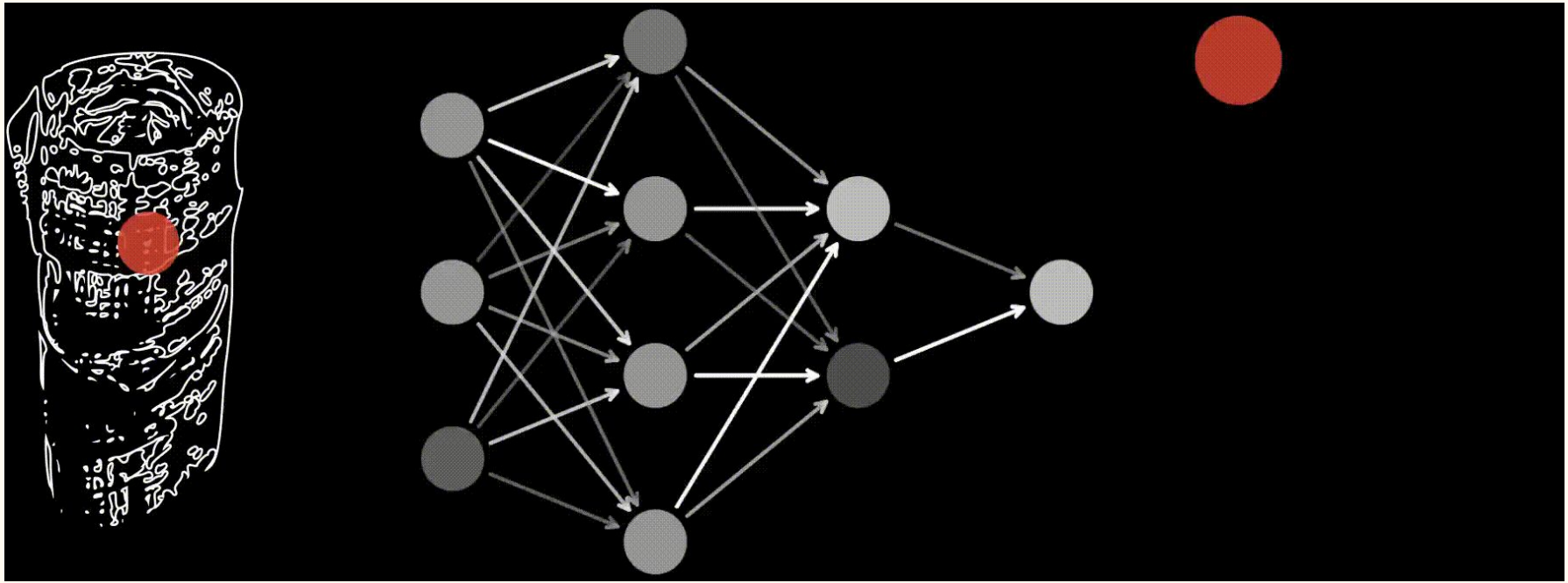
Applied Machine Learning 2023

Rediscovering Herculaneum

Andreas, Rasmus, Shaowen, Yasaswy, and Yi

Problem: Ink Detection Challenge

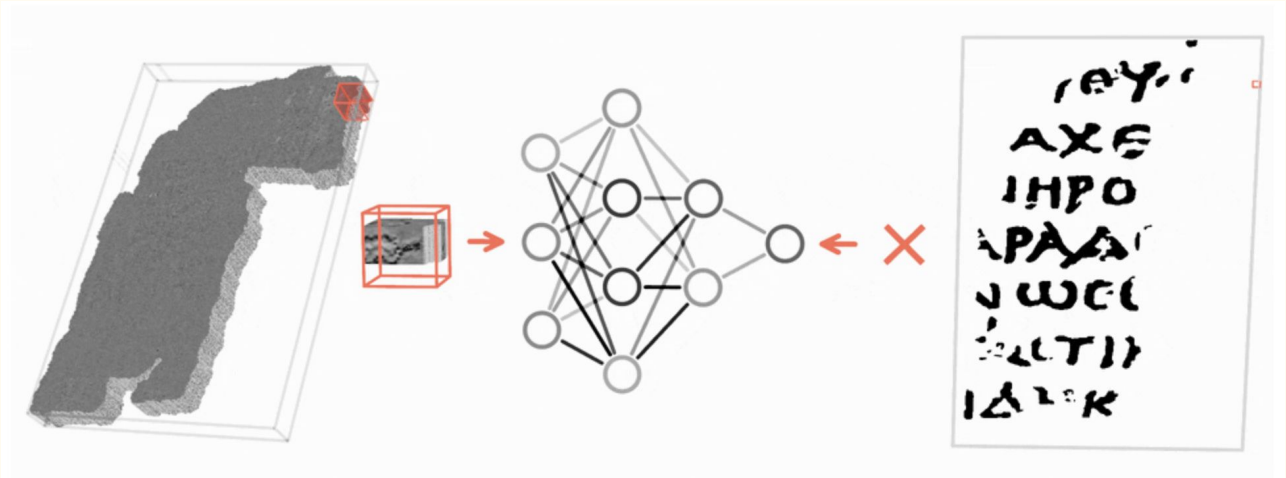
- Can you reconstruct the ink on a 2D image from 3D X-ray scans?



Data and labels

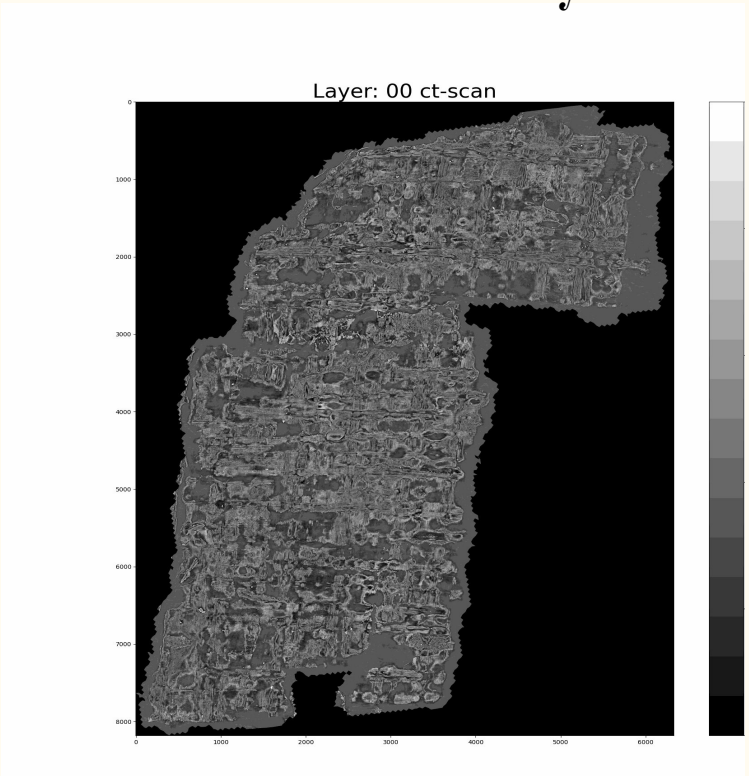
Data: Scanned images of a sheet - 65 layers of gigantic 2D pictures

