scientific reports

OPEN

Bayesian optimization and deep learning for steering wheel angle prediction

Alessandro Riboni[1], Nicolò Ghioldi[1], Antonio Candelieri[1] & Matteo Borrotti[1,2]✉

Automated driving systems (ADS) have undergone a significant improvement in the last years. ADS and more precisely self-driving cars technologies will change the way we perceive and know the world of transportation systems in terms of user experience, mode choices and business models. The emerging field of Deep Learning (DL) has been successfully applied for the development of innovative ADS solutions. However, the attempt to single out the best deep neural network architecture and tuning its hyperparameters are all expensive processes, both in terms of time and computational resources. In this work, *Bayesian optimization* (BO) is used to optimize the hyperparameters of a *Spatiotemporal-Long Short Term Memory* (ST-LSTM) network with the aim to obtain an accurate model for the prediction of the steering angle in a ADS. BO was able to identify, within a limited number of trials, a model—namely BO_ST-LSTM—which resulted, on a public dataset, the most accurate when compared to classical end-to-end driving models.

Over the last decade, significant progress has been made in automated driving systems (ADS). Given the current momentum and progress, ADS can be expected to continue to advance as variety of ADS products are going to become commercially available in the space of a decade[1]. It is envisioned that automated driving technology will lead to a paradigm shift in transportation systems in terms of user experience, mode choices and business models. Nowadays, a greater number of industrialists are increasing their investments in self-driving cars technologies and, more generally, in the automotive sector. ADS research and an increasing number of industrial implementations have been catalyzed by the accumulated knowledge in vehicle dynamics in the wake of breakthroughs in computer vision caused by the advent of deep learning[2–5] and the availability of new sensor modalities such as lidar[6].

Deep Learning (DL) has been widely used for the implementation of ADSs. Starting from the work of Krizhevsky et al.[2], different DL approaches have been proposed. Bojarski et al.[3] proposed a deep neural network (*PilotNet*) for predicting the steering angles thanks to a set of images captured from the car. Based on Krizhevsky et al.[2], Kocić et al.[4] proposed the so called *J-net*, that is a DL model suitable for end-to-end systems. All of the above mentioned methods follow direct supervised training strategies. A review of these methods for ADS can be found in Yurtsever et al.[7]. In this context, a ground truth is required for training, which normally consists in the ego-action sequence of an expert human driver, while the network learns to imitate the driver. More precisely, the training is performed on a set of camera images taken from the front car (input), which is a raw signal (*i.e.* pixels) and, for example, the steering angles (output) used to control the car. Previously mentioned methods are then trained using road images paired with the steering angles generated by a human tasked with driving a data-collection car. The prediction tasks are solved using Convolutional Neural Networks (CNNs)[8]. Generally, CNNs use layers that convolve the inputs with filters and compress them. As a result, CNNs can find features and transform them into simpler representations. This ability has made them useful in many different applications (e.g. arts and image classification, identification of movements)[9]. However, CNNs predict one frame at a time and generate future images recursively, which are prone to focus on spatial appearances and relatively weak in capturing long-term motions[10]. In order to capture a temporal relation, a possible solution is the use of Long Short Term Memory (LSTM)[8] networks. The core idea behind the LSTM architecture is a memory cell which can maintain its state over time as non-linear gating units which regulate the information flow into and out of the cell[11].

In order to overcome the limits of CNNs and exploit the advantages of LSTMs, Shi et al.[12] proposed the convolutional LSTM (ConvLSTM). ConvLSTM network, which is able to model the spatiotemporal structures simultaneously by explicitly encoding the spatial information into tensors, thus overcoming the limitation of

[1]Department of Economics, Management and Statistics, University of Milano-Bicocca, Milan, Italy. [2]Institute of Applied Mathematics and Information Technology, CNR-IMATI, Milan, Italy. ✉email: matteo.borrotti@unimib.it

vector-variate representations in standard LSTM where the spatial information is lost. Similar approaches have been also used in the context of ADS[13,14]. In both works, authors used 4 consecutive ConvLSTM layers as the first part of the deep network, then Yu et al.[13] completed the network with a fully connected layer and Bai et al.[14] used a 3D convolutional (3DConv) layer[15] followed by 2 fully connected layers.

DL has been successfully applied in different fields, gaining remarkable results. Nonetheless, a crucial issue dealing with DL remains, which is neural network architecture definition[16]. In fact, currently employed architectures and related hyperparameters settings have mostly been developed by human experts, a manual process which is both time-consuming and error-prone. For instance, in LSTM networks many parameters and variants can be used to solve the same problem. Performance is then influenced by the final version of the LSTM networks that is used. Greff et al.[11] called it a "*...search space odyssey...*".

In the Machine Learning (ML) and DL community, Bayesian optimization (BO)[17,18]—sometimes also named Sequential Model Based Optimization (SMBO)—has recently became the standard strategy for Automated Machine Learning (AutoML)[19] and Neural Architecture Search (NAS)[16]. BO is a general sample-efficient strategy for global optimization of black-boxes, which are expensive and multi-extremal functions, traditionally constrained to a box-bounded search space. Furthermore, BO has also been extended to the case of unknown constraints and/or partially defined objective functions[20–24], as well as to the case of *weakly specified search spaces*[25,26]. A recent review of automated ML techniques for DL approaches can be found in He et al.[27]. Specifically for NAS, a recent review is provided in[28], with also evolutionary approaches have been recently proposed, often in combination with BO[29–31].

This work proposes the application of *Bayesian optimization* (BO) combined with *Spatiotemporal-Long Short Term Memory* (ST-LSTM) network for the prediction of the steering angles in automated driving systems based on camera images (i.e. raw pixels) taken from the front car (called BO_ST-LSTM from now on). Unlike previous works[13,14], we have added a MaxPooling layer to reduce the complexity of the final network and we have applied the BO to optimize the hyperparameters and obtain a more suitable model for the experiment taken into account. More precisely, the main contributions of this study can be summarized as follows.

