A Deep Learning Graphical User Interface Application on MATLAB

Dr. Aditya Akundi, University of Texas, El Paso

Aditya Akundi is currently affiliated to Industrial Manufacturing and Systems Engineering department, and Research Institute for Manufacturing and Engineering Systems at University of Texas, ElPaso.

He earned a Bachelor of Technology in Electronics and Communication Engineering from Jawaharlal Nehru Technological University, India. He earned a Master of Science in Electrical and Computer Engineering at the University of Texas at El Paso (UTEP). Intrigued by Systems Engineering , he earned a Ph.D in Electrical and Computer Engineering, with a concentration in Industrial and Systems Engineering (ISE) at Unniversity of Texas in 2016. His research is focused on undersanding Complex Technical and Socio-Technical Systems from an Infromation Theortic approach. He has worked on a number of projects in the field of Electrical & Computer Engineering, Systems Engineering, Additive Manufacturing and Green Energy Manufacturing. His research interests are in Systems Engineering & Architecture, Complex systems, Systems testing and Application of Entropy to Complex Systems.

Prof. Tzu-Liang Bill Tseng, University of Texas, El Paso

Dr. Tseng is a Professor and Chair of Industrial, Manufacturing and Systems Engineering at UTEP. His research focuses on the computational intelligence, data mining, bio- informatics and advanced manufacturing. Dr. Tseng published in many refereed journals such as IEEE Transactions, IIE Transaction, Journal of Manufacturing Systems and others. He has been serving as a principle investigator of many research projects, funded by NSF, NASA, DoEd, KSEF and LMC. He is currently serving as an editor of Journal of Computer Standards & Interfaces.

Zejing Cao, The University of Texas at El Paso
Mr. Hoejin Kim, University of Texas, El Paso

Mr. Hoejin Kim is currently a Ph.D. student in Department of Mechanical Engineering at the University of Texas at El Paso. He received a bachelor degree in materials engineering technology at Korea Tech in 2008 and a master degree in manufacturing engineering technology at Oregon Institute of Technology in 2014. His research interests are focused on 3D printing of piezo-, pyro-, and dielectric materials for pressure and temperature sensors and quality control of 3D printed product using big data mining.

A Deep Leaning Graphical User Interface Application on MATLAB

Abstract

Deep learning, as a new era of machine learning techniques, shows the ability to generalize computational model through hierarchical layers by learning the features from a large amount of training data with ignorable human interventions. Deep learning is mainly implemented in areas of computer vision, audio, and speech processing. Moreover, deep learning has significantly improved state of the art in image classification. Image-based classification, the most classic but significant research topic, has been critically improved using deep learning architecture and outperformed the majority of state-of-the-art achievements. No bells and whistles, deep learning becomes the most popular research area. However, to train a comprehensive classifier, a large amount of data is inescapable, which apparently needs lots of computational time so that it requires fast GPUs to accomplish fast training. In general, powerful GPUs are expensive, and the process of solving compatibility problems in both hardware and software are reluctant due to very limited access to GPUs' implementations in most universities. The primary purpose of this paper is how to propagate principles of deep learning to students and build the bridge to make them learn and use deep learning more comfortable or more accessible. In this paper, we develop a Graphic User Interface (GUI) deep learning beginners to test classification results through CPU without massive training computation. We use the high-performance GPU to train the model, once finish the training, we save the parameters of training model, load it to the GUI and let the student do the testing.

Keywords

Deep learning, GUI, Image classification, MNIST.

Introduction and Background

Deep learning is a learning algorithm based on learning data representations. It also called deep structured learning. Deep learning is a part of machine learning family, but it is in a specific machine learning category. Compare to the traditional machine learning methodology, due to the architectures of deep networks, deep learning approaches need a significant amount of data for training. Fortunately, by training enormous amount of data, the deep learning network learns the predicted model of this data, and usually, this predicted model contains high representations of this data and gives users high accuracy to solve the classification problem. Nowadays, deep learning powers numerous parts of current society, such as web searching, recommendations from e-commerce websites, autonomous car and many applications that involve computer vision[1]. Deep learning algorithms work exceptionally well on the image classification problems, for both simple and complicated images. Notably, the deep convolutional neural networks work excellent on image classification. In this paper, we introduce a deep learning graphical user interface application based on the deep convolutional neural network.

The deep convolutional neural network has been in the core position of deep learning domain. Although convolutional neural network used as an image classification technique for simple digits or character recognition task in earlier time. However, due to the success of recent work,

[2] use a deep convolutional neural network to beat state-of-art in the ImageNet challenge, the deep convolutional neural network becomes the common and useful tool for image classification problems[3].

The convolutional neural network is similar to the traditional neural network; it also has weights and bias in each neuron. However, the convolutional neural network is more focus on complicated inputs, such as images. Convolutional neural networks are structured by three essential ideas, local receptive fields, shared weights and subsampling. A typical convolutional neural network contains multiple hidden layers, such as a convolutional layer, pooling layer, and fully connected layer[4].

A neural network is based on affine transformations, the input vector is multiplied by weights vector to produce an output. When the input is an image, depends on the type of image, it can be seen as a single signal channel multi-dimensional array or a three-channel multi-dimensional array(a color image consisting of three channels, the red, green and blue). A convolutional layer performs a linear transformation. In this process, only a couple of input information contribute to an output unit, the shared weights are applied to different locations in the input[3]. Figure 2 shows an example of convolution computation with a 3×3 kernel.

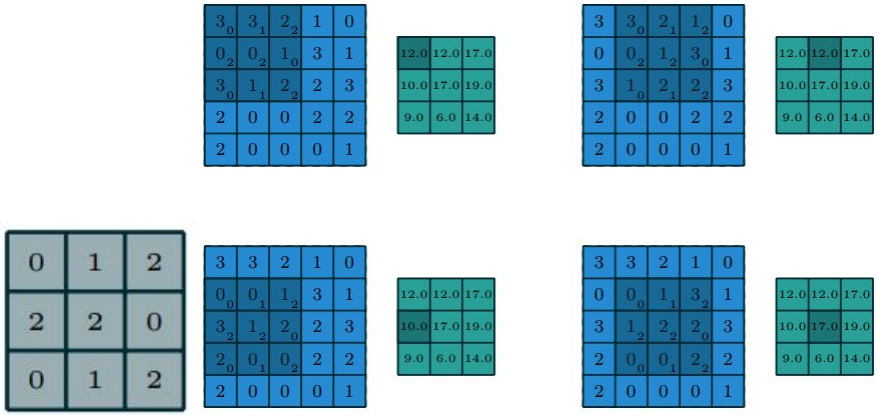

Figure 1: Compute the output values of convolution[3]

In a conventional neural network, pooling operations decrease the size of feature maps by utilizing some functions. Generally, there are two types of pooling, max pooling, and average pooling (also have other advanced poolings). Max pooling means to take the max pixel value of one specific region. On the other hand, average pooling means to take the average of one particular region. Figure 3 shows an example of max pooling.

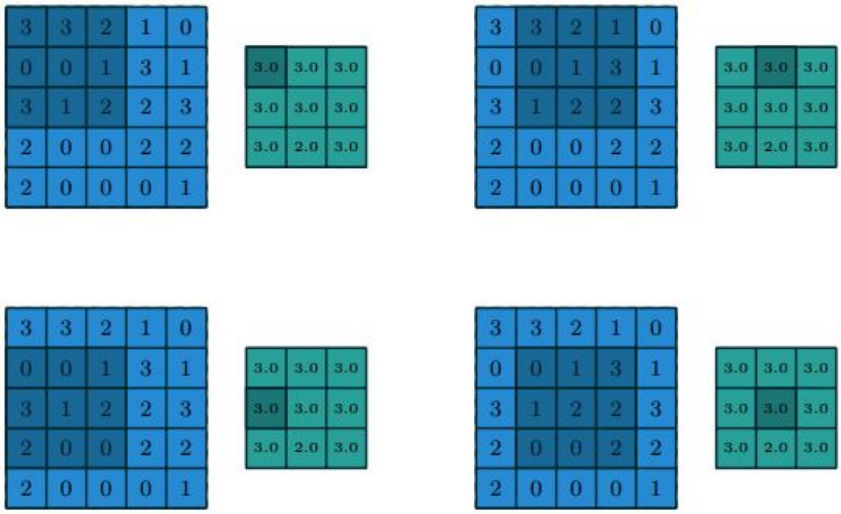

Figure 2: Max pooling[3]

Methodology

In order to help students to understand the deep learning concept easier, we design this graphic user interface(GUI) based on the Modified National Institute of Standards and Technology database(MNIST). The MNIST dataset contains 28×28 pixel images of ten different categories, from digit 0 to 9. It consists of 60,000 training and 10,000 testing samples[5]. Figure 3 shows a sample of MNIST dataset.

Figure 3: A sample of MNIST dataset

Convolutional neural network(CNN), as one of the most popular deep learning architectures, is the only deep learning algorithm we use without the necessity of unsupervised pre-training. Namely, we can directly feed our raw data into the network. Because the neural network with several full-connected layers is not practical for training when initialized randomly, the CNN structure we develop in this study is based on LeCun's model: a fully-connected layer followed

by several convolutional layers and subsampling layers, and each layer has a topographic structure.