Labels: Reconstructed ink based on an IR-scan - 2D image



Data visualized

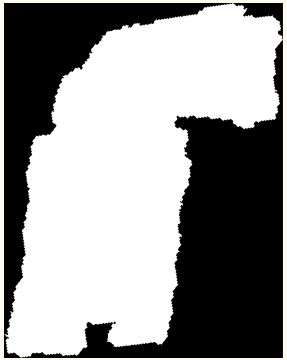
Visualization of z-layers



Real IR values

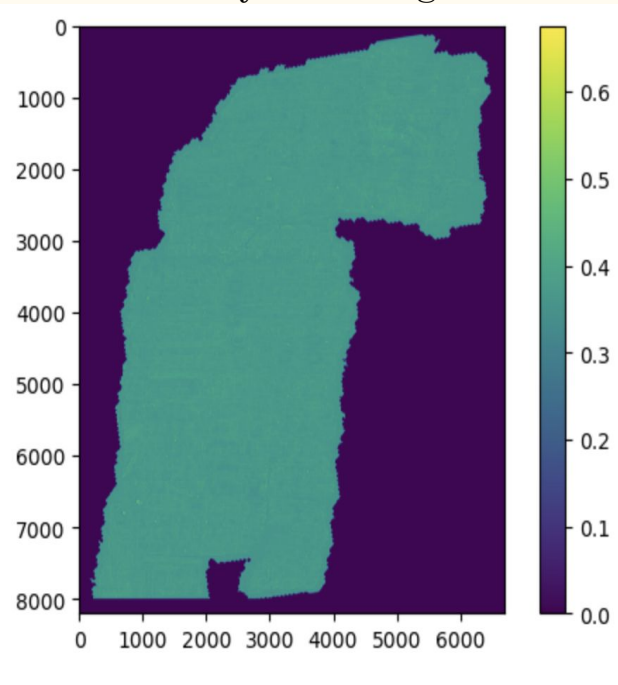
Ink Labels

Mask

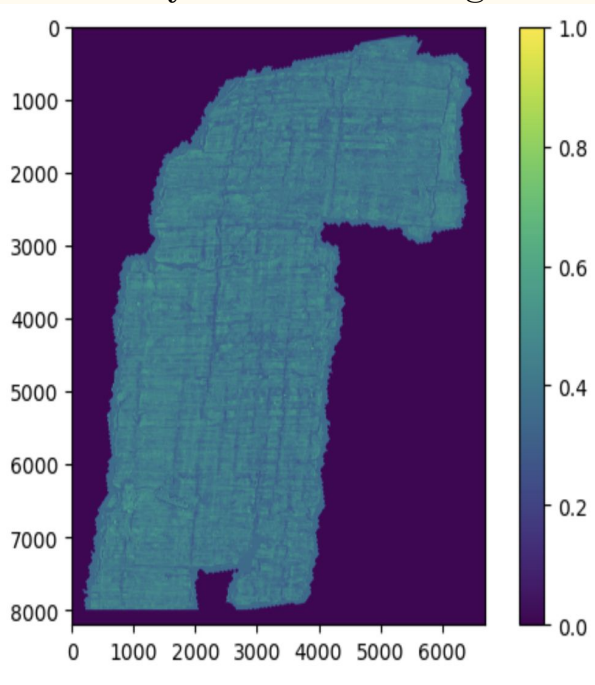


Data visualized 2

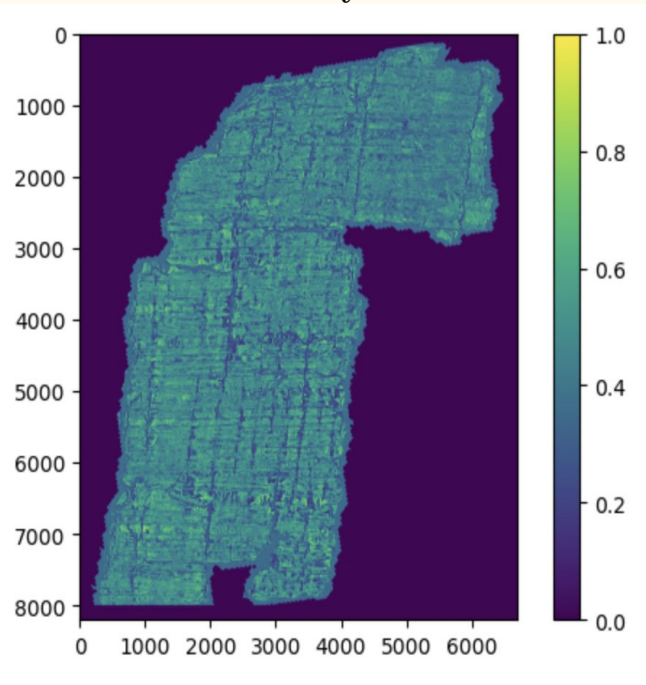
All layers averaged



Layers 22-35 averaged



Layer 30

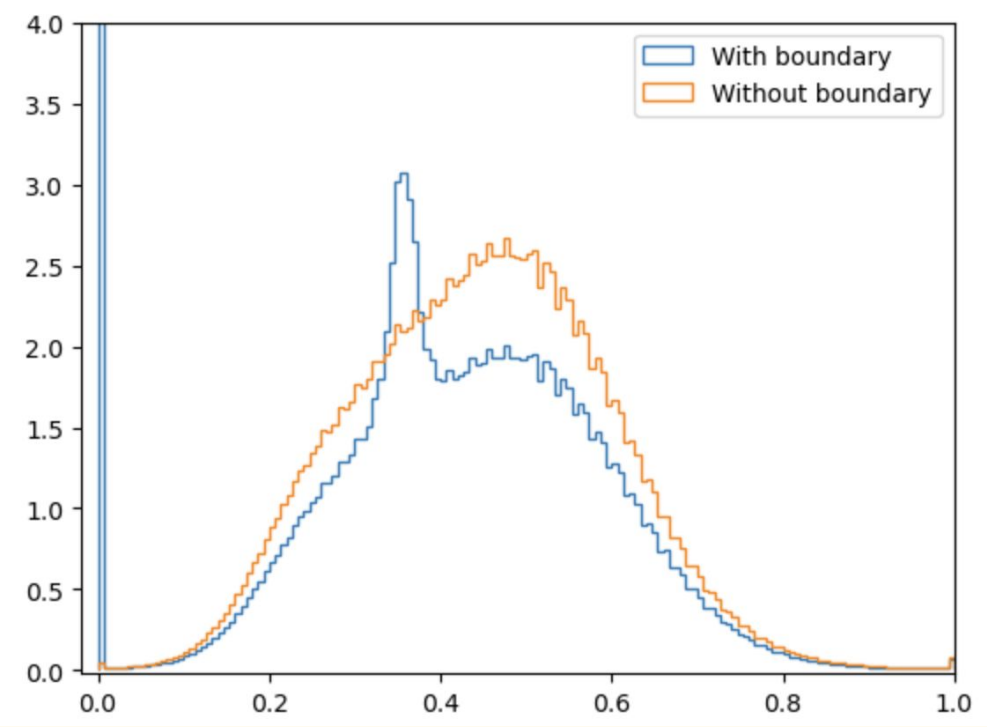


Preprocesing

Three main things we noticed:

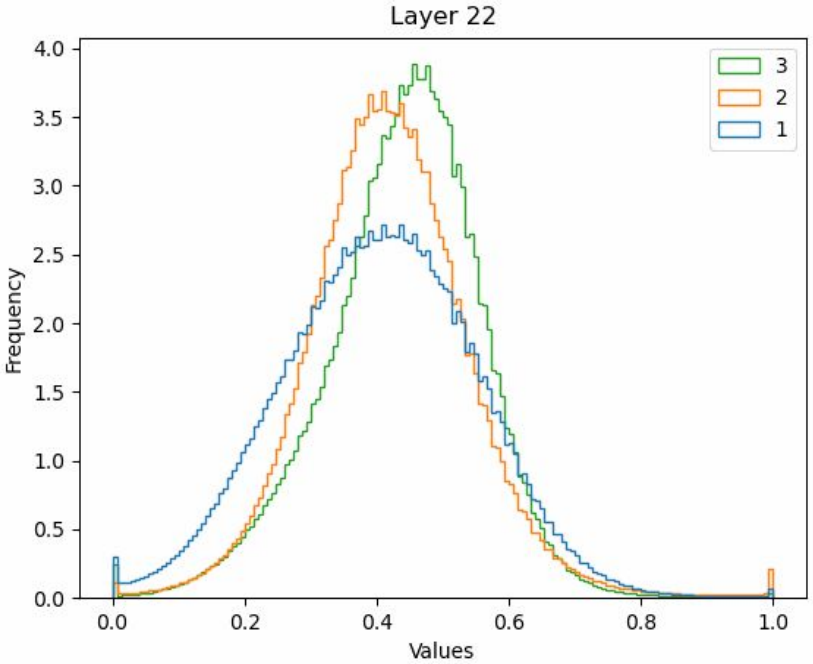
- Information in layer 40 and onwards is limited
- The boundary behaves very oddly
- X-ray layer distributions are different across images

Boundary behaviour

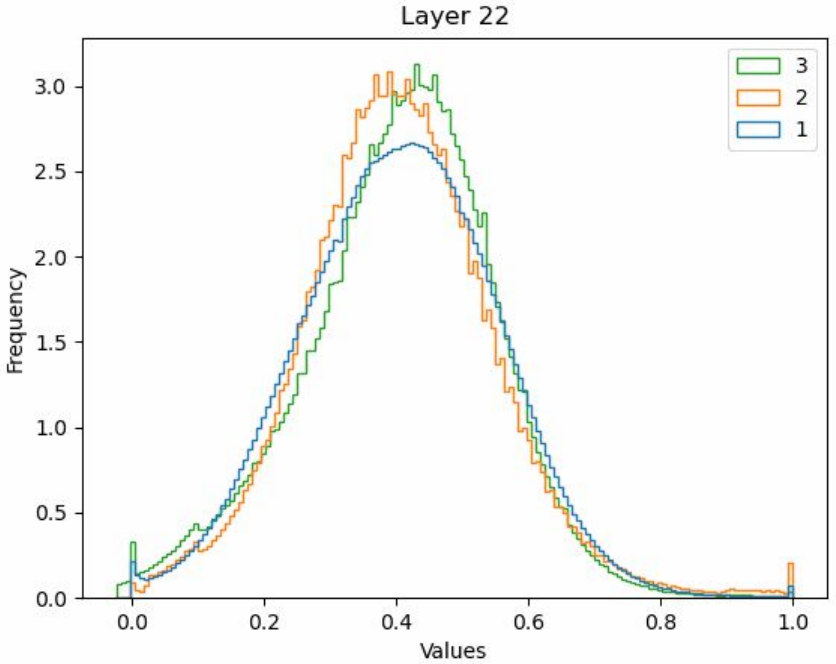


Calibration plots

Uncalibrated



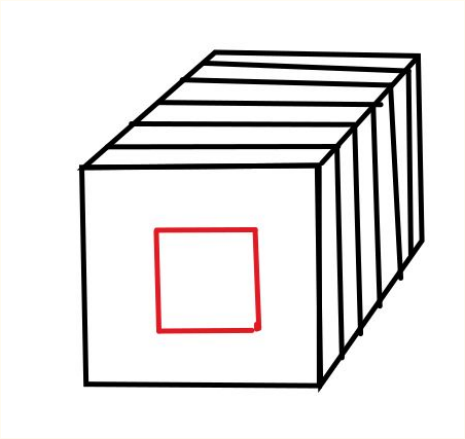
Calibrated



Approach and scoring

Two ways to approach

- Classification
- Image Segmentation



Train on Image 1 and 2 validate on Image 3

Most models were CNNs

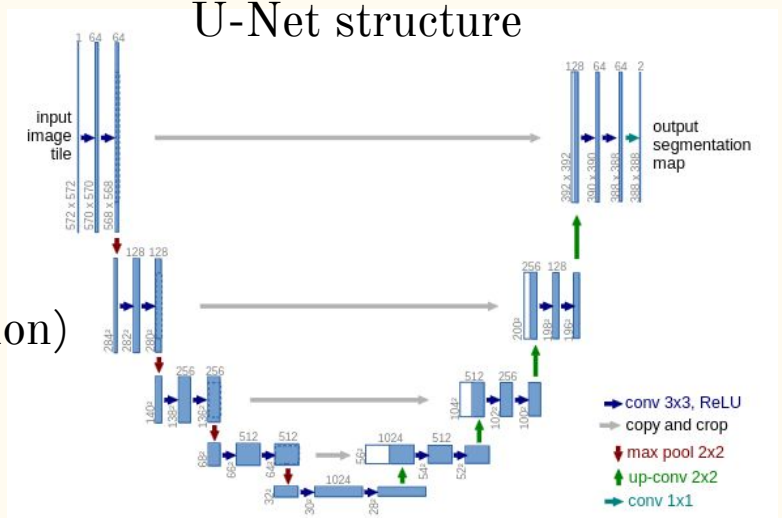
Competition score is F0.5 score

$$F_\beta = \frac{(1+\beta^2)\cdot \mathrm{P}\cdot \mathrm{R}}{\beta^2 \mathrm{P} + \mathrm{R}} \qquad \mathrm{P} = \frac{\mathrm{tp}}{\mathrm{tp}+\mathrm{fp}} \qquad \mathrm{R} = \frac{\mathrm{tp}}{\mathrm{tp}+\mathrm{fn}}$$

Models

- Encoder only models (Classification)
  - 2D CNN model
  - 3D CNN model
  - Encoder only Vision Transformer
- Encoder-decoder Models (Image Segmentation)
  - UNet model
  - Encoder-decoder Vision Transformer
- Simple Models
  - CatBoost
  - LDA

