Muographic Image Upsampling with Machine Learning for Built Infrastructure Applications

IOP Nuclear Physics Conference 2025

William O’Donnell, David Mahon, Guangliang Yang, Simon Gardner

1. MOTIVATION

University of Glasgow

PROBLEM: Non-Destructive Testing of Built Infrastructure

- It has been widely established that there is a growing amount of aged, concrete infrastructure coming to end of life.

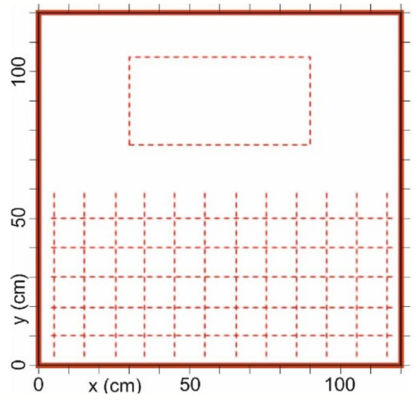
- However, current NDT techniques are limited in establishing high quality reconstructions of concrete interiors.

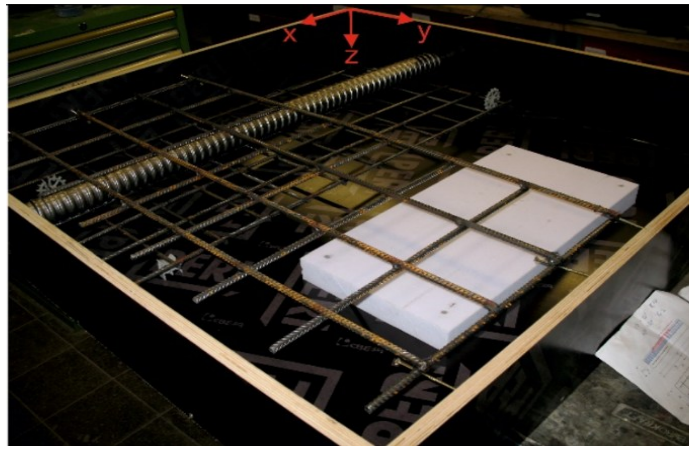
- A 2019 [1] study tested and compared NDT techniques:
  - X-Ray laminography
  - Ground penetrating radar (GPR)
  - Ultrasound
  - **Muography**

[1] Journal of Nondestructive Evaluation (2021) 40:65 https://doi.org/10.1007/s10921-021-00797-3

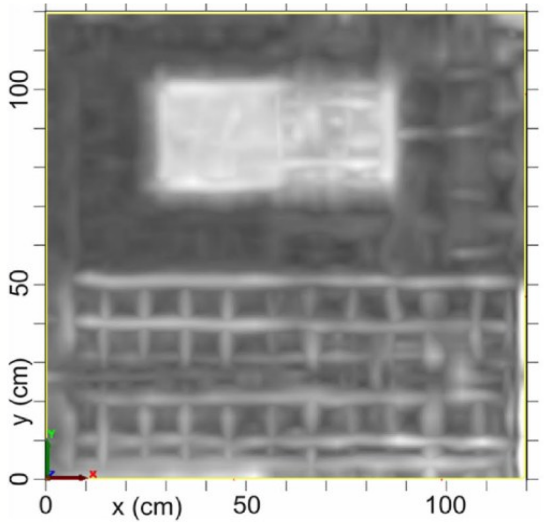
PROBLEM: Non-Destructive Testing of Built Infrastructure

Reference, z = 17cm

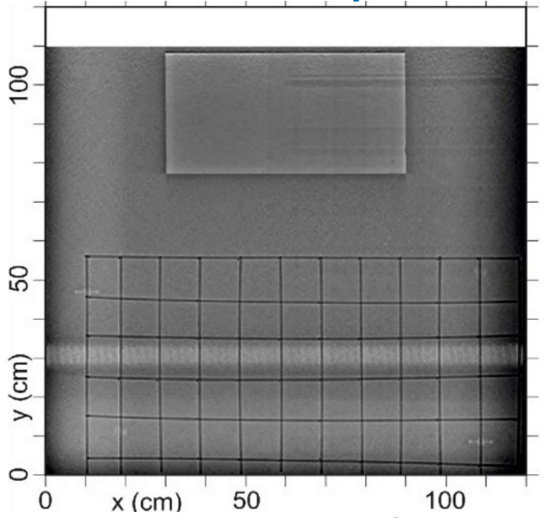




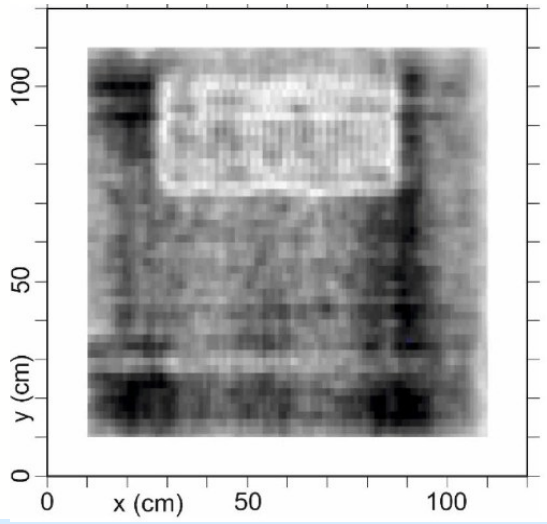
GPR



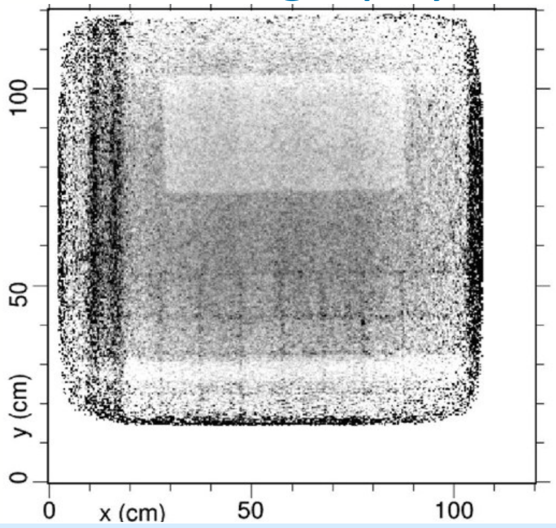
X-Ray



Ultrasound



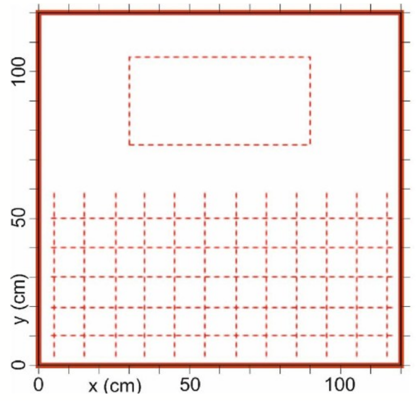
Muography

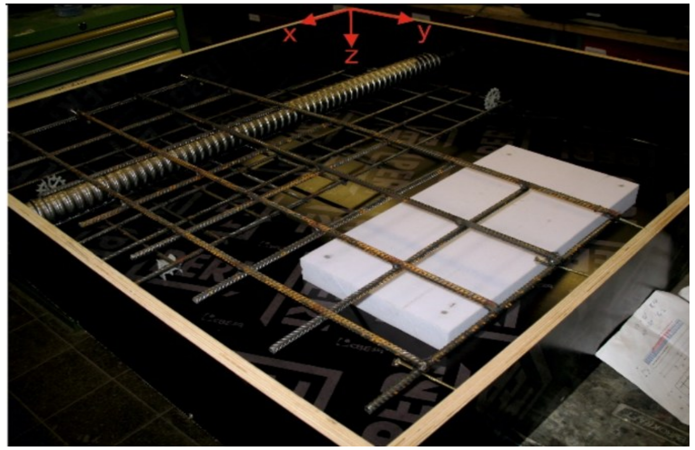


[1] Journal of Nondestructive Evaluation (2021) 40:65
https://doi.org/10.1007/s10921-021-00797-3

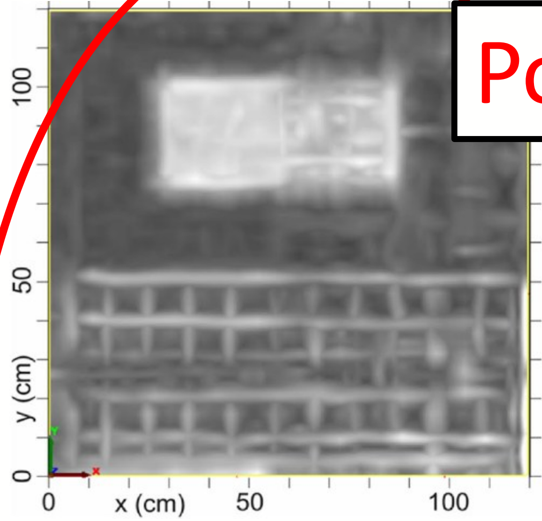
PROBLEM: Non-Destructive Testing of Built Infrastructure

Reference, z = 17cm

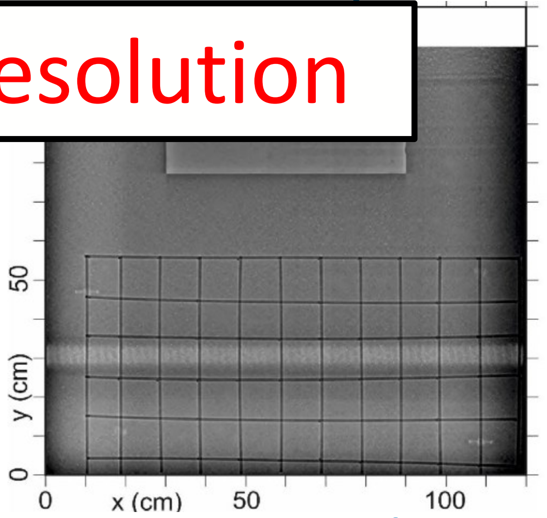




GPR

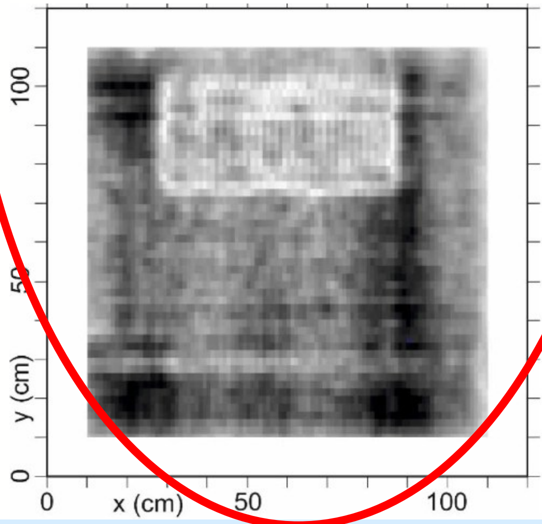


X-Ray

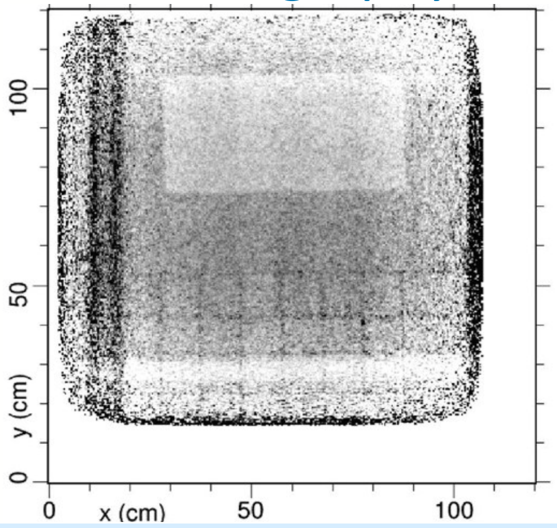


Poor Resolution

Ultrasound



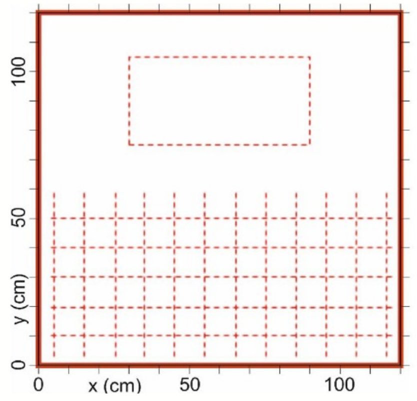
Muography

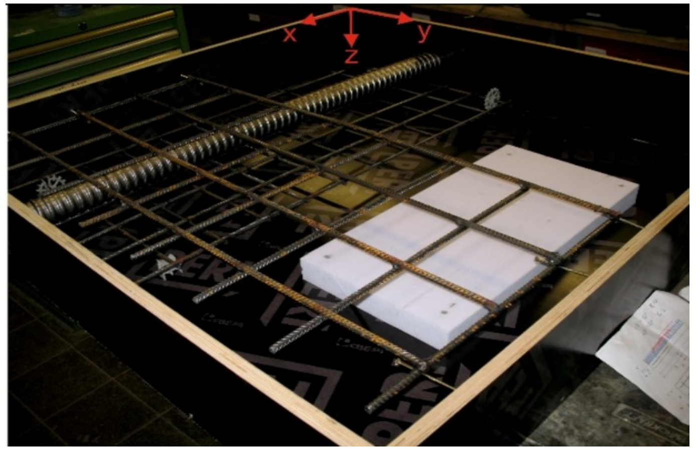


[1] Journal of Nondestructive Evaluation (2021) 40:65
https://doi.org/10.1007/s10921-021-00797-3

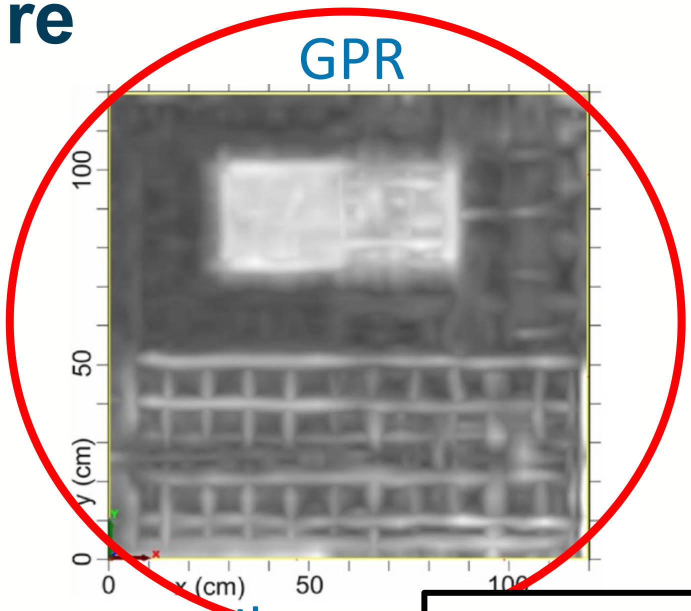
PROBLEM: Non-Destructive Testing of Built Infrastructure

Reference, z = 17cm

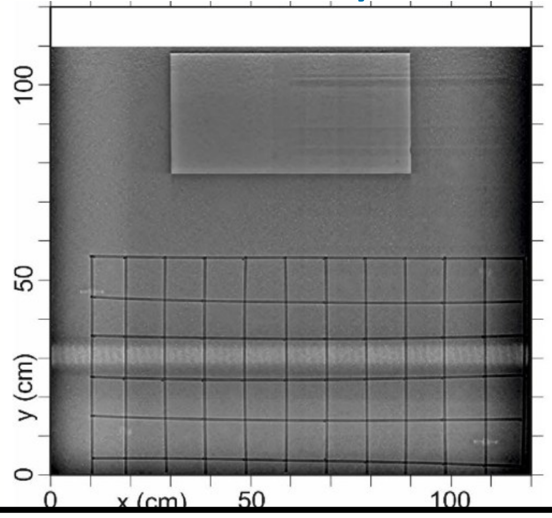




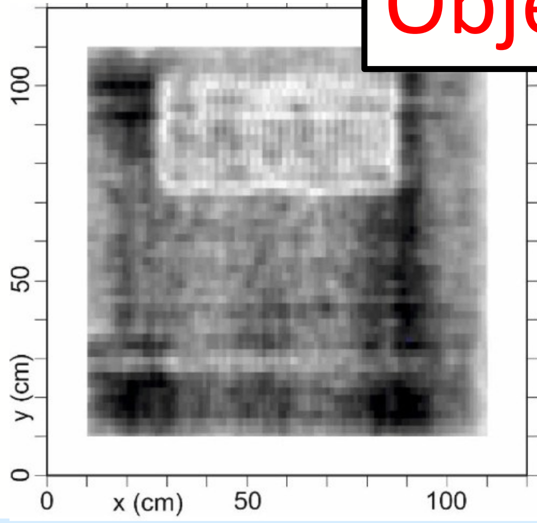
GPR

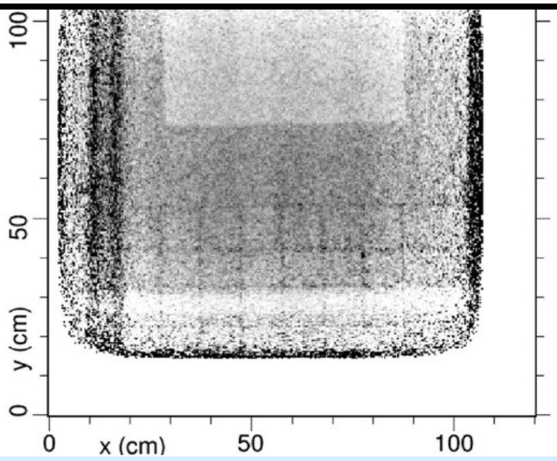


X-Ray



Ultrasou



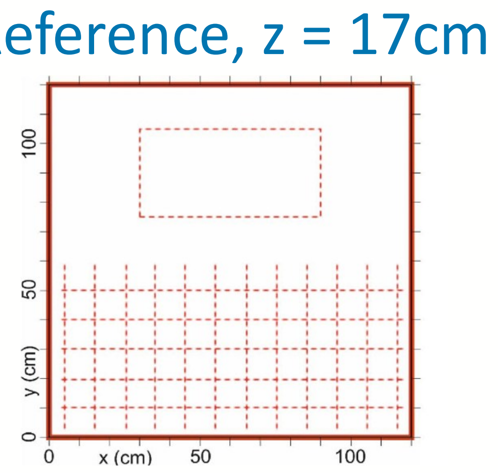


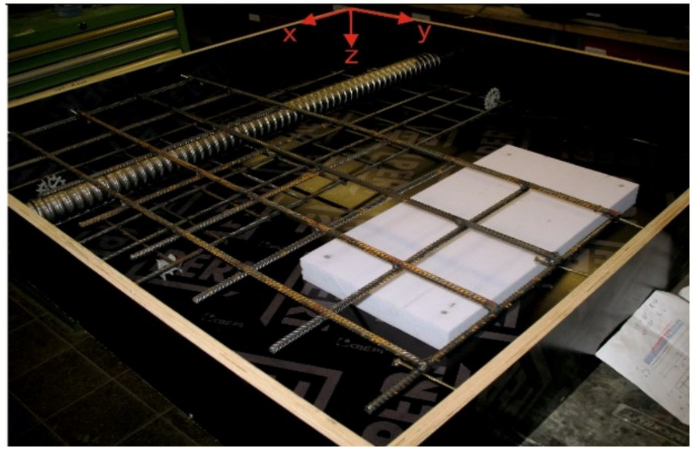
Object sizes misrepresented

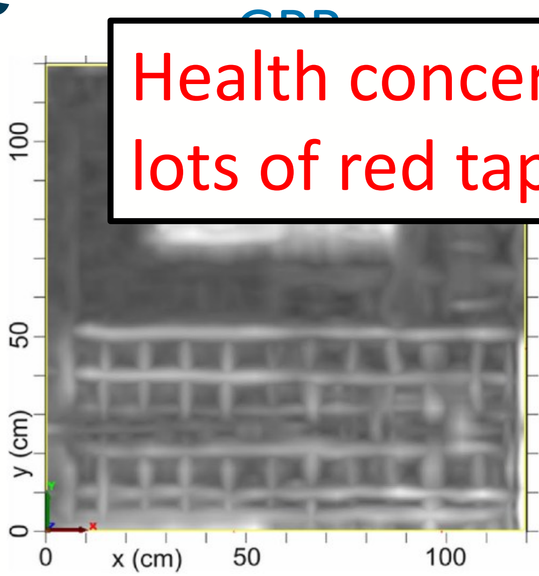
[1] Journal of Nondestructive Evaluation (2021) 40:65
https://doi.org/10.1007/s10921-021-00797-3

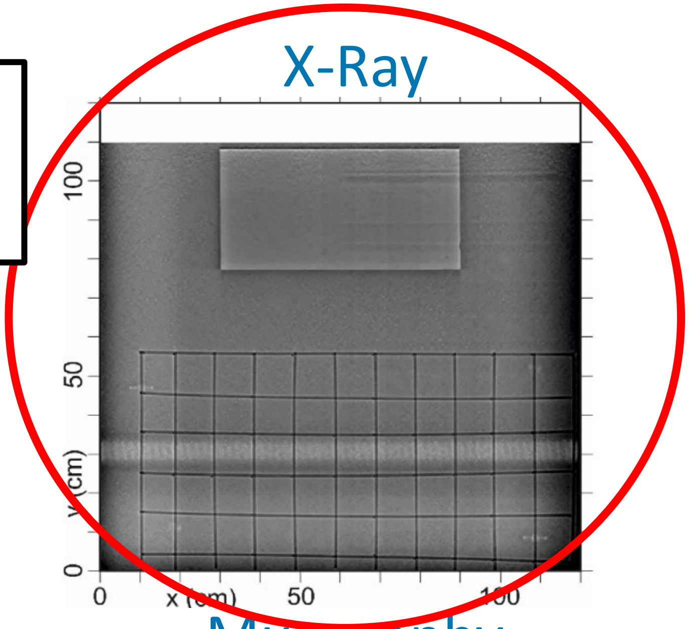
PROBLEM: Non-Destructive Testing of Built Infrastructure