1. We propose a sample efficient approach, based on BO, for optimally tuning an ST-LSTM based network to enhance steering wheel angle prediction: we named the resulting final model BO_ST-LSTM. BO_ST-LSTM has demonstrated higher accuracy with respect to competitors and a better generalization performance between validation and test set.
2. We propose a general hyperparameter optimization framework for DL methods in the context of automated driving systems (ADS) based on BO. This hybrid approach can improve the predictive power of DL architectures by finding the most suitable configuration for the problem under study.

Our results can be adapted and used in the development of ADSs to help obtain a more precise prediction of steering wheel angle, leading to safer car for drivers and traffic participants.

The following sections of the paper are organized as follows. "Related work" section will discuss related works in the field of automated driving systems (ADS). "Proposed method" section will present our method of prediction of steering angles by coupling BO and ST-LSTM, while "Experiments" section will explain the experiments and performances of our method. Finally, "Conclusion and future works" section will present discussions and conclusions of the paper.

Related work

Object recognition is an important task in different fields and it is often solved by using machine learning methods and large datasets. An example is the so-called *AlexNet*[2]. In this work, the authors trained a large convolutional neural networks on the subsets of ImageNet dataset[32] used in the ILSVRC-2010 and ILSVRC-2010 competitions[33], thereby obtaining the best results ever reported on these datasets in 2012.

In the context of ADS, Deep Learning (DL) has been widely used for their implementation[34]. Bojarski et al.[3] proposed a neural-network-based system known as *PilotNet*, which outputs steering angles given images on the road ahead. PilotNet training data contains single images sampled from videos recorder by a front-facing camera in the car, the former paired with the corresponding steering command. The PilotNet network architecture consists in a set of normalization layers, convolutional layers and fully connected layers. The proposed net is mainly tested for understanding the ability to recognize objects that can affect the steering. In the same year, Kim and Canny[35] proposed a visual attention model to train a convolution network end-to-end from images to steering angle. The attention model highlights image regions that potentially influence the network's output. Successively, a causal filtering step is applied to determine which input regions actually influence the output.

More recently, Kocić et al.[4] developed an end-to-end deep neural network (called *J-net*) suitable for deployment on embedded automotive platform modifying the AlexNet[2] solution, which is a convolutional neural network for image classification. Differently from AlexNet and PilotNet, J-Net is based on a set of convolutional layers and a set of max-pooling layers used to reduce the number of parameters. Compared with AlexNet and PilotNet, J-Net had comparable results in terms of predictive power, but with a lower complexity overall.

Motion planning is a fundamental technology for autonomous driving vehicles and, over the past years, novel deep learning approaches have been demonstrated to be power techniques. In Yu et al.[13], two main contributions can be found. Firstly, the authors introduced the Baidu Driving Dataset (DBB): DBB is a new driving dataset, which contains a 10,000-km frontal camera image and a vehicle motion attitude data of real road-conditions. Secondly, the authors have proposed an end-to-end reactive control model for lateral and longitudinal controls. The former referring to steering angle predictions, the latter referring to means accelerating and braking commands optimization. Similarly to other works[3,4], a set of pre-processing layers, convolutional layers and fully connected layers were used for steering angle prediction. The accelerating and braking problem was solved with a Convolutional Long Short Term Memory architecture (Conv-LSTM)[12] using 5 frames, taken from the front camera car, as input. The network architecture was composed of 4 Conv-LSTM layers and a fully connected layer.

The Conv-LSTM network combines image feature extraction capabilities from convolutional networks and the memory ability of LSTM networks. Starting from the work of Yu et al.[13], Bai et al.[14] proposed the so-called Spatiotemporal-Long Short Term Memory (ST-LSTM) network for motion planning value prediction in the context of autonomous vehicle. The ST-LSTM network is composed of 4 ConvLSTM layers followed by a 3-D convolutional (3DConv) layer[15] and 2 fully connected layers. As in Yu et al.[13], authors do not consider a single image as input but 3 continuous frames to ensure the real-time performance of motion result.

Recently, AutoML and NAS are becoming standard solutions for optimizing hyperparameters and architecture of neural networks, respectively, with examples in different application domains. For instance, in Nguyen et al.[36] an accurate and reliable multi-step ahead prediction model based on LSTM, whose hyperparameters have been optimized through BO, has been validated on steam generator data acquired from different French nuclear power plants for prognostic and health management of the plants. In Osmani et al.[37], an expensive and time-consuming design of a Deep Neural Network for human activity recognition has been addressed via BO in order to optimally and efficiently tune the deep neural architectures' hyper-parameters. With respect to ADS, a recent and interesting application of BO is devoted to generate simulation scenarios in order to improve accuracy and "safety" of the ADS[38–40]. In Kong et al.[41], authors proposed a deep Q-learning (DQL)-based energy management strategy (EMS) for an electric vehicle. In this work, BO is used to optimize the hyperparameter configuration of the DQL-based EMS. Another interesting work is Alizadeh et al.[42], in which a novel attention-based LSTM cell has been proposed and optimized by Bayesian optimization for streamflow postprocessing which outperformed the simple LSTM, GRU, a machine learning algorithm, and two statistical-based models.

Proposed method

Firstly, the general architecture of the ADS for steering angle prediction is hereby presented, followed by a short description of the two main ADS components: BO and ST-LSTM.

Figure 1 shows the general approach of the BO_ST-LSTM system design for steering angle prediction. The first part, the data layer, is composed of two procedures, namely data preprocessing and separation. In the first procedure, several preprocessing operations are applied. In the next stage, the upper and lower sections of the images are cropped to eliminate unnecessary information; then, resolution reduction is applied to each frame for computational reasons. The size is reduced from 455 × 256 pixels to 200 × 66 pixels. This dimension is equal to the one used as input for the PilotNet[2]. Finally, three consecutive images are stacked and used as the input frame dimension for the deep neural network. The steering angle associated with each tensor is the one related to the last frame.

The second procedure, called data separation, is used to split the dataset, and a usual machine learning principle is applied. The dataset is divided into training, validation and test sets in order to learn, optimize and test the final intelligent system.

