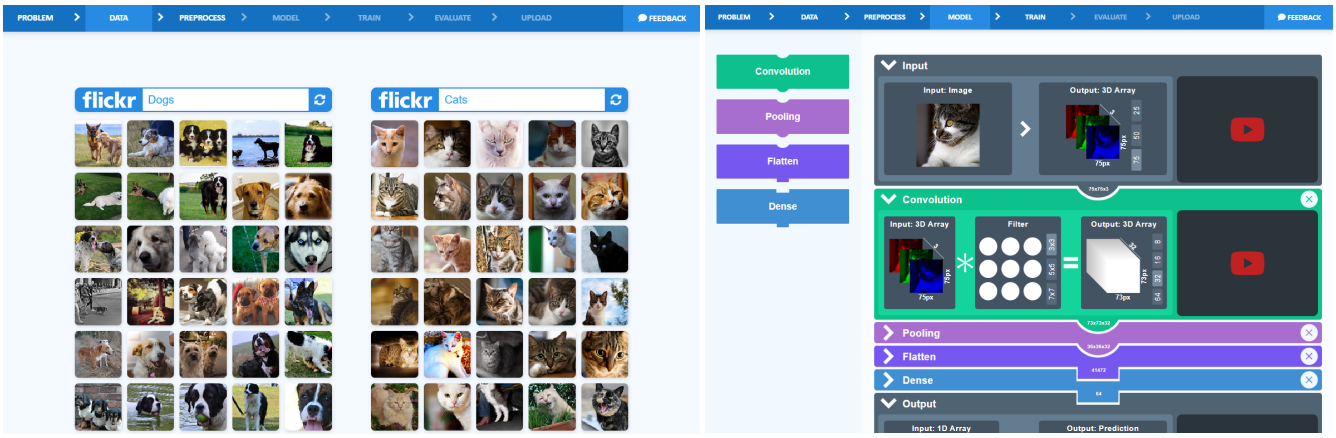
Demystifying Deep Learning: Developing a Learning App for Beginners to Gain Practical Experience

Sven Schultze
sven.schultze@uni-oldenburg.de
University of Oldenburg
Oldenburg, Germany

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

Susanne Boll
susanne.boll@uni-oldenburg.de
University of Oldenburg
Oldenburg, Germany



(a) Flickr integration for training and test datasets. **(b) Visual programming (puzzle-based) to create model.**

Figure 1: Implementation of our web application to demystify deep learning. *Best seen in color.*

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 for beginners 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. In the future, we plan to conduct a user study to evaluate our learning application online.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; • **Applied computing** → **Interactive learning environments**.

KEYWORDS

deep learning, machine learning, explainable, education

MuC'20 Workshops, Magdeburg, Deutschland

https://doi.org/10.18420/muc2020-ws111-334

1 INTRODUCTION

In recent years, deep learning 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 [6, 10, 13, 21]. 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 approaches [4, 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 [15]. 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, 27]. For example, visual programming languages can help one to understand the concepts of programming quickly [5, 25]. 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 understand 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. Our work contributes insights into the development of a user-centered learning application for deep learning and the app itself.

2 RELATED WORK

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 [3]). 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) [15].

Previous work proposes different visual analytic tools to support model explanation, interpretation, and debugging [8]. For example, Yosinski et al. suggest two different tools [27], 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 [7]. 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 [16]. Similar approaches are activation aggregation and neuron-influence aggregation [9]. 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.

2.2 Educational Applications for Deep Learning

Interactive visualizations can significantly increase beginners' understanding of program behavior [2]. They can be integrated into learning experiences with explorable explanations, for example [22]. 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[1] 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. Another example is Tensorflow Playground[2] 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. 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.

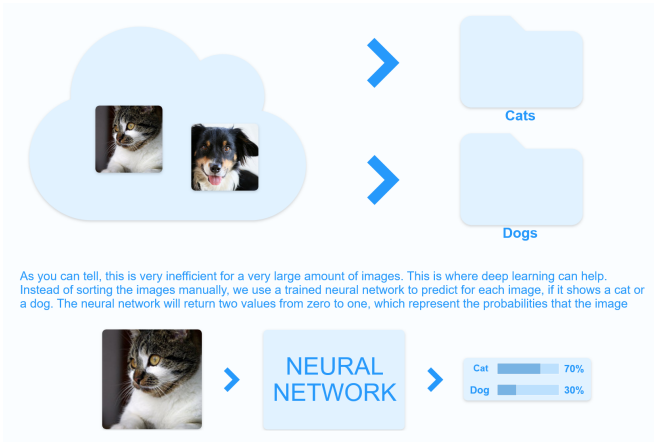
3 METHOD

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.

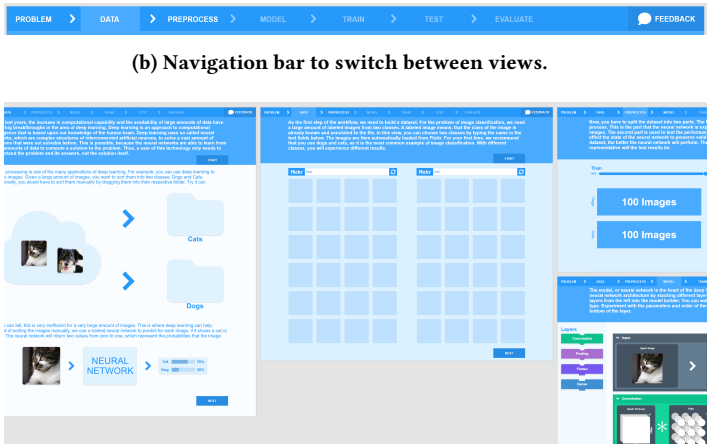
4 DEVELOPING THE LEARNING APPLICATION

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].

[1] Teachable Machine. https://teachablemachine.withgoogle.com
[2] Tensorflow Playground. https://playground.tensorflow.org



(a) Interactive introduction of classification problems.



(b) Navigation bar to switch between views.

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

Figure 2: Lo-Fi prototype development of our application in Adobe XD. *Best seen in color.*

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[3]. The prototype is shown in Figure 2. 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:

[3] Adobe XD. https://adobe.com/xd, last retrieved August 10, 2020

Problem. The first view the user sees is the problem view. Here, the classification problem is illustrated using an explorable explanation [22], where two images are manually classified into cats and dogs (see Figure 2a).

Data. The data view is split into two halves (see the prototype view in the center of Figure 2c). 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.

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 [18], which state that users should receive informative feedback for their actions, and Nielsen's heuristics for user interface design [17], 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[4] 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[5], and developed our server in Python using Tensorflow[6]. Client-server communication is based on sockets.

Users can create custom datasets with little effort. To achieve this, we use Flickr's REST-API[7], 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[8], allowing users to train networks using their hardware.

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.

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

[4] Our learning application. http://ai.uol.de
[5] Vue.js. https://vuejs.org, last retrieved August 10, 2020
[6] Tensorflow. https://www.tensorflow.org, last retrieved August 10, 2020
[7] Flickr API. https://www.flickr.com/services/api/request.rest.html, last retrieved August 10, 2020
[8] Tensorflow.js. https://www.tensorflow.org/js, last retrieved August 10, 2020

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. We trained the model for 20 epochs on each dataset with a 75%/25% train/test split (see Table 1).

Table 1: Resulting accuracy for different example datasets.

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

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 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, 26]. 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 3).

As Figure 3 shows, the backend trainer finished more than three times faster than the TensorFlow.js frontend trainer. 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

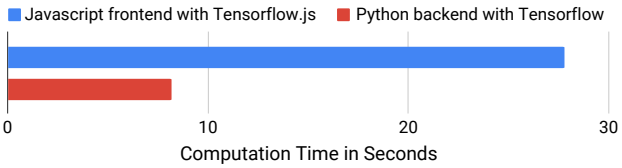


Figure 3: Computation time in frontend vs. backend

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).

6 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.