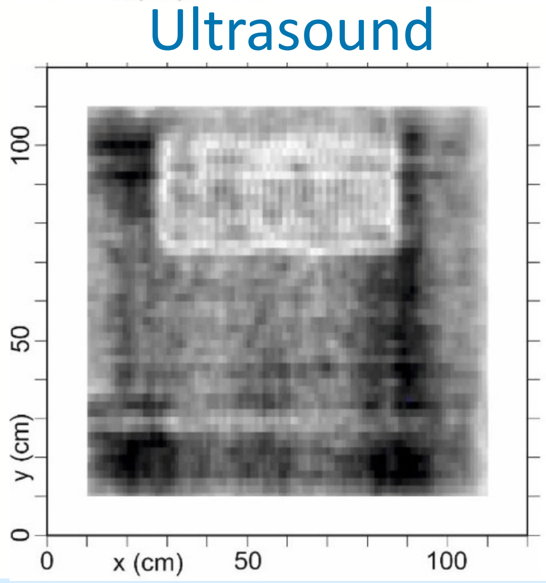
Health concerns: lots of red tape

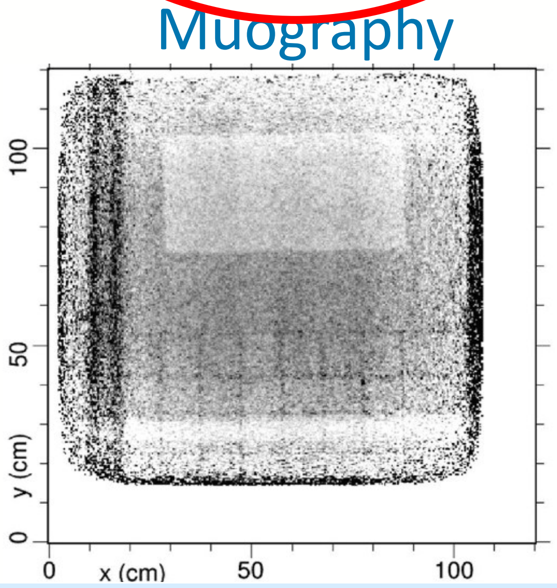








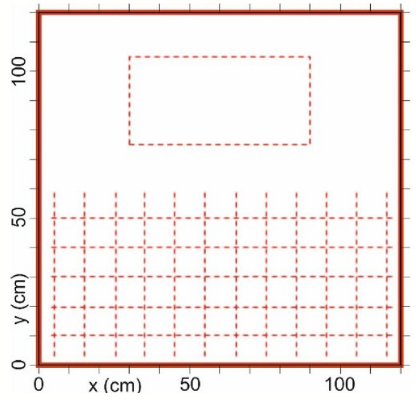


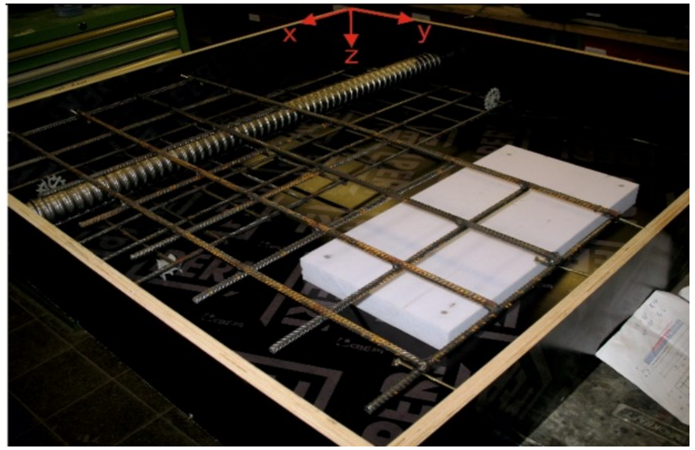


[1] Journal of Nondestructive Evaluation (2021) 40:65
https://doi.org/10.1007/s10921-021-00797-3

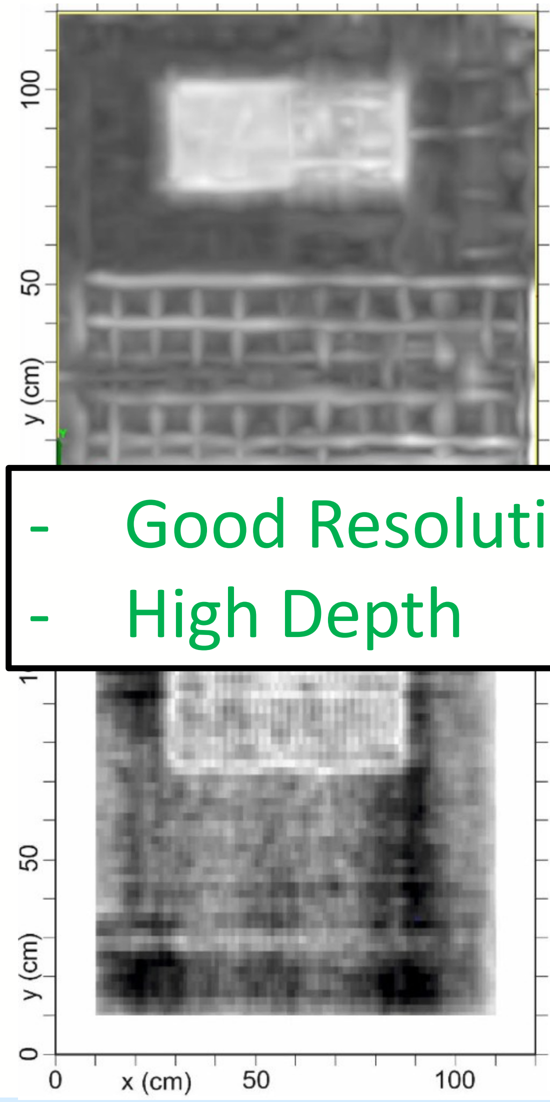
PROBLEM: Non-Destructive Testing of Built Infrastructure

Reference, z = 17cm

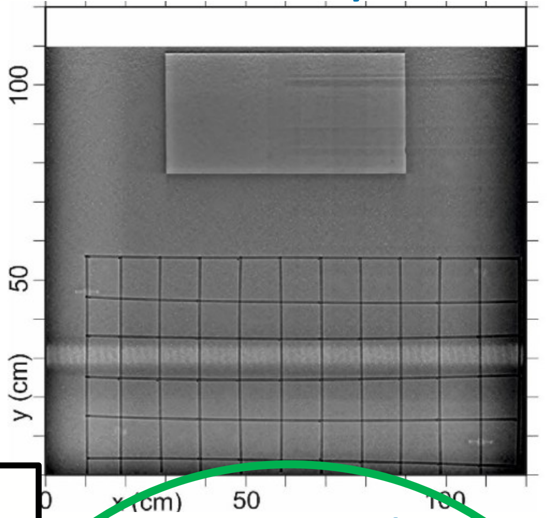




GPR

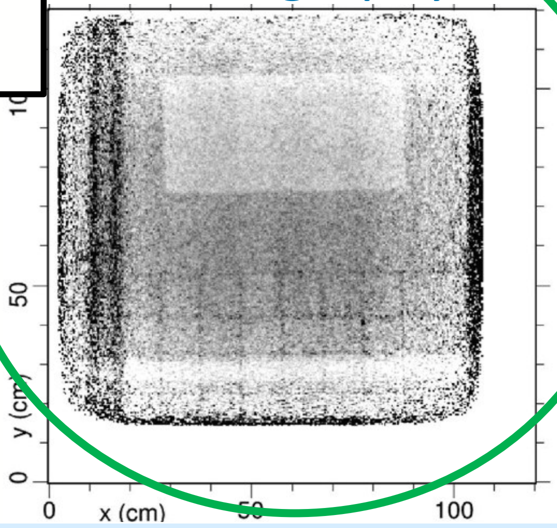


X-Ray



- Good Resolution
- High Depth

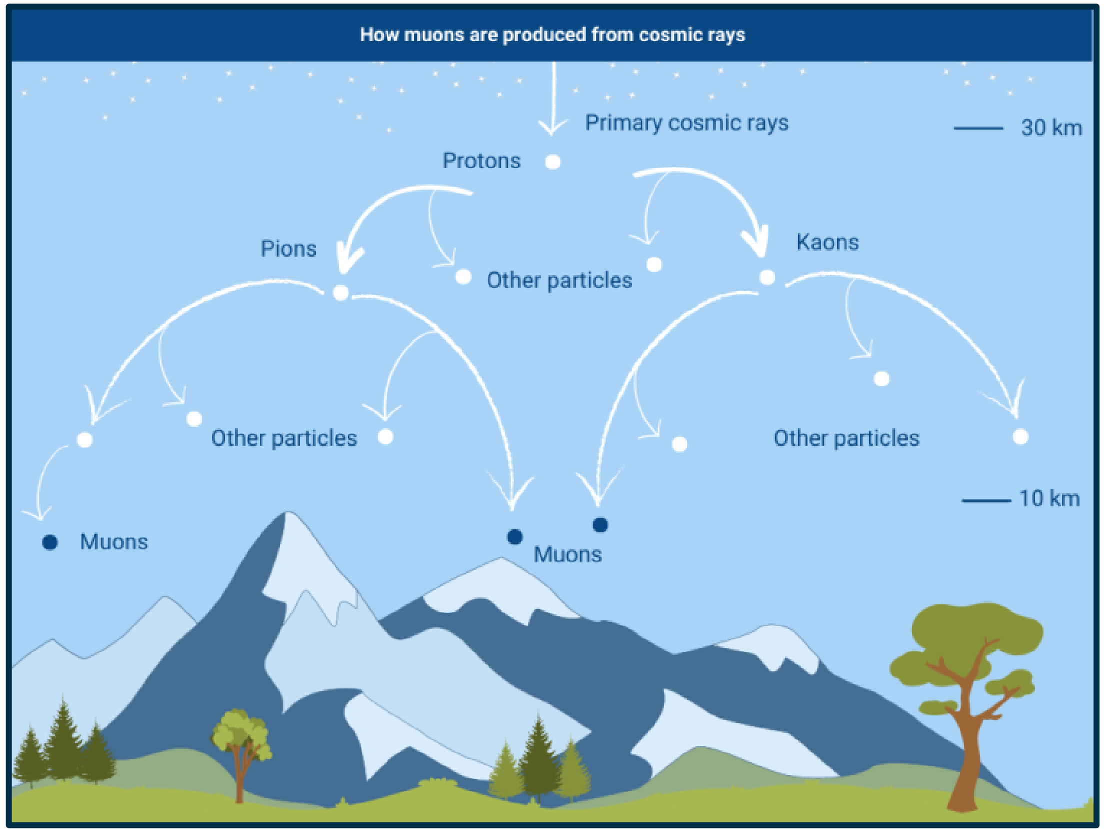
Muography



[1] Journal of Nondestructive Evaluation (2021) 40:65
https://doi.org/10.1007/s10921-021-00797-3

What is Muography?

- **Cosmic muons** are produced from the interaction of high energy **cosmic rays** and atomic nuclei in the upper atmosphere.
- They are **highly penetrating** **(~4GeV/c)**.
- However, a relatively **low flux** $(\mathbf{1cm^{-2}min^{-1}})$.
- Primary interaction is Coulomb scattering – common detectors are Emulsion Plates, gas detectors or **scintillators**.



Limitations of Muography

1. Muon imaging time

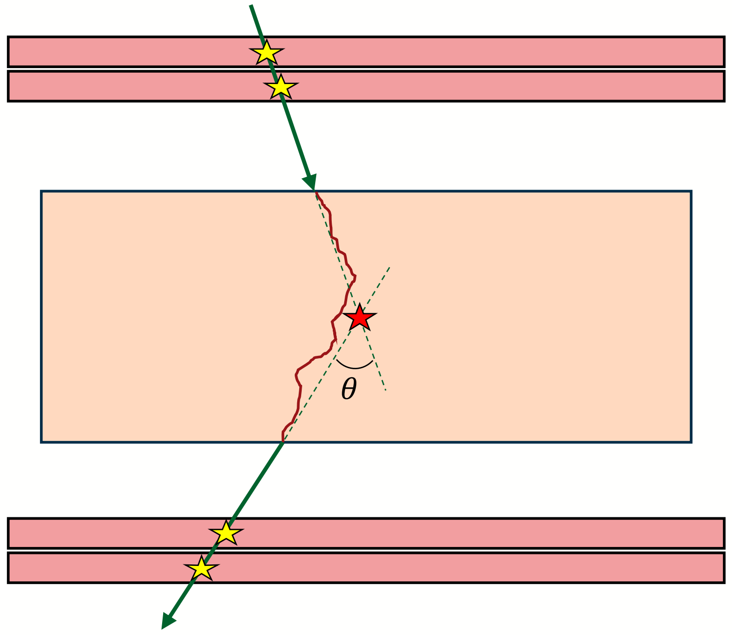
- Relies on a low **natural** muon flux.
- **Multiple scattering** makes it hard to model the muon path.
- Thus, requires high statistics - so images can take **days to months** to give reliable results.

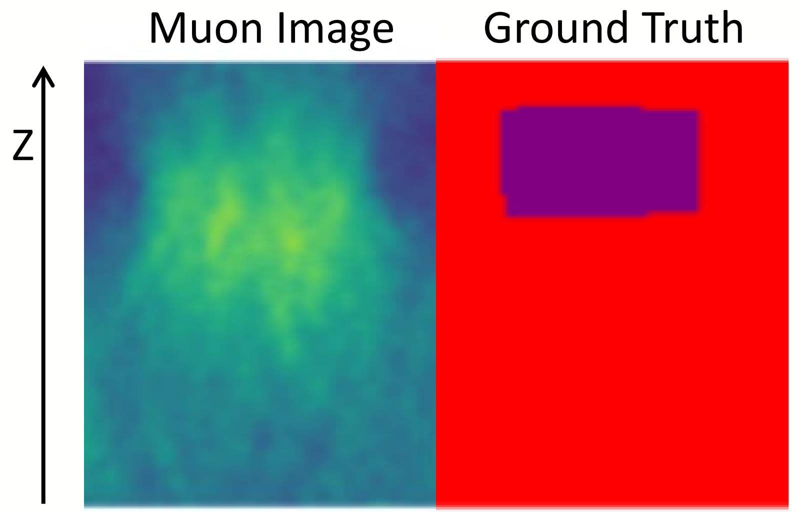
2. Z-plane smearing

- Objects **'smear'** in the direction perpendicular to the detector plane, creating **shadows or artefacts**.
- Limited **angular acceptance** ($\pm 30°$) and **inverse imaging problem** greatly reduces z resolution.

3. Interpretability

- Produces somewhat **noisy** images.
- Can be difficult to consistently and accurately interpret.

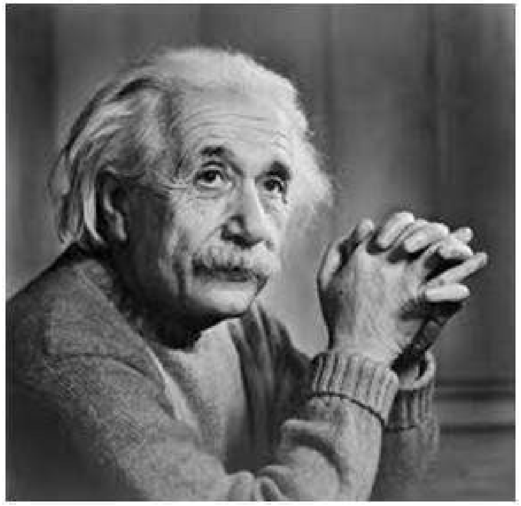




2. WHY USE MACHINE LEARNING?

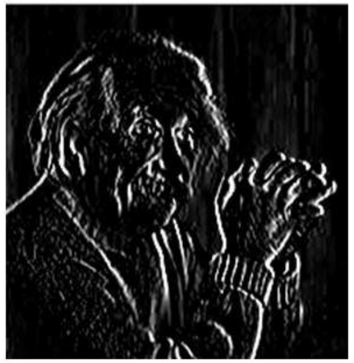
Convolutional Pattern Recognition

- **Convolutional filters** are powerful tools for detecting patterns in data.
- They are **localised**, with typical receptive fields of 3x3 to 5x5 pixels.
- However, these are **user-defined** and limited to high level feature extraction, so cannot capture complex patterns.



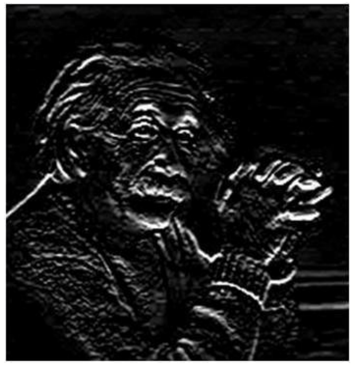
Vertical Sobel Filter

$$\otimes \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \rightarrow$$



Horizontal Sobel Filter

$$\otimes \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \rightarrow$$



Why Use Machine Learning?

- Instead of a user-defined kernel, **convolutional neural networks** (CNN’s) use a data-driven approach to optimise a kernel of learned parameters.
- These are then layered for **abstract feature learning** (deep learning), where features can be used to perform a given task.

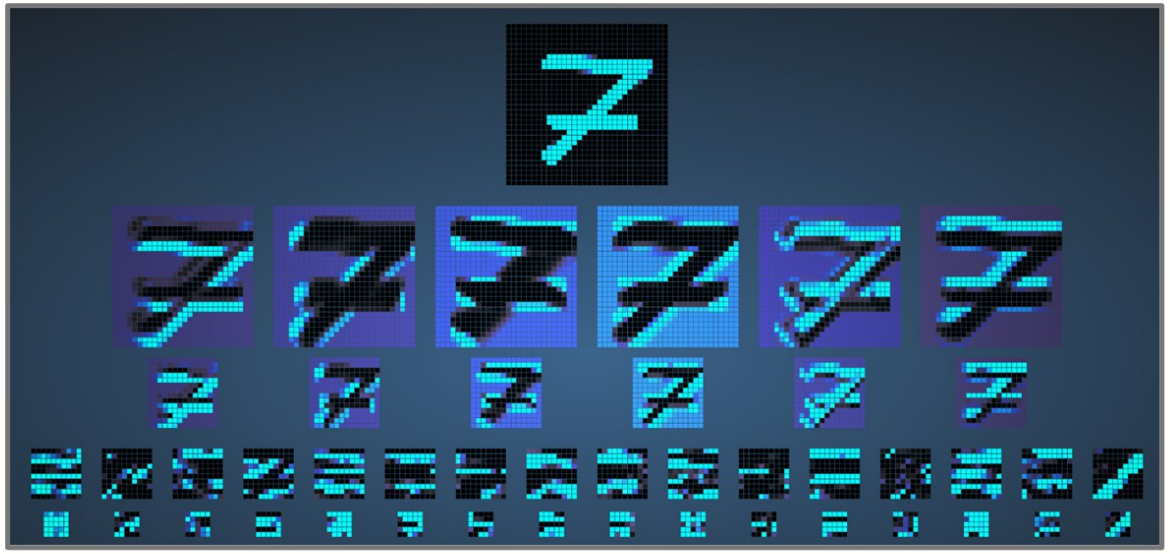


Figure: Example of abstract feature learning across many learned convolutions

Back to the problem at hand…

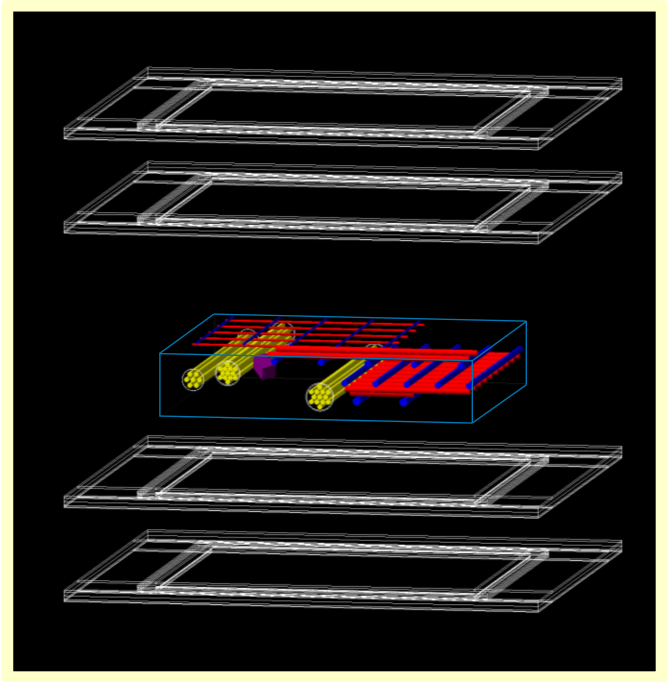
- Muon Imaging requires ***long exposure times*** to gather enough data for object resolution.
- **Why?** We're waiting for global pixelwise differences to exceed a threshold that allows ***human perception*** to identify objects.
- **Key Question:** Can we detect these differences ***before*** they become perceivable to the human eye?

3. CURRENT WORK

Creating a Dataset

- For a **supervised** task, we need inputs matched with ground truth labels.
- Due to the long sampling times, and volume of data required, we cannot rely on real data.
- We instead use muography data from physics simulations for ML model training.

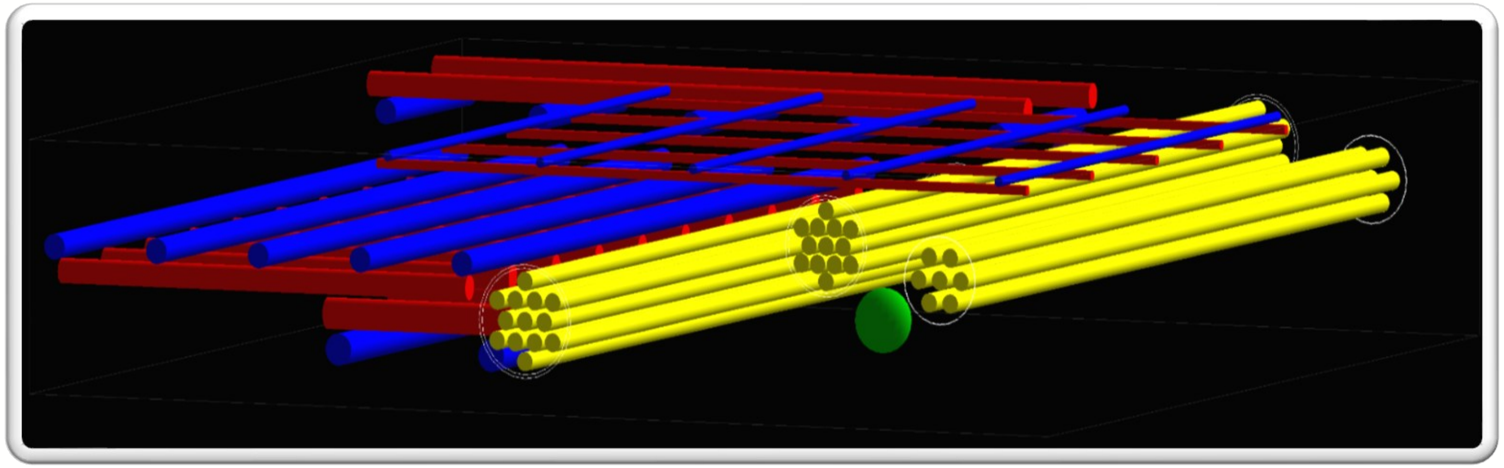
Simulation Specs:
- **Framework:** Geant4 with Ecomug.
- **Detector:** Lynkeos Muon Imaging System (MIS).
- **Block Dimensions:** 1m x 1m x 0.2m.
- **Sampling time:** 100 days (14.4×10^6 muons/day).
- Image reconstruction using point of closest approach (PoCA).