BO_ST-LSTM layer is responsible for the optimization and learning phases of the BO_ST-LSTM system. Training and validation sets are used to optimize the ST-LSTM network by means of Bayesian optimization[17]. This procedure is useful to develop a more robust architecture suitable for this specific task; then, during the optimization phase, an early stopping with patience equal to 5 is used to avoid overfitting and reduce the overall computation time. Given the optimized setting, the ST-LSTM is trained and its performance is measured on the test set.

Bayesian optimization. Bayesian optimization is a sample-efficient strategy for global optimization of black-boxes, expensive and multi-extremal functions, traditionally constrained over a box-bounded search space Ω:

$$\min_{\theta \in \Omega} g(\theta) \tag{1}$$

To solve such problem (1), BO uses two key components: a *probabilistic surrogate model* of the objective function $g(\theta)$ and an *acquisition function* (also called *infill criterion* or *utility function*) that is based on the current approximation of $g(\theta)$. The optimization of the acquisition function allows to select the next promising θ' where to evaluate the objective function. The observed value, $g(\theta')$ (or $g(\theta') + \varepsilon$ in the case that the objective function is also *noisy*), is then used to update the probabilistic model approximating $g(\theta)$ and the process is iterated until a given termination criteria is reached (e.g., a maximum number of function evaluations).

A Gaussian Process (GP)[44] is the most common choice for the probabilistic surrogate model. An alternative is offered by Random Forest (RF)[45], an ensemble learning method which, contrary to GP, is able—by construction—to deal with a complex search spaces Ω, spanned by mixed, categorical and conditional components of (θ). Conditional means that the value of a component of the solution vector $\theta_{[i]}$ depends on the value of at least another component $\theta_{[j]}$, with $i \neq j$.

Regardless its specific implementation, the aim of the probabilistic surrogate model is to provide an estimate (aka prediction) of $g(\theta), \forall\theta \in \Omega$, along with a measure of uncertainty about such estimate. These two elements are usually the mean and standard deviation of the prediction provided by the probabilistic surrogate model, denoted by $\mu(\theta)$ and $\sigma(\theta)$ respectively.

The acquisition function is aimed at driving the selection of the next θ' to be evaluated on the objective function, balancing between *exploitation*—that is choosing θ' whose associated prediction is not worse than the best function value observed so far—and *exploration*—that is choosing θ' whose prediction is largely uncertain. While

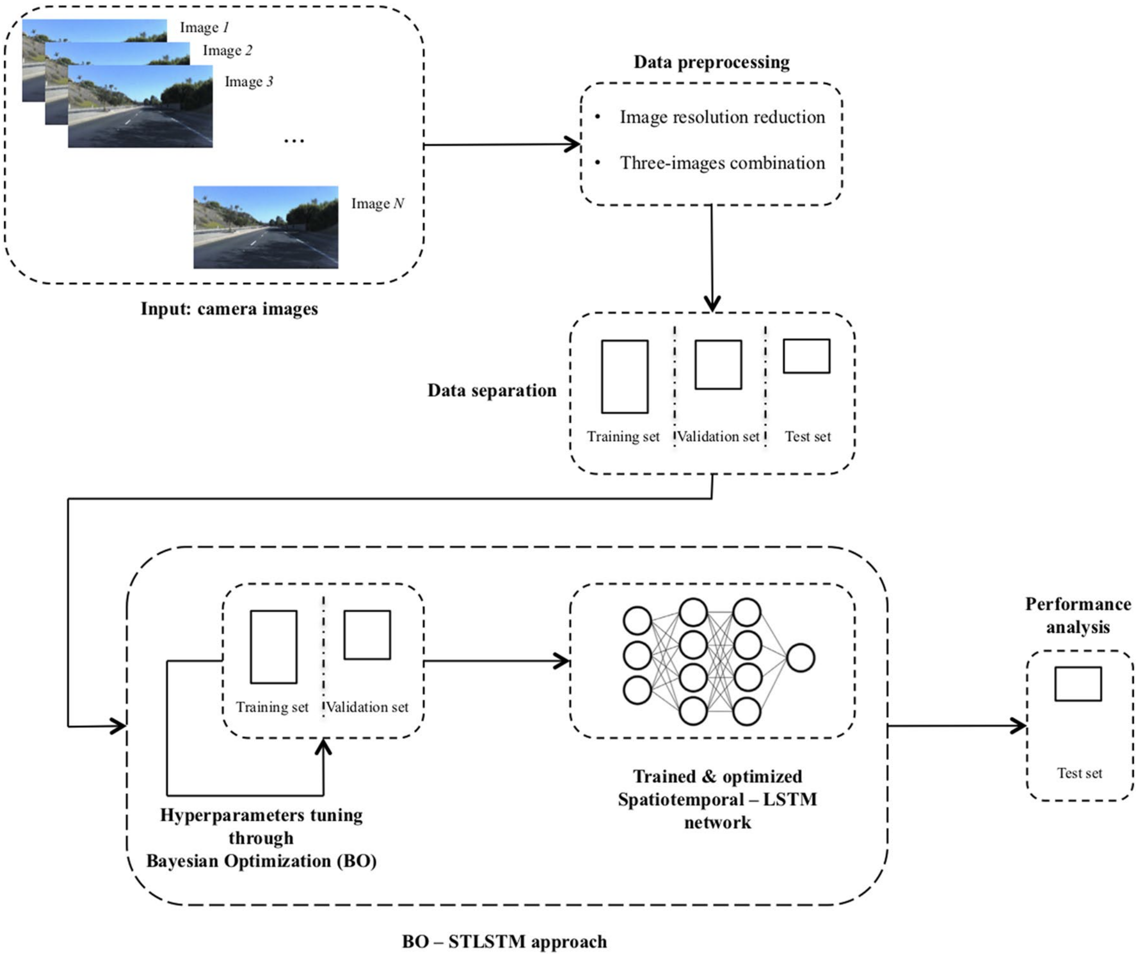

Figure 1. Overview of the BO-STLSTM system. Images in the *Input: camera images* box come from the SullyChen Dataset[43].

exploitation is associated to *local search*, exploration is associated to *global search*: the first is significantly driven by $\mu(\theta)$ while the second is significantly driven by $\sigma(\theta)$. Several acquisition functions have been proposed—an overview is provided in Frazier[17] and Archetti et al.[18]—each one offering a different mechanism to balance the exploitation-exploration trade-off. The most widely used acquisition functions are lower confidence bound (LCB), expected improvement (EI) and maximum probability of improvement (MPI).