6.1 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 [23]. 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 allow the user to add regularization to the model, which reduces overfitting [20]. Another option is the introduction of data augmentation [24]. However, this process by itself is computationally demanding, since it transforms every image in various ways to increase the size of the dataset artificially.

6.2 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.

6.3 Future Work

We plan to conduct an online study to evaluate the usability and learning support of our web application in the wild. Therefore, a short dialogue will ask users that try out the app if they are willing to participate in a user study. If they agree, we ask them to walk through the app from beginning to end and to train and evaluate a model. Then, we will ask them questions about their experience. For example, we will measure usability with the SUS questionnaire [1]. If users continue using the application, we ask them if we can collect data on their performance. Additionally, we plan to integrate our application into university courses on machine learning and acquire direct feedback via interviews.

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.

7 CONCLUSION

In this paper, we 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. Next, we plan to do an online evaluation to determine usability and what beginners are able to achieve with the web application.

ACKNOWLEDGMENTS

We would like to thank our deep learning experts Abdallah El Ali, Niels Henze, and Sven Mayer.

REFERENCES

[1] John Brooke et al. 1996. SUS-A quick and dirty usability scale. *Usability evaluation in industry* 189, 194 (1996), 4–7.

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

[3] Rich Caruana, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm, and Noemie Elhadad. 2015. 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*. 1721–1730.

[4] Wei Chen, Haifeng Wei, Suting Peng, Jiawei Sun, Xu Qiao, and Boqiang Liu. 2019. HSN: Hybrid Segmentation Network for Small Cell Lung Cancer Segmentation. *IEEE Access* 7 (2019), 75591–75603.

[5] Shuchi Grover and Satabdi Basu. 2017. 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*. ACM, 267–272.

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

[7] Fred Hohman, Nathan Hodas, and Duen Horng Chau. 2017. 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.

[8] Fred Hohman, Minsuk Kahng, Robert Pienta, and Duen Horng Chau. 2018. Visual Analytics in Deep Learning: An Interrogative Survey for the Next Frontiers. *IEEE Transactions on Visualization and Computer Graphics* (2018).

[9] Fred Hohman, Haekyu Park, Caleb Robinson, and Duen Horng Chau. 2020. Summit: Scaling Deep Learning Interpretability by Visualizing Activation and Attribution Summarizations. *IEEE Transactions on Visualization and Computer Graphics (TVCG)* (2020). https://fredhohman.com/summit/

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

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

[12] R. Ji, K. Li, Y. Wang, X. Sun, F. Guo, X. Guo, Y. Wu, F. Huang, and J. Luo. 2019. Semi-Supervised Adversarial Monocular Depth Estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2019), 1–1.

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

[14] C. Lewis. 1982. *Using the "thinking Aloud" Method in Cognitive Interface Design*. IBM T.J. Watson Research Center. https://books.google.de/books?id=F5AKHQAACAAJ

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

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

[17] Jakob Nielsen and Rolf Molich. 1990. Heuristic Evaluation of User Interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Seattle, Washington, USA) *(CHI '90)*. Association for Computing Machinery, New York, NY, USA, 249–256. https://doi.org/10.1145/97243.97281

[18] Ben Shneiderman, Catherine Plaisant, Maxine Cohen, Steven Jacobs, Niklas Elmqvist, and Nicholas Diakopoulos. 2016. *Designing the User Interface: Strategies for Effective Human-Computer Interaction* (6th ed.). Pearson.

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

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

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

[22] Bret Victor. 2011. Explorable Explanations. http://worrydream.com/ExplorableExplanations/

[23] Mathukumalli Vidyasagar. 2002. *A theory of learning and generalization*. Springer-Verlag.

[24] Jason Wang and Luis Perez. 2017. The effectiveness of data augmentation in image classification using deep learning. *Convolutional Neural Networks Vis. Recognit* (2017).

[25] David Weintrop and Uri Wilensky. 2017. Comparing block-based and text-based programming in high school computer science classrooms. *ACM Transactions on Computing Education (TOCE)* 18, 1 (2017), 3.

[26] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks?. In *Advances in neural information processing systems*. 3320–3328.

[27] Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. 2015. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579* (2015).