Creating a Dataset

Geometry Contents:

- 700 unique geometry configurations.
- **Rebar Grids**: 1-4 per volume, placed in XY plane.
- **Tendon Ducts:** 0-3 per volume, spanning along XZ or YZ planes.
- **Air voids:** 0-3 per volume, spherical.
- **'Unknowns':** 0-2 per volume, random shape and density.



- Dataset Diversity:
  - Randomise number of objects
  - Randomise placement.
  - Randomise geometric characteristics of objects.
- **Muon hits are gathered, scattering angles calculated, then volume is voxelised.**

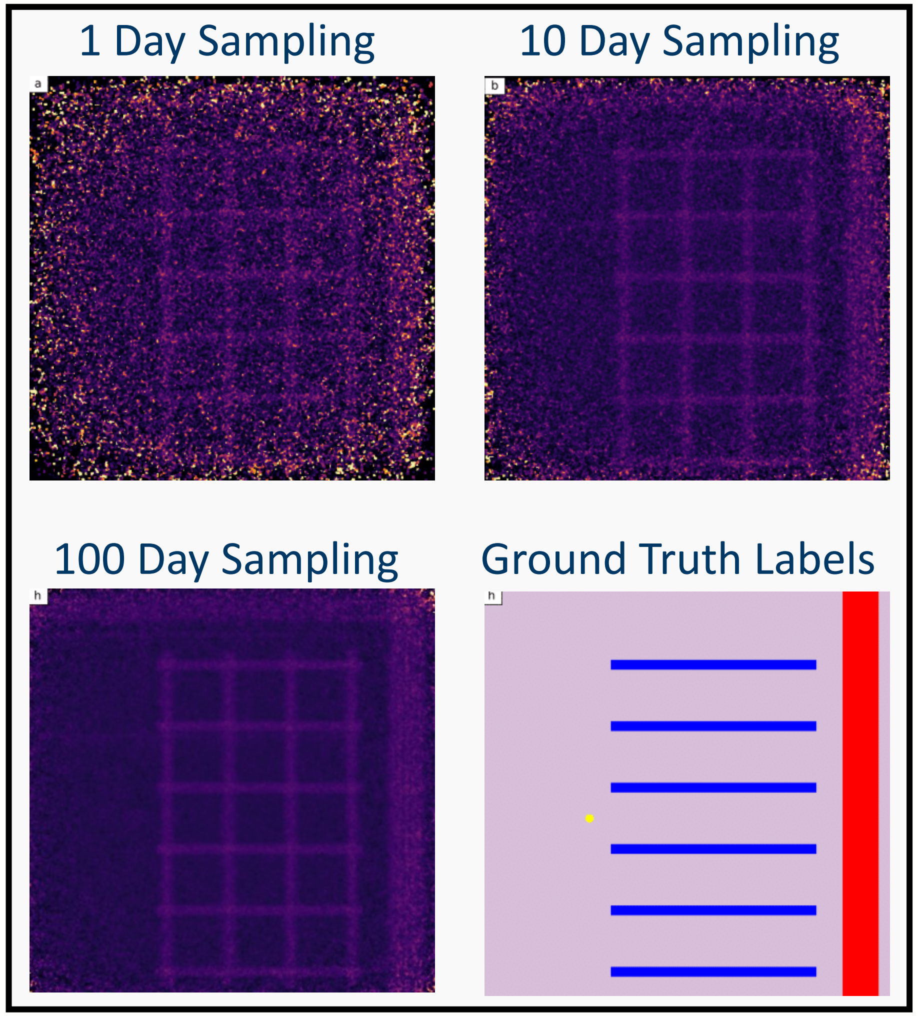
Creating a Dataset

2D Image resolution (XY plane): 500x500 pixels, 2mm.

- **Model Inputs:**
  - 100 image slices from each geometry.
  - Each slice has 100 different versions with a different sampling rate (increments of 1 day).
  - Input sampling rates are randomly sampled at each epoch for model generalisation.

- **Image Upsampling Ground Truths**:
  - Use as a control to measure differences for sampling rates.
  - Thus, choose highest available sampling rate: 100 days.

- **Segmentation Ground Truths:**
  - Produced directly from the Geant4 geometries, sliced up to produce a ground truth for each geometry slice.
  - One-hot encoded for model training.

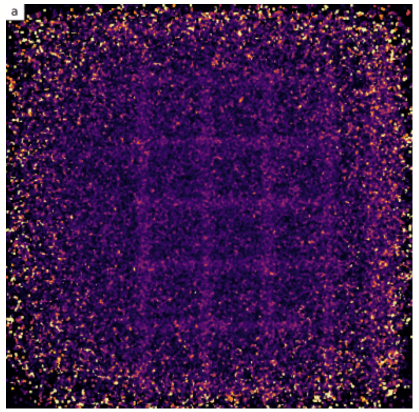


Task 1: Image to Image Upsampling

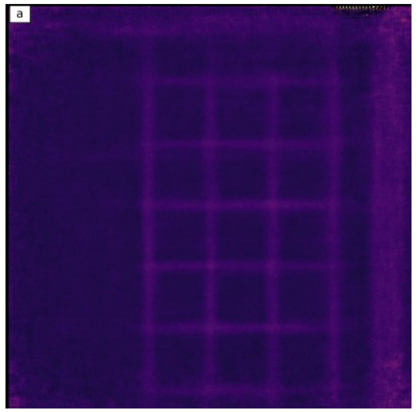
Model:

- Conditional GAN architecture used – using a **UNet** with adversarial training.
- cWGAN-GP, architecture based on **pix2pix** [2] with **ResNet6** [3] at the bottleneck.
- **Optimiser**: ADAM
- **Loss functions**: MAE (generator), Wasserstein loss (discriminator)

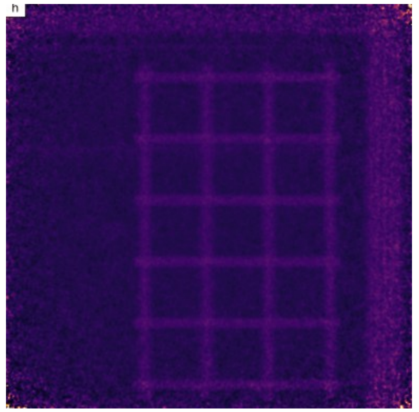
Input:
1 Day Sampling



Output:
Upsampled Image



Reference Truth:
100 Day Sampling



[2] 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017) 5967 – 5976 https://doi.org/10.1109/CVPR.2017.632

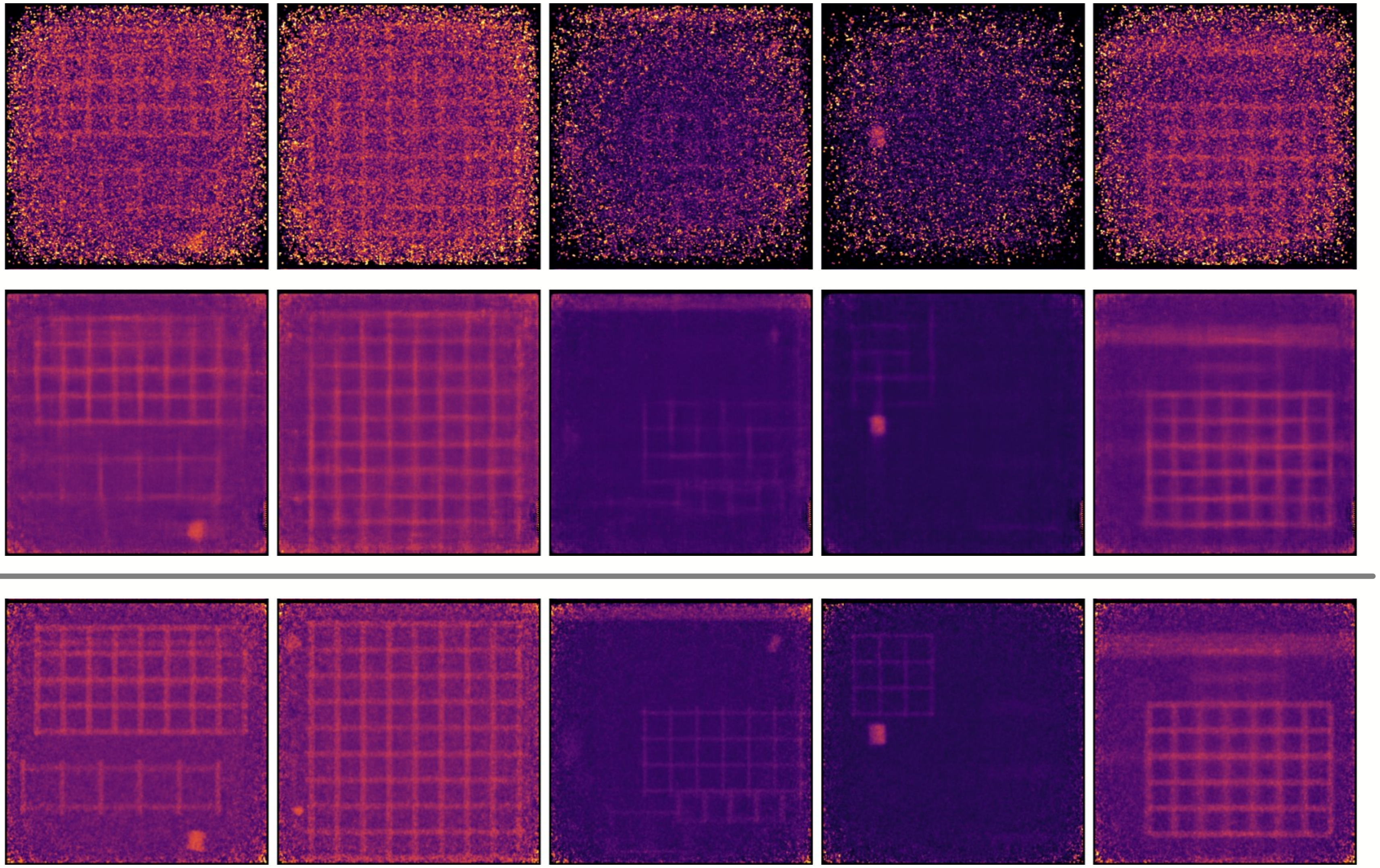
University of Glasgow

Task 1: Image to Image Upsampling

1 Day Muon Image

Upsampled 1 Day Image

Ground Truth **100 Day** Image



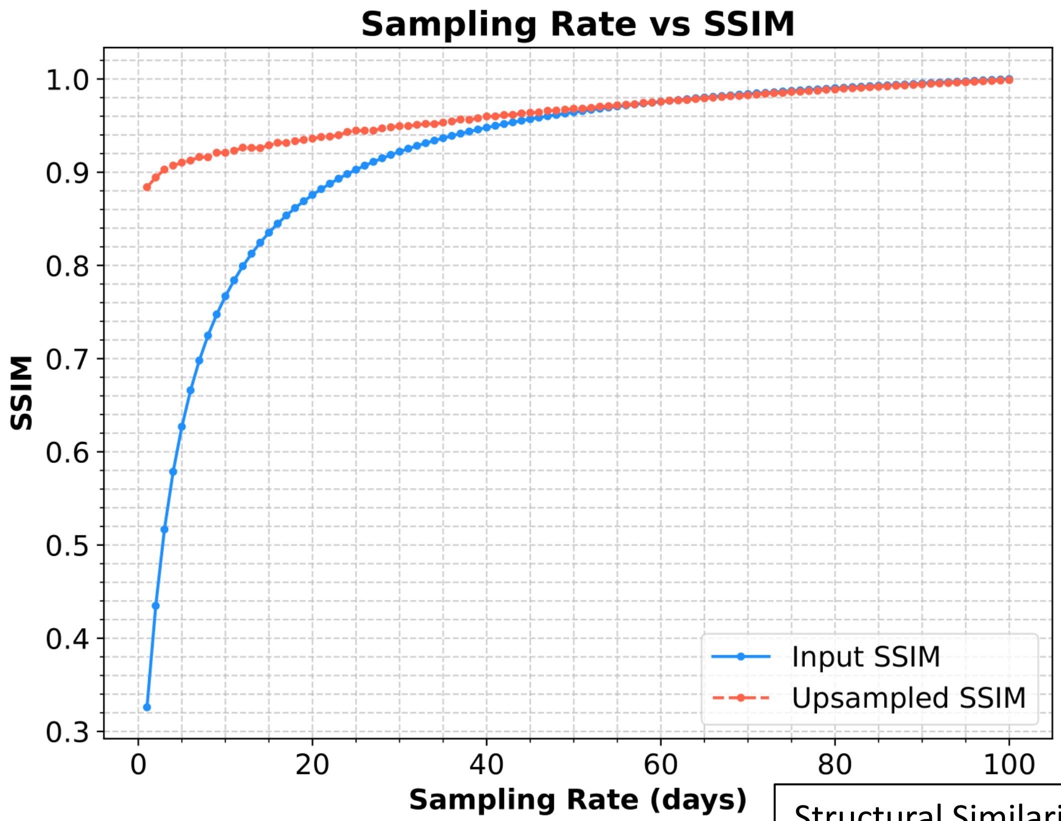
Preliminary Results: Upsampling

Key Takeaways:

1. The model can take **1 days'** worth of data to produces an image that would otherwise take **~20 days.**
2. At around 50-60 days, we see convergence.

➢ This metric doesn't capture the full picture.

➢ We can break down the images by object type using segmentation.



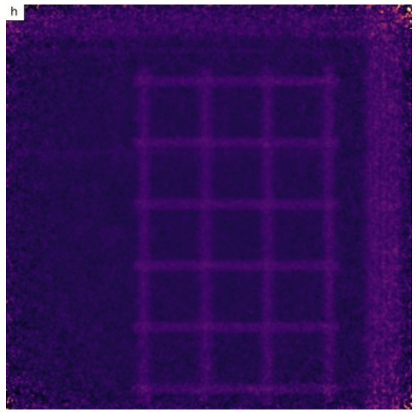
Structural Similarity Index Measure (SSIM) assesses:

- Luminance
- Contrast
- Structure

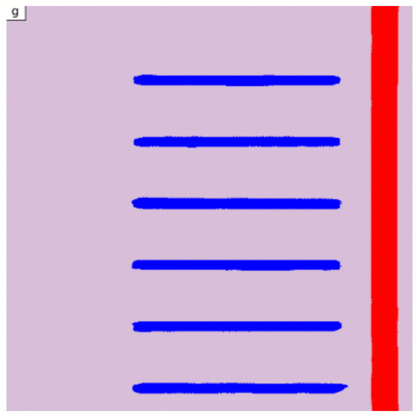
Task 2: Image Segmentation

- Performed on the **highest sampling** (100-day data), for development.
- Utilises the ground truth geometries from our simulation setup.
- **X-Y plane segmentation** – no z-information.
- Labels: concrete, rebar, ducts, voids, unknowns.
- Model: Same as upsampling model but using Dice and cross-entropy losses.

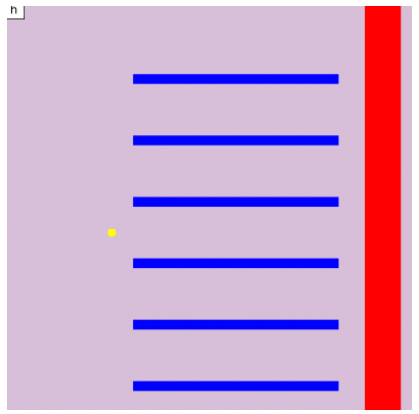
Input: 100 Day Muon Image



Output: Segmentation Map



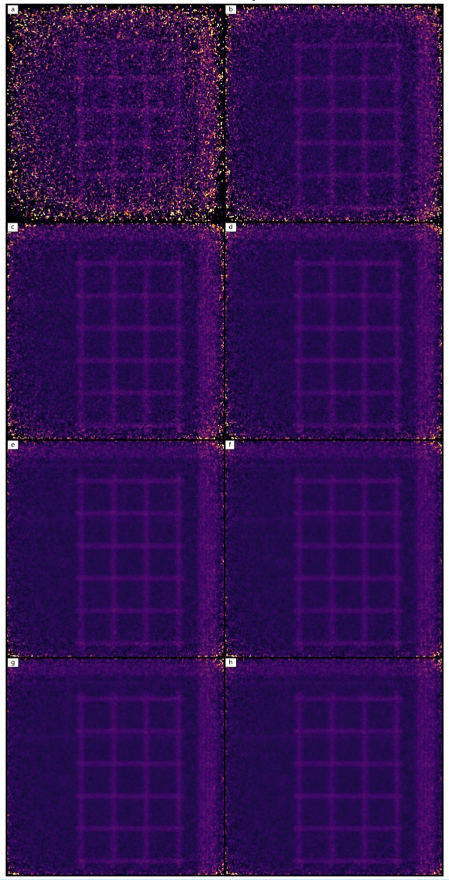
Reference: Ground Truth



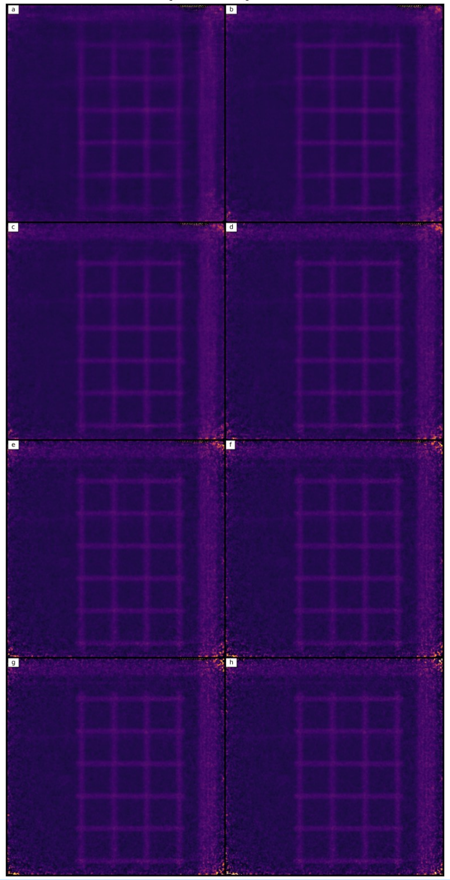
Preliminary Results: Segmentation

- Instead of a high sampled muography ground truth, we use the actual geometry (from the simulation).
- We perform **semantic segmentation** to classify each pixel in the image.

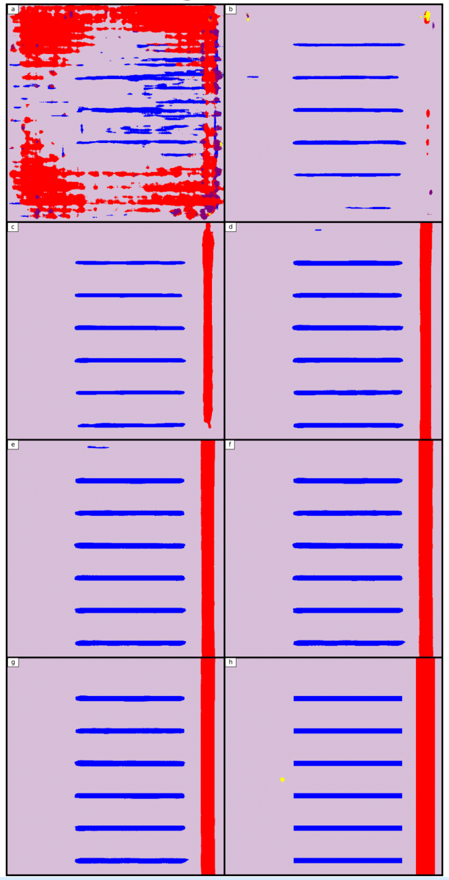
Muon Inputs



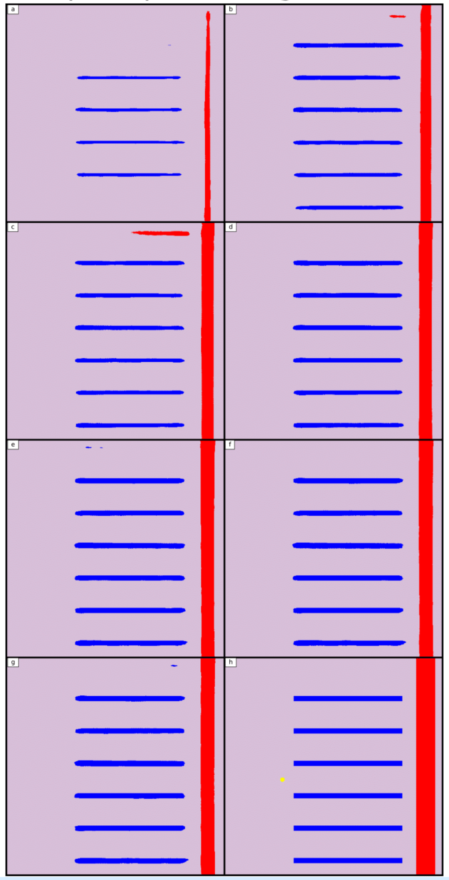
Upsampled



Segmented



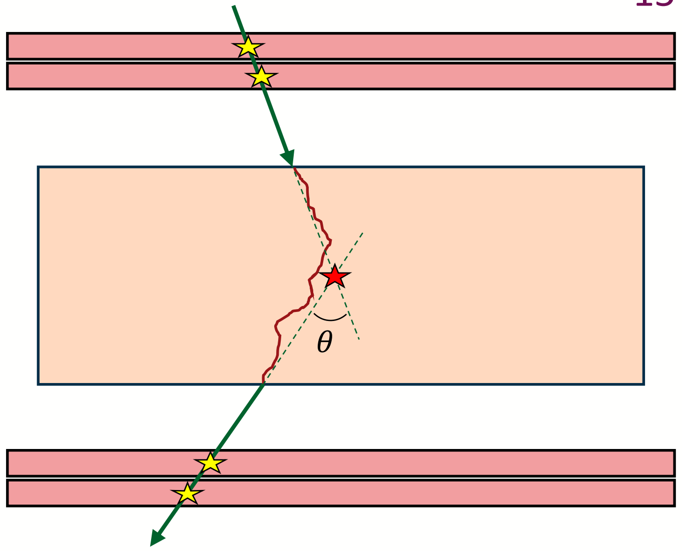
Upsampled & Segmented



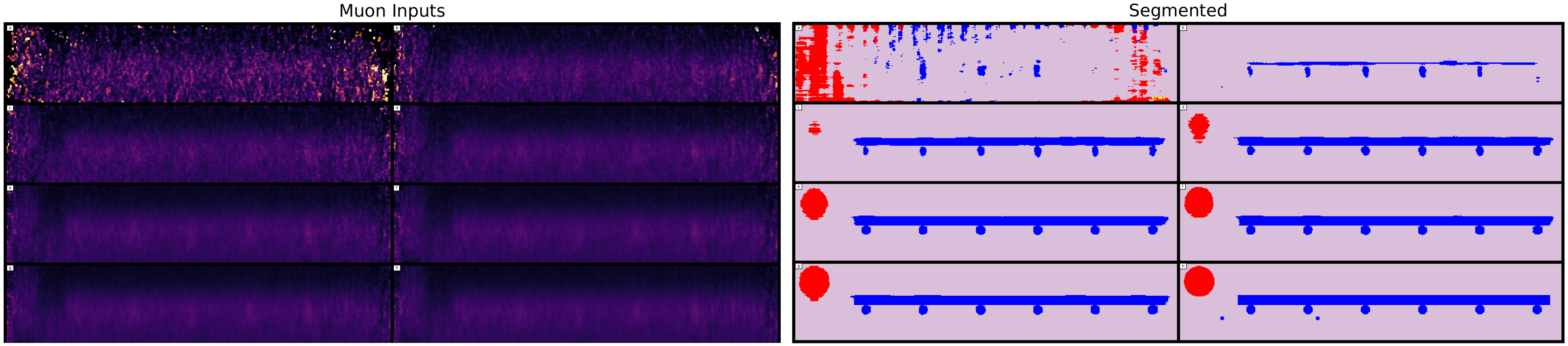
1 Day	5 Day
10 Day	20 Day
40 Day	60 Day
80 Day	Ground Truth

