Machine learning infrastructure on the frontier of virtual unwrapping

Stephen Parsons,[*] Jacob Chappell, C. Seth Parker and W. Brent Seales

University of Kentucky,
Lexington, KY, United States
E-mail: stephen.parsons@uky.edu, jacob.chappell@uky.edu, c.seth.parker@uky.edu, seales@uky.edu

Virtual unwrapping is a software pipeline for the noninvasive recovery of texts inside damaged manuscripts via the analysis of three dimensional tomographic data, typically X-ray micro-CT. Recent advancements to the virtual unwrapping pipeline include the use of trained models to perform the "texturing" phase, where the content written upon a surface is extracted from the 3D volume and projected onto a surface mesh representing that page. Trained models are critical for their ability to discern subtle changes that indicate the presence or absence of writing at a given point on the surface.

The unique datasets and computational pipeline required to train and make use of these models make it a challenge to develop succinct, reliable, and reproducible research infrastructure. This paper presents our response to that challenge and outlines our framework designed to support the ongoing development of machine learning models to advance the capability of virtual unwrapping. Our approach is designed on the principles of visualization, automation, data access, metadata, and consistent benchmarks.

International Symposium on Grids & Clouds 2021, ISGC2021
22-26 March 2021
Academia Sinica, Taipei, Taiwan (online)

[*]Speaker

1. Introduction

Virtual unwrapping [1] enables the noninvasive discovery of otherwise hidden texts inside damaged manuscripts or scrolls. The process starts with acquisition, typically imaging the artifact using X-ray micro-computed tomography (micro-CT). Next, a software pipeline progressively unveils the hidden contents until ultimately the text is revealed (Figure 1). The first phase, called segmentation, identifies and isolates the surfaces of the artifact in 3D space (Figure 1, top left). The surface mesh is then textured algorithmically based on the X-ray data surrounding the points along it (Figure 1, top right). Finally, the textured mesh is flattened to yield the previously hidden contents of the artifact (Figure 1, bottom).

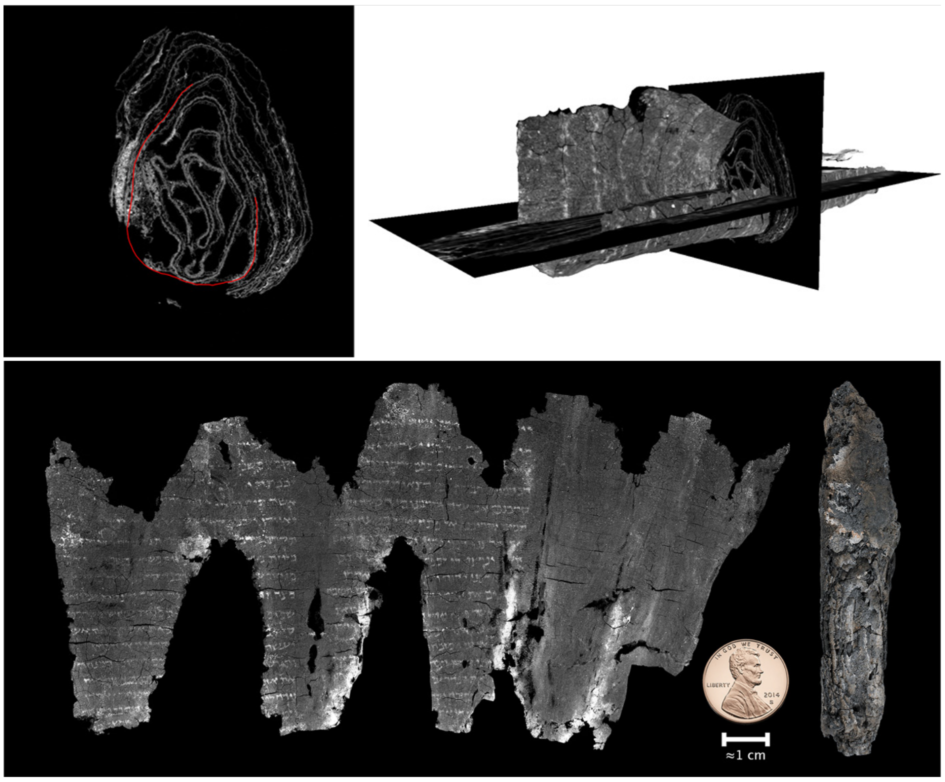

Figure 1: The virtual unwrapping pipeline (taken from [1]). Top left: segmentation stage, top right: texturing stage, bottom: result of virtual unwrapping pipeline and reference photograph of scroll.

