ReVACNN: Steering Convolutional Neural Network via Real-Time Visual Analytics

Sunghyo Chung[1], **Cheonbok Park**[1], **Sangho Suh**[1],
Kyeongpil Kang[1], **Jaegul Choo**[1], **Bum Chul Kwon**[2]
[1]Korea University, [2]IBM T.J. Watson Research Center
{sunghyo.chung, chunbok94, sh31659}@gmail.com,
{rudvlf0413, jchoo}@korea.ac.kr,bumchul.kwon@us.ibm.com

Abstract

Recently, deep learning has become exceptionally popular due to its outstanding performances in various machine learning and artificial intelligence applications. Convolutional neural network (CNN), a representative model of deep learning, has been successfully applied to solve computationally burdening tasks in various applications like computer vision. Despite its outstanding capability, training a CNN model properly is time-consuming and prone to overfitting and/or bad local minima. To address these issues, this study aims at improving the interpretation of the training process and using it for subsequent human intervention, specifically steering the training process of CNN model in the real time. In this paper, we present ReVACNN, a real-time visual analytic system for CNN, where (1) the overall training process (e.g., the amount of activations and that of changes each filter/layer has at a particular iteration/epoch) is visualized in network view and (2) the 2D embedding of trained filters within layers is visualized to show the relationships between filters as well as layers. In particular, ReVACNN allows users to perform several interactions in real time: (1) skipping the gradient descent update on the sub-part of a CNN model to reduce the subsequent training time and (2) steering filters interactively in the 2D embedding view to avoid bad local minima. At the end, we present several use cases that demonstrate the benefits users can gain from ReVACNN.

1 Introduction

Deep learning recently made major breakthroughs in numerous machine learning problems such as computer vision [1], speech recognition [2], and natural language processing [3, 4]. Beyond the traditional fully-connected model, the deep learning structure has evolved in various forms, including convolutional neural network (CNN) [5, 1], a type of neural network suited for real-world image classification and other tasks, as well as recurrent neural network [6] and long short-term memory (LSTM) network [7], another type of neural network utilizing dependencies in sequential and temporal data.

While significant achievements have been made, the understanding of underlying processes behind complicated deep learning models received less examination, and the need for tools and techniques for exploring and understanding the inner workings of these various models ensued. Some of the most urgent currently include the problems of properly training a deep learning model and avoiding underfitting/overfitting, both of which have been a time-consuming task that requires repetitive model selection and hyper-parameter tuning. For example, in order to train the model, a user has to constantly switch between different combinations of layers and nodes, the batch size, the step

30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

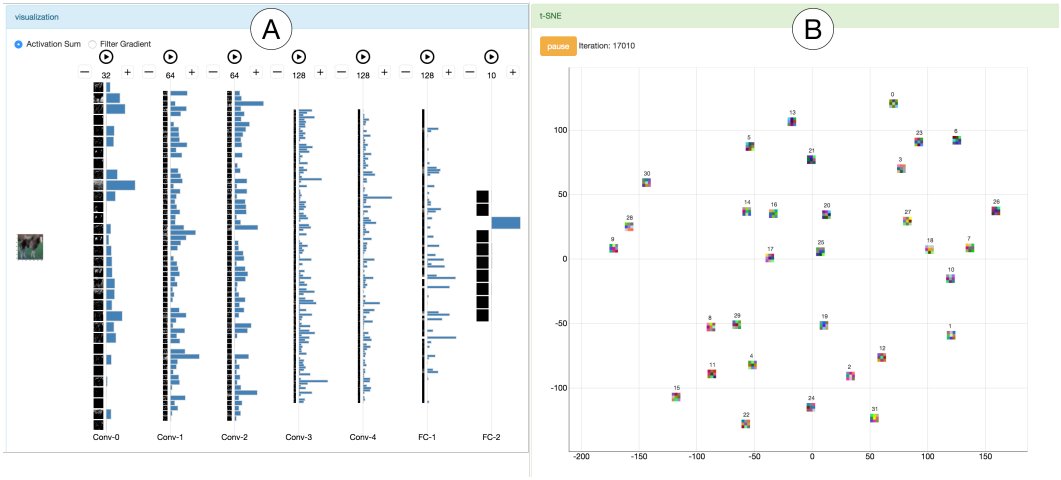


Figure 1: Overview of ReVACNN: (A) the network view; (B) the filter-level 2D embedding view.

size, and so on, but there have existed no intuitive or straightforward guidelines on how to properly perform this process.

In response, we present ReVACNN, a real-time visual analytic system for CNN, an expanded model from our previous work [8] with newly designed user interfaces and additional novel user interactions.

The main contributions of this paper are summarized as follows:

- Real-time visualization of how each node/filter in a deep learning model is being trained, e.g., the stability of nodes/filters and the relationships between them;
- Dynamic model steering by modifying structure of nodes/layers, facilitating update process and re-initializing filter coefficients in an intuitive manner in the training process.

The rest of this paper is organized as follows. Section 2 discusses related work. Section 3 presents detailed description of our system and its visual components. Section 4 presents usage scenarios. Finally, Section 5 concludes our discussion with plans for future work.

2 Related Work

Recently, there have been growing efforts towards building interactive visualization of deep learning for its in-depth understanding and user control.

Bruckner et al. [9] developed the system called deepViz, an interactive visualization based on the timeline framework that shows the heatmap representations of filters in each layer, the confusion matrix, and the clustered images at different check-points for understanding and diagnosing the network. ConvNetJS [1], which is implemented in web, made training convolutional neural network possible in a browser using the Javascript library. Bolei et al. [10] developed another web-based interface where a user can select the activation of a single data item at a particular layer and check the highly activated nodes together in the other layers.[2] Google also made a web interface called TensorFlow Playground [11][3] publicly available so that users can play with neural network models using several toy data sets.

As more analytic-oriented systems, Liu et al. proposed a visual analytic system for CNN called CNNVis [12], which provides the visualization of the learned features of nodes and the aggregation of low-level features into high-level ones. Notably, it performs clustering on layers and nodes and selectively visualizes representative ones to avoid the information overload from a visual clutter.

[1] http://cs.stanford.edu/people/karpathy/convnetjs/
[2] http://people.csail.mit.edu/torralba/research/drawCNN/drawNet.html
[3] http://playground.tensorflow.org/

Figure 2: 2D embedding view of ReVACNN

Even with various efforts mentioned above, a significant amount of room is available to improve the interactive visualization aspects of deep learning models along with the recent advancement in this field. In particular, the real-time monitoring and steering capabilities of deep learning have not been thoroughly addressed yet. Our proposed system allows users to monitor and steer the deep learning process in real time by interacting with a visual anlaytics system.

3 ReVACNN: Real-Time Visual Analytics for Steering Convolutional Neural Network

The main goal of ReVACNN is to provide real-time monitoring and steering capabilities on a CNN model using the visual analytics approach in an easy-to-use manner. As shown in Fig. 1, the main visualization modules of ReVACNN are composed of (1) the network view and (2) the 2D embedding view.

3.1 Network view

This module provides users with an overview of the activation of filters in layers. As shown in Fig. 1(a), the view shows the activation or the gradient maps of each node as well as a bar chart showing the total amount of (forward-propagated) activations or (back-propagated) gradients of individual filters during the training process.

It gives users the ability to monitor how the network is being trained in real time. For example, by checking those nodes with a large amount of activations, one can see which part or pattern of a given image is mainly captured by the model. On the other hand, the distribution of the gradient amount across different nodes/layers indicates which of them are relatively more converged or stable than the others and which are going through major changes.

In this visualization module, users can also easily add or delete filters in the hidden layer with simple point-and-click interactions (using the buttons on the top of each layer in Fig. 1(a)), and the change in the model is reflected in real time. Furthermore, users can selectively "freeze" particular nodes/layers so that they can be skipped in the subsequent training processes. Once these interactions are performed, the view gets dynamically updated so that users can visually check the effect of their interactions.

3.2 2D embedding view

As discussed above, the filter coefficients and the activation maps have frequently been the main subject of visualization when analyzing a convolutional neural network. In our system, we explore

