Functional Requirements and Design Constraints of Machine Learning in Digital Pathology

Leo Berman, Albert Bulik, Yuan Nghiem

The scope of the digital pathology project can be separated into two categories: functional requirements and design constraints. Functional requirements communicate our objectives while design constraints communicate our self-imposed and natural boundaries when conducting research.

Functional Requirements

Functional requirements outline what we intend to achieve and the metrics we use to determine if a goal has been met. Requirements are separated into negotiable and non-negotiable priorities (Table 1, Column 1). Non-negotiable priorities are critical to the project's success as a whole and negotiable priorities are stretch goals we may pursue if other requirements are satisfied. Columns 2-4 (Table 1) list the associated requirement and its metric target, and column 5 is a description or citation of its utility. See the table below.

TABLE I

Functional Requirements				
Priority	Requirement	Metric	Target Value, Range, Pass/ Fail	Justification
Non-negotiable	Inter-Rater Reliability	Cohen's kappa coefficient [0,1]	0.8	[1]
Non-negotiable	Class Dependent Performance	F1 score [0,1]	0.95	[2]
Non-negotiable	Whole-slide image (WSI) accuracy for each slide	{Pass,Fail}	Pass	[3a] [3b]
Non-negotiable	Semi-Supervised Model	Utilization of unlabeled images in final model	100%	[4a] [4b]

		(Percent)		
Non-negotiable	Graphical user interface (GUI) segments slides or tissue area	{Pass,Fail}	Pass	Presentation should communicate salient features to an untrained audience.
Non-negotiable	GUI displays malignancy probability	{Pass,Fail}	Pass	
Non-negotiable	GUI displays reference annotations	{Pass,Fail}	Pass	
Negotiable	Time to train model	(Days)	< 3 days	The less computer time it takes to train a model, the quicker we can evaluate and improve the algorithm.

To understand the table, it's important to understand what each of the requirements are.

Inter-rater reliability (Table 1, Row 1) allows us to compare the accuracy of our model to a baseline of human accuracy when classified by a qualified pathologist. This is important since whole-slide classification accuracy(percent of whole image classified correctly) doesn't provide enough context to understand the efficacy of our model.

Class dependent performance (Row 2) is a way for us to examine the model's overall performance through the lens of individual class performance. If our model gets 90% of its predictions correct, it's not an immediate indicator that it will perform with 90% accuracy. The model could be guessing 1 label 100% of the time, but if 90% of our data is that same label, it will look like it's doing a great job even though it got 0% accuracy on all the other labels. The F1 score allows us an in-depth view of how the model is performing on labels that happen to be a smaller part of the dataset.

Whole-slide image classification (Row 3) is a standard that is used by other organizations producing models. It's important we follow these standards to allow us to compare our progress.

The utilization of a semi-supervised model (Row 4) has become a necessity with Temple University's acquisition of funding. This new funding allows us access to more data, but with this new volume of data, not all of it has been annotated. A semi-supervised model allows us to combine the annotated datasets with the non-annotated datasets. A greater volume of data allows us to increase our models efficacy by increasing generalizability.

The GUI (Rows 5-7) will allow us to make more effective presentations, examine our results in a meaningful way, and give people without in-depth knowledge of our program, such as Dr. Picone, a way to examine our progress.

The time to train our model (Row 8) is important to allow us to prove our concept. In reality, this is a long term project and future models could take much longer to train. If we take much longer to train our model, we may not have enough time to present results by the end of the semester.

Design Constraints

Design constraints (Table 2) outline our limitations when conducting research and development. These limitations are partly self-imposed, such as adhering to code formatting standards to aid in readability, and partly naturally occurring as a result of limited availability of data. See the table below.