Lower confidence bound (LCB) is an acquisition function that manages exploration-exploitation by being optimistic in the face of uncertainty:

$$LCB(\theta) = \mu(\theta) - \xi\sigma(\theta), \tag{2}$$

where $\mu(x)$ and $\sigma(x)$ are mean value and standard deviation of the probabilistic surrogate model. $\xi \geq 0$ is the parameter to manage the trade-off between exploration and exploitation. More precisely, $\xi = 0$ is for pure exploitation; on the contrary, higher values of ξ emphasizes exploration. In Srinivas et al.[46] a schedule of ξ is proposed with convergence proof.

Expected improvement (EI) measures the expectation of the improvement on $g(\theta)$ with respect to the predictive distribution of the probabilistic surrogate model.

$$EI(\theta) = \begin{cases} (g(\theta^+) - \mu(\theta) - \xi)\Phi(Z) + \sigma(\theta)\phi(Z) & \text{if} \quad \sigma(\theta) > 0 \\ 0 \quad \text{if} \quad \sigma(\theta) = 0 \end{cases} \tag{3}$$

$g(\theta^+)$ is the best value of the objective function observed so far, ξ is used to balance between exploration-exploitation, $\phi(Z)$ and $\Phi(Z)$ are the probability distribution and the cumulative distribution of the standardized normal, respectively, with Z defined as follows:

$$Z = \begin{cases} \frac{g(\theta^+)-\mu(\theta)}{\sigma(\theta)} & \text{if} \quad \sigma(\theta) > 0 \\ 0 \quad \text{if} \quad \sigma(\theta) = 0 \end{cases} \tag{4}$$

Maximum probability of improvement (MPI) was the first acquisition function proposed in literature. As it is basically biased towards exploitation, it has been recently modified by including a parameter ξ to allow for a better exploration-exploitation trade-off:

$$MPI(\theta) = P(g(\theta) \leq g(\theta^+) + \xi) = \Phi\left(\frac{g(\theta^+) + \mu(\theta) + \xi}{\sigma(\theta)}\right) \tag{5}$$

At last, selecting θ' requires to solve an auxiliary optimization problem on the same search space Ω but computationally cheaper than (1), that is minimizing $LCB(\theta)$ or maximizing $EI(\theta)$ or $MPI(\theta)$.

Denote with $D_{1:n}$ a set of initial solutions (i.e., initial design), for example sampled by using Latin Hypercube Sampling (LHS) technique. The element D_i is $(\theta_i, g(\theta_i))$, with $i = 1, ..., n$ (i.e., we are considering the noise-free setting, without loss of generality). Furthermore, consider N as the maximum number of function evaluations. Then, the general BO algorithm is summarized as follows:

Algorithm 1: Bayesian Optimization algorithm

```
initialize $D_{1:n}$
while $n < N$ do
    update the probabilistic surrogate model depending on $D_{1:n}$:
    update $\mu_n(\theta)$ and $\sigma_n(\theta)$
    compute the updated acquisition function $\alpha_n(\theta)$
    $\theta_{n+1} \leftarrow \arg\max_{\theta \in \Omega} \alpha(\theta)$
    $n \leftarrow n+1$
end
return $\theta^+$, that is the point associated to the best value observed:
$\theta^+ : g(\theta^+) = \min\{g(\theta_1), ..., g(\theta_N)\}$
```

Spatiotemporal-long short term memory network. The ST-LSTM network[14] is based on Convolutional LSTM (ConvLSTM) layers, 3D Convolutional (3DConv) layers and Fully Connected (FC) layers with the aim to extract the spatiotemporal features of time-serial image segment. Bai et al.[14] proposed architecture with 4 ConvLSTM layers combined with a batch normalization step so as to prevent the eventuality of low training efficiency caused by data distribution offset in deep networks.

ConvLSTM uses multi-frame picture segments as input. In this way, spatial features can be extracted like a convolutional layer and the timing relationship can be obtained at the same time.

The four ConvLSTM layers are then used to extract the temporal hidden feature information and help the model learn more effectively from the data. Following the definition of Shi et al.[12], all the inputs $\mathcal{X}_1, \dots, \mathcal{X}_t$, cell outputs $\mathcal{C}_1, \dots, \mathcal{C}_t$, hidden states $\mathcal{H}_1, \dots, \mathcal{H}_t$, and input and forget and output gates (i_t, f_t, o_t) are 3D tensors whose last two coordinates are spatial dimensions. The key equations of ConvLSTM are shown in Eq. (6), where $*$ denotes the convolution operator and $\circ$ denotes the Hadamard product:

$$\begin{aligned}
i_t &= \sigma(W_{xi} * \mathcal{X}_t + W_{hi} * \mathcal{H}_{t-1} + W_{ci} \circ \mathcal{C}_{t-1} + b_i)\\
f_t &= \sigma(W_{xf} * \mathcal{X}_t + W_{hf} * \mathcal{H}_{t-1} + W_{cf} \circ \mathcal{C}_{t-1} + b_f)\\
\mathcal{C}_t &= f_t \circ \mathcal{C}_{t-1} + i_t \circ \tanh(W_{xc} * \mathcal{X}_{\sqcup} + W_{hc} * \mathcal{H}_{t-1} + b_c)\\
o_t &= \sigma(W_{xo} * \mathcal{X}_t + W_{ho} * \mathcal{H}_{t-1} + W_{co} \circ \mathcal{C}_{t-1} + b_o)\\
\mathcal{H}_t &= o_t \circ \tanh(\mathcal{C}_t)
\end{aligned} \tag{6}$$

where σ is the activation function (*i.e.* sigmoid function), b_i, b_f, b_c and b_0 are the biases related to $i_t, f_t, \mathcal{C}_t$ and o_t.

