Virtual Reality (2022) 26:159–171
https://doi.org/10.1007/s10055-021-00550-1

ORIGINAL ARTICLE

利用虚拟现实 民主化人工智能在生物医学图像分类中

Democratizing AI in biomedical image classification using virtual reality

Kevin VanHorn[1] · Murat Can Çobanoğlu[1]

Received: 15 September 2020 / Accepted: 9 June 2021 / Published online: 18 June 2021

Abstract
Artificial intelligence models can produce powerful predictive computer vision tools for healthcare. However, their development simultaneously requires computational skill as well as biomedical expertise. This barrier often impedes the wider utilization of AI in professional environments since biomedical experts often lack software development skills. We present the first development environment where a user with no prior training can build near-expert level convolutional neural network classifiers on real-world datasets. Our key contribution is a simplified environment in virtual reality where the user can build, compute, and critique a model. Through a controlled user study, we show that our software enables biomedical researchers and healthcare professionals with no AI development experience to build AI models with near-expert performance. We conclude that the potential role for AI in the biomedical domain can be realized more effectively by making its development more intuitive for non-technical domain experts using novel modes of interaction.

Keywords Deep learning · Machine learning · Neural nets · Virtual reality · Visualization

1 Introduction

Recent advances in computing have enabled the routine training and deployment of deep convolutional neural networks (CNN). This family of machine learning (ML) algorithms have enabled the solution of numerous supervised learning computer vision tasks. This has led to advances in multiple biomedical domains as disparate as clinical imaging, electronic health records, and wearable devices (Esteva et al. 2017; Le et al. 2016; Miotto et al. 2018; Vardhana et al. 2018). Novel applications of deep learning (DL), an important sub-field of artificial intelligence (AI), have the potential to transform biomedicine, with applications in both basic science as well as in the clinic. In parallel, technological advancements have made virtual reality (VR) viable in routine applications. VR offers inherent advantages for visualization and interaction, which can benefit the integration of an accessible deep learning tool. We have developed a VR platform that supports experiential learning (Kolb 2014) of DL through the construction and inspection of models in an intuitive and immersive environment. We introduce this tool and criteria for its design to exemplify how interpretable model construction can democratize applications of AI for biomedical images.

Machine intelligence development, at its core, requires a build, compute, and critique loop (Blei 2014). Biomedical researchers are effective at the critique of model performance when performing a biomedical classification task, but they are likely to have difficulty with the model building and computing parts. In theory, this can be overcome with collaboration and effective communication between individuals; in practice, it is difficult to form and effectively operate teams with as diverse backgrounds as machine learning and biology. This impedes the application of AI to biomedicine.

We present a generalizable set of principles and an implementation *(github.com/Cobanoglu-Lab/Devoreann)* thereof that enable a biomedical expert to effectively develop a CNN model on real-world data. While there have been attempts to address specific parts of the AI knowledge barrier, no previous work allows an ML novice to perform at near-expert levels.

Briefly, there are two categories of prior work targeted to people without training in ML: visualizations (Harley

✉ Kevin VanHorn
Kevin.VanHorn@utsouthwestern.edu

Murat Can Çobanoğlu
MuratCan.Cobanoglu@utsouthwestern.edu

[1] University of Texas Southwestern Medical Center, 5323 Harry Hines Blvd, Dallas, TX 75390, USA

2015; Meissler et al. 2019; Schreiber and Bock 2019; Keras.js 2019; TensorSpace.js 2019) and interactive platforms (Smilkov and Carter 2016; ConvNetJS 2019; Lobe 2019). Visualization methods such as Tensorspace.js (TensorSpace.js 2019) only show pre-trained models on a pre-determined dataset such as MNIST and are developed by ML experts for a computer science audience. While useful perhaps in familiarizing newcomers to the fundamental principles of ML, this methodology is not suitable for the needs of a biomedical specialist working to solve a specific problem.

Interactive platforms such as Tensor Playground (Smilkov and Carter 2016) allow for dynamic model construction but are likewise generally limited to fixed datasets as well because they are designed for education. Thus, such tools are not equipped with the capability necessary to prototype an AI system on a biomedical research dataset. The only accessible and viable alternative is "drag-and-drop" style tools (NVIDIA DIGITS 2019; Deep Cognition 2019; Azure Machine Learning 2019; Neural network modeler 2019; Neural Network Console 2019) which provide a breadth of features but require intermediate or expert knowledge of ML to configure and use. In controlled user studies, we found one such tool, Sony's Neural Network Console (Neural Network Console 2019), to be generally unintuitive for users with no background in ML (see Results). Hence, in the context of other works, our tool is the first that can effectively enable a biomedical researcher to prototype state-of-the-art AI techniques at near-expert performance.

2 Methods

2.1 Objectives

It has long been recognized that successful machine intelligence development requires an iterative process of model design, training, and criticism–Box's loop (Box 1976). Our primary objective was to create an easy-to-use and immersive platform that provides the opportunity to perform all three steps, with an emphasis on model critique. This is necessary for biomedical professionals with no prior experience in DL to define and train models that solve real biomedical domain problems and improve using their domain knowledge. Therefore, our desiderata were to:

I. Construct an immersive and engaging experience
II. Provide focused interactions
III. Require no prior technical expertise
IV. Present flexible deployment options
V. Prevent erroneous model design

Our implementation, *Devoreann*, achieves these goals as we elaborate throughout this section.

2.2 Design

VR platforms precisely translate movements from the physical world to the virtual world. This enables distinct advantages in interactivity and navigation when compared to a desktop application. An "intuitive" user interface requires effective interaction without consciously using previous knowledge (Naumann et al. 2007). Thus, a VR interface can better facilitate design principles such as discoverability, affordance, and explorability (Everett 2018) due to its ability to mimic known physical actions such as grabbing or turning to look around.

Furthermore, the immersive aspect of VR reduces "information clutter" and outside distractions (Bowman 2007), resulting in an experience that is more effective for maintaining attention and focus (Sacks et al. 2013). Another advantage is that a controlled environment is fully programmable; therefore any "wrong" decisions can be rectified automatically by the application without requiring the user to interpret error messages. Finally, experts often describe neural networks in terms of "layers," suggesting an easily communicable physical representation for the more complicated programmatic representation of a neural network. Therefore, we chose to implement our software in VR to satisfy our design goals I-III and assist V.

The core aspect of the application is building, training, and criticizing the CNN. The users simply grab layers and put them into the workspace to build (Fig. 1). If any arrangement needs to occur, users can simply pick up any layers and move them. We keep this functionality simple to ensure that users can interact with the application in a familiar way. An element of familiarity is especially important for users new to virtual reality so that their interaction within the interface is not an impedance. To discard a layer, the user can grab the object and then simply throw it away. We apply virtual physics to the object at this point so that it falls to the ground and disappears on contact with the floor. Users are not responsible for calculating data dimensionality, defining program variables, or performing other cognitive overhead as we automatically adjust them in the background.

We modeled the workspace after containers from Google's Material Design (Design–Material Design 2019). Specifically, we adopted a "slipstream" design, which acts as an adaptable list that centers layers within a defined data stream. We apply a linear interpolation informed by the number of elements contained in the workbench to translate objects when a user modifies the stream. Through this method, the sequence can move to free up space wherever the user is attempting to place a layer. As a result, the interaction model suggests functionality by design rather than instruction. Furthermore, we can also ensure that

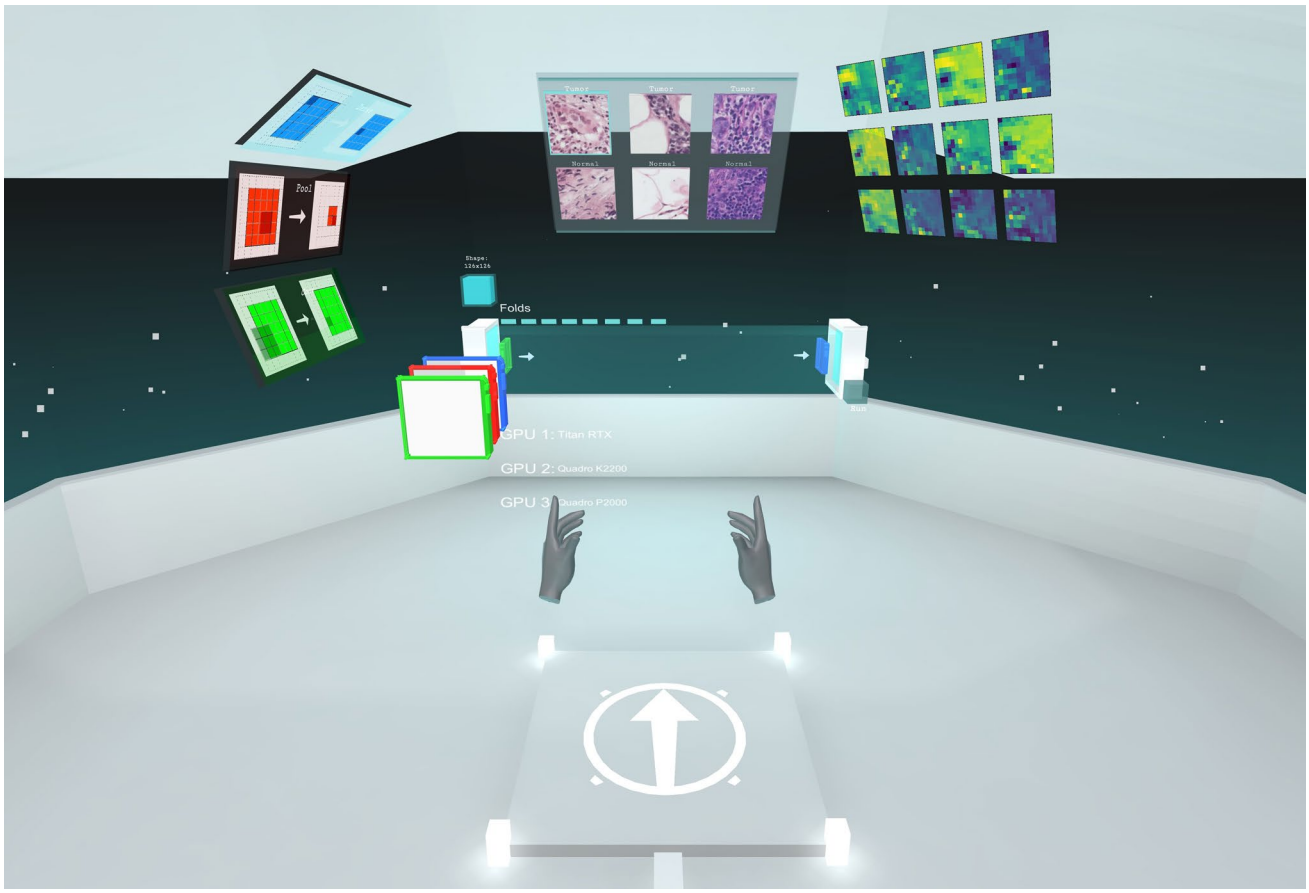

Fig. 1 An overview of our virtual reality environment in which medical professionals can construct deep learning models on a real-world histopathologic dataset

restrictions of the CNN are upheld. When users release dense layers at any position in the workspace, we propagate them to the back. This propagation communicates a common design choice when using this layer type in a CNN without direct instruction. Importantly, this ensures that it is impossible to design models that would produce an exception in Keras, in line with our design goal V. Similarly, we implement the core input and output layers of the model for the user to avoid grossly inaccurate or dysfunctional models. This process includes defining the initial shape with a convolution layer and classifying the final output. We communicate these operations to the user with a small convolution layer fixed at the leftmost point of the workspace and a small dense layer fixed at the right. The result is a smooth process where the user cannot create structurally erroneous models.

We found that VR necessitates certain considerations of continuity and spatial awareness to be effective. During testing, the scale and positioning of objects were especially important for user presence and ease of use. We positioned layers and the workbench to be comfortably within the user's immediate vicinity. Users do not have to move from their root positions. It was crucial that the application was functional without strenuous reaching or translational movement. We arranged the most relevant user interface (UI) components directly in front of the user and moved supplementary information, such as the model summary and layer animations, outside of the user's immediate field of view. This organization ensures that cognitive distractions and visual clutter are minimized for the main functional use cases of the tool. We also angle UI elements toward the user so that content is orthogonal to the user's field of view.

正交的，直角的

Certain architectural patterns tend to result in more successful convolutional neural networks. These patterns are specific to the classification domain and features of the dataset. Users can discern patterns for a dataset through iterative experimentation and informed evaluation. In Fig. 2, we illustrate an example workflow for a user. We report the area under the receiver operating characteristic curve (AUC) instead of model accuracy to better observe results with the influence of overfitting. Receiver operating characteristic (ROC) curves are widely accepted as the standard in diagnostic tests Obuchowski et al. (2004), and for alternate tasks this can easily be interchanged to Matthew's Correlation

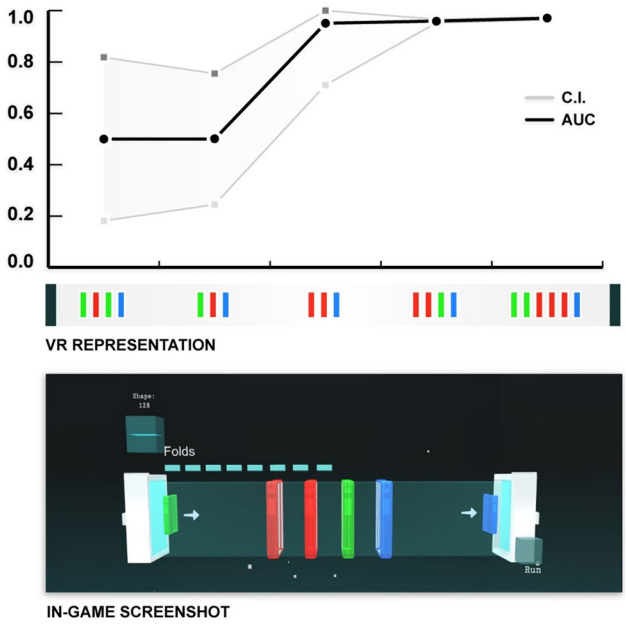

Fig. 2 An example of user progression within the VR application. The area under the ROC curve (AUC) and its 95% confidence interval (C.I.) are reported for each model built (*y*-axis) along with the VR model sequence representations (*x*-axis). Red, green, and blue objects represent max-pooling, convolutional, and dense layers, respectively

Coefficient Matthews (1975), F_1 score (Dice 1945; Sørensen 1948), or other evaluation metrics. The user starts with a larger, underperforming network. They then reduce the problem to three layers for simplification. The user further progresses by identifying the effectiveness of multiple pooling layers. The expert insight–unbeknownst to the user' is that max pooling layers reduce the dimensionality of the image so that the model is forced to prioritize features that are most important to the classification task. By the fifth stage, the user reintroduced convolutional layers to achieve an AUC of 0.97. With this progression, we communicate the need for pooling layers to make convolutional elements effective without overfitting the data. The first and fourth stages differ by only one layer, yet they produce significantly different results. This observation illustrates the volatility of networks and the need for reducing dimensionality.

As layers are added or removed from a CNN, the shape of the output tensor changes. To communicate this intuitively, we display the current shape with a peripheral at the top left of the workspace. This dimensionality indicator reflects the state of the model with a representational cube, chosen for its matrix-like shape (Fig. 3). The shape of the cube interpolates between sizes to reflect the state of the model. For example, adding a max pooling layer (with a configured stride of 2) reduces an input 128x128 pixel image to a shape of 64x64. In line with our goal of providing focused interactions (design goal II), we chose to exclude the fixed dimensions such as filter depth and color in this indicator. Before input passes through the first dense layer, we flatten the 2D image matrix to a 1D array. We display this reduction by visually flattening the cube indicator. Without an indicator, the rearrangement of layers would not have an easily

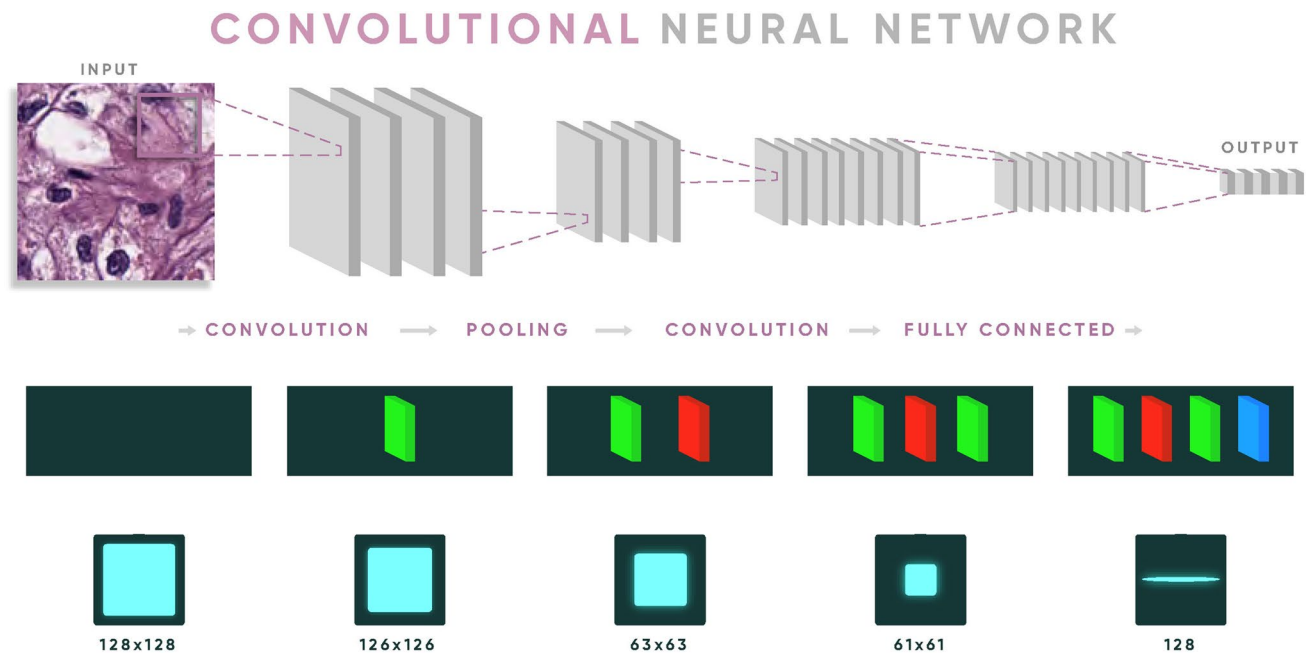

Fig. 3 Traditional representation of a neural network (top), our method of building a model (middle), and our dimensionality indicator within the VR environment to help build intuition as a model is built (bottom)

discernible effect on the model. An understanding of the shape and dimensionality of a model is crucial when defining a sequence of layers. This indicator provides real-time insight before training.

When the model is complete, we report the accuracy and 95% confidence interval at the display beneath the workbench. At this point, the application updates layers with their intermediate activations. We chose intermediate activations, because they reflect the meaning of each network filter and can be easily interpreted by users unfamiliar with deep learning. The image displayed corresponds to the first filter of the respective layer. As the user adds more layers, the resolution of the image reduces to reflect changes in shape. Dense layers after the flatten operation are not visualized because they are not spatially intuitive. We offer a more complete representation of the layer by projecting a 4x3 matrix of such activations at the user's right. This matrix contains additional filters for the layer that the user last touched. We also apply a ReLU activation by default to all convolutional and max pooling layers. The user can view the effects of this activation by holding a button on the controller. The user changes the matrix to their right by hovering on or picking up a layer.

Using the hover functionality, one can simply glide a hand through the sequence of layers to see a progression of the network. With this technique, the user can see the flow of data as it passes through the CNN. The user can then use this insight to modify the layer sequence. For example, the user may see that data is too large throughout the network. At this point, they would want to insert max pooling layers before any convolutional layers to trim down the data. For a comprehensive description of the model architecture, we display the Keras model summary output to the right of the user. We update the visualization with each new model run. The model summary contains the entire output shapes and additional details of the network. One such detail is the presence of ReLU activation layers and pre-configured layers automatically added by our system.

Certain models may be excellent at identifying one class of images but not another. Above the workspace, we display example images from the classes analyzed. When the user selects an image, we visualize the predicted activations on the next trained model. This information gives context and customization to the classification problem, supporting design goal I by providing an immersive and engaging experience. In this manner, the user can analyze the different effects of the CNN on each class. We provide this visualization to aid users in the refinement of their models. If many of the intermediate activations are blank, the model may not be effective. In this case, the user can reorder layers to better identify key features of the input image. The ReLU activations can also be insightful to the user. These activations pull out different features depending on the range of intensities in an image class. This visualization may motivate the user to reduce or increase the number of convolutional layers, for example.

2.3 Graphics

The VR environment is simplistic with minimal distractions (Fig. 1). We were able to place the individual in a unique and visually interesting setting. The user controls virtual hands to grab, throw, and place objects. The user can additionally point a laser from his/her index finger for item selection. When the appropriate button is pressed, we perform a continuous raycast from the finger to interactable elements, and only render the ray when an interactable object is intersected. We have also designed the environment to function with limited physical movement. The user only needs a small physical space to operate the application and can perform actions while standing or sitting. We implement the camera controller in the standard FOV of 90° which closely imitates the real-world FOV of an average human. We further minimize possible VR motion sickness by maintaining constant points of reference for the user and eliminating the need for translational movement.

Our VR interface surrounds the user with various tools for building deep learning models. We implement three types of filters in the context of the TensorFlow/Keras DL library (keras-team 2019). We limit the available layers to these core types for an easier introduction to the construction of CNNs (design goal III). These layers are visualized as rectangular objects in our application (Fig. 4). Users can grab layers from a tool shelf to their left and place them within a "workspace." The workspace is a defined region in front of the user that suspends a sequence of layers. Users can insert, remove, and rearrange layers within its bounds. These operations allow for simple construction and modification of a functional CNN. Each layer type is labeled and visually

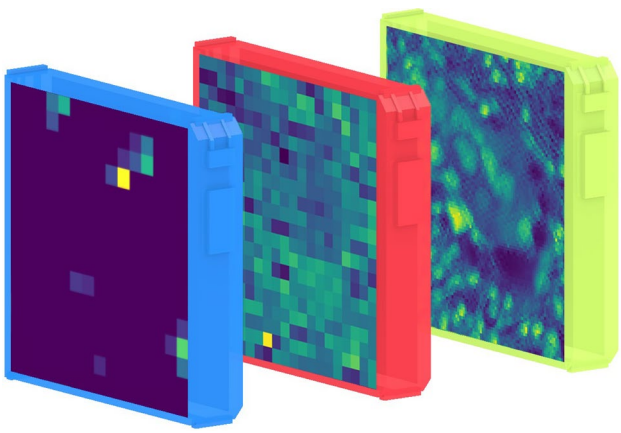

Fig. 4 A render of the representational layer objects in the VR application. Left to right: dense, max pooling, convolutional

elaborated with an animation that illustrates the behavior of the internal kernels. Animations are present at the user's left, above the available layers. After the user grabs a layer from the tool shelf, a new layer of the same type will spawn to fill its place. In this manner, there can be more than one layer of a given type within the model. The user defines the model from left to right, which is indicated by arrows within the workspace.

