Cartography M.Sc.

Area Feature Reconstruction from Historical Topographic Maps Using Different Deep Learning Architectures

Sasanka Madawalagama

Supervised by:

Dr. Nikolas Prechtel

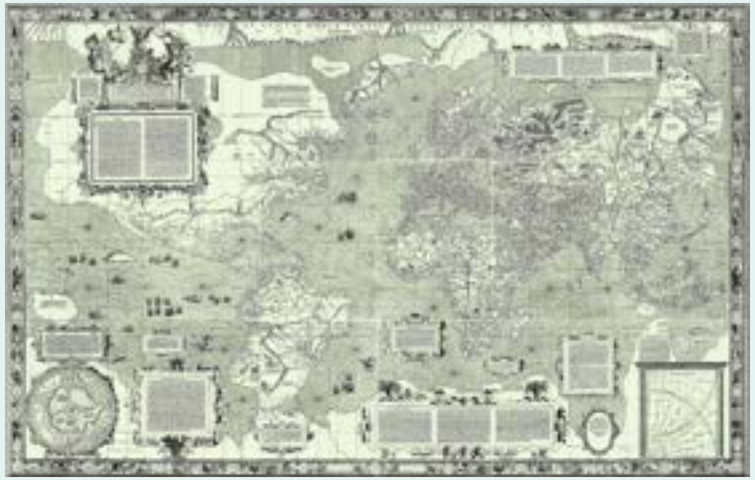
Prof. Markus Wacker

Outline

[1.1] Historical Maps

- Historical maps are an irreplaceable primary source of geographical and political information in the past.
- They are tools for reconstructing the past. Historical maps provide records of features, landscape, cities, and places that may not exist any more or that exist in dramatically transformed form.

Town plan of Imola, Italy by Leonardo da Vinci, 1502
(Ref: www.leonardo-da-vinci.net)



Mercator's World Map, 1569
(Ref:https://en.wikipedia.org/)

Mileage sheet from Saxony, sheet 1 – 180, 1807
(Ref: https://kartenforum.slub-dresden.de/)

[1.1] Digital Map Archives

USGS historical topographic map archive

Map Forum of the Saxon State Library

[1.1] Digital Map Processing
Unlocking the Data in Maps

Historical Maps

- Map Scans
- Metadata
  - Year of production
  - Title
  - Author

Digital Map Processing

1. Scanning
2. Geo-referencing
3. Extracting features
4. Cleaning / Fixing errors
5. Storing in geo-database

Historical GIS

- Multitemporal and multi-contextual spatial analyses
  - land-cover change
  - urbanization
  - glacial extents
  - political boundaries

[1.2] Challenges in Digital Map Processing

Established feature extraction methods of Digital Map Processing are either inefficient or does not scale well processing large numbers and varieties of historical maps (Chiang et al., 2020)

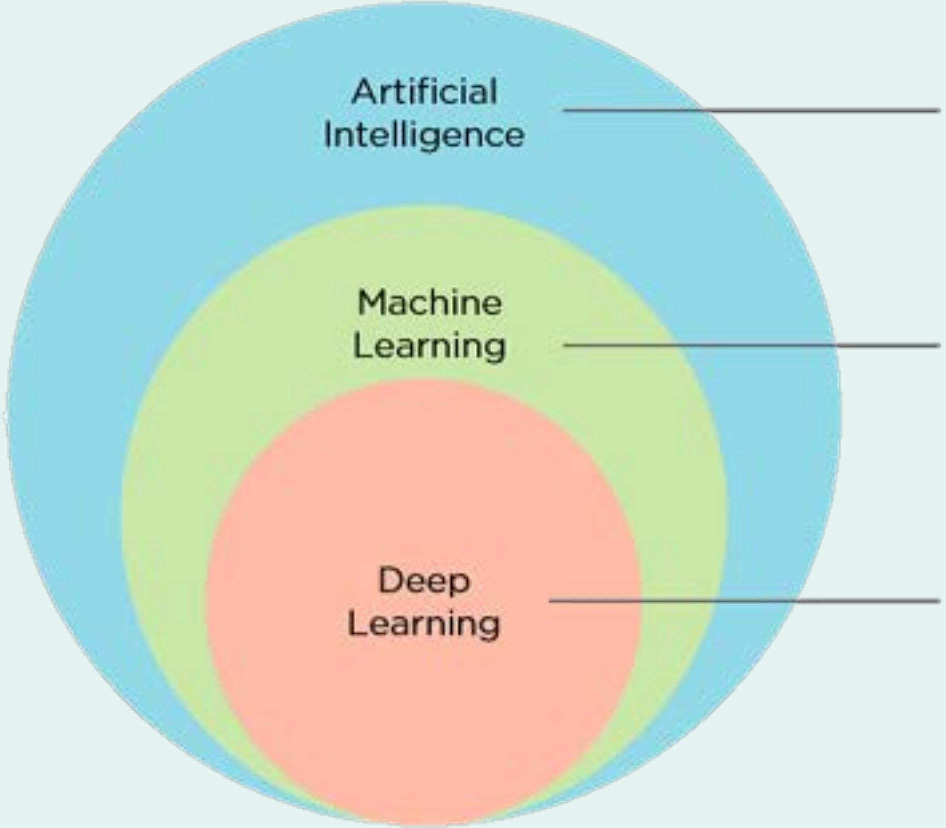
Different graphical qualities of Map Scans

Overlapping symbols

Effects of ageing

Variation of cartographic symbols

[1.3] Deep Learning



the ***concept*** of creating smart intelligent machines

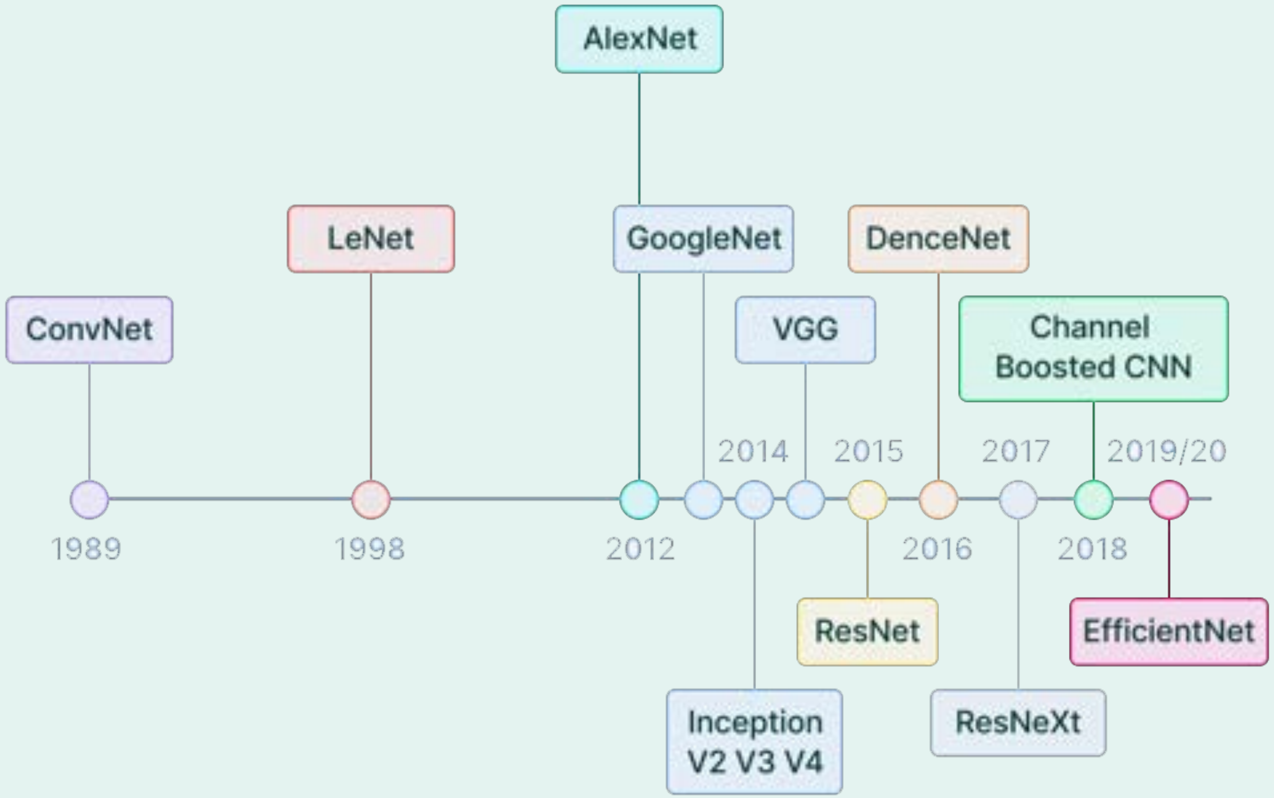
Application of AI that allows system to ***automatically learn*** and ***improve from experience***

Application of ML that uses ***complex algorithms*** and ***neural nets*** to ***train*** a model

Compared to machine learning methods, deep learning has the unique ability to learning complex algorithms that can never be explicitly programmed

[1.3] Deep Learning for Digital Map Processing

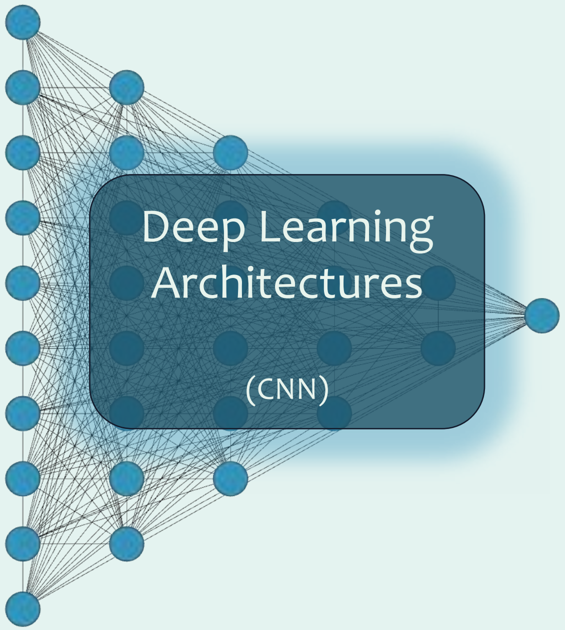
- **CNNs:** Convolutional Neural Networks
- CNNs: Achieves highest accuracy rate in complex image segmentation resulting paradigm shift in the field (Minaee et al., 2020)
- **CNN Model Architectures:** Advancements are made with the increasing computation power by solving the current limitation assessed by the core concepts of Deep Learning, **not by the application**. Trial and Error to find out best performing architecture for certain application.



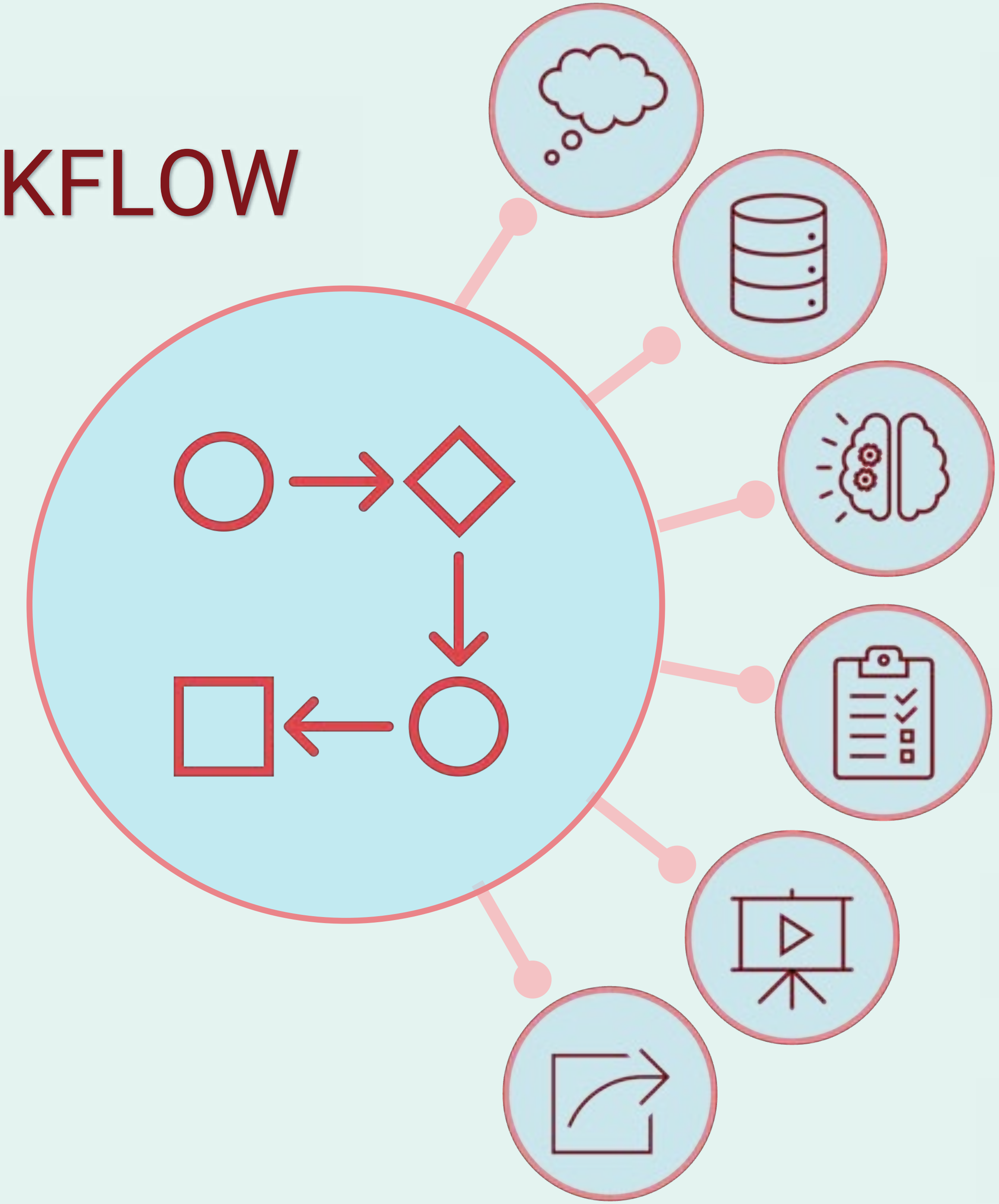
Milestones of CNN Model Architectures
(image: v7labs.com)

[1.4] Research Objective

Evaluate different deep learning architectures for digital map processing focusing on areal feature reconstruction from historical topographic maps.



[2] WORKFLOW



[2.1] Starting Point and Approach

[2.2] Data Preparation

[2.3] Deep Learning Pipeline

[2.4] Evaluation

[2.5] Results

[2.6] Next Steps

[2.1] Starting Point and Approach

Selected Area Features

Forest

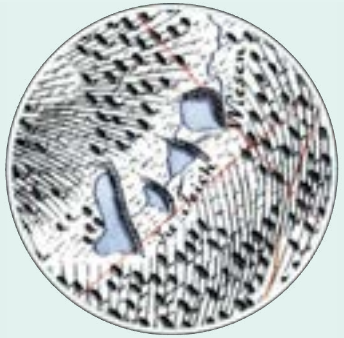
Buildings

Individual

Complex

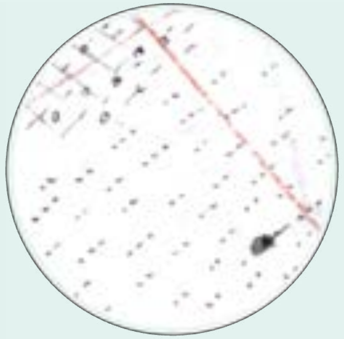
Water

Lakes



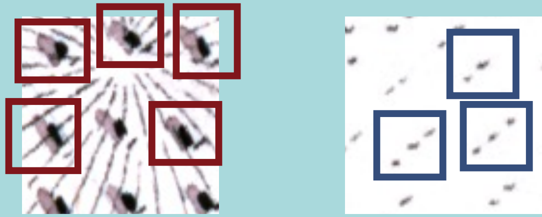
Rivers

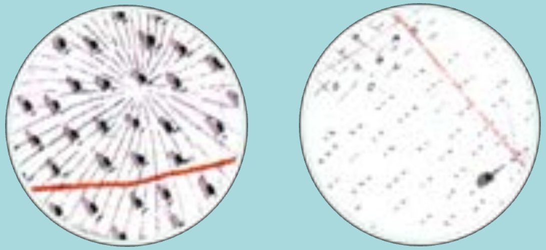
Meadow



[2.1] Starting Point and Approach

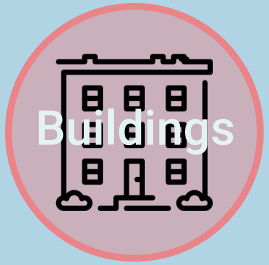
DL to Classify Selected Area Features

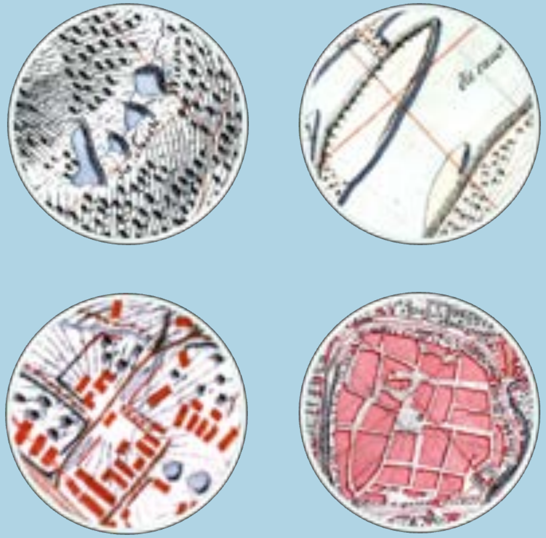




Object Detection

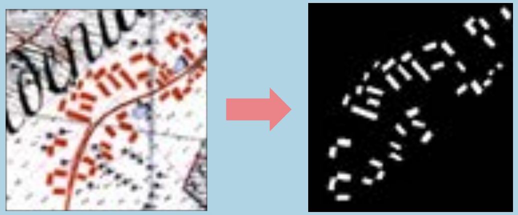
1. DL to detect the location of each symbol
2. Use clustering algorithm (e.g. DBScan) to reconstruct the area





