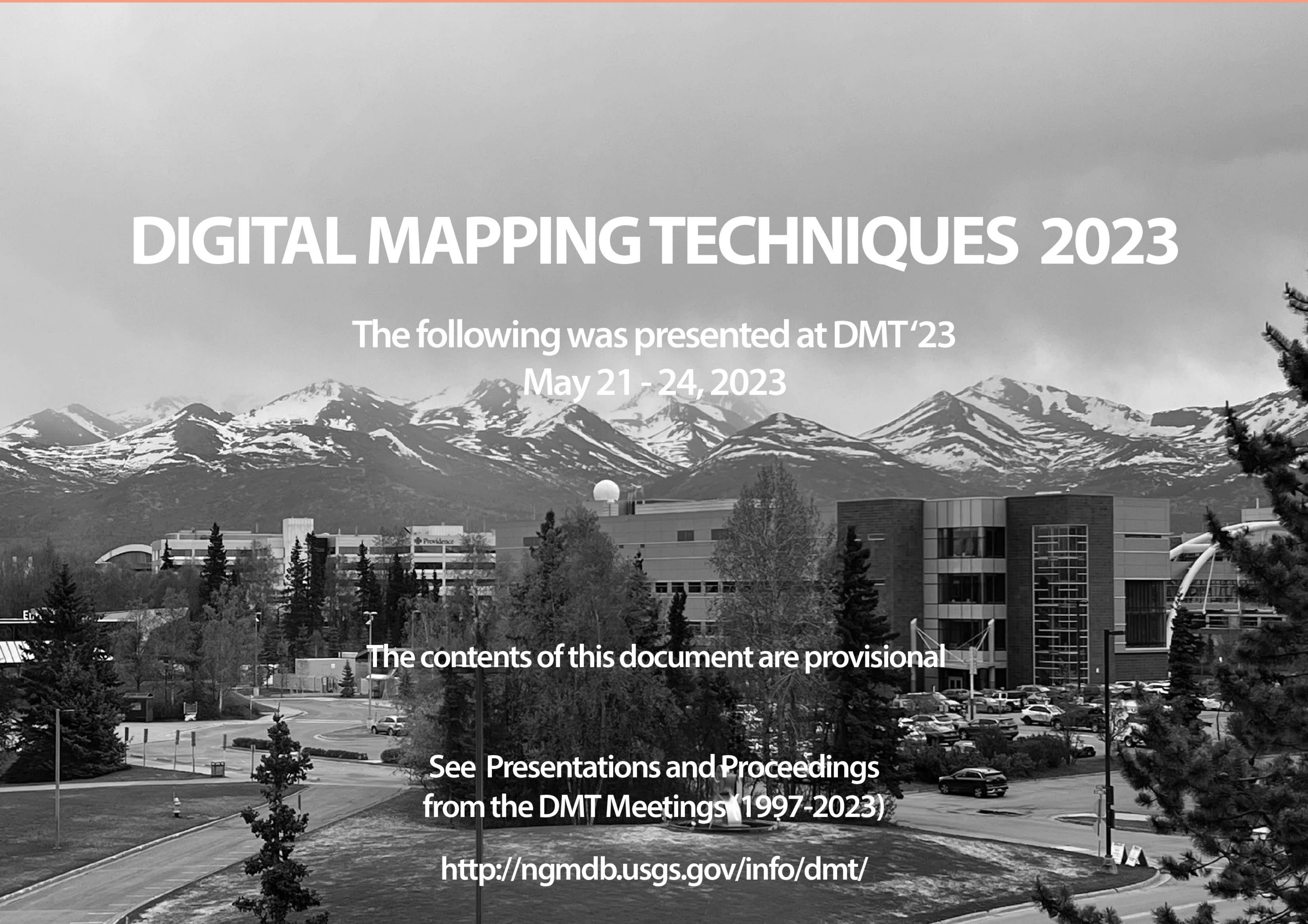


DIGITAL MAPPING TECHNIQUES 2023

The following was presented at DMT '23
May 21 - 24, 2023

The contents of this document are provisional

See Presentations and Proceedings
from the DMT Meetings (1997-2023)

http://ngmdb.usgs.gov/info/dmt/

From Paper to AI: Improving Geologic Mapping Workflows

Robert Clark, Jerry Krieger, and Darby DeBruhl — South Carolina Geological Survey

