Deep Learning in Automotive Software

Fabio Falcini and Giuseppe Lami, Information Science and Technologies Institute of the National Research Council of Italy

Alessandra Mitidieri Costanza, Fiat Chrysler Automobiles

// *Deep learning is becoming crucial to the development of automotive software for applications such as autonomous driving. Researchers have devised a framework that supports a robust, disciplined development lifecycle for such software.* //

DEEP LEARNING IS a branch of machine learning based on artificial neural networks (ANNs) that model high-level abstractions in input data by using a graph representation comprising multiple processing layers. For example, in computer-vision-based advanced driver assistance systems (ADASs),[1] deep-learning algorithms improve the detection and recognition of multiple objects, support object classification, and enable recognition and prediction of actions.

Many automotive companies consider deep learning to be a mature, viable technology, for not only improving ADAS performance but also making progress in functional domains such as engine management[2] and vehicle cybersecurity.[3] (For some examples of the current use of deep learning in the automotive domain, see the sidebar.) However, the introduction of deep-learning technology onboard cars is creating challenges for automotive software engineering, which must harmoniously incorporate this technology into its current state of the art.

Given deep learning's maturity and viability, it's fundamental to understand whether the deep-learning state of the art is aligned with automotive demands, at both the technical and methodological levels. At first glance, the answer seems to be "not entirely." The idea of solving a problem by training a neural network, instead of solving the solution using domain knowledge (feature engineering), is revolutionary for automotive software applications.

Here, we examine the basics of deep learning for automotive-software development, introduce a development lifecycle for this process, and show how this lifecycle meshes with current standards for automotive-software development.

Neural Networks and Deep Learning

ANNs can improve their problem-solving capabilities through learning triggered by the input of examples.[4] This feature is helpful in scenarios in which no detailed, complete, or predictable information exists about the problem, as is common in automotive-driving situations. ANNs' parallel structure lets them exploit powerful hardware to obtain timely computational results. Figure 1a shows a basic ANN.

Basic Deep Neural Networks

Deep neural networks (DNNs), which are synonymous with deep learning, are ANNs that model complex nonlinear relationships using multiple hidden layers of units between the input and output layers (see Figure 1b). Deep learning excels at finding patterns when the input is massive analog data—not a few numbers in tabular format but image

AUTOMOTIVE IMPLEMENTATIONS OF DEEP LEARNING

Google is making remarkable and highly visible investments in autonomous-vehicle development. Its prototype self-driving vehicles embed deep-learning-based technology that already can detect pedestrians in various and challenging scenarios. These deep-learning systems have achieved outstanding performance, making the error rate for machine vision lower than that for humans (the human benchmark is a 5 percent error rate). This achievement, also due to new hardware architectures using multiple GPUs, is pushing the migration of features based on traditional image-processing technology to deep-learning-based solutions.

This is just the beginning; even so, remarkable elements of AI are already available in vehicles. In the ADAS (advanced driver assistance system) domain, Tesla is reported to have implemented onboard neural-network functionality for vision, sonar, and radar processing that runs on the powerful Nvidia DRIVE PX 2 processor in the driving control unit.[1]

Several other suppliers are already active players. For example, Almotive (previously called AdasWorks) provides AI-based software solutions for the automotive industry for improving self-driving vehicles' safety using visual information from several cameras.[2] In addition, DENSO's R&D labs and other important companies' R&D labs are researching this area.

In the infotainment domain, the 2015 BMW 7 Series is reported to have been the first car to feature voice recognition based on deep-learning technology that also works in absence of wireless connectivity.[3]

From a hardware perspective, the electronic support required to embed deep learning in high-performance and safety-related automotive applications is so demanding that companies are aggressively developing new generations of chips. One example is Mobileye's upcoming cutting-edge EyeQ5 proprietary chip. Regarding commercially available electronics components, Intel is positioning its new Xeon Phi chip to compete in this market, which so far has been ruled by Nvidia's Tegra chip.