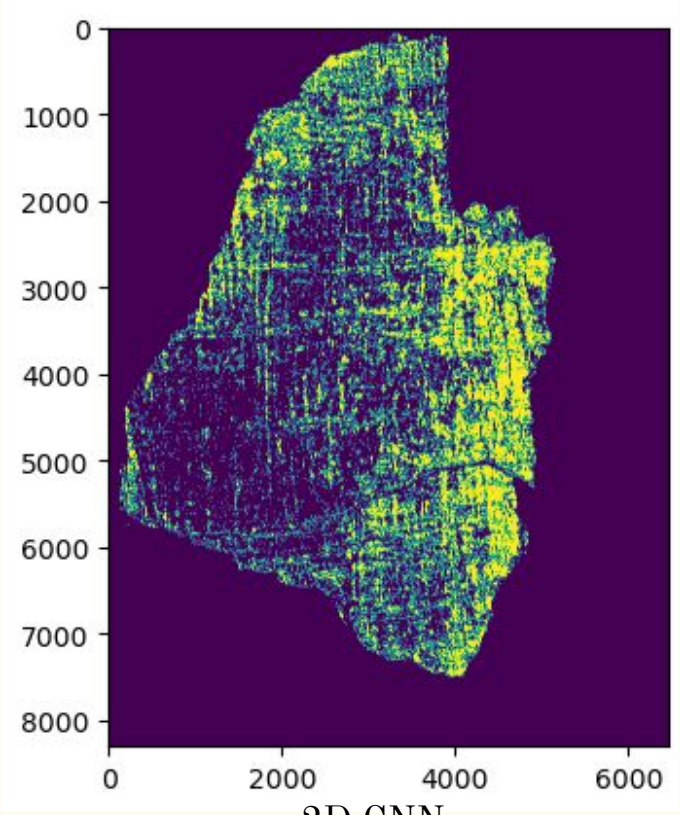
U-Net structure



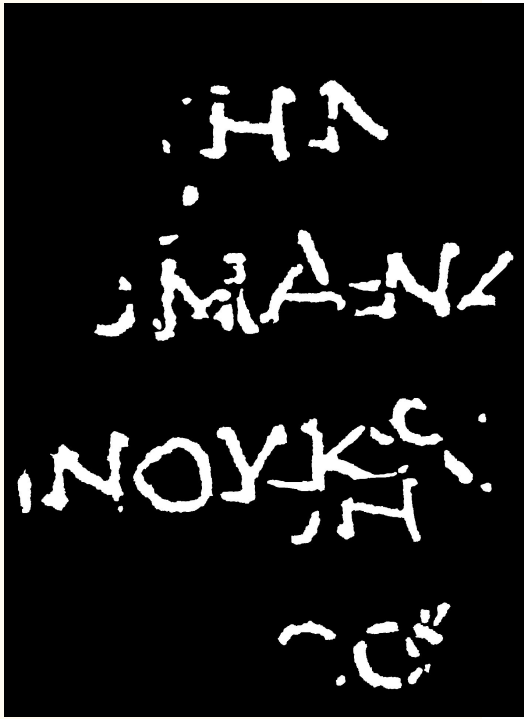
Threshold

- The models predict continuous spectrum in [0,1]
- Set a threshold
- Below the threshold, predict no ink
- Above the threshold, predict ink

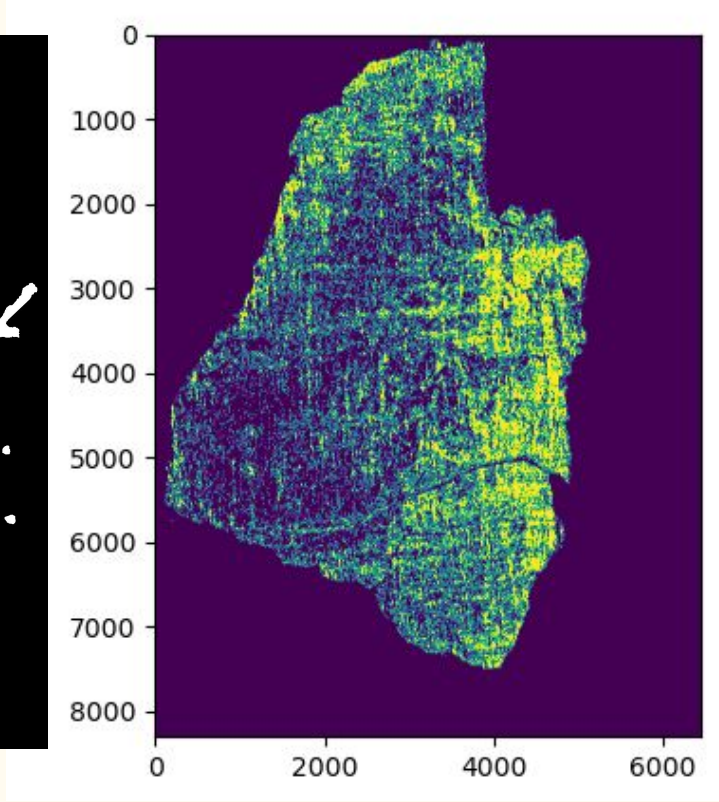
Reconstructions:



2D CNN

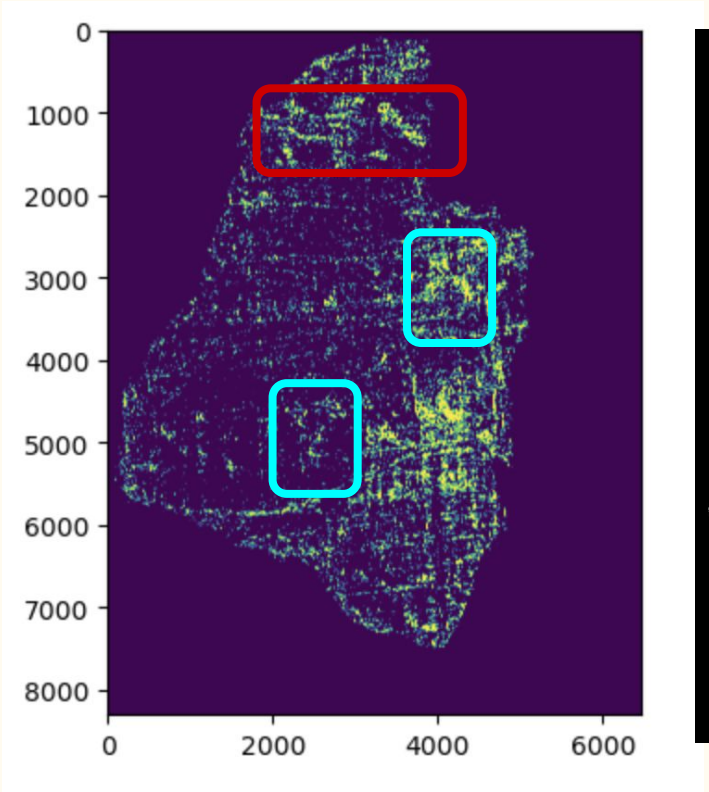


Threshold: 0.3

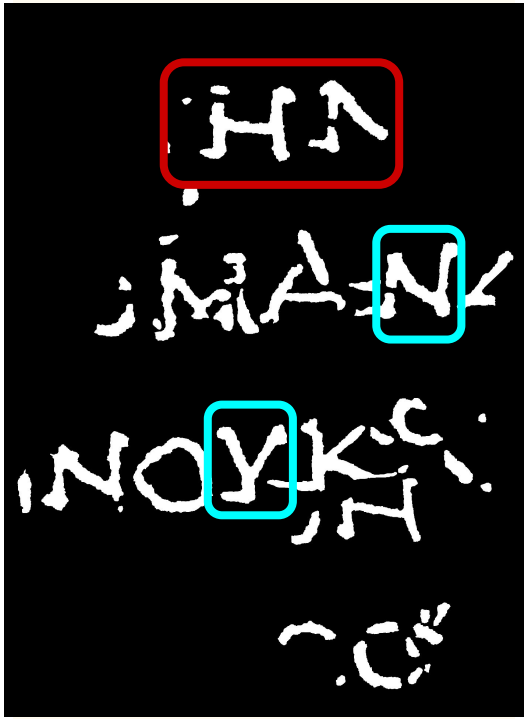


2D – 2D U-Net

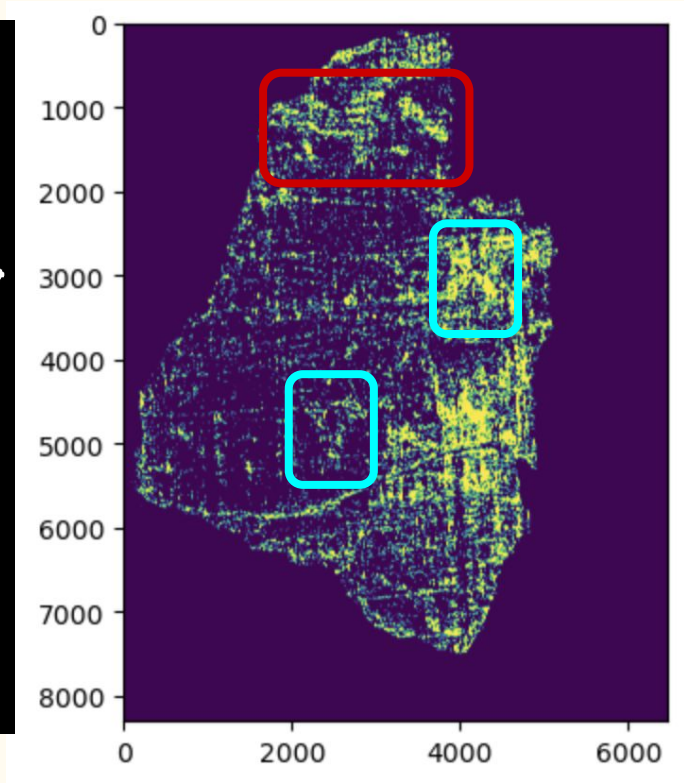
Reconstructions:



3D CNN

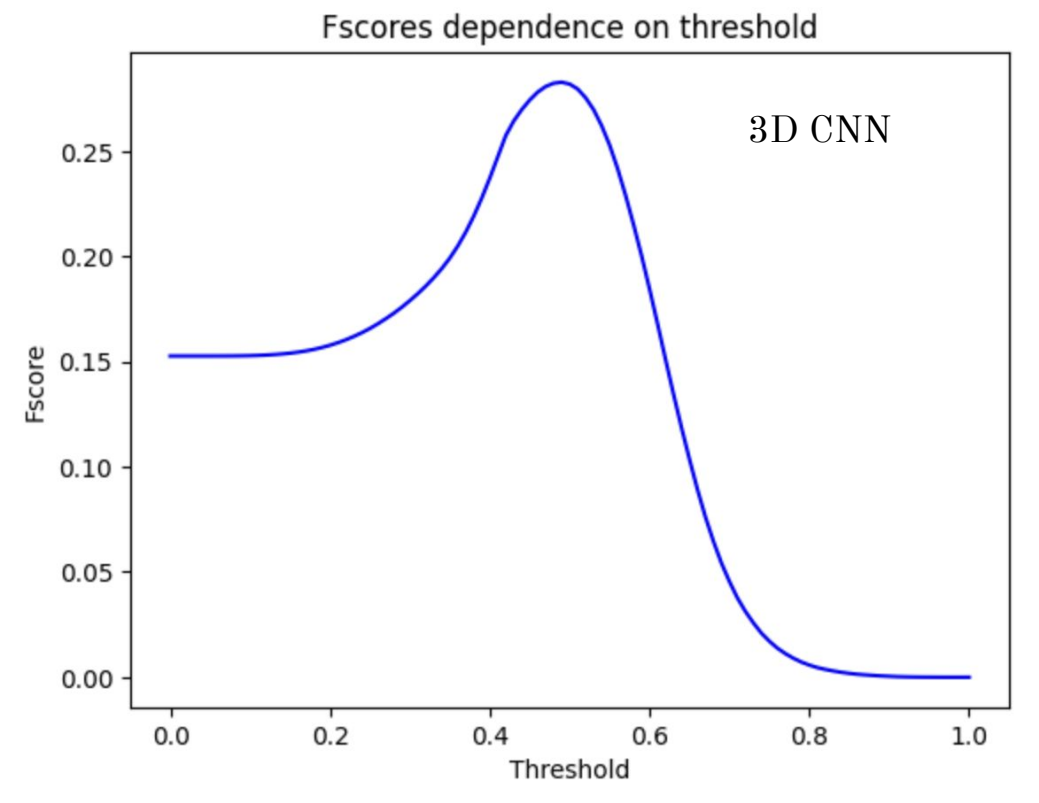


Threshold: 0.5



3D CNN

Importance of chosing right threshold



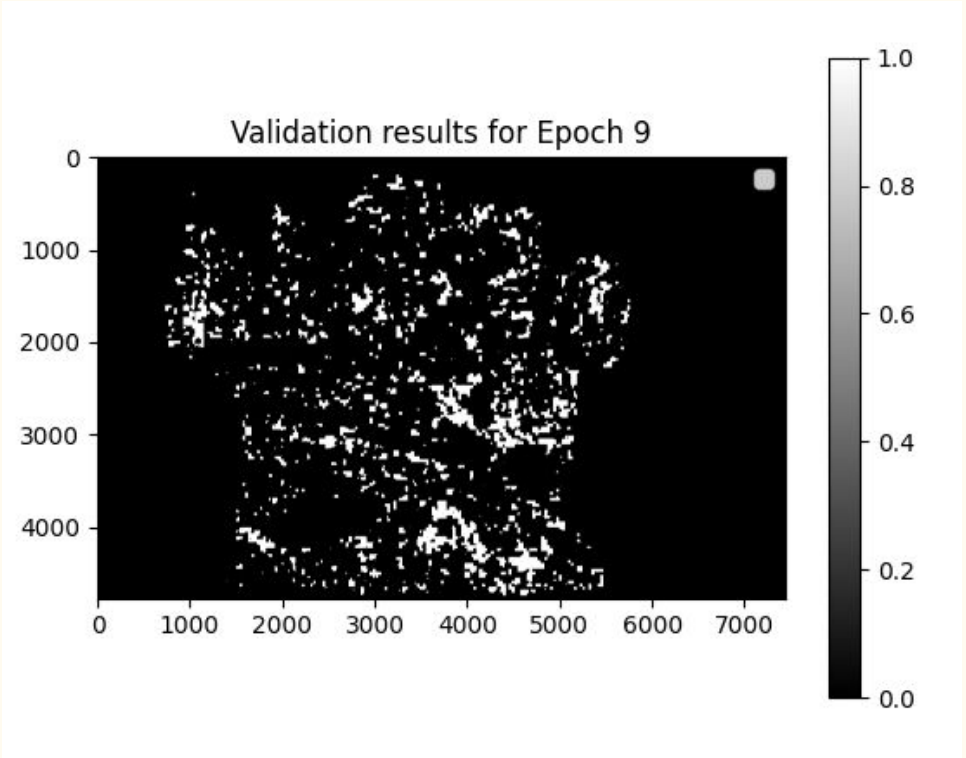
Evidence for lack of calibration

- Models preform better, when training and predicting on different parts of the same images