Live Life Outdoors

Preface

As the South Carolina Geological Survey (SCGS) enters its 198th year of operation, the importance of addressing its backlog of historical maps and drill logs has become increasingly evident. The process of scanning and digitizing paper products can be incredibly time-consuming, depending on the complexity of the material at hand. Up until this point, all historic map digitization at SCGS has been done heads-up, by hand, click-by-click. Traditional raster-to-vector conversion tools have presented their own challenges in their inability to distinguish the desired features of a document (*ex. contact lines*) from the undesired ones (*ex. topographic lines, such as those that may be present in a basemap*). The process of digitizing drill logs has similar obstacles; of the logs that are handwritten, many are sloppy enough to be undetectable by traditional optical character recognition (OCR), and the wide variety of differently-formatted log sheets over the years limits out many options for automation. Spurred by the ongoing conversation surrounding artificial intelligence (AI) online and in the media, SCGS has explored a range of AI-driven tools with the potential to enhance our digitization workflow. The results, while imperfect, remain promising nonetheless.

Amazon Textract

Amazon Textract is an Optical Character Recognition service provided as part of the Amazon Web Services (AWS) suite of cloud computing platforms. Textract uses machine learning algorithms pooled from various AWS services to help extract text and structured data for various types of documents. What sets Textract apart is its ability to analyze documents that do not follow a predefined template, such as a form or simple set of paragraphs. Instead, it leverages machine learning to extract text and data that align with a user-specified query, including typed or handwritten text. This feature is particularly useful for scenarios like processing handwritten drill logs we have collected at the South Carolina Geological Survey.

To complete this task, we asked Textract the following queries, with the results displayed in the spreadsheet below.

What is the drill hole number?

What is the date?

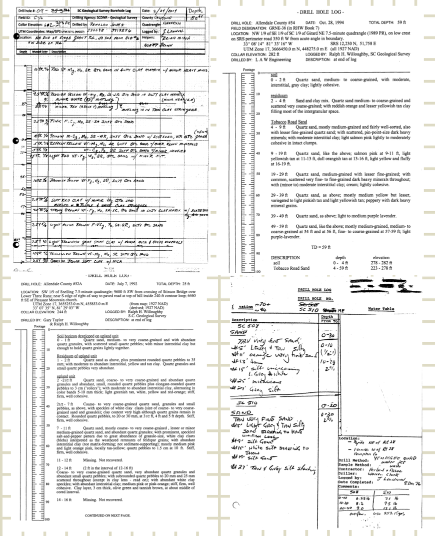
Who was it drilled by?

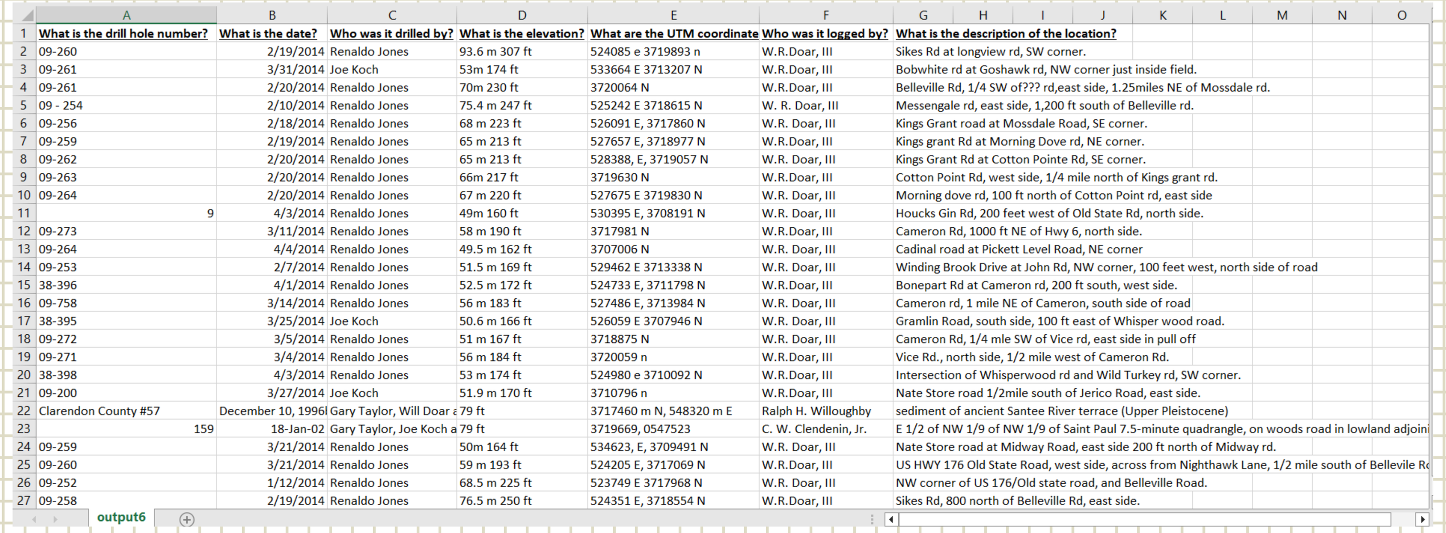
What is the elevation?

