Research Article

Sven Schultze, Uwe Gruenefeld*, and Susanne Boll

Demystifying Deep Learning: Developing and Evaluating a User-Centered Learning App for Beginners to Gain Practical Experience

https://doi.org/10.1515/icom-2020-0023

Abstract: Deep Learning has revolutionized Machine Learning, enhancing our ability to solve complex computational problems. From image classification to speech recognition, the technology can be beneficial in a broad range of scenarios. However, the barrier to entry is quite high, especially when programming skills are missing. In this paper, we present the development of a learning application that is easy to use, yet powerful enough to solve practical Deep Learning problems. We followed the human-centered design approach and conducted a technical evaluation to identify solvable classification problems. Afterwards, we conducted an online user evaluation to gain insights on users' experience with the app, and to understand positive as well as negative aspects of our implemented concept. Our results show that participants liked using the app and found it useful, especially for beginners. Nonetheless, future iterations of the learning app should step-wise include more features to support advancing users.

Keywords: Deep Learning, Machine Learning, explainable, education

1 Introduction

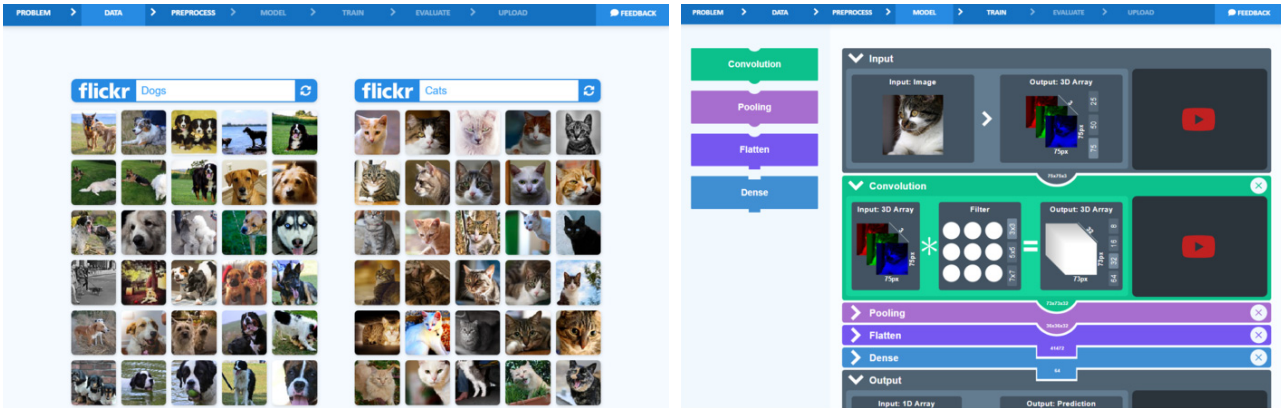
In recent years, Deep Learning (DL) has received considerable attention, with many beginners interested in learning the technology. Continually decreasing computational costs have made the technology practical and applicable to real-world problems [21, 13, 10, 9]. Nowadays, Deep Learning empowers users to address a broad range of problems, previously not considered practically solvable, and often in a more effective manner than other Machine Learning (ML) approaches [3, 12]. This potential has sparked interest, resulting in non-experts willing to learn the technology. However, understanding the concepts that constitute Deep Learning can be challenging.

In general, there are two different motivations to deal with Deep Learning: 1) to be able to develop deep-learning-based systems, and 2) to understand the decisions of these systems in everyday life [16]. If the latter is the motivation, people tend to have no technical background and are overwhelmed when dealing with theoretical foundations of Artificial Neural Networks (ANN), which are the very essence of Deep Learning. The connection between cause and effect is especially difficult to grasp when dealing with ANN [19]. Fortunately, understanding the theory in complete detail is not required to understand decision-making by intelligent systems. Moreover, applying Deep Learning workflows and solving practical problems relies heavily on experimentation. Thus, experience and intuition are vital for mastering Deep Learning. However, gaining this experience is challenging for beginners. It is time-consuming and can be frustrating because direct feedback is often missing. Furthermore, the barrier of entry is quite high because it requires solid technical skills, such as knowing a programming language.

We believe that a visualization-based learning application could lower the barrier of entry, empowering beginners to understand Deep Learning without first mastering a programming language. Previous work has unveiled the potential of visualization-based approaches [19, 28]. For example, visual programming languages can help one to understand the concepts of programming quickly [4, 26]. However, while visualization-based learning approaches have proven useful in many different scenarios, it remains unclear whether Deep Learning beginners can benefit from them as well.

In this paper, we followed the human-centered design process to develop an interactive visualization-based learning application that aims to support Deep Learning beginners during their first steps. Our goal is to develop an application, powerful enough to solve practical problems. To do so, we analyzed the existing work to under-

***Corresponding author: Uwe Gruenefeld,** University of Duisburg-Essen, Essen, Germany, e-mail: uwe.gruenefeld@uni-due.de
Sven Schultze, Susanne Boll, University of Oldenburg, Oldenburg, Germany, e-mails: sven.schultze@uol.de, susanne.boll@uol.de



(a) Flickr integration for training and test datasets used in our learning app.

(b) Visual programming (puzzle-based) to create the model architecture.

Figure 1: Implementation of our web application to demystify Deep Learning.

stand the state-of-the-art. We interviewed Machine Learning experts to find a suitable scenario for beginners and conducted focus groups to identify the application scope. Then, we developed a low-fidelity prototype, did a cognitive walk-through, and implemented our application. We finished with a technical evaluation to identify solvable classification problems and a user evaluation to understand users' experience when using the app. Our work contributes insights into the development of a user-centered learning application for Deep Learning that applies the complete human-centered design process [11] and the app itself hosted to invite researchers, teachers, and learners to try it out.[1]

2 Related Work

After a literature review, we identified two different categories of previous research as relevant to our work. The first category consists of research that helps user to understand the decision making process of complex systems based on artificial neural networks, and the second category includes related work on educational applications that explain how Deep Learning works.

2.1 Understanding Decisions of ANN-based Systems

ANNs often are complex networks consisting of many artificial neurons and connections, making it incredibly challenging to assess their behavior. Even experts cannot always predict how ANNs react in every possible situation, which is hugely problematic (e. g., in safety-critical contexts [2]). Hence, the question arises of how users without a technical background can trust them? To address this issue, the field of neural network interpretability has formed, following two objectives: 1) finding out what features ANNs learn to recognize (feature visualization), and 2) what kind of data is crucial in their decision process (feature attribution) [16].

Previous work proposes different visual analytic tools to support model explanation, interpretation, and debugging [7]. For example, Yosinski et al. suggest two different tools [28], one to demonstrate activation of neurons using the user's webcam as input, and another one to see how the layers of the network learn certain features. One more tool that explains what convolutional neural networks learn internally is ShapeShop [6]. It is an interactive experimentation environment in which users can create a custom dataset from simple shapes (circles, squares, and triangles), train a model, and view the experiment results. Analytic tools often use a method called activation maximization that focuses on input that highly activates a specific neuron [17]. Similar approaches are activation aggregation and neuron-influence aggregation [8]. However, while these tools can help users to understand the trained model better, they require fundamental knowledge about neural networks, making them better suited for more advanced users.

1 Our learning application. http://ai.uol.de/app

2.2 Educational Applications for Deep Learning

Interactive visualizations can significantly increase beginners' understanding of program behavior [1]. They can be integrated into learning experiences with explorable explanations, for example [23]. To support beginners, we will use explorable explanations in our application as well.

A few educational applications that help beginners understand Deep Learning exist. One example is Teachable Machine[2] by Google Creative Labs. It is a simple application that collects user-labeled images from the webcam, trains a neural network in the background, classifies the images in real-time, and visualizes the results (see Figure 2a). This teaches users a procedural intuition about deep-learning-based classifiers, allowing users to recognize them in everyday life, and understand how they are created. Another example is Tensorflow Playground[3] by Smilkov et al. [19], which allows users to experiment with neural networks via direct manipulation, allowing them to build intuition about the relationships between artificial neurons, loss functions, learning rates, and other concepts of Machine Learning (see Figure 2b). Nevertheless, while both applications address beginners, they do not explain the Deep Learning workflow. However, to apply learned concepts to practical problems, users must not only understand how neural networks work, but must also understand the workflow from finding a dataset to training and evaluating a model.

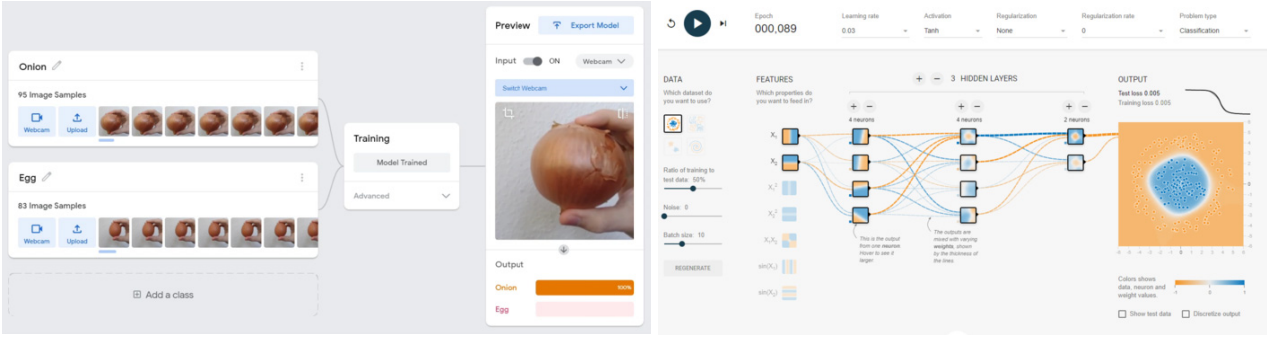
3 Method

In this paper, we develop a visualization-based learning application that allows users to create custom datasets with little effort, assemble neural network architectures with a puzzle-based interaction, train their models on a remote server, and evaluate it using uploaded images. To develop this application, we followed the human-centered design approach [11]. Hence, we started by defining the context of use, in which we focused on users with no Deep Learning or programming experience. To make the learning application available on a broad range of devices and following previous work [19], we developed it using standard web technologies. We interviewed experts to choose a learning scenario that is well-suited for novice users. Then, we conducted focus groups to understand what matters to beginners. Next, we created a low-fidelity (Lo-Fi) mock-up prototype and evaluated it with the thinking-aloud method [14], which allowed us to gain insights into the participants' cognitive processes during a walk-through of the interface. These insights helped us to eliminate misleading design elements and identify possible improvements. After that, we implemented a high-fidelity prototype in the form of a web application. Last, we conducted a technical evaluation to benchmark the performance and feasibility of our implemented learning application and a user evaluation to understand users' experience when using the app.

4 Developing the Learning Application

In the following, we describe all steps we undertook to develop our application. These steps are based on methods taken from the human-centered design approach [11].

2 Teachable Machine. https://teachablemachine.withgoogle.com
3 Tensorflow Playground. https://playground.tensorflow.org



(a) Teachable Machine (b) TensorFlow Playground

Figure 2: Related educational applications for Deep Learning.

4.1 Selecting a Learning Scenario with Expert Interviews

We conducted unstructured interviews with three experts that offered recurring workshops/tutorials on Machine Learning at scientific conferences in HCI. All interviewed experts agreed that classification tasks, more specifically image classification tasks (e. g., distinguishing between cats and dogs) are a good starting point for beginners. These tasks are easy to understand and provide many opportunities for interaction. Furthermore, image hosting providers such as Flickr provide free access to large datasets required as input for many Deep Learning algorithms. Hence, we focus on image classification tasks in our learning tool. Additionally, experts highlighted that beginners need quick feedback, empowering them to iterate over their solutions quickly and gain more practical experience in a shorter time.

For the classification, we decided to use convolutional neural networks (CNN) (a specific type of ANN) because they are frequently used for image processing. These networks consist of different building blocks (referred to as layers) that can be connected in various ways. We thought that limiting the available blocks could reduce the complexity of the task.

4.2 Defining the Application Scope with Focus Groups

After selecting a learning scenario suitable for beginners, we conducted focus groups to define the applications' scope. To have uninfluenced opinions from Deep Learning beginners, we carried out two focus groups. In the first group, two intermediates and one expert participated (all three had more than one year of experience). In contrast, the second group was composed of three beginners (with less than six months of experience). The six participants (2 female) were between 21 and 39 years old (mean: 26.0, standard deviation: 6.6).

We discussed the degree to which the app should guide users, collecting the result as Likert-items (ranging from 1=no guidance, maximum self-experimentation to 10=maximum guidance, no self-experimentation). The answers to this question were dependent on the participants' experience level. The beginners opted for more guidance (7, 6, 7). They agreed that they had problems finding an entry point to the field of Deep Learning because of its vastness. The experts chose lower guidance (4, 4, 3). They explained that experimentation is a big part of the Deep Learning workflow required to gain practical experience. Hence, we offer guidance in the beginning, but later switch to self-experimentation.

Then, participants debated which phases of the Deep Learning workflow are most important. For this, they ordered the following steps by priority: define the problem, gather data, build model, train, evaluate, and deploy. All participants agreed that the most critical phase is model building because it is the core of the workflow. In the second place, they ranked data gathering to build intuition about compositions and dimensions of datasets. This process takes time and should be dealt with thoroughly in the application. Participants also suggested to discard to deployment as is does not actively contribute to understanding Deep Learning. Next, participants decided on a basic set of layer types, resulting in the following layers: convolutional, pooling, flatten, and dense. Furthermore, both focus groups' participants suggested not to deal with optimizers and activation functions.

4.3 Designing a Lo-Fi Prototype

Together with an experienced UX Designer, we created a horizontal Lo-Fi prototype in Adobe XD.[4] The prototype is shown in Figure 3. Each step of the Deep Learning workflow is represented in a dedicated view. The views can be switched in the top navigation bar of our application. In the following, we explain all views in the order they appear in the navigation:

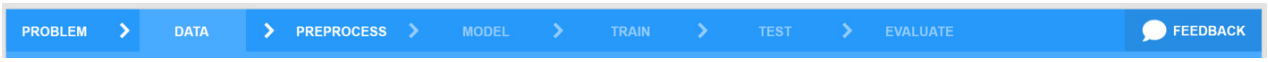
Problem

The first view the user sees is the problem view. Here, the classification problem is illustrated using an explorable explanation [23], where two images are manually classified into cats and dogs (see Figure 3b).

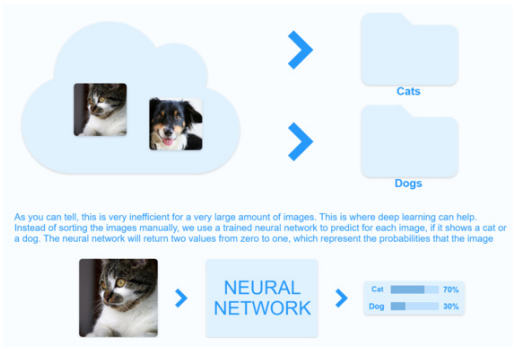
Data

The data view is split into two halves (see the prototype view in the center of Figure 3c). Each half contains a text field on top with a grid view of images underneath. These images can be loaded from the Flickr API by typing search terms into the text field. Each half represents one of the two classes that the classifier needs to distinguish. For the images, we decided to use Flickr because its API enables highly customizable image searches and already scaled-down resolutions, which is useful for reducing loading times.

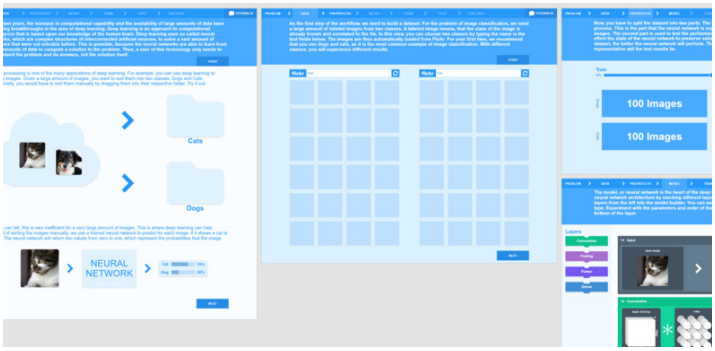
4 Adobe XD. https://adobe.com/xd, last retrieved December 15, 2020.



(a) Navigation bar to switch between views.



(b) Interactive introduction of image classification problems.



(c) Adobe XD screenshot that shows some of the designed view prototypes.

Figure 3: Lo-Fi prototype development of our application in Adobe XD.

Preprocess

In this view, users can interact with a slider to change the proportional composition of the train and test subsets of the dataset. The slider range is reduced by ten percent on both sides to ensure both datasets exist.

Model

The model view is where the network architecture is defined. It consists of a layer panel on the left, which allows the user to add different layer types to the model panel on the right (see Figure 1b for the already implemented view). In the model panel, the newly added layer shows an interactive visualization, illustrating the functionality of the layer. The user can interact with a few parameters, depending on the layer type. For example, the convolution layer allows the user to interact with the size of the convolution kernel, as well as the depth of the output.

Initially, only input and output layers are present in the model panel, and neither layer can be rearranged or deleted. Since layers can be arranged only in specific orders, they are represented as puzzle elements that follow the same constraints. If the user does not follow these constraints, the layers are visualized as disconnected, and an error message suggests fitting layers. If the model is valid, the user can proceed to the next view.

Train

In this view, users can use a set of controls to reset, pause, or start the training. Additionally, a slider allows users to set the number of epochs. When the training starts, the slider automatically moves to the left, as the remaining epochs decrease. A line chart displays the training progress by showing the accuracy for every trained epoch.

Evaluate

This view displays all images from the test subset of the data set with their prediction. It orders the pictures into four columns representing the true and false predictions for both classes and shows their respective class probabilities.

Upload

The upload view allows users to upload their images to predict them with a previously trained model. For these images, the view uses the same visualization technique for the predictions as the evaluate view. The images can either be dropped into a designated upload area or opened via the standard file dialog of the operating system.

4.4 Thinking Aloud to Unveil Design Flaws

After prototyping, we applied the thinking-aloud method, in which participants walk through the learning application while verbally expressing their thoughts to discover design flaws in our clickable Lo-Fi prototype [14]. The procedure went as follows: we explained the rules of the thinking-aloud method and informed them that we would record their thoughts and actions. Then we asked the participants to click through the prototype step by step and

to pretend to perform the image classification task. In the end, we discussed possible solutions for the problems the participants encountered.

We recruited three male participants from the age of 27 to 39 years (mean: 31.3, standard deviation: 6.7). All the participants had more than three years of experience in the field of human-computer interaction, while their level of expertise in Deep Learning varied widely.

Overall, we identified several user experience problems in the interface. For example, in some cases, participants mentioned that positive feedback is missing or that it would be great to present helping information only once but allow them to get it back with a simple button click. These comments are in line with the eight golden rules of interface design by Shneiderman [22], which state that users should receive informative feedback for their actions, and Nielsen's heuristics for user interface design [18], which say that the interface should be minimalistic. Furthermore, we unveiled some technical flaws in our design concept in terms of correctness and practicality.

4.5 Implementation of the Learning Platform

Following the thinking-aloud method, we implemented a high-fidelity prototype[5] in the form of a web application, incorporating all given feedback. We implemented the client-side of the application using the progressive Javascript framework Vue.js,[6] and developed our server in Python using Tensorflow.[7] Client-server communication is based on sockets.

Users can create custom datasets with little effort. To achieve this, we use Flickr's REST-API,[8] enabling users to load labeled images from the image hosting service via keyword-based search (see Figure 1a). Each class consists of 200 pictures with a resolution of 75x75 pixels (used resolution can be changed in model view), so 400 images in total. Additionally, users can replace specific images with new ones.

Furthermore, we implemented a purely client-based version using Tensorflow.js,[9] allowing users to train networks using their hardware.

5 Our learning application. http://ai.uol.de/app

6 Vue.js. https://vuejs.org, last retrieved December 15, 2020.

7 Tensorflow. https://www.tensorflow.org, last retrieved December 15, 2020.

8 Flickr API. https://www.flickr.com/services/api/request.rest.html, last retrieved December 15, 2020.

9 Tensorflow.js. https://www.tensorflow.org/js, last retrieved December 15, 2020.

5 Technical Evaluation

For the technical evaluation, we asked ourselves two questions: 1) what accuracies are possible with our learning application, and 2) how much slower is training on the client vs. on the server. For example, if one would deploy our web app for teaching a class on Deep Learning but without access to a server with GPU support.

5.1 Accuracy of Trained Models

First, we evaluated what accuracy is possible for known image classification problems in our application. Accuracy is important for user experience because a well-performing model can increase the users' satisfaction. We expected a lower model performance since we limited the dataset size and resolution, to achieve lower response times and allow quicker experimentation. Furthermore, we examine less-complex models trained in a reasonable amount of time. To evaluate the performance impact of these limitations, we tested a model architecture with four different datasets (see Figure 4).

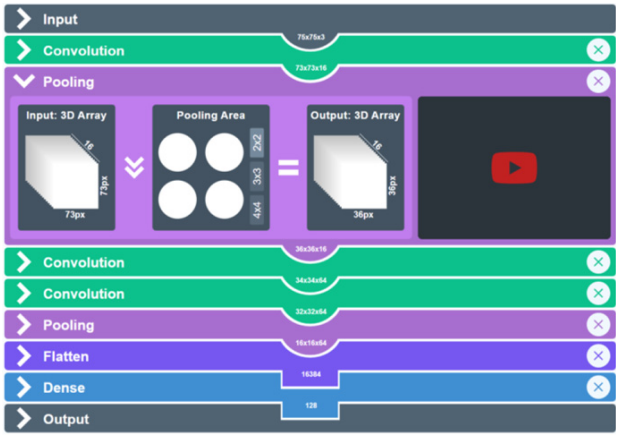


Figure 4: Model architecture used to evaluate the different example datasets.

We trained the model for 20 epochs on each dataset with a 75 %/25 % train/test split (see Table 1). While for simple problems like 'Pigs & Horses,' or 'Landscapes & Paintings,' the model achieves higher accuracy, the accuracy for the default problem 'Dogs & Cats' is below 70 percent. We think this is due to the variety of dog and cat breeds. Since the dataset is small, the test subset of the dataset likely contains images of breeds that the model

Table 1: Resulting accuracy for different example datasets.

First Class	Second Class	Accuracy
Dogs	Cats	65 %
Golden Retriever	American Shorthair	80 %
Pigs	Horses	87 %
Houses	Cars	90 %
Landscapes	Paintings	95 %

had never encountered during training. Hence, we recommend training more specific classes. For example, 'Golden Retriever & American Shorthair' achieves better results.

Previous analyses of Deep Learning models deliver more insights into this problem. They show that the primary features learned by the first convolutional layers in the models often resemble frequency- and orientation-selective kernels or color blobs [13, 27]. The recorded accuracy may be a result of this since images of pigs and landscapes both feature distinct color values and shapes compared to their respective counterparts in the datasets. For example, pigs are mostly pink, while horses are often brown. Similarly, landscapes mostly feature sharp and horizontal gradients, while paintings are colorful and have smaller, curved gradients in various orientations. The images of dogs and cats, however, are quite similar in this regard. Thus, this classification task might require more complex models or larger datasets, both of which are limited in our application to increase beginner friendliness.

5.2 Clients vs. Server for Training

For our application, we implemented two different training methods: 1) a Python backend based on Tensorflow, and 2) a frontend trainer based on TensorFlow.js. We experimented with a frontend-based approach because it allows the web application to scale better if it performs well enough on users' hardware.

To measure the performance of both approaches, we sequentially ran both configurations, including the server, on the same machine. Our machine consisted of an Intel Xeon E5-2678 CPU and an NVIDIA Quadro P5000 GPU. For the training, we set up a simple model in our application and trained it with a total of 200 images of two classes in 20 epochs in both configurations. We measured the time between the start and end of the training (see Figure 5).

As Figure 5 shows, the backend trainer finished more than three times faster than the TensorFlow.js frontend trainer.

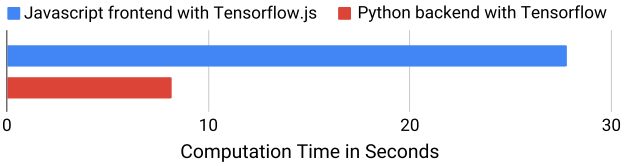


Figure 5: Computation time for different training architectures.

5.3 Discussion

In the following, we discuss the reached accuracy of the trained models and reflect on the different architectures (client vs. server) for training.

Model Training Accuracy

An important factor is the performance of the trained models. If users should stay motivated to use the application, a certain level of accomplishment is required. Unfortunately, the minimalistic design compromises performance in several ways. For example, the limited size of images in datasets enables users to interact with every picture but introduces limitations. We evaluated how these limitations affect performance in more difficult classification problems, for example, with the groups dogs and cats. Here, it is likely that the many different breeds of dogs and cats make it challenging for the model to learn the differentiating features of the two classes. However, in simpler classification problems, the results were much better. Thus, we recommend starting with them.

Training Architectures

We implemented two different training methods: a Python backend and a frontend trainer based on Tensorflow and Tensorflow.js respectively. From our results, we saw that the backend trainer finished more than three times faster than the frontend method. The reason is that TensorFlow for Python can access the GPU of the System directly. At the same time, the javascript library is limited by the API WebGL, which is intended for rendering graphics and not Deep Learning. We argue that the loss in power is quite significant. Furthermore, the library TensorFlow.js uses all available computational resources, often resulting in system lags. Thus, we recommend using the Python backend, and reverting to the frontend when the backend is not available (e. g., overload or connection issues). Moreover, new web APIs are required to make frontend-based solutions such as progressive web apps suitable for Machine Learning.

6 User Evaluation

To understand the user experience that Deep Learning beginners have with our learning app and gather further insights on factors benefiting or harming this experience, we conducted an online user evaluation. We had originally planned to conduct a lab study. However, to not put users at any risk during the ongoing pandemic, we decided against the in-person study.

6.1 Procedure

Participants became aware of our user evaluation through online advertisements. The advertisement text briefly pitched our app to generate interest, informed about our user evaluation, and contained a link to our web app. After clicking the link, participants could use the web app and click their way through all provided Deep Learning workflow steps (see Figure 3a), go back to previous steps, and interact with the app as they liked. We did not add any additional guidance for the user evaluation to ensure participants would get the out-of-the-box experience. When participants reached the test or evaluation step (see Figure 3a), a modal dialog box appeared after 30 seconds, asking participants to fill out a questionnaire in Google Forms. Additionally, participants could click the feedback button on the menu at any time to open the same dialog box with the link to the questionnaire. The questionnaire started with a short description explaining the purpose of the evaluation. After that, the user experience questionnaire (UEQ) followed [15]. Then, we asked three open questions about positive and negative aspects and potential improvements. They were followed by two 7-point Likert items to rate the app. Moreover, in the end, we ask three demographic questions.

6.2 Participants

We recruited 12 volunteer participants (7 males, 4 females, 1 preferred not to say), aged between 23 and 35 years (M=26.67, SD=3.34). Participants were recruited through the university online whiteboard and social media advertisements (Twitter, Facebook, and Reddit). Participants received no compensation for their time, which was communicated in the advertisement. We asked the participants to rate their level of experience with Machine Learning (ML), providing three options: two had no experience with ML (16.67 %), five stated they have less than six months of experience (41.67 %), and five said they have more than six months of experience with ML (41.67 %). Nonetheless, no participant had more than a year of experience with Machine Learning and only two of the participants (16.67 %) had some experience with Deep Learning already.

6.3 Results

In the following, we present the results of the conducted online questionnaire that we asked participants after trying out our web app.

User Experience Questionnaire

Looking at the results of the User Experience Questionnaire [15], the mean overall quality of the user experience is rated with 1.76. The pragmatic quality is rated with 1.81, and, in contrast, the hedonic quality is rated less with 1.625 (see Figure 6).

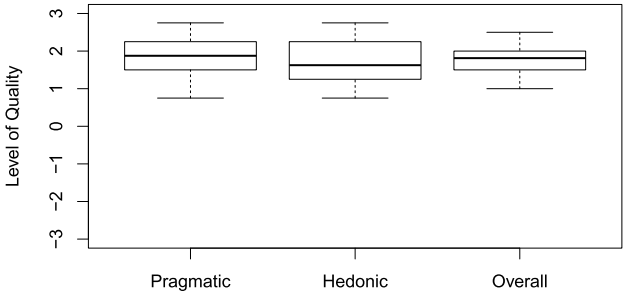


Figure 6: User experience questionnaire results from all participants.

Open-ended Questions

During the online questionnaire, we asked three open-ended questions. The first two asked participants to state what they liked and disliked about the learning app, and the last questions asked participants to state potential new features or improvements for the app. We employed affinity diagramming to sort and categorize atomic statements for each question [5].

What did You Like about the App?

Four participants highlighted the simplicity as what they liked most about the app, saying “it is so simple” (P12) and “trying out different neural networks with different settings is pretty easy, and it is satisfying to see the results of it” (P7). Moreover, five participants mentioned the explanations in the form of text and video as helpful. Additionally, our participants perceived the app’s animations and

interactivity positively, mainly referring to the model view and the implemented drag and drop for the Deep Learning model's visual "programming" as a pleasant experience. Furthermore, two participants stated that the step-by-step guide helped them to understand the workflow highlighting "the possibility to adjust a single step without having to redo the other ones" (P6) and stating that it is "very easy to understand the concept of Deep Learning by trying out different components" (P5). Finally, one participant said that it is great that no code is required (P11).

What did You Dislike about the App?

Five participants pointed out usability issues with the app. It was mentioned that the app has no continue button to get to the next step of the workflow (P4); instead, users have to click on actively look for the next step and click on it. Moreover, it was said that the example images in the model show cats and do not adapt to the currently selected classes, confusing users. Additionally, participants stated that they missed a progress bar for the loading of images in the training view and that the evaluation view's layout "is a little confusing" (P10).

Concerning the offered functionality of the app, three participants criticized the limited number of features. For example, one participant said that the "only a very basic, limited amount of layers; no activation function selection" (P12). Furthermore, one participant suggested that an example model "would have made it easier to understand how neural networks should be implemented" (P6). Additionally, one user missed the possibility to write code to describe the neural network (P5).

On a conceptual level, two participants stated problems with our application. One participant said that "some solutions are very bad" (P8), referring to the aspect that users can freely choose two image classes, which can potentially be very hard to classify (cf. Table 1). Moreover, another participant said that "wrong decisions are not punished enough – learning what is good needs more honest feedback" (P3).

Do You Have any Suggestions on how to Improve the App?

Overall, three participants suggested minor improvements of the user interface, asking for a continue button to go to the next step or saving the progress to come back to it later or not lose it accidentally, "I clicked by chance on an icon in my bookmark bar and had to go back but I lost all my progress" (P12). Three participants mentioned the explanation on the model view, saying that "the videos should be visible before placing the components and there should also be transcripts of the videos" (P10), and "a short description of each puzzle piece could be displayed when the mouse pointer is over it" (P9).

Furthermore, two participants suggested supporting users more during the creation of Deep Learning models. One participant said that "a step by step guidance when creating the neural network maybe helpful when creating a model for the first time" (P7), while another said that it makes sense to get suggestions for the next layer in the model architecture (P11). Currently, it is trial and error with an error message informing users when the layer cannot be connected; however, the participant said it would be better to see which layers would work beforehand.

Another suggestion from four participants is to include "expert levels to learn more details" (P3). For example, one participant said "maybe you could switch from the drag and drop for the layers to pseudocode" (P5), while another wanted a possibility to export the model to use it in Python (P1). Furthermore, it was suggested to add more features over time as users get more advanced, "add regularization methods like dropout, batch normalization and different activation functions" (P2).

Subjective Ratings

We asked the participants to rate two statements with 7-point Likert-scale items (1=strongly disagree, 7=strongly agree). Participants stated that the app helped them to gain experience with Deep Learning (Md=5, IQR=2.25), while they strongly agreed that the app is useful for Deep Learning beginners (Md=7, IQR=1).

Interestingly, the statement that the app is useful for Deep Learning beginners was rated differently depending on the participants' expertise. Participants with no experience in Machine Learning (Md=6.5, IQR=0.5) and less than six months of experience (Md=6, IQR=1) agreed less strongly with the statement than participants with more than six months of Machine Learning experience (Md=7, IQR=0). Since we had five participants for both less and more than six months of experience, we directly performed a Wilcoxon test that revealed no significant differences between the groups ($W=2.5$, $Z=1$, $p=0.424$, $r=0.32$) (r: > 0.1 small, > 0.3 medium, and > 0.5 large effect).

6.4 Discussion

In the following, we discuss the insights of the conducted user evaluation.

User Experience

The UEQ suggests that value below −0.8 represents a negative evaluation, while values above 0.8 are considered a positive evaluation [15]. The extremes of −3 and 3 represent horrible bad and extremely good user experiences, respectively. For pragmatic, hedonic, and overall quality, we achieved values above 0.8 for all participants, indicating a positive user experience of our learning app.

Trade-off between Interface Complexity and Feature Extent

Participants liked our app's simplicity and were mostly positive about the detailed explanations presented throughout the app. Moreover, they liked the app's interactivity and the visual programming of the model architecture in particular. However, participants mentioned that the functionality is too limited when users advance, and the app should offer possibilities to use the trained classifier for real-world scenarios. Since we focused on Deep Learning beginners, the offered functionality is a trade-off between a too complicated and overwhelming number of functions and possibilities on one side and too limited functionality that does not reflect a realistic and required set of features on the other side. We argue that our learning app is a good compromise between these two extremes, and we showed that it benefits the experience to focus on a concrete target group and skill set, especially when developing educational resources. This is highlighted for one by the good user experience and reflected in the rated Likert-items showing that participants saw the app as useful for beginners.