Semantic Segmentation

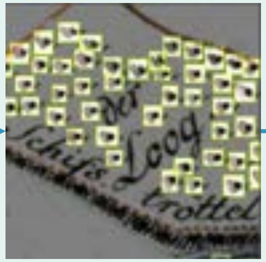
1. DL to classify each pixel of the map



[2.1] Starting Point and Approach
Classification Strategy

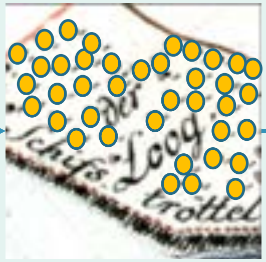
Input Data

→ Object Detection →

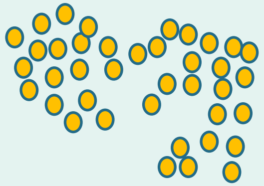


Detected Symbols
All individual symbols are detected using bounding boxes

→

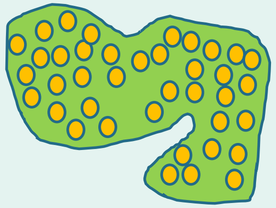


→



Symbol Locations
Individual symbol location is calculated

→ Clustering →



Vectorized Area
The area is extracted by clustering the individual points

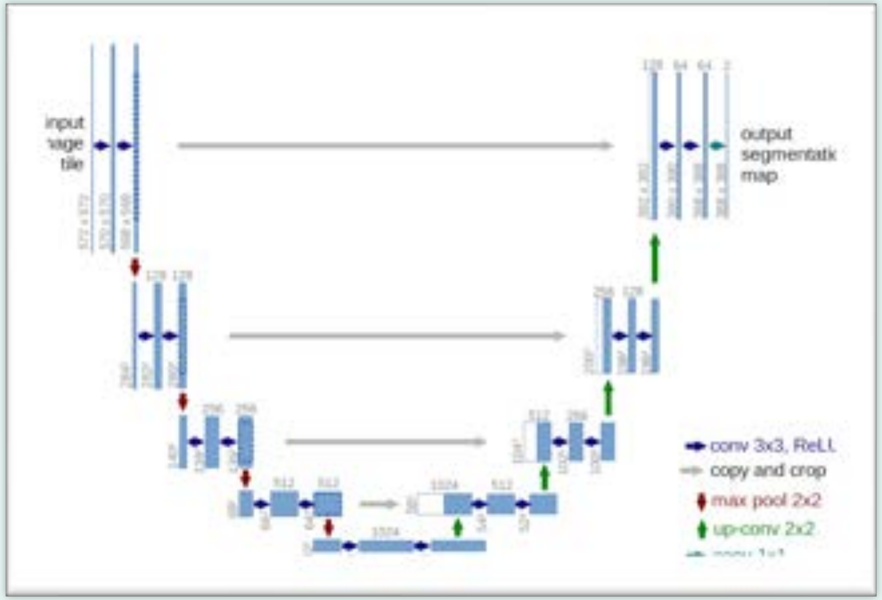
Input Data

→ Semantic Segmentation →

Output Mask
Each pixel of the mask provides value 1 or 0 corresponding to building and non-building class

[2.1] [RQ1] Starting Point and Approach
Deep Learning Architectures

Selected Semantic Segmentation Architectures:



UNet (with Batch Norm)

- Originally developed for biomedical image segmentation
- Most Influenced CNN for semantic segmentation
- Improved version (UNet + batch norm) will be used
- Use Case:

ICDAR 2021 Competition on Historical Map Segmentation (Chazalon et al., 2021):
Task 1: Detect Building Blocks
74.1 Panoptic Segmentation Quality

Cartographic Reconstruction of Building Footprints from Historical Maps: A study on the Swiss Siegfried Map (Heitzler & Hurni, 2020)
88% IoU

[2.1] [RQ1] Starting Point and Approach
Deep Learning Architectures

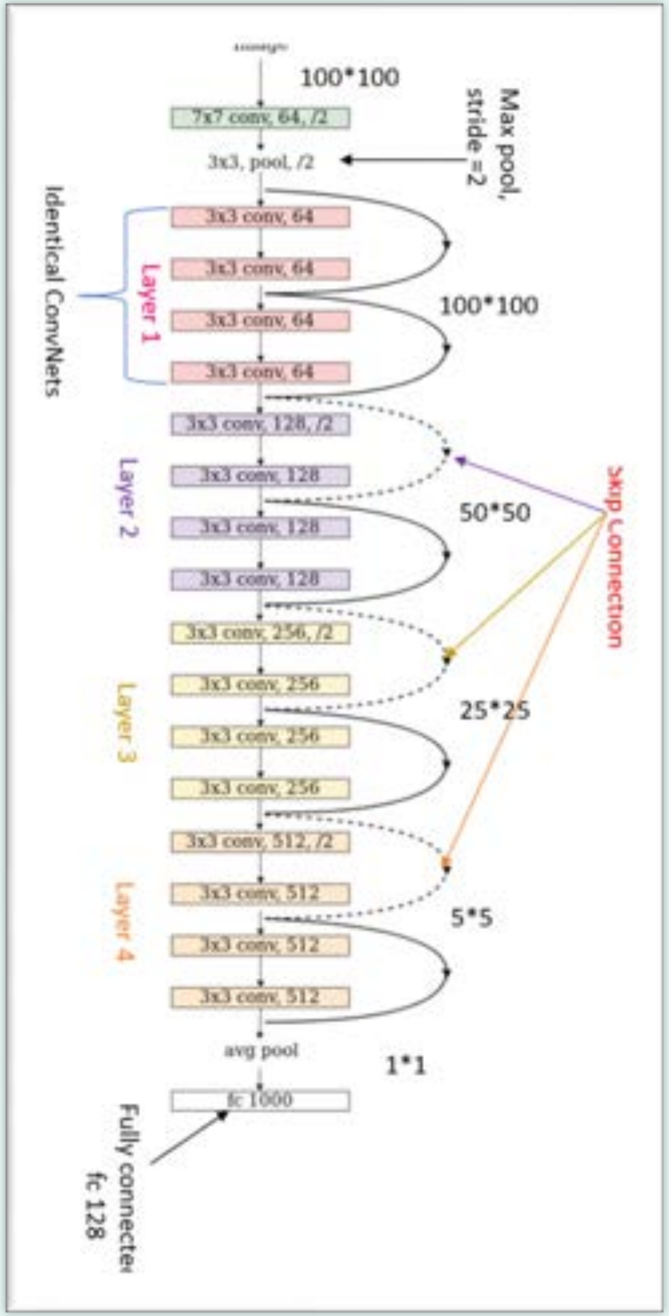
Selected Semantic Segmentation Architectures:

ResNet

- Originally developed by Microsoft Research in 2015
- Holds the first place for SpaceNet 1 (Satellite image dataset for building detection) benchmark 2022

  78.48 IoU
  ref: https://paperswithcode.com/sota/semantic-segmentation-on-spacenet-1
- Use Case:
  Geography-Aware Self-Supervised Learning (Ayush et al., 2022)

  Generic semantic segmentation of historical maps of Paris (Petitpierre et al., 2021).
  Buildings classification: 91% accuracy | Road networks classification: 75% accuracy



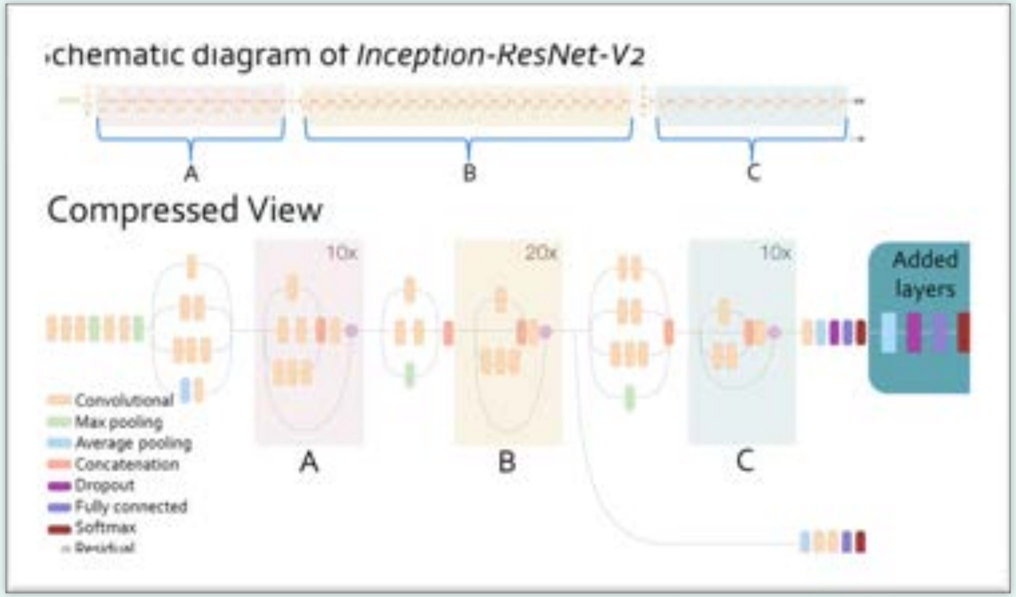
[2.1] [RQ1] Starting Point and Approach
Deep Learning Architectures

Selected Semantic Segmentation Architectures

InceptionResNet V2

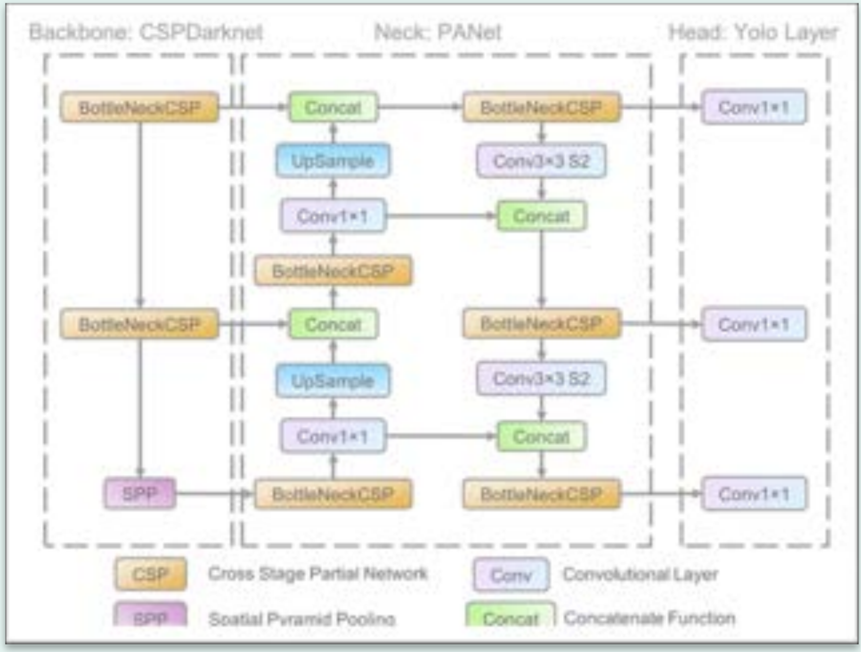
- Originally developed by Google Research in 2015
- Derivation of the original Inception
  - Winner of the 2015 ImageNet challenge with an error rate of 6.67%
- Use case:
  Comparison of Different U-Net Models for Building Extraction from HighResolution Aerial Imagery (Erdem & Avdan, 2020)

  F1 Score: 86.04, Best Performing Model



[2.1] [RQ1] Starting Point and Approach
Deep Learning Architectures

Selected Object Detection Architectures



YOLO (You Only Look Once)

- YOLO broke the traditional CNN implementation at its invention (Du, 2018) by combining two separate processes (detection+classification) into one process.
- designed to be simple yet effective in object detection, Used in real-time object detection applications
- Use case:
  Detect buildings from remote sensing imagery
  - Accuracy 88.5 (Ding & Zhang, 2021)
  - Accuracy between 88% and 98% in various scenarios (Kim & Hong, 2021)

Architecture	YOLO v5_m	YOLO v5_l	YOLO v5_x
Number of Parameters	21.2M	46.5M	86.7M

[2.2] Data Preparation
Pre-Processing

1. Color Homogenization

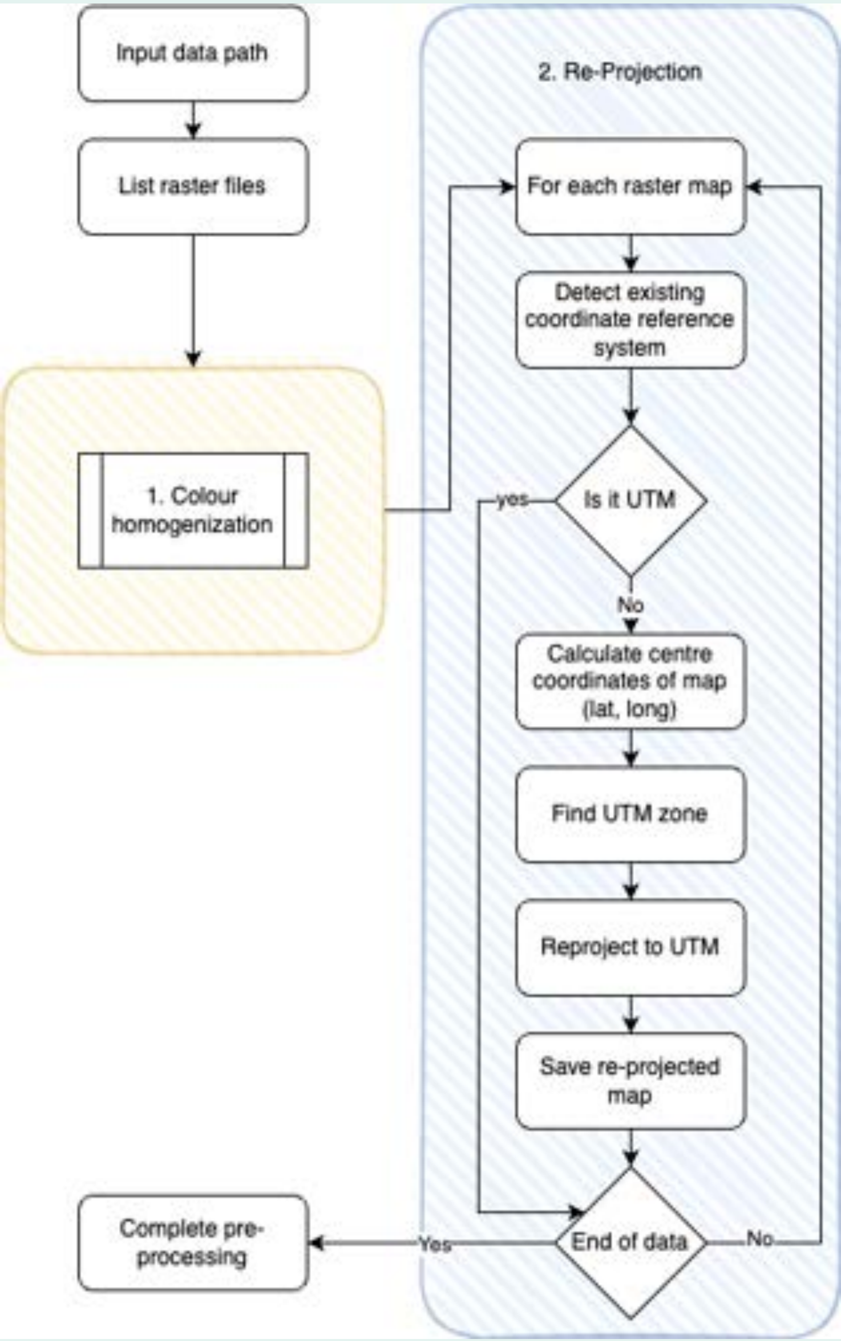
- purpose: Bring all the maps to same color levels
- Makes the task a bit easy for DL model
- Note: The workflow is already established by the working group

2. Reprojecting to Conformal Projection System

- purpose: Bring the maps to the way they are originally intended to be
- Important when using constrained vectorization

** automated with*

Workflow



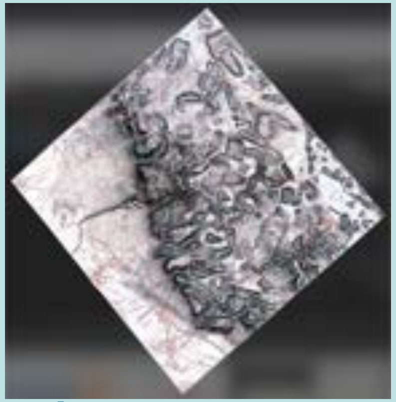
[2.2] Data Preparation

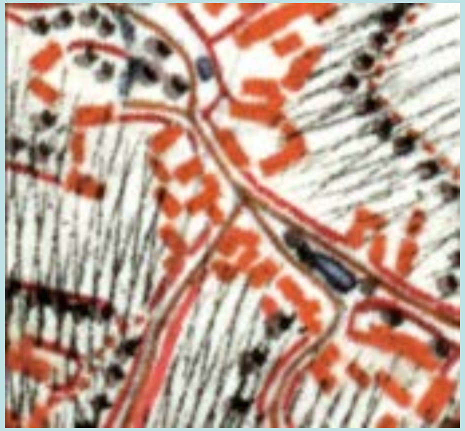
Pre-Processing

Original Maps




Preprocessed Maps



[2.2] [RQ2] Data Preparation
Creating Training Data

1. Tool to Efficiently Extract Training Patches

- <u>purpose:</u> Establish a method to speed up collection of training samples

- <u>Reason:</u> GIS software provide the similar functionality but its too complex for a simple task

- <u>Features:</u>
  - Semi-automatic extraction of training samples
  - Manual digitization capability
  - Automatically save the training samples
  - Can be extended to a web app so multiple users can contribute to create training samples

** made with*

[2.2] [RQ2] Data Preparation
Creating Training Data

2. Image Augmentation

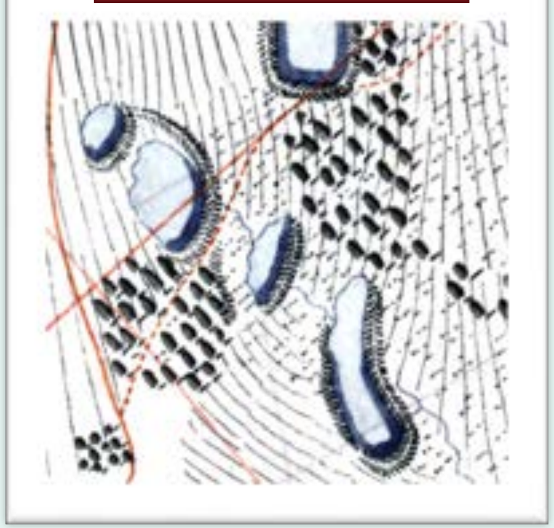
- purpose: Build a good number of training data from limited number of images

- method:

  Random
  - Crop
  - Rotation
  - Translation
  - Color Shift
  - Scale

** automated with* Python, Albumentations

Original Image



Augmented Images

[2.2] [RQ2] Data Preparation
Creating Training Data

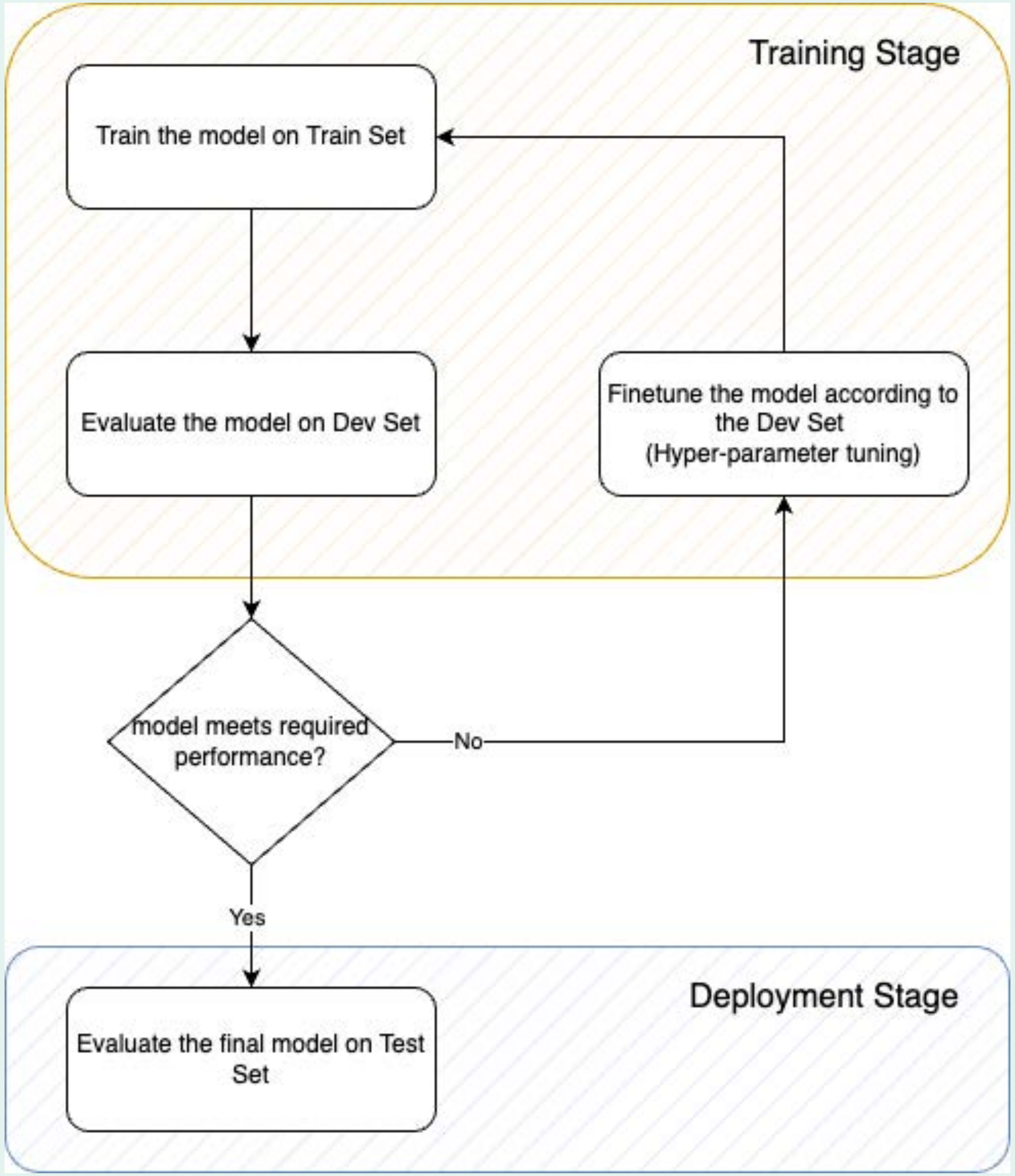
3. Splitting the Training Data

<u>Train Set</u>: Train and make the model learn the features in the data

<u>Dev Set</u>: Validation of model performance during the training

<u>Test Set</u>: Unbiased performance estimation of the final model

Train, Dev, Test Ratios : 70%, 20%, 10%



[2.3] [RQ3] Deep Learning Pipeline

Features

Compare different Deep Learning model architectures

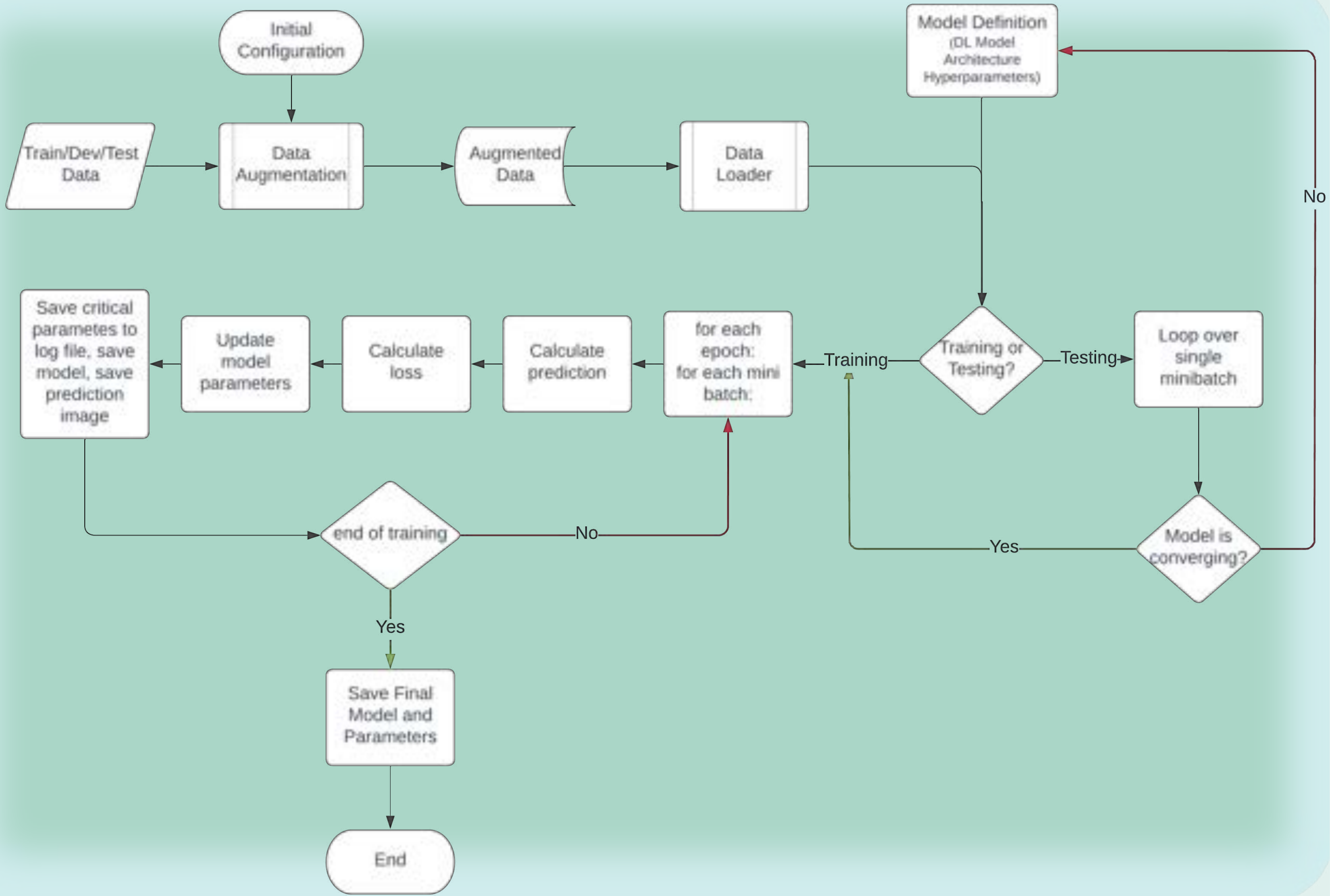
Ability to test model implementation prior to training

Automatically generates log files and images for evaluation

Integrated with a dashboard to visualize model performance.

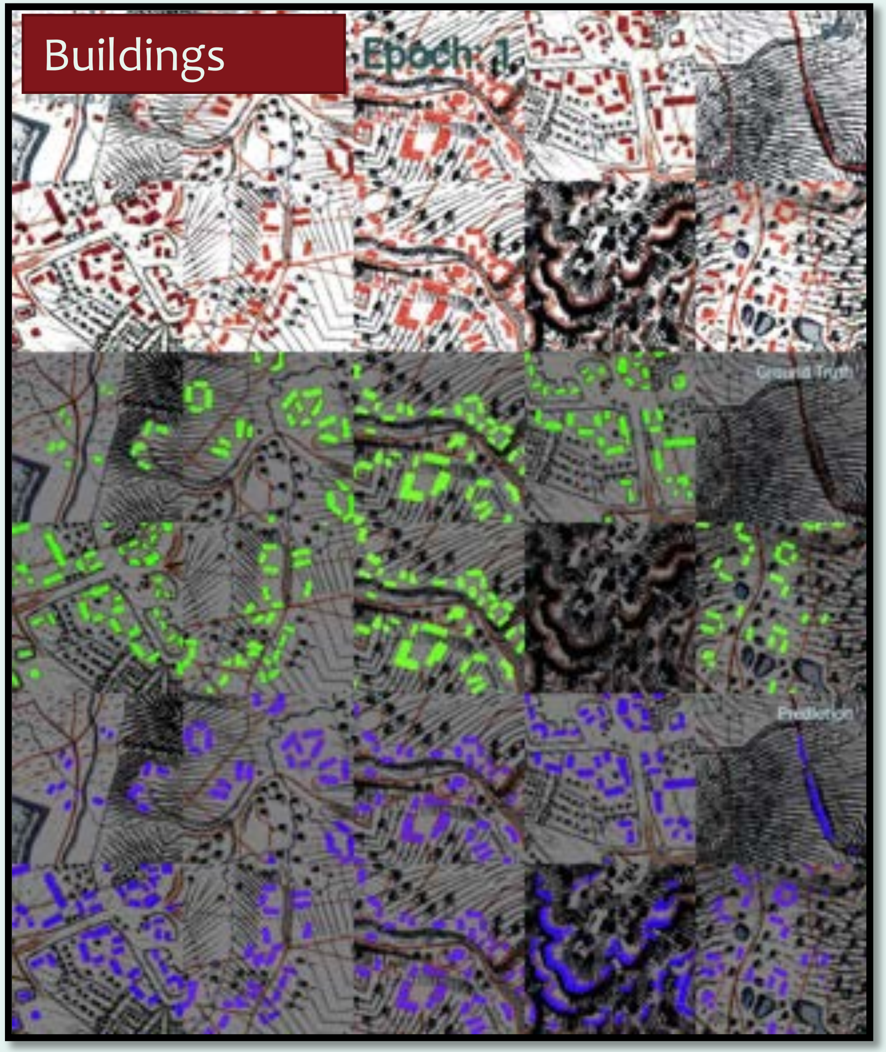
Google Colab integration

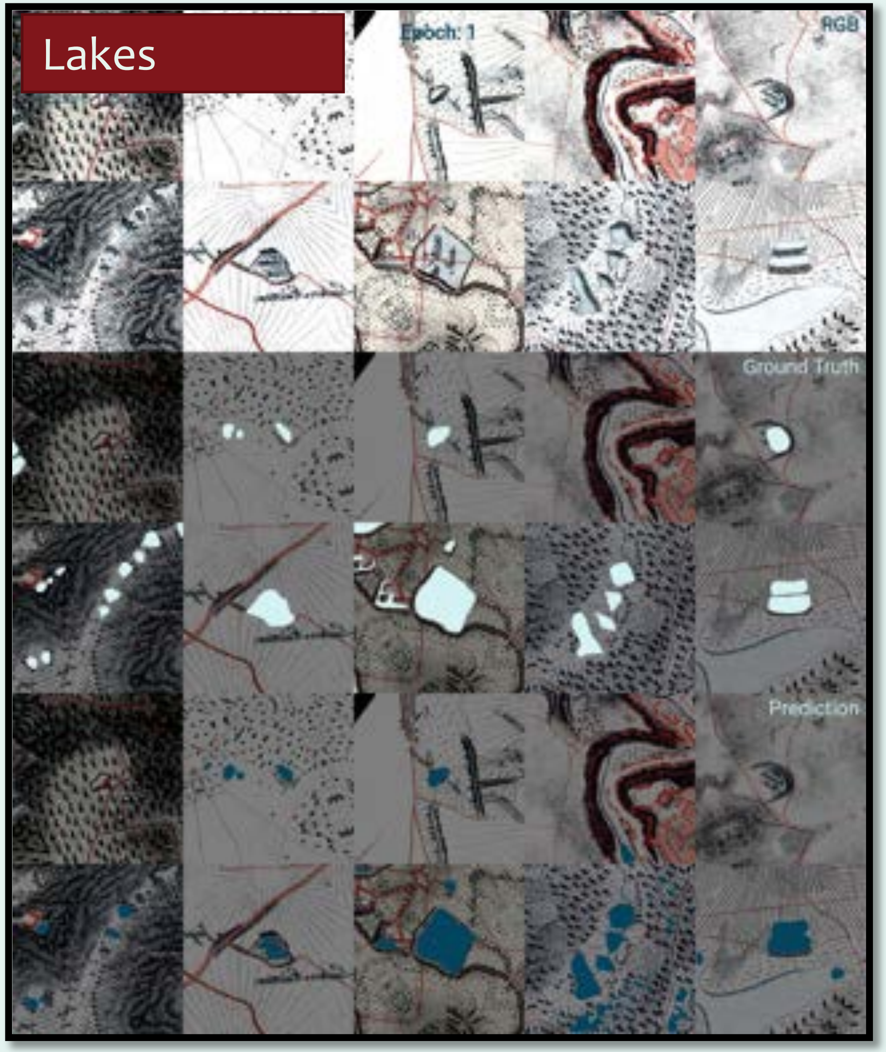
** made with* PyTorch



[2.3] [RQ3] Deep Learning Pipeline

Training Process



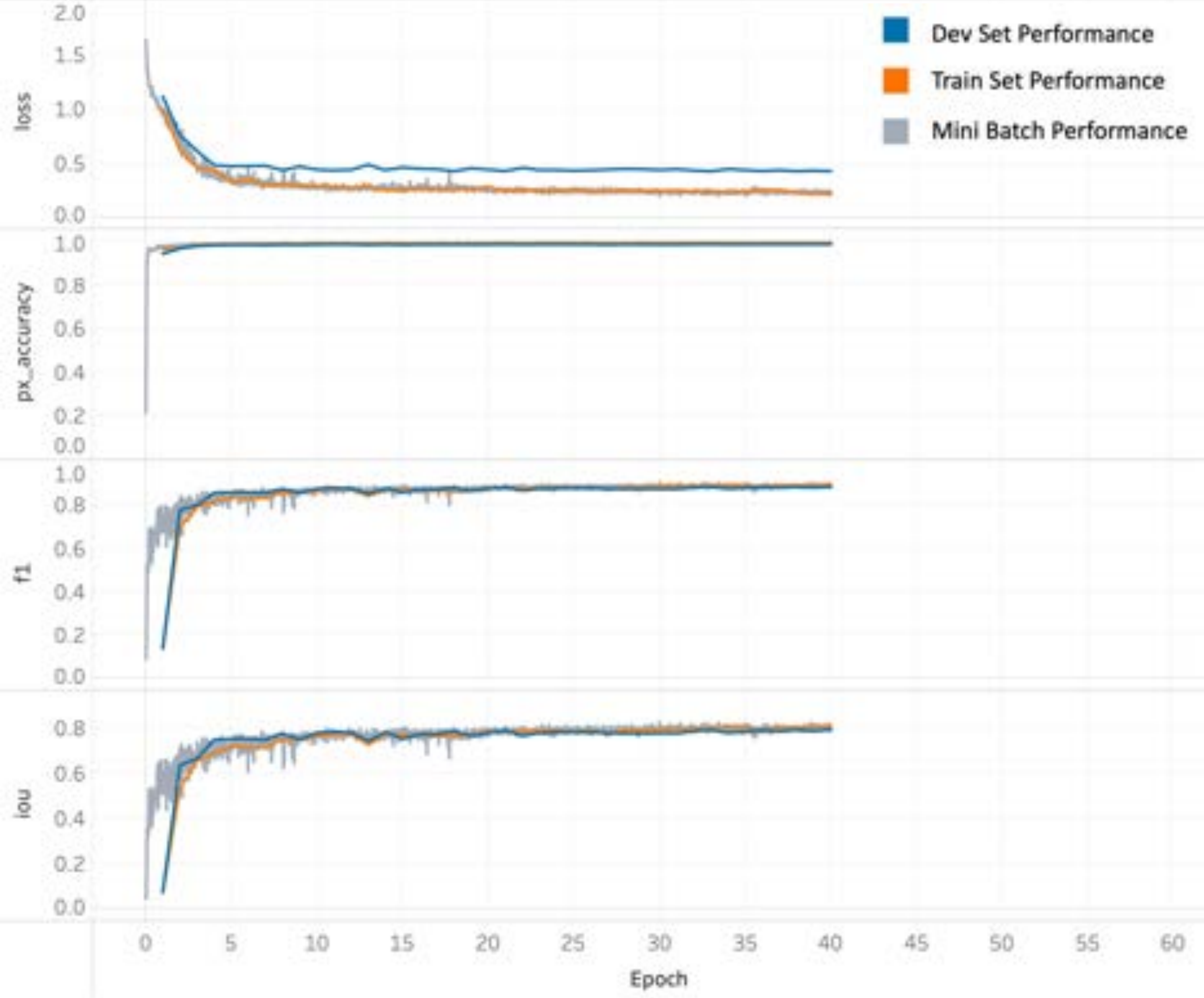


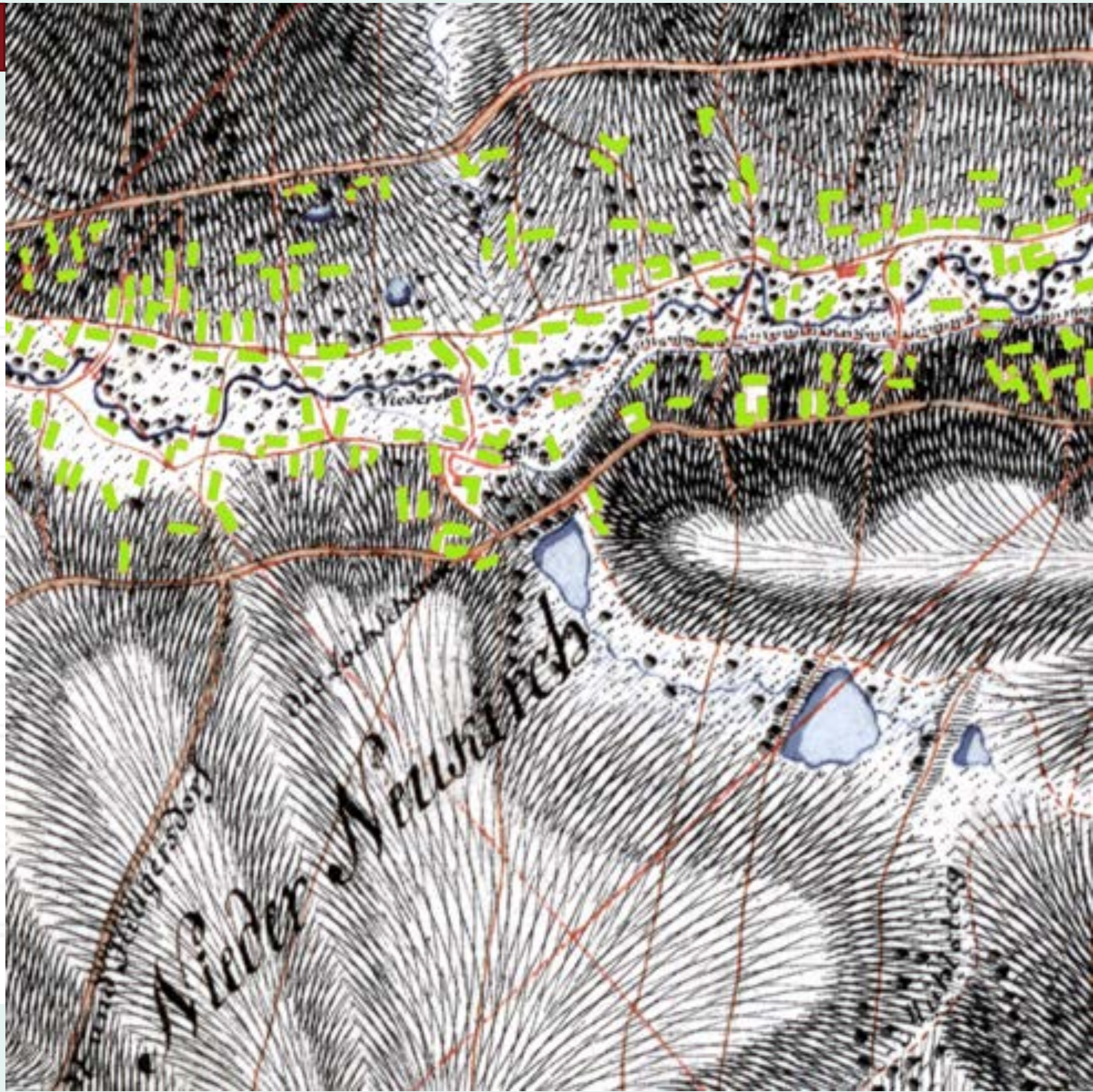
[2.4] [RQ4] Evaluation

- Evaluation matrices
  - **Pixel accuracy** → *percent of pixels that are classified correctly*
    $$=\frac{TP+TN}{TP+FP+TN+FN}$$
  - **IoU** Intersection-over-Union → *how successful is the prediction*
    $$=\frac{Area\ of\ Overlap}{Area\ of\ Union}$$
  - **F1 Score** → *combines the precision and recall of a classifier into a single metric*
    $$=\frac{2*precision*recall}{precision+recall}$$
  - **mAP** Mean Average Precision → Area under the precision-recall curve. (specifically for object detection)
- Processing time