What are the UTM coordinate numbers?

Who was it logged by?

What is the description of the location?

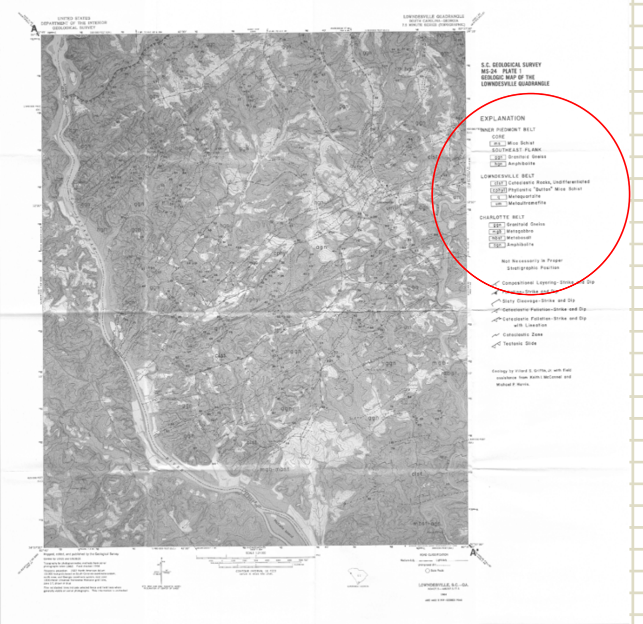


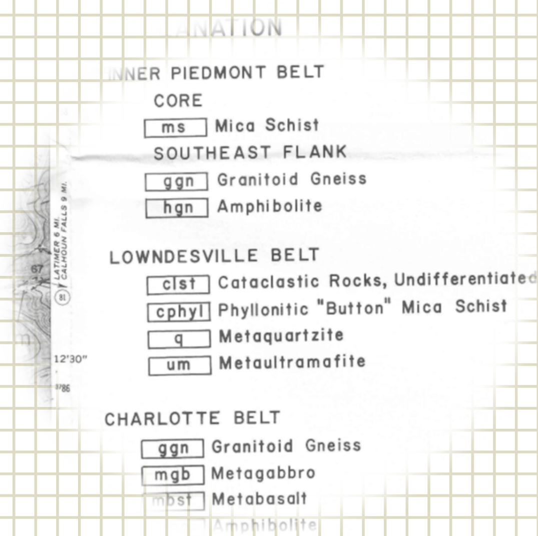


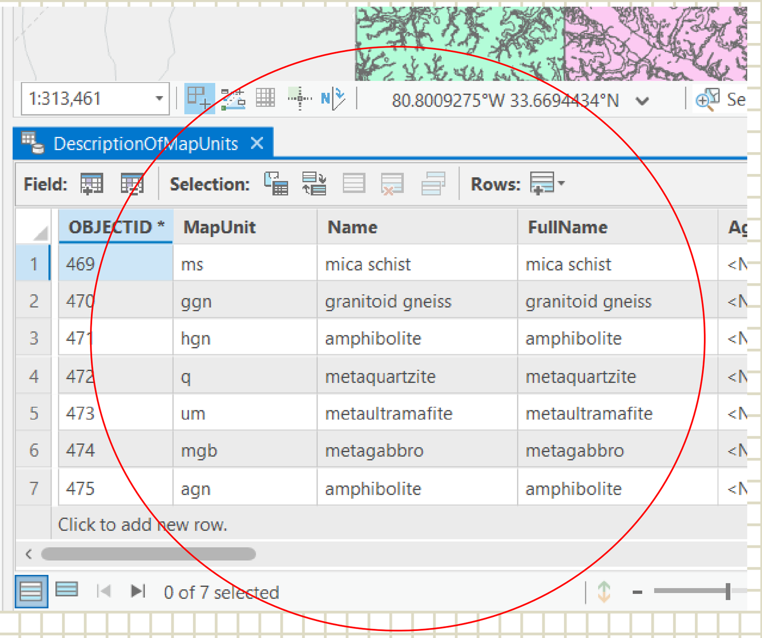
Google Vision API

While using ChatGPT and Bard as coding assistants

The Google Cloud Platform is another powerful suite of tools that provide organizations with everything from now-ubiquitous Google Drive and Google Docs, to data analysis and AI. Among its many products, we utilized the Google Cloud Vision API, one of a few powerful AI-based tools offered in Google's suite. This service provided us with deep image analysis capabilities, which we used to detect specific text within scans of our old paper geologic maps. This AI tool helped us extract specific text from scans of paper geologic maps, such as map units and their explanations, in order to fill in related fields within a corresponding DescriptionOfMapUnits file geodatabase table located in a GeMS-compliant geodatabase for that quadrangle. Google provides support for their service through a wide variety of programming languages, including Java, Python, and C#, however, we chose to use Python due to the ease of integration with ArcGIS's ArcPy Python module used to populate the geodatabase. To compensate for the lack of an in-house programmer, we heavily relied on web-based AI chatbots powered by large language models (LLMs) such as Google's Bard and OpenAI's ChatGPT. Despite OpenAI's lack of association with Google and ArcPy being a proprietary package, ChatGPT demonstrated a strong knowledge of coding and module-specific tasks.