Lilac = Concrete
Blue = Rebar Grid
Red = Tendon Duct
Yellow = Air Void

Preliminary Results: Segmentation



- Inverse imaging problem means images get **smeared in the z-direction.**
- Since the model is using known geometries (without smearing), it has **learned to distinguish between shadows and objects.**



- **Segmentation significantly reduces shadowing/smearing effects.**

1 Day	5 Day
10 Day	20 Day
40 Day	60 Day
80 Day	Ground Truth

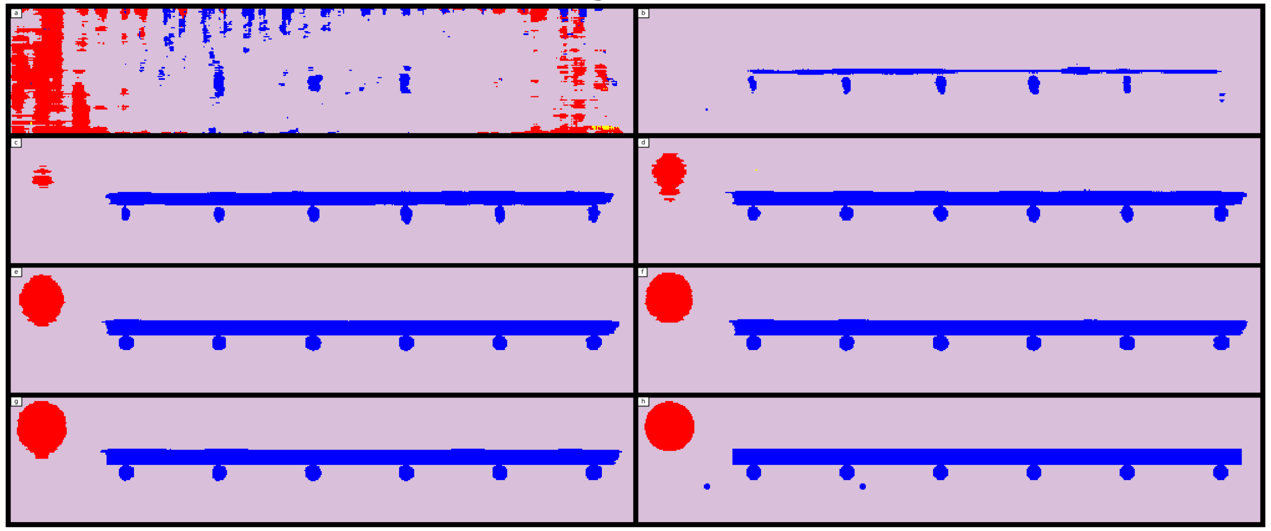
Lilac = Concrete
Blue = Rebar Grid
Red = Tendon Duct
Yellow = Air Void

Preliminary Results: Segmentation

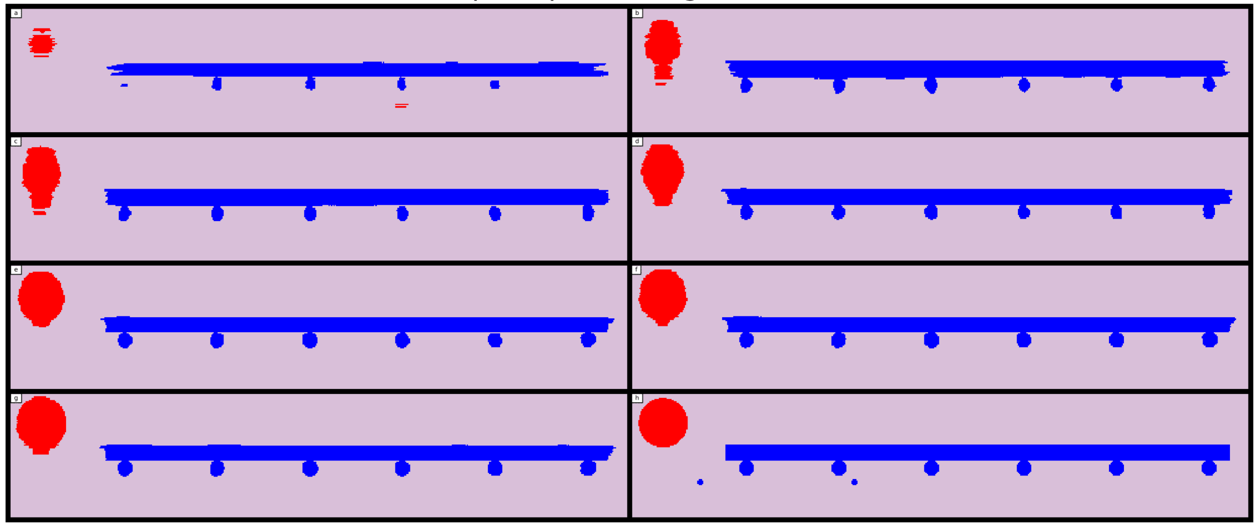
- Let's look at how upsampling changes our segmentation outputs:

1 Day	5 Day	Lilac = Concrete
10 Day	20 Day	Blue = Rebar Grid
40 Day	60 Day	Red = Tendon Duct
80 Day	Ground Truth	Yellow = Air Void

Segmented



Upsampled & Segmented



- Segmentation significantly reduces shadowing/smearing effects.
- **Upsampling shows a significant accuracy increase for low sampling rates.**

Preliminary Results: Segmentation

Ground Truth

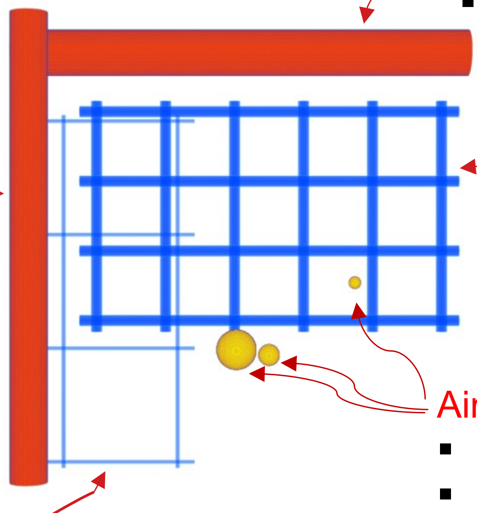
Duct 1:
- 80 mm Diameter
- z = 53 mm

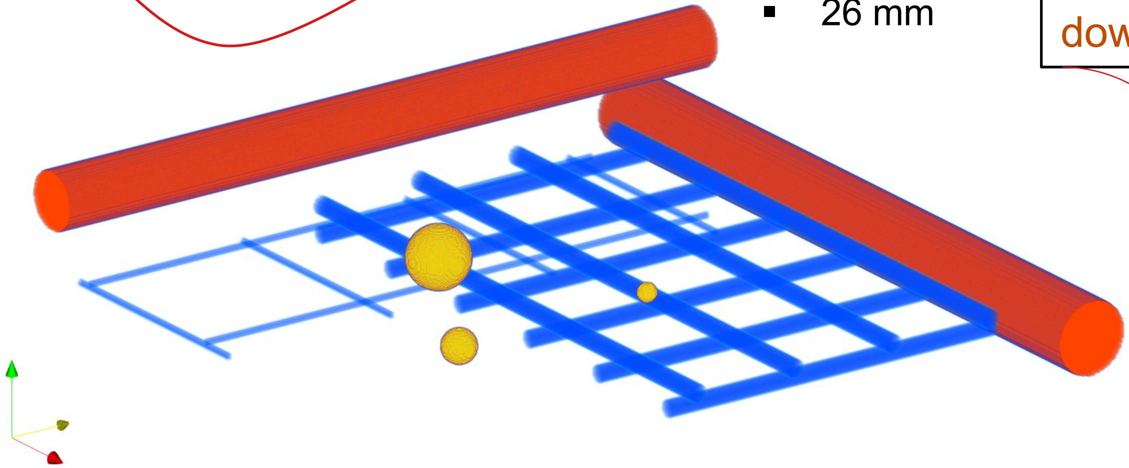
Rebar Grid 2:
- 10 mm Diameter
- z = 152 mm

Duct 2:
- 100 mm Diameter
- z = 75mm

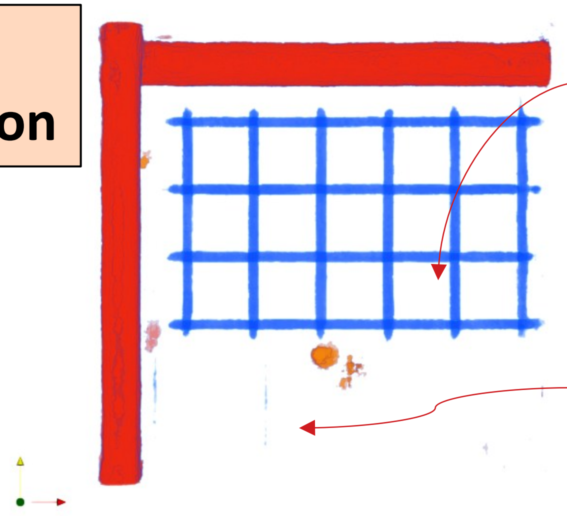
Rebar Grid 1:
- 25 mm Diameter
- z = 107 mm

Air Voids (diameter):
- 84 mm
- 47 mm
- 26 mm





100-Day Segmentation

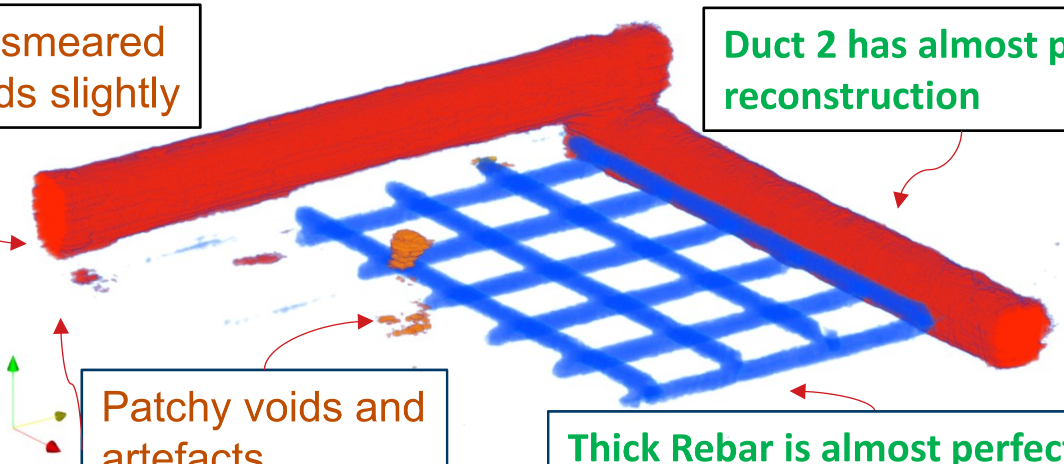


Smallest void non-existent

Thin bottom rebar is almost non-existent

Duct 1 is smeared downwards slightly

Duct 2 has almost perfect reconstruction



Patchy voids and artefacts

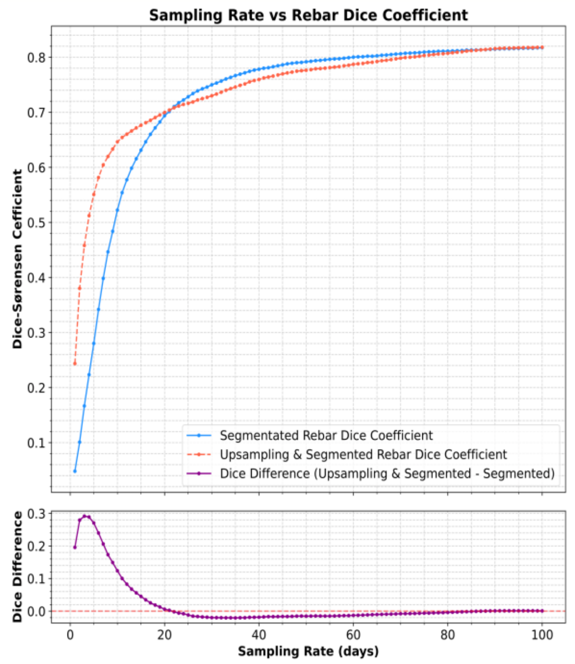
Thick Rebar is almost perfectly reconstructed

Preliminary Results: Upsampling and Segmentation

- Dice coefficient ranges from 0 (bad) to 1 (perfect).
- Above 0.7 is considered very good.

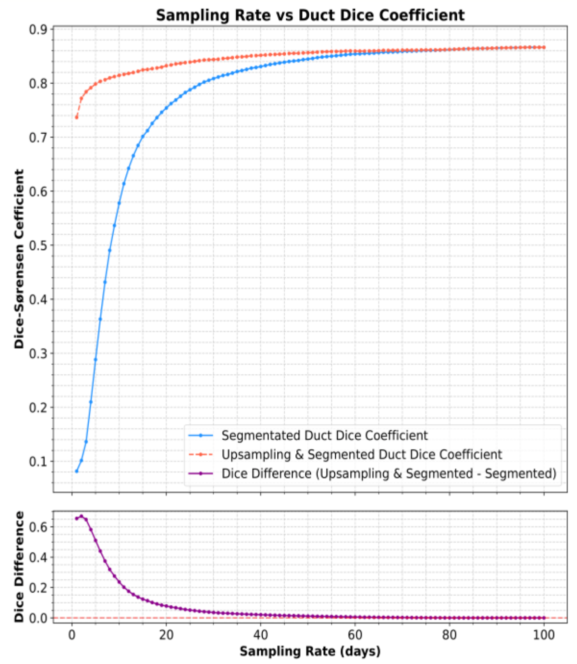
$$\mathrm{Dice}_i = \frac{2 \times \mathrm{TP}_i}{2 \times \mathrm{TP}_i + \mathrm{FP}_i + \mathrm{FN}_i}$$

Rebar Grids



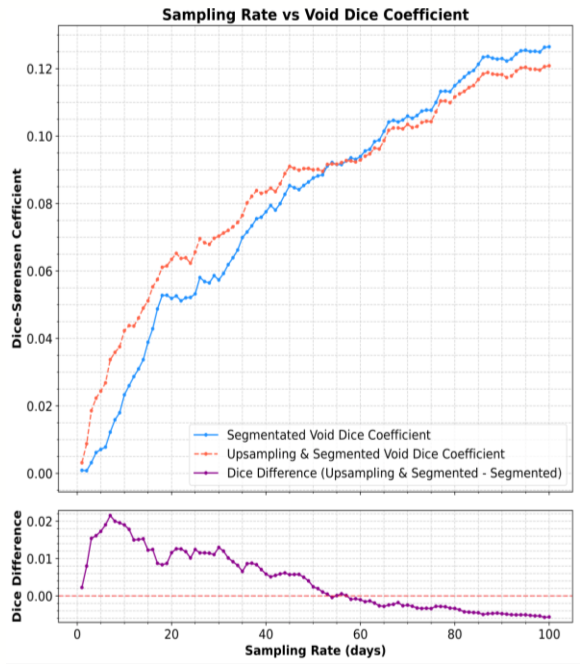
- Good perception increase at low sampling rates.
- After ~20 days sampling, outputs worse than inputs
- Upsampler may be altering data distribution such that it is hindering the segmenter.

Tendon Ducts



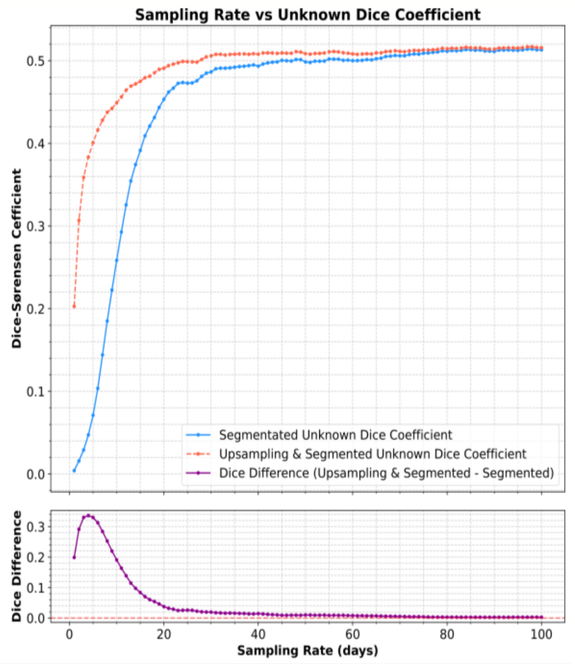
- Very high perception score, >70% for all sampling rates.
- Upsampling provides significant perception increase.
- Ducts are biased to edge placement and are large in size.

Air Voids



- Poor perception scores < 15%.
- Little difference with upsampling.
- Class contains lowest proportion of pixels, biasing it to not focus on upsampling these features.
- Harder feature to detect.

'Unknowns'



- Ok perception score, considering some small shapes and very low densities.
- Upsampling provides significant perception increase.
- Breakdown of density, size and shape would provide more insight.

Summary

Muon Imaging Problems:

1. Imaging time.
2. Z-plane smearing.
3. Interpretability.

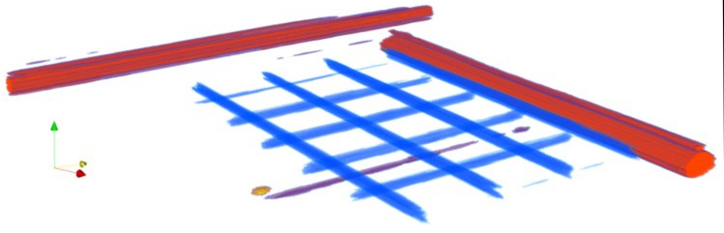
Machine Learning Solution:

- **Upsampling** significantly reduces imaging time and reduces noisy effects.
- **Segmentation** significantly reduces smearing effects AND provides automatic interpretation of results.
- However, work to be done to improve accuracies.

Moving Forward:

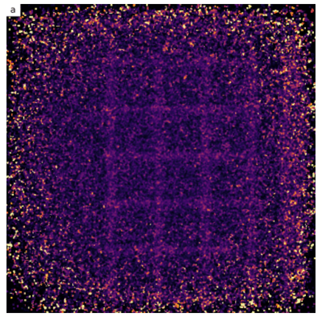
- Test model performance for lower sampling times, < 1 day.
- Improving model accuracies: use **3D** and **hybrid conv-transformer architectures** for larger context size.
- Test models on **real datasets**, informing future simulation design.

1 Day Upsampled and segmented:

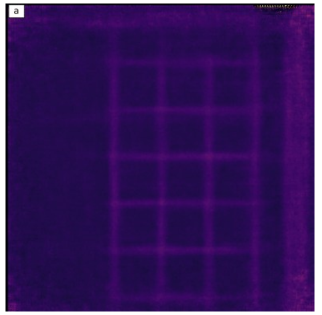


Data Pipeline:

Input:
1 Day Sampling image

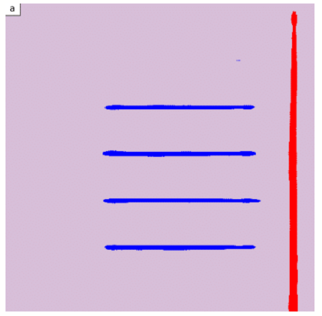


Step 1: Upsampling



Step 2: Segmentation

Output:
Upsampled & Segmented image



Thanks for Listening

Additional thanks to my supervisors D. Mahon, G. Yang and S. Gardner, as well as E. Niederleithinger from BAM for their guidance and support.

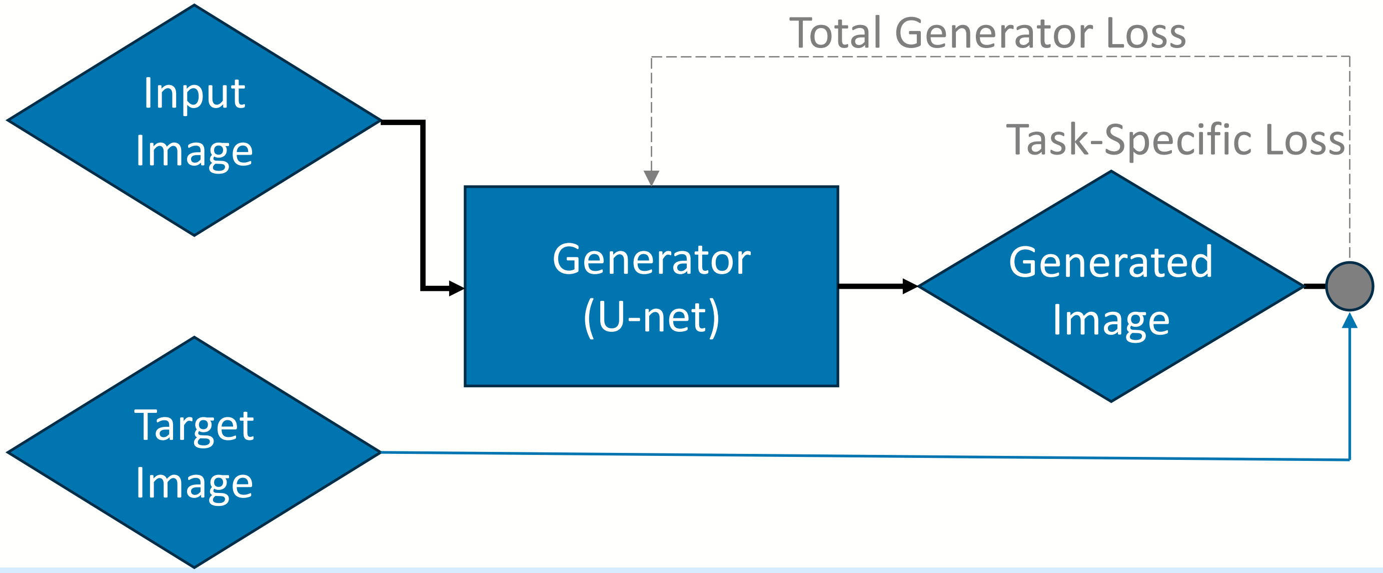
For more info, see https://doi.org/10.3390/particles8010033, where this work has been recently published.

