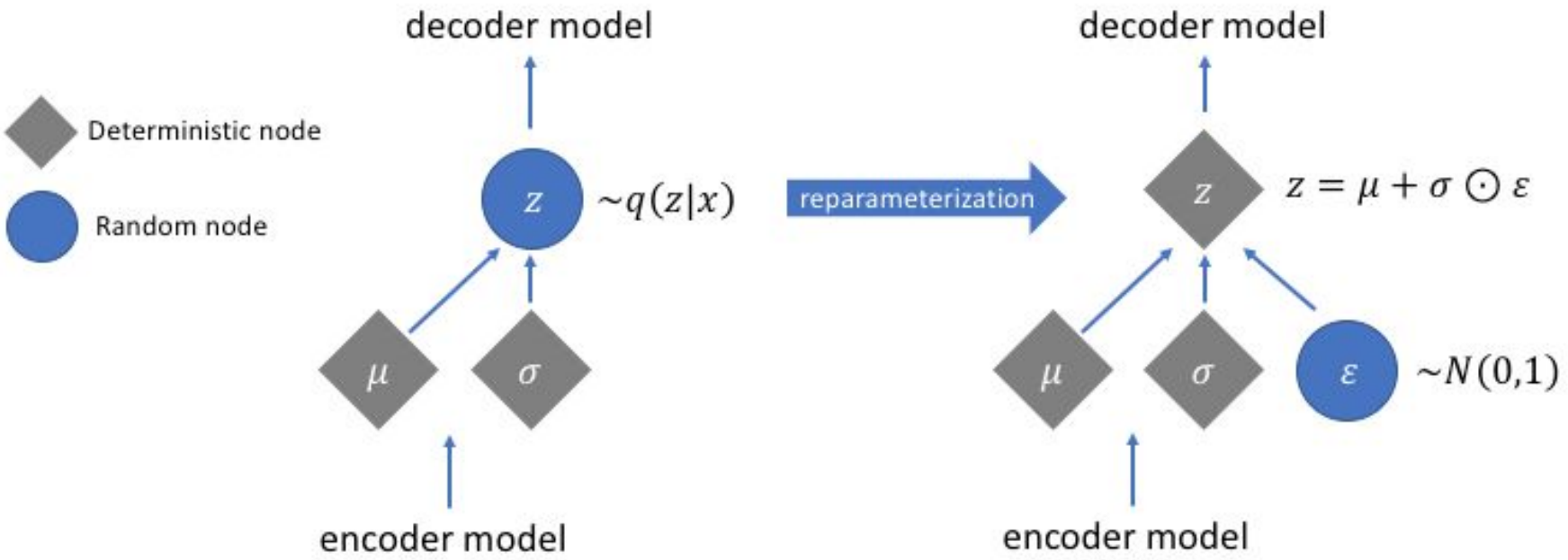
How to effectively backpropagate through the z samples to the Encoder?

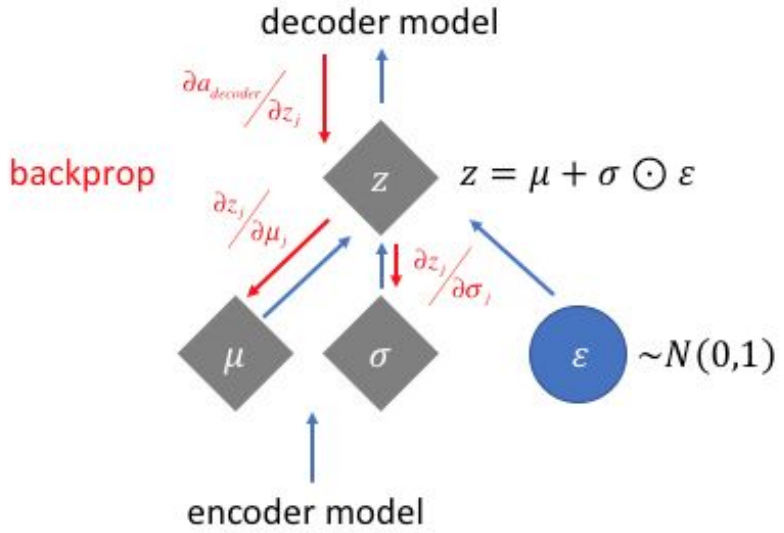
Reparametrization Trick

- $z \sim N(\mu, \sigma)$ is equivalent to
- $\mu + \sigma \cdot \varepsilon$, where $\varepsilon \sim N(0, 1)$
- Now we can easily backpropagate the loss to the Encoder.



Reparameterization Trick





VAE Training

Given a dataset of examples **X** = {X1, X2...}

Initialize parameters for Encoder and Decoder

Repeat till convergence:

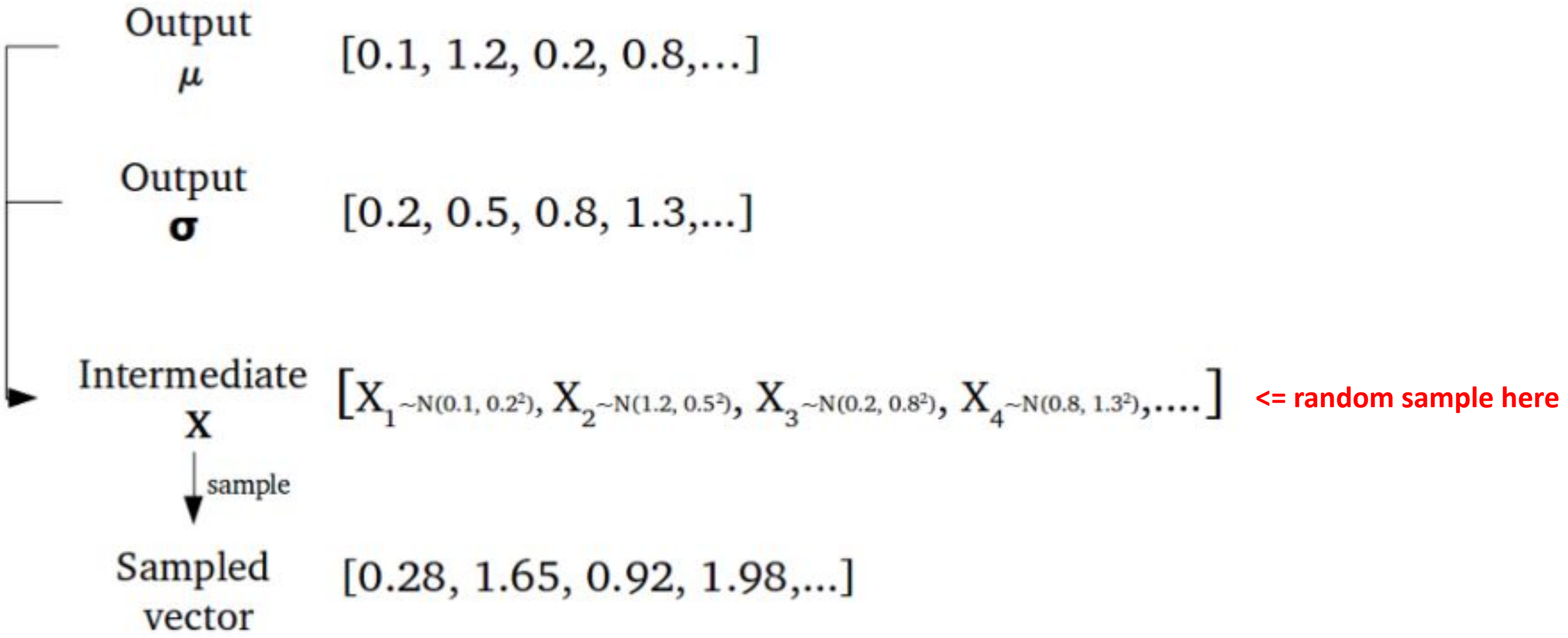
$\mathbf{X}^{\mathbf{M}}$ <-- Random minibatch of M examples from **X**

ε <-- Sample M noise vectors from N(0, I)

Compute $L(\mathbf{X}^{\mathbf{M}}, \varepsilon, \theta)$ (i.e. run a forward pass in the neural network)

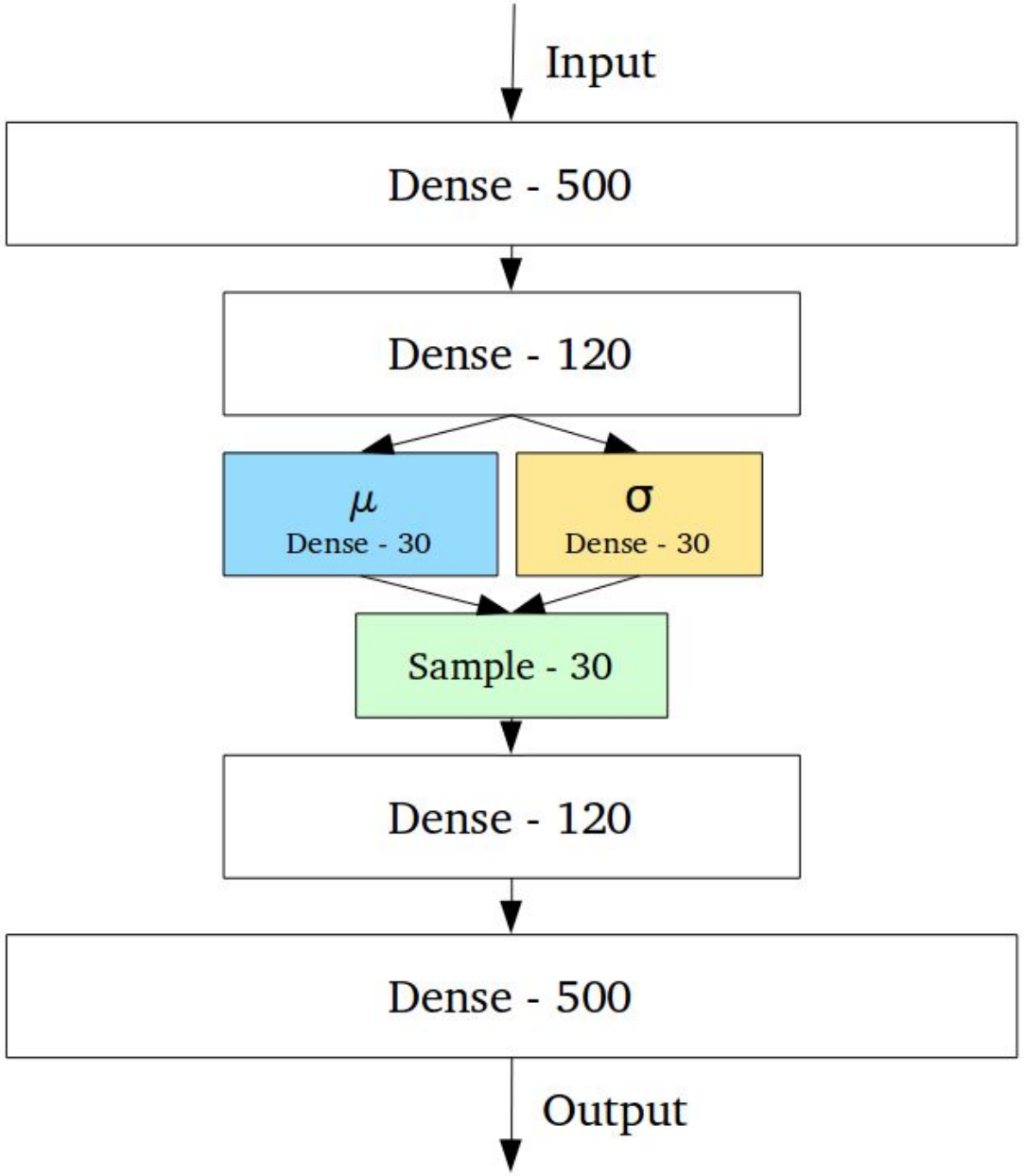
Gradient descent on L to updated Encoder and Decoder.