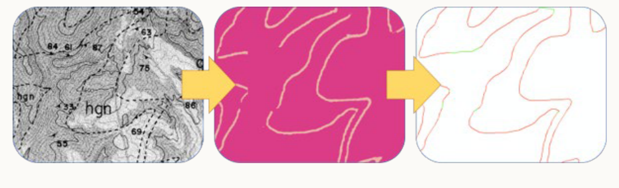
Deep Learning

Creating a pixel classification algorithm with ArcGIS Pro's Deep Learning toolset to digitize contact lines

ArcGIS Pro's Deep Learning toolset, which was initially released in 2019 as a part of the Image Analyst extension, utilizes convolutional neural networks (CNNs) to perform various image analysis tasks. A CNN is a type of neural network specifically designed for processing and analyzing visual data that is inspired by the organization and functioning of the human brain. After being trained on a human-validated dataset with known outcomes, the CNN is able to recognize patterns within the training dataset and can use this knowledge to calculate the probability of a certain condition being true or untrue — even if it has never been encountered before. ArcGIS's Deep Learning toolset offers four major applications for this technology: object detection, object classification, pixel classification, and point cloud classification. The tools are commonly used to identify objects or group them into classes within the context of satellite or aerial imagery, such as when extracting building footprints or identifying certain types of structures. SCGS's application is slightly unique — single-band, monochromatic imagery does not make for ideal training data; however, in spite of this limitation, the model's results are surprisingly accurate.

The workflow can be summarized as a three-step process, with each step creating the input for the next.