Nevertheless, to stronger contribute to a significant learning experience, it makes sense to introduce new concepts and features as users advance iteratively. Several participants stated that it would make sense to increase the complexity depending on the learners' progress. For example, it was suggested to increase the model architecture's complexity over time by introducing more layer types, activation functions, or Python code. However, it is difficult to identify which features should be added and when is the right time to do so. Here, our user evaluation showed that the user-centered approach we applied helped us to successfully identify an initial feature set that did not overwhelm the users. Hence, we believe that methods from the user-centered design approach such as focus groups are helpful to identify which features should be introduced at which stage of the learning process. Furthermore, these methods can help to specify the prerequisites that users need to fulfill to unlock new features.

Increasing the Usefulness for Long-term Motivation

Increasing complexity would allow users to benefit from the app over a longer time, effectively reducing the need to switch between several learning environments that only introduce a few concepts each. However, to improve the long-term motivation, participants said that they would like to see actual applications that work in combination with the learning app, empowering them to use their trained classifier beyond learning contexts. One could imagine different mobile apps that support a trained model's import directly from the learning app. For example, gallery apps could allow users to sort their pictures or automation apps such as IFTTT[10] would allow one to connect different apps and react to events. An example would be that incoming emails with attached images of animals are automatically stored in the user's gallery, while images that contain documents are placed into cloud storage. However, these possible use-cases are limited because it is hard to train for undefined classes. For example, to distinguish between pictures with buildings and pictures without them. Nevertheless, this is a problem with Deep Learning solutions in general.

7 General Discussion

The focus of our application is to empower beginners to gain practical experience with Deep Learning. Hence, we created a minimalistic design that teaches the user the basic concepts while solving practical problems.

Scope of Our Application

We tried to identify the essential features our application should include with methods from the human-centered design process. However, during our technical evaluation, we realized that our design does not include many countermeasures against overfitting. Overfitting could become a problem because the small dataset sizes encourage this phenomenon. Due to missing generalization, that is, the ability of the model to adapt to previously unseen data overfitting affects the training process very early [24]. Currently, our prototype only allows the user to stop the training process at the right time to avoid overfitting. Nevertheless, several features would enable users to combat this phenomenon. For example, the application could introduce the dropout layer type after the user encountered the effects of overfitting the first time. The dropout layer would

10 IFTTT. https://ifttt.com, last retrieved December 15, 2020.

allow the user to add regularization to the model, which reduces overfitting [20]. Another option is the introduction of data augmentation [25]. However, this process by itself is computationally demanding, since it transforms every image in various ways to increase the size of the dataset artificially.

User-centered Approach

From analyzing previous work, we learned that it is rare that applications in the domain of Deep Learning apply the complete human-centered design process. We argue that this is problematic, especially when considering how frequent DL solutions are deployed. Moreover, we saw that only a few educational tools are available to understand Machine Learning, with, to our knowledge, none of them focusing on the Deep Learning workflow specifically. Hence, we implemented a learning application for beginners that utilizes the complete human-centered design process. The results of our user evaluation support our claims that the application provides good user experience, empowering beginners to understand the Deep Learning workflow and gaining hands-on experience. Following the user-centered design process [11], allowed us to identify problems early in the process and helped to reduce the overwhelming complexity with which DL beginners are confronted. Our user evaluation unveiled that even a specific user group such as DL beginners has inherently a large heterogeneity. For example, some users missed writing code while others enjoyed using the visual programming for constructing the models; similarly, some said that they would like even more guidance with the existing features while others found them sufficient and instead expressed interest in a larger feature set. Here, applying methods from user-centered design can help to unveil these different user needs and help constructing positive user experiences.

Extending Beyond Image Classification

Currently, the learning application can only solve image classification tasks via convolutional neural networks which contributed to the workflow being rather straightforward. However, Deep Learning can be applied to a wide range of tasks such as language translation or virtual assistance. Each of these tasks has different requirements that need to be considered and it is likely that our application can not be extended to all of them. Nevertheless, we believe that the workflow is similar for all of these tasks and learned knowledge can be transferred to them. Moreover, our learning application can be extended to other tasks with similar requirements as well. For example, other classification tasks of time-irrelevant data such as the classification of different sounds. With the integration of Long-Short Term Memory (LSTM) networks, we believe that the scope of our application can be applied to recurrent neural networks as well, enabling processing of time-relevant data (e. g., continuous sensor data). With regard to the availability of datasets for these different tasks, we could allow users to record their own data (similar to [28]) and allow to upload and share it with other users of our app.

Limitations

There are some limitations to our underlying concept and current implementation. For example, depending on the two selected classes, it can be hard to achieve an acceptable training accuracy (e. g., for cats and dogs). Here, we decided to give users the freedom to choose the different classes without any restrictions; however, it can be frustrating for users if their classifier does not perform, and they do not understand why. As a first step, we suggest giving a general explanation in the evaluation view describing some typical reasons in case the results are below a certain threshold. To better address this issue, we think it makes sense to develop an intelligent recommendation system that can give more specific insights based on the selected parameters. However, given the complexity it is possible that such a system is hard to achieve. Another limitation of our current implementation is that the trained classifier cannot be used for real-life scenarios (instead, users can only upload pictures in the app to classify them). To keep users motivated, we should implement the possibilities discussed in the paragraph “Improving the Experience” such as third-party apps that allow to use the trained classifier.

Future Work

We already addressed the addition of the dropout layer type. This feature allows users to combat the effects of overfitting, a common solution for this problem in real-world scenarios. In the future, the application could allow the user to deploy their trained model to an external device (e. g., their smartphone). We could implement an app that enables the user to download their trained model from the backend, for example, for the classification of images from the smartphone’s gallery with a respective companion app. Furthermore, future work should investigate ways to iteratively extend the applications’ functionality, especially when to present which new functionality to the user.

