The Importance of UX for Machine Teaching

Martin Lindvall
Linköping University
Sectra AB
martin.lindvall@sectra.com

Jesper Molin
Sectra AB
jesper.molin@sectra.com

Jonas Löwgren
Linköping University
jonas.lowgren@liu.se

Abstract

In this position paper, we argue that UX designers should take an increasing responsibility for the process and tools used in the generation of training data for machine learning algorithms. We provide a number of annotated examples from our UX practice within the medical imaging domain to highlight different ways that a UX approach can help to select training data set, facilitate initial generation and ensure that the final systems become self-sufficient on training data, so that the systems can efficiently improve performance over time.

Introduction

One of the most important developments from a UX perspective in the machine learning (ML) domain is that algorithms today are able to improve their performance by adding more training data. This entails that processes and tools for the generation of training data can have a large impact on the success of ML projects. In many domains, the designers of the teaching systems do not themselves hold the expertise required to create training data, which means that human-centered design methods can play a key role in building systems that aid generation of training data.

This *teaching* aspect of building machine learning systems has recently received some attention. In Simard et al. (2017) the authors emphasize the role of the teacher and their interaction with data as a key factor for building machine learning systems at scale and argue for making *machine teaching* a discipline in its own. Cramer and Thom (2017), identifying and reflecting upon the impact of design decisions on ML outcomes, pose a series of questions relating to how the role of curators and annotators affect the ultimate end-user experience.

To emphasize the role of UX practice for generating training data we will highlight some key ideas illustrated by examples drawn from our work within a specific domain: medical imaging. We will describe four interactive systems that have been created and used within digital pathology, i.e. diagnosing and reviewing digital gigapixel-sized microscopic images of tissue samples such as biopsies and surgical specimens.

These examples together describe a typical two-step process we have used when designing new ML-based systems. First, we need to bootstrap a large enough dataset so that the algorithm used in the first version of the system performs sufficiently. Second, we need to ensure that the system can collect training data automatically when it is deployed, i.e, by receiving user corrections. This will make the system self-sufficient on training data, enabling a continuous improvement of the ML model. Our four annotated examples of this process are based on our own experience as UX-designers active in the medical imaging field. Two of the examples are prototypes and two are finished products that we have either designed ourselves or followed closely.

Efficient bootstrap teaching

An early step in the creation of an ML-based system, when no prior training data exist, is to somehow create an initial dataset. For pathology images this typically consist of drawing outlines over tissue regions and classifying these. Because it is a highly specialized domain this usually means engaging pathologists, who tend to be rather expensive teachers. Since it is important to make efficient use of these individuals and their knowledge, it seems sensible to align the design of the teaching environment with their experience.

Rapid interactive segmentation A well-known semi-automatic approach to assigning categories to visual regions is an *interactive segmentation tool*. The user of such tools typically use a paintbrush-style interaction to assign areas to given categories (called ”seeds”), and while doing so, areas similar to the one marked are also assigned to the same category (McGuinness and O’Connor 2010).

When we applied a human-centered design perspective to the construction of such a tool we gained valuable insights; for our initial prototype (see figure 1), the interaction was experienced as a *trading of control between human and machine*, where the human waits for the machine response after drawing an area. After a noticeable delay, the results are received and the human can make a correction, wait again, and then repeat the process. Typically, the user would be both intrigued and annoyed by the automatic assignment of the areas that were not specifically drawn over, sometimes resulting in long back-and-forth correction cycles without noticeable progress.

In a revised version, we aimed for *rapid fine-grained interaction* where spreading would be constructed as an in-

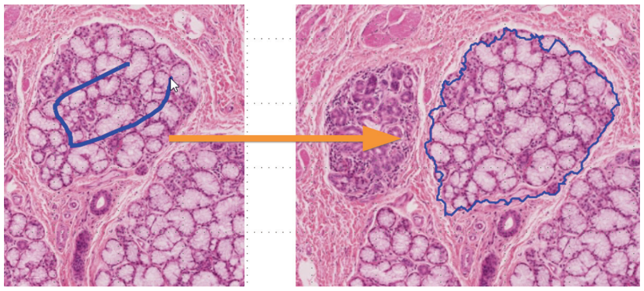


Figure 1: The initial version of our interactive segmentation tool. The user draws a path and waits for the response.

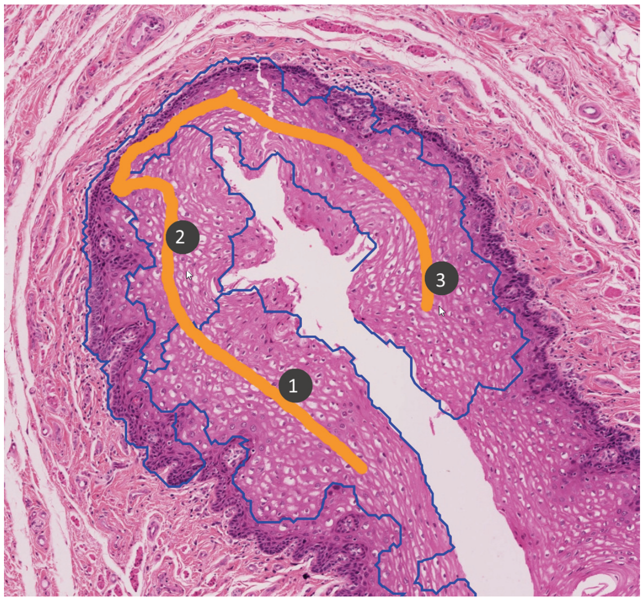


Figure 2: The revised segmentation tool. The user draws and results update in realtime, here shown for three points in time.

cremental and collaborative effort between user and system, rather than being computed slowly but accurately in every coarse-grained step (see figure 2). The tool was changed so that the threshold required for spreading increased with the distance from the original area. Additionally, we added pre-computations so that results of user input typically arrive in less than 40ms, a time during which the user is not blocked from giving more input. We postulate that the more fine-grained interaction lets the user gain an intuitive understanding of the underlying mechanism and its limitations by observing many predictions over time. Overall, we believe this real-time version of the tool to be novel and much preferable to using traded control, an effect we hope to validate in future work.

Creating intrinsic rewards Another approach to bootstrapping the initial training data set is to design a useful manual tool that generate training data as a side-effect. This approach is somewhat similar to the ESP game (von Ahn and Dabbish 2004), a two-player guessing game that created labeled training data as a side-effect of play. In the medical domain with professional users it would be inappropriate to deploy games to generate training data. Instead, the manual tool should aid the clinician in their decision making as a result of providing the tool with labels.

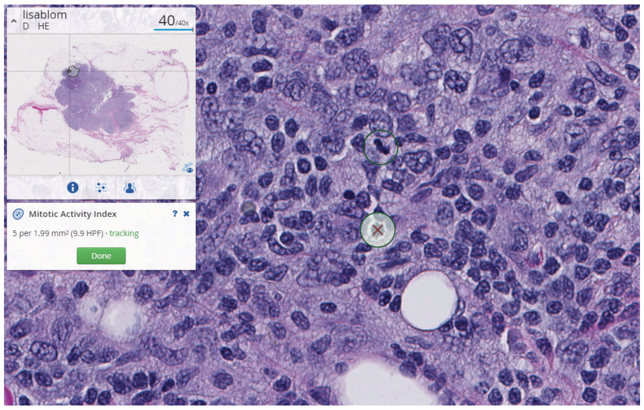


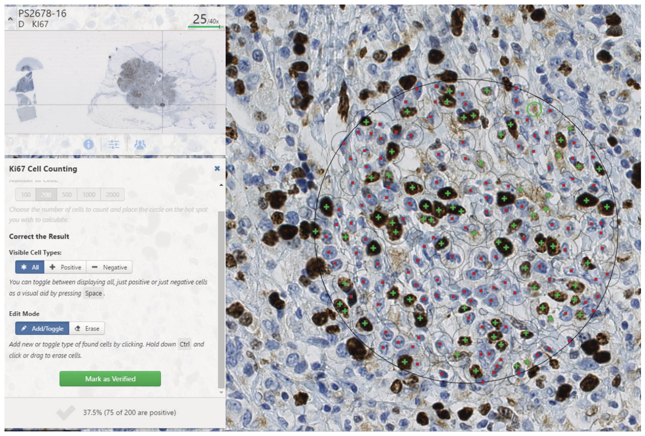
Figure 3: A manual tool to help pathologists to keep track of mitotic figures. This is used to generate training data for a future algorithm.

We have created one such tool to support pathologists in manual mitotic counting (figure 3). In this diagnostic task, the pathologist should go through ten fields of view in the highest magnification and count the number of mitotic figures. When performing this task it can be challenging to keep track of the number of mitotic figures as well as the number of fields of view. In the tool, this task is supported by keeping track of the reviewed area when navigating in the image. The user can also click on detected mitotic figures, which are then stored. Upon completion, the mitotic density can be derived using the number of stored mitotic figures and the total tracked area. Even though the tool works by rather simple means, it still turns out to be very useful for the pathologist. The side-effect is that every time a mitotic figure is clicked on, a training data example is generated. Additionally, the tracked areas that were reviewed but not clicked on can be used as examples of non-mitotic figures. By deploying this tool into a delivered product, it will generate a bootstrapping dataset of mitotic figures that can be used to train a ML-based detection system.

Designing for user corrections

Once ML systems are deployed, user corrections of the ML predictions can be used to generate additional training data. However, the UX designer needs to design specifically for this possibility. Our experience so far indicates that the most important factors for this type of design are to make sure that machine errors become apparent and that the class labels are chosen in such a way that they are easy to interact with.

This can be exemplified by ML systems used to quantify immunostains. Immunostaining is a technique used to chemically visualize protein expression in cells. A common protein used to quantify proliferation in tumor cells is KI-67.



Choose the number of cells to count and place the circle on the hot spot you wish to calculate.

You can toggle between displaying all, just positive or just negative cells as a visual aid by pressing Space.

Add new or toggle type of found cells by clicking. Hold down Ctrl and click or drag to erase cells.

37.5% (75 of 200 are positive)

Figure 4: An example of a symmetric input-output ML-system of cell counting system for Ki-67 stainings.

When using the KI-67 immunostain, the nucleus becomes brown if the cell is positive for this protein and appears blue from the background staining if it is not.

When designing an ML pipeline, two apparent choices of class labels for this problem exists: pixel labels and nuclei labels placed on the center of the nuclei. If pixel labels are used, pixels belonging to positive and negative nuclei can be visualized to the user as an overlay on top of the original image, occluding the nuclei. The user can then accept the result as is, or revert to manually counting the cells. If nuclei labels are used, the result can be visualized by placing glyphs on the center of each detected nucleus. This makes it is easier for the user to detect errors, since less ink is used to visualize the result and the original image becomes more visible. It also becomes easier to perform correction of misplaced markers since less precision is needed to click on markers than on pixels.The second approach was implemented as a product, and is shown in Figure 4.

This product illustrates the seminal principle of direct manipulation (Shneiderman 1982) that the result is presented in an *input-output symmetric way* where the user can directly manipulate the labeled data. By designing the system to allow for such direct manipulation and providing an *intrinsic reward* in terms of the actual nuclei count, user corrections can directly be used to retrain and improve the underlying machine learning model.

Another example of an ML direct manipulation interface is our patch gallery prototype shown in Figure 5, where the goal is to estimate the distribution of classes in an area. In this prototype, we generate a grid pattern over a user selected area and extract a small image patch for each point in the grid. We then feed each patch to an ML algorithm that classifies the patches into different categories, which is then shown in a sorted gallery. Each defined class in the trained model is shown as patches in the same gallery, and the user can then 1) click on a patch to see it in the main view to get a sense of its context in the tissue, and 2) change a label by either dragging the patch to the correct category or by clicking on the button or the corresponding shortcut key.

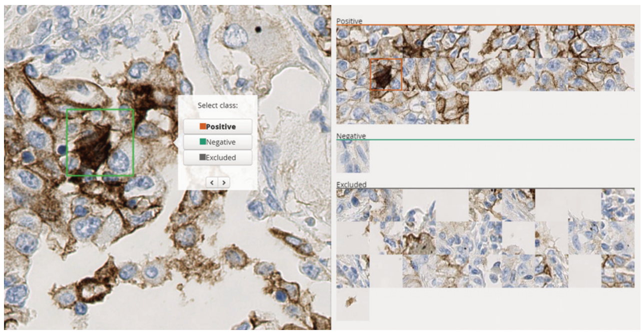


Figure 5: Patch gallery prototype, samples from the tissue is generated and classified by an ML algorithm into three classes.

Both these systems share the property with the mitotic counter in the previous section that the generated parameter can be derived from manual input only. If the nuclei detection algorithm failed to detect any nuclei, the user could still manually click on all the nuclei to calculate the KI-67 index. However, the amount of clicking would likely overwhelm the user. These user correction systems do not strictly need an ML component, but practical usability requires automated support with a certain level of prediction accuracy.

Another crucial factor when designing this type of user correction system is that the user correction accuracy needs to be higher than that of the ML component alone, in order for the generated training data to add value when retraining the ML model.

Discussion

In the design of these tools, we have paid special attention to ensuring that manual, unassisted, work-flows are preserved and as outlined in the previous section, compatible with the assisting tools. Furthermore, as the performance of models improve using the self-generating training data, we expect that our initial user interfaces need to be redesigned or augmented with interactions that are adapted to ML components with much higher performance. It is our ambition to design these so that the user can step through these ”levels of intelligence”, providing corrections and simultaneously teaching and verifying results at different levels, forming a verification staircase (Molin et al. 2016) as opposed to a steep cliff where the user has to validate all or nothing. We believe one possible way to achieve this could be to create our abstractions so that the user can always decompose a higher abstraction in terms of a lower one, an idea similar to the hierarchies of ecological interface design (Vicente and Rasmussen 1992) that we hope to explore in future work.

As pointed out by Dove et al. (2017), the interaction design community is still new to using ML as a design material. We are thus cautious to move beyond annotated examples towards more compact formulations of generative knowledge such as design patterns or principles at this stage. The examples described here may form the basis of transferable design knowledge when generalized to domain-independent visual reasoning tasks including an ML component. However, we need further studies in order to refine and validate this approach.

Conclusion

In this paper, we presented a number of annotated examples of how to manage training data generation from a UX perspective. The pattern emerging from these examples is that many of our ML projects become two-step processes. First, a training data set is created so that the initial trained ML model can reach an accuracy that will be acceptable to early-adopter users. Then by using different data collection methods designed into the first version of the product, it becomes self-sufficient in terms of training data. This allows the product to improve over time. As this process continues, the ML component will at some point become so good that the initial user interface might no longer be valid, and needs to be adapted to an ML component that performs on a higher level. How this is done is a promising area of further research. Our current plans involve a systematic explorative design effort of automation performance in the design of human-automation collaboration for visual reasoning tasks. Hopefully this will lead us toward the abstraction of genre-related generative design knowledge.

Looking at ML-based product development from the view of training data generation, we can learn that decisions made by the UX designer have an enormous impact on project success. Each step of training data generation needs to get the motivations right so that users are willing and able to provide corrections. The choice of what the training data set should consist of and thus what the ML model should predict is tightly connected to how the user interface should look, behave and be interacted with.

We challenge all UX professionals to take charge of the ML development cycle to make use of this powerful technology in the medical domain.

Biography

Martin Lindvall. Martin is an industrial Ph.D student at Linköping University exploring interaction design using machine learning as a material for creating effective ensembles of skilled medical practitioners and AI. Martin's background includes a M.Sc in Cognitive Science and ten years of experience designing and developing medical information systems as senior research engineer at Sectra.

Jesper Molin is research scientist and UX designer at Sectra exploring and designing ML-based tools used within clinical routine pathology. Jesper's background includes a M.Sc in *Applied physics and electrical engineering* and a now almost finished Ph.D in Human-Computer Interaction from Chalmers University of Technology.

Jonas Löwgren is professor of interaction and information design at Linköping University, Sweden. His expertise includes collaborative media, interactive visualization and the design theory of the digital materials.

Acknowledgments

This work was partially supported by the Wallenberg Autonomous Systems and Software Program (WASP).

References

Cramer, H., and Thom, J. 2017. Not-So-Autonomous , Very Human Decisions in Machine Learning : Questions when Designing for ML. Technical Report SS-17-04, The AAAI 2017 Spring Symposium on Designing the User Experience of Machine Learning Systems, Stanford Univ.

Dove, G.; Halskov, K.; Forlizzi, J.; and Zimmerman, J. 2017. UX Design Innovation: Challenges for Working with Machine Learning as a Design Material. In *CHI '17: Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, 278–288. ACM.

McGuinness, K., and O'Connor, N. E. 2010. A comparative evaluation of interactive segmentation algorithms. *Pattern Recognition* 43(2):434–444.

Molin, J.; Woźniak, P. W.; Lundström, C.; Treanor, D.; and Fjeld, M. 2016. Understanding design for automated image analysis in digital pathology. In *Proceedings of the 9th Nordic Conference on Human-Computer Interaction*, 58. ACM.

Shneiderman, B. 1982. The future of interactive systems and the emergence of direct manipulation. *Behaviour & Information Technology* 1(3):237–256.

Simard, P.; Amershi, S.; Chickering, M.; Edelman Pelton, A.; Ghorashi, S.; Meek, C.; Ramos, G.; Suh, J.; Verwey, J.; Wang, M.; and Wernsing, J. 2017. Machine Teaching: A New Paradigm for Building Machine Learning Systems. Technical Report MSR-TR-2017-26, Microsoft Research.

Vicente, K. J., and Rasmussen, J. 1992. Ecological interface design: theoretical foundations. *IEEE Transactions on Systems, Man, and Cybernetics* 22(4):589–606.

von Ahn, L., and Dabbish, L. 2004. Labeling images with a computer game. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '04, 319–326. New York, NY, USA: ACM.