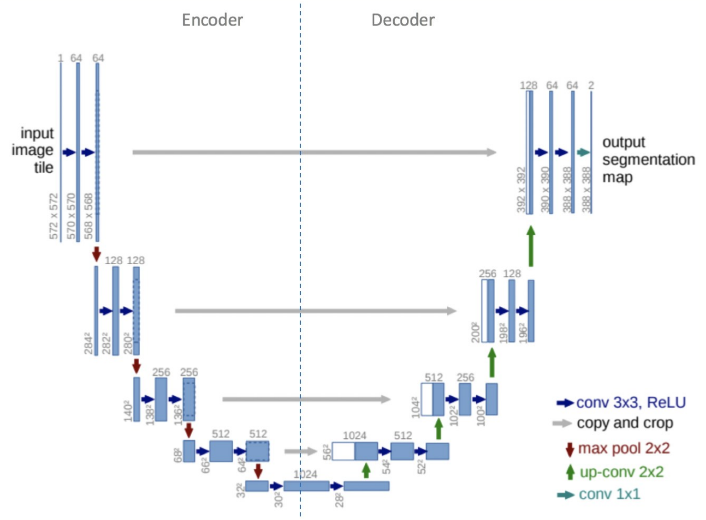
Backups

The Conditional GAN (cGAN)

- cGANs are the supervised version of the GAN (conditioned on an input).
- Contain two parts: **generator** and **discriminator**.
- **Adversarial process**: compete until Nash equilibrium is reached.
- The model used is heavily based on the pix2pix architecture [2].

[2] https://phillipi.github.io/pix2pix/

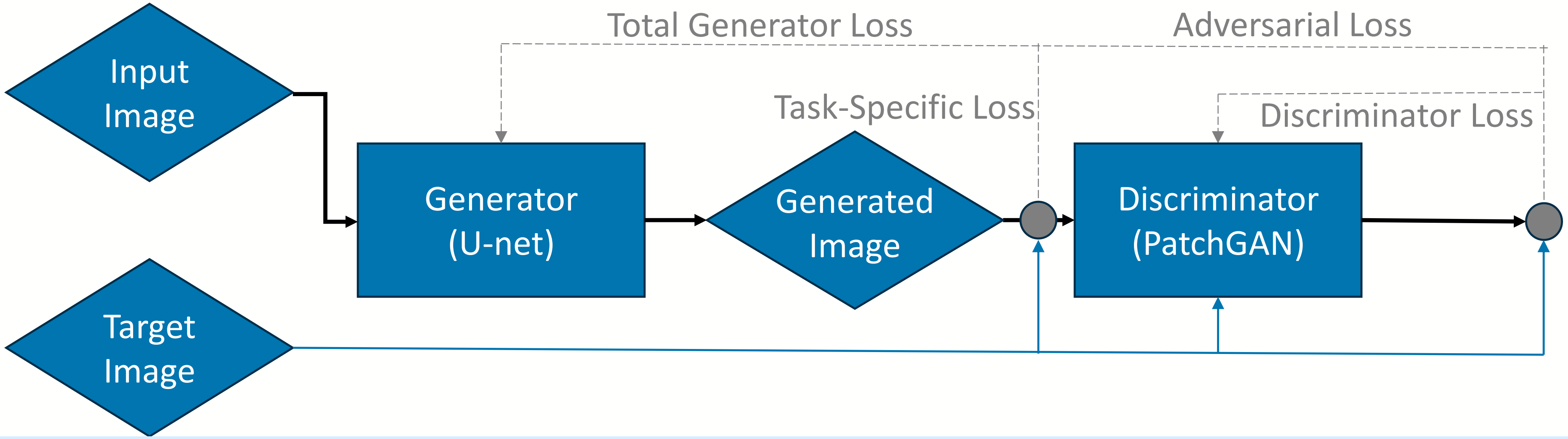




The Conditional GAN (cGAN)

- cGANs are the supervised version of the GAN (conditioned on an input).
- Contain two parts: **generator** and **discriminator**.
- **Adversarial process**: compete until Nash equilibrium is reached.
- The model used is heavily based on the pix2pix architecture [2].

[2] https://phillipi.github.io/pix2pix/



Future Work

1. Model Optimisation

- Model is in early stages and requires development for reliable reconstruction of all materials.
- Move towards models that increase context size: **global** context, **3D** context.
- Optimisation of method (do we upsample, then segment – or do we make one model for end-to-end).

2. Defect Segmentation Task

Ultimate goal is to perform defect segmentation. Defects include:

- ➢ Rebar corrosion.
- ➢ Voids, honeycombing and cracks in concrete.
- ➢ Tendon duct: strand placement/corrosion, air spaces.

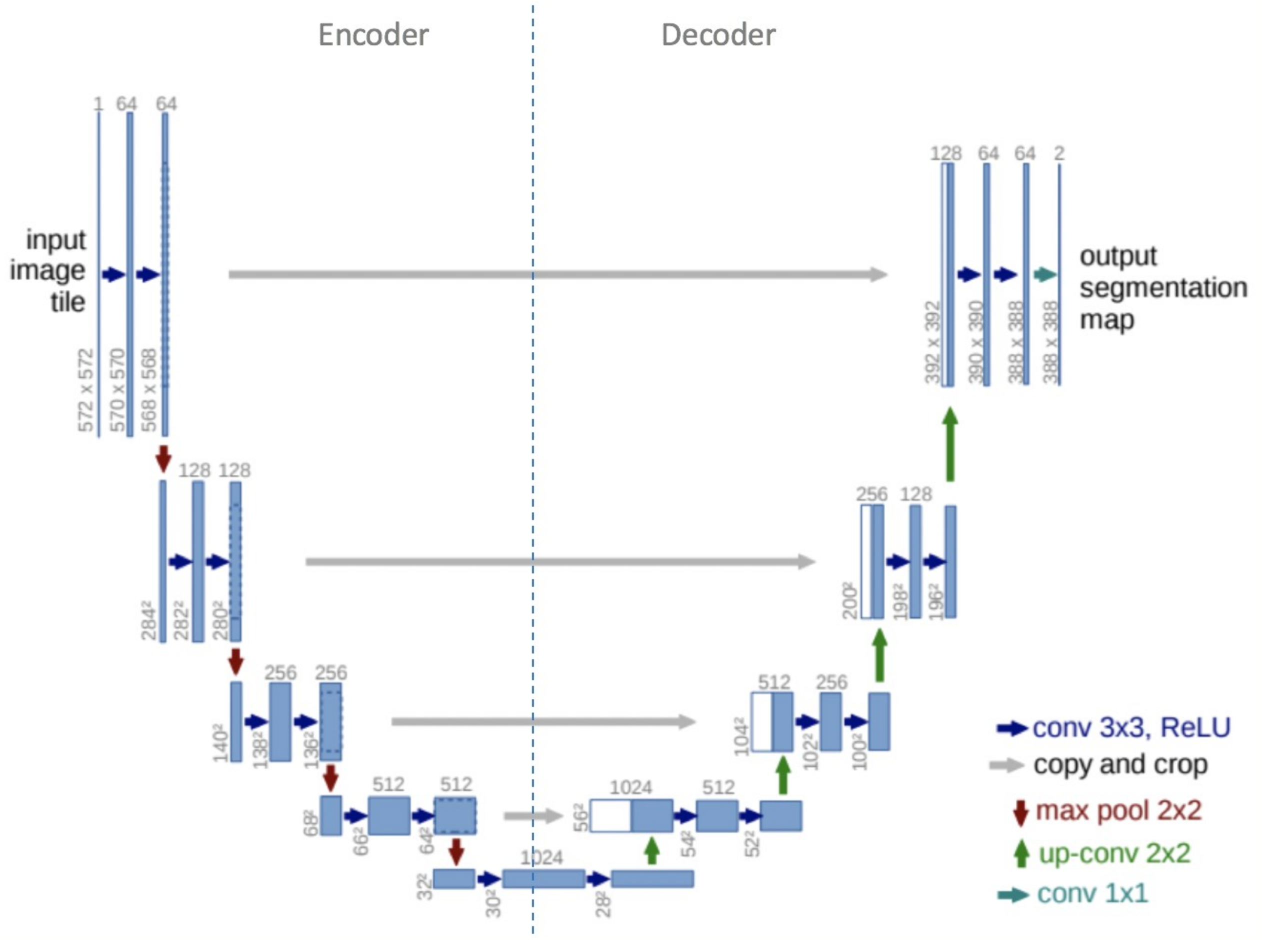
3. Model Generalisation

- Assessing models on real datasets.
- Non-ideal object placement.
- Handling of different detector orientations.
- Handling of a variety of detector spacings.

University of Glasgow

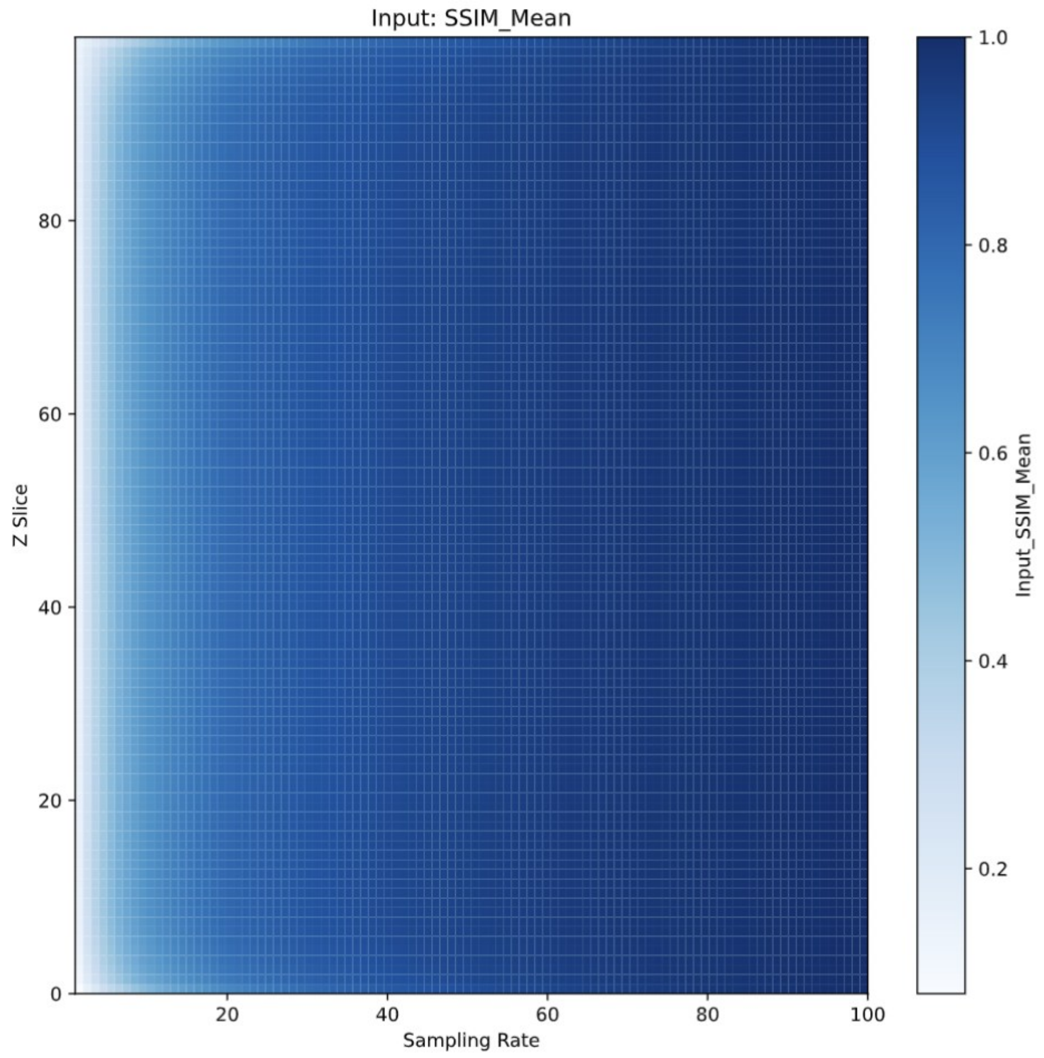
U-Nets

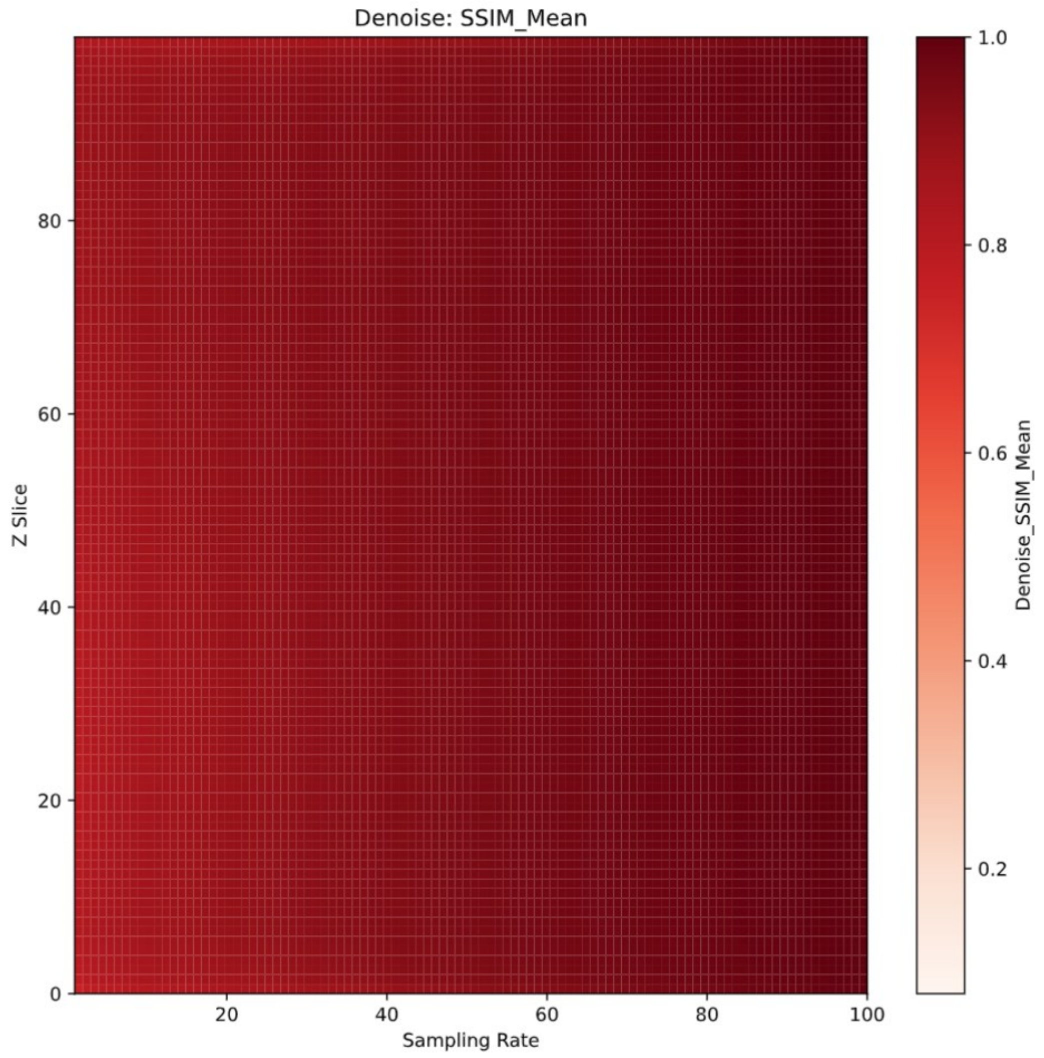
- Standard encoder-decoder CNNs are lossy – lose information.
- Introduce 'skip connections' between layers in the encoder and decoder.
- Allows for uncaptured, minor details to be preserved while keeping model complexity low.
- U-Nets are widely used for I2I translation tasks, especially in medical imaging.