Virtual unwrapping has been applied successfully to a growing variety of artifacts, but some are more readily processed than others. While some inks such as iron gall inks can exhibit high contrast in X-ray and are easily distinguished from the writing surface, there is commonly very low contrast between carbon inks and their written substrates. In some cases, such as the charred papyrus scrolls from Herculaneum, the substrate and ink are both carbon-based and have further been carbonized together. In this situation, the ink and substrate exhibit very similar chemical profiles, and finding methods to distinguish one from the other using noninvasive imaging is an area of active research.

Recently, we augmented the texturing phase of virtual unwrapping using convolutional neural networks, and were able to reveal carbon ink in X-ray for the first time [2]. Figure 2 illustrates the results, where the machine learning-based methods are able to significantly outperform the prior state of the art in revealing the location of carbon ink. Additionally, the neural network is capable of learning a photorealistic color representation of the page, and can generate this image from X-ray data alone.

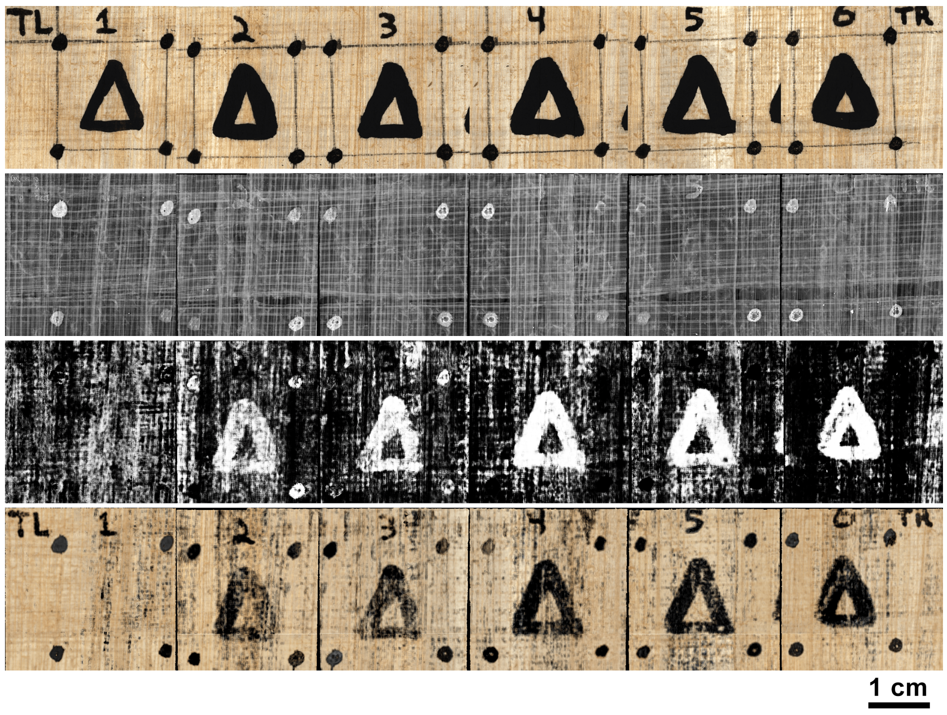

Figure 2: Prior work (taken from [2]) using neural networks to identify carbon ink from X-ray micro-CT. Column numbers indicate coats of ink applied, hence difficulty of network to learn weaker signal in leftmost column. From top: 1) reference photograph, 2) virtual unwrapping output (prior state of the art), 3) result using neural network to predict carbon ink, 4) result using neural network to recreate reference photograph.

This breakthrough result focused on lab-made scrolls and small artifacts from the Herculaneum collection. Naturally, we wish to continue this work and extend the methods to work not only on small fragments but on larger collections of fragments as well as full intact scrolls. This continued effort and the resulting research infrastructure to support it are the subjects of this paper. At its core, the development of machine learning methods relies on an iterative process. First, a model is trained, typically using a complex computational pipeline and ample processing time. Next, the model is evaluated using previously unseen data to understand expected performance on real-world tasks. Finally, adjustments are made to the machine learning model and/or the data processing pipeline, and another experiment is started.

