Blending intuition with reasoning – Deep learning augmented with algorithmic logic and abstraction

Context

Artificial intelligence is today one of the hottest buzzwords in science and technology. Whether it's real-time speech recognition or self-driving cars, AI-based systems are a reality embedded in our daily and future lives. Machine learning, a subfield of AI, has seen the most advancement in recent years, due to the availability of mass data combined with powerful hardware architectures. Machine learning is based on statistical learning to find patterns. Within machine learning, deep learning [LeCun15] is a subfield that has gained much traction since several high-profile success stories, e.g., DeepMind's AlphaGo beating the best Go player in the world.

Deep learning is based on neural networks, which are networks composed of artificial neurons, with non-linear activation functions, connected by weighted edges, imitating synapses. A neural network is organized with an input layer of data, one or several hidden layers to process the data, and an output layer of results. Through thousands to millions of input data, neural networks are trained for a specific task by self-updating its weights. Unlike classical computer reasoning, the statistical method by which a neural network solves a problem can be seen as a very primitive form of intuition.

Problem

Deep learning, in essence, is only a continuous geometric transformation [Chollet17]. Indeed, the data flowing through the neural network is represented as vectors or matrices. Therefore they only represent a set of points in a n-dimension space that are transformed to other sets of points in a m-dimension space. It does not matter what kind of learning is being performed (e.g., supervised, unsupervised, reinforcement), the neural network supporting the deep learning is limited to the same geometric logic. Since deep learning is only geometric logic, it can only learn the programs to solve a very limited subset of problems (e.g., pattern recognition problems) compared to the huge set of problems present in real life. So far, the only real success of deep learning has been its ability to self-tune the hard-coded continuous geometric transformation that lets it map space X to space Y, if it is given a large enough amount of human-annotated data. Unlike a human being, a neural network does not have the ability for reasoning and global generalization through algorithmic logic.

Furthermore, as the geometric transformations are hard-coded and then fine-tuned by training, although they are tremendously powerful for a given task, any deviation in the input data may give surprising and unpredicted results. For example, a network trained for autonomous driving in an European city would not necessarily perform as expected in an Southeast Asian city. However, many companies do not have large enough datasets to develop and train new and complex networks from scratch. They may also have existing legacy solutions adapted to sub-problems of their business market. It is therefore necessary to abstract, encapsulate, reuse and compose neural networks both homogeneously (i.e., assembling neural networks or sub-networks among themselves), and heterogeneously (i.e., assembling with classical expert algorithms). The composability issue is all the more difficult as a composition of reused trained neural networks does not guarantee better results, and may even lead to worst accuracies.

Related work

Although lacking in deep learning, algorithmic logic and abstraction is innate to classical software engineering. Indeed, in programming, primitives “if” blocks, loops, switches, etc… allows one to hard-code algorithmic logic. Behaviors modeled at higher levels of abstraction contain control flows and control nodes that bring conditional execution compared to a purely dataflow oriented model like neural networks. In software architectures, paradigms like abstraction, inheritance, polymorphism, encapsulation, etc… are only a few means to build modular a reusable software. Today very mature methodologies and their tools (e.g. Model-Driven Engineering [Schmidt06], Component-Based Modeling [Heineman01]), fully support the development of modular software that are easier to reuse and maintain.

There are however a certain number of neural networks that try to bring such algorithmic logic and abstraction in the deep learning world. For example Recurrent Neural Networks [Graves13] can be seen as a loop. When it comes to network reuse, we currently use weights that have been trained through a previous similar problems. Transfer learning [Pan10] connects an existing trained network to a new network so less training is necessary on the new network for a similar problem. Such approaches can be seen as early attempts at introducing algorithmic logic and abstraction for modularity in deep learning.

Objective

In this thesis, we propose to blend reusable algorithmic intelligence, providing the ability to reason, with reusable geometric intelligence, providing the ability of intuition. To achieve such an objective, we can explore some initial ideas like (1) integrating programming control primitives in neural networks, (2) applying software architecture paradigms in neural networks models, and (3) assembling modular systems using libraries containing both algorithmic modules (classical reusable software) and geometric modules (trained neural networks or sub-networks).

Perspective

The results of this thesis will be a stepping stone towards meta-learning based solutions. Just as neural networks can learn to solve a problem, meta-learning neural networks can learn how to build a neural network to solve a problem. The meta-learning mechanisms can consist in assembling abstract logic and geometric modules published by developers (or AIs themselves!) in a marketplace of intelligent behaviors (e.g., [Kaggle]). Dynamically deploying intelligent behaviors from a marketplace is one of the visions of the ongoing [BRAIN-IoT] European project.

The results of this thesis may also help engineers assemble AI systems without the expertise necessary to build the internal architecture of a neural network and fine-tune its features through training. Companies with less data may also reuse trained networks at a higher level of abstraction, thus demanding minimal adaptation effort, and they may even perform modular training, i.e., training differently sub-networks within a global network. Finally, as neural network traceability and validation is becoming primordial, being able to modularize their architecture may help with their validation through a compositional approach.

Laboratory and recruitment procedure

The PhD student will be attached to the CEA LIST research institute (Saclay, France) and benefit from its experience in deep learning [N2D2, Chabot17] and AI-enhanced model-based engineering [Modelia]. A co-supervision with an academic laboratory is possible.

To apply for this thesis, the candidate shall at least have a Master's degree in computer science or data science. An initiation to artificial intelligence or optimization is appreciated but not required. Software engineering skills are appreciated.

In the first phase, they shall send a CV, score sheets, an optional letter of recommendation, and any other document deemed necessary by the candidate.

In the second phase, the candidate shall have a discussion with the supervisors of this thesis (in person or by visual call). During this discussion, if they wish, the candidate may make a short presentation (30 minutes maximum) of their previous work and/or a presentation of their vision to tackle the foreseen challenges, with respect to the state of the art.

Contact

Shuai Li
Project Manager / Research Engineer
Institut Carnot CEA LIST, DILS/LSEA
CEA Saclay – Nano-INNOV – Bat. 862 – PC 174
91191, Gif-sur-Yvette Cedex, France
+33 1 69 08 76 33
shuai.li@cea.fr

References

- [BRAIN-IoT] http://www.brain-iot.eu/
- [Chollet17] Chollet F., "Deep learning with Python", Manning Publications Company, 2017.
- [Graves13] Graves A., Mohamed A-R., Hinton G., "Speech recognition with deep recurrent neural networks", Proceedings of ICASSP'13, Vancouver, Canada, 2013.
- [Heineman01] Heineman G.T., William T.C., "Component-based software engineering: Putting the pieces together", Addison-Westley, 2001.
- [Kaggle] https://www.kaggle.com/
- [LeCun15] LeCun Y., "Deep learning", Nature, 521, pp. pages 436–444, May 2015.
- [Pan10] Pan S.J., Yang Q., "A survey on transfer learning", IEEE Transactions on Knowledge and Data Engineering, 22:10, pp. 1345-1359, 2010.
- [Schmidt06] D.C. Schmidt D.C., "Model-driven engineering", IEEE Computer, 39:2, 2006.
- [N2D2] https://github.com/CEA-LIST/N2D2/blob/master/README.md
- [Chabot17] Chabot F., Chaouch M., Rabarisoa J., Teulière C., Chateau T., "Deep MANTA: A Coarse-to-fine Many-Task Network for joint 2D and 3D vehicle analysis from monocular image", Proceedings of CVPR'17, Honolulu, USA, 2017.
- [Modelia] https://modelia.eu/