By pressing or selecting a button to the right of the workspace, users can then train a model. While computing, a display in front of the user reports the status of cross-validation during training. By default, we train the model with a *k*-fold stratification across multiple GPUs and display the progress of each GPU and fold. An individual configuring the application can, however, run the application on as few folds or GPUs as desired. Upon completion of a model, the same display reports the median accuracy of the model and 95% confidence interval against a testing set. A confidence interval is pertinent to the user because DL models vary with each model trained. Variance exists mainly due to random model initialization. For this purpose, a confidence interval provides better insight as compared to a single accuracy value by communicating the variability of a constructed architecture. We include an expanded technical description of the 3D visual environment and how it relates to user experience in Supplementary Note 4.

2.4 Configurability

The software runs as a standalone application operated by the standard requirements of the Oculus Rift VR headset. We chose this hardware for its intuitive controls that mimic grabbing and releasing actions. The Oculus Rift also benefits from a widely integrated application programming interface (API) for game engines. We developed the front-end of the application in the Unity game development platform (Unity 2018.2.15f1) and modified the standard "Oculus Integration for Unity" asset for base interaction functionality. We built the DL back-end with the TensorFlow/Keras framework using Python3. The Unity application and backend communicate core requests via gRPC (https://grpc.io/) and TCP (Transmission Control Protocol). gRPC is an open-source, bidirectional remote procedure call system developed by Google which exists as an interface between different programming languages over the network. We render and export intermediate activation images locally using OpenCV. One can configure our software to implement any set of image patches. While larger, multi-dimensional datasets benefit from computation over GPU clusters or our local parallel GPU support, this is not required. The tool can be easily modified for use on one GPU with a smaller dataset such as MNIST (Lecun et al. 1998).

A supervising user can configure our application to process and display a custom dataset. We specify "supervising," because the application is targeted as an explanatory tool, and non-technical users are not intended to modify the dataset and/or model parameters in keeping with design goal III. We intentionally separate the user from hyperparameters, optimizers, choice of activation functions, focusing instead on the model framework. As discussed in our accompanying user study, too many choices can overwhelm non-expert users and become an impediment to the experimentation process.

We update the in-game visuals and back-end model to reflect any changes in the data. In this manner, we can provide the benefit of our environment across classification tasks and domains. With further adaptation, a similar application could construct neural networks for additional fields that use CNNs. One such task is the label-free prediction of fluorescence images for transmitted-light microscopy (Ounkomol et al. 2018). In this case, one would not flatten the image, but instead output full-sized image predictions. Our application provides a configuration file that allows the user to specify parameters for the model and dataset. Supervising users can modify the batch size, the number of epochs, and details specific to each layer type. If the user provides custom data, they specify the number of classes, the name of each category, and the number of data points. The frontend and backend both update to reflect these changes. If the user specifies any new classes, the input selector above the workbench updates accordingly. In this situation, the user must provide up to six input images for the generation of activations.

As a proof of concept, we have integrated histopathologic image patches from the Camelyon16 dataset (Bejnordi et al., 2017) that are classified according to the presence of metastatic tissue for breast cancer (Liu et al. 2018). We obtained patches for binary classification from the PatchCamelyon benchmark (Veeling 2019). This adaptation of the Camelyon16 dataset labels image patches based on whether a tumor tissue exists in the center region of an image. With our integration of this new problem set, we have achieved accuracy upwards of 80% with minimal testing. We quickly found that to construct effective models for this particular dataset, users needed to construct larger networks by adding more layers. We tested a similar internal histologic dataset with seven classifications based on pathologist annotations: tumor, stroma, background, normal, inflammatory, blood, and necrosis. For targeting environments with less computational resources, we also tested MNIST (Lecun et al. 1998), SVHN (Netzer et al. 2011), Fashion-MNIST (Xiao et al. 2017), and EMNIST Letters (Cohen et al. 2017).

For the PatchCamelyon benchmark implemented in the user study, we compile models with the binary cross-entropy loss function and Adam optimizer in Keras (Kingma and

Ba 2017). Convolutional layers have 64 filters and a kernel size of (3,3). Convolutional layers are always followed by a ReLU activation as previously mentioned. Max pooling layers have a pool size of (2,2), and dense layers have 32 neurons. Every model has a convolutional layer and an accompanying ReLU activation to begin. At the end of each model, a dense layer exists with two neurons and a softmax activation. The number of neurons corresponds to the final number of classes and should be adjusted to reflect the task. If the user does not insert a dense layer, we put a flatten layer before the final classification. Otherwise, we perform this flatten immediately before the first dense layer that the user provides. For our training, we measured the performance of models using an eight-fold stratified cross-validation strategy. The application trains models at each fold until convergence with a low patience value. This means that the model can run over more epochs to achieve more consistent results. Along with the median model accuracy, we report a 95% confidence interval (CI) from the resulting accuracy of each fold (Kearnes et al. 2016).

2.5 Technical implementation

We support users in different environments and allow for extensibility. Virtual reality can be computationally taxing on local hardware. Quick response time for model training is crucial in such a platform, because user engagement contributes to many of the educational benefits in VR (Dede 2009). To this effect, a supervisor installing our application can configure the environment to be deployed independent of the resources available. One of our design goals (IV) was to enable flexible deployment through three modalities: cloud-based, networked, and local use. By taking advantage of these configurations, our solution allows for rapid, mobile deployment across one or more GPUs, distributed over one or multiple machines.

The front-end Unity application can run on any VR-ready PC or laptop. When a user builds and executes a model within the interface, we pass the layer sequence to an external DL architecture. The core component of the backend application is a persistent gRPC server that receives the layer sequence. The resulting accuracy of the trained model is served to the Unity C# gRPC client. Each instance of the Unity application connects to the server through a configurable communication channel. One can run the server locally, on a local network, or on a cloud-based computing platform. As a result, users can deploy the platform with distributed resources and independent of network limitations.

For single GPU use, we immediately train the model on the same thread as the server. When the training across a variable number of folds is complete, we pass the resulting accuracy to the front-end client. For multi-GPU use, we spawn a separate, persistent process for each GPU. Processes asynchronously consume jobs corresponding to each fold. We report the median accuracy across all the folds.

We stream progress across each fold to the Unity application from a separate persistent Python server to a TCP client in C# when models are training. Users can visualize the progress of their network within the VR interface. When the server is not run locally, we pass the model to the user's machine to compute the intermediate activation images in a final persistent Python client. Python implementations are all persistent, so that the primary server only has to be run once. The user can therefore suspend the application and begin a new session without configuring the backend.

2.6 User study

We conducted a user study to assess the benefits of our development environment over a state-of-the-art drag-and-drop alternative: SONY's Neural Network Console (Neural Network Console 2019). We sought to compare against a beginner-friendly tool that allowed visual network construction on non-trivial data but found our solution to be novel in this field. As discussed in Results, the freely available entry-level tools simply do not allow network construction on customizable data. For this reason, we chose Neural Network Console, which features node-based construction and dataset customization for intermediate to expert users.