[2.5] Results
Buildings

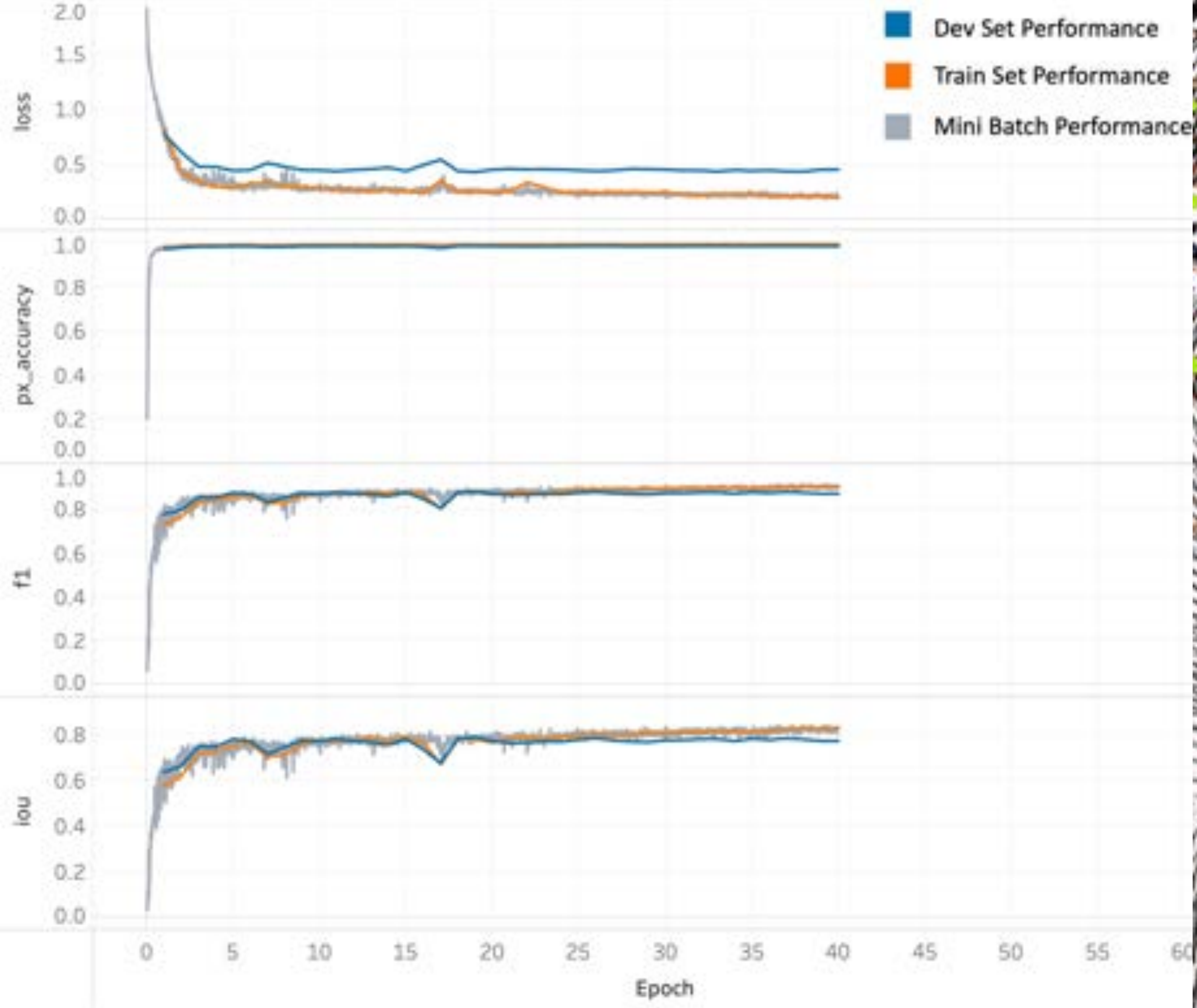
UNet

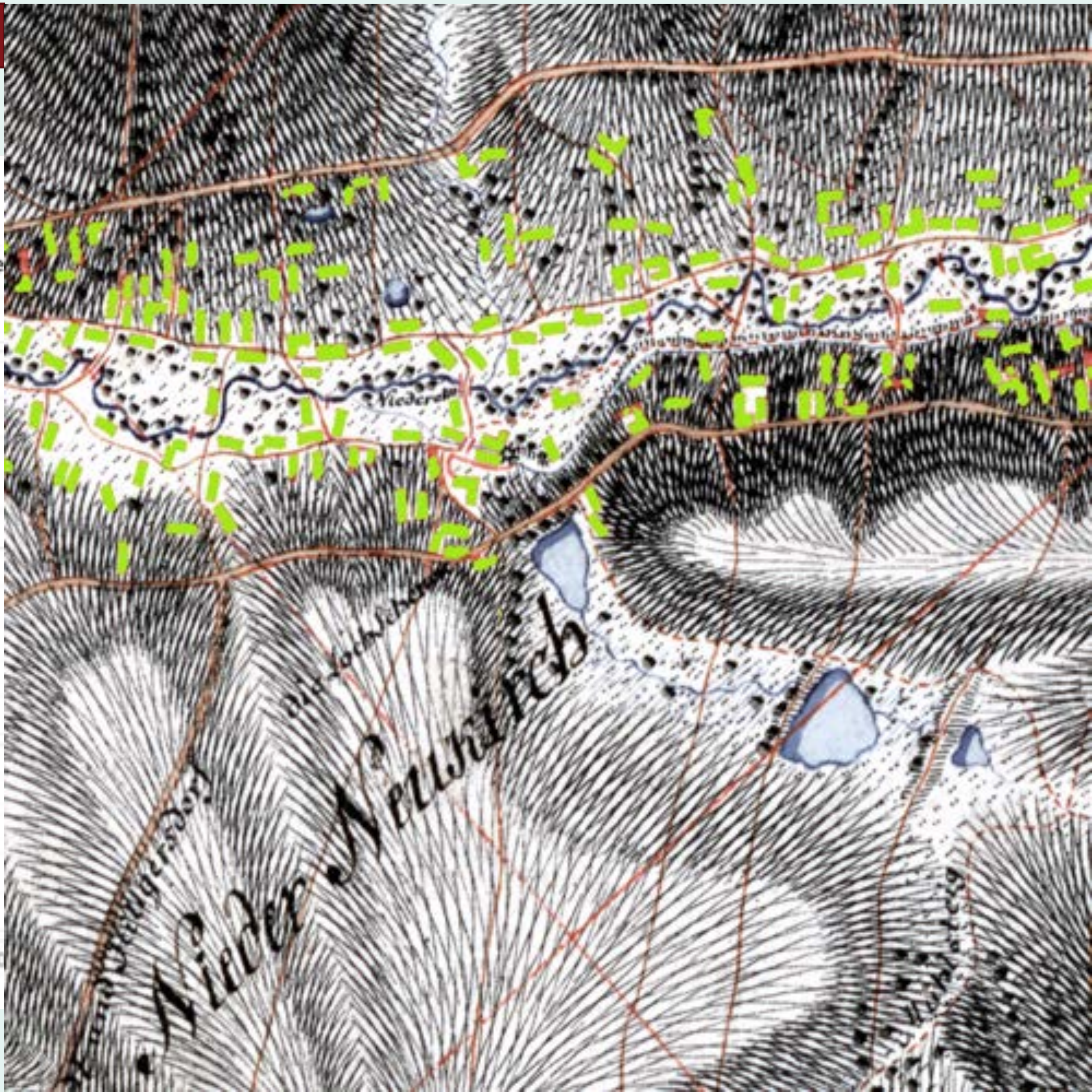




[2.5] Results
Buildings

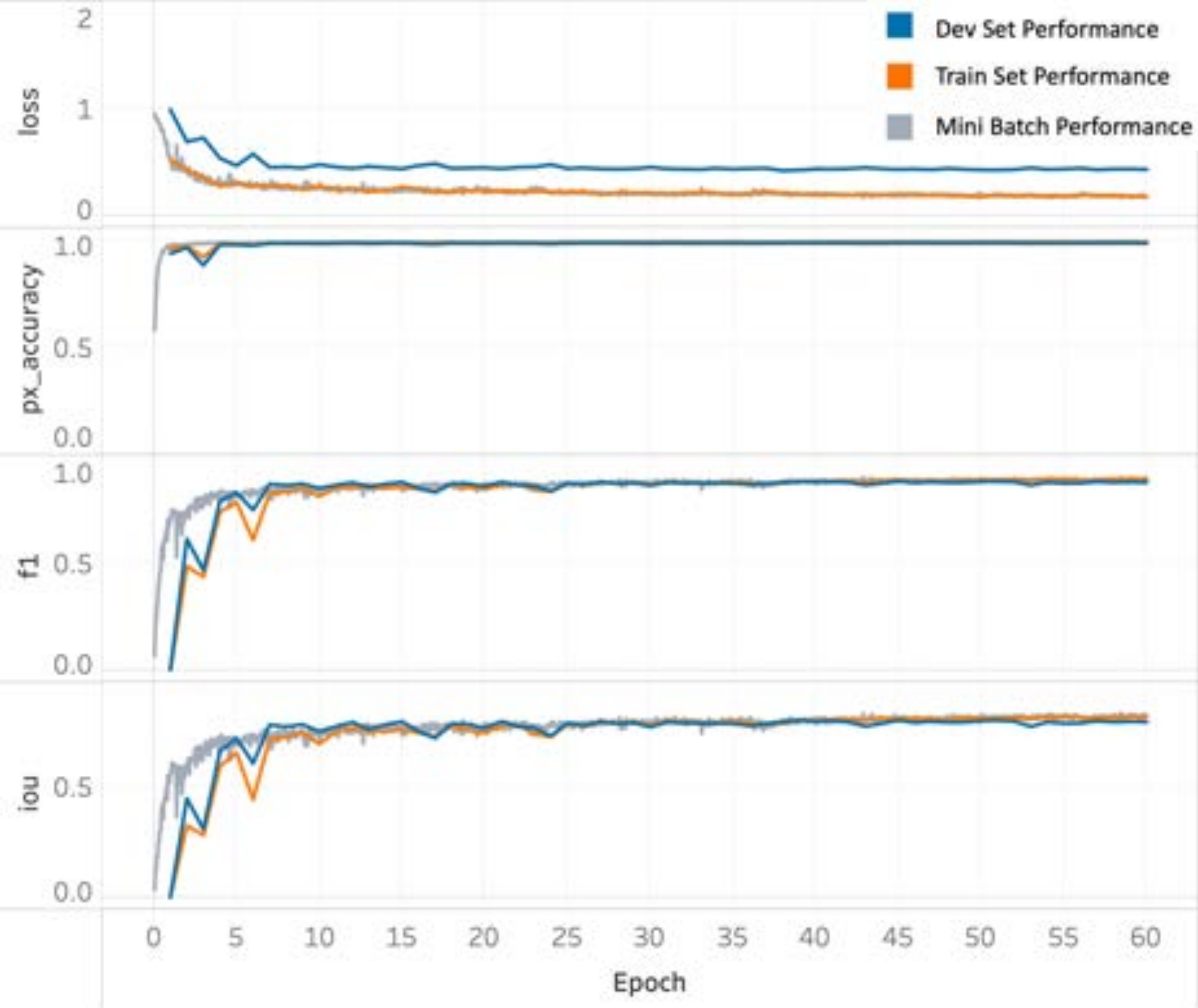
InceptionResNet

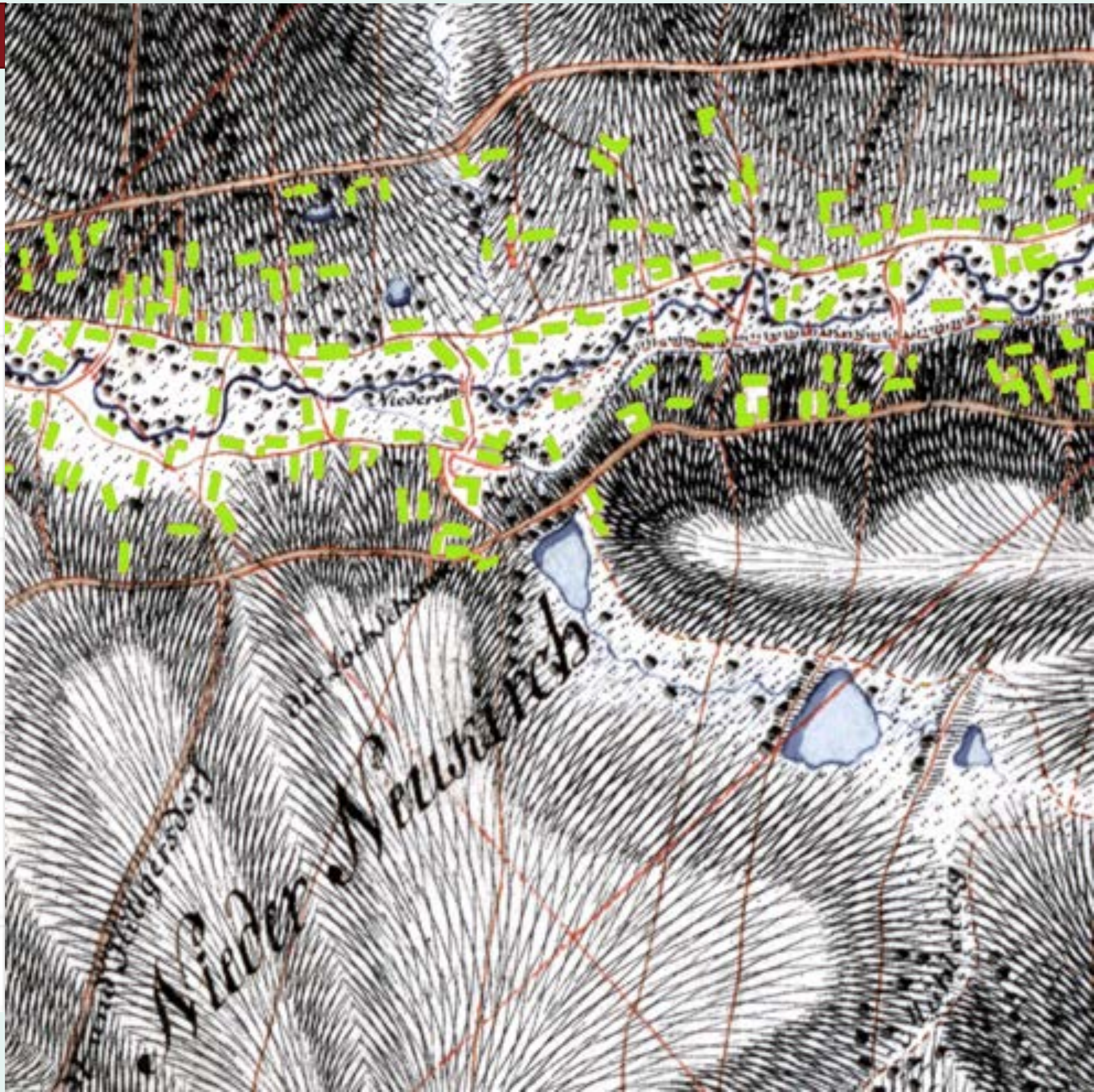




[2.5] Results Buildings

ResNet





[2.5] Results
Buildings: Misclassifications

Misclassifications occurred in Rock Symbols and Building Complexes

Possible Solutions:

1. Including more samples of rocks in the training data and making them true negatives

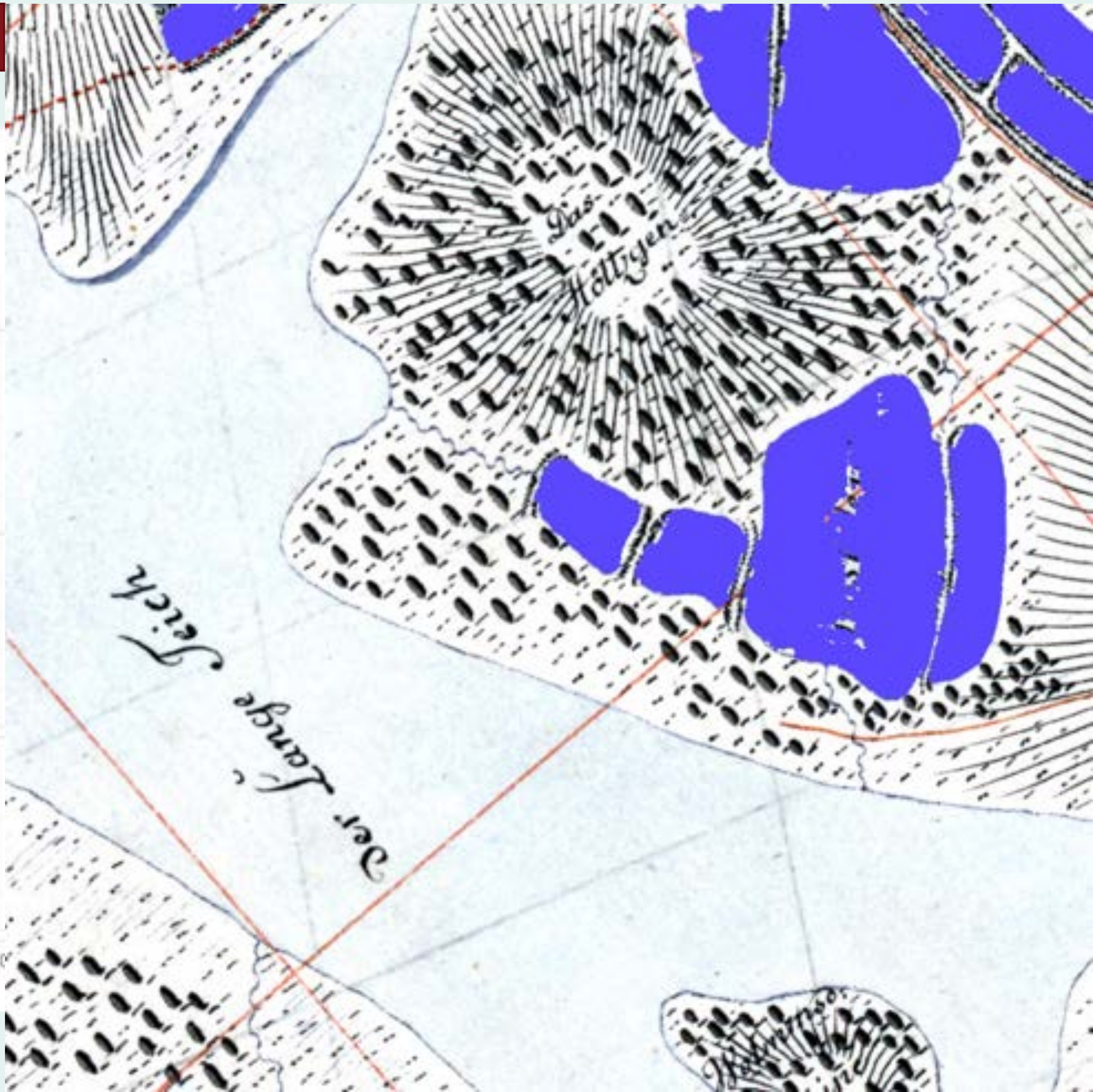
2. multi-class semantic segmentation approach

[2.5] Results
Buildings

Architecture	UNet	InceptionResNet	ResNet
Input Parameters			
Input image size (px)	256	256	256
Number of images in Train Set	1664	1664	1664
Number of images in Dev Set	608	608	608
Hyper Parameters			
Number of epochs	40	40	60
Mini batch size	32	32	64
Initial learning rate	0.001	0.001	0.003
Learning rate decay	Step	Step	Reduce on Plateau
Optimiser	Adam	Adam	Adam
Loss function	IoU+BCE loss	IoU+BCE loss	IoU+BCE loss
Accuracy Parameters			
Best epoch	25	25	**40**
Pixel accuracy	0.985	0.984	**0.985**
F1-Score	0.878	0.876	**0.884**
IoU	0.782	0.779	**0.792**
Performance Parameters			
Time to compute one epoch	00:02:21	00:01:40	**00:00:39**
Total time to train	01:34:15	01:06:47	00:38:31