References

1. M. Pressman, "Inside Nvidia's New Self-Driving Supercomputer Powering Tesla's Autopilot," *CleanTechnica*, 25 Oct. 2016; cleantechnica.com/2016/10/25/inside-nvidias-new-self-driving-supercomputer-powering-teslas-autopilot.
2. "CEVA and AdasWorks to Demonstrate Free Space Detection for Autonomous Driving at AutoSens Conference 2016," CEVA, 15 Sept. 2016; ceva-dsp.mediaroom.com/2016-09-15-CEVA-and-AdasWorks-to-Demonstrate-Free-Space-Detection-for-Autonomous-Driving-at-AutoSens-Conference-2016.
3. L. De Ambroggi, "Artificial Intelligence Systems for Autonomous Driving On the Rise, IHS Says," IHS, 13 June 2016; technology.ihs.com/579746artificial-intelligence- systems-for- autonomous-driving- on-the-rise-ihs- says.

data (pixels) or audio data. Until advances in 2006, DNNs were outperformed by shallow neural networks that relied on feature engineering.

A DNN's structure is flexible and can be customized through the selection of attributes such as the number of hidden layers, number of units per layer, and number of connections per unit. These attributes, called *hyperparameters*, define the deep-learning-based system's structure and behavior.

Deep learning has these characteristics:

- I/O mapping that's created through the learning process,
- interconnected nonlinear computational elements (neurons or nodes),
- adaptability to environmental changes, and
- fault tolerance. (Because DNNs are distributed, localized faults in hidden layers lead to performance degradation rather than system failure.)

DNNs' processing capability is stored in *interunit connection weights*, which are obtained through adaptation to a set of training patterns.

Convolutional Neural Networks

Convolutional neural networks (CNNs), also called ConvNets, are DNNs that manage data in the form of arrays with some spatial structure (see Figure 1c). They emulate the visual cortex's behavior; they perform well on visual-recognition tasks because the convolution operation (in the shape of matrix products) can capture image features. CNNs have *convolutional layers* and *sampling*