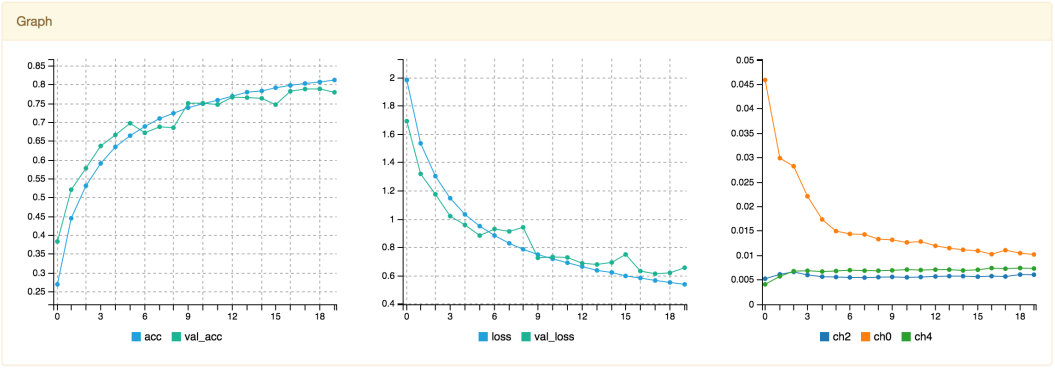


Figure 3: Training statistics view of ReVACNN

them using the 2D embedding view computed by t-distributed stochastic neighbor embedding (t-SNE). As shown in Fig. 2, this view shows the relationships of individual filters/nodes at a user-selected layer. Using the vector representation of each filter, a user can choose one among its (vectorized) filter coefficients, filter gradients, its activation map of a particular image, or its activation gradient maps so that s/he can explore various aspects of filters being trained. In this view, each filter is represented as a rectangle showing the details of the chosen input information, along with the circle showing the degree of convergence as the total sum of the gradients over the last epoch.

Using this view, one can obtain an idea about which filters are closely related to one another, i.e., which filters capture redundant information and which part of the information is not being captured. In general, the first-layer filters directly look at the raw pixel data of an input image, and thus their images are often the most interpretable among all the layers. Normally, an analysis of the first-layer weights can help users recognize whether the network has been successfully trained. Users can assess the success of training based on whether the trained filters have smooth transitions among them so that they can capture as diverse patterns as possible. However, since such analysis relies on subjective judgment, it is clear that users need additional evidences to decide whether such an analysis is reasonable.

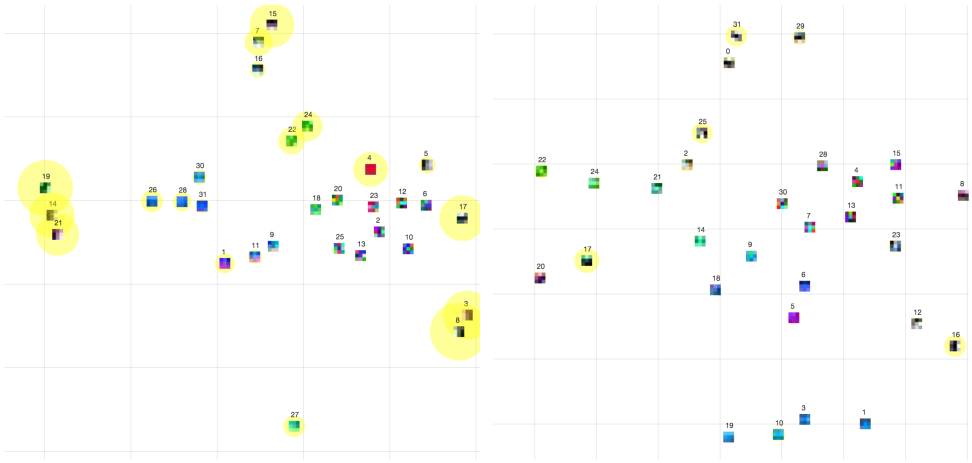
3.3 Training statistics visualization

During training, the loss function serves as a clue for identifying whether the network is properly being trained. Thus, our module, shown in Fig. 3 displays the training loss as a line chart. Users can keep track of the temporal progress of the loss function. In addition, other statistics, such as training accuracy and validation accuracy, are updated for each input image and shown to users for an in-depth analysis. To facilitate the understanding of how each node/layer has been trained, it is important that we directly enable users to observe changes in the training accuracy immediately.