8 Conclusion

This paper developed a web-based learning application that helps beginners understand Deep Learning workflows and solve practical problems. We followed the human-centered design approach to create a user-friendly interface and conducted a technical evaluation to demonstrate which problems can be solved in the app and a user evaluation to understand users' experience. Our results show that our learning application is well-suited for beginners and that the simplicity and interactivity helps to understand Deep Learning and empowers them to train their image classifiers. In the future, we will investigate possibilities to include more functionality to support advancing users iteratively and explores solutions to deploy the trained classifier in real-world scenarios.

Acknowledgment: We would like to thank our Deep Learning experts from the field of HCI Abdallah El Ali, Niels Henze, and Sven Mayer.

References

[1] Peter Brusilovsky. Explanatory visualization in an educational programming environment: Connecting examples with general knowledge. In Brad Blumenthal, Juri Gornostaev, and Claus Unger, editors, *Human-Computer Interaction*, pages 202–212, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.

[2] Rich Caruana, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm, and Noemie Elhadad. Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1721–1730, 2015.

[3] Wei Chen, Haifeng Wei, Suting Peng, Jiawei Sun, Xu Qiao, and Boqiang Liu. Hsn: Hybrid segmentation network for small cell lung cancer segmentation. *IEEE Access*, 7:75591–75603, 2019.

[4] Shuchi Grover and Satabdi Basu. Measuring student learning in introductory block-based programming: Examining misconceptions of loops, variables, and boolean logic. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, pages 267–272. ACM, 2017.

[5] Gunnar Harboe and Elaine M. Huang. Real-world affinity diagramming practices: Bridging the paper-digital gap. In *Proc. 33rd Annual ACM Conf. Human Factors in Computing Systems*, pages 95–104. ACM, 2015.

[6] Fred Hohman, Nathan Hodas, and Duen Horng Chau. Shapeshop: Towards understanding deep learning representations via interactive experimentation. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*. ACM, 2017.

[7] Fred Hohman, Minsuk Kahng, Robert Pienta, and Duen Horng Chau. Visual analytics in deep learning: An interrogative survey for the next frontiers. *IEEE Transactions on Visualization and Computer Graphics*, 2018.

[8] Fred Hohman, Haekyu Park, Caleb Robinson, and Duen Horng Chau. Summit: Scaling deep learning interpretability by visualizing activation and attribution summarizations. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 2020.

[9] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[11] ISO. Ergonomics of human-system interaction – part 210: Human-centred design for interactive systems. Standard ISO-9241-210:2019, International Organization for Standardization, Geneva, CH, 2019.

[12] R. Ji, K. Li, Y. Wang, X. Sun, F. Guo, X. Guo, Y. Wu, F. Huang, and J. Luo. Semi-supervised adversarial monocular depth estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1, 2019.

[13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[14] C. Lewis. *Using the "thinking Aloud" Method in Cognitive Interface Design*. Research report. IBM T.J. Watson Research Center, 1982.

[15] Bettina Laugwitz, Theo Held, and Martin Schrepp. Construction and evaluation of a user experience questionnaire. In Andreas Holzinger, editor, *HCI and Usability for Education and Work*, pages 63–76, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[16] Zachary C Lipton. The mythos of model interpretability. *Queue*, 16(3):31–57, 2018.

[17] Anh Nguyen, Alexey Dosovitskiy, Jason Yosinski, Thomas Brox, and Jeff Clune. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In *Advances in neural information processing systems*, pages 3387–3395, 2016.

[18] Jakob Nielsen and Rolf Molich. Heuristic evaluation of user interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '90, page 249–256, New York, NY, USA, 1990. Association for Computing Machinery.

[19] Daniel Smilkov, Shan Carter, D Sculley, Fernanda B Viégas, and Martin Wattenberg. Direct-manipulation visualization of deep networks. *arXiv preprint arXiv:1708.03788*, 2017.

[20] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[21] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[22] Ben Shneiderman, Catherine Plaisant, Maxine Cohen, Steven Jacobs, Niklas Elmqvist, and Nicholas Diakopoulos. *Designing the User Interface: Strategies for Effective Human-Computer*

Interaction. Pearson, 6th edition, 2016.
[23] Bret Victor. Explorable explanations, Mar 2011.
[24] Mathukumalli Vidyasagar. *A theory of learning and generalization*. Springer-Verlag, 2002.
[25] Jason Wang and Luis Perez. The effectiveness of data augmentation in image classification using deep learning. in *Convolutional Neural Networks Vis. Recognit*, 2017.
[26] David Weintrop and Uri Wilensky. Comparing block-based and text-based programming in high school computer science classrooms. *ACM Transactions on Computing Education (TOCE)*, 18(1):3, 2017.
[27] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.
[28] Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579*, 2015.

Bionotes

Sven Schultze
University of Oldenburg, Oldenburg, Germany
sven.schultze@uol.de

Sven Schultze is a Master Student for Computer Science at the University of Oldenburg. He conducts research in the field of Machine Learning and Human-Computer Interaction. He is especially interested in Deep Learning, Computer Vision, and User Experience.

Uwe Gruenefeld
University of Duisburg-Essen, Essen, Germany
uwe.gruenefeld@uni-due.de

Dr. Uwe Gruenefeld is a Postdoc Researcher in Human-Computer Interaction at the University of Duisburg-Essen, Germany. He is fascinated by Augmented and Virtual Reality, with a strong interest in Intelligent User Interfaces. His research has mainly focused on investigating Peripheral Visualization, Attention Guidance, and Multimodal Interfaces.

Susanne Boll
University of Oldenburg, Oldenburg, Germany
susanne.boll@uol.de

Prof. Dr. Susanne Boll is Professor of Media Informatics and Multimedia Systems in the Department of Computing Science at the University of Oldenburg in Germany. She serves on the executive board of the OFFIS Institute for Information Technology in Oldenburg and heads the competence cluster Human Machine Collaboration. Her research area lies at the intersection of human computer interaction and interactive multimedia. She is developing novel interaction technology that is shaped towards a respectful and beneficial cooperation of human and technology in a future more and more automated world.