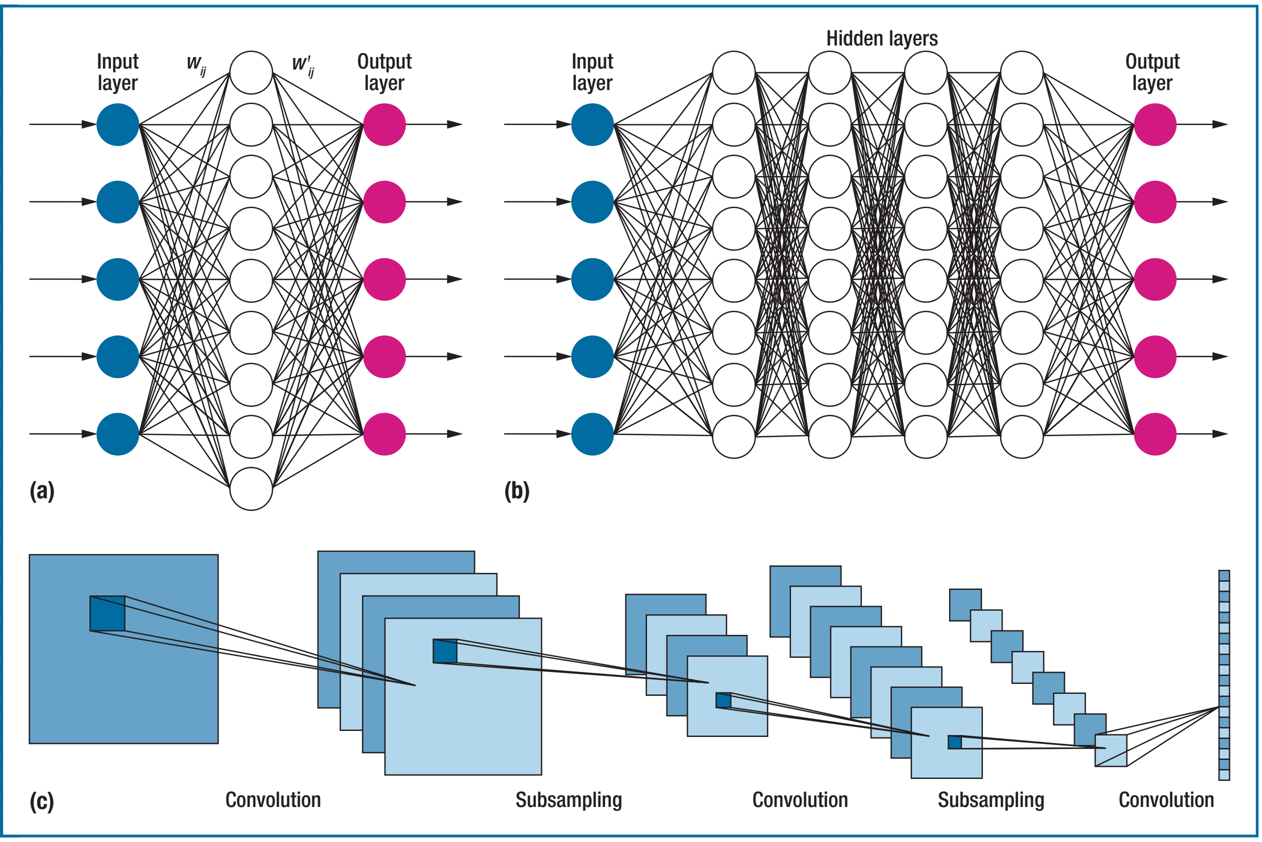

FIGURE 1. Examples of artificial neural networks (ANNs). (a) A basic ANN. w_{ij} and w'_{ij} stand for the weight of an input from element *i* belonging to a layer to element *j* belonging to the next layer. (b) A deep neural network (DNN), which models complex nonlinear relationships using multiple hidden layers of units between the input and output layers. (c) A convolutional neural network, a type of DNN that manages data in the form of arrays with some spatial structure.

layers that allow the encoding of image properties. Essentially, a CNN transforms 3D input (for example, an image with *W* rows, *W* columns, and three color channels) in a feed-forward mode[4] along the network.

CNNs, because of their characteristics such as input data segmentation and a high degree of parameterization (up to hundreds of thousands), are of special interest for automotive visual applications such as object, vehicle, and road-marking detection.

Recurrent Neural Networks

Recurrent neural networks (RNNs) are DNNs in which connections between units form a directed cycle.[5] They've been used successfully in speech recognition and natural-language processing.

From Theory to Practice

Here we illustrate a C implementation of some of the main deep-learning concepts. Figure 2 presents trivial samples of the neuron and connection data structures and a `create` function for a neuron.

Automotive applications can easily have DNNs with up to 10 layers and thousands of nodes. Their development benefits from the availability of consolidated software frameworks such as Theano, Caffe, Torch, Neon, TensorFlow, Deeplearning4J, and CNTK (the Microsoft Cognitive Tookit).

Training a DNN

In deep learning, the training itself actually functions as a programming activity. Here, for simplicity, we use the example of computer vision. To be able to match the input to the statistically expected correct result, DNN training requires a large set of image frames (typically extracted from video clips of driving scenarios) featuring shapes, edges, and colors.

This training is key because it allows full exploitation of the DNN's ability to detect an object, taking into account the image's context.

Typically, training a deep-learning algorithm takes days to weeks, forcing projects to compromise between accuracy and time to deployment. The training can employ one of three main strategies:[6]

- *Supervised training* uses a sample of the environmental knowledge based on training data in the form of pairs of an input and the corresponding target output, using appropriate labels.
- *Unsupervised training* leverages statistical regularities in the input data.
- *Reinforcement training* relates to taking actions in the environment to maximize the long-term reward; this is somewhat of a trial-and-error approach.

Some automotive applications effectively combine supervised training with reinforcement training.

Generally, the choice of training strategy depends largely on the type of DNN and the problem under consideration. In our experience, supervised training is used widely in DNN-based automotive-software development.

Datasets

Applying supervised training to automotive software requires massive annotated training datasets whose exact dimensions aren't publicly available but range from hundreds of thousands to millions of images. A training dataset typically contains annotations or labels of positive and negative regions.

Additional preprocessing can improve the training's accuracy and robustness. Figure 3 shows a typical example of training-data preprocessing: a bounding box corresponding to pedestrians that precisely delimits the human body's shape. Preprocessing also has a weighty organizational impact in terms of the know-how and human resources to be dedicated to this time-consuming task.

Supervised training also employs validation datasets to avoid overfitting and thus to select the most suitable algorithm by comparing the algorithms' performance. Overfitting occurs when a model is excessively complex and begins memorizing the training data rather than learning to generalize from trends.