This iterative development is easiest in an established ecosystem where datasets and tools have been built over time to support the maturation of the models and methods. An established ecosystem offers at least the following benefits:

- **Clear metrics and benchmarks** allow progress to be measured quickly and consistently. Once performance metrics and possibly test datasets are in place, one can quickly compare a model's performance against prior methods. These also allow for a clear view of the field's advancement over time, as the state-of-the-art continually improves.

- **Well understood datasets**, sometimes accompanied with a large or complete set of labels, are likely to contain few edge cases and have samples that can be easily interpreted by humans. Often, the associated task is feasible for a human agent, and thus the dataset includes a built-in sensible baseline of human performance.
- **Existing software frameworks** allow for rapid development using prior work as building blocks for constructing and fine-tuning new approaches.

One example of an established ecosystem like the one discussed here is the ecosystem around natural image classification, with a number of well-known datasets [3–5] and existing models [6–8]. The advancements and successes enabled by such an ecosystem are astonishing, In the span of a decade or less, the image classification task and the tools to support it have ushered in a new era of deep learning algorithms, software frameworks, and expert practitioners.

Of course, these ecosystems have to start somewhere, and those involved in their early construction had few existing tools at their disposal. Instead, they had to develop datasets, methods and frameworks in parallel. Working with machine learning for heritage science, we find ourselves in a similar position: we have large, varied image datasets; the proportion of labeled data to unlabeled is small; and there are few existing standards or software libraries to support the datatypes and formats involved.

We have seen the successes possible when robust ecosystems are created around particular tasks: as a direct result of the aforementioned advancements, image classification models are now widely deployed. The masses now have access to this technology which, for example, allows them to organize and search their own photographs based on image content. We aim to use this and other machine learning ecosystems as inspiration for our development of machine learning for heritage science by building datasets, tools, and models, making them widely available, and encouraging others to contribute. Eventually, what are now difficult research problems in heritage science can be solved and their solutions packaged in tools available to us all. In this particular case, we wish to make it easy for scholars around the world to reveal the hidden contents of damaged written artifacts.

Because we are building our methods and ecosystem simultaneously, we systematically build our approach based on the following principles: visualization, automation, data access, metadata, and consistent benchmarks. The next sections will discuss these principles in more detail, and the specific tools used to support them. Many of these tools are one of many options for their respective roles. While individual tools have documentation and communities of support, we have not found as much documentation of complete, end-to-end pipelines of this kind. It is our goal that by publishing this approach we can provide a realistic picture of the infrastructure required to develop machine learning methods in cultural heritage.

2. Experimental Overview

Figure 3 shows an overview of our experimental setup. This approach is designed to tighten the feedback loop between writing code and observing its effect on model performance. An experiment is typically carried out as follows: First (Fig 3.1), the data is made available on the compute cluster used to perform these tasks. In our case, the data consists of 3D X-ray micro-CT images, and supporting files we have created. Next (Fig 3.2), the actual computation takes place. During this

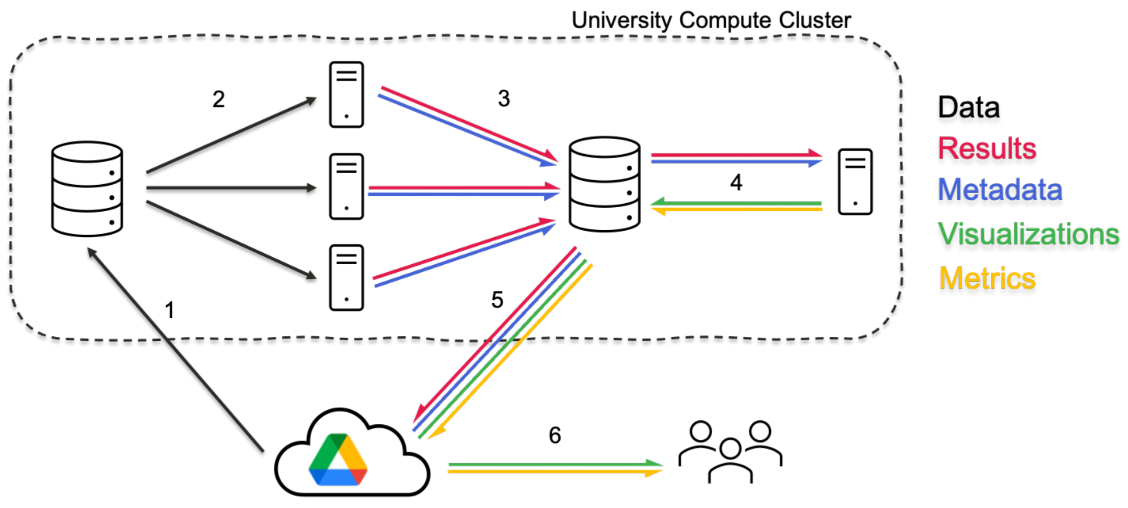

Figure 3: Experiment overview. 1) Data copied to compute cluster, 2) machine learning experiment run on compute nodes, 3) results and metadata saved to disk, 4) composite visualizations and metrics computed, 5) results, metadata, visualizations and metrics uploaded to shared cloud access, 6) team members view visualizations and metrics.

step, one or multiple neural networks are trained using the input CT data and label images we have provided. This step represents the core of the actual computation. The steps that follow are all to make sense of and sort through the results.

In (Fig 3.3), once the models are trained, they are used to generate result images. These images reflect the models' ability to reveal ink from within a scanned scroll, for instance. Metadata is also saved to disk, recording all information necessary to recreate a given experiment.

In (Fig 3.4), the result images, which may arrive as a set of many small image files, are composited into more organized visualizations that provide us with everything we want to see at a glance. Where applicable, the models' performance metrics on our set of benchmarks is also generated at this stage. Step (Fig 3.5) shows the initial results, metadata, compiled visualizations, and metrics all being uploaded back to a central location. From this point, the team can all easily access the visualizations and metrics without special software or access to the compute cluster.

3. Visualization

Direct visualization of a trained model's ability to reveal text from inside a scanned artifact is the most important tool we have to evaluate the model performance. The result of applying a trained model to a mesh inside a volume should ideally be a clear image revealing the object's content. Typically, the content is textual, and we are seeking models that generate images clearly revealing text.

Each time we train a model, many images are generated during the process. Determining whether the model is successful typically relies on a visual inspection of these images, which can be tedious and time consuming. It is preferable to get the information needed at a glance. In order to do this, we rely on a collection of custom Python scripts to automatically generate summary images after each training session. An example image in Figure 4 shows the results of a training run that trained on exposed text from multiple manuscript fragments. The images along the diagonal are the models' predicted output, while the images outlined in red represent the training data.

