The importance of UX for machine teaching

Martin Lindvall, Jesper Molin
Lindvall: Linköping University, Linköping, Sweden
Lindvall & Molin: Sectra AB, Linköping – Sweden
martin.svenson@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 make sure 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 emphasis 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 design decisions' impact 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 specimen.

These examples together describe a typical two step process we've 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 becomes sufficient. 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 for continuous improvement of the ML-algorithm. 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-algorithm, when no prior training data exist, is to somehow create an initial dataset. For pathology images this typically consist of drawing outlines over regions and classifying these. Because it is a highly specialized domain, this usually means making use of pathologists, which typically are rather expensive teachers. Since it's important to make efficient use of these individuals and their knowledge, it seems sound to design the teaching environment with a mind toward their user experience.

Rapid interactive segmentation A well-known semi-automatic approach to assigning categories to visual regions is an *interactive segmentation tool* that uses handcrafted features to initially oversegment the image (into areas called superpixels). The user of the tool then uses a paintbrush-style interaction to assign areas to given categories (called "seeds"), and while doing so, areas similar to the one marked is also assigned 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 a few 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 some area. After a noticeable delay, the results were received and the human can make one correction, wait again, and then repeat the process. Typically, the user would be both intrigued and annoyed by the automatic assignment of

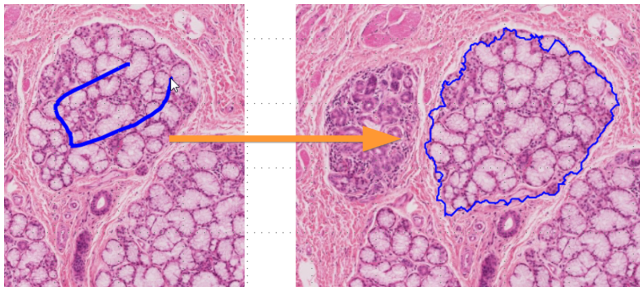

Figure 1: The initial version of our interactive segmentation tool. The user draws a path and waits for the response.

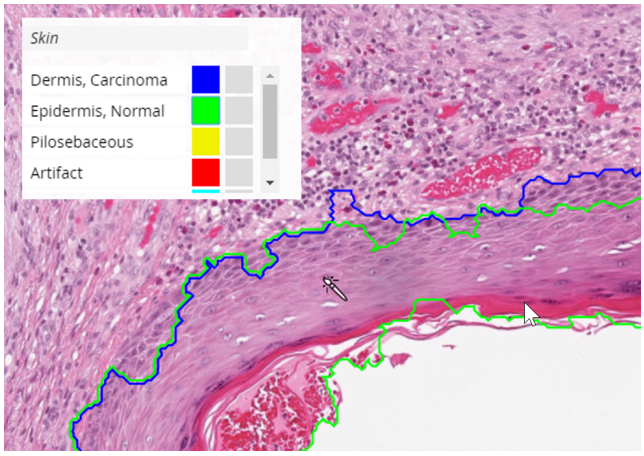

Figure 2: The revised segmentation tool. The user draws and results update instantly as the movement progress.

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 a rapid fine-grained interchange over accurate spreading (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 precomputations 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 real-time interaction lets the user gain an intuitive understanding of the underlying mechanism and its limitations by observing many predictions. 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.

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 similar to, the ESP game (von Ahn and Dabbish 2004), a two-player guessing game that create labeled training data as a side-effect. In the medical domain with professional users, it would be inappropriate to deploy games to generate training data. Instead, a manual tool should be useful for clinical

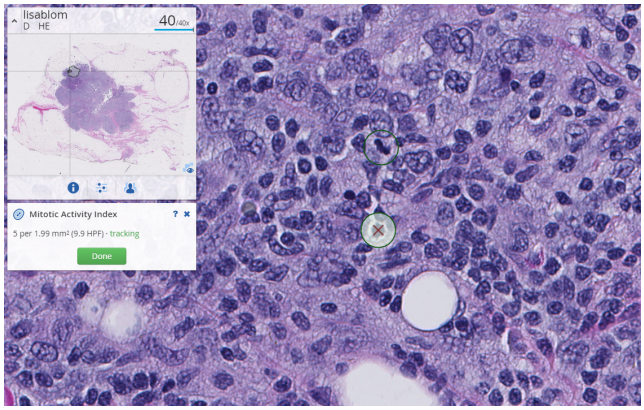

Figure 3: A manual tool to help pathologists to keep track of mitotic figures. This used to generate training data for a future algorithm.

decision making.

We have created one such tool to aid pathologists with manual mitotic counting. In this diagnostic task, the pathologist should go through 10 field of views in the highest magnification and count the number of mitotic figures. When performing this task, it can be challenging to both keep track of the number of mitotic figures as well as the number of fields of views. 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 is then stored. This means at the end the mitotic density can be derived using the number of stored mitotic figures and the total tracked area. A tool like this is very useful for the pathologist as a manual tool. The side-effect works so that every time a mitotic figure is clicked on, a training data example is generated. Additionally, the tracked area that is not click 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 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 specifically design for this possibility. We think that the most important factors for this type of design is 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. When using the KI-67 immunostain, the nuclei becomes brown if the cell is positive for this protein and appear blue from the background staining if they dont́.