In addition, supervised training uses test datasets to achieve the desired performance characteristics such as accuracy. These datasets represent a valuable proprietary asset of suppliers and original equipment manufacturers (OEMs).

```
typedef struct _connection {
  float inter_connection_weight;
  struct _neuron * from;
} NN_connection;

typedef struct _neuron {
  float param;
  NN_connection * neuronConnections;
} NN_neuron;

NN_neuron* create_neuron(int n) {
  NN_neuron* newNeuron = malloc(sizeof(NN_neuron));
  newNeuron -> param = 0

  connection * a = malloc(n*sizeof(NN_connection));
  newNeuron -> neuronConnections = a;
  return newNeuron;
}
```

FIGURE 2. A portion of DNN source code showing trivial samples of the neuron and connection data structures and a `create` function for a neuron.

The Learning Algorithm

During training, the learning algorithm populates an information structure, called a classifier, by learning from the training dataset. Accordingly, the training dataset's environmental knowledge is transferred to the classifier and implicitly to the DNN.

Several learning algorithms are available for supervised training; the most popular is backpropagation.[7] Backpropagation has two phases. In the first phase, a training input pattern is fed to the network input layer. The network then propagates the input pattern from layer to layer until the output layer generates the output pattern. If this pattern diverges from the expected output, an error is computed and propagated backward through the network from the output layer to the input layer. As the error is backpropagated, the second phase occurs, in which the network's interunit connection weights are modified.

A Lifecycle for Deep Learning: The W Model

The software side of DNN development is a highly iterative activity comprising five steps:[8]

1. DNN requirements identification,
2. Learning-algorithm development,
3. DNN training,
4. DNN training validation, and
5. DNN validation.