Processed — Classified — Vectorized

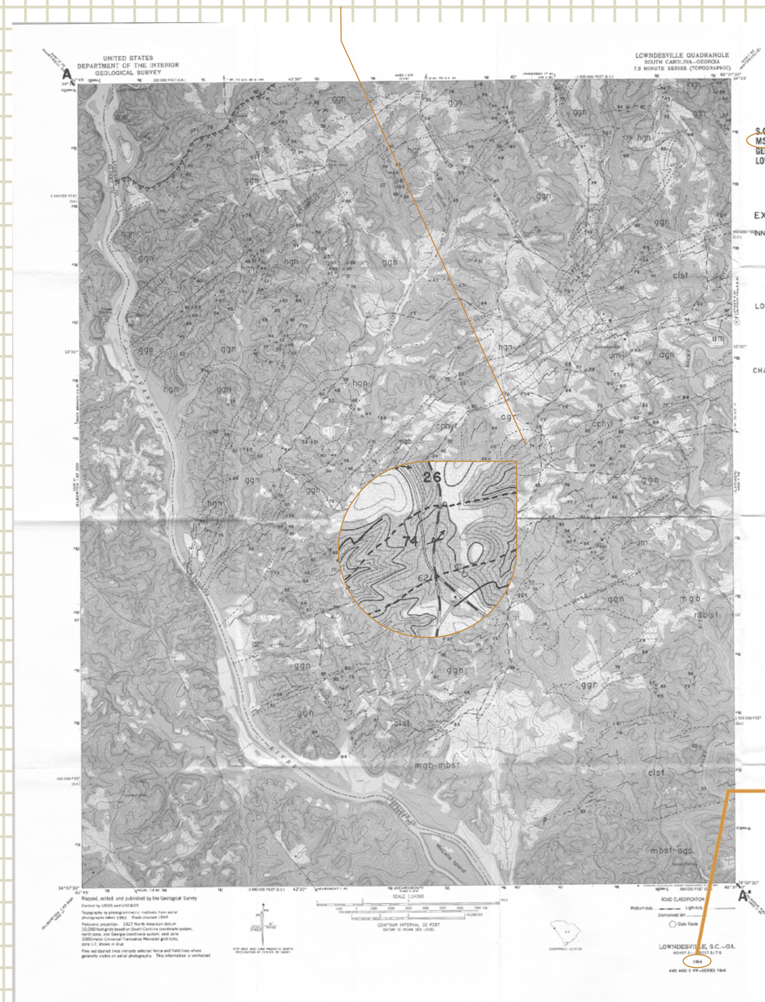


Before starting, the user should be aware that they need to manually install the required Deep Learning packages, which are updated with each ArcGIS release and are not automatically included in its initial software installation.

The Lowndesville 7.5 minute quadrangle, located in western South Carolina's Abbeville County, was digitized manually and used to train the model.

Its symbology is quite simple, but still complicated enough to make traditional raster-to-vector conversion tools impractical. Note the lack of variety in symbology — every contact line is dashed, with no visual distinction between certain, approximate, inferred, or concealed features. Compare this with the complexity of the basemap and surrounding features; the topography and roadway data overlaid on the map, as well as the orientation measurements and cataclastic zone markers, visually clutter the image in a way that would have previously required a human's input to parse. The combination of these circumstances makes deep learning a suitable solution.

It is the only digitized part of a **larger map series**; we planned to test the efficacy of the finished model by digitizing the rest of the series and comparing the results to Lowndesville.

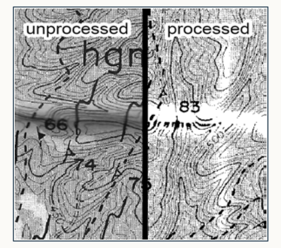


Stylistically speaking, the Lowndesville quadrangle is representative of many of our other **60s-era paper maps** — which could then be digitized with the finished model, if successful.

1. Processing the image

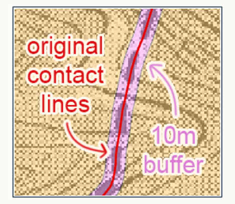
We found it massively beneficial to the reliability of the final model to take a few simple image processing steps with the training data before starting.

1. If not already done, scan the paper map, ensuring that the resolution and size of the scan match the images that are to be processed by the final model. Verify that the final raster has an 8-bit unsigned pixel type. If not, import it into ArcGIS and re-export it with the appropriate pixel type.



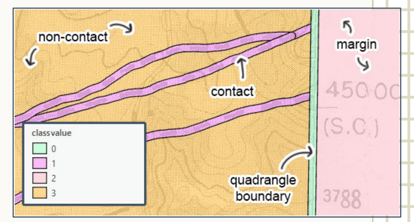
2. Open the scanned map in an image processing program, such as Photoshop, and tweak the contrast and brightness of the image. Experimenting with the black point and posterize settings may also yield good results. There is no one-size-fits-all combination of filters that will be appropriate for every map; do whatever makes the contacts stand out as much as possible in the scan at hand.

3. Georeference the raster and digitize the contacts in the training dataset. If starting from line features, apply a buffer to convert the lines into polygons, ensuring that all pixels of the feature are well within the polygon boundaries. In the Lowndesville quadrangle, a 10m buffer was adequate to cover the entire line in most areas of the map.



4. In the same feature class, create polygons for all other types of features in the original map you wish to classify, including undesired ones. For example, in the case of Lowndesville, these additional features included the map's margin, its quadrangle boundary, and everything that fell in the space between each contact (referred to as "noncontacts" in the attribute table).

5. Create a double-type field named "classvalue" and assign a unique integer to each type of feature. The actual value assigned is arbitrary – all that matters is that as it remains distinct. The values assigned here determine how pixels will be sorted in the classified raster.