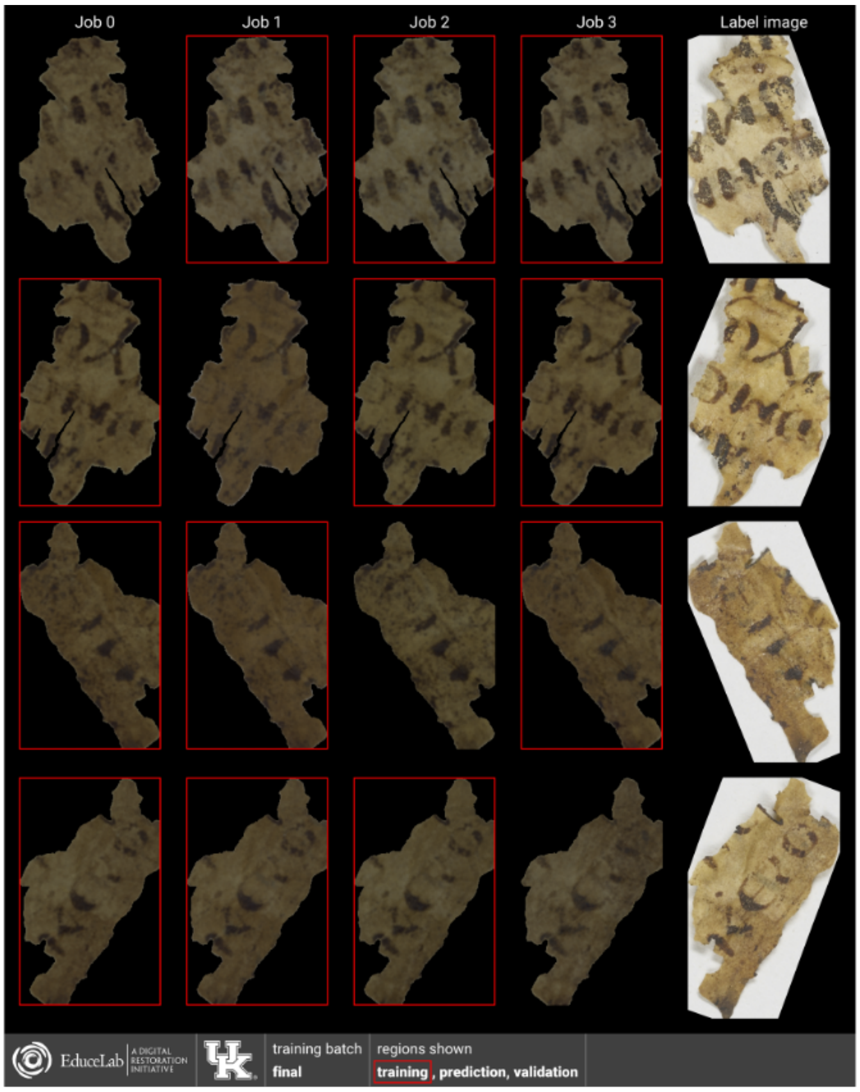

Figure 4: Visualization example from an experiment across multiple manuscript fragments. Each of the first four columns represent a separate model being trained. Each model is trained on three fragment faces (outlined in red) and validates on the other. The rightmost column shows the label photographs revealing the known correct output. Successful models generate images revealing that text, as shown here.

Composite images such as this allow us to quickly evaluate the performance of a model during and after training. There are many other visualizations generated automatically with each experiment submitted. For example, we generate animated GIF images showing the progress of the model's generated images throughout the course of a training run. These animated images can resemble Figure 4 but additionally show the models' convergence rate at a glance.

4. Automation

In the experiment overview shown in Figure 3, each step represents computation or a data transfer that must take place. When not automated, running a computational pipeline and processing the results for inspection generates significant operational overhead. We have therefore automated each of these steps in sequence, so that the entire pipeline is executed automatically with each experiment and there is a tight feedback loop between coding new experiments and viewing their results. This section briefly reviews some of the tools we use to accomplish this.

Singularity [9] allows us to develop containers with complex software environments that can then be deployed consistently across a variety of hardware. This is similar to the functionality of tools like Docker [10], but can be used without root access on the host and makes the transfer of container files straightforward.

The Slurm Workload Manager [11] is invaluable for utilizing the resources of the compute cluster available at our university. Using Slurm, it is possible to queue an array of experiments to be run which will then execute when their required hardware becomes available. Additionally, we use Slurm to define job dependencies so that summary visualization jobs can run only after their training jobs have all completed. Slurm can also be configured to send email notifications when jobs are complete, which is helpful as the jobs often wait in a queue before executing and then can run for multiple days before completion.

This pipeline requires data transfers at various stages, and Rclone [12] has been a capable command line tool for this purpose. Rclone supports a wide variety of storage providers, and in our case creates automatic backups of results and visualizations to Google Drive.

5. Data

The datasets used in our research can be terabytes in size, have hundreds of thousands of individual files, and are updated frequently. The machine learning experiments further create many files each, and the resulting visualizations need to be immediately visible to the research team without having to log into a dedicated research machine. These two requirements can be seen as the need for programmatic access to remote storage (for automation) and web access to remote storage (for users). Many storage solutions are ideal for one of these access pathways, but are not well suited for the other. The best solution for us given these requirements has been to use Google Drive. Rclone, as mentioned above, can automatically upload results to Google Drive where they are then accessible to the team from any device. Google Drive does have several limitations on the number and size of files and is not optimized to be used in this way. That said, given other considerations such as cost, this has made the most sense at our institution. In our case this is due largely due to the Google Drive Education Program, provided as a free resource by the University. We suspect that the most appropriate storage and transfer solutions would vary by institution.