Latent Layer Replacement



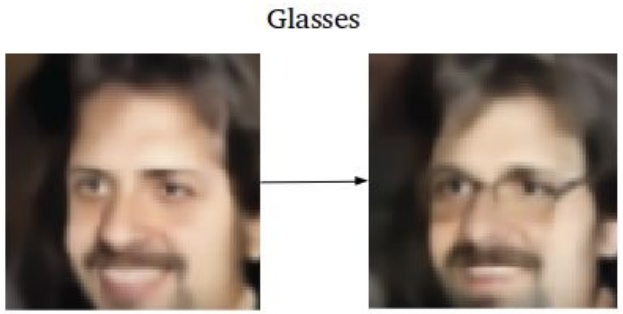
Stochastically generating encoding vectors

Variational Autoencoder – An Example

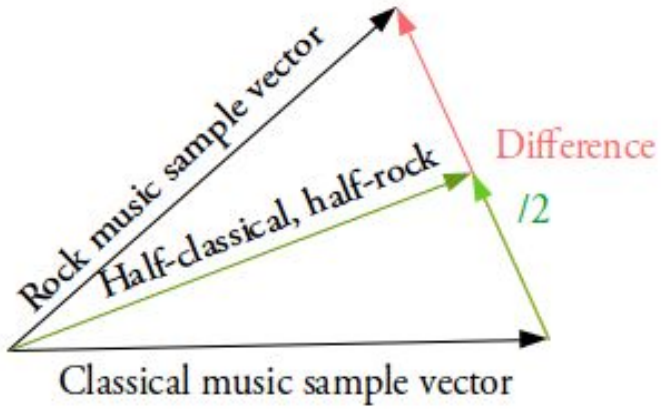


Generating celebrity-lookalike photos

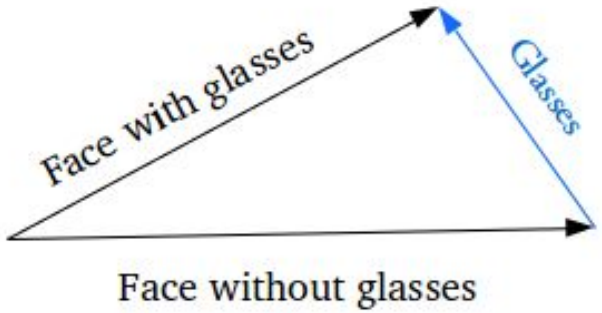
Vector Arithmetic



Exploring a specific variation of input data[1]



Interpolating between samples

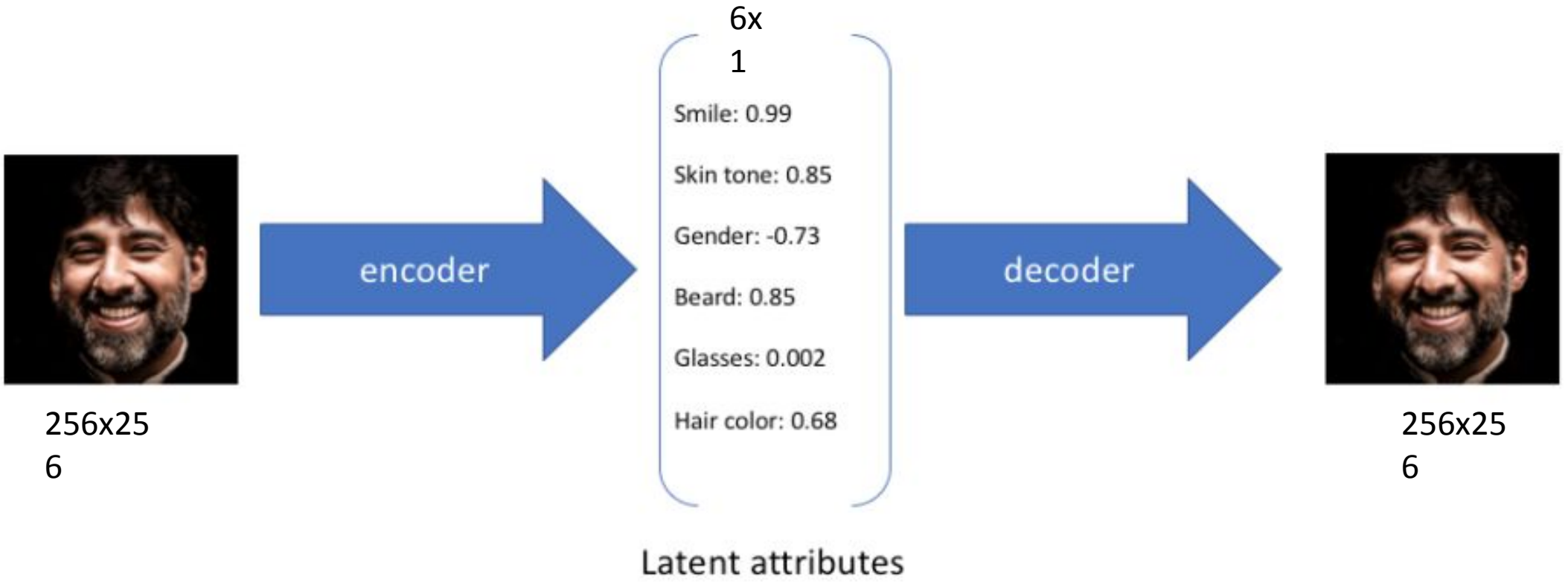


Adding new features to samples

For example, if you wish to generate a new sample halfway between two samples, just find the difference between their sample vectors, and add half the difference to the original, and then simply decode it.

What about generating *specific features*, such as generating glasses on a face? Find two samples, one with glasses, one without, obtain their encoded vectors from the encoder, and save the difference. Add this new "glasses" vector to any other face image, and just decode it.

Latent Attributes



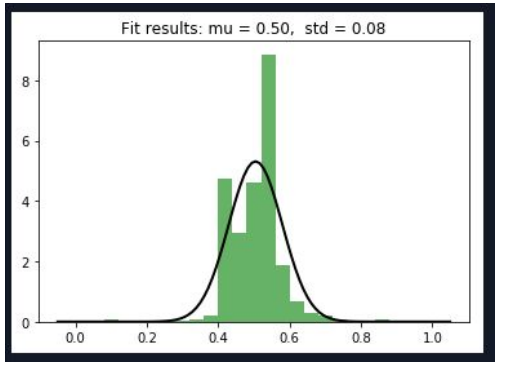
- Standard autoencoder outputs a single value for each encoding dimension
- The decoder takes these values and attempts to recreate the original input
- Instead, we represented each **latent attribute** for a given input **as a probability distribution – Variational Autoencoder**

- ***Results were Still Mixed as shown in the Subsequent Slides***

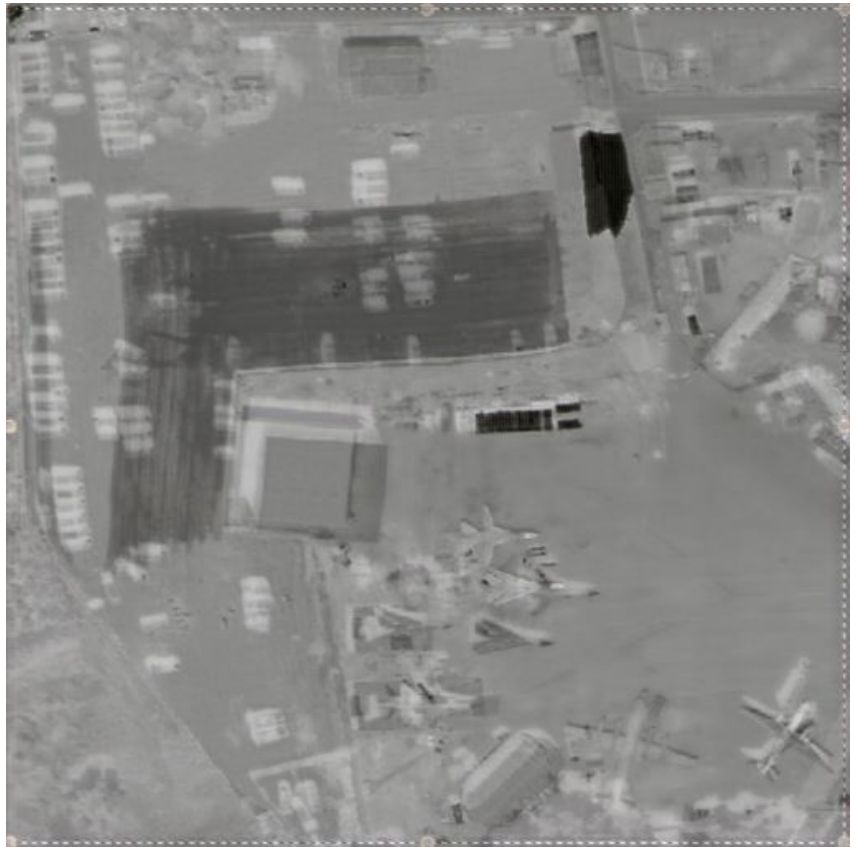
x_1

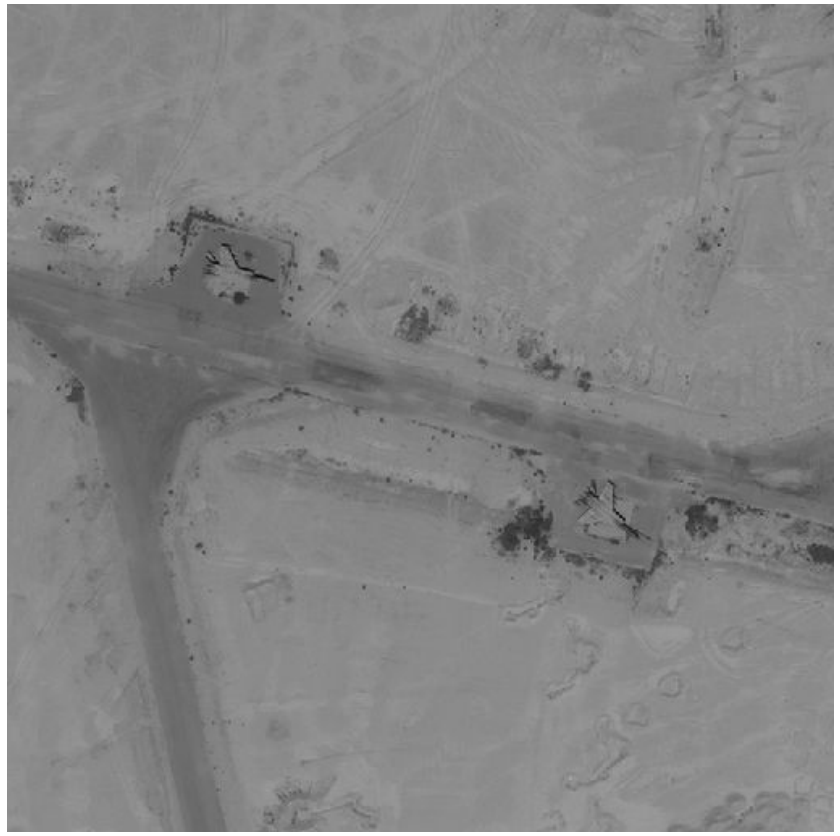
x_2

x_1 – x_2



Change map (decoded2-decoded1)

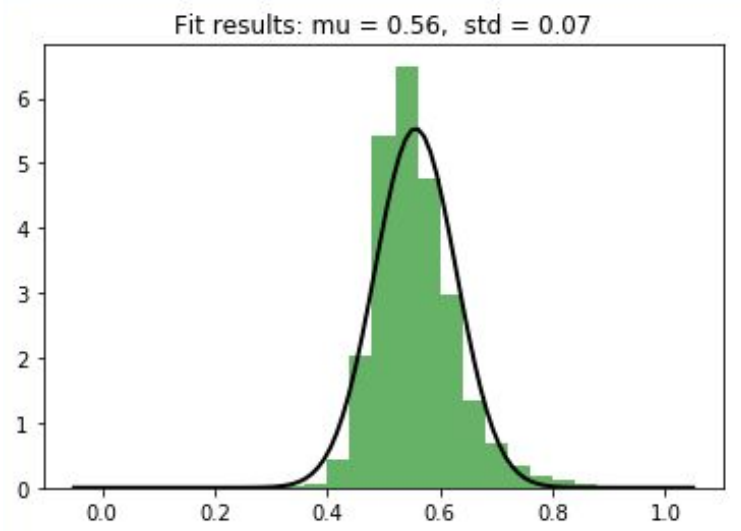




x_1

x_2

Difference image distribution



change map

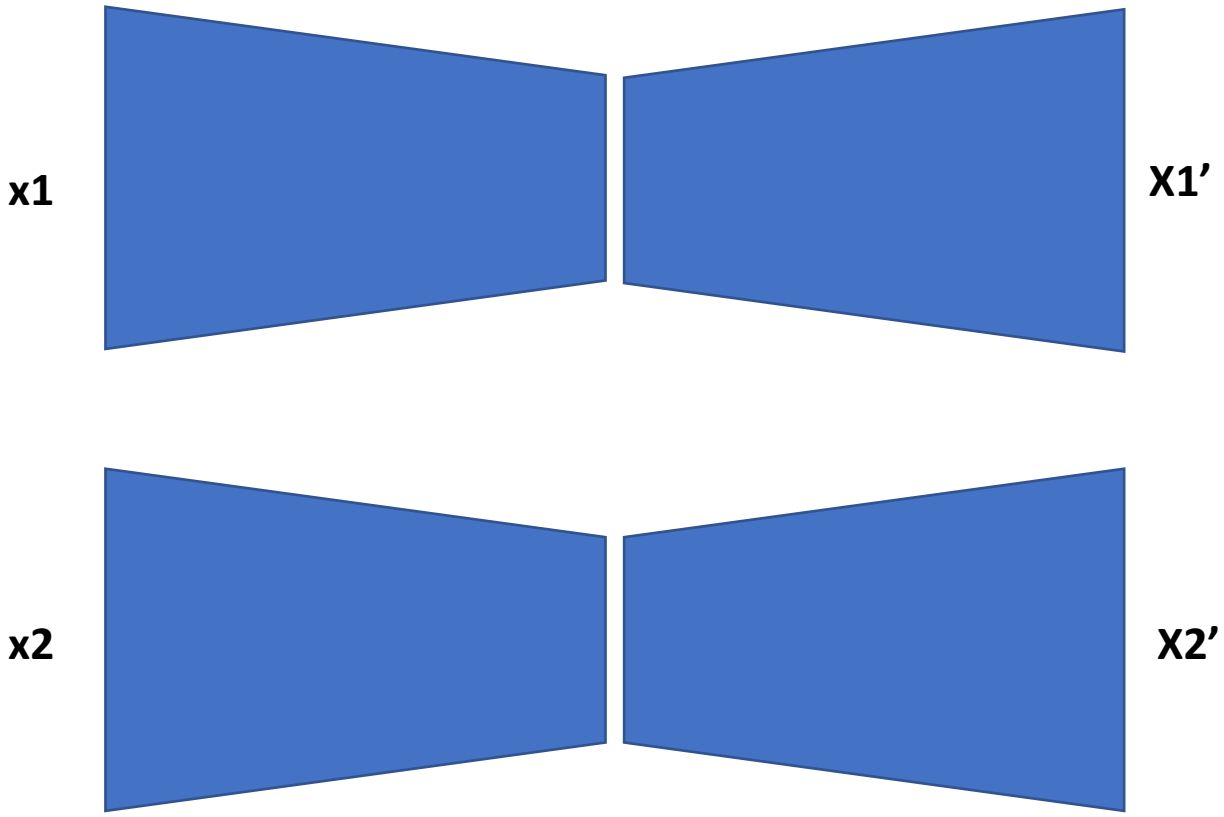
Key Issues Pending or Addressed

- Trying ***smaller image*** pairs (32x32, 64x64, 128x128 etc)
- Dealing with ***Negative Numbers*** after Subtraction:
  - RGB is 0-255 => scale range to (0,1]
- ***Balancing intensity*** between pairs of images before change detection algorithm
  - Normalization of image pair to zero mean and unit variance
- Change Map ***Intensity Thresholding*** for Additional Background Suppression
- **Satellite angle Changes:**
  - ***Aspect Ratio***
  - **Pixel Intensity Variations on Image Pairs**
- **Issues of Global Change versus *Use Case* based Change Detection/Specificity**

Two Autoencoder Approach to Change Map Generation

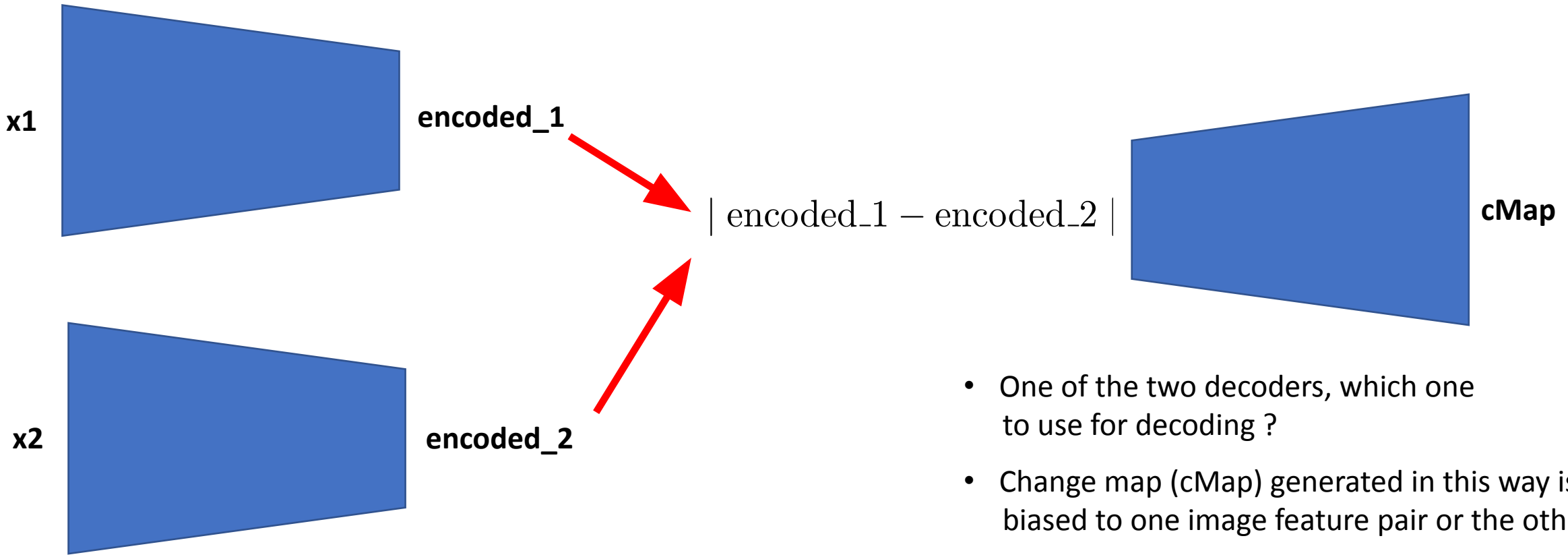
Training Phase:

- Train two separate encoders per image
- Two separate encoded (latent layers)
- Two separate decoded images x1’ & x2’



Change Map from Difference in Latent Layers

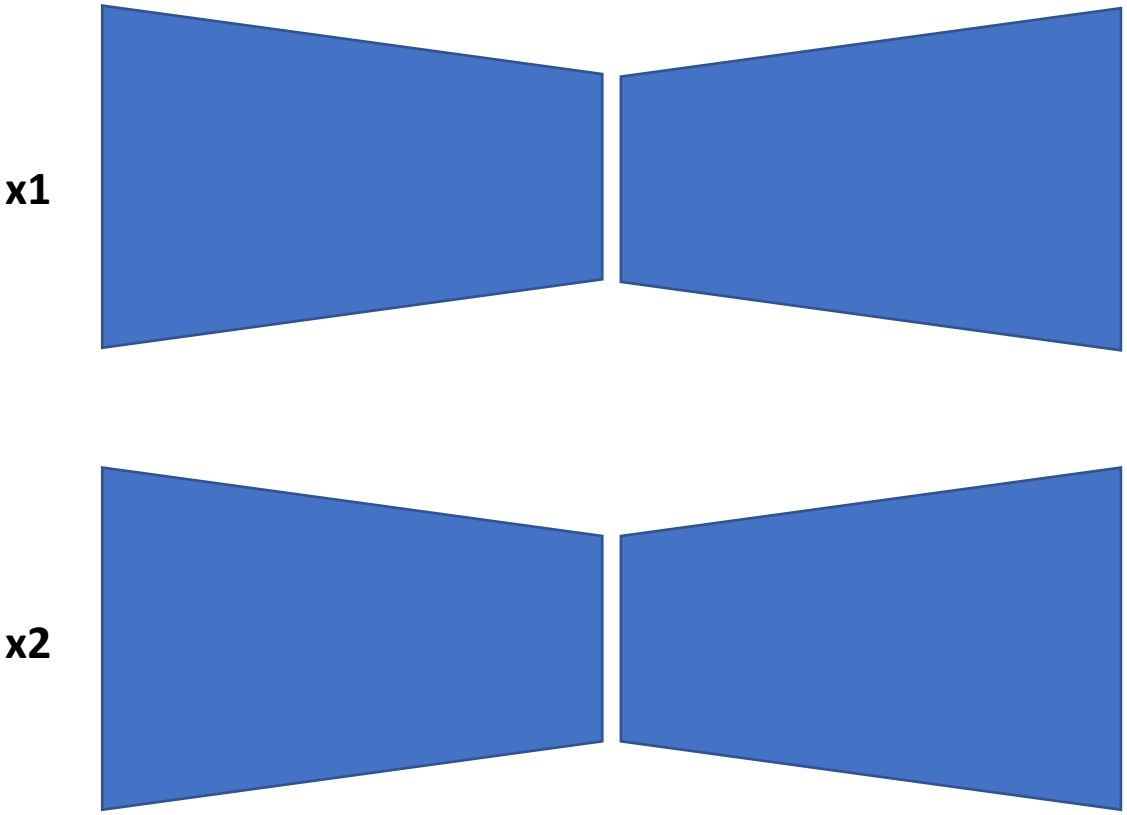
Inference Phase :



- One of the two decoders, which one to use for decoding ?
- Change map (cMap) generated in this way is biased to one image feature pair or the other !

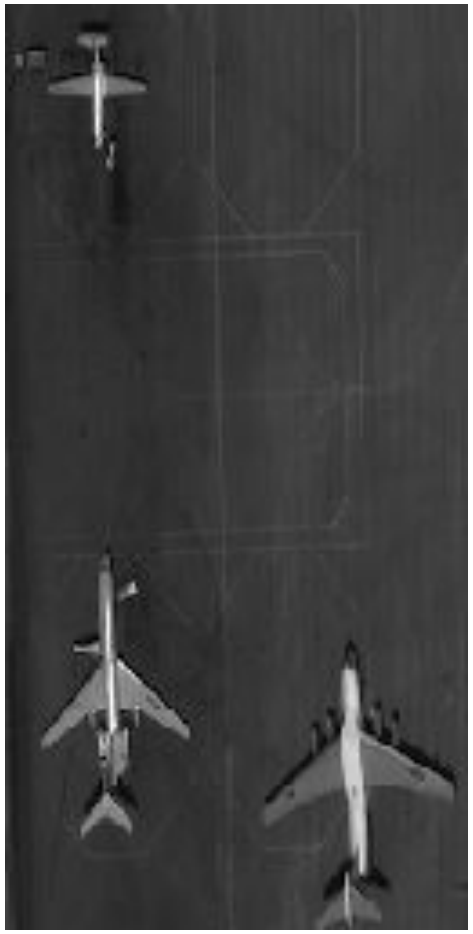
Two Autoencoder Approach to Change Map Generation

Alternative Phase: Inference Approach



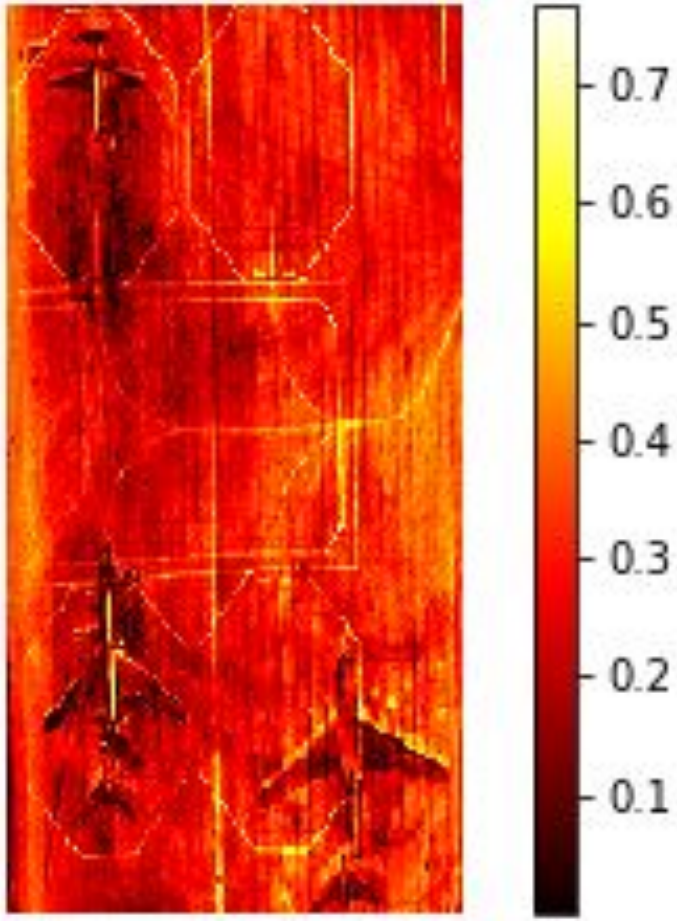
- Change map is derived from the decoded images
- Reduced latent space registers the image pair making them shift invariant relative to each other

$$\mathrm{cMap} = | x1' - x2'$$

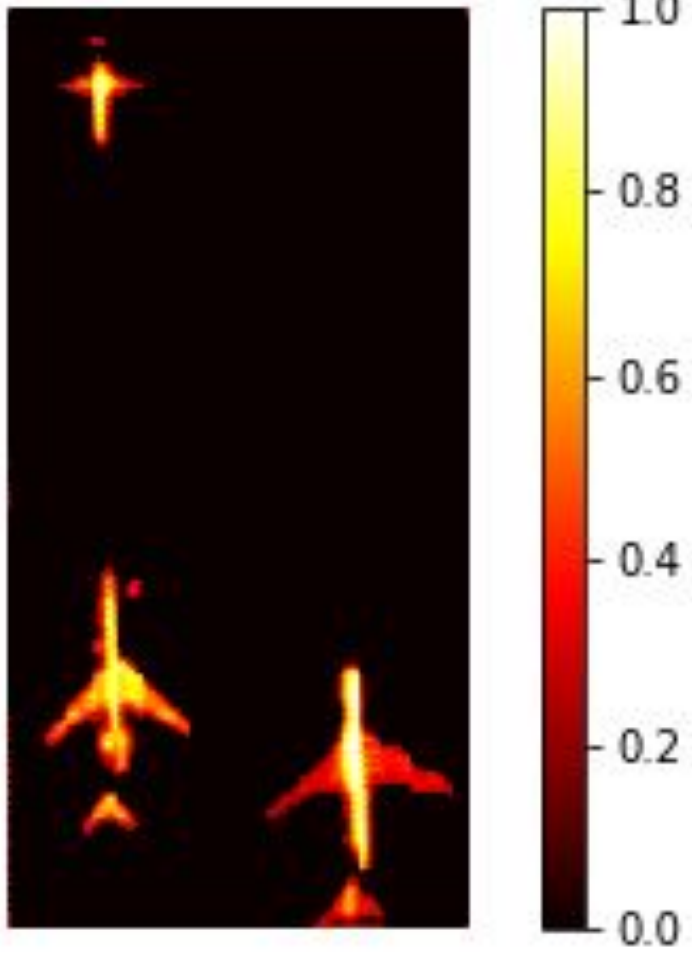


x1

x2



Difference Image

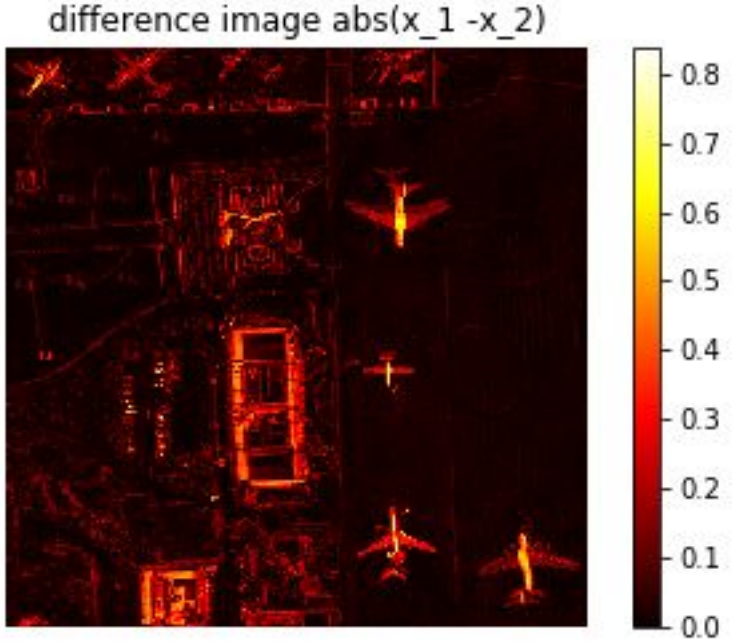


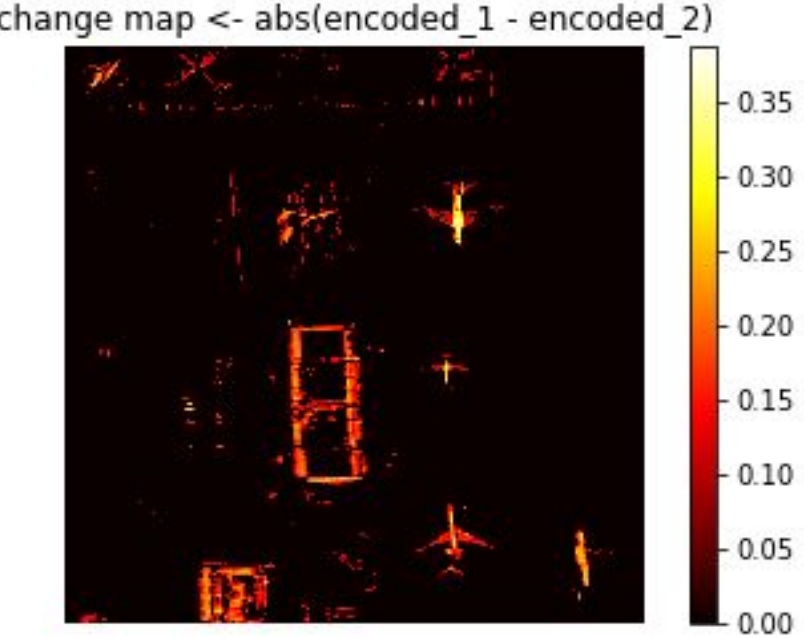
Change map with thresholding

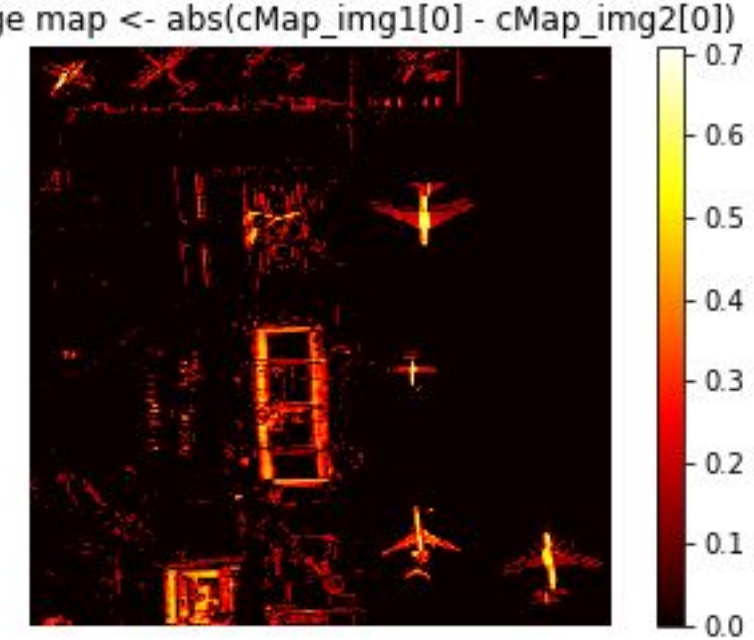
- No Registration issues
- No image pair intensity uniformity issues since x_2 has no objects of interest
- Proposed autoencoder method out performs in quality of change map

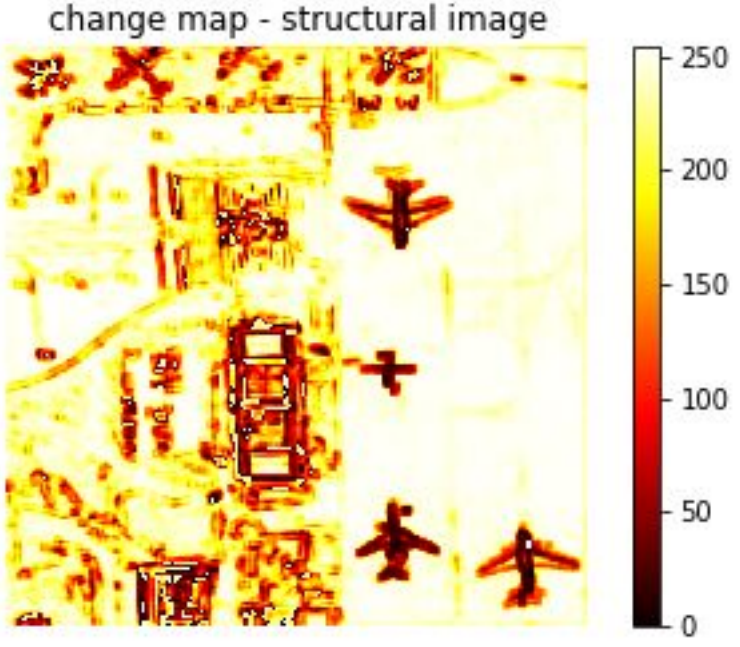
x1

x2

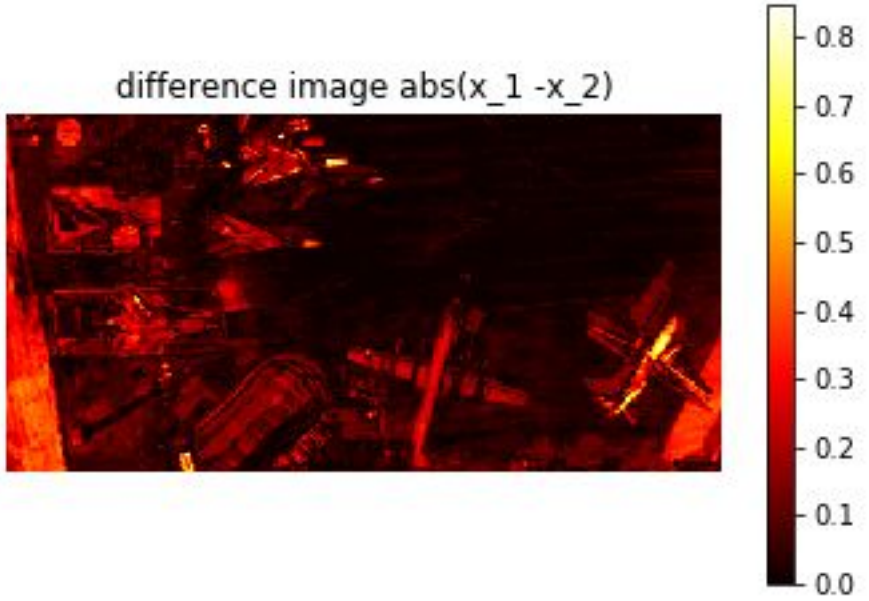

difference image abs(x_1 -x_2)
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0

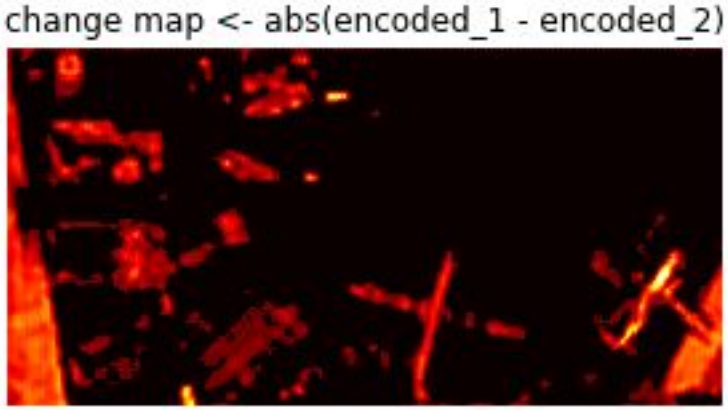

change map <- abs(encoded_1 - encoded_2)
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00

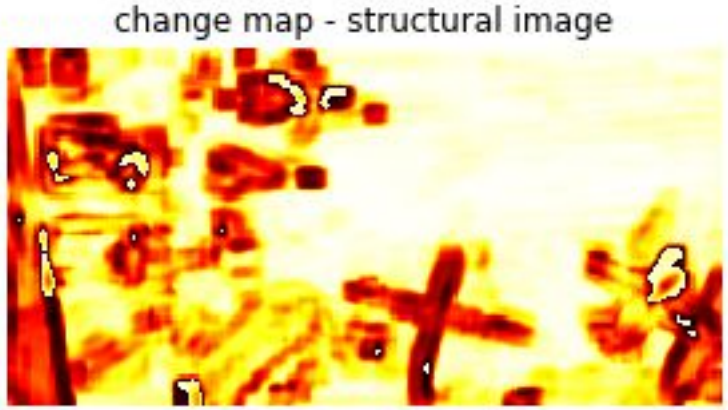

change map <- abs(cMap_img1[0] - cMap_img2[0])
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0