The algorithm begins by extracting random sub-patches from the ROIs mentioned above, with the size of the sub-patches referred to as the "receptive field size." In each layer, the neuron is associated with a fixed two-dimensional position, and its receptive field is corresponding to the input from previous layer (the first layer is corresponding to the original input image).

Several neurons are connected to the same location at every layer, and each neuron is a linear combination of its corresponding neurons in the previous layer with its set of input weights. The neurons at different locations are associated with the same set of weights, but different corresponding input patches.

The architecture of this CNN contains two convolution layers, two max-pooling layers, and one fully-connected layer is showed in figure 4.

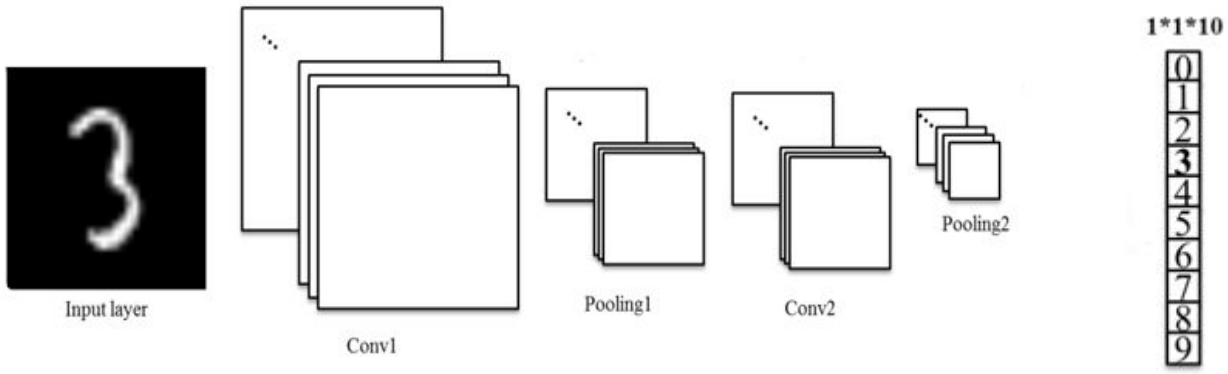

Figure 4: The architecture of the CNN.

The GUI contains two parts, the training GUI, and the testing GUI. Due to the training process took much longer to run, separate them would be more comfortable for students. Figure 5 and 6 show these two GUIs. For the training GUI, the student can type the name of the model(classifier), and save the model. Epoch number, learning rate, and batch size are also adjustable. The loss function can express the performance of the training.

The testing GUI is more sample than the training GUI. The student can type the order number of the testing dataset, for this example, we have 10,000 testing images, test from first to 10,000th images. Prediction bar shows the predicating valve that predicted by the model; the actual input bar shows the input value, from 0 to 9.

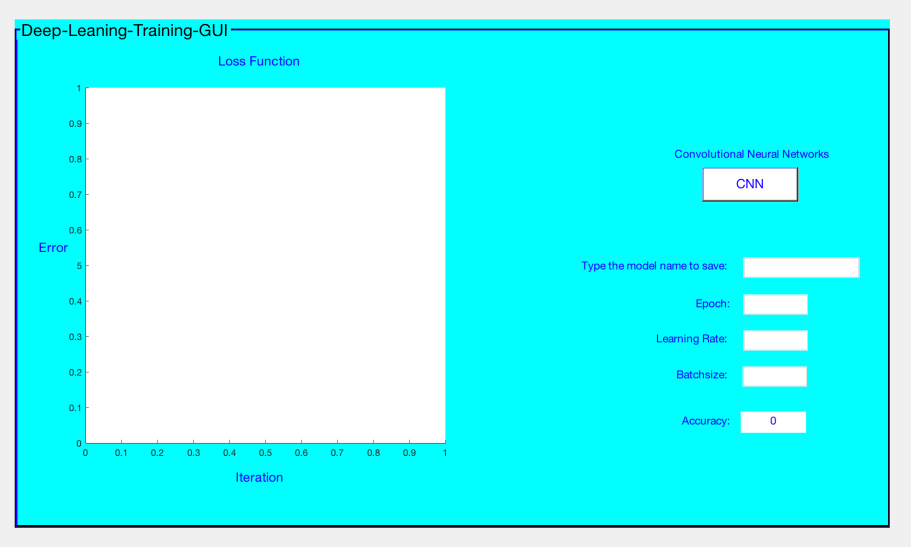

Figure 5: The training GUI.