If we view the states as the hidden representations of moving objects, a ConvLSTM with a larger transitional kernel should be able to capture faster motions while one with a smaller kernel can capture slower motions[12].

3DConv is used to better capture the temporal and spatial characteristics of videos. In fact, the traditional 2D convolutional layer operates on time-serial images using a simple convolutional layer to identify each frame of the video. Nevertheless, this method does not take into account the inter-frame motion information in the time dimension.

The convolution operation in 3DConv has a time dimension of three, meaning that the input is composed of three consecutive frames of images, compliantly with the data pre-processing adopted in Bai et al.[14]. In this structure, each feature map in the convolutional layer is connected to multiple adjacent successive frames in the previous layer, thus capturing motion information.

Experiments

The proposed methodology is evaluated and compared with the classical end-to-end driving models on the public dataset SullyChen Dataset[43]. The data were recorded around Rancho Palos Verdes and San Pedro, California, using a 2014 Honda Civic. The temporal resolution of images is 0.05 seconds. A first version of the dataset has been used by Qian et al.[47] on their work. Figure 2 shows some examples. The dataset contains both straight and

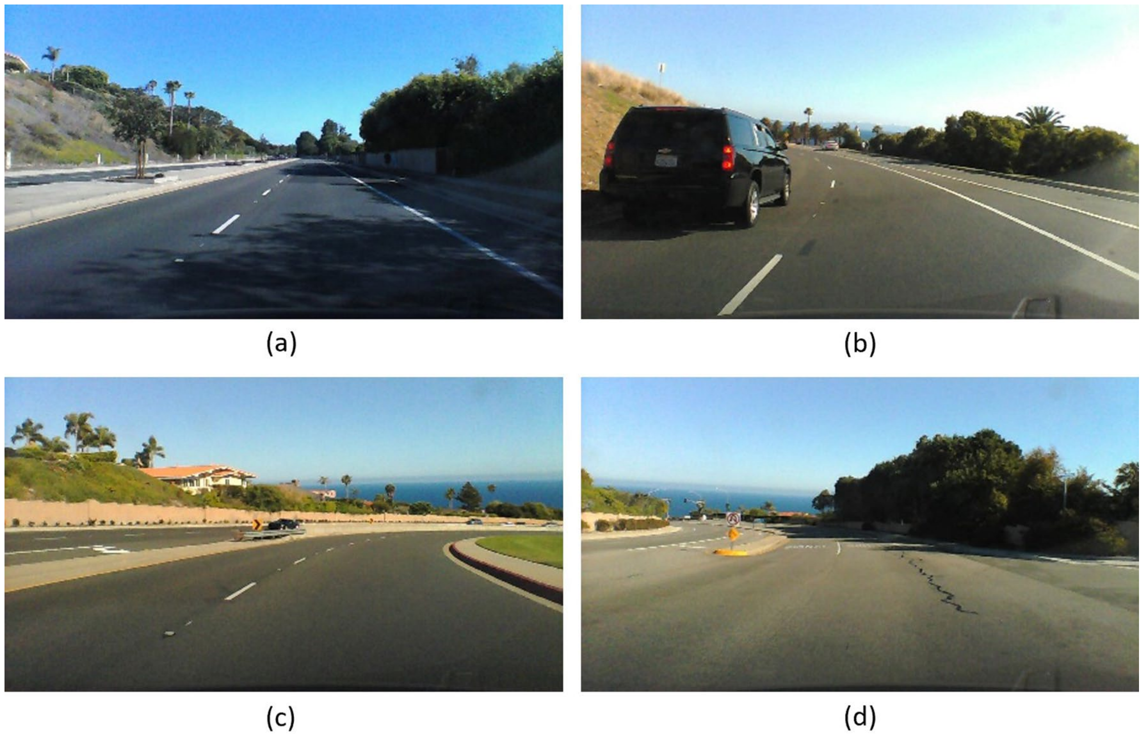

Figure 2. Images from the SullyChen Dataset[43]: (**a**) straight road with trees and shadow, (**b**) road with left curve with obstacle (car), (**c**) road with right curve (**d**) road with a gradual curve to the left with traffic island.

mixed roads. Furthermore, pictures are taken on junctions, primary and secondary roads. For more examples, see also Qian et al.[47].

In this work, we used the updated version which includes 63,000 images of the frontal road as well as the steering angles; for computational reasons, we decided to use around the 62% (39,000) of the total number of images. The considered dataset is then split in 80% (31,200 images) and 20% (7800). The first subset is again divided following the same percentage (80–20%) in order to get training and validation set. Images are shuffled during the training phase. Originally, images come with a dimension of 455 $\times$ 256 pixels; in this study, however, the size has been reduced to 200 $\times$ 66 pixels.

To ensure reproducibility of experiments, we shared both data and our code as a public GitLab project, at the following link: https://gitlab.com/ub-dems-public/cs-labs/user-ariboni/csp-drive-dl.

Methodology. Three different architectures were implemented and trained for 15 epochs, with batch size equal to 50 and validated with different setting according to their definitions: PilotNet[3], J-net[4] and modified version of the ST-LSTM proposed by Bai et al.[14]. Table 1 summarizes settings for all networks.

PilotNet is a deep learning network mainly based on convolutional layers. More precisely, the network consists of 9 layers, including a normalization layer, 5 convolutional layers and 3 fully connected layers. The convolutional layers were designed to perform feature extraction and were chosen empirically through a series of experiments that varied layer configurations (see Bojarski et al.[3] for more details). Strided convolutions were used in the first three convolutional layers with a 2×2 stride and a 5×5 kernel and a non-strided convolution with a 3×3 kernel size in the last two convolutional layers. The five convolutional layers are followed by three fully connected layers leading to an output control value (the steering angle).

As the previous network, J-Net is basically a convolutional neural network. In this case, the network consists of 5 layers, including a normalization layer, 3 convolutional layers and one fully connected layer. In order to reduce the size of the deep neural network layers, the authors applied three max-pooling operators, one after each convolutional layer. The size of all max-pooling operators is 2×2. Eventually, the last layer of the J-net is a fully-connected layer composed of ten nodes and followed by a simple output layer for the steering angle prediction.