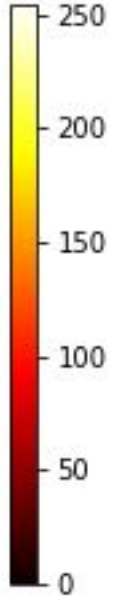

change map - structural image
250
200
150
100
50
0

128 x 256



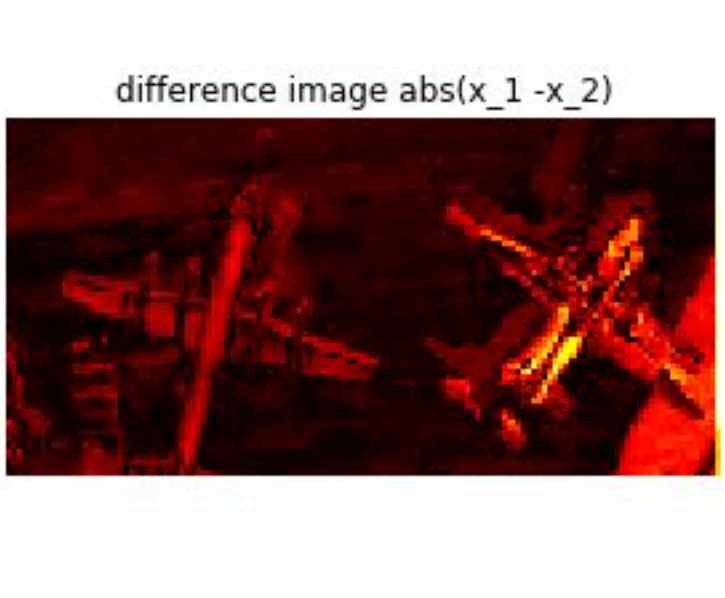


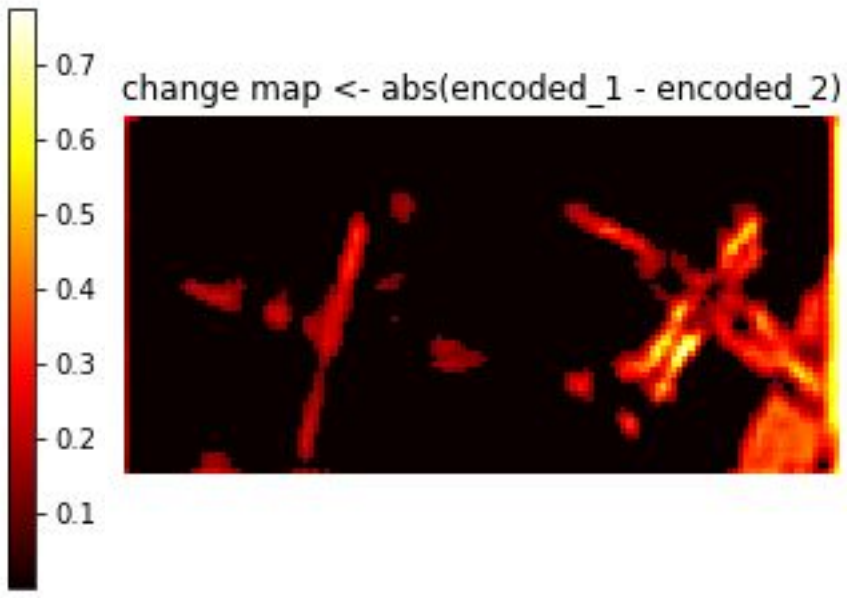


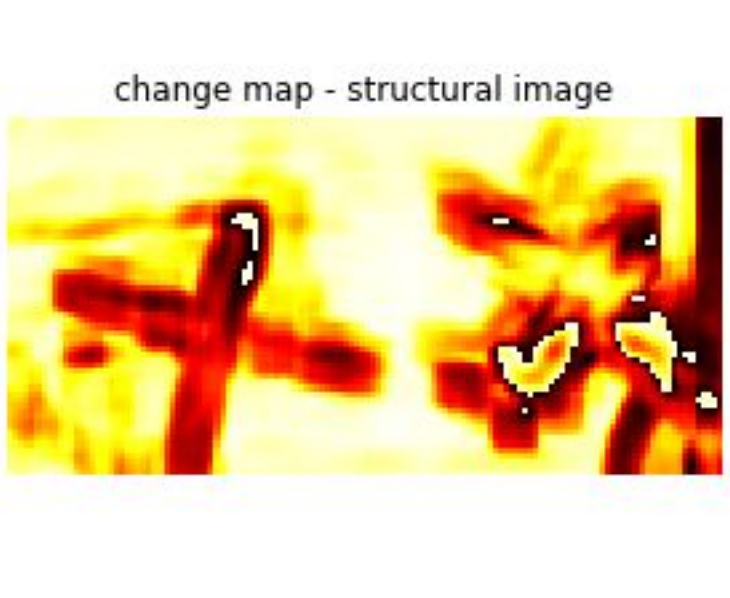


- Reduced image size but we still see fixed structures in the change map such as buildings
- This is mostly from image pair ***registration issues***, ***aspect ratio differences*** and **pixel intensity differences**

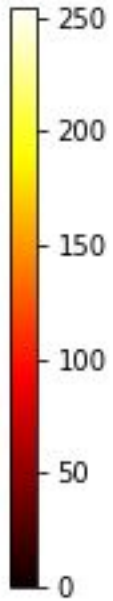
64 x128


difference image abs(x_1 -x_2)


change map <- abs(encoded_1 - encoded_2)


change map - structural image

0.7
0.6
0.5
0.4
0.3
0.2
0.1

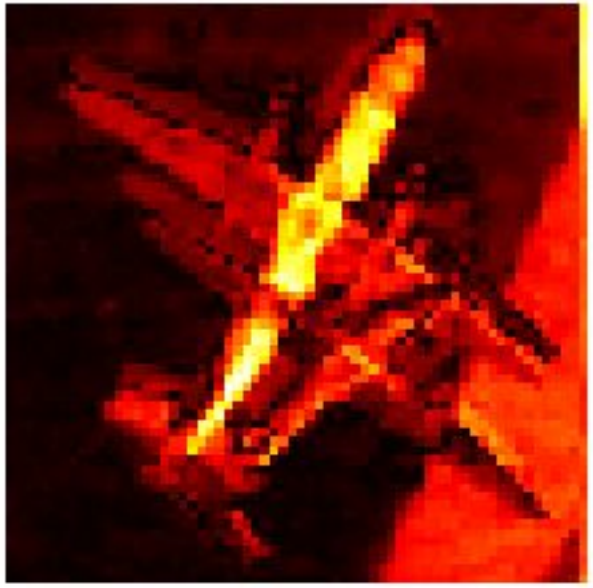

250
200
150
100
50
0

64x64

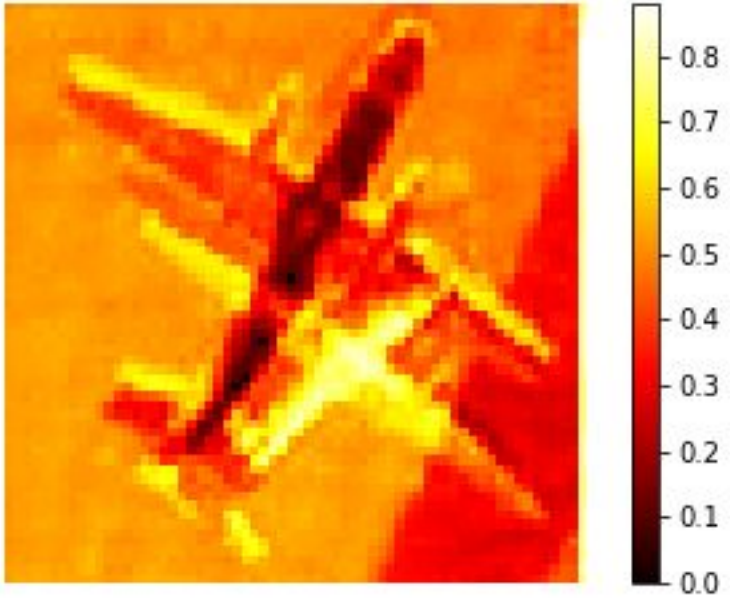
smaller sample:

- Same plane but it has moved
- Different aspect ratios present challenges

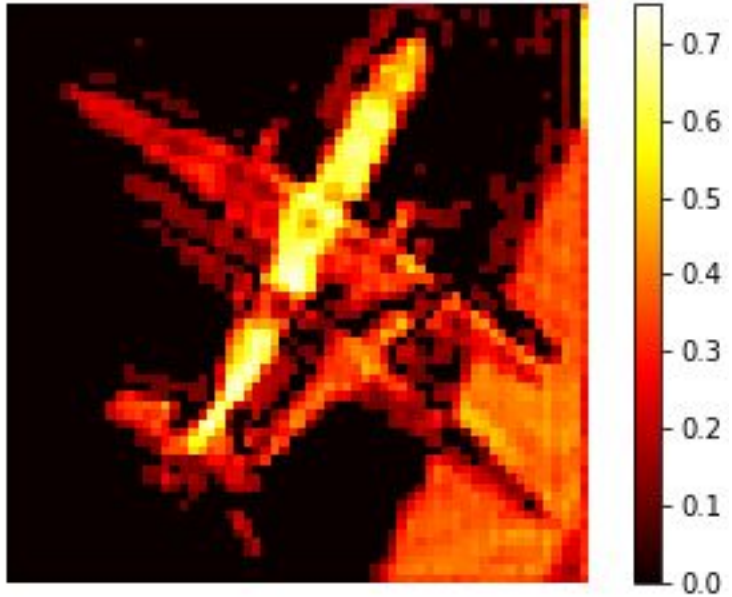
difference image abs(x_1 -x_2)

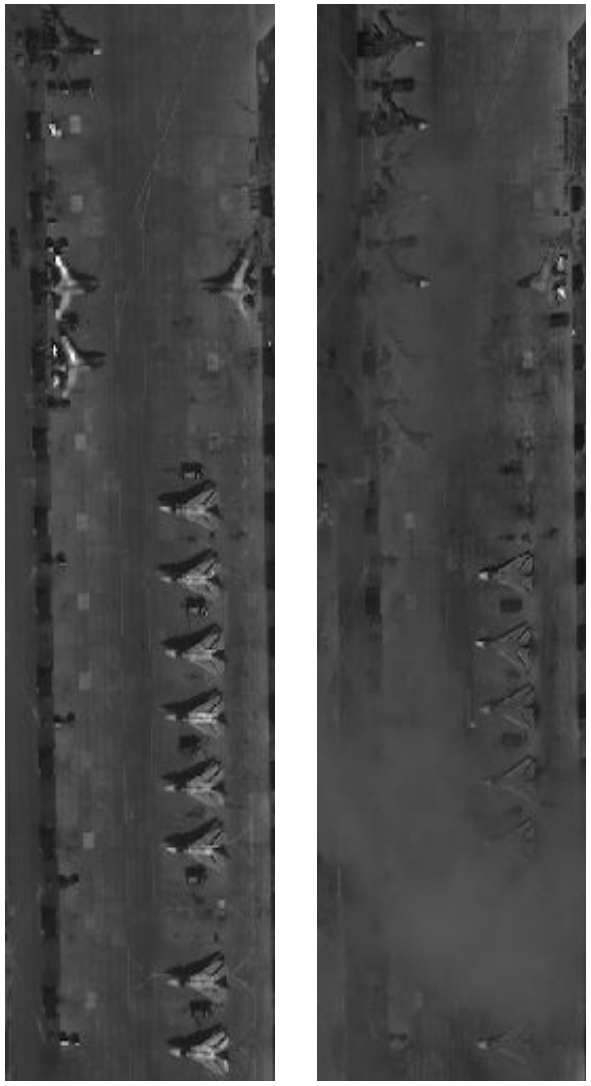


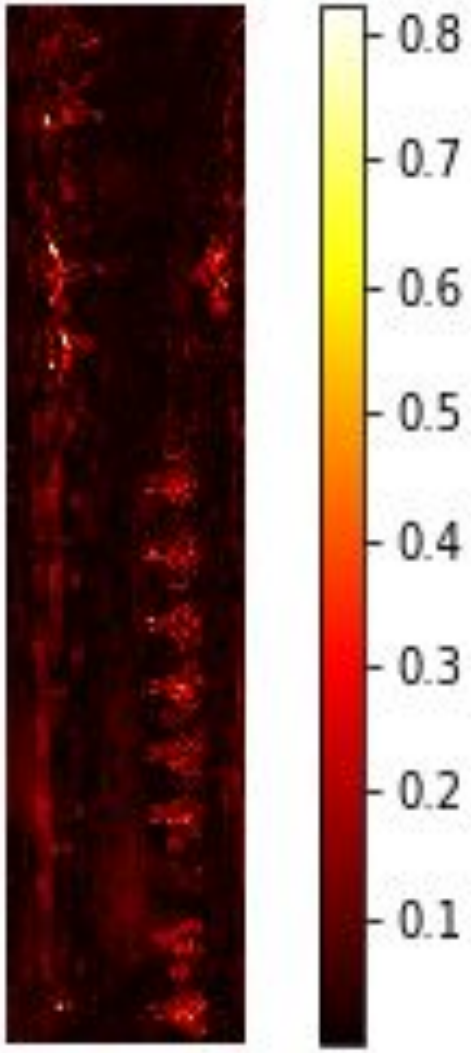
change map <- abs(encoded_1 - encoded_2)

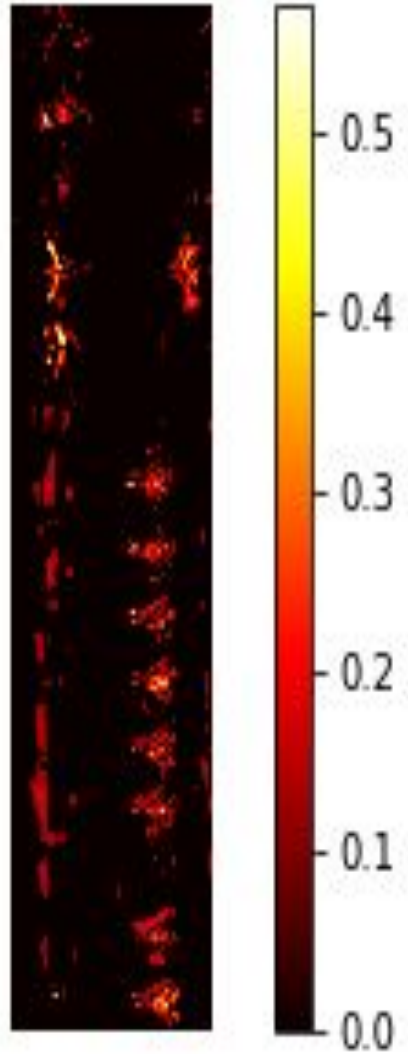


change map <- abs(cMap_img1[0] - cMap_img2[0])





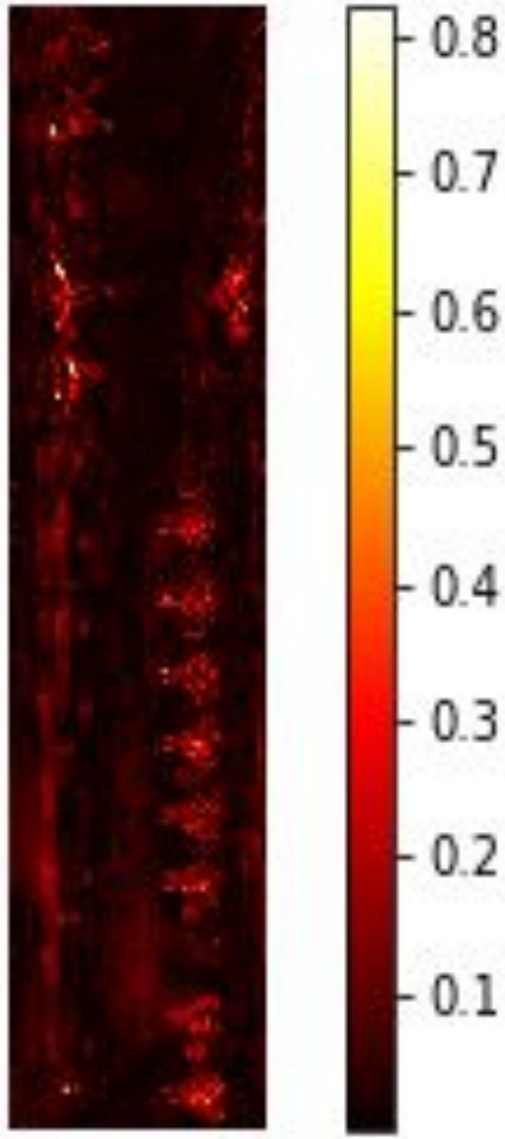




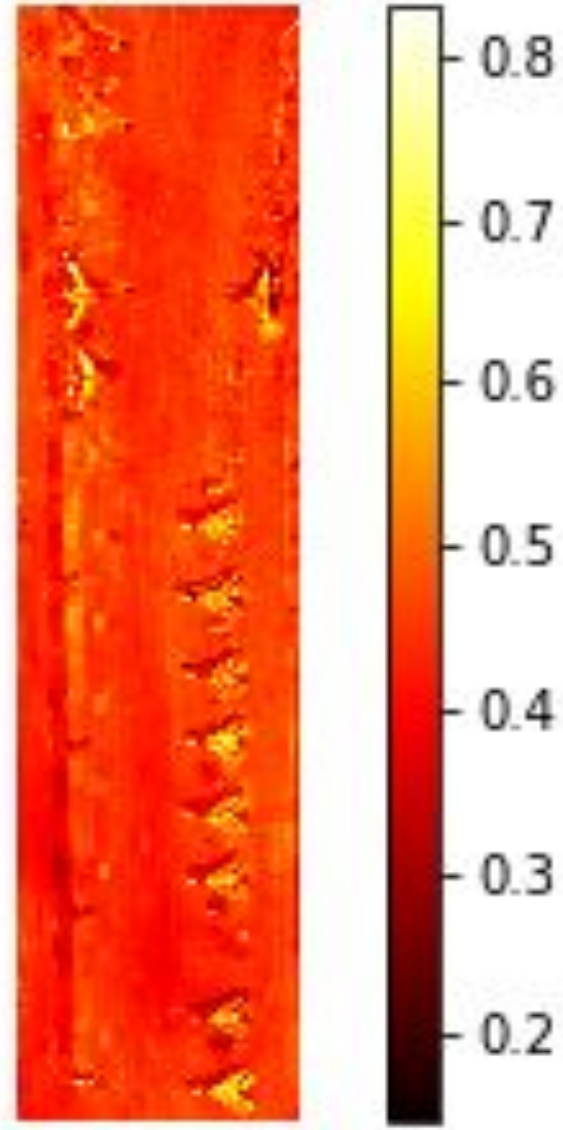
512x128

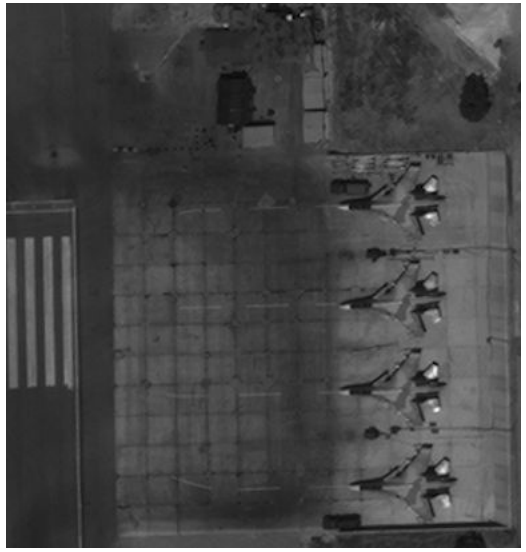
challenges of cloud cover ...

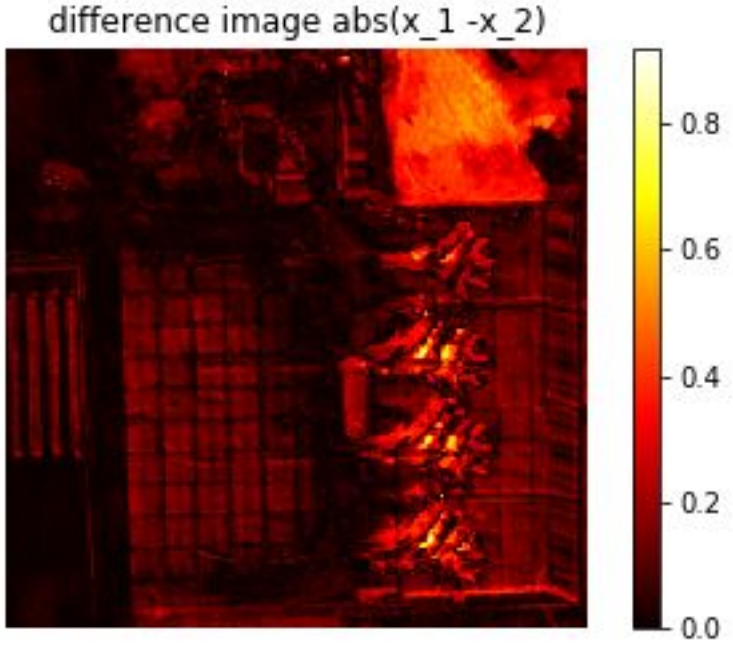
difference image abs(x_1 -x_2)

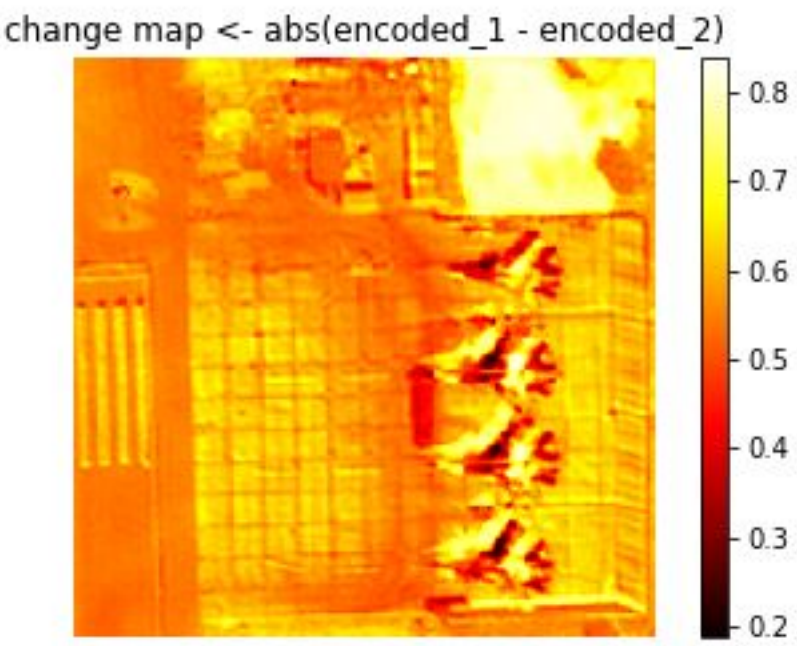


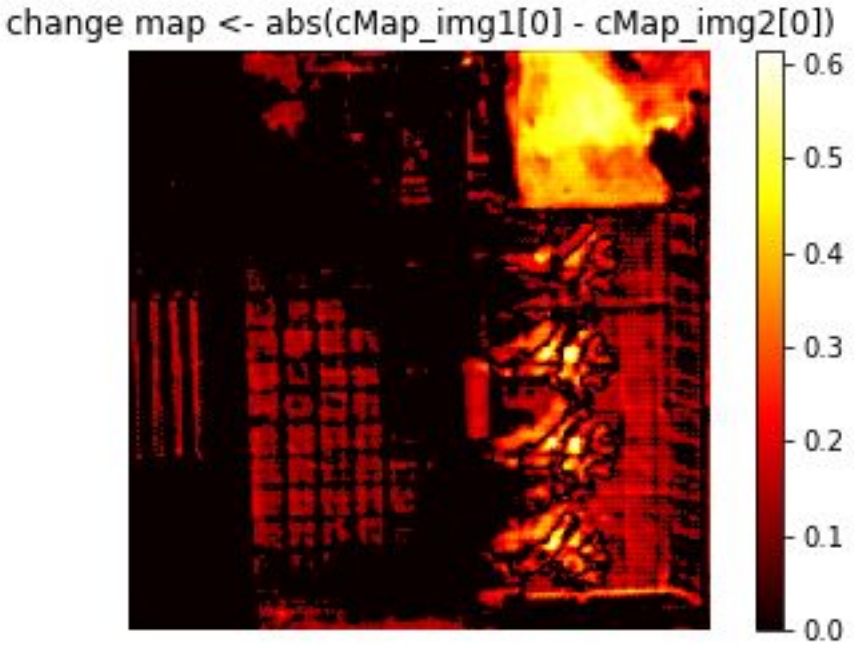
change map <- abs(encoded_1 - encoded_2)






difference image abs(x_1 -x_2)
0.8
0.6
0.4
0.2
0.0


change map <- abs(encoded_1 - encoded_2)
0.8
0.7
0.6
0.5
0.4
0.3
0.2


change map <- abs(cMap_img1[0] - cMap_img2[0])
0.6
0.5
0.4
0.3
0.2
0.1
0.0

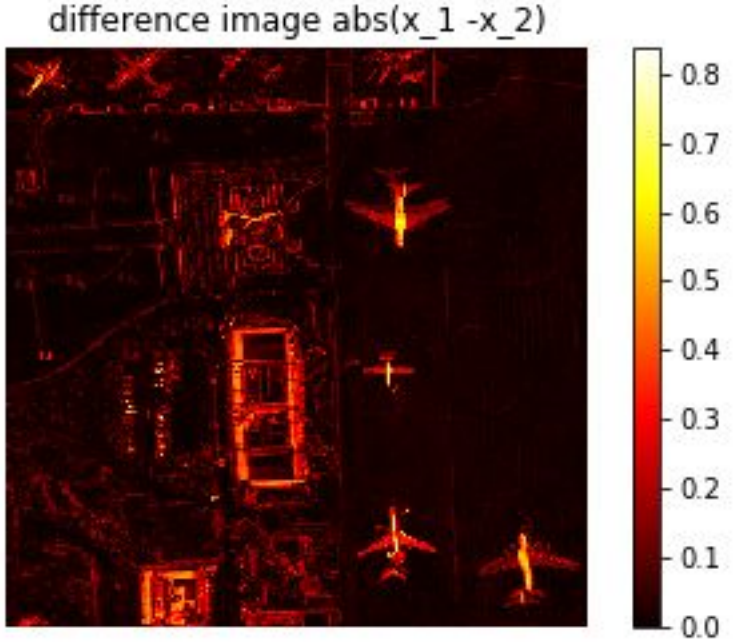
256x256

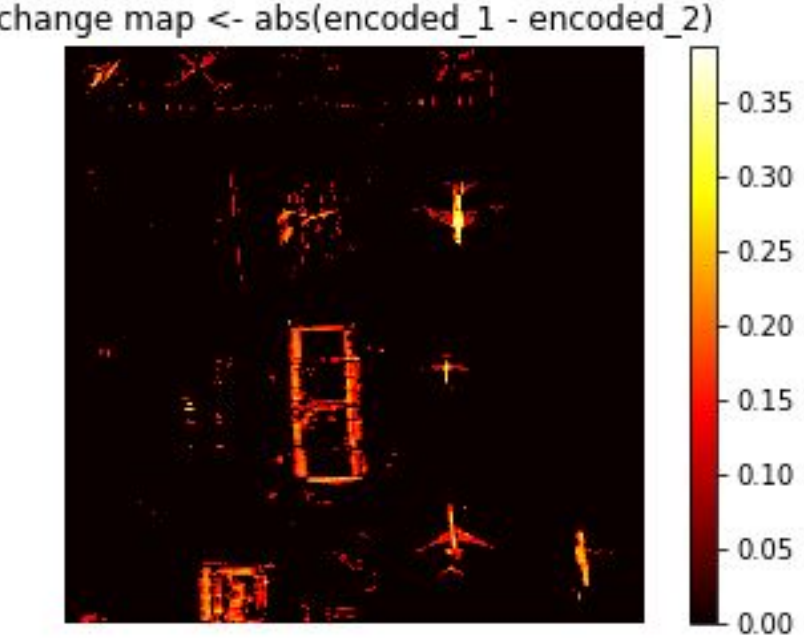
Normalization of Image Pairs

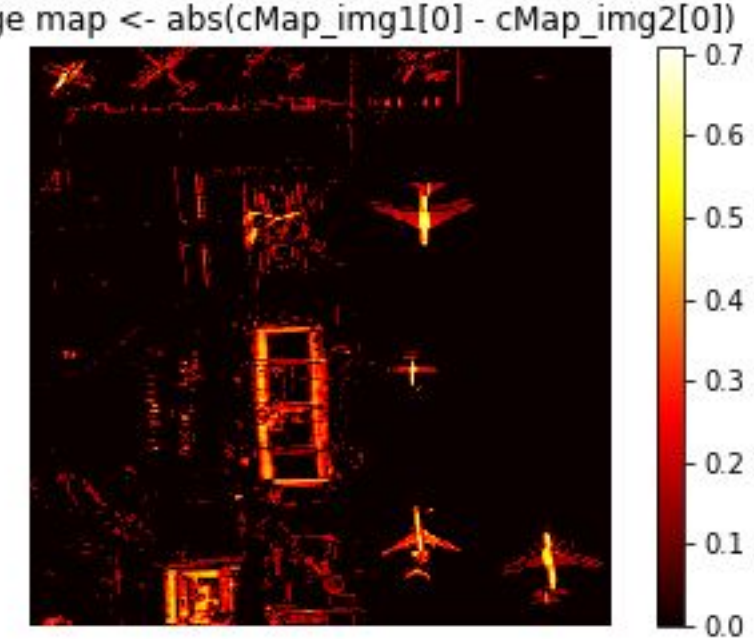
- One strategy to approximately equalize the intensity between the image pairs is to normalize the images pixel intensities to have zero mean and unit variance

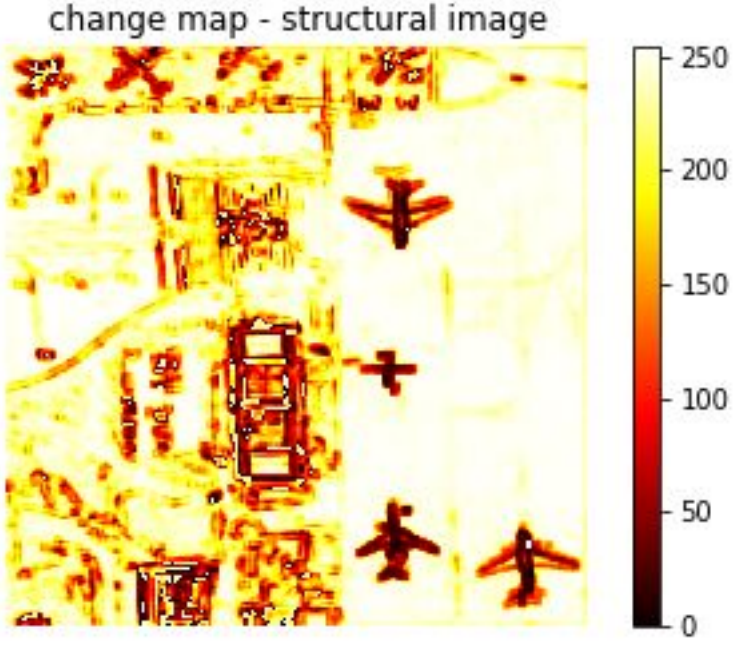
x_1

x_2

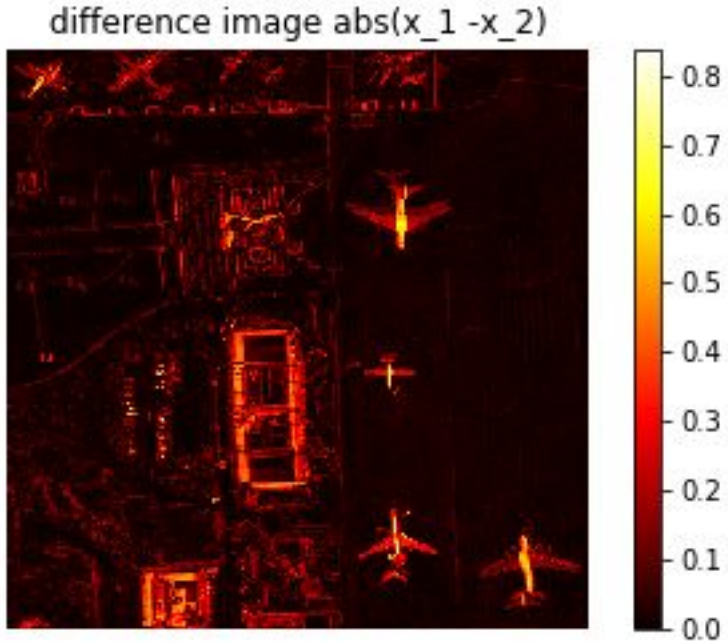

difference image abs(x_1 -x_2)
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0

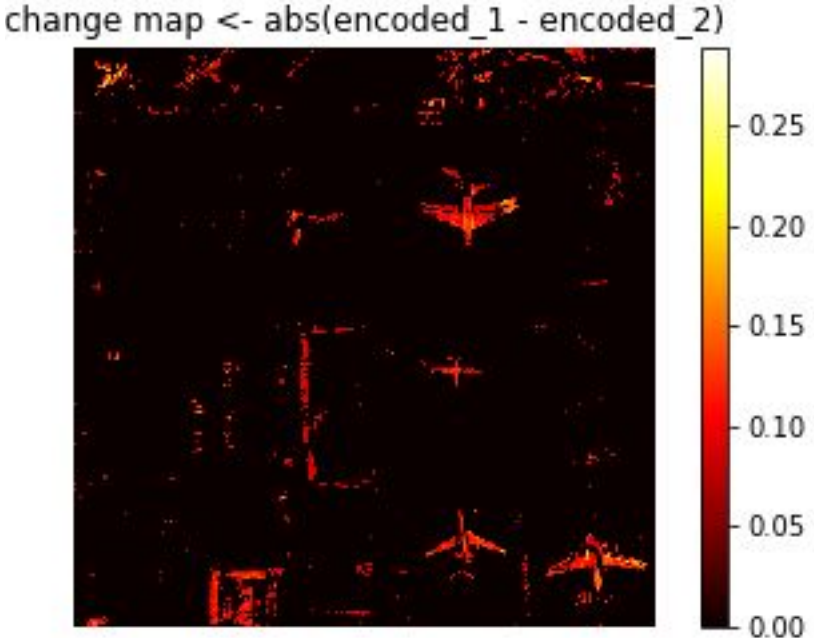

change map <- abs(encoded_1 - encoded_2)
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00

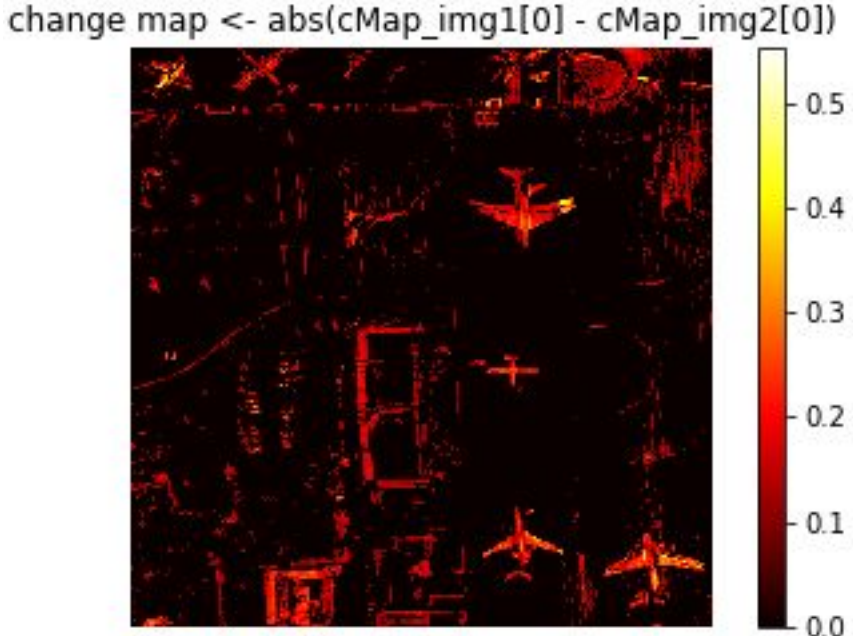

change map <- abs(cMap_img1[0] - cMap_img2[0])
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0

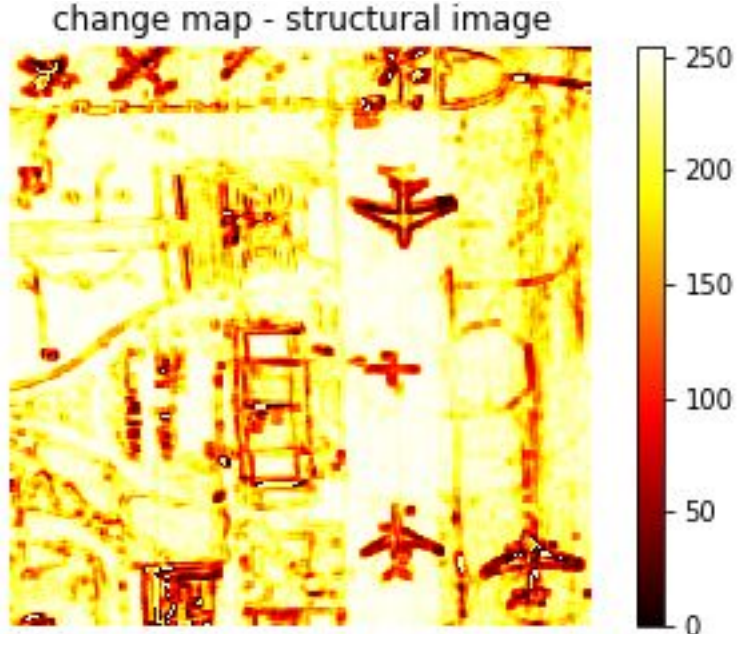

change map - structural image
250
200
150
100
50
0

Normalization Images (mean = 0, var = 1)





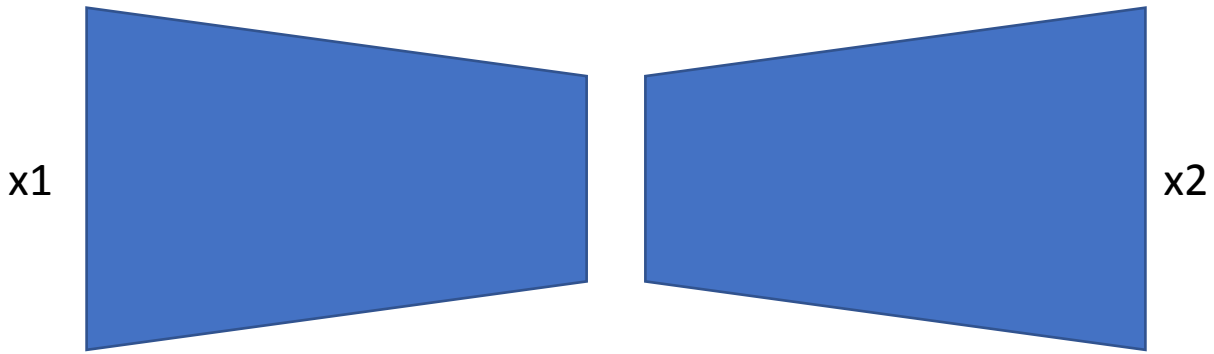




Single Autoencoder with Batch Normalization

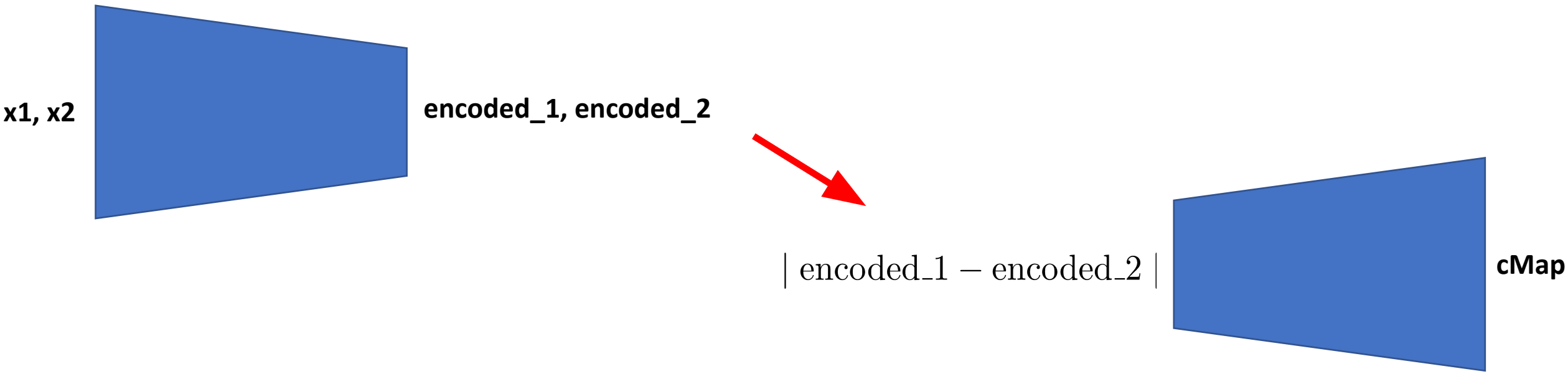
Alternative Extension with Encoding Common Features