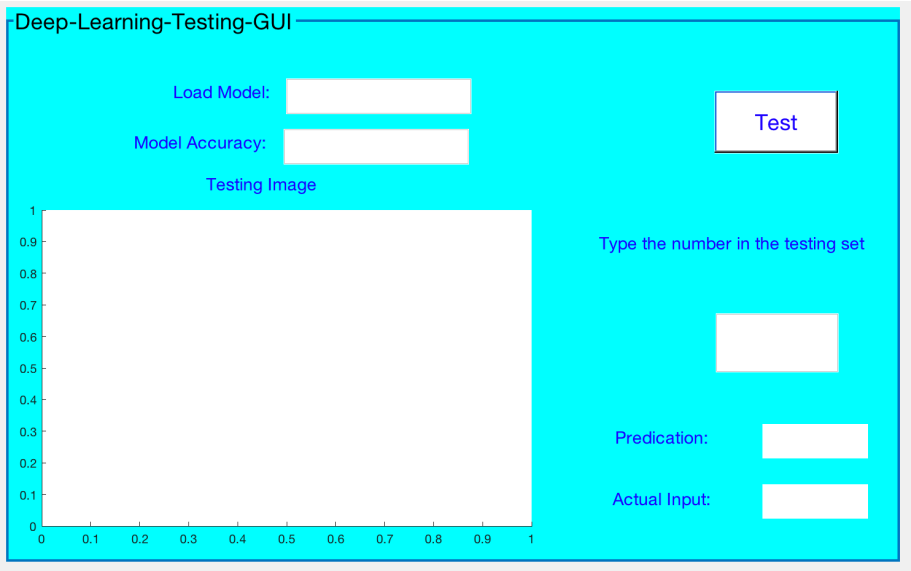

Figure 6: The testing GUI.

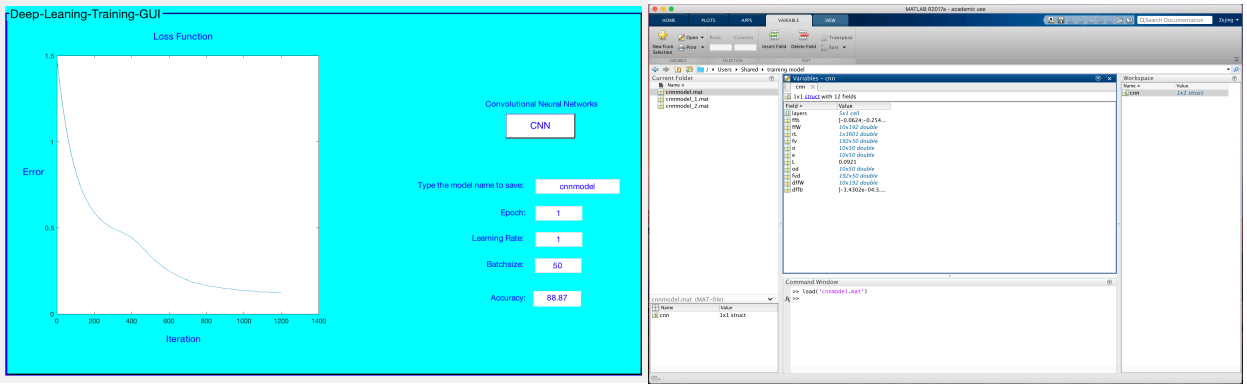

Figure 7: Training the CNN, and the saved model named cnnmodel.