6. Metadata

In machine learning, but particularly in cultural heritage applications, it is important that each experiment is transparent and reproducible, both by ourselves and by independent reviewers. While executable code can be and is readily shared amongst researchers, machine learning experiments are also defined by the complex interaction between the input datasets, the model being evaluated, and the parameters used for training and evaluation. To ensure end-to-end reproducibility for our experiments, a JSON file containing all run time parameters is created and stored alongside each experimental run.

```json
{
    "Arguments": {
        "data": "/pscratch/seales_uksr/dri-datasets-drive/MorganM910/MS910.volpkg/
            working/segmentation/fragments_p60.json",
        "output": "/pscratch/seales_uksr/dri-experiments-drive/inkid/results/MS910/p60/
            spatial_subvolumes/01",
        "feature_type": "subvolume_3dcnn",
        "label_type": "rgb_values",
        "loss": "cross_entropy",
        "subvolume_method": "nearest_neighbor",
        "subvolume_shape_microns": [
            384.0,
            384.0,
            384.0
        ],
        "subvolume_shape_voxels": [
            48,
            48,
            48
        ],
        ...
    },
    "Region set": {
        "ppms": {
            "frag1positive": {
                "path": "/pscratch/seales_uksr/dri-datasets-drive/MorganM910/MS910.
                    volpkg/working/segmentation/fragment1/007_M910F1_50kV_200uA_8um/v3 (
                    skeleton, 1 PPM)/positive/frag1positive.ppm",
                "volume": "/pscratch/seales_uksr/dri-datasets-drive/MorganM910/MS910.
                    volpkg/volumes/20200519191237",
                ...
    },
    "Command": "/usr/local/dri/ink-id/.venv/bin/inkid-train-and-predict /pscratch/
        seales_uksr/dri-datasets-drive/MorganM910/MS910.volpkg/working/segmentation/
        fragments_p60.json /pscratch/seales_uksr/dri-experiments-drive/inkid/results/
        MS910/p60/spatial_subvolumes/01 --subvolume-shape-voxels 48 48 48 --final-
        prediction-on-all --prediction-grid-spacing 8 --label-type rgb_values --
        subvolume-shape-microns 384 384 384",
    "Git hash": "608e97",
    "SLURM Job ID": "129864"
}
```

Figure 5: Abridged example of a metadata JSON file created during a machine learning experiment. The file contents contain all information required to reproduce this experiment.

Figure 5 shows an abridged example of one of these metadata files. The metadata stores all arguments passed to the program as well as the complete executed command. In addition, information about the complete datasets used for training, validation, and prediction is recorded (abbreviated here under “Region Sets”). Finally, version information about the code itself is stored in the form of a Git hash so that this code could be loaded again if needed. Additional information is stored such as the Slurm job identification number, and validation metric results are added to this metadata file at the end of an evaluation run.

Storing and using these metadata files has been critical to our team's ability to reproduce and verify our own results. As we are continually running new experiments, it becomes impossible to remember or manually record the details behind each image we produce. The metadata files keep a complete record that tells us exactly what was done to create any given image, and we frequently make use of this information. One example usage is to retry a previous experiment with a new technique, where we want to make sure to hold all other variables constant. This information has also been crucial to us as we share our generated images with their respective textual scholars. Keeping a complete provenance record allows us to answer any questions about the methods used to generate images, which can affect the way a scholar interprets or interacts with the text.

7. Benchmarks

Since this collection of datasets is being newly developed, there are no existing benchmarks or performance standards against which we can measure our trained models. As a result, it has become critical to establish some of our own new benchmarks so that there are consistent measures across different methods and implementations.

In our particular case, we are interested in generating images that reveal the hidden contents of the artifact. For those cases where we have exposed text available, we can use photographs of those surfaces as label images against which we can compare the generated results. A number of metrics are used to evaluate the similarity between the generated images and their respective ground truth images. The structural similarity index measure (SSIM) [13], peak signal-to-noise ratio (PSNR), mean squared error (MSE) and others can be used jointly, as they tend to measure different aspects of the image similarity. MSE better measures the different between image histograms, while more structural measures like SSIM better measure the presence of desired details in the image. The value of these benchmarks is best realized when they are integrated into the automated pipeline, as discussed in Section 4. In this setting, leaderboards can measure state-of-the-art performance on our various tasks, and these can be automatically updated when new models are trained. This is a significant improvement over trying to remember which models created the best looking images, and manually browsing through many prior experiments to see if we have achieved a new best.

Well-defined and automated benchmarks like these have aided the great successes of datasets such as [3–5], and we wish to model those successes. It is our goal that by being deliberate at this stage, we can enable similar successes in the heritage science realm.