Briefly, we used the PatchCamelyon (Veeling 2019) dataset to define a multi-class supervised learning problem and two associated tasks. We measured user performance by objective and subjective metrics, with detailed information on the procedure available in Supplementary Note 1. Before starting, we presented users with a brief overview of machine learning and VR, as described in Supplementary Note 2. Our hypotheses were that our interface would be more effective objectively by increased ease of use in addition to being more enjoyable subjectively. Thus, in *Results* we measure statistical significance of independent samples with a one-sided t test. The user study design and documentation were reviewed by the UT Southwestern Human Research Protection Program Office (HRPPO) and approved for exempt status. We conducted the study in an empty room and instructed the participant to remain seated to minimize "VR sickness." We ensured that there was ample room for head and arm movement, spacing the user away from the table to prevent physical collision and entanglement with headset wires. Finally, we sanitized equipment prior to each session and provided eye coverings for use with the headset.

The dataset contains images of lymph node sections labeled by the presence of metastatic tissue at the center of the image. This is a real dataset prepared to facilitate solutions to a real medical problem, therefore, the user study is grounded to real deep learning challenges. We reduced the data in each class only sufficiently as to render the time

spent training each model feasible. We used this dataset to construct two tasks for users.

In the first task, users were instructed to construct deep learning models with maximal accuracy. Users were given a maximum time constraint of 30 min, and the proctor was not permitted to speak with the user over suggested operations or actions that would benefit the user's score. The task ended when either the maximum time had elapsed, or the user decided that they were unable to improve the model. The target measurement for this task was the percent improvement between the starting AUC of 0.5 (reflecting a random, or "coin-toss" predictor) and the user's most accurate model during the allotted time was recorded. Previously published work reports that a specifically designed and expert-tuned model achieves 89.8% accuracy on this task (Veeling et al. 2018). We found that approximately 80% accuracy (or 60% improvement) was practically achievable using either interface, but it was difficult to proceed much beyond that. This demonstrated that neither the data reduction nor the limitations of these easy-to-use interfaces prevented achieving reasonably good performance. It also showed that the ceiling for improvement over baseline was limited to around 60% in practice. We hypothesized that our interface would confer a performance advantage to beginner and intermediate DL users.

In the second task, we gave the users a faulty architecture that was initially at 50% accuracy and asked them to find the single manipulation that would make it achieve around 70% accuracy. We terminated the task when the user could restore model performance and measured the time until successful task completion. We truthfully informed the users that the provided model was exactly one operation away from being restored to full performance, and specified that this operation could be removal, insertion or replacement of one layer, or swapping any two layers. In both interfaces, the correct answer was to add a single max pooling layer (for details, see Supplementary Note 3). Users were again given a maximum time constraint of 30 min, and the proctor was not permitted to suggest operations that would benefit the user's score. We designed this task to measure the user's understanding and deductive reasoning within a DL environment.

After these two tasks, we asked the users to anonymously rate their experience by asking them "How enjoyable was each tool?" and gave them the option to select a number between one and ten (inclusive) for both interfaces. This allowed us to assess the subjective quality of the experience. We hypothesized that the VR experience would be more enjoyable.

In order to make sure that any user-specific effects did not bias the results, each participant used both interfaces. This ensures that if a user were to perform outstandingly well, they would contribute to each interface's results, thereby negating any biases due to individual variation. Users were identified with an anonymized, generic identification number. We determined the ordering for each user with a balanced random shuffle.

Finally, we asked the users to self-report their expertise in DL. Since our tool has an educational focus, we only included the beginner and intermediate level users in our hypothesis testing analysis. Nevertheless, we recruited users from all skill levels, with the intention that the expert users would serve to demonstrate a benchmark. We performed a power analysis and determined that a total of $n \geq 10$ users were necessary, and we recruited 14 individuals. Ten users self-identified their deep learning experience as "beginner," two as "intermediate," and two as "expert." After 4 weeks, we met the minimum number of 10 beginner and intermediate level users, and we had no other interested users, therefore we terminated the user study. The user study was executed using a machine with a six-core Intel Xeon CPU E5-1650 with 32GB RAM and an NVIDIA GTX 1080 GPU for the front-end. The backend was run on a separate machine with two 16-core Intel Xeon E5-2683 CPUs, 256GB RAM, and three NVIDIA GPUs (TITAN RTX, Quadro K2200, and Quadro P2000).

3 Results

3.1 User study

We assessed the contributions of our method with a user study. As discussed in the introduction, the existing beginner-level ML development tools did not allow training on real-world data. The best alternative we found for a biomedical scientist who wanted to develop CNN models without coding was Sony's Neural Network Console (SNNC). The SNNC has an easy-to-use graphical interface relying on drag-and-drop style interactions common to desktop software. Furthermore, it allows model training on large-scale and user-supplied real-world data at no cost.

The most significant difference between our tool and SNNC is that we designed our tool explicitly for users having no background in ML. The SNNC is, like our tool, a graphical interface for creating SNNC also enables the construction of arbitrary CNN architectures by the user, training of the model, and assessment of its performance. The main difference between our tool and SNNC is that we deliberately chose to focus on usability, while SNNC has a more diffuse design. The main difference between our implementation and SNNC is not ML functionality, but rather effectiveness of facilitating real-world ML tasks for users with no prior training.

To construct the tasks, we used a real-world pathological patch classification dataset Veeling (2019) comprised of lymph nodes images derived from the Camelyon16 challenge

(Bejnordi et al. 2017). A successful computer vision solution for Camelyon16 can augment the daily work of pathologists. We used this dataset to construct two tasks.

For the first task, users were instructed to construct deep learning models with maximal accuracy within 30 min. We had the users start with approximately a random classifier (having AUC=0.5) and considered the most accurate model they could build and test within the time limit. Previously published work reports that a specifically designed and expert-tuned model achieves 89.8% accuracy on this task (Veeling et al. 2018). We measured the predictive performance of the models that participants could build using Devoreann and SNNC for this task.

In the second task, we started users with a faulty architecture that was initially at 50% accuracy and asked them to find the single manipulation that would make it achieve around 70% accuracy. We terminated the task when the user could restore model performance and measured the time until successful task completion. We truthfully informed the users that the provided model was exactly one operation away from being restored to full performance, and specified that this operation could be removal, insertion or replacement of one layer, or swapping any two layers. In both interfaces, the correct answer was to add a single max pooling layer (for details, see Supplementary Note 3). Users were again given a maximum time constraint of 30 min, and the proctor was not permitted to suggest operations that would benefit the user's score. We designed this task to measure the user's understanding and deductive reasoning within the context of CNN model design.

After these two tasks, we asked the users to anonymously rate their experience by asking them "How enjoyable was each tool?" and gave them the option to select a number between one and ten (inclusive) for both interfaces. This allowed us to assess the subjective quality of the experience. We hypothesized that the VR experience would be more enjoyable.

In order to ensure that user-specific effects did not bias the results, each participant used both interfaces. If a user performed outstandingly well, on merit of their individual talents independent of either software, they would contribute to each interface's results, thereby negating any biases due to individual variation from unfairly effecting only one method. Users were identified with an anonymized, generic identification number. We determined the ordering for each user with a balanced random shuffle.

Finally, we asked the users to self-report their expertise in DL. Since our tool has an educational focus, we only included the beginner and intermediate level users in our hypothesis testing analysis. Nevertheless, we recruited users from all skill levels, with the intention that the expert users would serve to demonstrate a benchmark. We performed a power analysis and determined that a total of $n \geq 10$ users were necessary, and we recruited fourteen individuals. Ten users self-identified their deep learning experience as "beginner," two as "intermediate," and two as "expert." After 4 weeks, we superseded the minimum number of ten beginner and intermediate level users, and we had no other interested users, therefore we terminated the user study.

We evaluated the performance of twelve beginner and intermediate users across our interface, Devoreann, and SNNC. Users participated on a voluntary basis. Twelve of the fourteen users were biomedical researchers or healthcare professionals. The two users with no experience in a medical field were both beginner users. As reported on the post-assessment questionnaire, users spent 4.04 h per week playing video games on average, and eight of the fourteen users tested had used VR before.