Differently from PilotNet and J-Net, ST-LSTM network is composed of a series of ConvLSTM layers and a 3DConv layer. The deep network proposed in Bai et al.[14] consists of 6 layers, including 4 ConvLSTM layers, one 3DConv layer and one fully connected layer followed by the final output layer. Each ConvLSTM layer is followed by a batch normalization step. In order to reduce dimensionality, we introduced in this work a max-pooling layer operating of size $2 \times 2 \times 2$ on 3D input. Furthermore, we added a dropout regularization technique before the output prediction to avoid overfitting. This latter architecture is improved by means of Optimization[17] to propose a more robust solution. This procedure is performed with three different acquisition functions and, in order to limit the impact of the random component, we executed 10 runs for each function. Details of this process are provided in the next sections.

PilotNet	J-Net	ST-LSTM
Normalization layer	Normalization layer	Normalization layer
Conv2D	Conv2D	ConvLSTM2D
Conv2D	MaxPooling2D	BatchNormalization
Conv2D	Conv2D	ConvLSTM2D
Conv2D	MaxPooling2D	BatchNormalization
Conv2D	Conv2D	ConvLSTM2D
Flatten	MaxPooling2D	BatchNormalization
Dense	Flatten	ConvLSTM2D
Dense	Dense	BatchNormalization
Dense		Conv3D
		MaxPooling3D
		Flatten
		Dense
Output value	Output value	Output value

Table 1. Setting for all networks.

Parameters name	Domain space	Domain type
ConvLSTM$_1$: Num. feature maps	$\{4, 8, 10, 16\}$	Discrete
ConvLSTM$_2$: Num. feature maps	$\{4, 8, 10, 16\}$	Discrete
ConvLSTM$_3$: Num. feature maps	$\{4, 8, 10, 16\}$	Discrete
ConvLSTM$_4$: Num. feature maps	$\{4, 8, 10, 16\}$	Discrete
Conv3D: Num. feature maps	$\{1, 2, 3\}$	Discrete
FC: Num. of neurons	$\{5, 10, 25, 50\}$	Discrete
Dropout	$\{0.0, 0.5\}$	Continuous
Learning rate (Adam optimizer)	$\{0.01, 0.001, 0.0001, 0.00001\}$	Discrete

Table 2. Parameters of the ST-LSTM architecture for Bayesian optimization.

The developed architectures are trained using stochastic gradient descent. The algorithm seeks to update the weights of the model to reduce the error between actual and estimated values. In order to compute this error, it is necessary to choose a suitable loss function; as such, having to handle a regression problem, we resolved to use the mean squared error (MSE). As reported in Eq. (7), this measure is given by the average of the square of the difference between actual y_i and estimated values $\hat{y}_i$.

$$MSE = \frac{1}{n}\sum_{t=1}^{n}(y_i - \hat{y}_i)^2 \tag{7}$$

BO_LT-STM: optimizing the hyperparameters of an ST-LSTM. As described in "Proposed method" section, Bayesian optimization is a sample-efficient strategy for global optimization of black-boxes. BO uses probability to find a minimum of an objective function in order to obtain better performance in the testing phase and reduce the optimization time. For this specific task, we have used the mean squared error on the validation set as the objective function to be minimized. Also, this procedure is executed on eight different hyper-parameters of the ST-LSTM architecture. Table 2 shows the parameters considered with the relative domain spaces.

The BO process is performed through the Python suite named GPyOpt and by using the Gaussian Process (GP) as a probabilistic surrogate model and through three different acquisition functions: lower confidence bound (LCB), expected improvement (EI), and maximum probability of improvement (MPI). As far as the acquisition functions' hyperparameter is concerned (i.e., ξ), we have adopted default values suggested by the GPyOpt library.

We compared the obtained results in order to identify the most suitable acquisition function for the optimization process. A set of initial random solutions composed of five configurations is defined to update the surrogate model at each execution. The optimization process starts from these configurations and lasts 20 iterations. The acquisition function selects the new configuration to be evaluated at each run by managing the trade-off between exploration and exploitation. Therefore, in order to limit the randomness's impact in the choice of initial solutions, we have performed each optimization process with the relative acquisition function for ten times, with different seeds.

Figure 3 shows the evolution of the so-called *best seen*, that is the minimum validation error value observed during the BO iterations in this study. Solid lines represent the average, while shaded areas are standard deviation

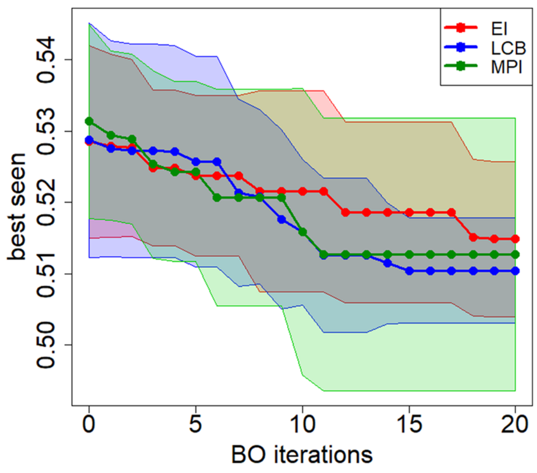

Figure 3. Comparison among GP-based BO processes using three different acquisition functions.

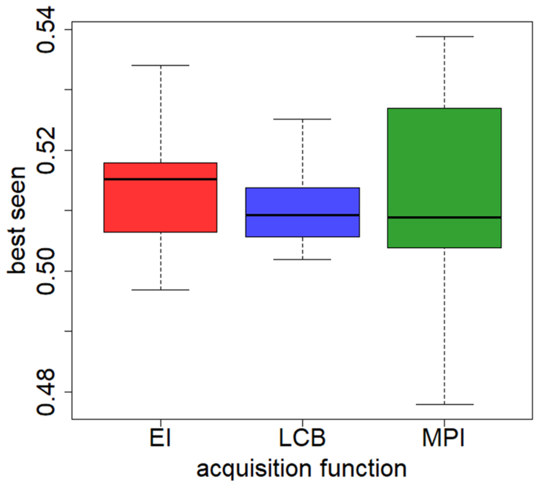

Figure 4. Comparison between the distribution of the *best seen* obtained from the different acquisition function.

over the ten experiments. At the first iteration, the *best seen* is the minimum validation error observed on the initial random configurations; then, the *best seen* of the successive 20 evaluated configurations is reported at a later stage.

All of the three acquisition functions allow, on average, to reduce the error by about two percentage points during the BO process. All of them, in fact, lead to identifying configurations with a similar error value. Since the average results obtained are similar, the standard deviation of the various *best seen* values at the end of BO was considered to identify the most performing acquisition function. As shown in Fig. 4, the *best seen* value determined by LCB has a lower standard deviation compared to the other two acquisition functions. For this reason, ST-LSTM has been re-trained while considering the best configuration obtained through BO with the LCB acquisition function.

Figure 5 displays a graphical representation of the final ST-LSTM optimized architecture.

Results. This section summarizes the most relevant results of the study. The predictive performance indicators selected to compare the developed architectures are the MSE (Eq. 7), the mean absolute error (MAE) and the standard deviation of absolute error (St. AE).

$$AE = \left|\hat{y}_i - y_i\right|, \qquad i = 1, \ldots, n \tag{8}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n} AE_i, \tag{9}$$

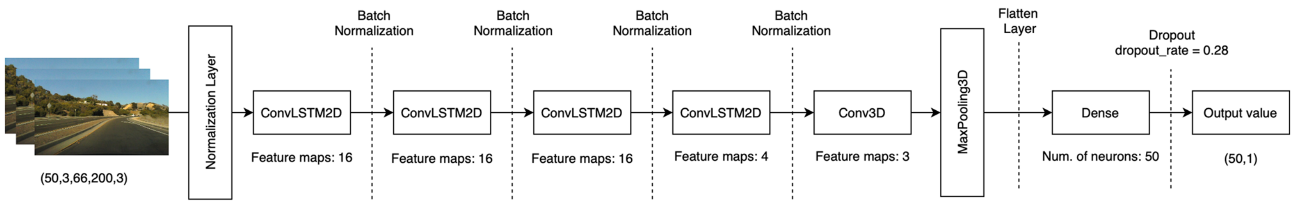

Figure 5. Structure of BO_ST-LSTM architecture. Images used at the beginning of the structure come from the SullyChen Dataset[43].

		PilotNet	J-Net	ST-LSTM	BO_ST-LSTM
Training	MSE	0.0209	**0.0114**	0.0405	0.1831
	MAE	0.0870	**0.0697**	0.1181	0.1971
	St. AE	0.1155	**0.0810**	0.1630	0.3798
Validation	MSE	0.6814	0.5842	0.6139	**0.5019**
	MAE	0.4409	0.4262	0.4710	**0.4042**
	St. AE	0.6979	0.6345	0.6263	**0.5820**

Table 3. Average values of the prediction performance indicators on the training and validation sets for the four architectures. In bold the best value for each indicator.

	Training			Validation		
	MSE	Bias^2	Variance	MSE	Bias^2	Variance
PilotNet	0.0209	0.0004	0.0205	0.6814	0.0350	0.6464
J-Net	0.0114	0.0002	0.0112	0.5842	0.0440	0.5402
ST-LSTM	0.0405	0.0001	0.0404	0.6139	0.0755	0.5384
BO_ST-LSTM	0.1831	0.0002	0.1829	0.5019	0.0130	0.4881

Table 4. Bias-variance tradeoff decomposition.

$$St.\,AE = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}\left(AE_i - MAE\right)^2} \tag{10}$$

where $\hat{y}_i$ is the ith predicted value of the response system, y_i is the ith actual steering wheel angle and n is the number of observations in the evaluated set (validation or test set). Table 3 summarizes the overall results on training and validation sets. BO_ST-LSTM is the best approach in mainly all the indicators on the validation set. All approaches show a possible problem related to overfitting, indeed all of them obtained really good results on the training set that are not confirmed on the validation set. For this reason, the bias-variance trade-off is investigated in what follows.

Table 4 separately reports the MSE on training and validation, for each one of the four Deep Neural Network (DNN) models considered. Then, we have decomposed the MSE into bias and variance in order to investigate possible differences in the balance offered by the four models. More precisely, bias and variance are estimated as follows, under the noise-free assumption (i.e., if $x_i = x_j$ then $y_i = y_j$):

$$Bias^2 = \left(\frac{1}{N}\sum_{i=1}^{N}\hat{y}_i - \frac{1}{N}\sum_{i=1}^{N}y_i\right)^2 \tag{11}$$

$$Variance = \frac{1}{N}\sum_{i=1}^{N}\left((y_i - \hat{y}_i) - \overline{err}\right)^2 \tag{12}$$

with $\overline{err} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)$. Finally, $MSE = Bias^2 + Variance$.

A DNN model should simultaneously achieves low variance and low bias in order to minimize the MSE. Generally speaking, if we assume f as an unknown real function and $\hat{f}$ as the estimated function on a training sample then we can define the meaning of variance and bias. Variance refers to the amount by which $\hat{f}$ would change considering a different training set. Bias is related to the error that is introduced by approximating f by using a simpler model. The considered DNN models share a common pattern looking at bias and variance: all of them

		PilotNet	J-Net	ST-LSTM	BO_ST-LSTM
Training	MSE	0.2917	0.0159	0.0442	0.0204
	MAE	0.2464	0.0793	0.1187	0.0888
	St. AE	0.4806	0.0982	0.1734	0.1117
Test	MSE	0.2810	0.3204	0.2758	0.2700
	MAE	0.3781	0.3918	0.3866	0.3848
	St. AE	0.3715	0.4086	0.3555	0.3492

Table 5. Average values of the prediction performance indicators on the training and test sets.

have low bias and high variance. This is an evidence that we are considering a set of models that tend to overestimate the training set. As a consequence, small changes in the training set can result in significant changes in $\hat{f}$.