8. Discussion

In developing new models and datasets within heritage science, we are up against the simultaneous challenges of developing novel methods and a supporting ecosystem. We believe these challenges are best met with a deliberate and systematic approach, that iteratively "bootstraps" from the initial phases into a mature framework and community. The focus areas of visualization, automation, data, metadata, and benchmarks have allowed us to make significant progress towards this vision.

There are challenges to this approach. Notably, the simultaneous development of datasets, software frameworks, test infrastructure, and experimental models leads to slow development as team members can be spread thin by multitasking. We make an effort to focus on one area at a time, improve it, and then cycle through the others in order to provide systematic and consistent improvement. In practice, we sometimes end up working on system improvements that span multiple of these areas and can be slow to integrate. These are best approached by careful feature design and separation, and by team planning and communication, as is true elsewhere in program design.

Another significant challenge in the past year of research has been coping with the global COVID-19 pandemic. While it has been difficult, the new constraints have actually aligned well with many of the objectives outlined in this paper. Supporting a remote research team has forced us to automate as much as possible, to make clear visualizations that are easy for other team members to access, and to document our processes and results. These actions have helped us make strides in our machine learning infrastructure amidst an otherwise demanding environment.

While our work focuses now on heritage science, we believe that the principles outlined in this paper would be beneficial to other research teams working to create machine learning methods and datasets in any new domain. While the data and tools will vary according to different research environments, the systematic approach adds structure to what can otherwise be a disorderly undertaking.

Acknowledgments

We would like to thank Christy Chapman for extensive operational support and Mami Hayashida for infrastructure support and feedback. We thank the University of Kentucky's Center for Computational Sciences and Information Technology Services Research Computing for use of the Lipscomb Computing Cluster resources. This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. 1839289. Any opinion, findings, and conclusions or recommendations expressed in this material are those of the authors(s) and do not necessarily reflect the views of the National Science Foundation. This material has been made possible in part by the National Endowment for the Humanities: Democracy demands wisdom. Any views, findings, conclusions, or recommendations expressed in this article, do not necessarily represent those of the National Endowment for the Humanities.

Supplementals

The code implementing the processes outlined in this paper can be found at `https://gitlab.com/educelab/ink-id`.

References

[1] W.B. Seales, C.S. Parker, M. Segal, E. Tov, P. Shor and Y. Porath, *From damage to discovery via virtual unwrapping: Reading the scroll from en-gedi*, *Science advances* **2** (2016) e1601247.

[2] C.S. Parker, S. Parsons, J. Bandy, C. Chapman, F. Coppens and W.B. Seales, *From invisibility to readability: Recovering the ink of herculaneum*, *PloS one* **14** (2019) e0215775.

[3] A. Krizhevsky and G. Hinton, *Learning multiple layers of features from tiny images*, *Master's thesis, Department of Computer Science, University of Toronto* (2009) .

[4] Y. LeCun, C. Cortes and C. Burges, *Mnist handwritten digit database*, *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist* **2** (2010) .

[5] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma et al., *ImageNet Large Scale Visual Recognition Challenge*, *International Journal of Computer Vision (IJCV)* **115** (2015) 211.

[6] A. Krizhevsky, I. Sutskever and G.E. Hinton, *Imagenet classification with deep convolutional neural networks*, *Advances in neural information processing systems* **25** (2012) 1097.

[7] K. Simonyan and A. Zisserman, *Very deep convolutional networks for large-scale image recognition*, *arXiv preprint arXiv:1409.1556* (2014) .

[8] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov et al., *Going deeper with convolutions*, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

[9] G.M. Kurtzer, V. Sochat and M.W. Bauer, *Singularity: Scientific containers for mobility of compute*, *PloS one* **12** (2017) e0177459.

[10] D. Merkel, *Docker: lightweight linux containers for consistent development and deployment*, *Linux journal* **2014** (2014) 2.

[11] A.B. Yoo, M.A. Jette and M. Grondona, *Slurm: Simple linux utility for resource management*, in *Workshop on job scheduling strategies for parallel processing*, pp. 44–60, Springer, 2003.

[12] N. Craig-Wood, "Rclone."

[13] Z. Wang, A.C. Bovik, H.R. Sheikh and E.P. Simoncelli, *Image quality assessment: from error visibility to structural similarity*, *IEEE transactions on image processing* **13** (2004) 600.