- A trained CNN can classify, which image a block is from with a 94% accuracy

- This accuracy is decreased slightly (down to 88%) by using calibrations

Reconstructions



Validation results of models

List of f-scores

- Encoder only Vision Transformer: **0.27**
- Encoder-decoder Vision Transformer: **0.21**
- 2d CNN encoder: **0.24**
- 3d→1d CNN encoder: **0.28**
- 2d→2d UNet: **0.26**
- 3d→2d UNet: **0.28**
- 3d→2d UNet like model: **0.34**
- CatBoost: **0.16**
- LDA: **0.15**

Looking back

Preprocessing is important

Pretrained models require 3 input channels

Hardware limitations

- Kaggle allows 30h/weekly with P100 GPU (13GB RAM, 16GB VRAM)
- Not enough for these very large images

Moving forward

- Better augmentations
- Look at calibrations and the preprocessing step
- Maybe look at more advanced denoising algorithms in preprocessing and postprocessing
- More advanced (pretrained) models
  - TransUNet
  - SegFormer
  - Segment Anything Model (SAM)
  - Ensembles
  - etc.
- Use Adversarial to punish model if it can predict which image a patch originated from

Thank you all for your attention!

Appendix

1. About the Problem
2. Models used
3. More reconstruction
4. Hyperparameters
5. Models we didn’t get to try
6. Miscellaneous

A.1 - About the Problem

The papyrus scrolls of Herculaneum

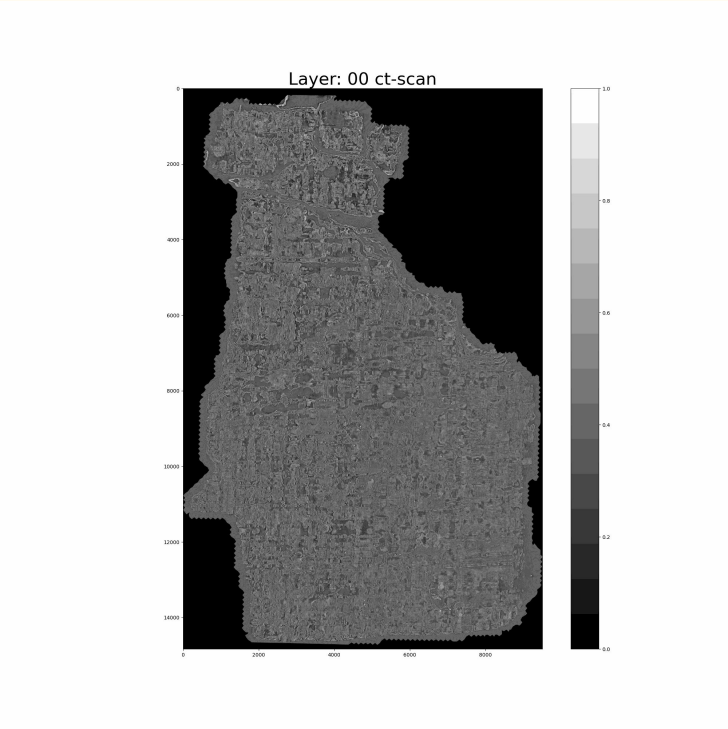
- Library in Herculaneum buried in 79 AD (contains works of Epicurean philosophy)
- Scrolls/fragments carbonized by the heat of the volcanic debris
- Brittle and sensitive scrolls read through tomography
- Carbon ink on carbonized papyrus a little hard to read
- Classification data from manually opened and studied scrolls

More information can be found at:

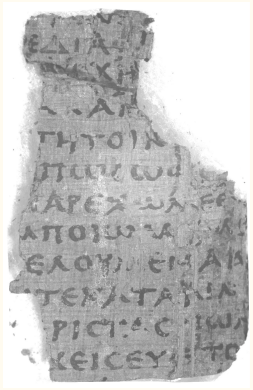
- *https://scrollprize.org/ink_detection,*
- *Reading the Herculaneum Papyri: Yesterday, Today, and Tomorrow - YouTube*