3.4 Implementation details

The system consists mainly of the web front-end and the server back-end. The server, which uses a Python web framework called Flask,[4] is connected with a deep learning library, where the the model training status is sent to visualization in real time and interactive steering is propovided. As a deep learning library, we have used Keras.[5] It is a modular neural networks library running on top of either Tensorflow or Theano. In the experiment, we used Tensorflow as our backend. The front-end of our web-based system is implemented using HTML, CSS, and Bootstrap. D3.js[6] and Keras-Hualos [7] are used to visualize scalable vector graphic elements using the model parameter sets that are being streamed from the deep learning model in the server. All the computations are performed with Python and connected to the client side via the Jupyter Notebook.[8]

[4] http://flask.pocoo.org/
[5] https://github.com/fchollet/keras
[6] https://d3js.org/
[7] https://github.com/fchollet/hualos
[8] https://ipython.org/notebook.html



(a) Clear cluster pattern (b) No cluster patterns

Figure 4: Comparison of cluster patterns of filter coefficients

4 Usage Scenarios: Real-Time Monitoring and Steering

In this section, we present several use cases demonstrating the advantage of our system in monitoring and steering the deep learning model in real time.

4.1 Experimental settings

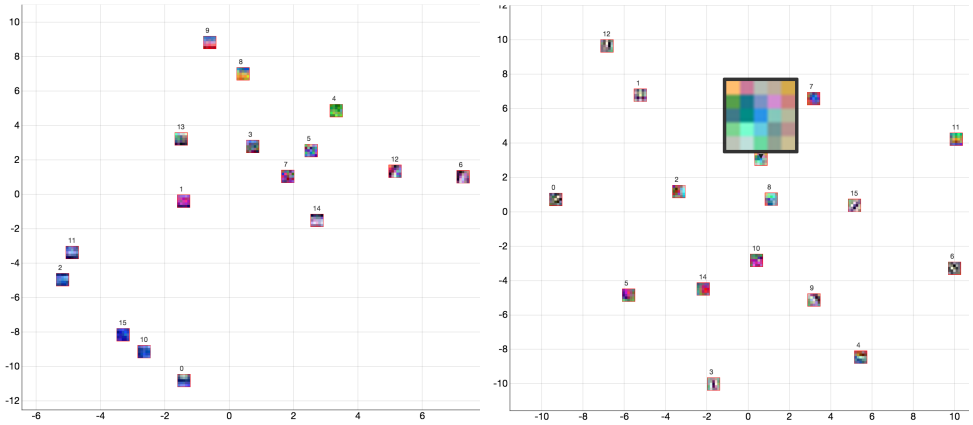
We used a small VGG-style neural network to train and test our system. It has an input layer having $32 \times 32 \times 3$ input images and three convolutional layers with a filter size of 3×3, with the stride size of 1 and padding of 1 where the three convolutional layers, each of which has 32, 64, and 64 filters, respectively, have both ReLU followed by each convolutional layer and pooling layers behind the first and the third convolutional layers, then followed by two fully connected layers and a softmax layer with ten classes as the last layer of our network.

Datasets. We used the CIFAR-10 [13] dataset as our training data, which consists of 60,000 32x32 color images in ten classes, with 6,000 images per class. The ten classes are truck, ship, horse, frog, dog, deer, cat, bird, and automobile. We split the entire data into 50,000 training and 10,000 test images.

4.2 Real-time monitoring

Cluster patterns. In general, neural network and deep learning models are sensitive to initialization, hyper-parameters, and other settings. Thus it is difficult to properly train the model so that it performs reasonably well even for the training data. Our filter-level 2D embedding view provides important insights about the characteristics of a properly trained model. When the accuracy was quite low, the 2D embedding view of filter coefficients showed a clustered pattern among these filters, as shown in Fig. 4a. This indicates that those filters belonging to a particular cluster capture somewhat redundant patterns from input data. In other words, they are not trained well enough to extract diverse patterns from the training data. On the other hand, where the accuracy reaches high, the 2D embedding view of filter coefficients exhibits somewhat evenly distributed filters with no clear cluster patterns, as shown in Fig. 4b. This example indicates an important characteristics of a well-trained model that the diversity of trained filters is generally desirable in achieving a greater classification accuracy.

RGB patterns. The first convolutional layer takes an input image with the depth of three RGB channels and generates each filter that linearly combines all the three channels. Similar to the previous case, when the model is not properly trained, e.g., showing a low accuracy value, we found that a trained filter often reflects only a single color channel as opposed to a combination of all the three color channels, as seen from all-blue colored filters in Fig. 5a. However, as seen in Fig. 5b, when the model shows a relatively good performance, each filter usually combines the information from all the



(a) Improperly trained filters (b) Properly trained filters

Figure 5: RGB patterns of the first-layer filters

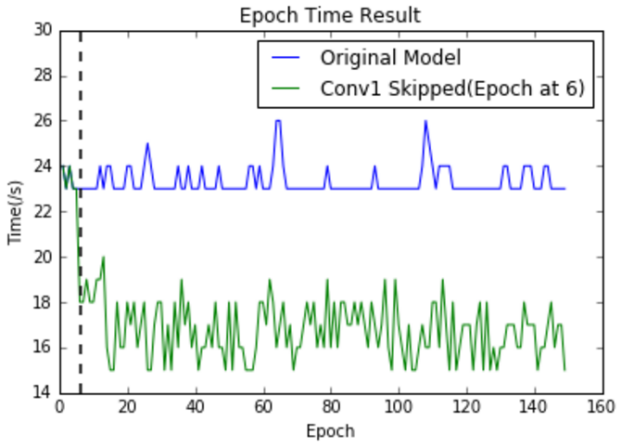


Figure 6: Computing times per epoch for the original model and the Conv1-Skipped (Epoch at 6) model on CIFAR-10 dataset. The vertical dotted line displays a particular epoch at which the layer was freezed.

color channels. Based on such different RGB patterns, one can infer that those filters that combine all the channels of the previous activation maps contribute to improving the generalization ability of the trained model.

4.3 Freezing layers

Another obstacle facing the use of deep learning is the time-consuming nature of training time. ReVACNN offers a solution to this problem through a feature we call “layer freeze,” which by skipping the gradient descent update enables satisfactory reduction of training time without significant loss in performance. We compared original model with other models that undergo layer freeze. Fig. 6 shows significant decrease in training time (at least 4 seconds) for each epoch by Conv1-Skipped (Epoch 6) model–which does not update its first convolution layer after epoch 6–compared to the original model. In this instance, the layer freeze was imposed at epoch 6 (indicated by vertical dotted line in the same figure) since t-SNE gradient view showed the signs of stability at this particular epoch.

5 Conclusion and Future Work

In this paper, we proposed ReVACNN, a real-time visual analytics system for convolutional neural network. It supports exploring and steering the network by visualizing the activation and gradient levels of layers and nodes. Additionally, we provided a filter-level 2D embedding view by applying t-SNE to various filter information, such as filter coefficients and the activation maps. Through these capabilities offered by our system, a user can obtain in-depth information such as whether the network is being properly trained as well as other insights about the trained filters. By using such information, a user can flexibly tune the model parameters (e.g., the number of filters within a layer) and achieve better performances.

Our current work provides many interesting opportunities for future work. Specifically, we plan to focus on more advanced dynamic steering capabilities. To support the interactive addition/removal of nodes/layers, their initialization could be carefully performed so that the newly added nodes/layers can capture complementary information of data to the existing nodes/layers. Additionally, when removing nodes/layers, we could recommend those that have a minimal impact to the overall performance, e.g., a redundant node from clustered nodes. We may be able to define the criteria to determine such minimal effects in various ways by using, say, the variable importance score of each node/layer.

Acknowledgments

This research was supported by Institute for Information & communications Technology Promotion (IITP) and Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (NRF-2016R1C1B2015924).

References

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

[2] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

[3] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Prof. the 25th International Conference on Machine Learning (ICML)*, pages 160–167, 2008.

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[5] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[6] Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179 – 211, 1990.

[7] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, Nov 1997.

[8] Chung Sunghyo, Suh Sangho, Park Cheonbok, Kang Kyeongpil, Choo Jaegul, and Kwon Bum Chul. Revacnn: Real-time visual analytics for convolutional neural network. In *ACM SIGKDD Workshop on Interactive Data Exploration and Analytics (KDD-IDEA)*, 2016.

[9] D. Bruckner, J. Rosen, and E. R. Sparks. deepviz: Visualizing convolutional neural networks for image classification. 2014.

[10] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Object detectors emerge in deep scene cnns. *arXiv preprint arXiv:1412.6856*, 2014.

[11] Daniel Smilkov, Shan Carter, D. Sculley, Fernanda B. Viegas, and Martin Wattenberg. Direct-manipulation visualization of deep networks. In *ICML Workshop on Visualization for Deep Learning*, 2016.

[12] Mengchen Liu, Jiaxin Shi, Zhen Li, Chongxuan Li, Jun Zhu, and Shixia Liu. Towards better analysis of deep convolutional neural networks. *arXiv preprint arXiv:1604.07043*, 2016.

[13] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.