DARVIZ: A Visual IDE to build Deep Learning Models

Shreya Khare
IBM Research, India
shkhare4@in.ibm.com

Naveen Panwar
IBM Research, India
naveen.panwar@in.ibm.com

Akshay Sethi
IIIT Delhi, India
akshay14133@iiitd.ac.in

Anush Sankaran
IBM Research, India
anussank@in.ibm.com

Senthil Mani
IBM Research, India
sentmani@in.ibm.com

Rahul Aralikatte
IBM Research, India
rahul.a.r@in.ibm.com

Neelamadhav Gantayat
IBM Research, India
neelamadhav@in.ibm.com

ABSTRACT

With abundance of research papers in deep learning, keeping up with the literature and adoption of the existing works become a challenge. Also due to the availability of a plethora of libraries, re-implementing deep learning models in a different library is a daunting task. To make the life of a deep learning developer or student easy, we propose a novel system, DARVIZ, to visually design and generate code for deep learning model designs. DARVIZ has a powerful drag-and-drop framework built over NODE-RED that can design deep learning models in an platform agnostic manner. The execution ready code for the designed models could be generated in Caffe and Keras (Tensorflow and Theano) in an automated fashion. While designing the model, a real-time static validation routine debugs the designs and provide corrective suggestions fix the model errors. Further, DARVIZ could import (i) any existing Caffe design models in prototxt files, or (ii) a research paper containing the deep learning design as a figure or table; extract the design, and visually present it in the NODE-RED based visual editor. The implementation code could be generated for in Caffe and Keras for the extracted designs.

CCS CONCEPTS

• **Computing methodologies → Information extraction**; • **Software and its engineering → Abstraction, modeling and modularity**;

KEYWORDS

deep learning, visual programming, model abstraction

ACM Reference Format:
Shreya Khare, Naveen Panwar, Akshay Sethi, Anush Sankaran, Senthil Mani, Rahul Aralikatte, and Neelamadhav Gantayat. 2018. DARVIZ: A Visual IDE to build Deep Learning Models. In *Proceedings of CODS COMAD 2018*. Goa, India, 4 pages. https://doi.org/

1 INTRODUCTION

The fundamental aim of artificial intelligence (AI) is to understand natural unconstrained data and perform complex tasks, like humans do. Machines automatically learn to perform these tasks using large volumes of data, the branch of study which is popularly called as data science. The yesteryear data scientists were more proficient with data processing platforms such as R and Matlab. As more than 80% of the data is unstructured in nature [1], making automated algorithms understand and infer the data is the biggest challenge. The advent of deep learning has revolutionized the perspective of data science enabling end-to-end learning from data [5]. Deep learning algorithms will learn a succinct representation of the data in an unsupervised fashion ie., the representation will be learnt directly using the data without the need for manual feature engineering. These algorithms have shown out-of-the-box successful results in various tasks such as automatic speech recognition [3], image recognition [4], natural language processing [2], and recommendation systems [10].

To enable implementation of deep learning algorithms, there are different libraries such as Caffe (C++/ Prototxt), Tensorflow (Python), and Torch (Lua) based out of different programming langauges. In our initial research, we conducted a survey across 100 developers and practitioners to understand the practical challenges involved in deep learning model development [6]. Summarizing the qualitative survey results, building applications using deep learning has the following challenges:

(1) Understanding the concepts of deep learning and the syntax of multiple libraries is a time taking and arduous task,
(2) There is very little communication across these libraries restricting public code reproducibility,
(3) Learning a deep model is like a black box, with minimum understanding of its internal working for a given data.

In this research we propose a novel cloud based framework called DARVIZ: Deep Abstract Representation, Visualization and Verification, to enable quick development and prototyping of deep learning model designs. DARVIZ is a visual IDE providing higher level abstraction of deep learning development, without the need for programming even a single line of code. Also, DARVIZ is a library agnostic development platform providing high interoperability across the libraries. This framework is made publicly available at http://darviz.mybluemix.net.

CODS COMAD 2018, Jan 2018, Goa, India
2018.

[1] https://www.ibm.com/blogs/business-analytics/data-is-everywhere/

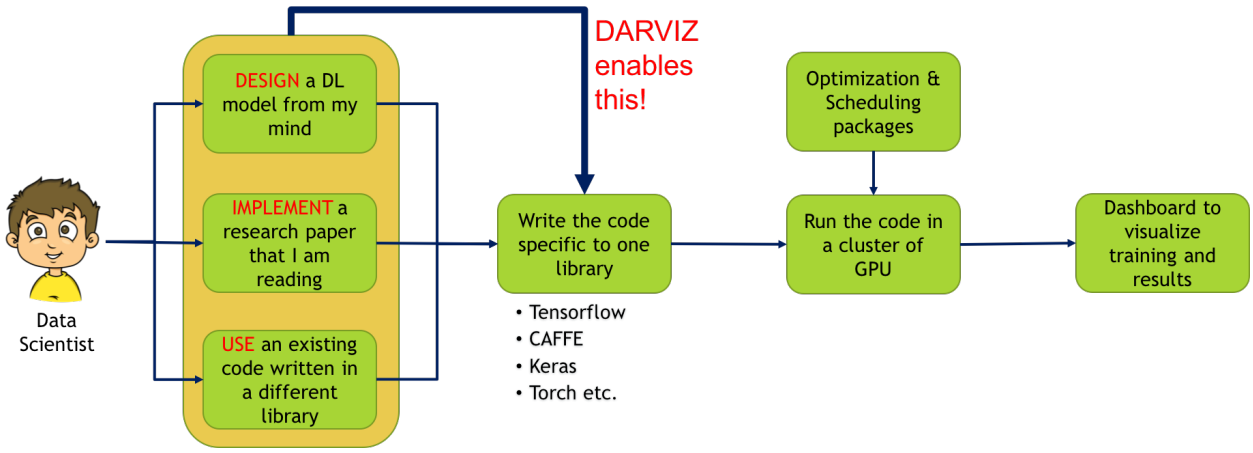

Figure 1: The primary features and use of the proposed DARVIZ system.

2 PROPOSED FRAMEWORK

The end-to-end components in the most popular pipeline followed by data scientists are shown in Figure 1. Three different use case occur often as explained below:

(1) For implementing a deep learning design model, a highly intuitive visual development framework is created using NODE-RED. This visual development framework allows a data scientist to create deep learning model designgs in a library agnostic fashion and extract execution ready code in two different libraries - Keras (Theano and Tensorflow) and Caffe.
(2) Research papers often offer state-of-art deep learning model designs which novice and beginners find hard to understand. DARVIZ could automatically parse the figures and tables in a research paper and extract the deep learning design explained.
(3) Deep learning code written by other developers usually exists in only one library and programming language. To be able to use the deep learning code available in a different library, DARVIZ offers code interoperability through model driven development. Currently, DARVIZ could read any CAFFE design file and converts into Keras code without any loss.

3 VISUAL DESIGNING OF A DL MODEL

Deep learning models are graph-like structures having a set of predefined layers and corresponding hyperparameters. DARVIZ provides an intuitive drag-and-drop UI, where a user can construct the complete network using basic units like layers. DARVIZ currently supports 23 distinct layers grouped as data layers, text processing layers, image processing layers, loss layers, core layers, and metric layers. These layers can be used to build huge deep learning networks processing both text and image data. The deep network designed can be automatically translated as an execution ready code in three libraries such as Caffe and Keras (Tensorflow, Theano). DARVIZ also enables data handling and manipulation capabilities through the user interface. Code for data preprocessing

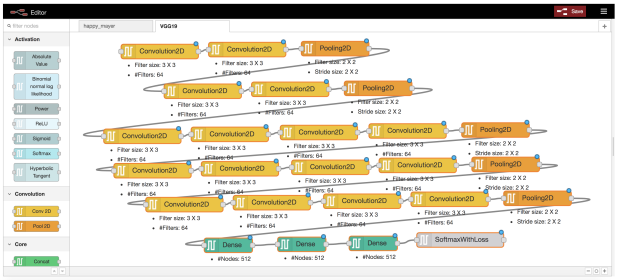

Figure 2: A library agnostic design of the popular VGG19 deep learning model using the DARVIZ framework.

in the supported formats and structure can be generated along with the design flow code within a few clicks. Consider one of the deep learning models used in computer vision called VGG19 network [1]. Typically in Tensorflow, this constitutes 400 lines of code (LOC) and takes a few hours to days to implement, based on the developer's expertise. However, the same model could be designed using the drag-and-drop UI within minutes, as shown in Figure 2.

Another common challenge developers face is debugging their implemented deep learning code for logical errors. Often, design errors are identified after hours of training and GPU usage. To overcome this, DARVIZ performs real-time static validation of the model reducing error probability of the model, significantly. A rulebook is pre-defined for each layer describing the input tensor dimensions and output tension dimensions, along with set of all possible previous layers, next layers, and rules constraining each of the hyperparameters. For example, if a network contains a series of operations which reduces the input image size significantly (below zero), the system prompts user at those set of layers and also provides constructive suggestions. This real-time suggestion is performed by translating the knowledge and efforts of researchers in the deep learning community to a comprehensive list of rules.

Consider the example of a popular Kaggle contest of cats vs. dogs classifier (https://www.kaggle.com/c/dogs-vs-cats). Different models exist for performing the task and the code is available in different

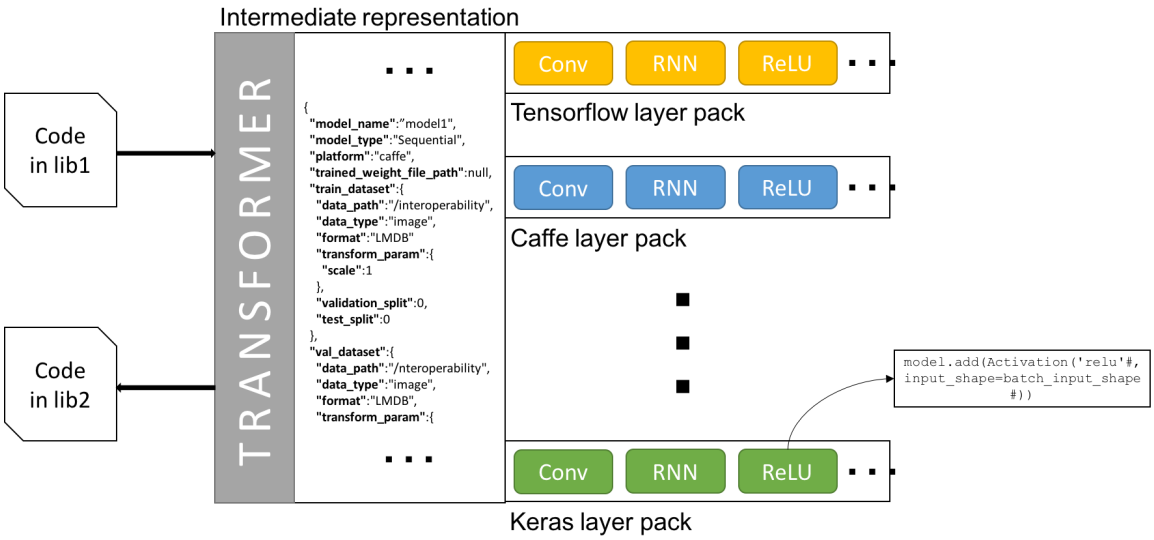

Figure 3: Internal DARVIZ framework describing the proposed intermediate representation for deep learning design models and the code interoperability flow.

libraries. Even if a data scientist does not know the syntax of any of these libraries, a model could be designed in an abstract manner as explained in https://goodboyanush.github.io/blogs/darviz_cats_dogs.html.

4 USE AN EXISTING CODE

Researchers and developers working in deep learning tend to release their code-base implemented in a particular library. DARVIZ enables easy code conversion of a deep learning design from one library to another through model driven development. The imported code is converted into the abstract internal design representation which can then be converted to the required output language's code, as explained in Figure 3. Currently, any code available in Caffe (design and solver prototxt) could be easily converted to a Keras (Tensorflow and Theano) code. This easy importing of deep learning models designed in one platform to another adds to the efficiency of the developers, considerably. DARVIZ performs interoperability by creating a abstract universal schema and platform-specific inference engine. A platform-specific inference engine consists of templates and dictionaries, which maps code generated in a particular platform to the abstract universal schema. Given a Caffe based deep learning network, the Caffe specific inference engine translates the code to the DARVIZ's abstract universal schema, as mentioned previously, this schema then gets realized to Keras code using a Keras based inference engine. Consider the Caffe implementation of the the DeepID paper [9] by *joyhuang9473* available at https://github.com/joyhuang9473/deepid-implementation. The code could be imported and visualized in DARVIZ as shown in Figure 4. For data scientists working in Keras language, they could import the Caffe code and convert it directly into Keras code without any transformation loss.

While Caffe offers the model design in a structured format such as a protobuf files, importing implemented code from other libraries

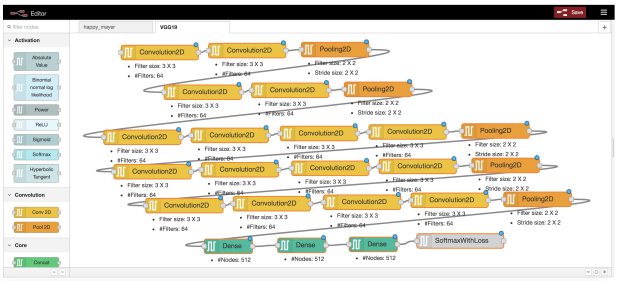

Figure 4: The design obtained in DARVIZ by automatically parsing a Caffe code. The design could further be extracted as a Keras code.

such as Tensorflow and Theano is not straightforward. However indirectly, accessing the computational flow graph of a designed model is possible through source code which provides a predefined structure to the deep learning model. For example, the computational graph of a session model in Tensorflow could be generated by executing the code without training. The input deep learning model could be potentially imported using such structured representations.

5 RESEARCH PAPER TO CODE

DARVIZ also provides the capability to understand deep learning models available in research paper in the form of figures and tables. Given a research paper, DARVIZ extract figures and tables providing the architecture details, parses them to generate the abstract computational graph. This computational graph can then be converted to platform-specific code. After extraction of tables and figures from the pdf of the paper, it provides the entire design in the visual drag-and-drop framework. The users could edit the design

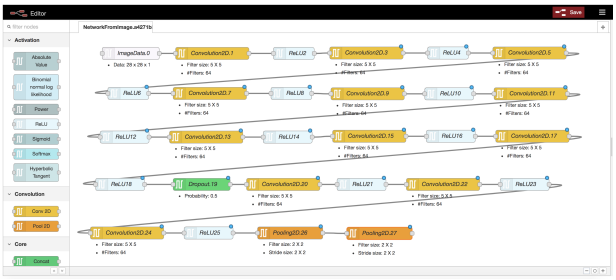

Figure 5: The design obtained in DARVIZ by automatically parsing the PDF of a research paper [8]. The research paper explains the deep learning design through a table.

and further generate the the execution ready source in both Caffe and Keras. The details of the technique is described in our recent publication [7].

Due to the lack of ground truth availability, evaluating such a feature is a huge challenge. Thus, we create a sophisticated grammar that defines and generates deep learning model designs. The grammar defines the possible next layers for a given layer and also the bounds of all the hyperparameters of a layer. Assuming, that every deep learning model starts with a data layer (image or text), the successive layers could be picked at random following the grammar along with the random selection of hyperparameters. A simulated dataset of more than 216, 000 valid deep learning models was generated along with its source code in both Caffe and Keras. Both Keras and Caffe libraries offer visualization of the deep learning model as images. Using the image visualizations of all those simulated deep learning models, the corresponding design was automatically extracted through image analysis techniques. Extracting the deep learning design from the image was more than 96% for Keras images and more than 92% for the Caffe images.

As an example of research paper parsing through tables, the popular research paper by Springenberg [8]. The PDF of the research paper could be uploaded in the DARVIZ system and the visual design is obtained as shown in Figure 5. Developers could then extract the Caffe or Keras code for the imported research paper's design.

6 CONCLUSION AND DISCUSSION

Thus, DARVIZ provides capabilities of quickly and easily implementing deep learning model designs and also generate source code in multiple libraries. Using three different use cases, we believe that DARVIZ could democratize deep learning based application development. The entry level barrier for deep learning is also reduced substantially using DARVIZ. DARVIZ also provides a flexible framework and hence addition of custom layers or supporting new libraries will be a mere extension of the platform. DARVIZ is freely available at https://darviz.mybluemix.net/.

REFERENCES

[1] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Return of the devil in the details: Delving deep into convolutional nets. *arXiv preprint arXiv:1405.3531*, 2014.

[2] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167, 2008.

[3] 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.

[4] 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.

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

[6] Anush Sankaran, Rahul Aralikatte, Senthil Mani, Shreya Khare, Naveen Panwar, and Neelamadhav Gantayat. Darviz: deep abstract representation, visualization, and verification of deep learning models. In *Proceedings of the 39th International Conference on Software Engineering: New Ideas and Emerging Results Track*, pages 47–50. IEEE Press, 2017.

[7] Akshay Sethi, Anush Sankaran, Naveen Panwar, Shreya Khare, and Senthil Mani. DLPaper2Code: Auto-generation of code from deep learning research papers. In *AAAI Conference on Artificial Intelligence*, 2018.

[8] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

[9] Yi Sun, Xiaogang Wang, and Xiaoou Tang. Deep learning face representation from predicting 10,000 classes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1891–1898, 2014.

[10] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. Collaborative deep learning for recommender systems. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1235–1244, 2015.