X1 and x2 put in a batch to train the same autoencoder



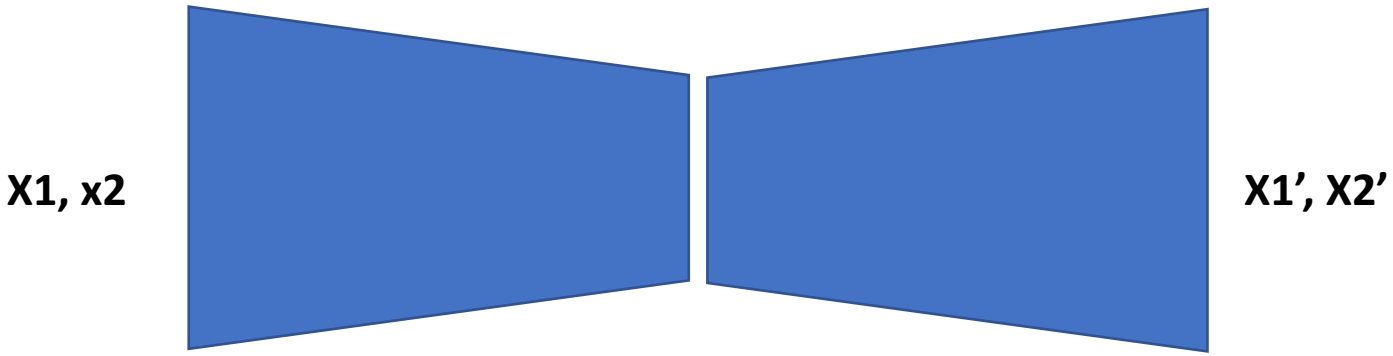
Change Map from Difference in Latent Layers

Inference Phase:



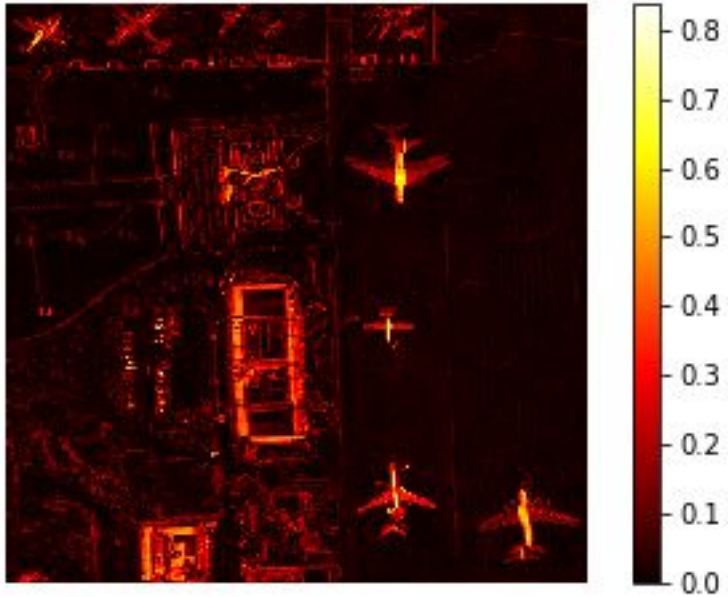
Change Map from Difference in Decoded Images

Inference Phase:

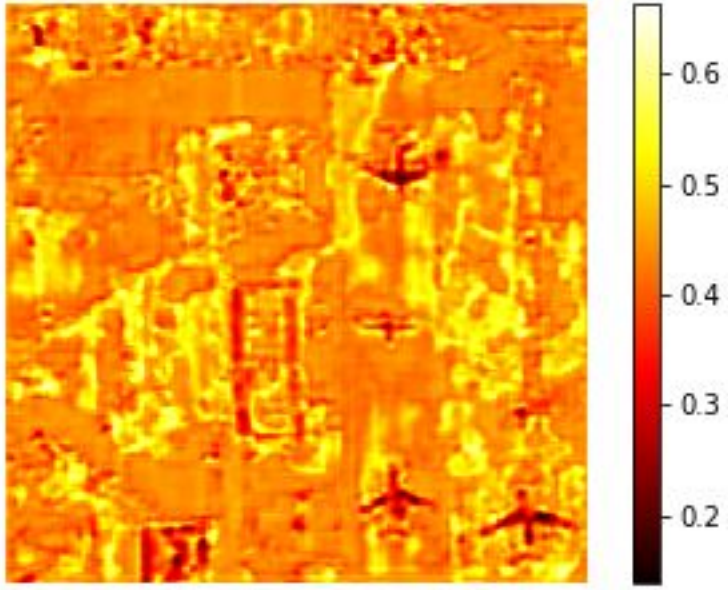


$$\mathrm{cMap} = | \; x1' - x2'$$

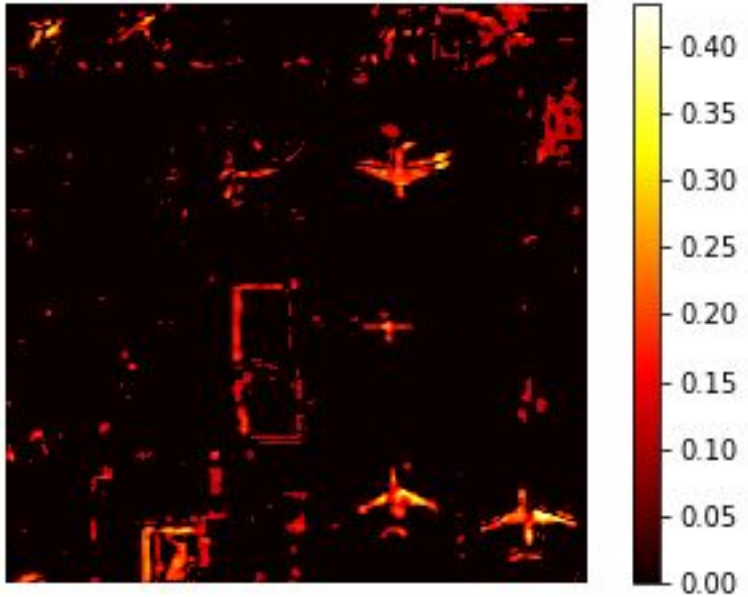
difference image abs(x_1 -x_2)



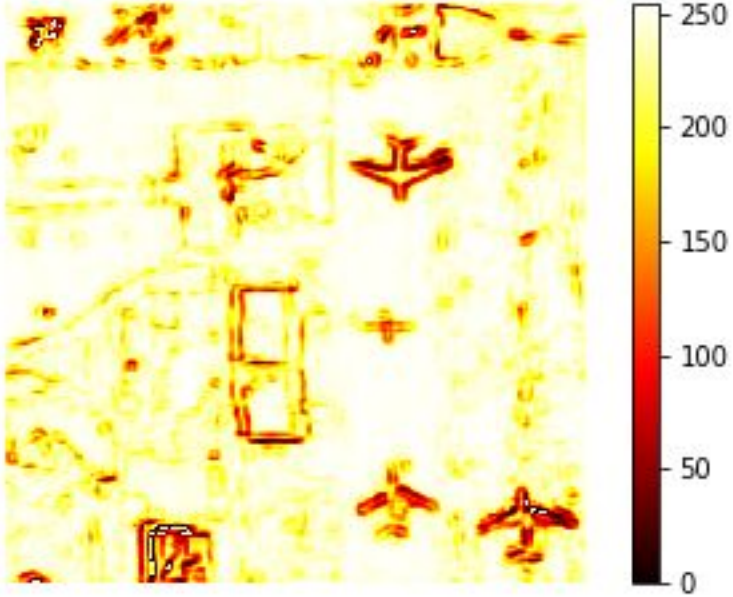
change map <- abs(encoded_1 - encoded_2)



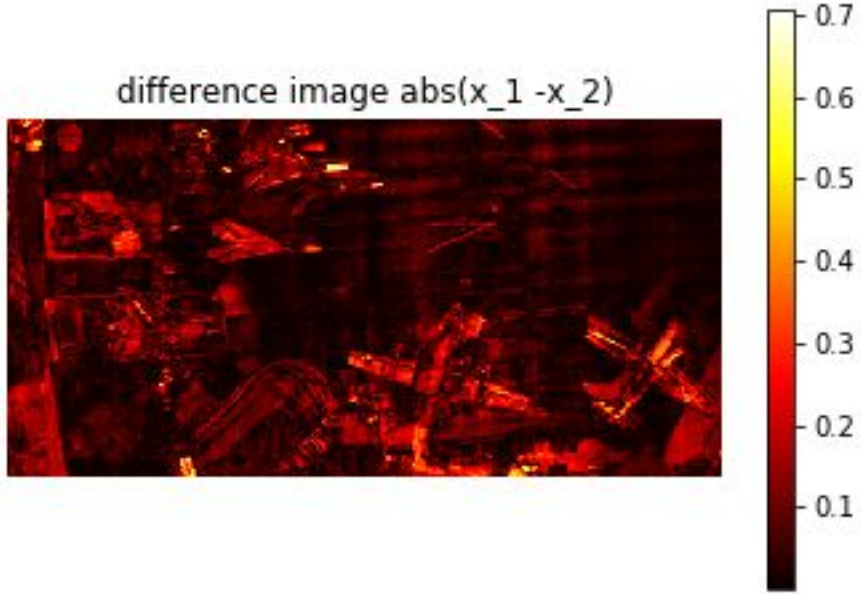
change map <- abs(cMap_img1[0] - cMap_img2[0])

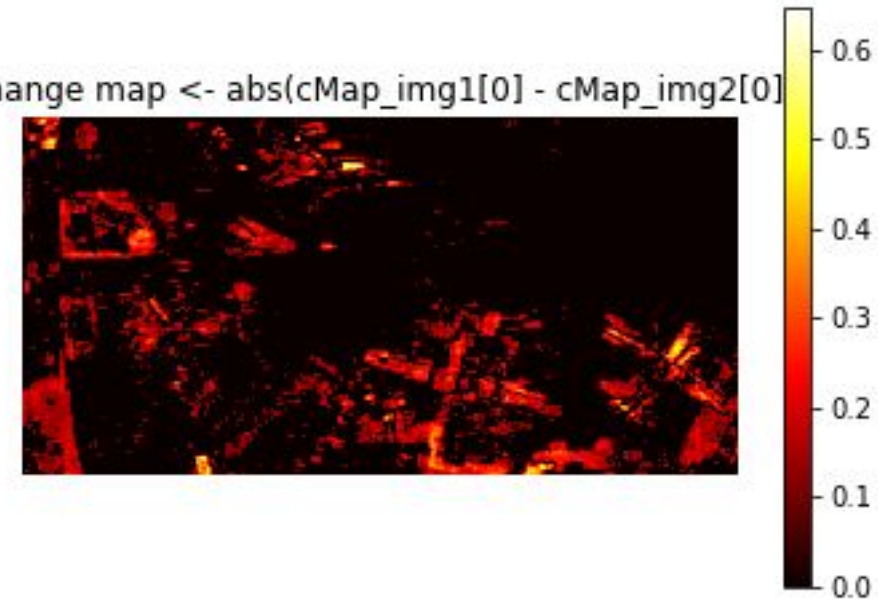


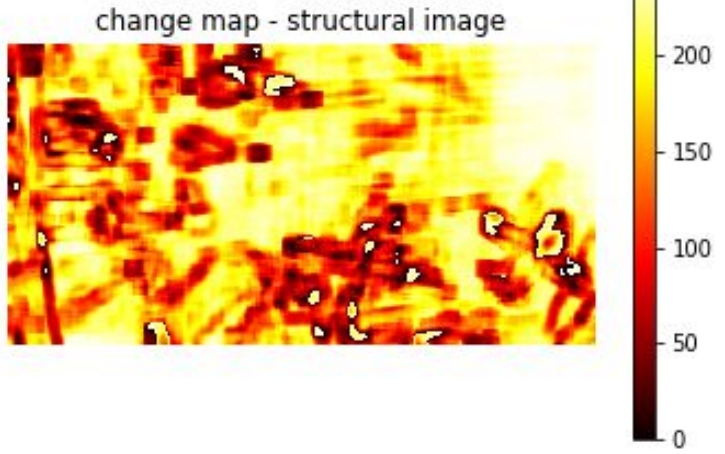
change map - structural image



128x256







- Improved suppression of background and common structures such as buildings
- Approach could benefit from extensions proposed on the next slide

Recommendation/Ways Forward

- Investigate ways of ***normalizing pixel intensity*** between image pairs
  - Intensity of each pixel derived in terms of the min/max from the pair.

- Use of ***Transfer Learning*** in Autoencoders to Reduce Training Time for New Image Pairs

- Defining ***Specific Use Cases*** for Change Map Generation

- Dealing with Image Pairs with Same Objects but different ***Aspect Ratios***

Code Repositories

Change Map Autoencoder Types

1. **Deep Autoencoder**: 2x 4 Conv layer deep autoencoders, for encoding pair of images for change detection.

2. **Variational Autoencoder**: Uses two variational autoencoders to encode the image pair separately and to generate the change map

3. **Single Batch Autoencoder**: Encodes the image pair to compare using the same encoder and generates the change map from the same encoder.

Testing the Code

- To run the code, I recommend that you use a visual development environment such as spyder or pycharm to display the results graphically at the end of the run. I have included unit test data in both repositories for first run validation of your development environment. In all cases, look for a section of code given below to modify for runs on custom data:

```
def parse_cmd_line():
    p = argparse.ArgumentParser(description='change detector')
    p.set_defaults(verbose=0, quiet=False)
    p.add_argument('-d', '--dir', help='image directory', default='../small_pairs_tiles/')
    p.add_argument('-f', '--filename', help='filename string of image pair', default='c')
    p.add_argument('-e', '--epochs', help='number of epochs', default=400)
    args = p.parse_args()
    return args
```

https://www.dropbox.com/sh/3qafzwqadez3pb6/AABlqZJj2j5KImeTXbI9RJUVa?dl=0

ADD COMMENTS ON SIZE OF IMAGES ???

ADD INPUT OUTPUT DETAILS IN README