When designing a 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

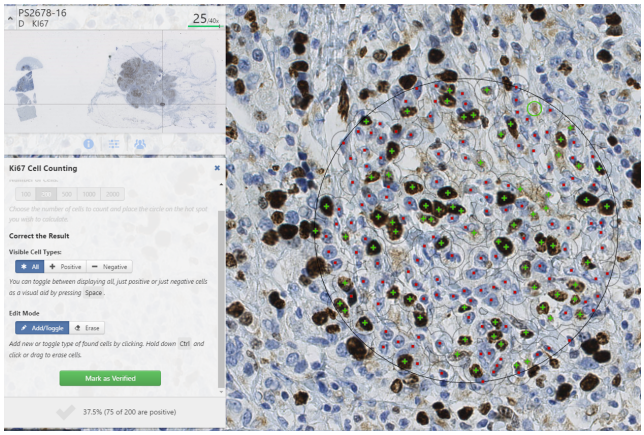

Figure 4: An example of a symmetric input-output ML-system of cell counting system for Ki-67 stainings.

used, pixels belonging to positive and negative nuclei can be visualized to the user as an overlay on top of the original image. The user can then accept the result as is, or fall back to manually counting the cells. If nuclei labels are used, the result can be visualized by placing glyphs on center of each detected nuclei. 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.

In this product, we can say that the result is presented in an *input-output symmetric way*, where the user can directly manipulate the labeled data. By designing the interaction using this type of direct manipulation, user corrections can directly be used to retrain the underlying machine learning model.

Another example of a ML direct manipulation interface is given by our patch gallery prototype shown in Figure 5. 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 a ML-algorithm that classifies the patches into different categories, which is then shown in a sorted gallery. Each defined class in the ML-algorithm shows 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.

Both these systems share the property with the mitotic counter in the previous section in 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-algorithm, but they would become unusable without or with a ML-algorithm below a certain prediction accuracy.

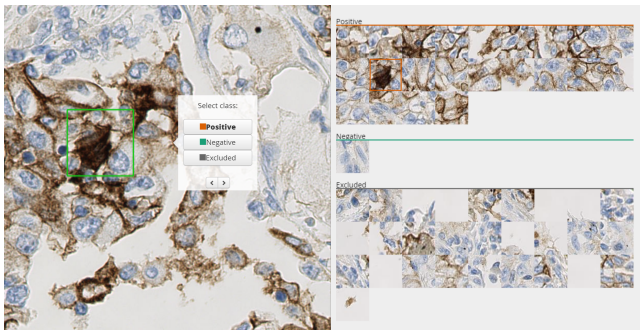

Figure 5: Patch gallery prototype, samples from the tissue is generated and classified by an ML-algorithm to sort into three classes. The review each sample in its context to the left, and change the classification if needed.

Another crucial factor when designing this type of user correction system, is that the user correction accuracy needs to be higher than the ML-algorithm alone. Otherwise the generated training data will not add any value when retraining the ML-algorithm.

Discussion

In the design of these tools, we've paid special attention to ensuring that manual 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 to be designing new interactions that acts with a higher degree of intelligence. 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.

For instance, in the context of the bootstrap tool the next step might be to allow the user to paint without specifying category, gradually seeing the results of the machine classification, but still being able to fall back to rapid corrections where necessary. As the predictions further improve, we might reach the scenario where the entire image is segmented from the start and the user has to only use the lower levels to correct limited parts.

As a topic of further exploration, we've noted that training data is very dependent on the problem setting for the ML-project, "learning the right thing" according to Yang (2017). In other words, determining what should be input and what should be predictions is a very early design decision. Thus, we believe a focus on the training data can act as *foci* for facilitating very important design discussions at the appropriate stage of development, i.e, before annotations begin. In our experience, the notions of how the final interaction will provide value co-evolve with discussions on determining an annotation protocol (which is in fact a discussion about problem setting) - forming a *wicked problem* that can be explored using design methods.

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 a two-step process. First, an initial training data set is created so that the initial algorithm can reach an accuracy that will accepted by early adopter users. Then by using different data collection methods designed into to the first version of the product, it becomes self-sufficient on training data. This allows the product/system to improve the algorithms performance over time. If this is allowed to continue, the algorithm will at some point become so good that the initial user interface might no longer be valid, and needs adapted to an algorithm that has much higher performance. How this done, is an interesting future area of research.

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 for success. Each step of training data generation needs get the motivations right so that users are willing to provide corrections. The choice of what the training data set should consist of have an impact on how the user interface should look like.

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

Biography

Martin Lindvall. Martin is an industrial Ph.D student at Linköping University exploring interaction design for 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.

Symposium presentation format

We believe the above material would be most suited to be presented and somewhat elaborated in a twenty minute presentation, including brief prerecorded demonstrations of relevant interactions.

A poster session where we are able to bring tablets to enable us to show particular details to interested parties is a viable alternative.

References

Cramer, H., and Thom, J. 2017. Not-So-Autonomous , Very Human Decisions in Machine Learning : Questions when Designing for ML. *The AAAI 2017 Spring Symposium on Designing the User Experience of Machine Learning Systems Technical Report SS-17-04* 412–414.

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.

Simard, P. Y.; Amershi, S.; Chickering, D. M.; Pelton, A. E.; 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.

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.

Yang, Q. 2017. The Role of Design in Creating Machine-Learning-Enhanced User Experience. *The AAAI 2017 Spring Symposium on Designing the User Experience of Machine Learning Systems Technical Report SS-17-04* 406–411.

Acknowledgments

This work was supported in part through a grant by the Wallenberg Autonomous Systems and Software Program.