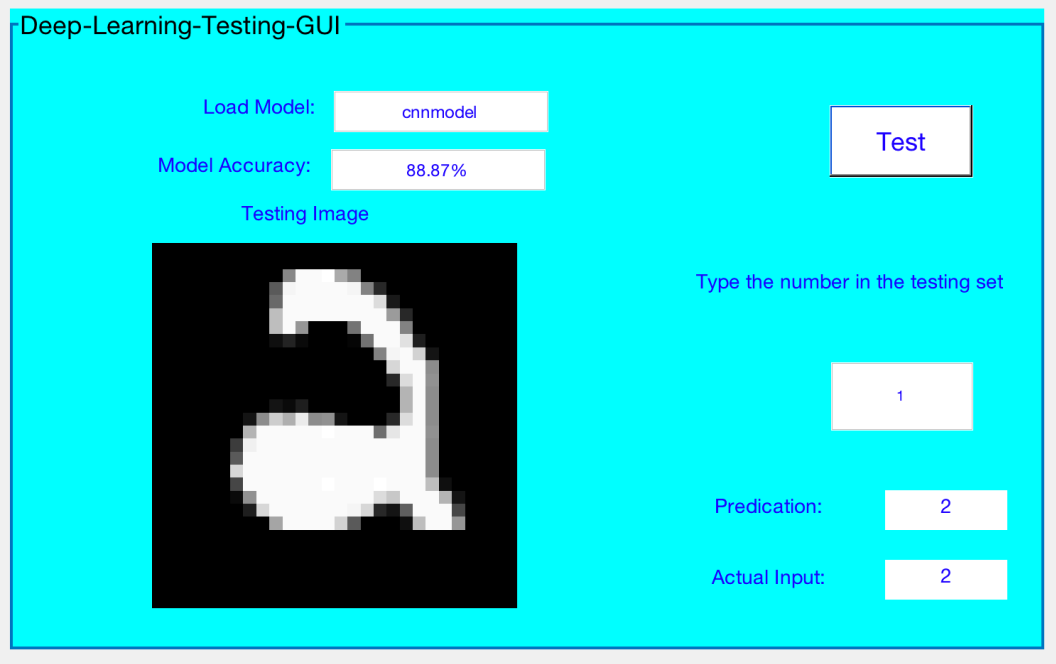

Figure 8: The testing result based on the cnnmodel.

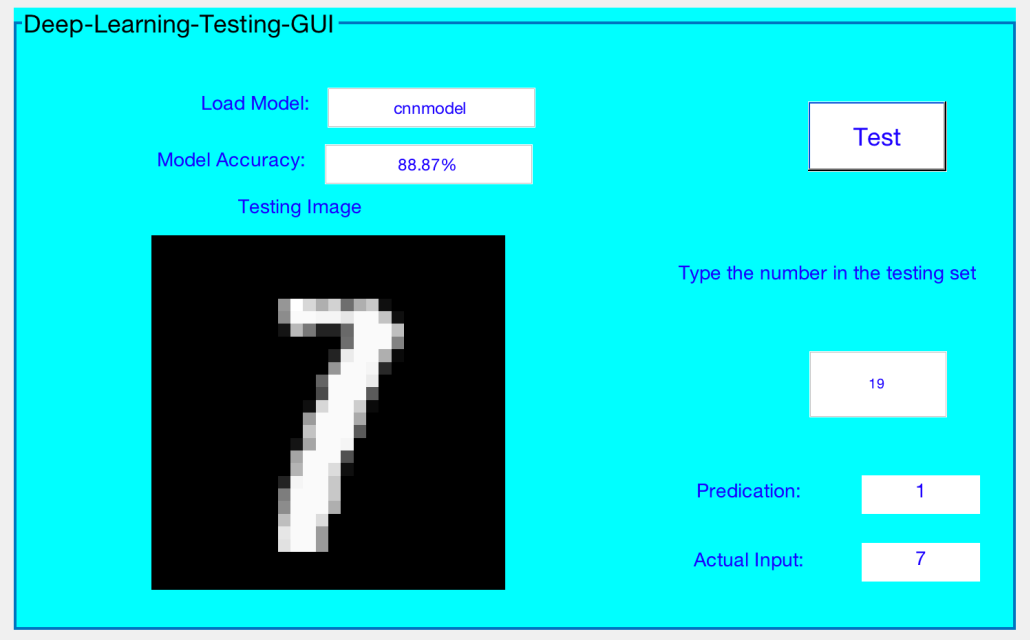

Figure 9: One example of the wrong prediction.

Conclusion

In this paper, we designed a simple and easy-to-use GUI for deep learning beginners who are passionate to learn and implement deep learning algorithms. Furthermore, our design prototype is model compatible, which can be utilized to train or test on any deep learning models with minor modifications. For the purpose of simplicity, we used a popular and well-organized public dataset MNIST as an example. We introduced some basic principles of CNN architecture including convolution and pooling, the two components of our GUI design: training and testing GUI, as well as how to functionally load a well-trained deep learning model and test the prediction results on specific cases, which aims to serve as a trigger to make more students benefit from studying deep learning, one of the most popular and promising techniques in the 21st century.

References

1 Y. LeCun, Y. Bengio and G. Hinton, "Deep learning," *Nature,* vol. 521, *(7553),* pp. 436-444, 2015.

2 A. Krizhevsky, I. Sutskever and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in Advances in Neural Information Processing Systems, 2012, .

3 V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," ArXiv Preprint arXiv:1603.07285, 2016.

4 Y. LeCun and Y. Bengio, "Convolutional networks for images, speech, and time series," The Handbook of Brain Theory and Neural Networks, vol. 3361, (10), pp. 1995, 1995.

5 Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition." *Proceedings of the IEEE*, 86(11):2278-2324, November 1998.