The proposed interface enabled significantly higher objective performance metrics and was also significantly more enjoyable for the participating users (Fig. 5). Our platform afforded ML novice users the predictive performance near experts using the SNNC. Thus, we demonstrate that medical professionals with little-to-no DL experience can effectively use such a tool to develop and better understand AI methods for image classification. We detail the experience of two users to illustrate the user study process in Supplementary Note 5. In addition, we investigate the benefits of an accessible tool for understanding DL in Supplementary Note 6.

In Task 1, users were able to achieve a significantly greater percent improvement in accuracy over the base model in our interface (Devoreann mean: 56.99%, SNNC mean: 1.80%, p value: 6.98e−11, one-sided t test). This result is similar if we disregard the two individuals with no experience in a medical field (Devoreann mean: 56.51%, SNNC mean: 1.68%, p value: 9.084e−09, one-sided t test). Many users were unable to improve a model in SNNC because their networks would frequently fail to compile. Users were able to achieve a wide range of accuracies and confidence intervals during a single session in Devoreann. One user commented in reference to SNNC: "It's overwhelming; it's just too many choices. You don't know where to start."

In Task 2, users were able to more quickly identify an operation which would correct an inefficient model using our interface (Devoreann mean: 181.60 sec., SNNC mean: 376.14 sec., p value: 5.71e−02, one-sided t test). In the entire study, there was only a single case where a user was not able to complete the task, and it occurred with SNNC in Task 2 for a beginner. The VR users also had less variation in their completion times compared to the alternative.

In the subjective evaluation, users enjoyed the proposed interface significantly more than the alternative (Devoreann mean: 8.58/10, SNNC mean: 4.71/10, p value: 6.49e−6, one-sided t test). Users mentioned in reference to Devoreann a

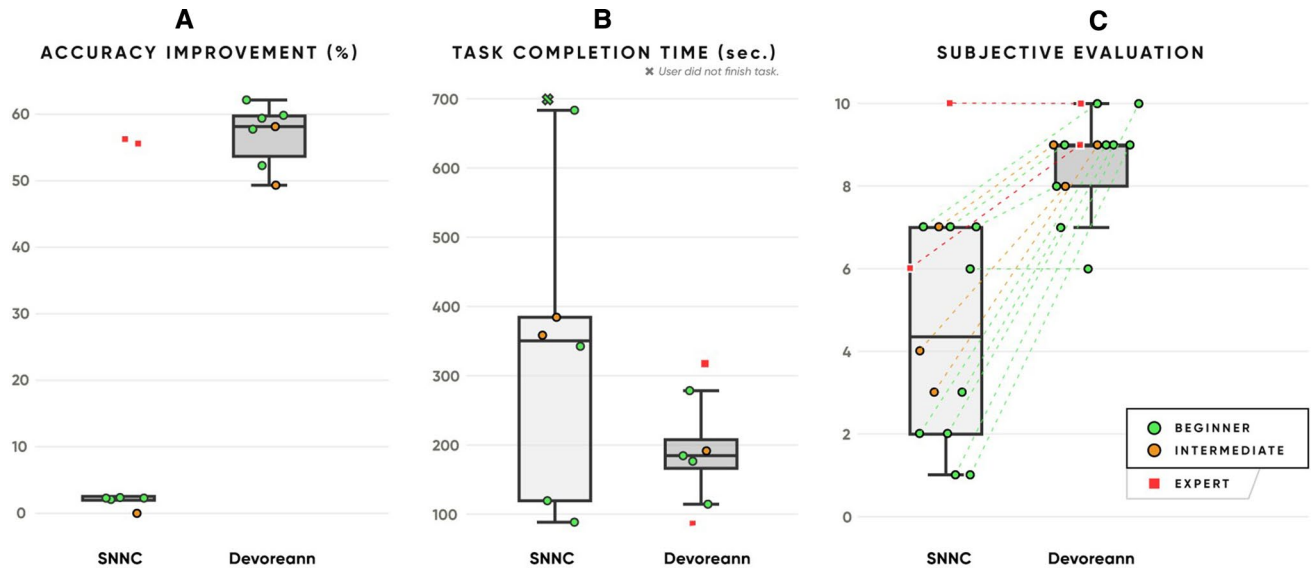

Fig. 5 Users built expert level accurate models in Task 1 using Devoreann but not using SNNC (**A**). They also corrected a faulty model faster in Task 2 using the proposed model (**B**). They enjoyed the proposed VR interface more than the alternative (**C**) where dashed lines indicate the scores from the same user. We include expert users to serve as a benchmark. The theoretical maximum for this dataset with an expert-tuned model achieves 89.8% accuracy (Veeling et al. 2018) (79.6% improvement from 50%)

"great feel in 3D," with an "easy to manipulate" interface which was "more intuitive and easier to make the connection with what [was] taking place." Four users wrote that the tool was "fun" or "engaging", and eight users wrote that Devoreann was "easier" and/or "more intuitive." Both experts included in the study indicated that our tool was "easy to use" but had limited ability to make very small or very extensive changes. One expert commented that "the visual aspect, including showing the actual images and the layer animations, was fun and informative" and "the whole process was incredibly streamlined, from building to training and evaluation."

4 Discussion

We developed an interactive VR visualization and development environment to aid in the democratization of AI by providing an accessible means of constructing and analyzing deep learning for biomedical image classification. We then showed with a user study that our proposed environment enables non-expert users in a medical field to build more accurate models and troubleshoot existing models faster, when compared to a state-of-the-art drag-and-drop alternative. We also found that users also enjoyed the experience in virtual reality significantly more. In summary, we found our tool to be effective for enabling non-experts to understand and organize the structure of DL models. We suggest that an application of this type could be directly applied to cross-domain applications in biomedical image classification and provide sufficient benefits in understanding and prototype development.

One of the primary reasons that users performed better in our proposed interface was, simply put, that our tool enabled the users to easily run Box's loop–build, compute, critique–until success. Experimentation with model construction was easier, more intuitive, and designed to avoid the logic errors that the alternative tool was prone to. The user feedback we received suggests this was mainly due to our design goals for creating a smooth, uncluttered, and intuitive interface. Both in Task 1 and Task 2, the breadth of features available in the SNNC indeed proved to be a limiting factor for ease-of-use in beginner and intermediate level users. This pattern also negatively affected task completion times for users in the alternative compared to those in our proposed interface.

We believe that the contributions of VR per se over the 2D alternative were a more enjoyable experience, fewer distractions, and increased immersion. Users subjectively reported enjoying VR more, which can partially explain the higher objective scores since positive emotions help with information retention (Hernik 2018; Um 2012). Furthermore, the high degree of immersion in virtual reality minimizes distractions, which can also partially explain the better scores, because learning performance suffers when distractions are present (Dietz and Henrich 2014; Zureick et al. 2018). We therefore argue that the increased immersion provided by VR and the unique method of interaction partially explains the higher scores. In summary, we argue that our interface achieved better outcomes

through intuitive affordances for user actions, immersion, and ease of use.

Virtual reality introduces multiple challenges into educational and professional applications including cost, design, and physical limitations. We designed the VR environment specifically to mitigate some of the risks of virtual reality including physical collision, motion sickness, sanitization, and the presence of wires (details in Methods). When asked about the disadvantages of such an interface, one user noted that the duration of use when wearing the head-mounted display could be difficult to some. Another noted that "components placed far away [forced the] user to move his/her head" extensively. In this regard, the introduction of virtual reality for development use is as challenging as it is beneficial. Topics of existing investigations into UX for VR include reducing physical side-effects and the presence, immersion, and engagement of the user (Kim et al. 2019).