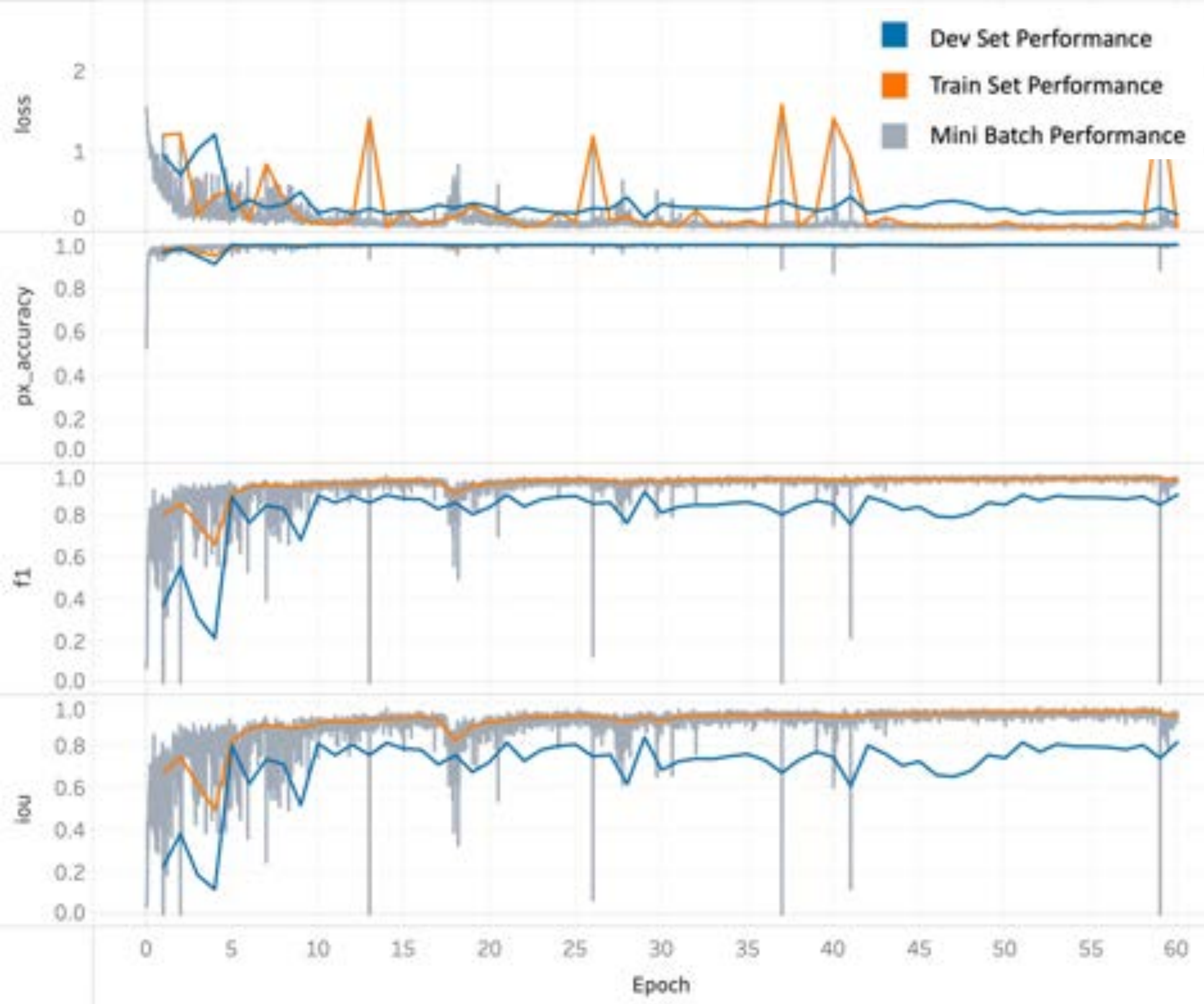
[2.5] Results
Lakes

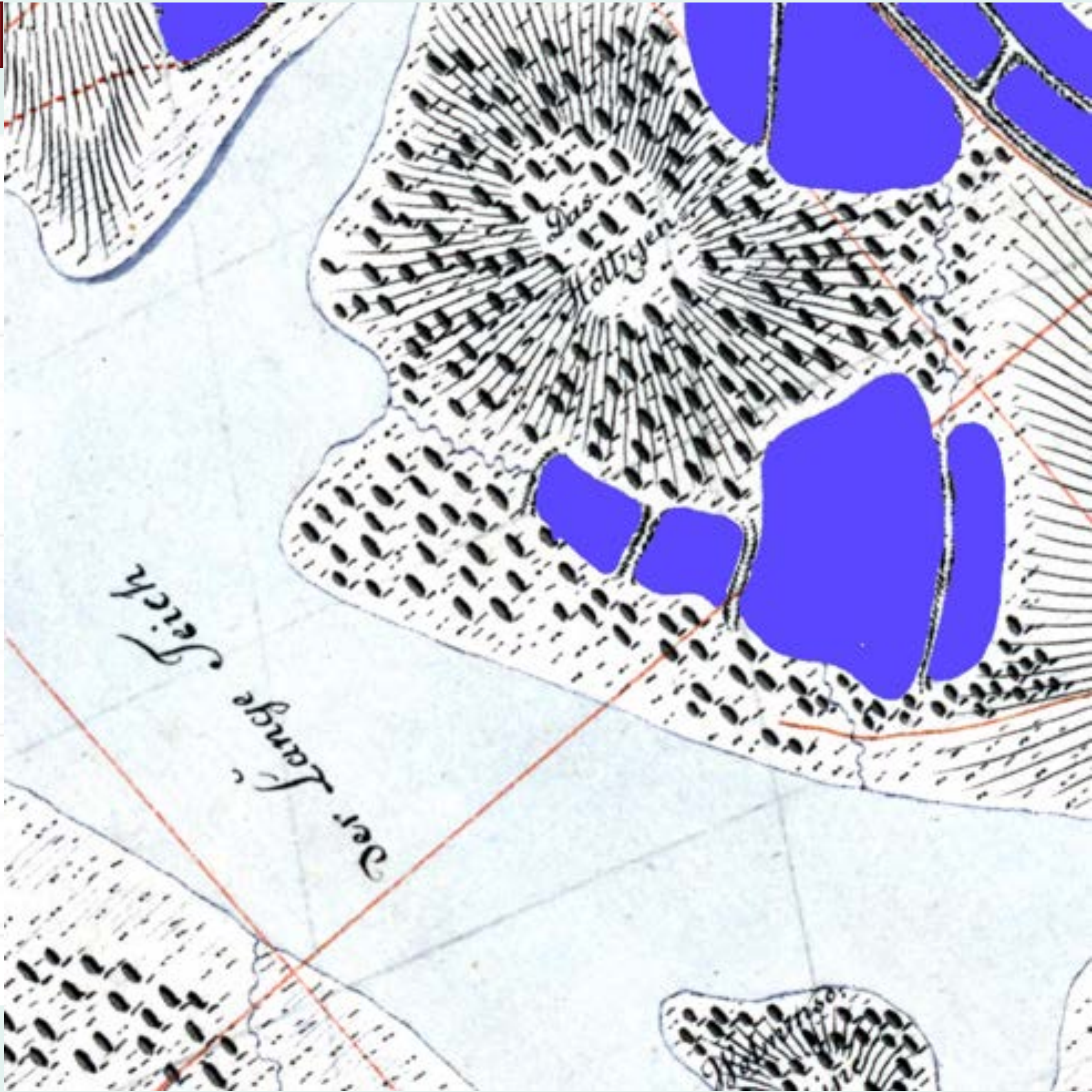
UNet



[2.5] Results
Lakes

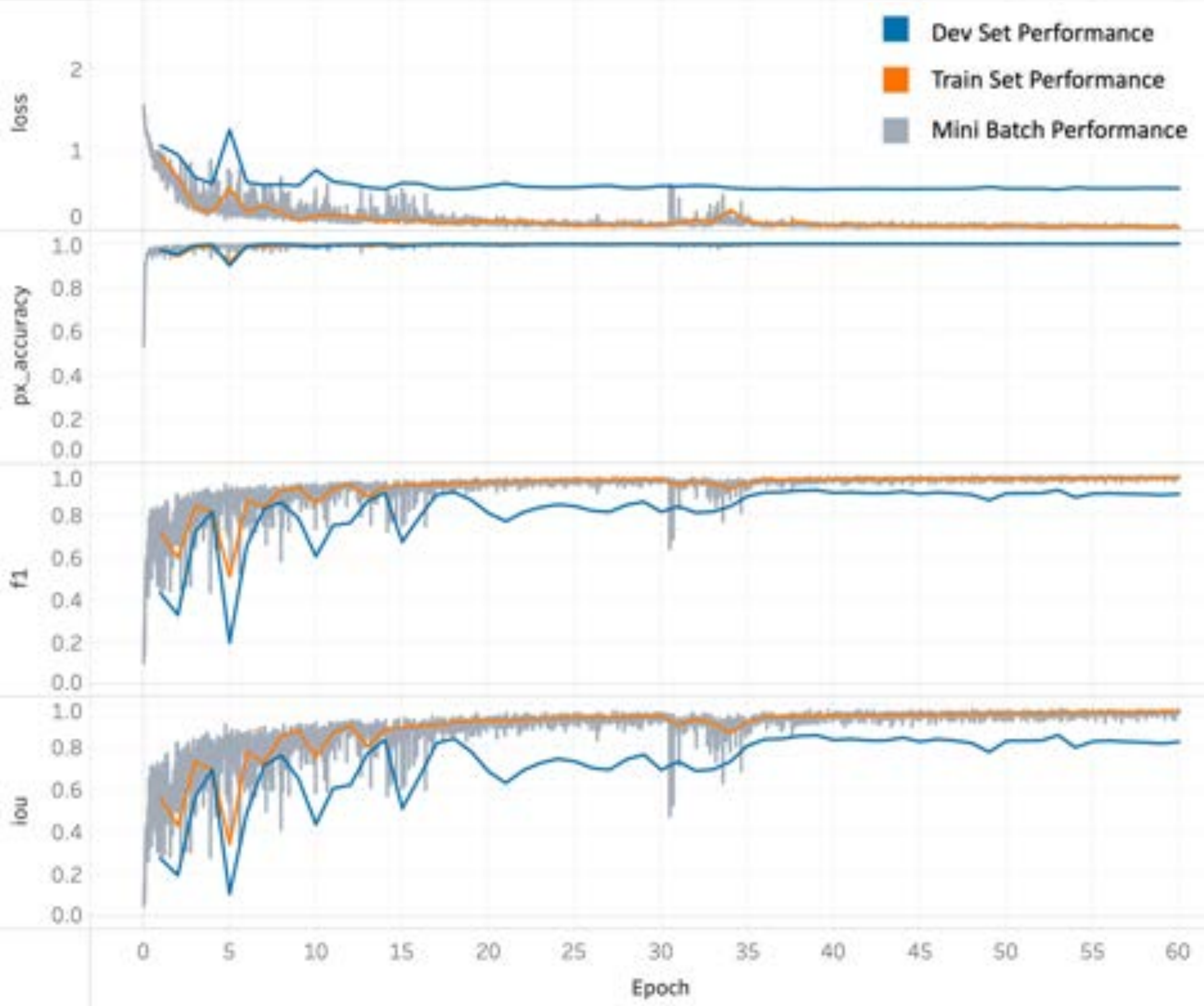
InceptionResNet

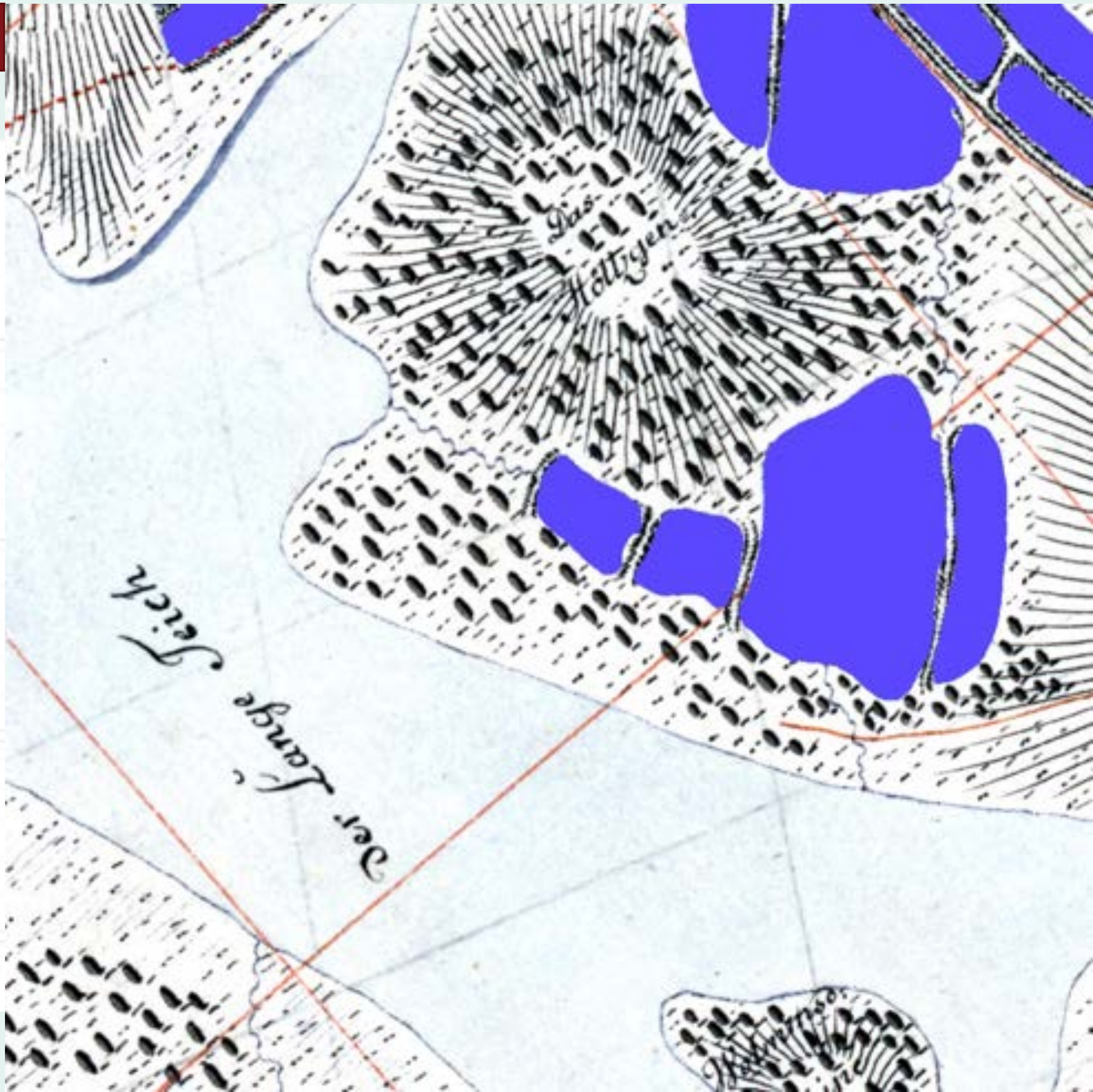




[2.5] Results
Lakes

ResNet





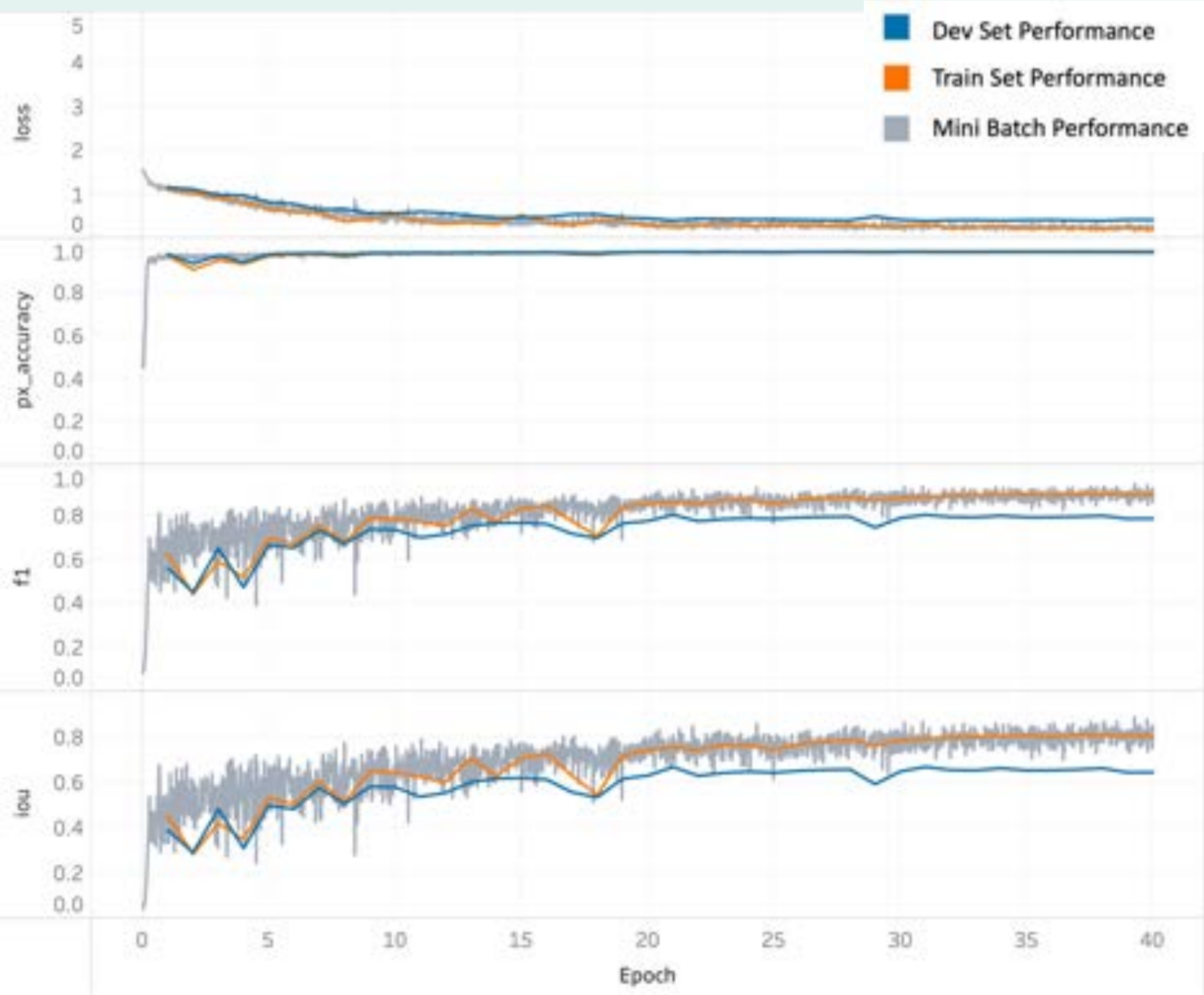
[2.5] Results

Lakes

Architecture	UNet	InceptionResNet	ResNet
Input Parameters			
Input image size (px)	512	512	512
Number of images in Train Set	800	800	800
Number of images in Dev Set	104	104	104
Hyper Parameters			
Number of epochs	*41	60	60
Mini batch size	8	12	16
Initial learning rate	0.0007	0.001	0.0007
Learning rate decay	Step	Step	Step
Optimiser	Adam	Adam	Adam
Loss function	IoU + BCE	IoU + BCE	IoU + BCE
Accuracy Parameters			
Best epoch	40	55	**40**
Pixel accuracy	0.996	0.996	**0.997**
F1-Score	0.865	0.885	**0.909**
IoU	0.763	0.794	**0.833**
Performance Parameters			
Time to compute one epoch	00:03:30	00:02:38	**00:01:41**
Total time to train	03:30:08	02:38:00	**01:41:18**

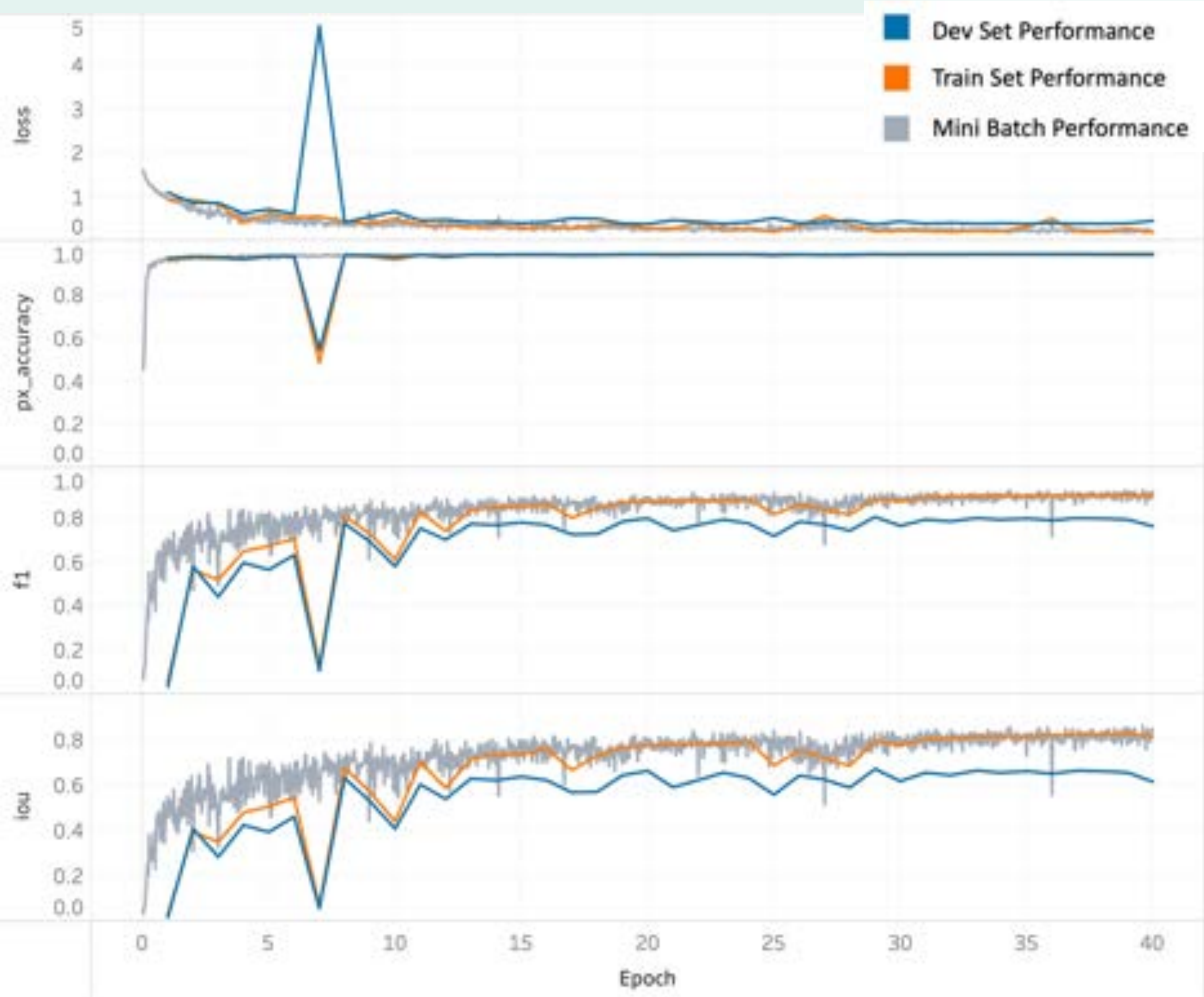
[2.5] Results
Rivers

UNet



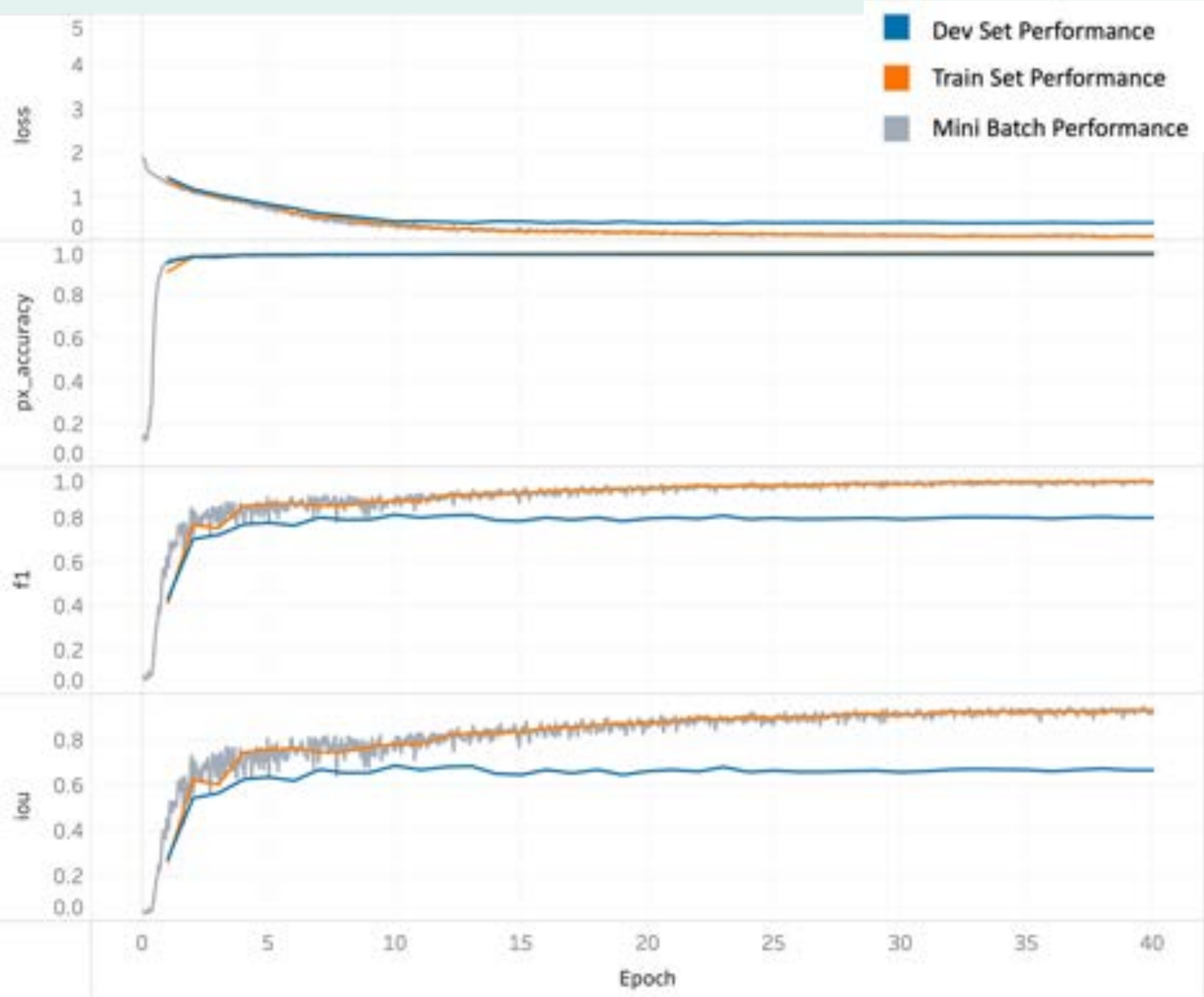
[2.5] Results
Rivers

InceptionResNet



[2.5] Results
Rivers

ResNet



[2.5] Results

Rivers

Architecture	UNet	InceptionResNet	ResNet
Input Parameters			
Input image size (px)	512	512	512
Number of images in Train Set	352	352	352
Number of images in Dev Set	88	88	88
Hyper Parameters			
Number of epochs	40	40	40
Mini batch size	8	12	16
Initial learning rate	0.0005	0.001	0.0005
Learning rate decay	Step	Step	Step
Optimiser	Adam	Adam	Adam
Loss function	IoU+BCE	IoU+BCE	IoU+BCE
Accuracy Parameters			
Best epoch	35	20	**10**
Pixel accuracy	0.987	0.988	**0.988**
F1-Score	0.789	0.796	**0.813**
IoU	0.651	0.661	**0.685**
Performance Parameters			
Time to compute one epoch	00:02:07	00:01:19	**00:00:45**
Total time to train	01:24:42	00:53:00	00:30:01

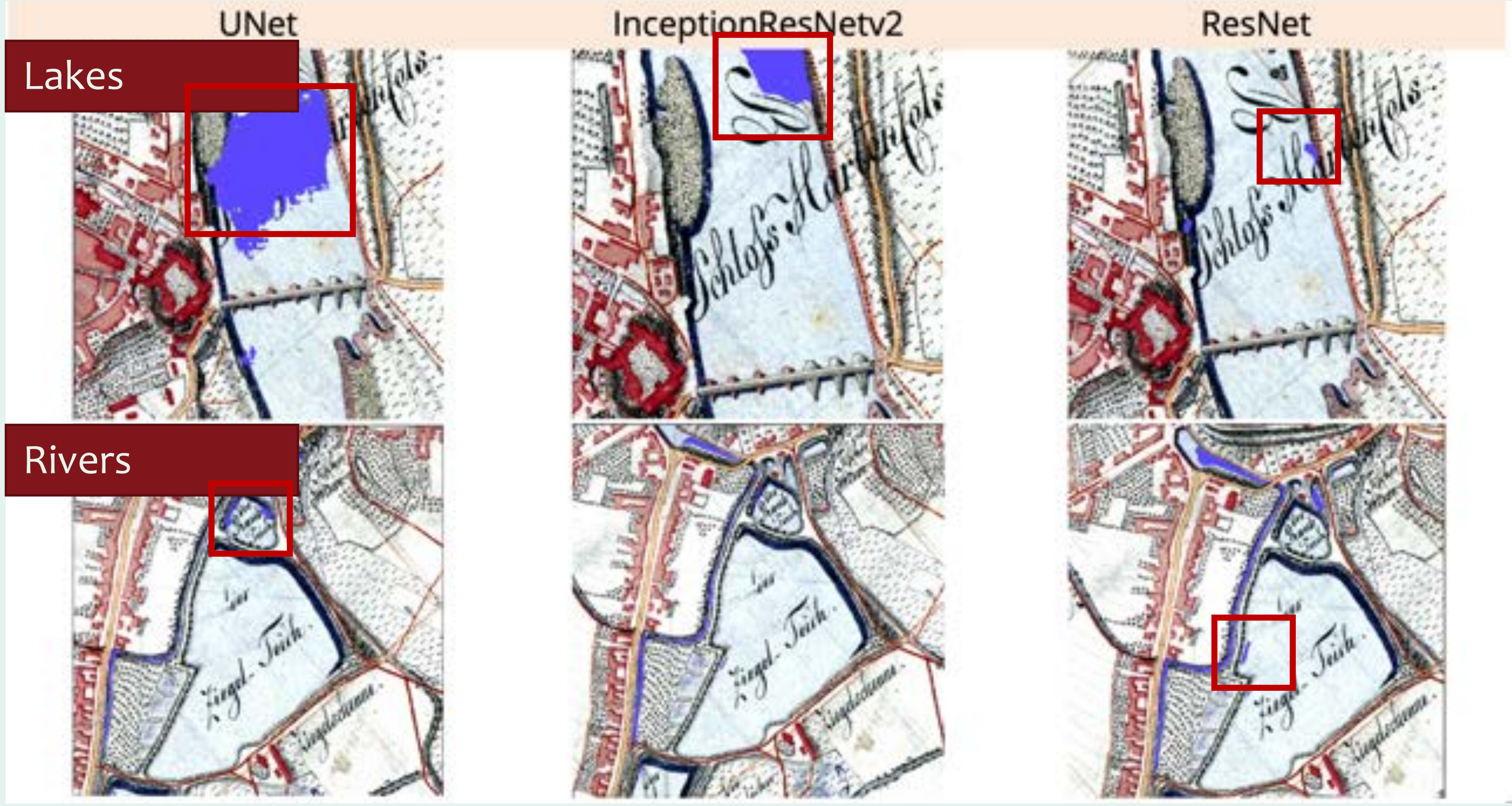
Rivers and Lakes: Misclassifications

Misclassifications occurred in Rivers and Lakes misclassifying the other class

Possible Solutions:

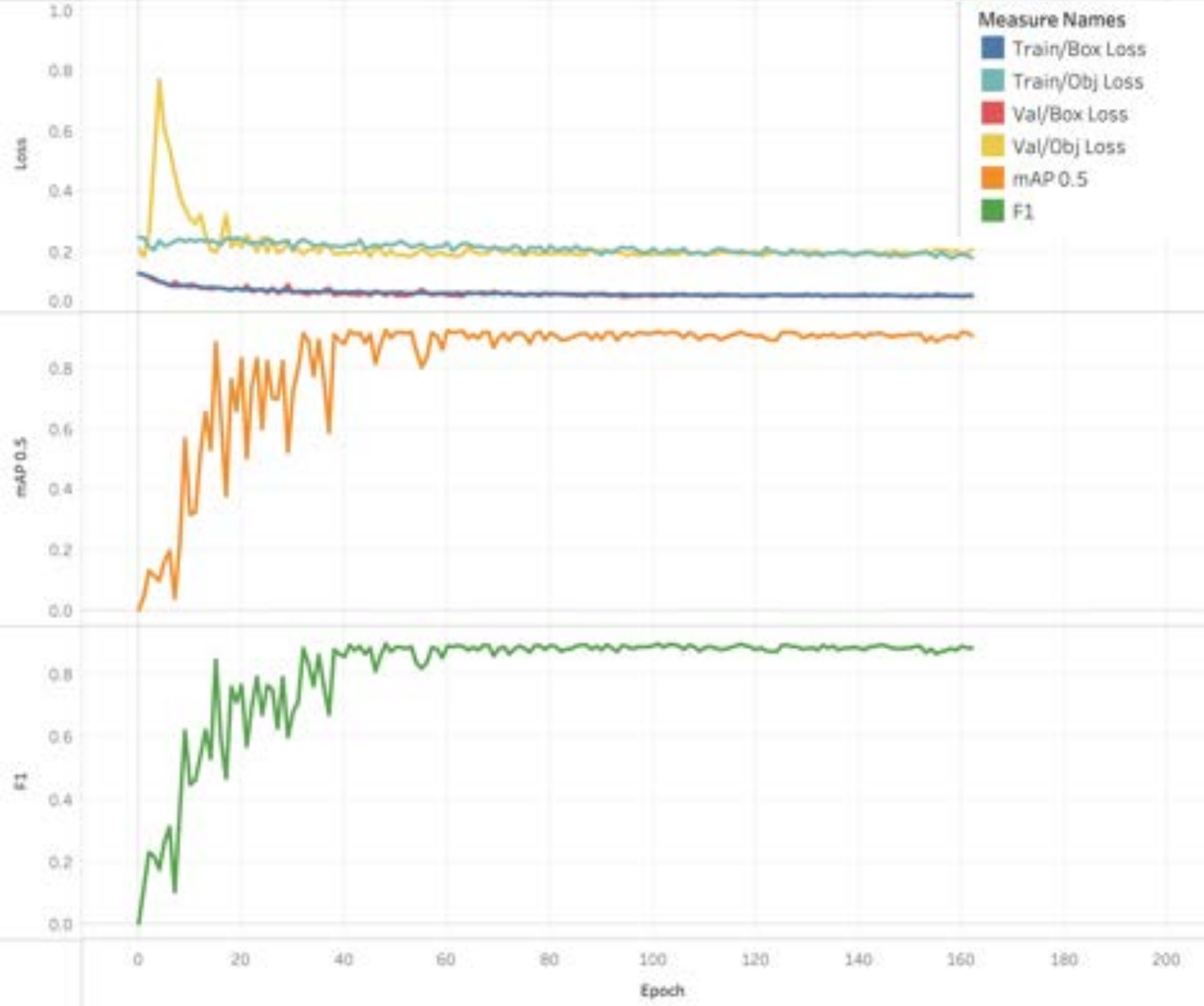
1. Including more training data

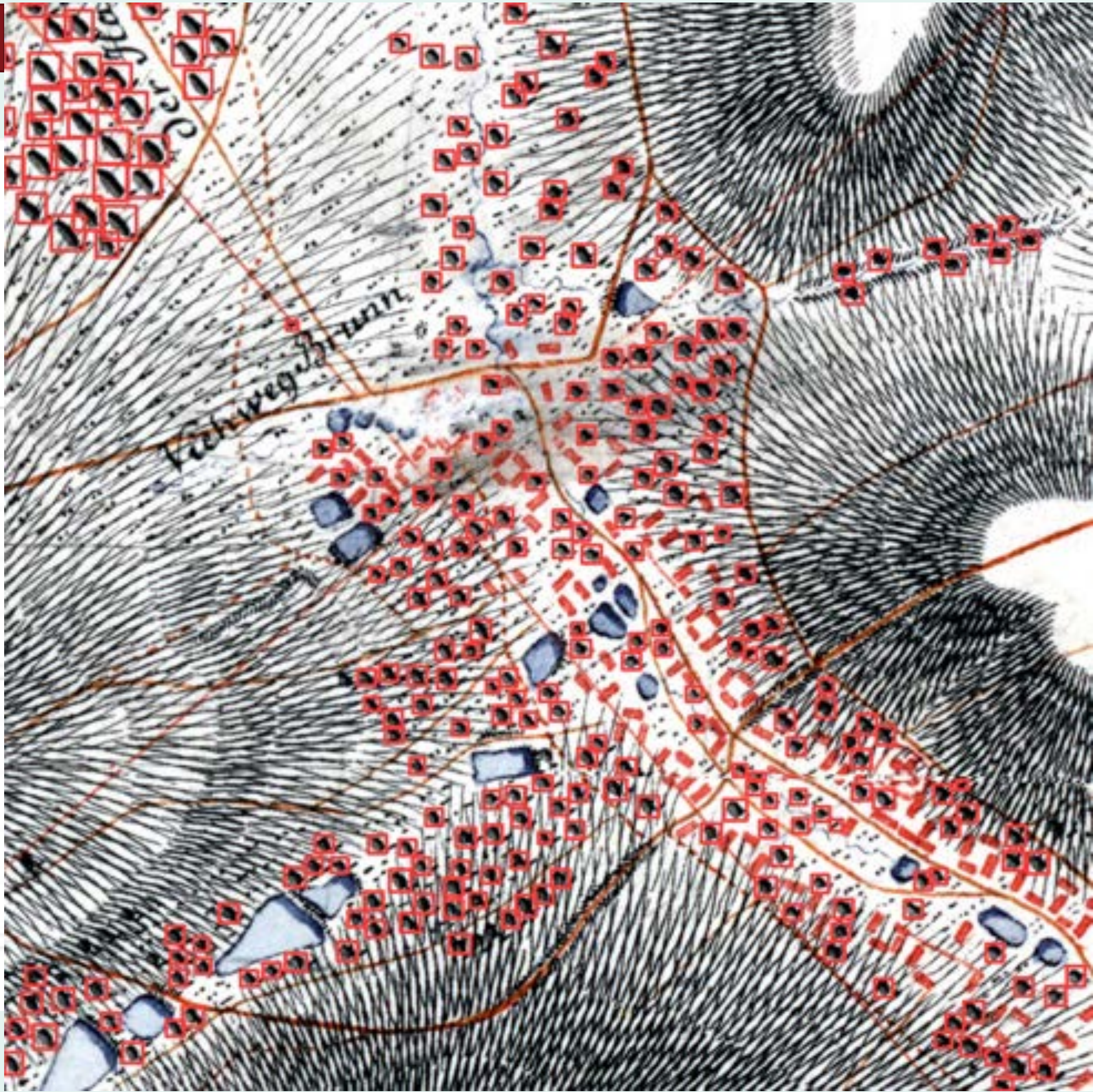
2. multi-class semantic segmentation approach