6. Use ArcGIS's topology tools to rectify any gaps or overlaps within the polygon feature class, ensuring that the features are properly connected and aligned.

2. Training the model

We found that running any of the steps in section 2 with files on a network or via a remote connection can yield unpredictable errors; we highly recommend storing all files and running all processes on your local machine.

2a. *Export Training Data for Deep Learning*
This tool accepts the raster and vector data from the previous task as input and creates two outputs: a folder of metadata files, in the format specified, and a folder of image chips.

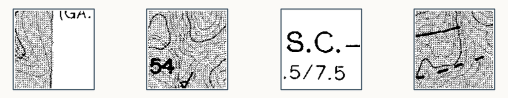


Image chips, which generally assume square or rectangular shapes, are smaller pieces that represent localized regions or segments of the larger image. They can be understood as the computer taking a snapshot; the size of the snapshot can be set with the Tile Size X and Tile Size Y parameters, and the distance between the snapshots can be set with the Stride X and Stride Y parameters. By providing a varied set of image chips, the model can learn patterns more accurately and recognize features in more diverse contexts.

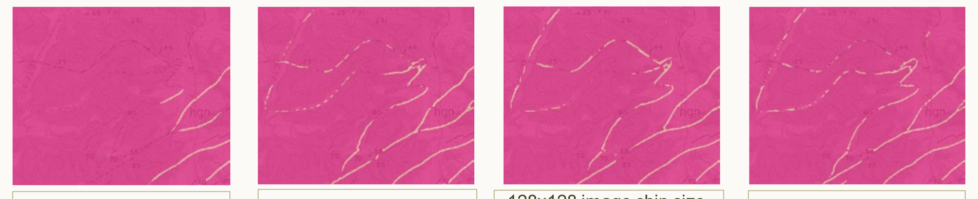
Below: Image chips of the same area, using SCGS's chosen tile size (left, 128px x 128px) vs. ArcGIS' default size (right, 256px x 256px). Notice the difference in scale between the two sizes. The default setting generated 8,192 chips, while the 128px setting generated 33,632 — a drastic increase in detail, but also in processing time.

2b. Train Deep Learning Model

This tool is by far the most computationally intensive part of this workflow. Using the image chips and metadata generated by the Export Training Data For Deep Learning tool, a model is trained to look for patterns by analyzing each pixel in the training dataset and the type of feature it belongs to.

- It is highly recommended to use a dedicated GPU to complete this task; this can be assigned in the Environment settings of this tool. Using a CPU to train a model can take several days or even weeks, but a GPU can complete the same task in a matter of hours.
- One major restriction is that only certain NVIDIA GPUs can be used for this purpose. Esri's Deep Learning frameworks utilize NVIDIA's CUDA platform, which is proprietary. The absence of compatible hardware will require that all calculations be performed with the computer's CPU instead.

SCGS trained and tested several models before settling on the optimum parameters for this dataset. The results of some of these experiments can be seen below.



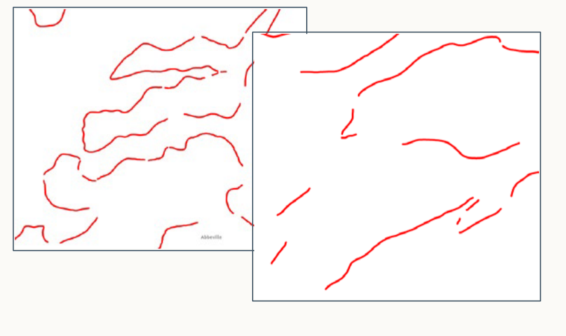
No image processing | Default image chip size (256x256) | 128x128 image chip size, training data from multiple quads | 64x64 image chip size

Ultimately, the model that seemed to perform the best on both the training map and the rest of the map series was:

- trained on a single quadrangle
- processed in Photoshop
- of the DeepLabV3 model type, which meshed better with single-band imagery than the other pixel classification models
- exported as image chips with a 128px tile size and a 64px stride size

3. Vectorizing the classified raster

After pixel classification was complete, the raster was converted back into lines via the ArcScan toolbar in ArcMap.



Once we were satisfied with the results of Lowndesville, we ran through the workflow on the other maps in MS-24. In the below figure, lines digitized autonomously by the computer can be seen in red, while corrections made by a human reviewer are in green: