# CNN 101: Interactive Visual Learning for Convolutional Neural Networks

**Zijie J. Wang**
Georgia Institute of Technology
jayw@gatech.edu

**Nilaksh Das**
Georgia Institute of Technology
nilakshdas@gatech.edu

**Robert Turko**
Georgia Institute of Technology
rturko3@gatech.edu

**Fred Hohman**
Georgia Institute of Technology
fredhohman@gatech.edu

**Omar Shaikh**
Georgia Institute of Technology
oshaikh@gatech.edu

**Minsuk Kahng**
Oregon State University
minsuk.kahng@oregonstate.edu

**Haekyu Park**
Georgia Institute of Technology
haekyu@gatech.edu

**Duen Horng (Polo) Chau**
Georgia Institute of Technology
polo@gatech.edu

## Abstract

The success of deep learning solving previously-thought hard problems has inspired many non-experts to learn and understand this exciting technology. However, it is often challenging for learners to take the first steps due to the complexity of deep learning models. We present our on-going work, CNN 101, an interactive visualization system for explaining and teaching convolutional neural networks. Through tightly integrated interactive views, CNN 101 offers both overview and detailed descriptions of how a model works. Built using modern web technologies, CNN 101 runs locally in users' web browsers without requiring specialized hardware, broadening the public's education access to modern deep learning techniques.

## Author Keywords

Interactive visualization; deep learning education; machine learning education.

## CCS Concepts

•**Human-centered computing** → **Visual analytics;** •**Computing methodologies** → **Machine learning;**

## Introduction

Deep learning has become a driving-force in our daily technologies. Its continued success and potential in various hard problems have attracted immense interest from non-

experts to learn this technology. However, it has a steep learning curve for many beginners.

Since deep learning models are complex, it can be challenging for non-experts to learn the fundamentals. Inspired by human's brain structure, deep neural network models typically leverage many layers of operations to reach a final computed decision [10]. There are many types of network layers, each having a different structure and underlying mathematical operations. Therefore, understanding deep learning models requires users to keep track of both *low-level mathematical operations* and *high-level integration* of such operations within the network.

To address this challenge, we are developing **CNN 101** (Figure 1): an interactive visualization system that helps students learn *convolutional neural networks* (CNN), a foundational deep learning model architecture [10], more easily. CNN 101 joins the growing body of research that aims to explain the complex mechanisms of modern machine learning algorithms with interactive visualization, such as TensorFlow Playground [12] and GAN Lab [7]. For a demo video of CNN 101, visit https://youtu.be/g082-zitM7s. In this ongoing work, our primary contributions are:

1. **CNN 101**, a novel web-based interactive visualization tool that helps users better understand both CNNs high-level model structure and low-level mathematical operations. Advancing on few existing and prior interactive visualization tools that aim to explain CNN to beginners [4, 8], CNN 101 integrates a more practical model and dataset for learners to explore. Conventionally, deploying deep learning models requires significant computing resources, e.g., servers with powerful hardware. However, even with a dedicated backend server, it is challenging to support a large number of concurrent users. Instead, CNN 101 is developed using modern web technologies, where all results are directly computed in users' web browsers. CNN 101 helps broaden public's education access to modern deep learning technologies.

2. **Novel interactive visualization design of CNN 101**, which uses *overview + detail*, *interaction*, and *animation* that simultaneously summarizes complex model structure, and provides context for users to interpret detailed mathematical operations. CNN 101 presents significant advancement over existing work by explaining how CNNs work at different abstraction levels while helping users fluidly transition between such levels to gain a more comprehensive understanding. Existing work focused on fewer aspects. For example, Harley et al. [4] used a 3D interactive node-link diagram to illustrate CNN structure and neuron activations of a pre-trained model, but the interface did not visually dissect different neuron's computation processes. Conversely, expert-facing deep learning visualization tools focus on interpreting what CNN models have learned rather than explaining the underlying operations [5].