[2.5] Results
Forests

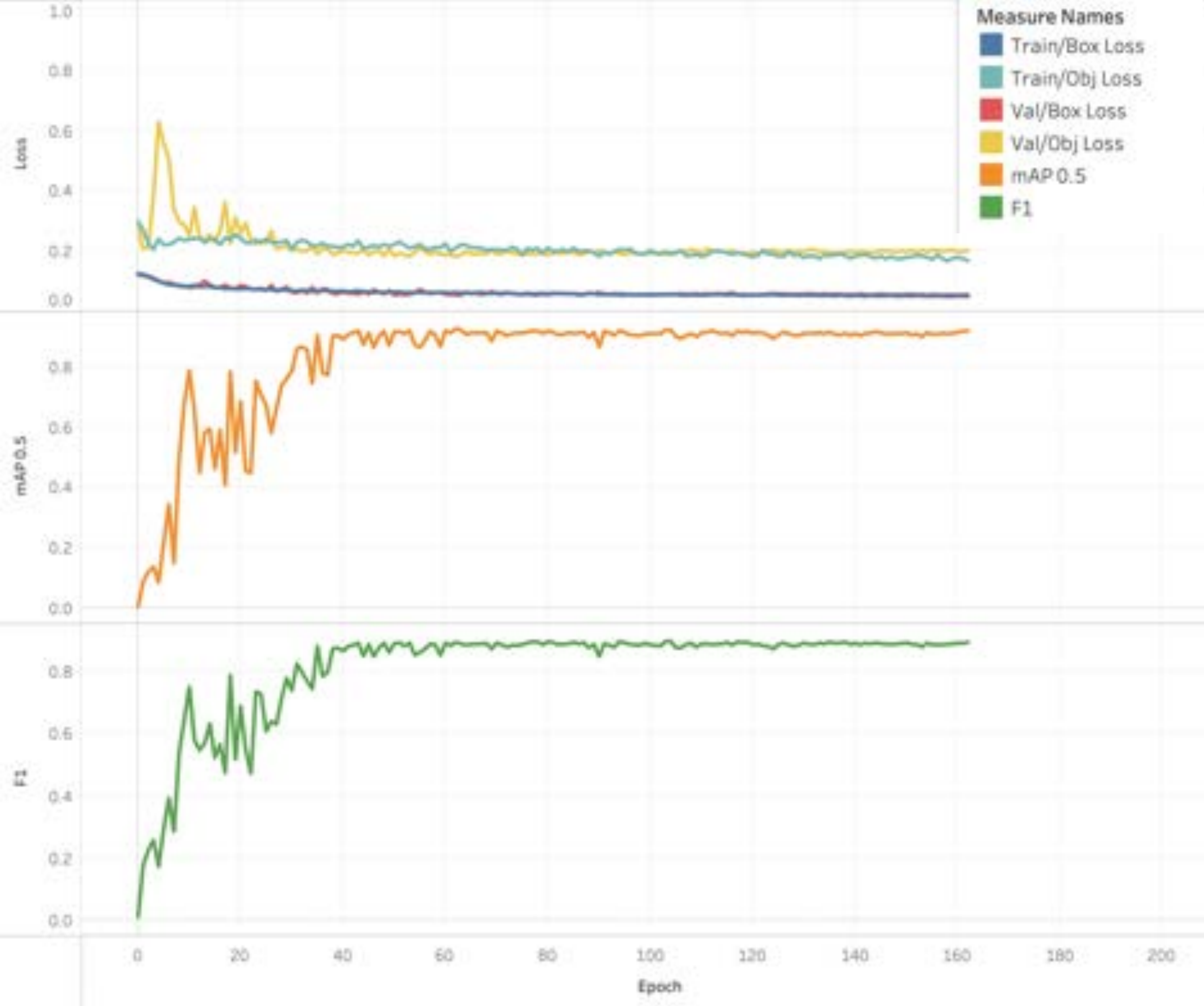
YOLO v5m

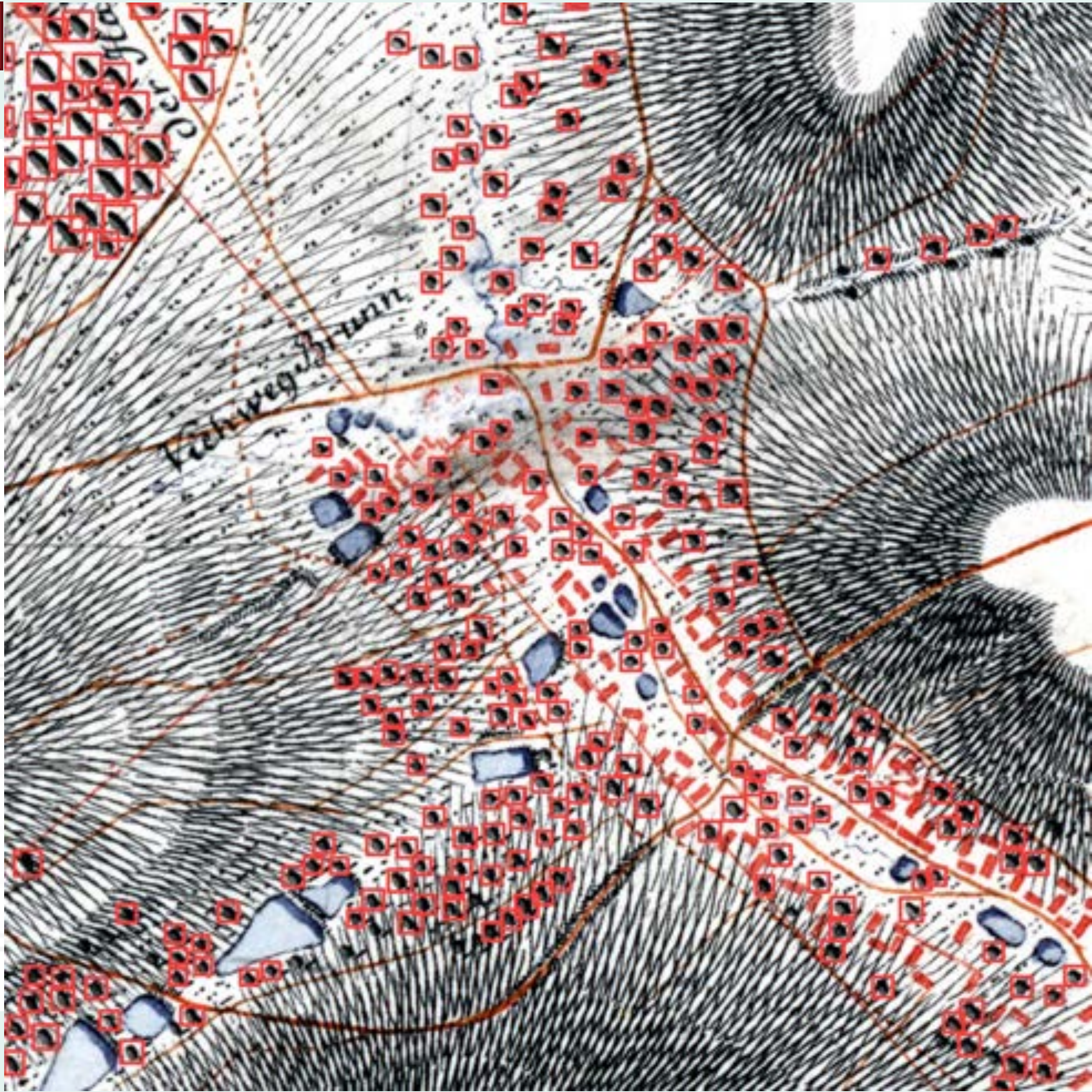




[2.5] Results
Forests

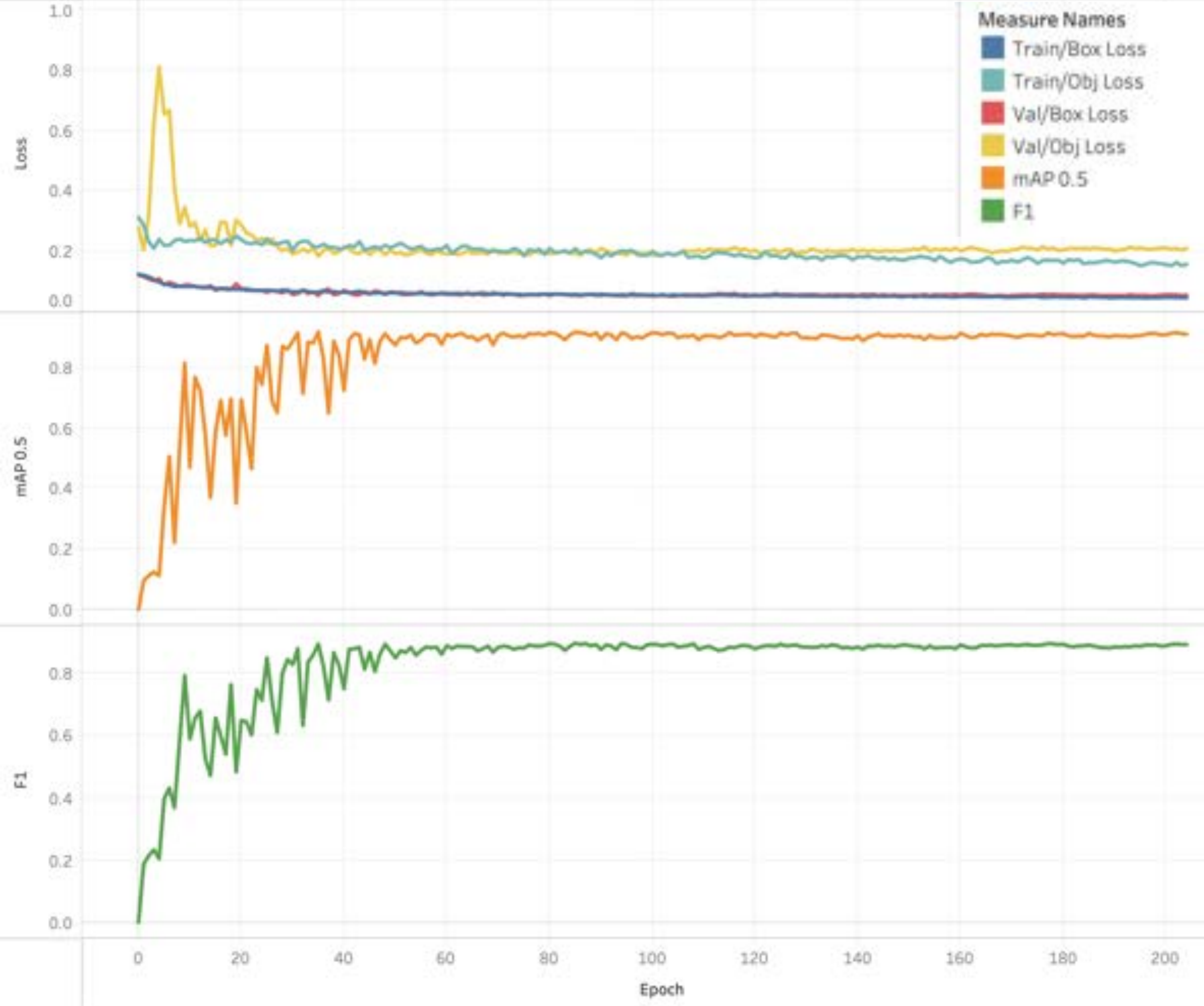
YOLO v5l

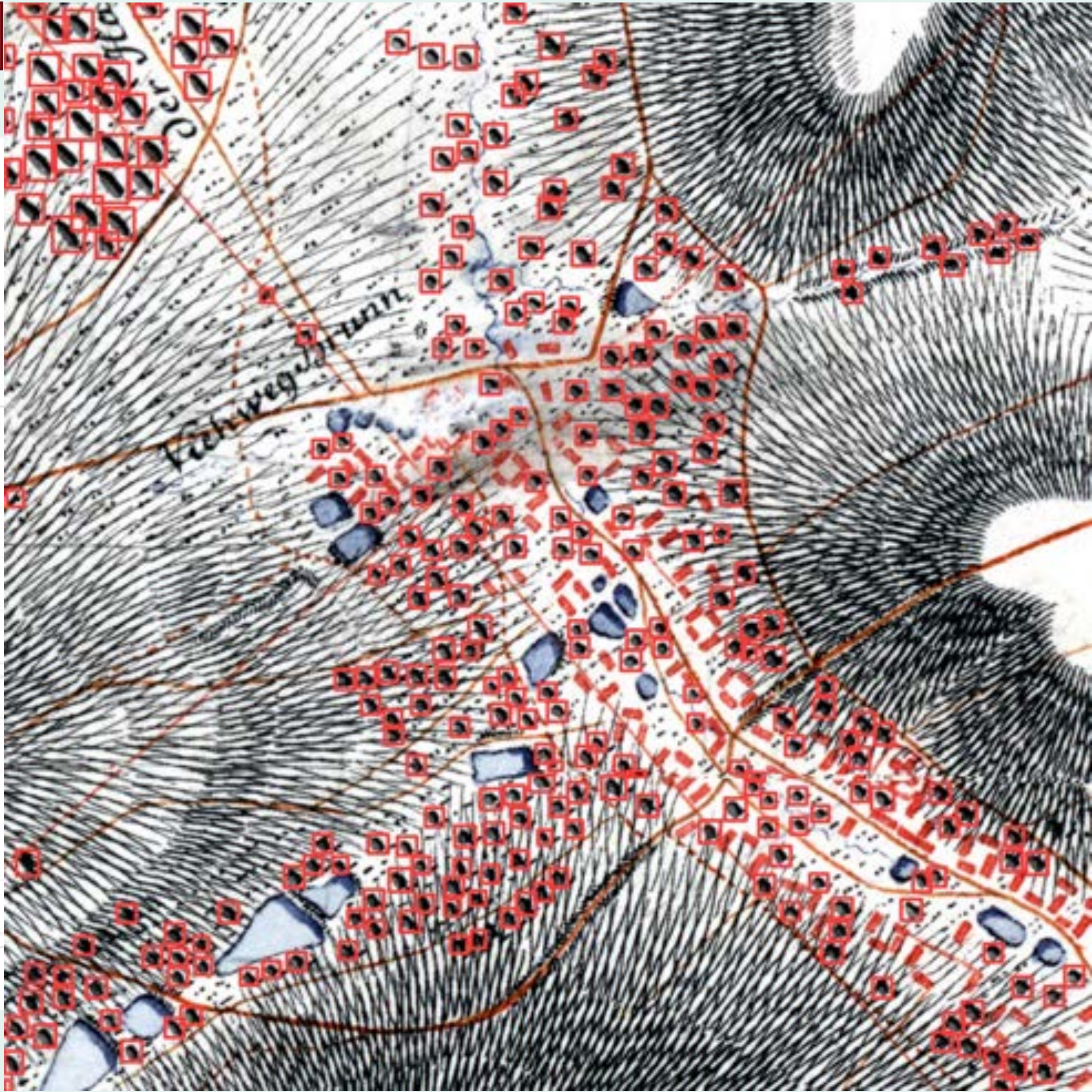




[2.5] Results
Forests

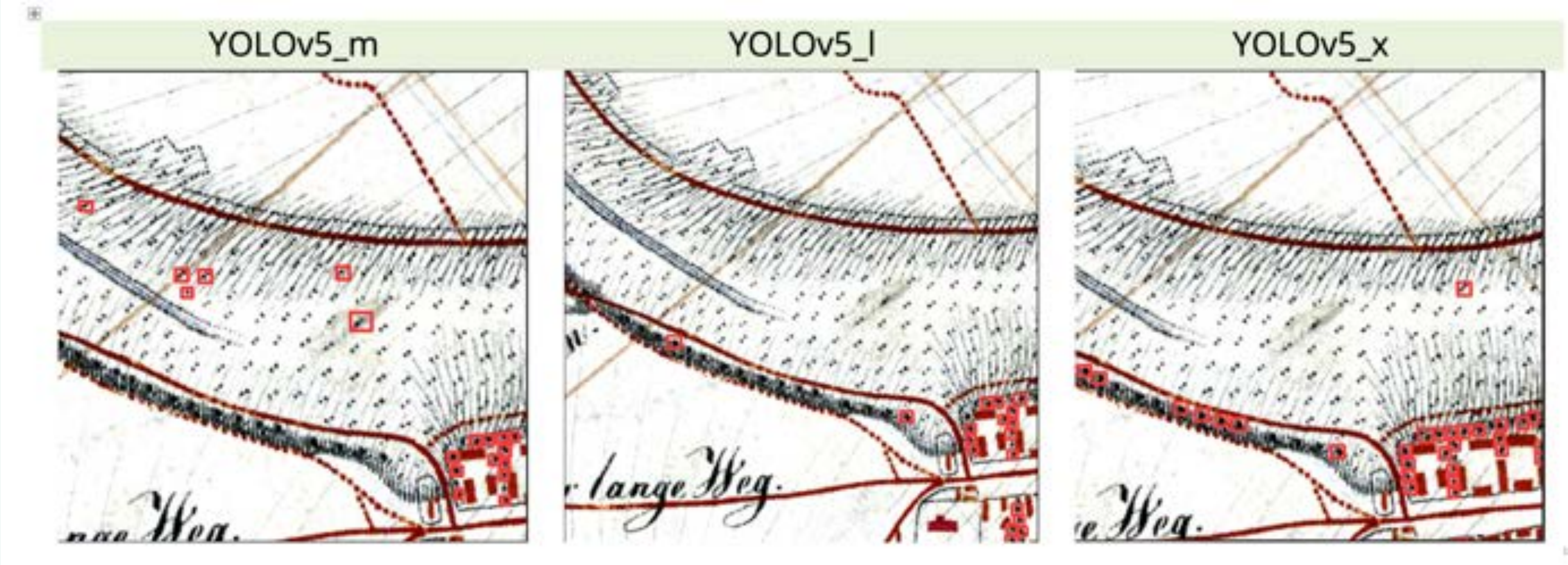
YOLO v5x





[2.5] Results

Forests: Misclassifications



- Negligible number of misclassifications in all the models. For example, the meadow symbol is misclassified in a few locations as tree symbols in YOLOv5_m and YOLOv5_x models, but the misclassifications were even less in YOLOv5_l.

- The error of omission is less in the YOLOv5_x model.

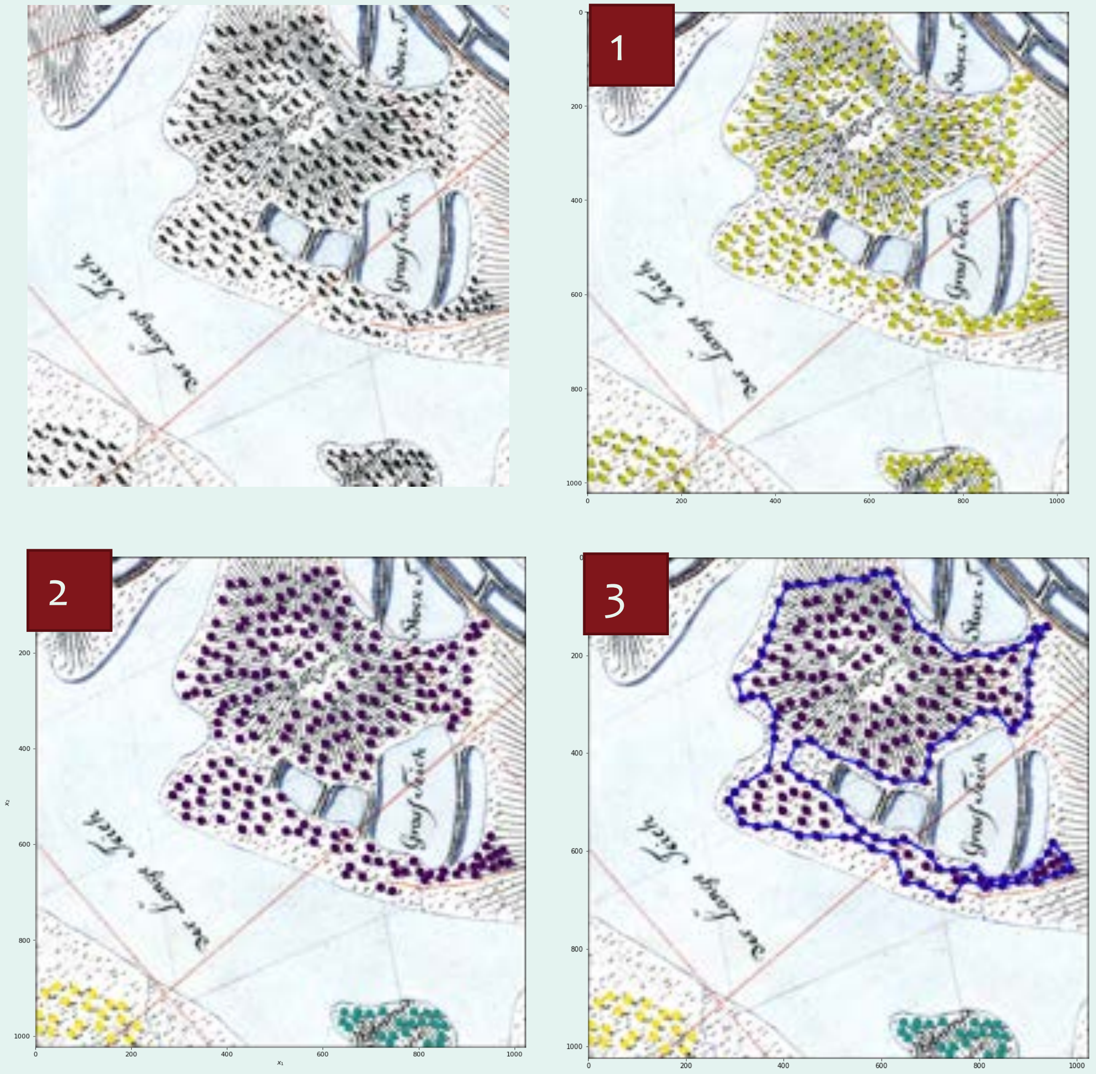
[2.5] Results Forests

Architecture	YOLOv5_m	YOLOv5_l	YOLOv5_x
Input Parameters			
Input image size (px)	416	416	416
Number of images in Train Set	213	213	213
Number of images in Dev Set	31	31	31
Average annotations per image in Train Set	80	80	80
Average annotations per image in Dev Set	86	86	86
Total annotations in Train Set	17102	17102	17102
Total annotations in Dev Set	2661	2661	2661
Hyper Parameters			
Number of epochs	200	200	200
Mini batch size	32	32	32
Initial learning rate	0.01	0.01	0.01
Learning rate decay	Constant	Constant	Constant
Optimiser	SGD	SGD	SGD
Loss function	CIoU	CIoU	CIoU
Early stopping after (epochs)	100	100	100
Accuracy Parameters			
Best epoch	100	119	101
F1-Score	0.891	0.896	0.894
mAP @0.5	0.917	0.92	0.916
Performance Parameters			
Time to compute one epoch	00:00:27	00:00:31	00:00:38
Total time to train	00:17:46	00:20:53	00:25:31

[2.5] Results
Forests: Vectorisation

Next step of extracting forest area from extracted tree symbols are

1. Centre coordinates extraction
2. DBSCAN clustering
3. Converting cluster to polygon



Current Limitations

1. Calculating DBSCAN parameters
2. Separating tree symbols belonging to the forest class and individual trees

[2.5] Results

Buildings (complex) & Meadow

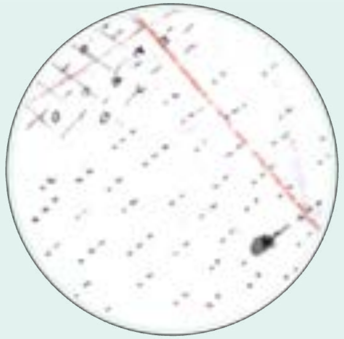
Extraction of Building Complex and Meadow classes was not able perform due to time limitation of creating training data

Similar process can be used to evaluate these two classes

Building Complex – Semantic Segmentation
Meadow – Object Detection

Buildings - Complex

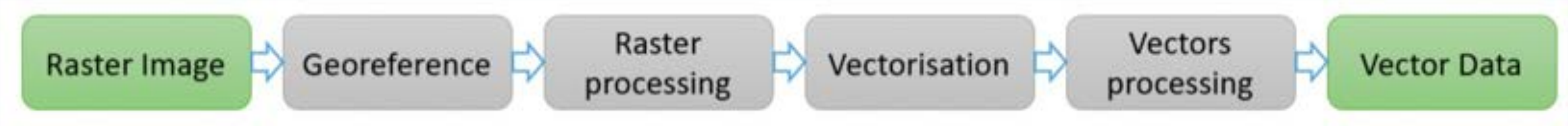
Meadow



[2.6] Next Steps

1. Single Class Segmentation to Multi Class Segmentation, classifying all the classes at once and evaluating performance

2. Extending to a complete vecotorisation workflow



Overview of a GIS-based pipeline for Digital Map Processing (Drolias & Tziokas, 2020)

[3]

CHALLENGES

Lack of Training Data

It was identified that object detection models for detecting tree symbols and the semantic segmentation model for building classification works remarkably well compared to the other classification models obtained in this study. The reason is having a good number of training data set.

Possible Solution:

Crowdsourcing

Computational Power

Training deep learning models demands a lot of computational power, which cannot be fulfilled with a consumer-grade computer. Cloud computing is one solution to these limitations. In this study, the free tier of Google Colab cloud computing service is used, which comes with resource limitations such as limited memory, GPU and time limitations.

Possible Solution:

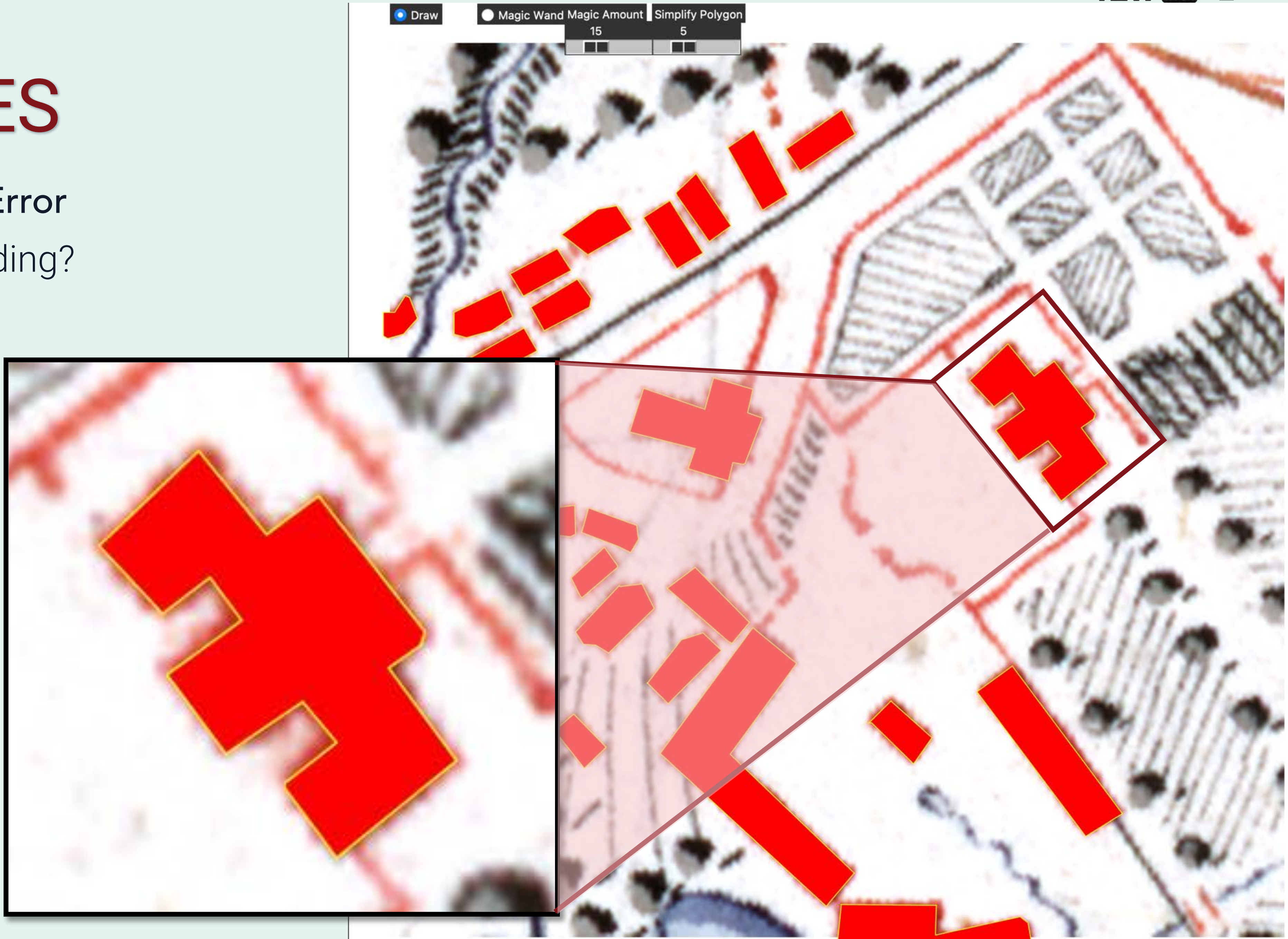
Commercial cloud computing services

[3]

CHALLENGES

Effects of Digitization Error

What is the Correct building?



[4]

CONCLUSION

1. Selection of a proper deep learning architecture has a significant influence in terms of performance and accuracy, which is an impactful factor when deploying the models in real-world applications.
2. However, solving the fundamental challenges of deep learning, such as scarcity of training data, should be addressed first to unlock the technology's full potential.
3. It can be concluded that deep learning is the technology that can make a change in digital map processing to unlock the vast amount of data hidden in historical map archives.

[5]

REFERENCES

Ayush, K., Uzkent, B., Meng, C., Tanmay, K., Burke, M., Lobell, D., & Ermon, S. (2022). *Geography-Aware Self-Supervised Learning* (arXiv:2011.09980; Version 7). arXiv. http://arxiv.org/abs/2011.09980

Chazalon, J., Carlinet, E., Chen, Y., Perret, J., Duménieu, B., Mallet, C., Géraud, T., Nguyen, V., Nguyen, N., Baloun, J., Lenc, L., & Král, P. (2021). *ICDAR 2021 Competition on Historical Map Segmentation* (arXiv:2105.13265; Version 1). arXiv. http://arxiv.org/abs/2105.13265

Erdem, F., & Avdan, U. (2020). Comparison of Different U-Net Models for Building Extraction from High-Resolution Aerial Imagery. *International Journal of Environment and Geoinformatics*, *7*(3), 221–227. https://doi.org/10.30897/ijegeo.684951

Groom, G., Levin, G., Svenningsen, S., & Linnet Perner, M. (2020). Historical Maps – Machine Learning Helps Us over the Map Vectorisation Crux. *Automatic Vectorisation of Historical Maps: International Workshop Organized by the ICA Commission on Cartographic Heritage into the Digital. Budapest – 13 March, 2020*, 91–100. https://doi.org/10.21862/avhm2020.11

Heitzler, M., & Hurni, L. (2020). Cartographic reconstruction of building footprints from historical maps: A study on the Swiss Siegfried map. *Transactions in GIS*, *24*(2), 442–461. https://doi.org/10.1111/tgis.12610

Minaee, S., Boykov, Y., Porikli, F., Plaza, A., Kehtarnavaz, N., & Terzopoulos, D. (2020). *Image Segmentation Using Deep Learning: A Survey* (arXiv:2001.05566). arXiv. https://doi.org/10.48550/arXiv.2001.05566

Thank You


UNIVERSITY OF TWENTE.
ITC
TECHNISCHE UNIVERSITÄT DRESDEN
Technical University of Munich
TUM
TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology
TU WIEN