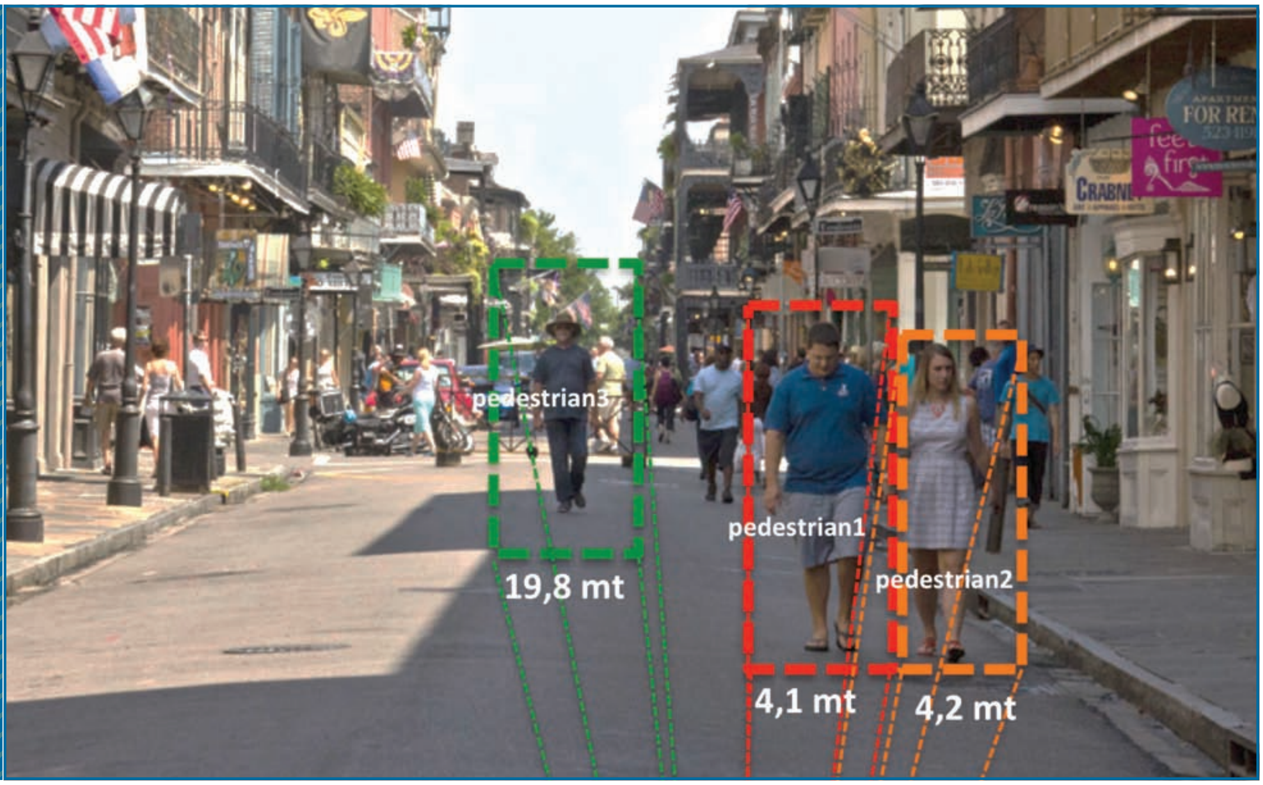

FIGURE 3. Annotated training data. This typical example of training-data preprocessing shows a bounding box corresponding to pedestrians that precisely delimits the human body's shape.

Unlike traditional development, deep-learning development needs the support of empirical design choices driven by heuristics. Development often starts from well-known learning algorithms that have been proven effective in comparable problems or domains. This is because the learning-process results are difficult to understand and thus manage.

The expected DNN requirements also include performance demands, which are carefully targeted during DNN validation. These demands are expressed in terms of a statistical benchmarking of the DNN's functional behavior (that is, the error rate).

The development and adaptation of the appropriate algorithms are broadening R&D labs' skills. These algorithms typically are based on convolutional approaches that enable the DNN's learning capability in automotive applications. In fact, the notable presence of scientists, even at the industrial level, who are cooperating with engineers during DNN design is evidence of how challenging this promising setting is.

During training, the learning algorithm is evaluated and the interunit connection weights are repetitively and experimentally adjusted. This loop continues until the prediction reaches its target accuracy.

The Need for Robust, Predictable Development

Although automotive software engineering welcomes innovation and outstanding functional performance, it still demands a robust, predictable development cycle. So, it's important to approach deep learning from a more controlled V-model perspective to address a lengthy list of challenges. Such challenges include requirements criteria for the training, validation, and test datasets; criteria for training-data preprocessing; and the management of very large sets of parameters.

A more structured conception of the deep-learning lifecycle will be instrumental in reaching a controlled development approach that can't be addressed by the mere functional benchmarking obtained with validation activities. It's essential to pursue both high performance at the functional level and a high-quality development process.

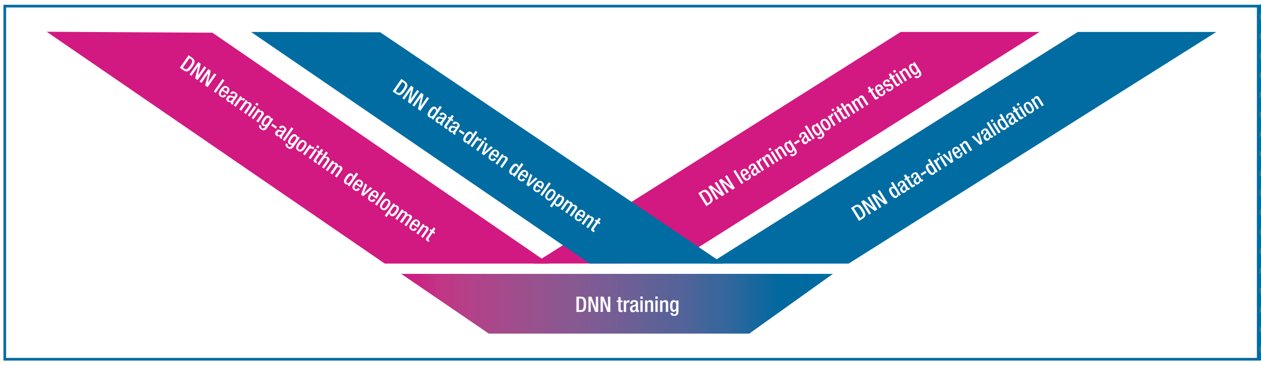
DNN learning-algorithm development
DNN data-driven development
DNN learning-algorithm testing
DNN data-driven validation
DNN training

FIGURE 4. The W model for deep learning using DNNs. This model conceptually integrates a V model for data development with the standard V model for software development.

However, deep learning intrinsically introduces features to software development that don't completely fit the V model. Data's central role in this context (for example, for DNN training and training validation) makes essential the introduction of what we call the *W model*. To support it, we employ the term "programming by example" to highlight data's importance in developing systems based on deep-learning technology.