A.1 - Image 2

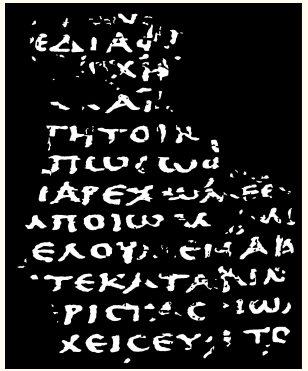
Visualization of z-layers



Real IR values



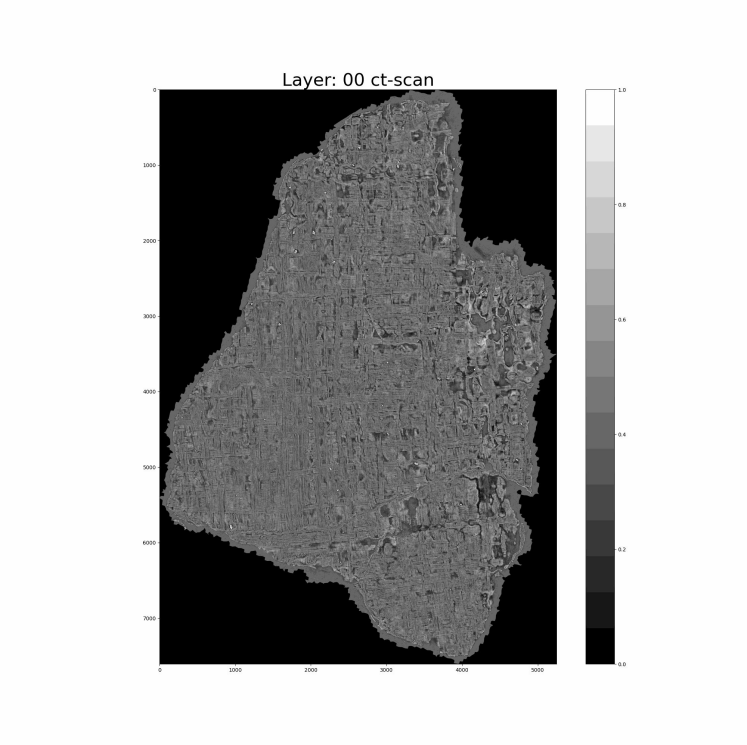
Ink Labels



Mask

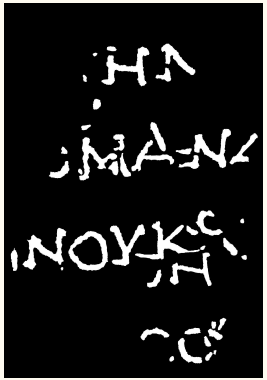
A.1 - Image 3

Visualization of z-layers



Real IR values

Ink Labels

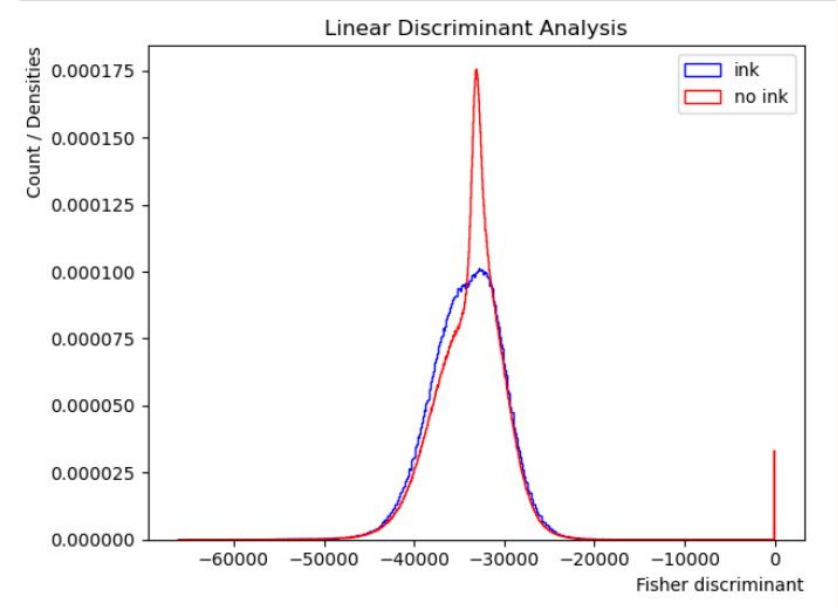


Mask

A.2 - Linear Discriminant Analysis

Not very much information in a single column of pixels.

Peaks are from the boundary

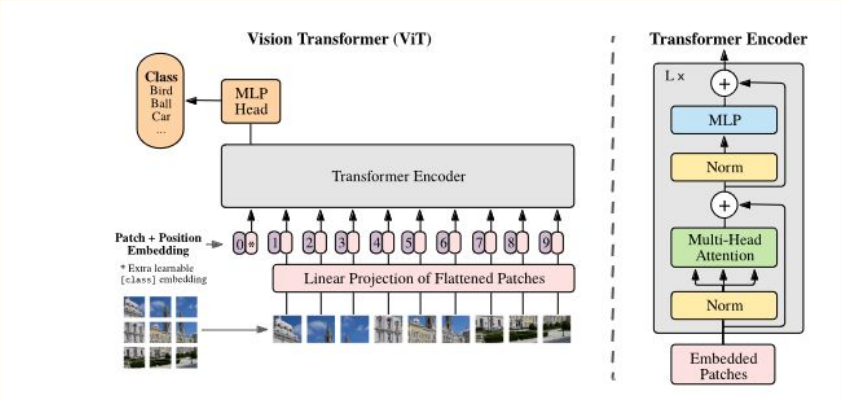


A.2 - Models

Vision Transformers

Vision Transformer (ViT)

- “Create sequence from Image”
- Encoder only transformer
- MLP Classification from a class token



ViT Original paper

A.2 - Models

Sequence to Sequence (Vision) Transformers

Convert both input and output image to a sequence.

Use a traditional encoder-decoder Transformer to “translate” between input and output, in a similar fashion to how an encoder-only Vision Transformer works.

Didn’t perform very well on our dataset, possibly due to the fact that transformers usually require very large amounts of data.

Can be generalized to TransUNet / Segment Anything Model (SAM)

A.2 - Models

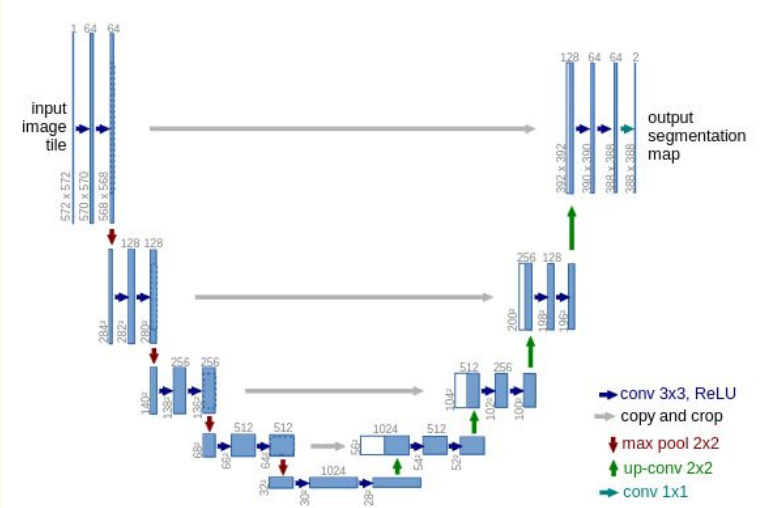
UNET

Encoder-decoder style CNN

Allows for arbitrary CNN Backbone encoder

Can be generalized to TransUNet

- Combining Vision Transformer with UNet

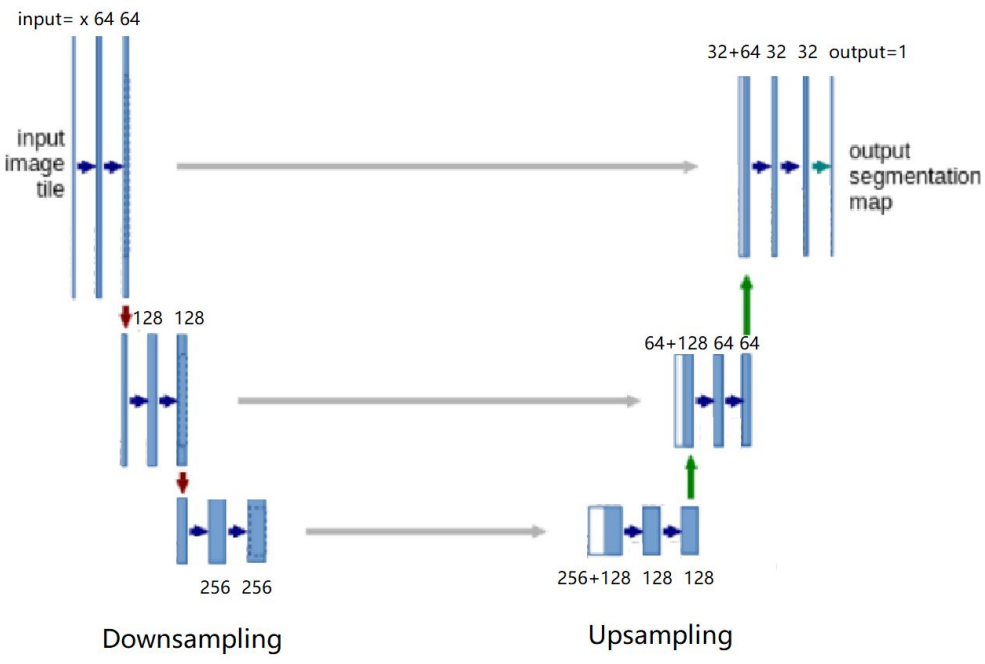


UNet Original Paper

TransUNet Original Paper

A.2 - Models

Sample 2D UNet Model We Use



x : the number of training layers

loss function : BCEWithLogitsLoss

optimizer : Adam (learning rate = 0.0005)