Future improvements could drive this effort in two divergent directions: improving customization and analysis methods for intermediate users, and further increasing the explanatory benefit. To enable higher-complexity use of the proposed VR interface, some obvious improvements include adding more layer types, developing new UI elements for fine-tuning layer properties, and enabling nonlinear model designs. In addition, the front-end could be connected to an automated hyperparameter search engine, tuning model parameters more effectively and quickly than humans (Lisha et al. 2018; HIPS/Spearmint 2020). Finally, improved visualization techniques–such as gradient, guided backprop (Springenberg et al. 2015), or PatternNet (Kindermans et al. 2017)–could replace the activations on the intermediate layers. Both experts that participated in our study indicated a positive and engaging experience in VR. Due to this, along with benefits in immersion and interactivity, we suggest that a more comprehensive VR platform that could be beneficial for expert use in the future. To improve the demonstrational benefit and interpretability of this tool, one could add more explanatory elements, visualize data flow in 3D, and provide more tuning options to illustrate their effects to the user.

As with 2D, certain applications of deep learning in higher dimensions can benefit from 3D algorithmic tools. Many 2D visualization methods can be extended to 3D including saliency (Ma et al. 2017), heatmaps (Yang et al., 2018), and occlusion mapping (Reddy et al. 2019). Visualizations are especially pertinent for analyzing networks that need to operate on geometric or volumetric data. One such method is a 3D CNN that can recognize features that are difficult to manufacture in a complex 3D geometric model (Ghadai et al., 2018). Existing methods of analysis can also be projected into 3D such as the transfer of a saliency map to a volumetric dataset using 3D eye tracking (Ma et al. 2017). Although we did not integrate 3D visualizations due to accessibility and computational complexity, future work in 3D/4D data visualization can benefit from easier navigation and a more comprehensive visualization of data possible in VR.

We proposed a novel streamlined interface within virtual reality to create an effective introduction to the field of deep learning applied to biomedical image classification for non-expert medical professionals. In a user study, we found our approach to be effective and robust for building CNNs with ease. Users consistently reported a more enjoyable experience when using our tool over the alternative. Through an independent case study, we identified how an accessible introduction to DL can guide understanding aid in constructing domain-relevant models. We conclude that the application of DL in the biomedical field could benefit by integrating an easy-to-use interface with the interaction benefits of virtual reality. By providing researchers and healthcare professionals with the ability to prototype AI for classification of medical imaging, such a tool can improve collaboration efforts and enable immediate analysis of the viability of AI for a given task.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s10055-021-00550-1.

Acknowledgements The software is a derivative of work from the UT Southwestern hackathon, U-HACK Med 2018, and has continued development under the same Principal Investigator (Murat Can Çobanoğlu) and lead developer (Kevin VanHorn). The project was originally proposed by Murat Can Çobanoğlu, with preliminary draft code submitted to the NCBI-Hackathons GitHub under the MIT License. We thank hackathon contributors Meyer Zinn (*UT Southwestern Medical Center*), Xiaoxian Jing (*Southern Methodist University*), Siddharth Agarwal (*University of Texas Arlington*), and Michael Dannuzio (*University of Texas at Dallas*) for their initial work in design and development. We further thank Meyer Zinn for continued review of the manuscript and for his design of the initial RPC framework which was instrumental in the development of the work. We thank all our user study participants and thank the administration of the Lyda Hill Department of Bioinformatics for their patience and guidance.

Funding Lyda Hill Department of Bioinformatics startup funds awarded to M.C.C.

Code availability https://github.com/Cobanoglu-Lab/Devoreann.

Declarations

Conflicts of interest/Competing interests The authors declare that they have no conflict of interest.

References

Azure Machine Learning (2019) Microsoft Azure. Retrieved December 26, 2019 from https://azure.microsoft.com/en-us/services/machine-learning/

Bejnordi B, Mitko V, Paul Johannes van D, Bram van G, Nico K, Geert L, Jeroen A. W. M. van der L, Meyke H, Quirine FM, Maschenka B, Oscar G, Nikolaos S, Marcory CRF van D, Peter B, Francisco B, Andrew HB, Dayong W, Aditya K, Rishab G, Humayun I, Aoxiao Z, Qi D, Quanzheng L, Hao C, Huang-Jing

L, Pheng-Ann H, Christian H, Elia B, Quincy W, Ugur H, Mustafa Ü, Rengul C-A, Matt B, Vitali K, Alexei V, Oren K, Muhammad S, Nasir R, Ruqayya A, Korsuk S, Talha Q, Yee-Wah T, David T, Jonas A, Peter H, Mira V, Kimmo K, Leena L, Pekka R, Kaisa L, Shadi A, Bharti M, Ami G, Stefanie D, Nassir N, Seiryo W, Shigeto S, Yoichi T, Hideo M, Hady AP, Vassili K, Alexander K, Vitali L, Gloria B, Fernandez-Carrobles MM, Ismael S, Oscar D, Daniel R, Rui V (2017) Diagnostic assessment of deep learning algorithms for detection of lymph node metastases in women with breast cancer. JAMA 318(22) 2199–2210. https://doi.org/10.1001/jama.2017.14585

Blei DM (2014) Build, compute, critique, repeat: data analysis with latent variable models. Ann Rev Stat Appl 1(1):203–232. https://doi.org/10.1146/annurev-statistics-022513-115657

Bowman DA, McMahan RP (2007) Virtual reality: how much immersion is enough? Computer 40(7):36–43. https://doi.org/10.1109/MC.2007.257

Box George EP (1976) Science and statistics. J Am Stat Assoc 71(356):791–799. https://doi.org/10.2307/2286841

Cohen G, Afshar S, Tapson J, André van S (2017). EMNIST: an extension of MNIST to handwritten letters. arXiv:1702.05373 [cs] (February 2017). Retrieved December 27, 2019 from arXiv:1702.05373

ConvNetJS (2016) Deep Learning in your browser. Retrieved May 12, 2019 from https://cs.stanford.edu/people/karpathy/convnetjs/

Dede C (2009) Immersive interfaces for engagement and learning. Science 323:66–69. https://doi.org/10.1126/science.1167311

Deep Cognition (2017) DeepCognition.ai. Retrieved December 26, 2019 from https://deepcognition.ai/

Design (2020)Material design. Retrieved May 25, 2019 from https://material.io/design/

Dice LR (1945) Measures of the amount of ecologic association between species. Ecology 26:297–302. https://doi.org/10.2307/1932409

Dietz S, Henrich C (2014) Texting as a distraction to learning in college students. Comput Hum Behav 36:163–167. https://doi.org/10.1016/j.chb.2014.03.045

Esteva A, Kuprel B, Novoa RA, Ko J, Swetter Susan M, Blau Helen M, Thrun S (2017) Dermatologist-level classification of skin cancer with deep neural networks. Nature 542:115–118. https://doi.org/10.1038/nature21056

Everett NM (2018) Intuitive design, eight steps to an intuitive UI. Black Watch Publishing

Ghadai S, Balu A, Sarkar S, Krishnamurthy A (2018) Learning localized features in 3D CAD models for manufacturability analysis of drilled holes. Comput Aided Geom Des 62:263–275. https://doi.org/10.1016/j.cagd.2018.03.024

Harley AW (2015) An interactive node-link visualization of convolutional neural networks. in advances in visual computing. In: George B, Richard B, Bahram P, Darko K, Ioannis P, Rogerio F, Tim M, Mark E, Regis K, Eric R, Zhao Y, Gunther W (eds) Springer, Cham, pp 867–877. https://doi.org/10.1007/978-3-319-27857-5_77

Hernik J, Jaworska E (2018) The effect of enjoyment on learning, pp 508–514. https://doi.org/10.21125/inted.2018.1087

HIPS/Spearmint (2020) Harvard intelligent probabilistic systems group. Retrieved January 9, 2020 from https://github.com/HIPS/Spearmint