Introducing the W Model

The W model conceptually integrates a V model for data development with the standard V model for software development (see Figure 4).

This lifecycle model acknowledges that both software development and data development drive deep learning. The design and creation of training, validation, and test datasets, together with their exploitation, are crucial development phases because the DNN's functional behavior is the combined result of its architectural structure and its automatic adaptation through training.

By definition, deep learning moves away from feature engineering. This aspect makes the W model an appropriate, useful representation of this sophisticated paradigm.

Automotive-Software Standards and Deep Learning

Software development for onboard automotive electronic control units (ECUs) is subject to proprietary OEM norms and several international standards. The most relevant and influential standards for deep learning are Automotive SPICE (Software Process Improvement and Capability Determination)[9] and ISO 26262,[10] which both rely on the V model. However, these standards are still far from addressing deep learning with dedicated statements.

Automotive SPICE provides a process framework that structures, at a high abstraction level, software development activities. It allows assessment of these activities' capabilities by matching the activities to predefined sets of process requirements.

ISO 26262 targets safety-related automotive development; its scope includes systems, hardware, and software engineering. It already addresses configuration and calibration data, even though this aspect is an order of magnitude simpler and plainer than the development of DNN datasets.

The ISO/AWI PAS 21448 standard,[11] which is in advanced development, is also relevant for deep learning. It addresses the fact that for some ADAS applications, a fault-free system can still suffer from safety violations (for example, a false-positive detection of an obstacle by radar). This situation can occur because developing a system that can address every possible scenario is extremely problematic.

Because of Automotive SPICE's pervasive adoption and holistic coverage of automotive-software development, it's the appropriate reference for systematically analyzing deep learning for automotive software engineering and for promoting a mature, harmonized methodology for deep learning.

Figure 5 overlays the software part of the V model of Automotive SPICE 3.0 on the W model. This hints at the need to organize deep learning's evolution according to Automotive SPICE's process requirements. The same approach might apply to ISO 26262.

Given the automotive market's stringent requirements for ECU development in terms of rigor and control, the substantial growth and stabilization of deep learning is critical. Just scratching the surface raises important methodological issues—for instance, the traceability

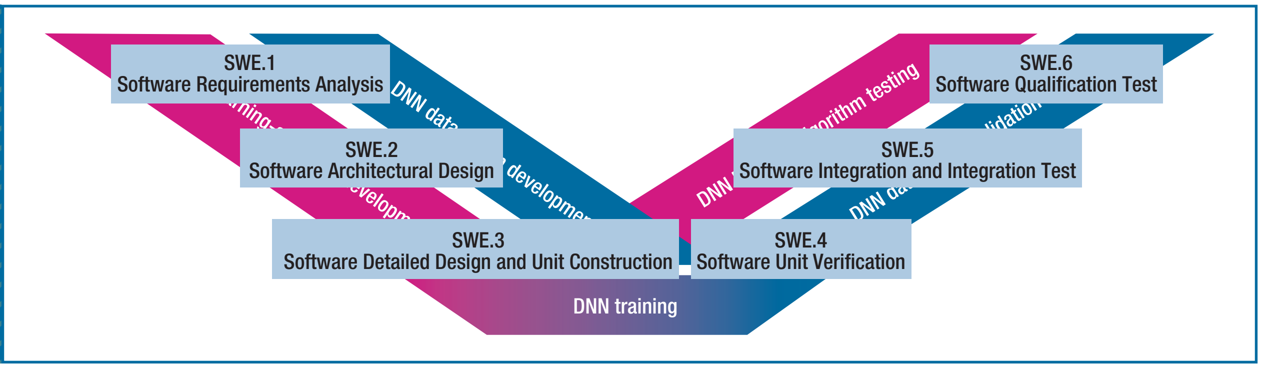
SWE.1
Software Requirements Analysis
SWE.2
Software Architectural Design
SWE.3
Software Detailed Design and Unit Construction
SWE.4
Software Unit Verification
SWE.5
Software Integration and Integration Test
SWE.6
Software Qualification Test
DNN training
DNN data
development
algorithm testing
validation
DNN

FIGURE 5. The W model from the Automotive SPICE (Software Process Improvement and Capability Determination) 3.0 software perspective. This hints at the need to organize deep learning's evolution according to Automotive SPICE's process requirements.

infrastructure among the deep-learning development stages.