epochs : 30

A.2 - Models

(simple) 3D CNN Models

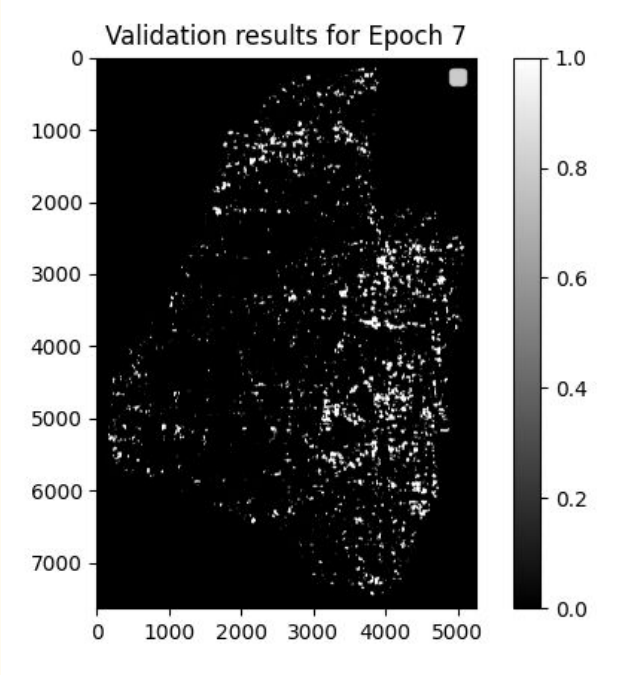
Two different CNN models have been tried.

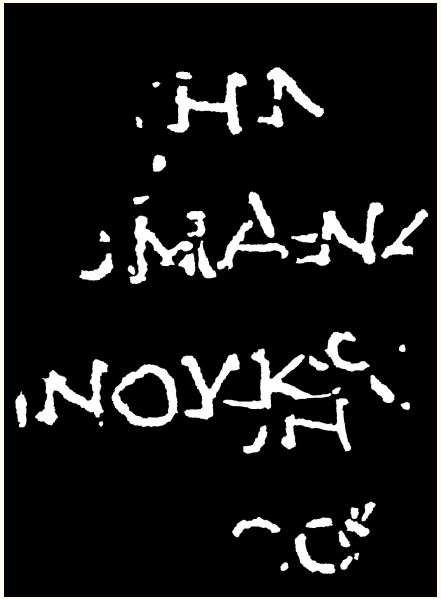
- First model: 3D encoder only (3 layers) followed by a linear layer. This model preformed with an f-score of 0.29
- Second model: UNet like mode, with a 3D encoder (3 layers + 3 pools) followed by a 2D decoder (3 layers + 1 upsample). There were however no skip-connections and no linear bottleneck. The best f-score was 0.34

Both models were only trained on layers 22-35

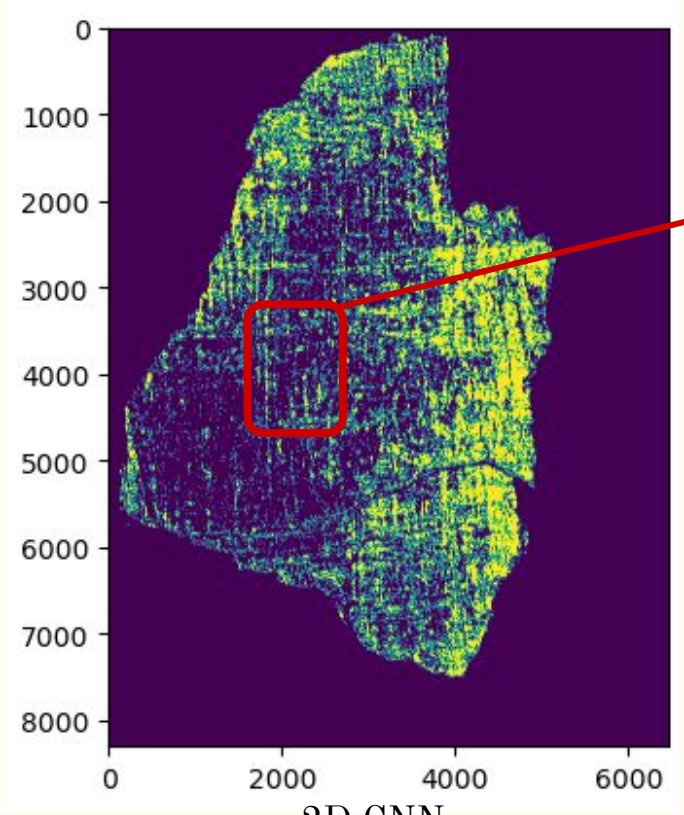
A.3 - Additional Reconstructions

ViT





A.3 - Additional Reconstructions



2D CNN

- ‘Invisible letter’ where there is no ink?
- ‘Information’ from random noise :)

A.4 - Hyperparameters

Due to hardware constraint and long training time, we didn’t do much dedicated hyper parameter optimization.

A.4 - Hyperparameters for best performing models

ViT

- Embedding dim: 512
- Attention heads: 8
- Patch size: (4x16x16)
- Feedforward dim: 2048
- Encoder layers: 6
- z layers: (0 - 39)
- Hull/Nucleus: (64x64) / (16x16)
- lr: 3e-5
- optimizer / lr scheduler: Lion / Linear warmup - cosine annealing

A.5 - Models we didn’t get to try

We went with the approach of first trying simple models and get them to work and gradually make the models more complex. However due to the preprocessing difficulties, we never really got to try cutting edge models for “semantic segmentation”.

Following are some models, which weren’t tried due to time and hardware limitations:

TransUNet (combination of Transformer with UNet) Original Paper

SegFormer (CNN+Transformer hybrid) Original Paper

Segment Anything Model (Modified Encoder-Decoder Transformer) Original Paper

A.5 - Models we didn’t try (continued)

It is also worth noting, that due to the very special nature of this segmentation task, almost none of the standard vision models will work out of the box, since they expect 3 input channels.

For the few pretrained models we’ve tried (ResNet34/50, ViT B), their feature extraction abilities also don’t seem to transfer well, to this very noisy data. And for the models to get any reasonable scores, they basically need to be retrained entirely.

A.6 - Miscellaneous

Lion (Optimizer)

- Recent State-of-the-art optimizer
- Used for training some of the models
- Pytorch Implementation: GitHub
- Lion: Original Paper

Also available on the nightly release of tensorflow