Kearnes S, Kevin M, Marc B, Vijay P, Patrick R (2016) Molecular graph convolutions: moving beyond fingerprints. J Comput Aided Mol Des 30(8):595–608. https://doi.org/10.1007/s10822-016-9938-8

Keras.js (2018) Run Keras models in the browser. Retrieved May 12, 2019 from https://transcranial.github.io/keras-js/#/

keras-team/keras (2019) Keras. Retrieved December 31, 2019 from https://github.com/keras-team/keras

Kim YM, Rhiu I, Yun MH (2019) A systematic review of a virtual reality system from the perspective of user experience. Int J Hum Comput Interact. https://doi.org/10.1080/10447318.2019.1699746

Kindermans P-J, Schütt KT, Alber M, Müller K-R, Erhan D, Kim B, Dühne S (2017) Learning how to explain neural networks: PatternNet and PatternAttribution. arXiv:1705.05598 [cs, stat] (May 2017). Retrieved May 11, 2019 from arXiv:1705.05598

Kingma DP, Ba J (2017) Adam: a method for stochastic optimization. arXiv:1412.6980 [cs] (January 2017). Retrieved January 8, 2020 from arXiv:1412.6980

Kolb DA (2014) Experiential learning: experience as the source of learning and development. FT Press

Le H, Dimitris S, Tahsin K, Gao Y, Davis James E, Saltz Joel H (2016) Patch-based convolutional neural network for whole slide tissue image classification. Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit 2016:2424–2433. https://doi.org/10.1109/CVPR.2016.266

Lecun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proce IEEE 86(11):2278–2324. https://doi.org/10.1109/5.726791

Lisha L, Kevin J, Giulia D, Afshin R, AmeetT (2018) A novel bandit-based approach to hyperparameter optimization, hyperband, p 52

Liu Y, Kohlberger T, Norouzi M, Dahl GE, Smith JL, Mohtashamian A, Olson N, Peng LH, Hipp JD, Stumpe MC (2018) Artificial intelligence–based breast cancer nodal metastasis detection: insights into the black box for pathologists. Arch Pathol Lab Med (October 2018), arpa.2018-0147-OA. https://doi.org/10.5858/arpa.2018-0147-OA

Lobe (2019) Deep learning made simple. Retrieved November 14, 2019 from https://lobe.ai

Ma B, Jain E, Entezari A (2017) 3D Saliency from eye tracking with tomography. In: Burch M, Chuang L, Fisher B, Schmidt A, Weiskopf D (eds) Eye tracking and visualization, mathematics and visualization. Springer, Cham, pp 185–198. https://doi.org/10.1007/978-3-319-47024-5_11

Matthews BW (1975) Comparison of the predicted and observed secondary structure of T4 phage lysozyme. Biochim Biophys Acta (BBA) Protein Struct 405:442–451. https://doi.org/10.1016/0005-2795(75)90109-9

Meissler N, Wohlan A, Hochgeschwender N, Schreiber A (2019) Using visualization of convolutional neural networks in virtual reality for machine learning newcomers. In: 2019 IEEE international conference on artificial intelligence and virtual reality (AIVR), pp 152–1526. https://doi.org/10.1109/AIVR46125.2019.00031

Miotto R, Wang F, Wang S, Jiang X, Dudley JT (2018) Deep learning for healthcare: review, opportunities and challenges. Brief Bioinform 19(6):1236–1246. https://doi.org/10.1093/bib/bbx044

Naumann A, Hurtienne J, Habakuk Israel J, Mohs C,Christof Kindsmüller M, Meyer HA, Hu S ßlein (2007) Intuitive use of user interfaces: defining a vague concept. In: Engineering psychology and cognitive ergonomics (lecture notes in computer science). Springer, Berlin, pp 128–136. https://doi.org/10.1007/978-3-540-73331-7_14

Netzer Y, Wang T, Coates A, Bissacco A, Wu B, Ng AY (2011) Reading digits in natural images with unsupervised feature learning. In: NIPS workshop on deep learning and unsupervised feature learning 2011. Retrieved August 11, 2020 from http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf

Neural Network Console (2020) Retrieved December 26, 2019 from https://dl.sony.com/

Neural network modeler (2019) IBM Watson. Retrieved December 26, 2019 from https://dataplatform.cloud.ibm.com/docs/content/wsj/analyze-data/ml-canvas-nnd-nodes.html

NVIDIA DIGITS (2015) NVIDIA Developer. Retrieved December 26, 2019 from https://developer.nvidia.com/digits

Obuchowski NA, Beiden SV, Berbaum KS, Hillis SL, Ishwaran H, Song HH, Wagner RF (2004) Multireader, multicase receiver operating characteristic analysis: an empirical comparison of five methods1. Acad Radiol 11:980–995. https://doi.org/10.1016/j.acra.2004.04.014

Ounkomol C, Seshamani S, Maleckar MM, Collman F, Johnson GR (2018) Label-free prediction of three-dimensional fluorescence images from transmitted-light microscopy. Nat Methods 15(11):917. https://doi.org/10.1038/s41592-018-0111-2

Reddy ND, Vo M, Narasimhan SG (2019) Occlusion-net: 2D/3D occluded keypoint localization using graph networks. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), 2019, pp 7326–7335

Sacks R, Perlman A, Barak R (2013) Construction safety training using immersive virtual reality. Constr Manag Econ 31:9. https://doi.org/10.1080/01446193.2013.828844

Schreiber A, Bock M (2019) Visualization and exploration of deep learning Networks in 3D and virtual reality. In HCI international (2019) posters (communications in computer and information science). Springer, Cham 206–211. https://doi.org/10.1007/978-3-030-23528-4_29

Smilkov D, Carter S (2016) Tensorflow—neural network playground. http://playground.tensorflow.org

Sørensen TJ (1948) A method of establishing groups of equal amplitude in plant sociology based on similarity of species content and its application to analyses of the vegetation on Danish commons. I kommission hos E, Munksgaard, København

Springenberg JT, Dosovitskiy A, Brox T, Riedmiller M (2015) Striving for simplicity: the all convolutional net. arXiv:1412.6806 [cs] (April 2015). Retrieved January 9, 2020 from arXiv:1412.6806

TensorSpace.js (2019) Retrieved May 11, 2019 from https://tensorspace.org/index.html

Um ER, Plass JL, Hayward EO, Homer BD (2012) Emotional design in multimedia learning. J Educ Psychol 104(2):485-498. https://doi.org/10.1037/a0026609

Vardhana M, Arunkumar N, Lasrado S, Abdulhay E, Ramirez-Gonzalez G (2018) Convolutional neural network for bio-medical image segmentation with hardware acceleration. Cognit Syst Res 50:10–14. https://doi.org/10.1016/j.cogsys.2018.03.005

Veeling B (2019) The PatchCamelyon (PCam) deep learning classification benchmark.: basveeling/pcam. Retrieved May 31, 2019 from https://github.com/basveeling/pcam

Veeling BS, Linmans J, Winkens J, Cohen T, Welling M (2018) Rotation Equivariant CNNs for Digital Pathology. arXiv:1806.03962 [cs, stat] (June 2018). Retrieved January 7, 2020 from arXiv:1806.03962

Xiao H, Rasul K, Vollgraf R (2017) Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. arXiv:1708.07747 [cs, stat] (September 2017). Retrieved December 27, 2019 from arXiv:1708.07747

Yang C, Rangarajan A, Ranka S (2018) Visual explanations from deep 3D convolutional neural networks for Alzheimer's disease classification. AMIA Annu Symp Proc 2018:1571–1580

Zureick AH, Burk-Rafel J, Purkiss JA, Hortsch M (2018) The interrupted learner: How distractions during live and video lectures influence learning outcomes. Anat Sci Educ 11(4):366–376. https://doi.org/10.1002/ase.1754