TABLE II

Design Constraints				
Priority	Constraint	Metric	Target/Limit	Justification
Non-negotiable	Programming language must be Python	{Pass,Fail}	Pass	This is the language the research lab maintains.
Non-negotiable	Data availability	Available training data (Bytes)	3.5 terabytes	Larger datasets correlate to more robust machine learning models.
Non-Negotiable	Adherence to ISIP standard for Python code formatting	{Pass,Fail}	Pass	[5]
Negotiable	Adherence to	{Pass,Fail}	Pass	[6]

PEP8 Python standard for code formatting

Writing our program using Python (Table 2, Row 1) is necessary because our code will be inherited by Dr. Picone's research group. The students involved in that group code primarily in Python, so if we were to use another language, while it could possibly return better results in the short term, it wouldn't be maintainable.

The amount of data available (Row 2) is a physical limitation. For each piece of data that has been annotated, a person has to manually go through and generate the annotations. Since this is an expensive and time consuming process, there is a limited amount of data (3.5 Terabytes) that we have access to train our model on.

The last two rows are standards constraints. Row 3 is a lab standard for Temple's Institute for Signal and Information Processing (ISIP). We use ISIP's tools for data processing tasks and format our code according to ISIP style guidelines[5] for consistency and interoperability between components. Row 4 mentions the industry standard for Python formatting (PEP8). Where ISIP guidelines don't take precedence, we defer to PEP8 standards[6] for the sake of maintaining and publishing our code for a wider readership. PEP8 standards contain recommendations on indentation, commenting, and conventions such as consistency among function return types (see excerpt below).

```
# Correct:

def foo(x):
    if x >= 0:
        return math.sqrt(x)
    else:
        return None

def bar(x):
    if x < 0:
        return None
    return math.sqrt(x)

# Wrong:

def foo(x):
    if x>=0:
        return math.sqrt(x)

def bar(x):
    if x<0:
        return
    return math.sqrt(x)
```

References

[1]S. C. Wetstein *et al.*, "Deep learning-based breast cancer grading and survival analysis on whole-slide histopathology images," *Scientific Reports*, vol. 12, no. 1, Sep. 2022, doi: https://doi.org/10.1038/s41598-022-19112-9.

[2]J. Zhu, M. Liu, and X. Li, "Progress on deep learning in digital pathology of breast cancer: a narrative review," *Gland Surgery*, vol. 11, no. 4, pp. 751–766, Apr. 2022, doi: https://doi.org/10.21037/gs-22-11.

[3a]M. Khened, A. Kori, H. Rajkumar, G. Krishnamurthi, and B. Srinivasan, "A generalized deep learning framework for whole-slide image segmentation and analysis," *Scientific Reports*, vol. 11, no. 1, Jun. 2021, doi: https://doi.org/10.1038/s41598-021-90444-8.

[3b]M. C. Comes *et al.*, "A deep learning model based on whole slide images to predict disease-free survival in cutaneous melanoma patients," *Scientific Reports*, vol. 12, no. 1, Nov. 2022, doi: https://doi.org/10.1038/s41598-022-24315-1.

[4a]C.-L. Chen *et al.*, "An annotation-free whole-slide training approach to pathological classification of lung cancer types using deep learning," *Nature Communications*, vol. 12, no. 1, Feb. 2021, doi: https://doi.org/10.1038/s41467-021-21467-y.

[4b]M. Abdoos and H. Azary, "A Semi-supervised method for tumor segmentation in mammogram images," *Journal of Medical Signals & Sensors*, vol. 10, no. 1, p. 12, 2020, doi: https://doi.org/10.4103/jmss.jmss_62_18.

[5]"Automatic Speech Recognition: Internet Accessible Speech Technology," *Piconepress.com*, 2024. https://isip.piconepress.com/projects/speech/software/tutorials/general/ (accessed Sep. 05, 2024).

[6]G. van Rossum, B. Warsaw, and N. Coghlan, "PEP 8 – Style Guide for Python Code," *peps.python.org*, Jul. 05, 2001. https://peps.python.org/pep-0008/