University of Glasgow

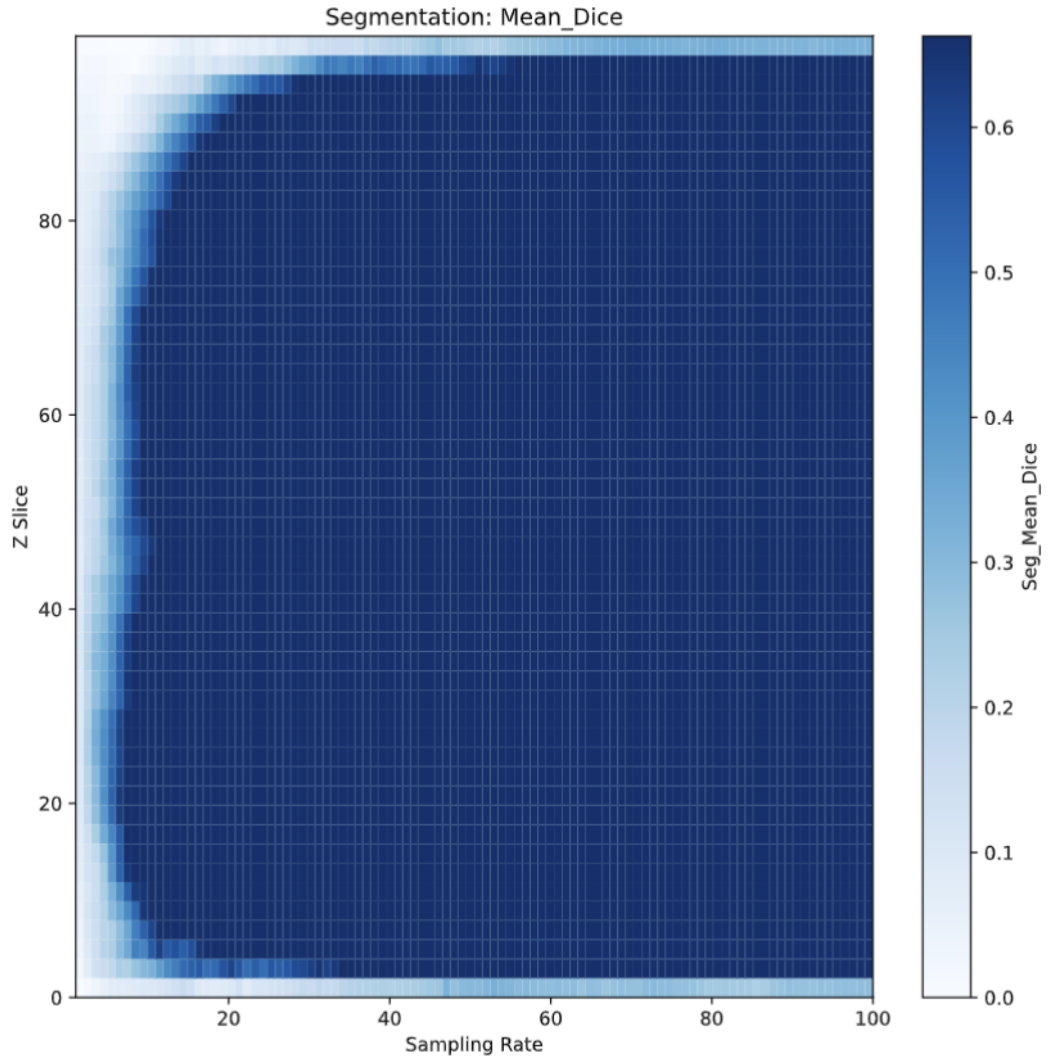
Z-dependence

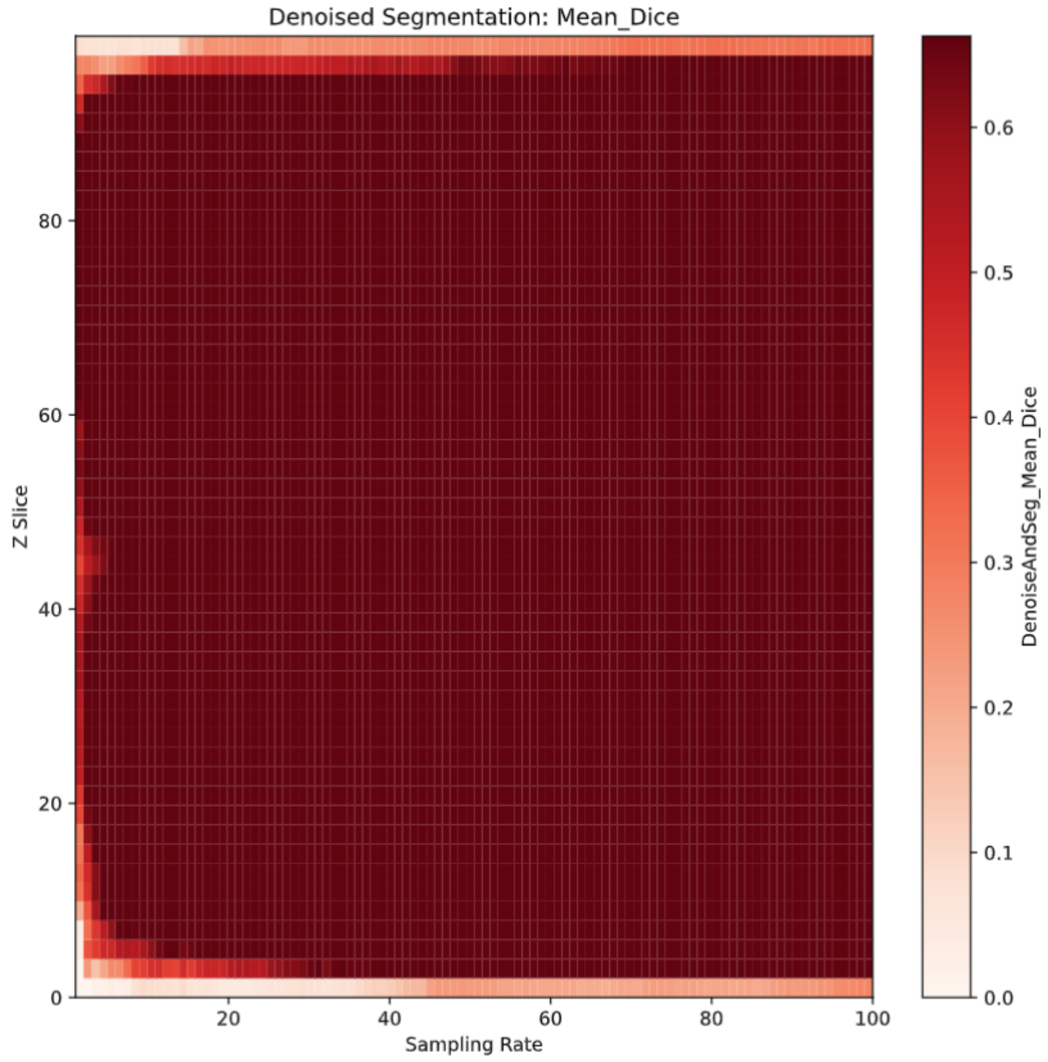




-- BACKUP SLIDES --

Z-dependence

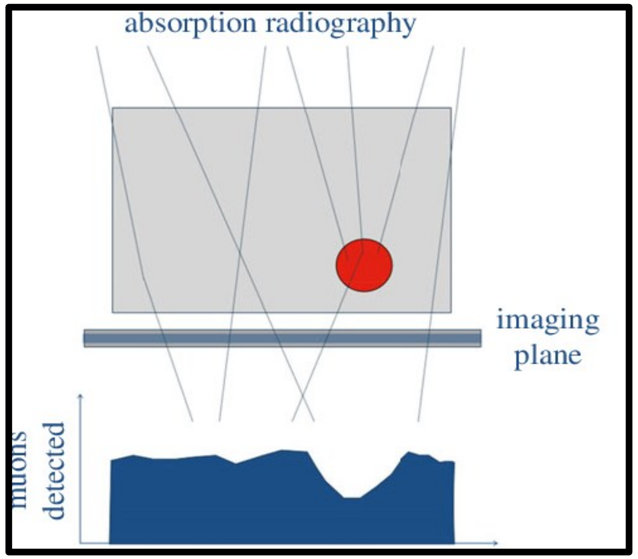


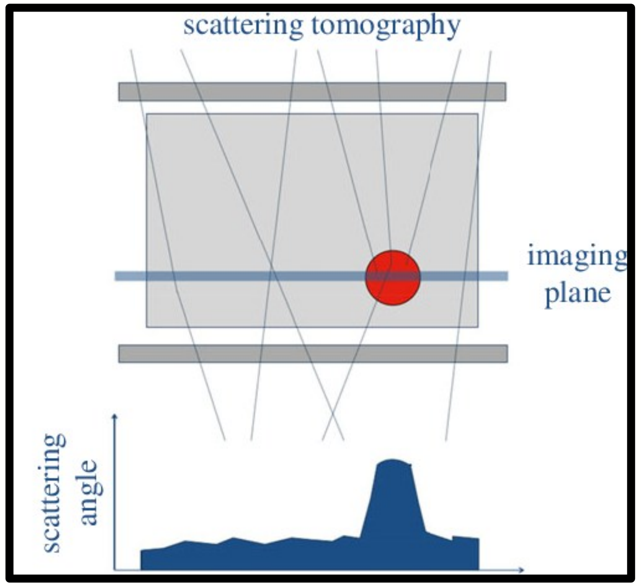


University of Glasgow

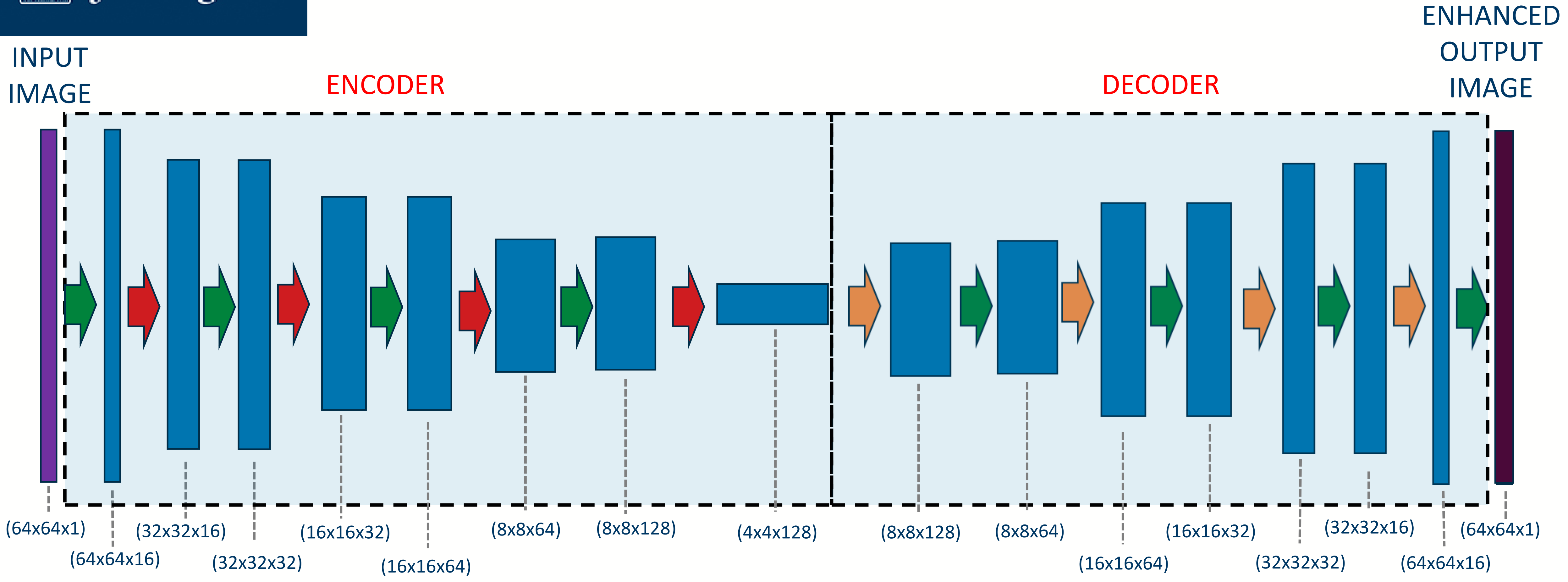
What is Muography?

- **Absorption Radiography:**
  - Uses muon **attenuation** (stopping).
  - Two detector planes **behind** the object are required.
- **Scattering Tomography:**
  - Uses reconstructed **scattering angles** of muons.
  - Two detector planes **in front and behind** the object are required.
- This has been successfully applied to:
  - Nuclear waste characterization,
  - Border control,
  - Mining,
  - (and others).





CNN: Encoder-Decoder Architecture



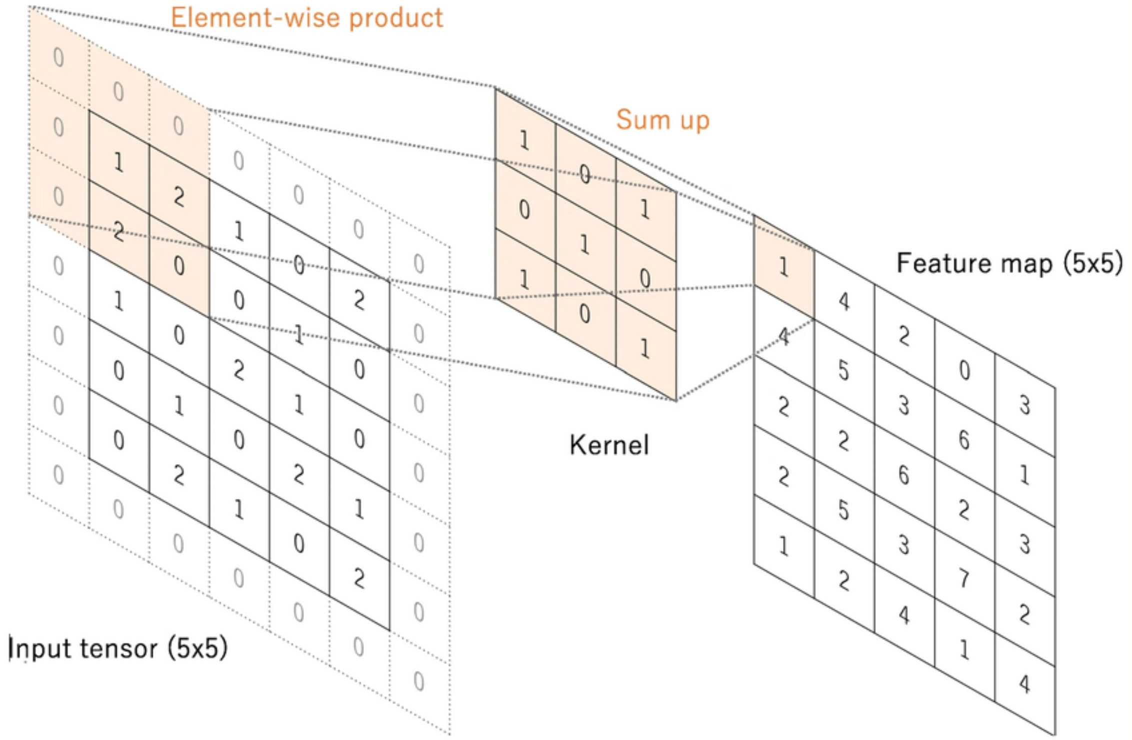
🟩 Convolution and activation function

🟥 Pooling Layer (2x2 window)

🟧 Upsampling Layer

Convolutional Feature Extraction

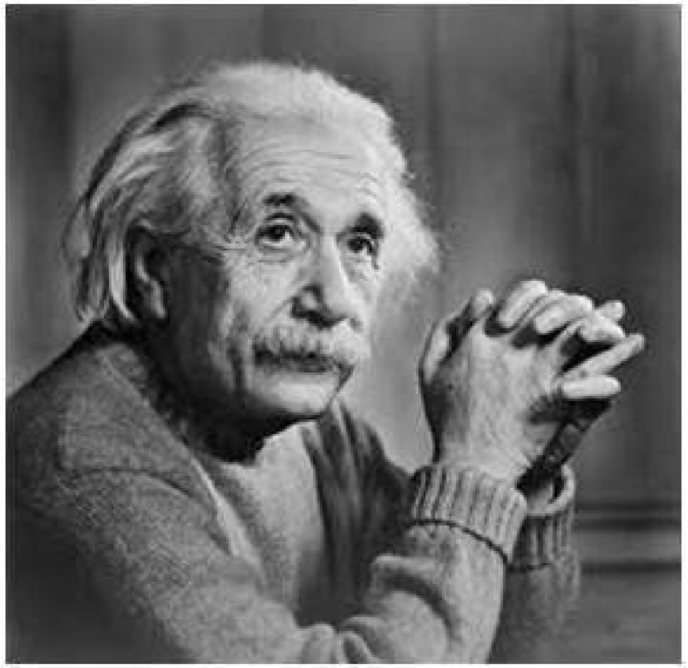
- Convolution operations have been used for image processing for a long time.
- The feature extracted from an input image depends on the kernel.
- Convolution of the input with a kernel produces a feature map.
- Many different kernels can be performed, each looking for different features and each producing a feature map.



$$= \sum_{i=1}^{3}\sum_{j=1}^{3}\begin{bmatrix}0*1 & 0*0 & 0*1\\ 0*0 & 1*1 & 2*0\\ 0*1 & 2*0 & 0*1\end{bmatrix} = \begin{bmatrix}0 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 0\end{bmatrix} = 1$$

Convolutional Feature Extraction [2]

- Example: Highlight vertical and horizontal features of this picture of Einstein.

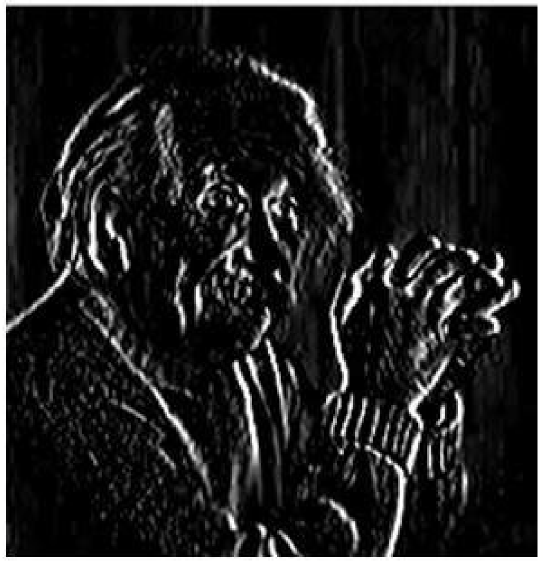


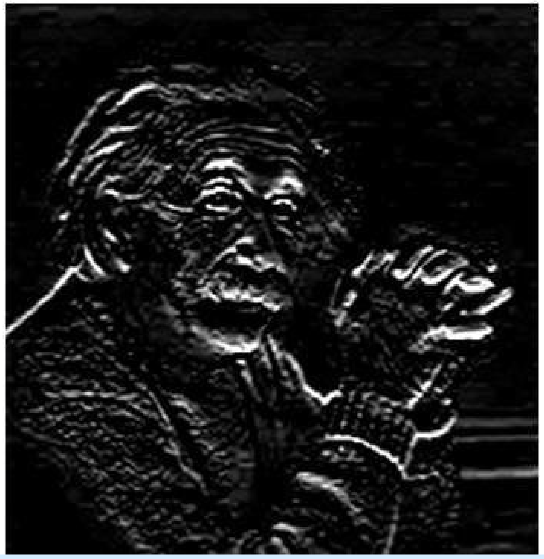
Vertical Sobel Filter

$$\otimes \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

Horizontal Sobel Filter

$$\otimes \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$





Convolution in CNNs [1]

- CNNs however learn the kernels they use – allowing for complex task-specific learning.
- The learnable parameters in a CNN are the components of these kernels – each containing a set of weights ($w_{i,j}$) and a single bias term (b):

$$O_{i,j} = w_{i,j} \times I_{i,j} + b$$

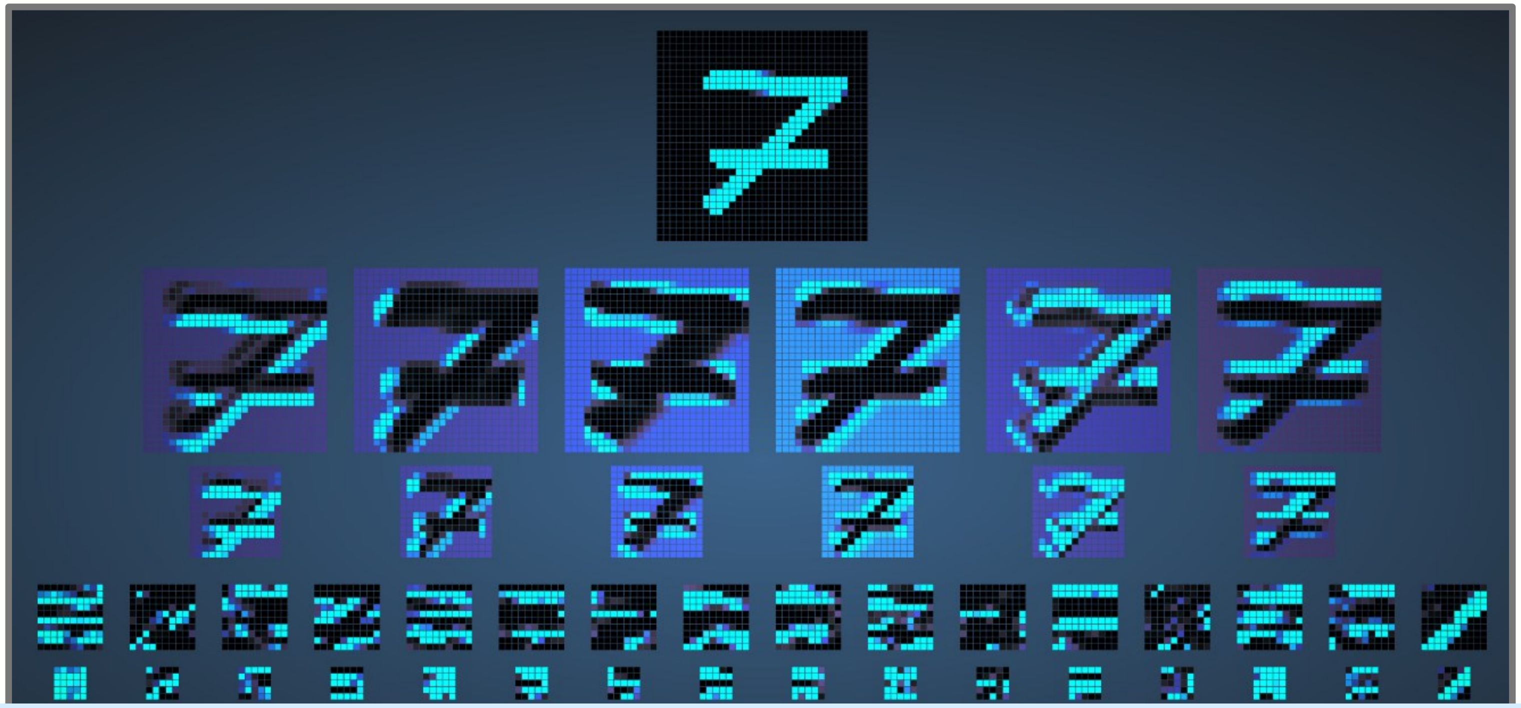
- $I_{i,j}$ is the input (3x3 window of input)
- $w_{i,j}$ are the weights of the 3x3 kernel
- b is the bias term
- $O_{i,j}$ is the 3x3 output of the element-wise product with bias.

9 params for one 3x3 kernel

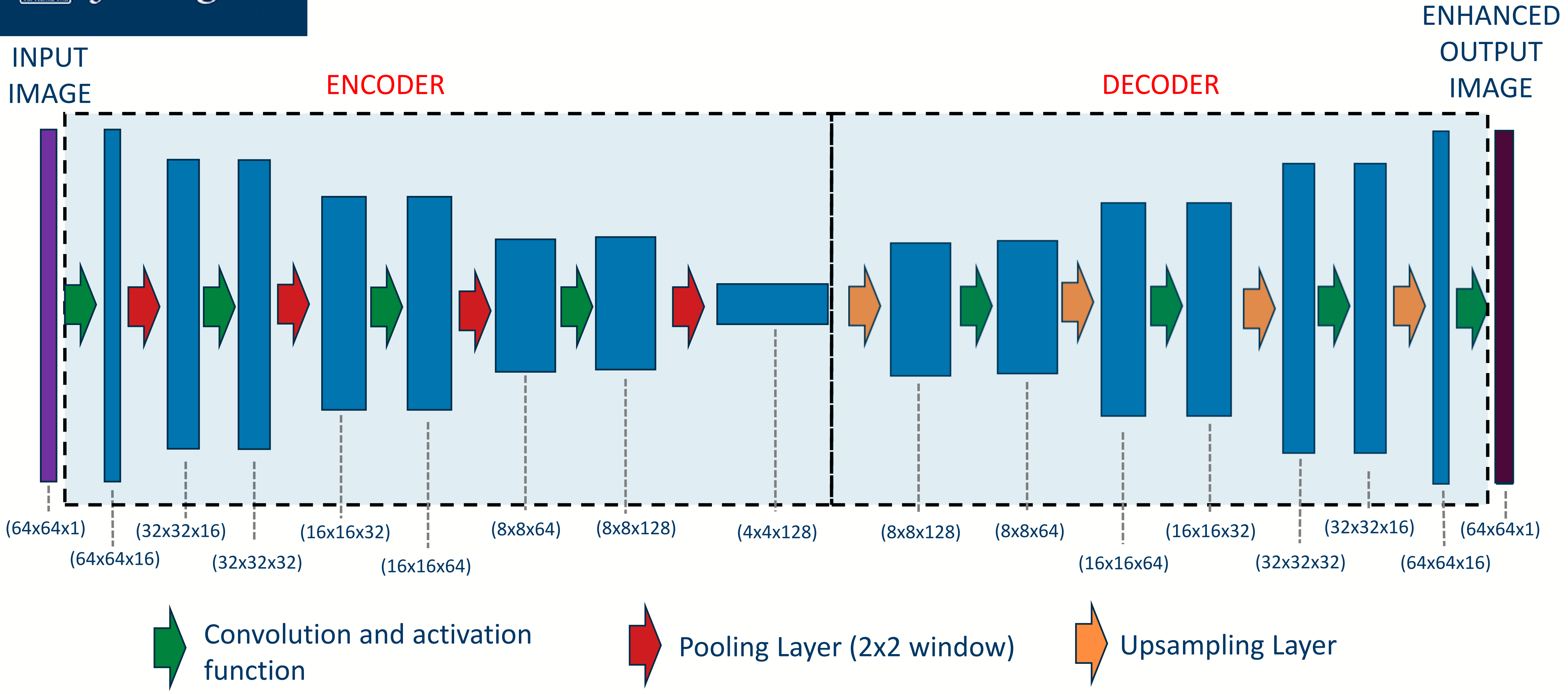
$$\begin{array}{|c|c|c|}\hline w_{0,0} & w_{1,0} & w_{2,0} \\ \hline w_{0,1} & w_{1,1} & w_{2,1} \\ \hline w_{0,2} & w_{1,2} & w_{2,2} \\ \hline \end{array} + b$$

Convolution in CNNs [2]

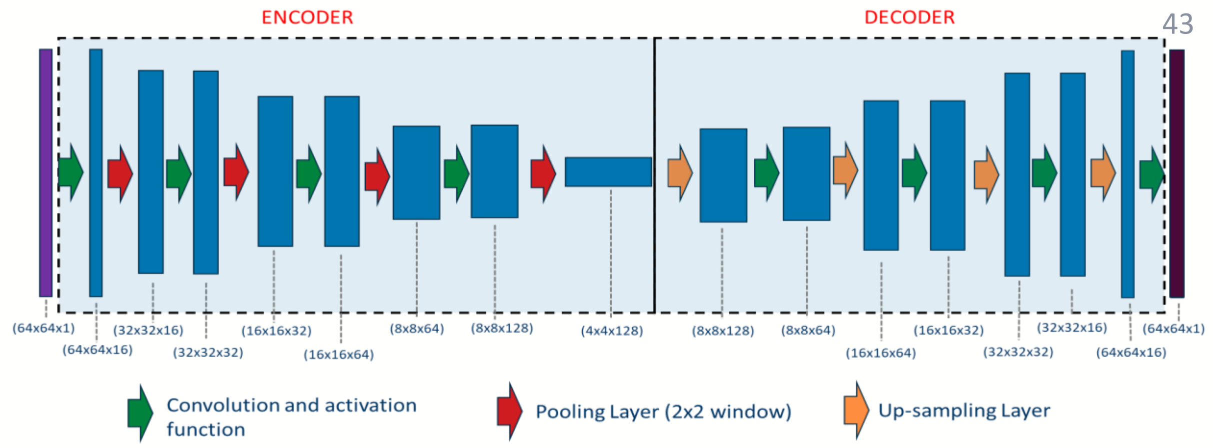
- As well as task-specific learning, CNNs also allow for complex hierarchical feature extraction using multiple layers.
  - – Top layers extract simple features such as edges.
  - – Deeper layers can extract complex features, combining information of the feature maps from the previous layer (e.g. boxes).