We hope our design will inspire research and development of interactive education tools that help democratize more artificial intelligent technologies.

## System Design and Implementation
CNN 101 is an interactive system for illustrating how a trained CNN model classifies an image (Figure 1). It enables users to explore the CNN structure and underlying operations in a browser. To elucidate the CNN's complex process of classifying images, CNN 101 consists of three views: (1) **Overview** (Figure 1A) shows the big picture of the CNN, describing how the input image is connected to the classification likelihood through different layers;

The *Overview* Ⓐ visualizes activation maps of all neurons as heatmaps connected with edges.

When user clicks a convolutional neuron in Ⓐ, the view transitions to the *Convolutional Intermediate View* (Ⓐ → Ⓑ).

The *Flatten Intermediate View* appears when an output neuron is selected instead (Ⓐ → Ⓒ).

Ⓑ demonstrates the relationship between selected convolutional neuron and its previous layer.

Ⓑ transitions to the *Detail View* which illustrates the convolution operation on selected input neuron (Ⓑ → Ⓓ).

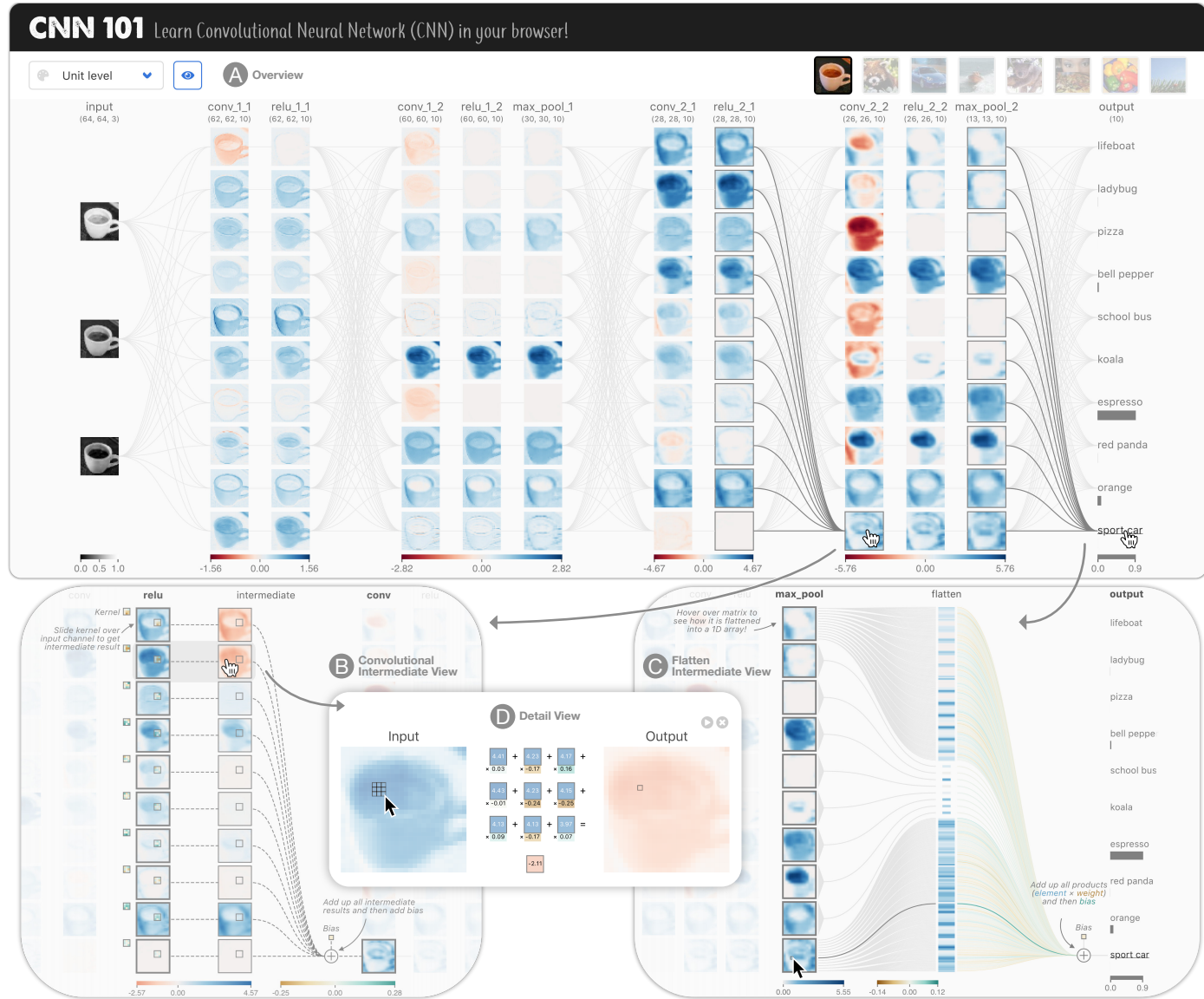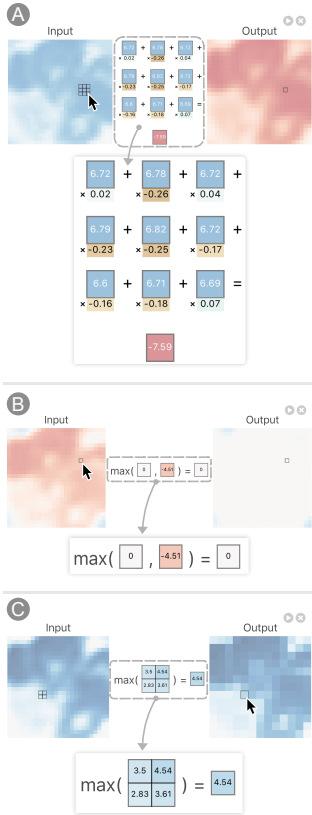Ⓒ explains the flatten layer between the second last layer and output layer.



**Figure 1:** The CNN 101 user interface and its tightly coupled, multiple views.

**Figure 2:** The Detail View explains the underlying mathematical operations of a CNN. Ⓐ shows the element-wise dot-product occurring in a convolutional neuron. Ⓑ visualizes the Rectified Linear Unit (ReLU) operation, whereas Ⓒ illustrates how max pooling works. Users can hover over heatmaps to display an operation's input to output mappings.

(2) **Intermediate View** (Figure 1B-C) dissects the relationship between one neuron and its previous layer; (3) **Detail View** (Figure 1D) interactively visualizes the inner workings of different CNN operations. Transitions of these views follow an *overview-to-detail* order and are animated to help users assimilate the relationship between different states.

**Overview.** This view is the starting view of CNN 101 (Figure 1A). It shows activation heatmaps of neurons in all layers. Neurons in consecutive layers are connected with edges, and hovering over one neuron highlights its incoming edges. Convolutional and output neurons connect to all neurons in the previous layer, whereas other neurons connect to only one neuron from the earlier layer.

We show heatmaps with a symmetric diverging red-to-blue colormap where zero is encoded as white. For example, darker red pixels indicate smaller negative values while darker blue pixels indicate larger positive values. We group our CNN layers into four units and two modules (Figure 3). Each unit has at most one convolutional layer. The last two units (Module 1) are duplicate of the first two units (Module 2). Users can change the heatmap colormap scope based on defined layer groups. This option enables users to compare neuron activations in different levels and contexts.
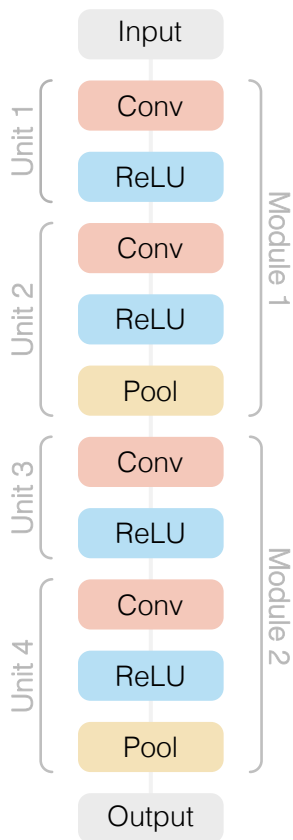
**Intermediate View.** CNN 101 has two types of Intermediate Views: the Convolutional Intermediate View and the Flatten Intermediate View. When users click a convolutional neuron in the Overview, the Convolutional Intermediate View (Figure 1B) applies a convolution on each input node of the selected neuron. Then, it displays these intermediate results as heatmaps. This view also visualizes associated convolution kernel weights as small heatmaps, which slide over input and intermediate result heatmaps. This animation mimics the CNN's internal sliding window. In addition, Edges in the Convolutional Intermediate View are animated

as flowing dash-lines, which help signify the order and direction of this intermediate operation.

The Flatten Intermediate View (Figure 1C) explains a flatten layer, which is often used in a CNN to reshape the second last layer into a dense layer, so the fully connected output layer can make classification decisions. This view encodes each flatten layer neuron as a short line, with the same color as its source element (pixel) in previous layer. Also, each short line is connected to its source and intermediate result with edges, whose color further encodes model weight value. When users hovers over an element in the source heatmap, its associated short line and edges are highlighted.

**Detail View.** This view has three variants designed for convolutional (Figure 2A), activation (Figure 2B), and pooling layers (Figure 2C), respectively. The Detail View provides the user with a low-level, interactive analysis of the mathematical operations occurring at each layer. Users not only can observe each operation run on an interval displayed by a sliding input region, but also directly interact with Detail View by hovering over pixels to visualize the operation on the selected input region to yield the resulting output values. By providing a straightforward and interactive visualization of the input and output of multiple fundamental CNN operations, the Detail View allows users who are unfamiliar with CNN mechanisms to understand its mathematical intricacies.

Moreover, with CNN 101's *overview-to-detail* transition hierarchy and *focus + context* layout, users can learn how each single low-level operation contributes to the high-level CNN flow. For example, a particular convolution output, explained in the Detail View, is only an intermediate result. To compute the output of a convolutional neuron, one needs to add up all these intermediate results with bias, as de-

Input

Unit 1
Conv
ReLU

Unit 2
Conv
ReLU
Pool

Module 1

Unit 3
Conv
ReLU

Unit 4
Conv
ReLU
Pool

Module 2

Output

**Figure 3:** Tiny VGG has the same 3×3 convolution and 2×2 pooling layers as in VGGNet, but with a smaller network depth. Since Tiny VGG layers are grouped into units and modules, CNN 101 allows users to control how each unit and module is colored in the Overview heatmap.

scribed in the Overview and the Intermediate View. Advancing over existing tools, CNN 101's hierarchical design builds up user's mental model to understand this connection.

**CNN Model Training.** Inspired by the popular deep learning architecture VGGNet [11] and Stanford's lecture notes on CNN [9], we train a Tiny VGG on Tiny ImageNet dataset [1] for demonstration purpose. Tiny ImageNet has 200 image classes, a training dataset of 100,000 64×64 color images, and a validation/test dataset of 10,000 images each. Our model is trained using *TensorFlow* [2] on images from 10 selected everyday classes (lifeboat, ladybug, pizza, bell pepper, school bus, koala, espresso, red panda, orange, and sport car) with batch size and learning rate fine-tuned using a 5-fold cross-validation scheme; it achieves a 70.8% top-1 accuracy on the test dataset.

**Front-end Visualization.** We use *TensorFlow.js* [13] to load our trained Tiny VGG and compute forward propagation results directly in user's browser. We use *D3.js* [3] to visualize the network structure and implement interactions and animations. Our implementation is robust, so the this visualization prototpye can be quickly applied to other dataset and linear CNN models.

## Preliminary Results: Usage Scenarios

We now present two usage scenarios where CNN 101 assists users to learn CNN process and gain develop learning intuitions.

**Understanding layer relationship through visualizing intermediate operations.** An undergraduate student Sally is learning about various types of CNN layers in her introductory machine learning course. She does not fully understand how the final output layer maps previous 2D matrices into a class probability number. Sally starts investigating Tiny VGG with CNN 101 by inspect-

ing layer dimensions. She quickly noticed that the output layer has dimension 10, while its previous layer has dimension 13×13×10. Sally hovers over the output class `sport car` and sees its incoming edges from the previous layer. CNN 101 helps her quickly recognize essential basic information that there are 10 image classes, 10 neurons in the `max_pool_2` layer, and that each output class connects to 10 previous neurons. Then, Sally clicks on `sport car`, causing the *Overview* to transition to the *Flatten Intermediate View*, displaying the flatten layer between the `max_pool_2` layer and the `sport car` class label (Figure 1C). By hovering over the heatmap in the `max_pool_2` layer, Sally sees the highlighted edges connecting each matrix element first to the flatten layer, and then to the `sport car` class label. Through CNN 101's interactive visualization, Sally realizes that the illustrations in most deep learning tutorials have in fact been skipping this important information in CNN, i.e., that: (1) there is actually a "hidden" layer that unrolls the output of the `max_pool_2` layer into a 1D array, and (2) output layer connects to an intermediate flatten layer instead of directly to the `max_pool_2` layer.

**Learning layer operation in multiple abstraction levels.** Harry is a biology researcher who has learned about CNNs in an online deep learning course. Since he plans to train a CNN model for his project, he uses CNN 101 to review the inner workings of different CNN layers. Harry launches CNN 101 and skims through all layers on the *Overview*. He has forgotten what exactly the ReLU layer does upon reaching it in the interface. However, CNN 101 immediately helps him notice that all previous heatmap red pixels disappear in ReLU layers (Figure 1A). After selecting other input images and having the same observation, Harry guesses that ReLU layers ignore negative values and only propagate positive values. He clicks on a ReLU neruon, which causes the *Overview* to transition to the *Detail View*. See-

ing ReLU's underlying equation, $\max(0, x)$, revealed on the *Detail View* (Figure 2B), Harry is very happy that his hypothesis is validated. By offering both an overview and a detailed explanation of the ReLU activation function, CNN 101 helps Harry understand how ReLU layer works in different abstraction levels.

## Ongoing Work and Conclusion

**User customization.** We are working on extending CNN 101's interactivity to promote user engagement and to explain more CNN concepts. Besides choosing an input image from Tiny ImageNet, we plan to support users to upload their own images, capture images from webcam, and free form drawing. These options can enable users to engineer images to test their hypothesis regarding CNN operations during learning [6]. For example, if one user is confused about how the convolution operation works on multiple channels, she can create an image that only has non-zero values in the red channel and feed it into CNN 101. Then she can learn that convolutions are performed independently on input channels, by observing the intermediate results and activation maps on the first convolution layer.

Currently CNN 101 explains convolution, activation, and pooling operations at single-neuron-level as well as layer-level. However, these operations have fixed hyper-parameters. For example, the convolution process always uses a $3\times3$ kernel with padding of 0 and stride of 1. We are working to support users to configure these settings and observe the results in real time. Such interactive "hypothesis testing" and experimentation help users learn other advanced deep learning architectures more easily.

**Planned evaluation.** Despite the increasing popularity of applying interactive visualization to teach deep learn-

ing concepts, little work have been done to evaluate how effective these tools are [6]. We plan to run a user study to compare the educational effectiveness of CNN 101 and that of conventional educational mediums such as (static) tutorials, textbooks, and YouTube lecture videos. We plan to recruit undergraduate students who have basic machine learning background and are new to deep learning. Our study will have two conditions: *CNN 101* v.s. *conventional tools*. We will randomly assign students into these conditions, and they will use respective tools to learn how CNN works. Each participant will complete a pre-test quiz and a post-test quiz, allowing us to quantify and more deeply understand the education effectiveness of CNN 101.

**Deployment.** We are working to deploy and open-source CNN 101, similar to TensorFlow Playground [12] and GAN Lab [7], so that it will be easily accessible by learners from all over the world.

**Conclusion.** CNN 101 takes steps toward democratizing deep learning that has been closely impacting people's daily lives. Through applying interactive visualizing techniques, CNN 101 provides users with an easier way to learn deep learning mechanisms and build up neural network intuitions. We plan to extend CNN 101's capabilities to support further user customization and personalized learning; we will deploy and open-source CNN 101 and also evaluate it in depth to help build design principles for future deep learning educational tools.

## Acknowledgements

## REFERENCES

[1] 2015. Tiny ImageNet Visual Recognition Challenge. (2015). https://tiny-imagenet.herokuapp.com

[2] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. TensorFlow: A System for Large-Scale Machine Learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*. USENIX Association, Savannah, GA, 265–283.

[3] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. 2011. D³ Data-Driven Documents. *IEEE Transactions on Visualization and Computer Graphics* 17, 12 (Dec. 2011), 2301–2309.

[4] Adam W. Harley. 2015. An Interactive Node-Link Visualization of Convolutional Neural Networks. In *Advances in Visual Computing*. Vol. 9474. Springer International Publishing, Cham, 867–877.

[5] Fred Hohman, Minsuk Kahng, Robert Pienta, and Duen Horng Chau. 2019. Visual Analytics in Deep Learning: An Interrogative Survey for the Next Frontiers. *IEEE Transactions on Visualization and Computer Graphics* 25, 8 (Aug. 2019), 2674–2693.

[6] Minsuk Kahng and Duen Horng Chau. 2019. How Does Visualization Help People Learn Deep Learning? Evaluation of GAN Lab. In *Workshop on EValuation of Interactive VisuAl Machine Learning systems*.

[7] Minsuk Kahng, Nikhil Thorat, Duen Horng Chau, Fernanda B. Viegas, and Martin Wattenberg. 2019.

GAN Lab: Understanding Complex Deep Generative Models using Interactive Visual Experimentation. *IEEE Transactions on Visualization and Computer Graphics* 25, 1 (Jan. 2019), 310–320.

[8] Andrej Karpathy. 2016a. ConvNetJS MNIST demo. (2016). https://cs.stanford.edu/people/karpathy/convnetjs/demo/mnist.html

[9] Andrej Karpathy. 2016b. CS231n Convolutional Neural Networks for Visual Recognition. (2016). http://cs231n.github.io/convolutional-networks/

[10] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521, 7553 (May 2015), 436–444.

[11] Karen Simonyan and Andrew Zisserman. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv:1409.1556 [cs]* (April 2015). arXiv: 1409.1556.

[12] Daniel Smilkov, Shan Carter, D. Sculley, Fernanda B. Viégas, and Martin Wattenberg. 2017. Direct-Manipulation Visualization of Deep Networks. *arXiv:1708.03788 [cs, stat]* (Aug. 2017). arXiv: 1708.03788.

[13] Daniel Smilkov, Nikhil Thorat, Yannick Assogba, Ann Yuan, Nick Kreeger, Ping Yu, Kangyi Zhang, Shanqing Cai, Eric Nielsen, David Soergel, Stan Bileschi, Michael Terry, Charles Nicholson, Sandeep N. Gupta, Sarah Sirajuddin, D. Sculley, Rajat Monga, Greg Corrado, Fernanda B. Viégas, and Martin Wattenberg. 2019. TensorFlow.js: Machine Learning for the Web and Beyond. *arXiv:1901.05350 [cs]* (Feb. 2019). arXiv: 1901.05350.