This improvement path is thus pivotal to the attainment of the ambitious future goals for automotive software, such as fully autonomous driving.

As deep learning ushers in radical changes to automotive software, the W model is a promising and easily understood basis for the comprehensive integration of deep learning with traditional automotive software engineering.

The Italian Automotive Software Process Improvement Network (Automotive SPIN; www.automotive-spin.it) has launched an open working group to facilitate the harmonization of deep-learning practices in automotive software engineering using standards such as Automotive SPICE 3.0, the next edition of ISO 26262, and ISO/AWI PAS 21448. Furthermore, Automotive SPIN will address the W model's characterization.

References

1. J. Levinson et al., "Towards Fully Autonomous Driving: Systems and

Algorithms," *Proc. 2011 IEEE Intelligent Vehicles Symp.* (IV 11), 2011; ieeexplore.ieee.org/document/5940562.
2. A. Parlak et al., "Application of Artificial Neural Network to Predict Specific Fuel Consumption and Exhaust Temperature for a Diesel Engine," *Applied Thermal Eng.*, vol. 26, nos. 8–9, 2006, pp. 824–828.
3. M.-J. Kang and J.-W. Kang, "Intrusion Detection System Using Deep Neural Network for In-Vehicle Network Security," *PLoS ONE*, vol. 11, no. 6, 2016; journals.plos.org/plosone/article?id=10.1371/journal.pone.0155781.
4. J. Schmidhuber, "Deep Learning in Neural Networks: An Overview," *Neural Networks J.*, vol. 61, 2015, pp. 85–117.
5. S. Haykin, *Neural Networks and Learning Machines*, Prentice-Hall, 2009.
6. J. Credi, "Traffic Sign Classification with Deep Convolutional Neural Networks," master's thesis, Dept. of Applied Mechanics, Chalmers Univ. of Technology, 2016.
7. Y. LeCun, Y. Bengio, and G. Hinton, "Deep Learning," *Nature*, vol. 521, no. 7553, 2015, pp. 436–444.
8. H. Miao et al., "ModelHub: Lifecycle Management for Deep Learning," Univ. of Maryland, 2015; www.cs.umd.edu/class/spring2016/cmsc396h/downloads/modelhub.pdf.
9. *Automotive SPICE Process Assessment / Reference Model*, ver. 3.0, Verband der Automobilindustrie, 2015; www.automotivespice.com/fileadmin/software-download/Automotive_SPICE_PAM_30.pdf.
10. *ISO 26262—Road Vehicles—Functional Safety—Part 1: Vocabulary*, Int'l Standard Org., 2011.
11. *ISO/AWI PAS 21448—Road Vehicles—Safety of the Intended Functionality*, Int'l Standard Org., 2016.

ABOUT THE AUTHORS

FABIO FALCINI is an independent software professional and a research fellow at the Information Science and Technologies Institute of the National Research Council of Italy. His main research and activity areas include industrial software engineering, functional safety, deep learning, and software quality. Falcini received a master's in software engineering from TecnoPadova. He's an Automotive SPICE (Software Process Improvement and Capability Determination) Principal Assessor and actively contributes to the ISO 26262 community. Contact him at fabio.falcini67@gmail.com.

GIUSEPPE LAMI is a researcher at the Information Science and Technologies Institute of the National Research Council of Italy and is the head of the institute's System and Software Evaluation Center. His research interests relate mainly to software quality evaluation and software process assessment and improvement. Lami received a PhD in information technology engineering from the University of Pisa. He's an Automotive SPICE (Software Process Improvement and Capability Determination) Principal Assessor and the president of Automotive SPIN (Software Process Improvement Network) Italia. Contact him at giuseppe.lami@isti.cnr.it.

ALESSANDRA MITIDIERI COSTANZA is a senior software specialist and software quality specialist at Fiat Chrysler Automobiles, where she develops and applies methodologies for software and supplier management. She received a Laurea in electronic engineering from Sapienza University. She's an Automotive SPICE (Software Process Improvement and Capability Determination) Provisional Assessor and a member of the Automotive SPIN (Software Process Improvement Network) Italia management board. Contact her at alessandra.mitidieri@fcagroup.com.