CNN: Encoder-Decoder Architecture

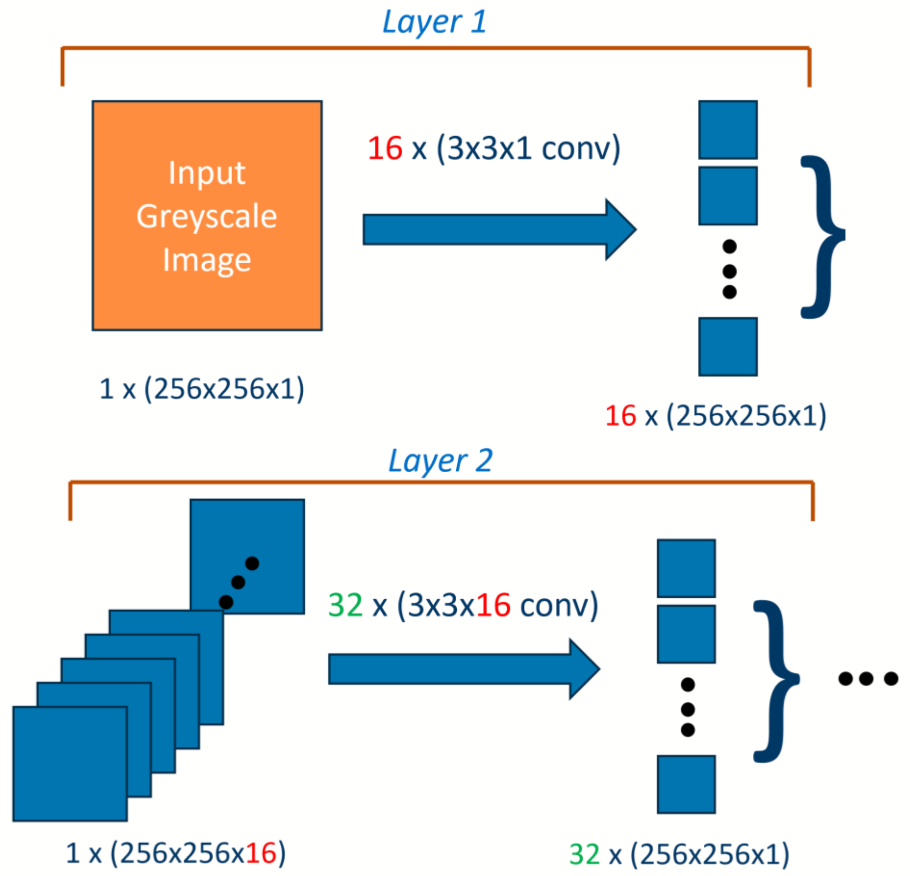


Convolutional Neural Networks (CNNs) [1]



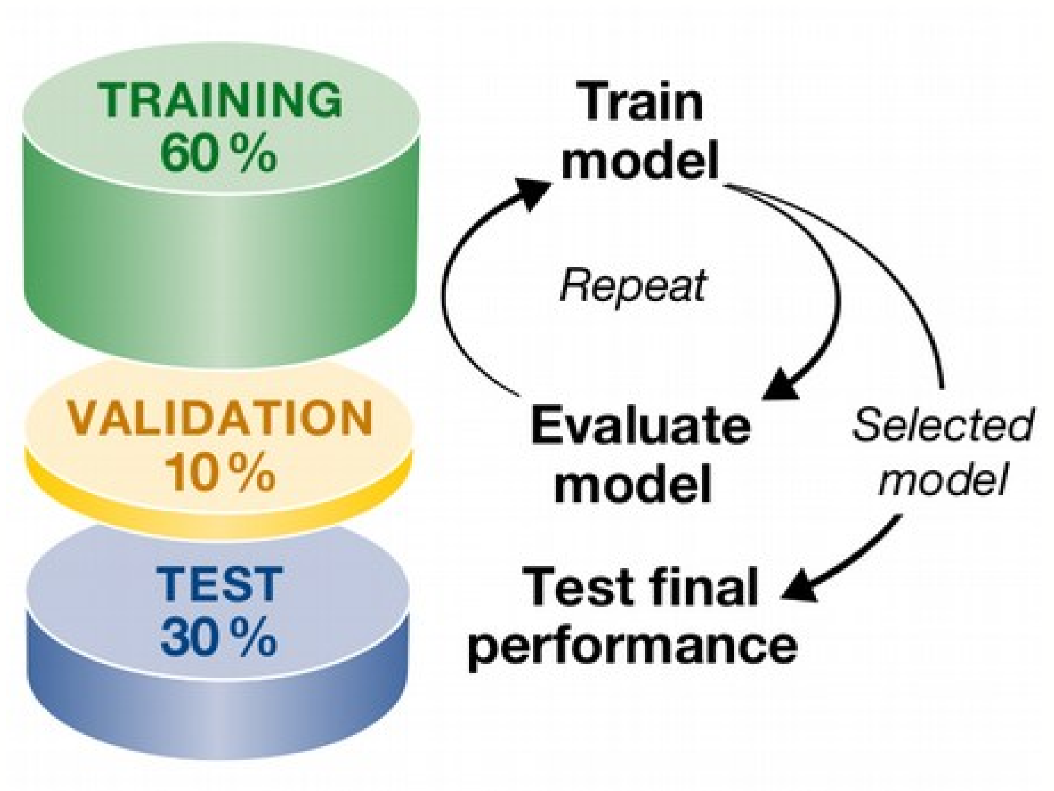
Convolutional Layer:

- Primary Layer in a CNN.
- Used to extract and implement image features by using convolutional kernels.
- Kernel parameters converge to their final state over the course of training.
- Often on the order of 100's of filters per layer - meaning even simple models have 10^5 - 10^6 total parameters to learn.



Training a Model [1]: Datasets

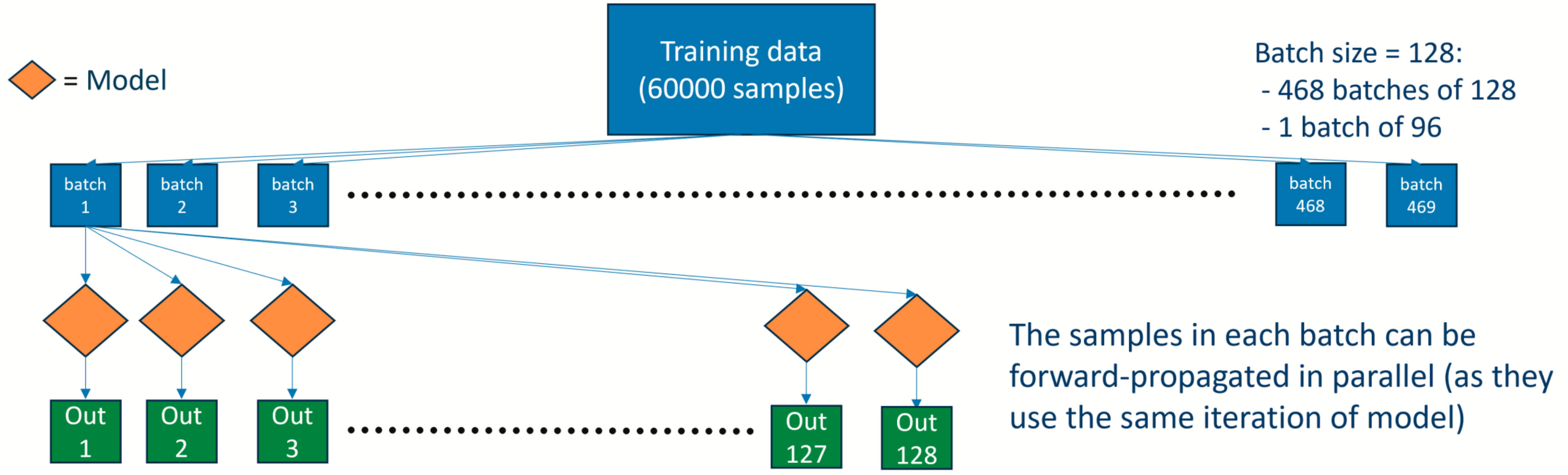
- Model Methodology:
  - – Minimise complexity
  - – Minimise loss (maximise accuracy)
  - – Maximize generalisation
- Having a *large, diverse* dataset for the model to learn from is key to successfully training a generalised model.
- Unsuitable datasets are often the limiting factor when training a model.
- Datasets are randomly shuffled and split up into three parts:
  - **Training data**
  - **Testing data**
  - **Validation data**



Training a Model [2]: Forward Propagation

- The training set is split into batches (batch size is dependent on the memory available during training).
- For each batch:

1. Each sample is forward-propagated through the model to produce an output.



The samples in each batch can be forward-propagated in parallel (as they use the same iteration of model)

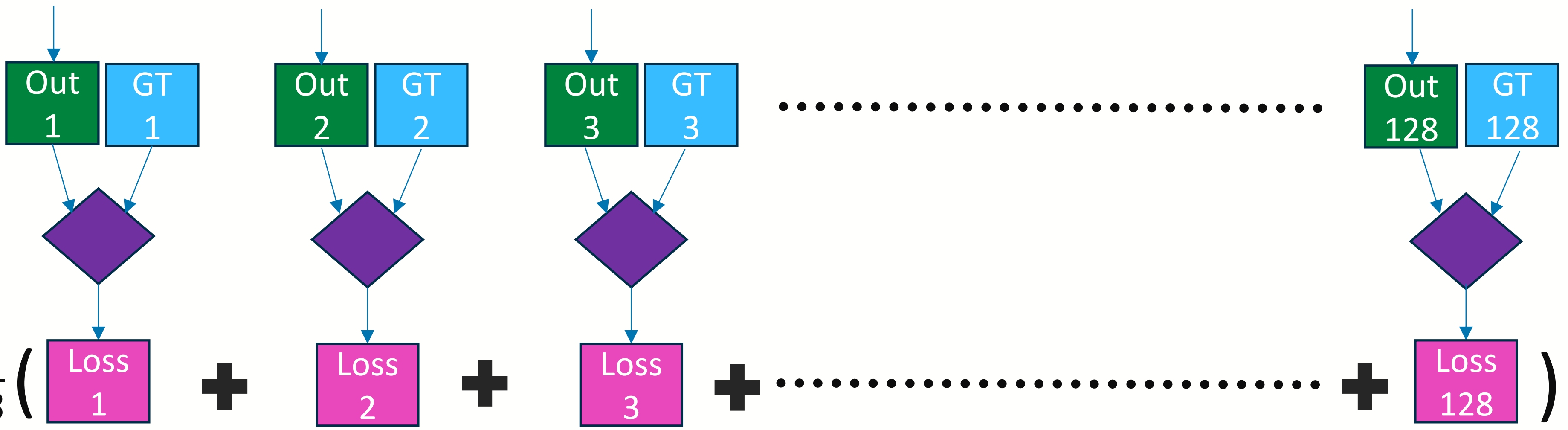
Training a Model [3]: Loss Calculation

2. Outputs are compared to the ground truth labels using a loss function. An average batch loss is calculated from all batch samples.

- A loss function is a differentiable function that numerically compares the generated output and ground truth to measure the similarity.
- One example is the mean-squared error (MSE) loss: MSE = $\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$

- y_i is the GT pixel value
- $\hat{y}_i$ is the out pixel value
- n is the number of pixels in the image

◆ = Loss Function

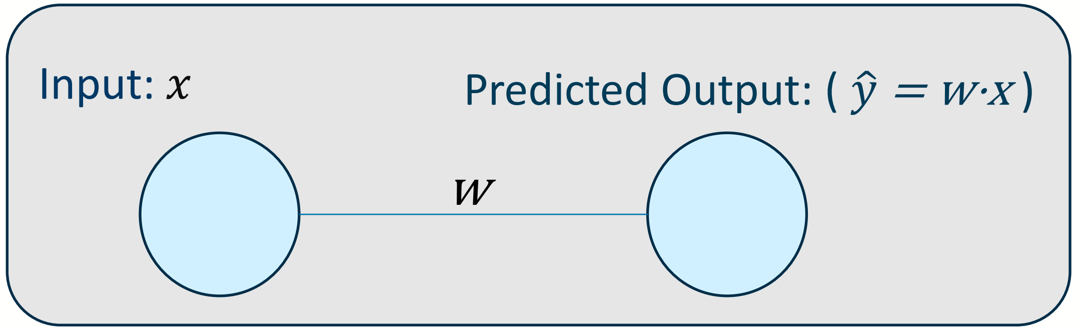


$$\textbf{Batch Loss} = \frac{1}{128}\left(\text{Loss 1} + \text{Loss 2} + \text{Loss 3} + \cdots + \text{Loss 128}\right)$$

Training a Model [4]: Gradient Calculation

3. The average loss is then backward-propagated through the model, calculating gradients of the loss w.r.t. the parameters of the model.

- For simplicity, consider a simple neural network with two nodes using MSE loss:

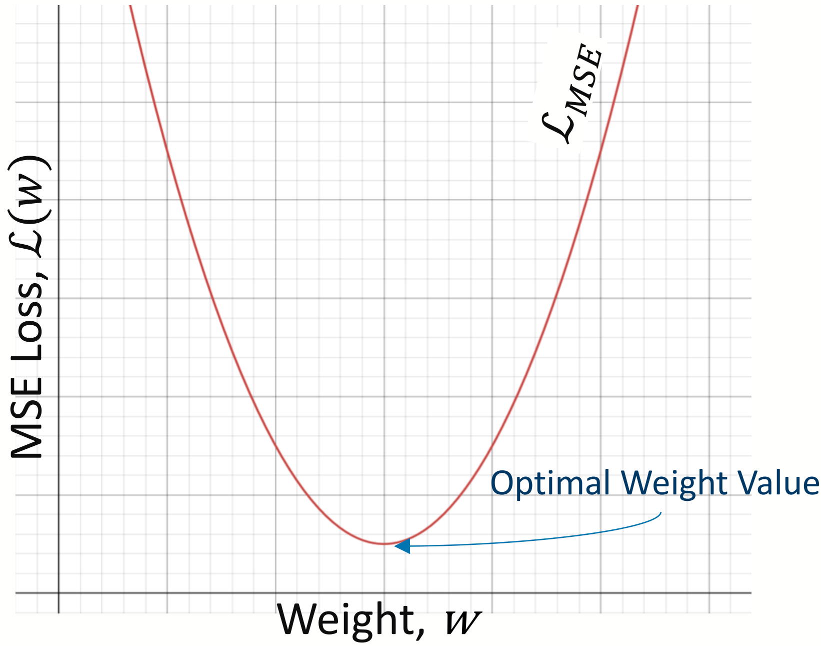


$$\mathcal{L}_{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

$$\mathcal{L}_{MSE} = \frac{1}{n}\sum_{i=1}^{n}(w{\cdot}x_i - y_i)^2$$

- y is the GT output value
- $\hat{y} = w{\cdot}x$ is the predicted output value
- n is the number of items in each batch

Example: minimising MSE loss wrt to weight



- We first calculate the gradient of the loss $\mathcal{L}$ with respect to the parameter of the model (w). We do this using the chain rule:

$$\frac{\partial \mathcal{L}}{\partial w} = \frac{\partial \mathcal{L}}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial w}$$

$$\frac{\partial \mathcal{L}}{\partial w} = \frac{2}{n}\sum_{i=1}^{n}(w{\cdot}x_i - y_i) \cdot x_i$$

Training a Model [5]: Optimisation

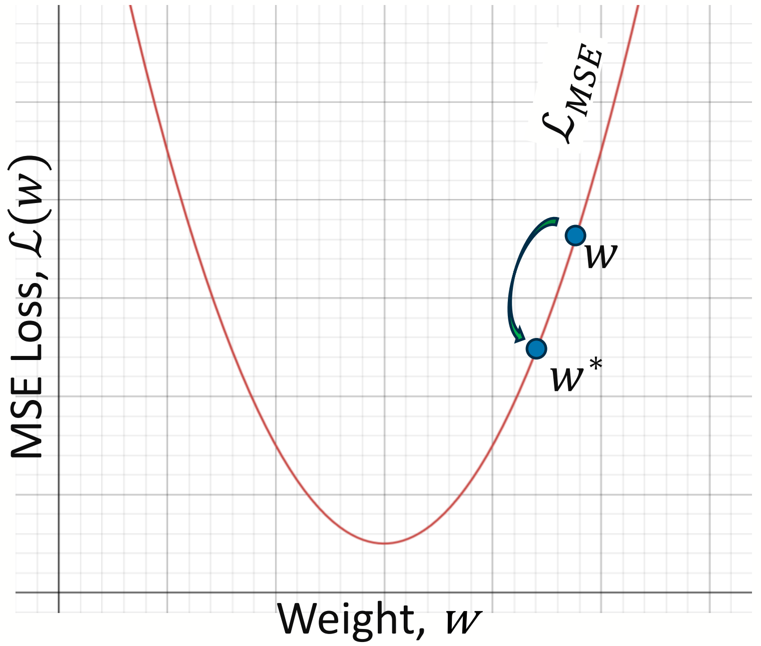
4. An optimisation algorithm then calculates new parameter values based on the gradients.

- One of the simplest algorithms is gradient descent, which directly moves the weights towards the minimum.
- The negative gradient gives the direction towards the minimum, with a pre-defined parameter called the learning rate, α, determining the step size.
- The new weight, *w**, is calculated by:

$$w^* = w - \alpha \frac{\partial \mathcal{L}}{\partial w}$$

$$w^* = w - \alpha \frac{1}{2n} \sum_{i=1}^{n} (w x_i - y_i) \cdot x_i$$

Example: minimising MSE loss wrt to weight



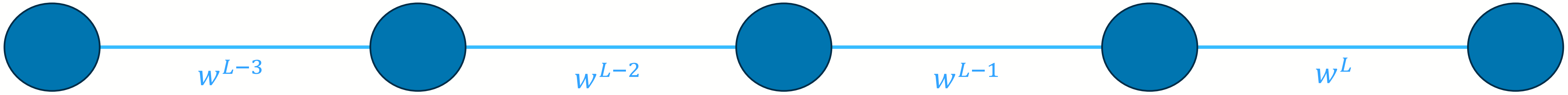
University of Glasgow

Training a Model [6]: Extending to >1 Layers

Input: x_i

$a_i^{L-3} = w^{L-2} \cdot x_i$ $\quad$ $a_i^{L-2} = w^{L-2} \cdot a_i^{L-3}$ $\quad$ $a_i^{L-1} = w^{L-1} \cdot a_i^{L-2}$

Predicted Output: $a_i^{L} = \hat{y}_i = w^{L} \cdot a_i^{L-1}$



Red = Layer outputs obtained from forward propagation

Blue = Old weights

Green = New weights

$$w^{L} = w^{L} - \alpha \frac{\partial \mathcal{L}}{\partial w^{L}}$$

$$w^{L} = w^{L} - \alpha \frac{\partial a_i^{L}}{\partial w^{L}} \cdot \frac{\partial \mathcal{L}}{\partial a_i^{L}}$$

$$w^{L-1} = w^{L-1} - \alpha \frac{\partial \mathcal{L}}{\partial w^{L-1}}$$

$$w^{L-1} = w^{L-1} - \alpha \frac{\partial a_i^{L-1}}{\partial w^{L-1}} \cdot \frac{\partial \mathcal{L}}{\partial a_i^{L-1}}$$

$$w^{L-1} = w^{L-1} - \alpha \frac{\partial a_i^{L-1}}{\partial w^{L-1}} \cdot \frac{\partial a_i^{L}}{\partial a_i^{L-1}} \cdot \frac{\partial \mathcal{L}}{\partial a_i^{L}}$$

$$w^{L-2} = w^{L-2} - \alpha \frac{\partial \mathcal{L}}{\partial w^{L-2}}$$

$$w^{L-2} = w^{L-2} - \alpha \frac{\partial a_i^{L-2}}{\partial w^{L-2}} \cdot \frac{\partial \mathcal{L}}{\partial a_i^{L-2}}$$

$$w^{L-2} = w^{L-2} - \alpha \frac{\partial a_i^{L-2}}{\partial w^{L-2}} \cdot \frac{\partial a_i^{L-1}}{\partial a_i^{L-2}} \cdot \frac{\partial a_i^{L}}{\partial a_i^{L-1}} \cdot \frac{\partial \mathcal{L}}{\partial a_i^{L}}$$

$$\boxed{\begin{aligned} \mathcal{L}_{MSE} &= \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2 \\ \mathcal{L}_{MSE} &= \frac{1}{n} \sum_{i=1}^{n} (w^{L} \cdot a_i^{L-1} - y_i)^2 \\ \frac{\partial \mathcal{L}}{\partial a_i^{L}} &= \frac{2}{n} \sum_{i=1}^{n} (w^{L} \cdot a_i^{L-1} - y_i) \end{aligned}}$$

$$w^{L-3} = w^{L-3} - \alpha \frac{\partial \mathcal{L}}{\partial w^{L-3}}$$

$$w^{L-3} = w^{L-3} - \alpha \frac{\partial a_i^{L-3}}{\partial w^{L-3}} \cdot \frac{\partial \mathcal{L}}{\partial a_i^{L-3}}$$

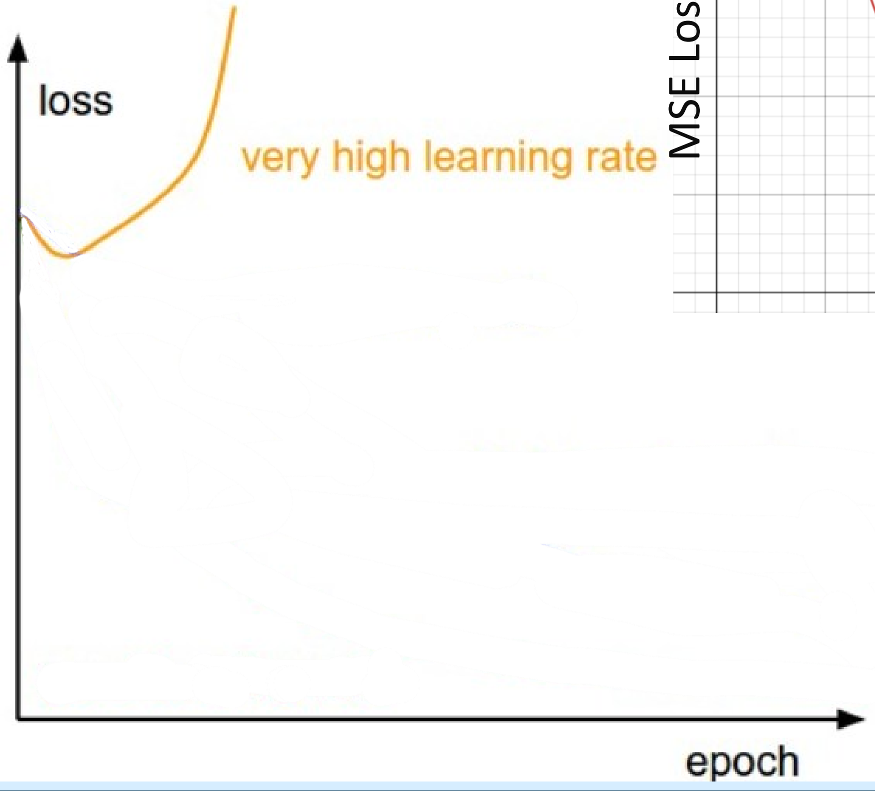
$$w^{L-3} = w^{L-3} - \alpha \frac{\partial a_i^{L-3}}{\partial w^{L-3}} \cdot \frac{\partial a_i^{L-2}}{\partial a_i^{L-3}} \cdot \frac{\partial a_i^{L-1}}{\partial a_i^{L-2}} \cdot \frac{\partial a_i^{L}}{\partial a_i^{L-1}} \cdot \frac{\partial \mathcal{L}}{\partial a_i^{L}} = w^{L-3} - \overset{\text{Learning Rate}}{\alpha} \sum_{i=1}^{n} [\underset{\text{Input}}{x_i} \cdot w^{L-2} \cdot w^{L-1} \cdot w^{L} \cdot \underset{\text{Batch Size}}{\frac{2}{n}} (\overset{\text{Generated Output}}{w^{L} \cdot a_i^{L-1}} - \overset{\text{Label}}{y_i})]$$

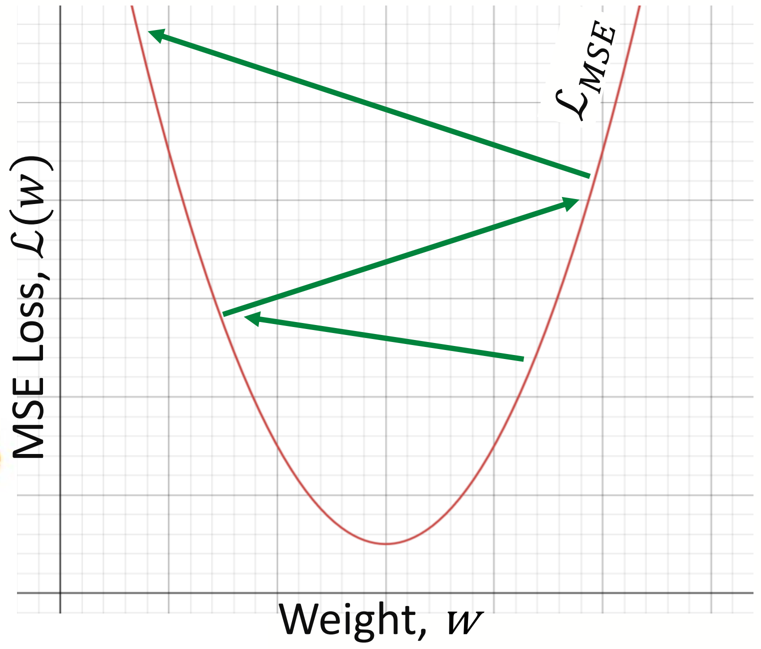
-- BACKUP SLIDES --

Model Evaluation [1]: Training Loss

5. Each pass of the full training dataset (all batches) is called an epoch. Models are often run for hundreds of epochs for optimal accuracy.

- Training the model over many epochs allows more time for the model parameters to converge towards their minima.
- We can evaluate the model over its training by calculating the loss and any other metrics at the end of each epoch.

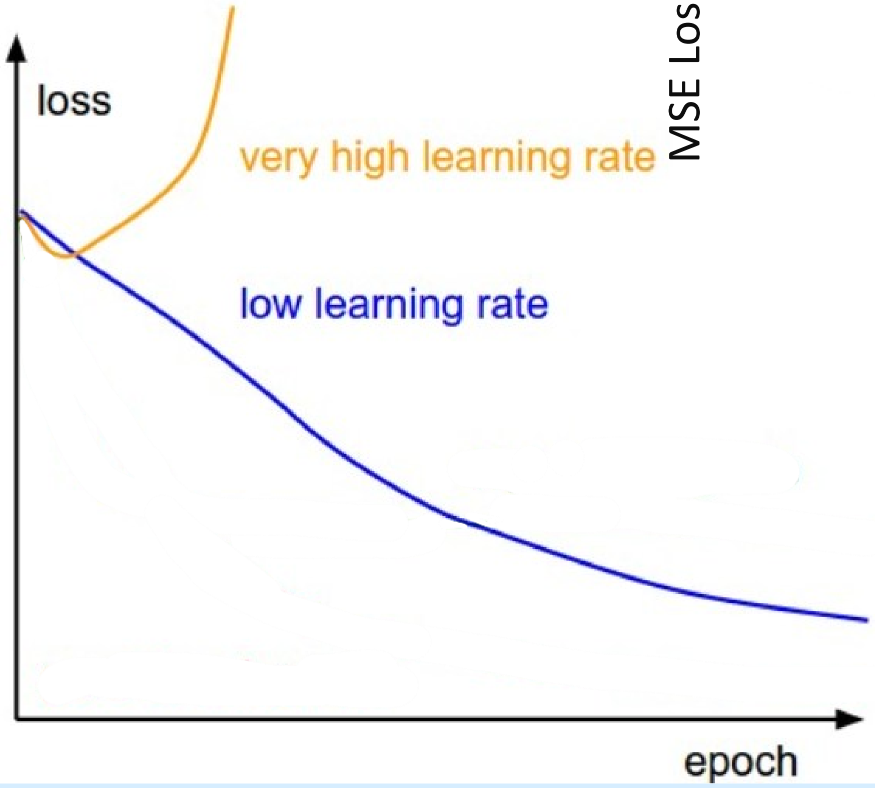


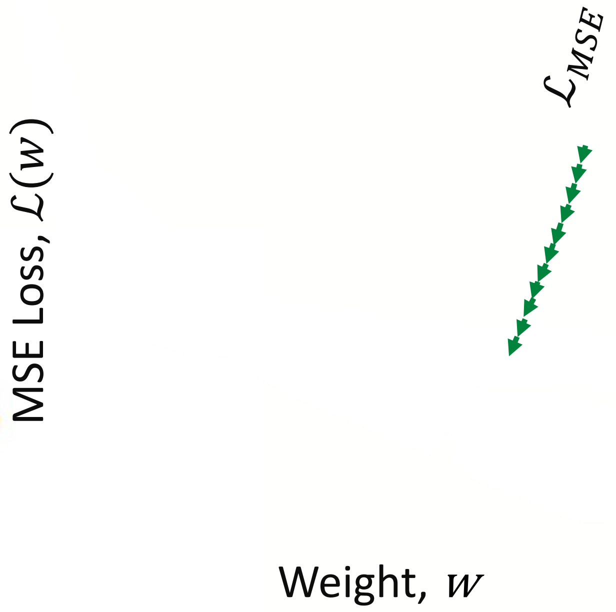


Model Evaluation [1]: Training Loss

5. Each pass of the full training dataset (all batches) is called an epoch. Models are often run for hundreds of epochs for optimal accuracy.

- Training the model over many epochs allows more time for the model parameters to converge towards their minima.
- We can evaluate the model over its training by calculating the loss and any other metrics at the end of each epoch.

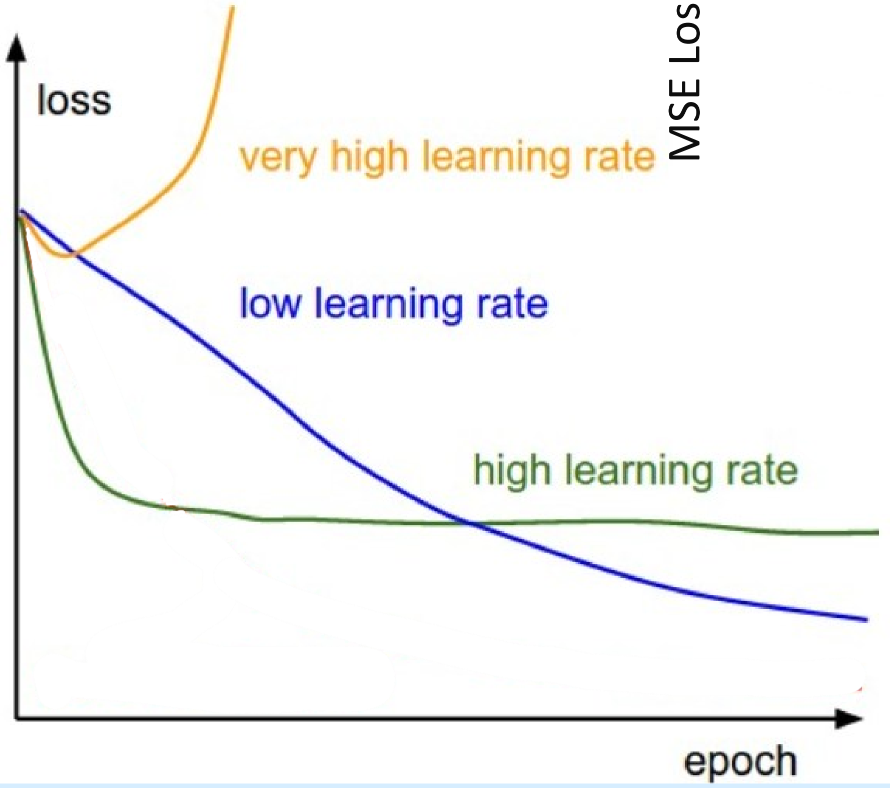


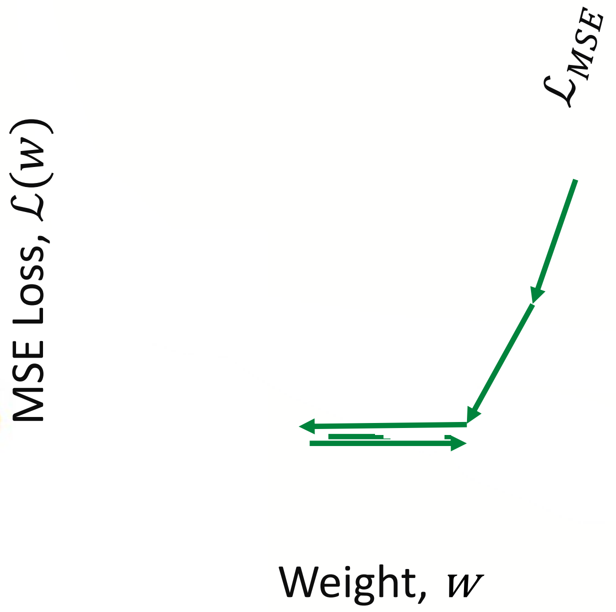


Model Evaluation [1]: Training Loss

5. Each pass of the full training dataset (all batches) is called an epoch. Models are often run for hundreds of epochs for optimal accuracy.

- Training the model over many epochs allows more time for the model parameters to converge towards their minima.
- We can evaluate the model over its training by calculating the loss and any other metrics at the end of each epoch.

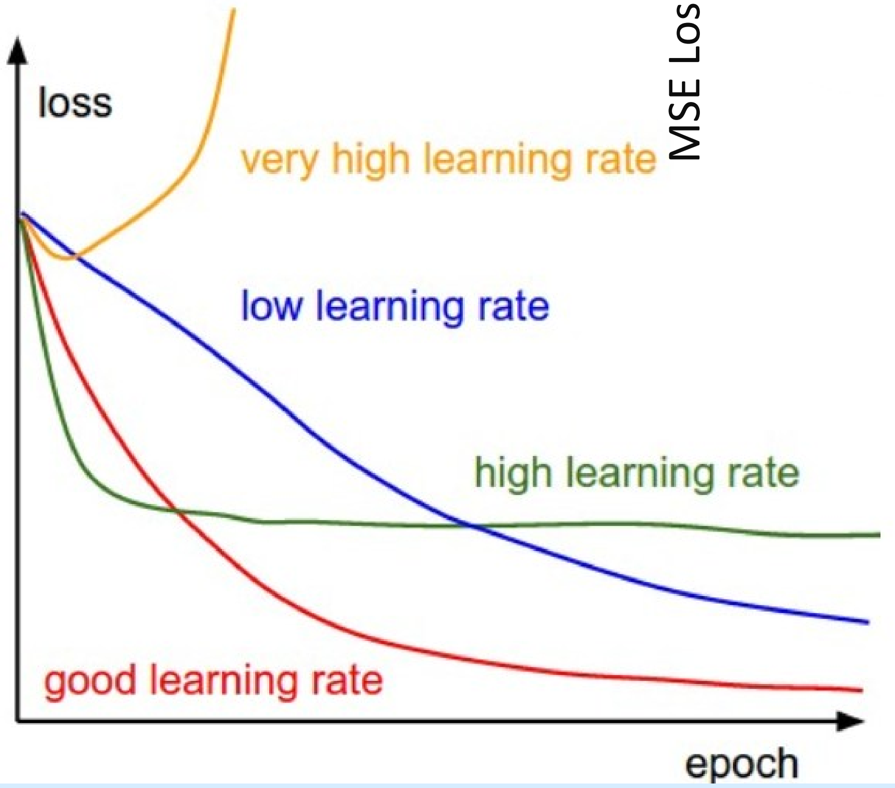


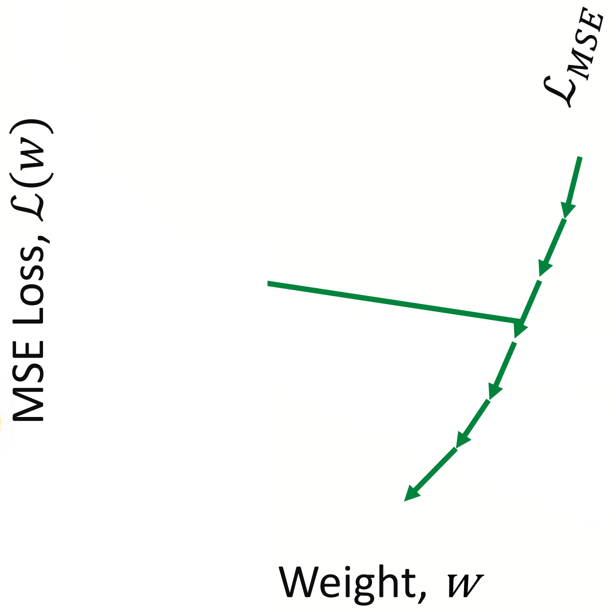


Model Evaluation [1]: Training Loss

5. Each pass of the full training dataset (all batches) is called an epoch. Models are often run for hundreds of epochs for optimal accuracy.

- Training the model over many epochs allows more time for the model parameters to converge towards their minima.
- We can evaluate the model over its training by calculating the loss and any other metrics at the end of each epoch.

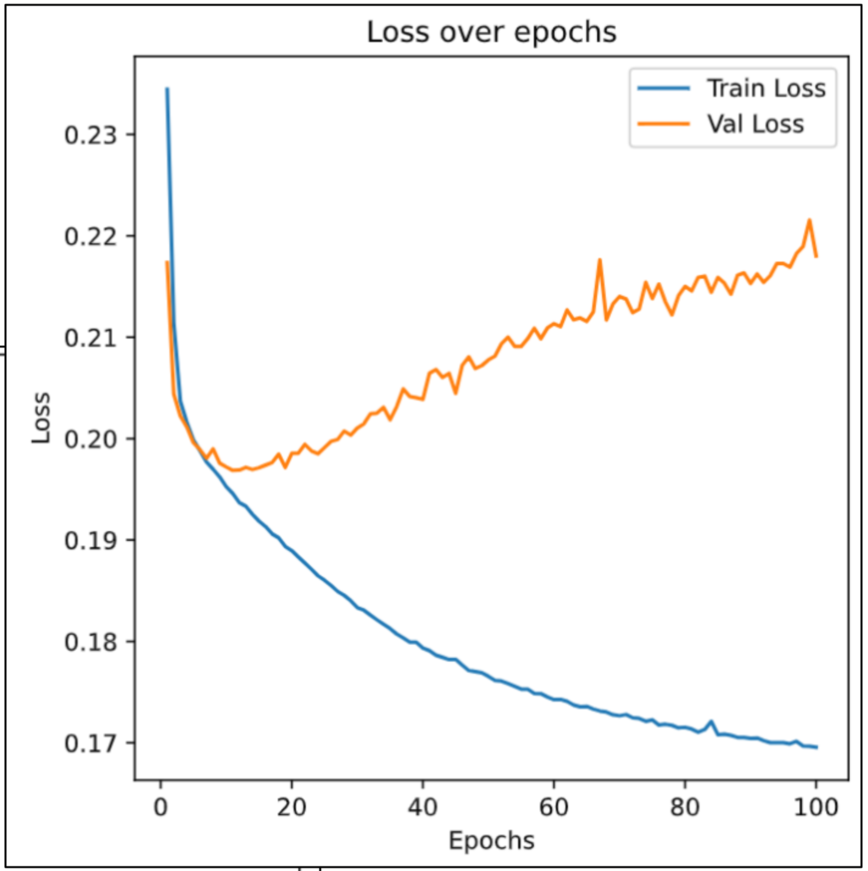


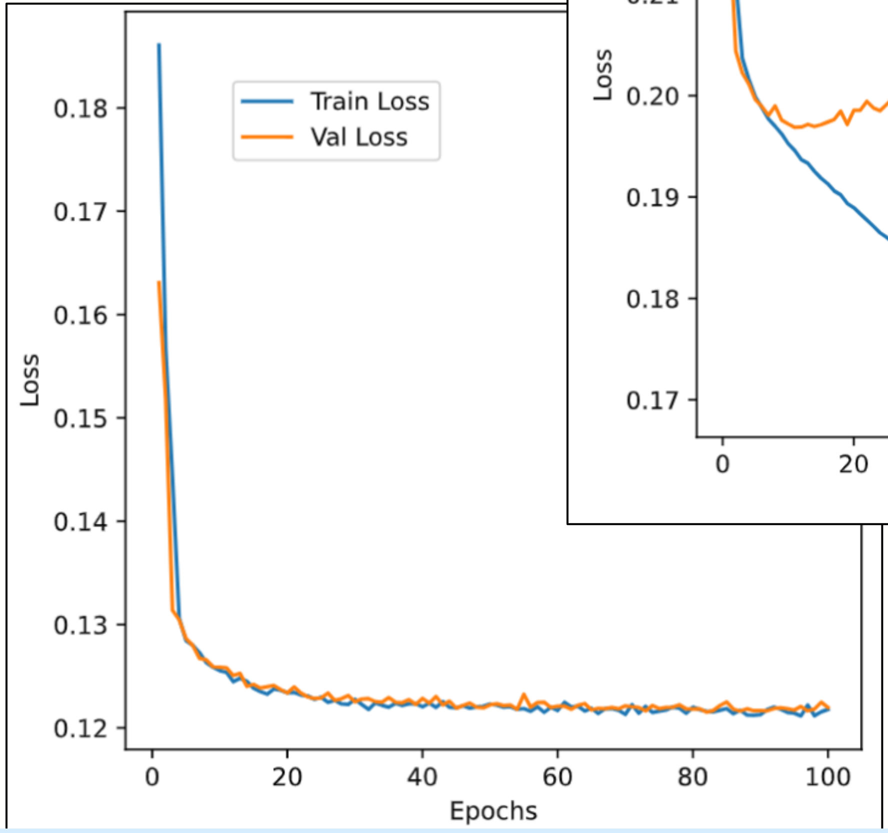


Model Evaluation [2]: Validation Loss

- Many iterations over the training data may lead to 'overtraining'.
- This is where the model does not generalise well to unseen data.
- We can evaluate model generalisation at the end of each epoch, using the unseen validation dataset.
- The trained model is finally evaluated using the test dataset.

Overfit Data

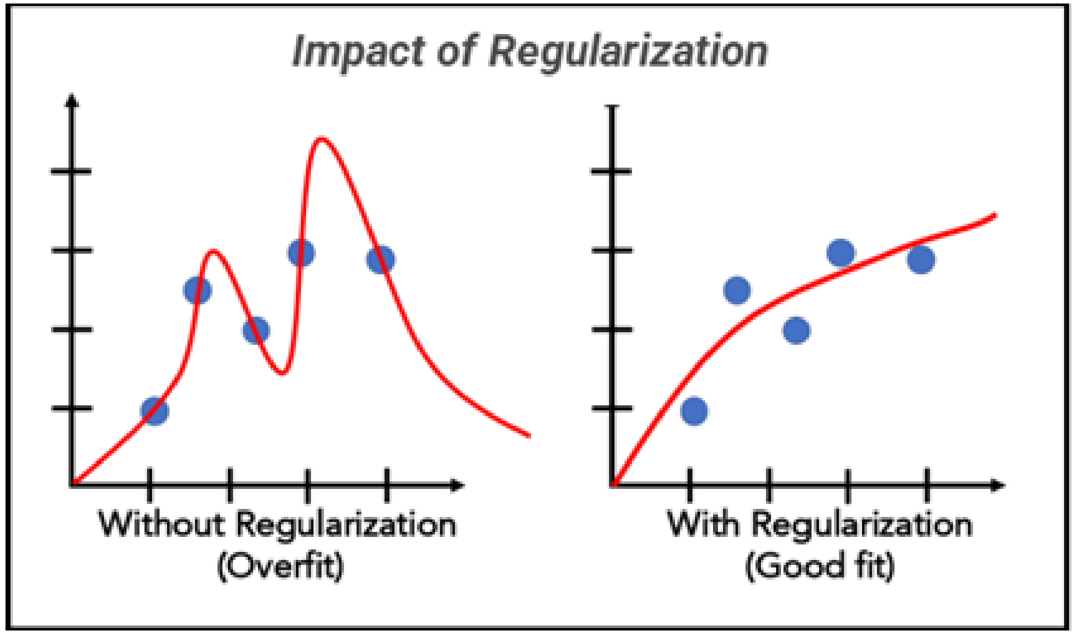




Well-fit Data

Model Regularisation

- Overfitting often means that the model is too complex as it learns the training data too well.
- However, models require a high complexity to learn hierarchical features.
- Therefore, the model is penalised during training, called regularisation.
- This involves:
  - Dropout
  - Batch Normalisation
  - Early stopping
  - Training Data augmentation



University of Glasgow

Summary of Machine Learning for I2I Translation

CNN Architecture

- Layered convolutions are used for hierarchical feature extraction.
- Activations are used to introduce non-linearity, reduce complexity and improve learning.
- Pooling is used to reduce resolution.
- Upsampling is used to increase resolution back up, with learned features implemented with more convolutions.
- Regularisation is used to penalise the model and ensure the model generalises well to unseen data.
- U-net and CGAN architectures are good for I2I tasks.

Model Training

- Training data is split into batches.
- For each batch:
  - Forward passed to generate outputs.
  - Average batch loss calculated.
  - Backpropagation: gradients calculated.
  - New parameters updated using gradients and optimiser
- A run of all batches is an epoch.
- Validation and training losses are used to evaluate the model after each epoch.
- The final model is evaluated on the test data.