Focusing on MSE indicator, BO_ST-LSTM is the most promising model, as it provides the lowest MSE value on validation. This was quite expected because the goal of BO was to minimize this indicator. Moreover, BO_ST-LSTM resulted in the highest MSE on the training set (one order of magnitude greater than the other three DNN models), making it less prone to overfitting. Finally, as already pointed out, MSE is basically made up of the variance component for all the DNN models, both in training and validation. However, BO_ST-LSTM has the lowest variance MSE component in the validation set. In addition, both bias and variance are reduced using the BO strategy with respect to the other approaches.

Assuming that we can select only one DNN model to be deployed in a real-life application, our choice would be BO_ST-LSTM. This conclusion is also motivated by a pairwise Mann-Whitney U test performed on the prediction errors of the four models on the validation set. More specifically, prediction error for BO_ST-LSTM is significantly lower than the other three models (p-value < 0.001); ST-LSTM and PilotNet are significantly similar in terms of prediction error on the validation set (p-value $= 0.866$), as well as PilotNet and J-Net (p-value $= 0.054$). Finally, ST-LSTM and J-Net resulted significantly different in terms of prediction error on the validation set (p-value $= 0.004$).

All the approaches were then re-trained on the dataset consisting of both the training and validation data, setting 15 learning epochs. Results are summarized in Table 5: the inclusion of validation data leads to models with small values of MSE on test, lower than MSE on validation (i.e., a reduction of around 50%). Results on the test set confirm that BO_ST-LSTM is the most performing model, providing the smallest MSE again. Thus, if we could select only one DNN model, depending on the MSE on the validation set, choosing BO_ST-LSTM would be fine. It is important to remark that, although BO_ST-LSTM, ST-LSTM and PilotNet resulted in really close values of MSE on test set, ST-LSTM and PilotNet would be our third and fourth choice respectively.

Conclusion and future works

This paper is meant to address the prediction of the steering wheel angle in a self-driving system via Deep Learning. We started from an ST-LSTM, which can be considered the most suitable model for the specific application targeted, as resulted from the empirical comparison against PilotNet and J-Net. Moreover, the hyperparameters tuning of the ST-LSTM, performed via BO, allowed to decrease further prediction error of this specific DNN architecture, both on the validation and test set. More specifically, GP-based BO with LCB acquisition function proved to be the best hyperparameters optimization strategy.

Nevertheless, some limitations ought to be considered. Although BO is a sample efficient global optimization strategy, running a single hyperparameter optimization process has required—in our experimental setting—to train and evaluate 25 different ST-LSTM models (5 sampled via LHS plus 20 via BO). While this "cost" is compensated by an MSE on validation which is significantly lower than the other DNN models, the difference between the MSE of BO_ST-LSTM and ST-LSTM, on the test set, is quite negligible. Unfortunately, a principled comparison, in terms of computational costs, is not possible because this information is not reported in the papers related to the other DL models considered in the study. Reproducing all the experiments of the other research groups is for sure expensive and—potentially—unfair. This is the reason why the comparison is not possible and, in any case, it is out of the scope of this paper.

Moreover, it is important to remark that results do not indicate ST-LSTM is better than J-Net or PilotNet necessarily, as these two latter models have not been optimized meaning that their optimized model might outperform the BO-ST-LSTM model.

Future works will address (a) the possibility to also optimize the architecture of the ST-LSTM, moving from hyperparameters optimization towards Neural Architecture Search (NAS)—and (b) its formulation as a multi-objective or constrained optimization problem by considering not only MSE but also other requirements, such as jointly minimizing the inference time (multi-objective) or keeping it lower than a fixed threshold (constrained). Furthermore, comparison of other models such as 2D-CNN-LSTM or 3D U-Net along with ST-LSTM models could also provide better insights on which model provides better performance. Besides, the usefulness of the application of other useful modules such as the Spatiotemporal attention module could be assessed.

Received: 24 November 2021; Accepted: 10 May 2022
Published online: 24 May 2022

References

1. Chan, C.-Y. Advancements, prospects, and impacts of automated driving systems. *Int. J. Transp. Sci. Technol.* **6**, 208–2016 (2017).
2. Krizhevsky, A., Sutskever, I. & Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, Vol. 25, 1–9 (2012).
3. Bojarski, M. *et al.* Explaining how a deep neural network trained with end-to-end learning steers a car. *arXiv* http://arxiv.org/abs/1704.07911, 1–8 (2017).
4. Kocić, J., Jovičić, N. & Drndarević, V. An end-to-end deep neural network for autonomous driving designed for embedded automotive platforms. *Sensors* **19**, 1–26 (2019).
5. Li, G., Yang, Y., Qu, X., Cao, D. & Li, K. A deep learning based image enhancement approach for autonomous driving at night. *Knowl.-Based Syst.* **213**, 106617 (2021).
6. Schwarz, B. Lidar: Mapping the world in 3D. *Nat. Photon.* **4**, 429–430 (2010).
7. Yurtsever, E., Lambert, J., Carballo, A. & Takeda, A. A survey of autonomous driving: Common practices and emerging technologies. *IEEE Access* **8**, 58443–58469 (2020).
8. Goodfellow, I., Bengio, Y. & Courville, A. *Deep Learning* (MIT Press Cambridge, 2017).
9. Balderas, D., Ponce, P. & Molina, A. Convolutional long short term memory deep neural networks for image sequence prediction. *Expert Syst. Appl.* **122**, 152–162 (2019).
10. Wang, Y., Long, M., Wang, J., Gao, Z. & Yu, P. S. Predrnn: recurrent neural networks for predictive learning using spatiotemporal lstms. In *In Proceeding of the 31st Conference on Neural Information Processing Systems (NIPS2017)*, 1–10 (2017).
11. Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R. & Schmidhuber, J. Lstm: A search space odyssey. *Trans. Neural Netw. Learn. Syst.* **28**, 2222–2232 (2017).
12. Shi, X. *et al.* Convolutional LSTM network: A machine learning approach for precipitation nowcasting. *Adv. Neural Inf. Process. Syst.* **2015**, 1–12 (2015).
13. Yu, H., Yang, s., Gu, W. & Zhang, S. Baidu driving dataset and end-to-end reactive control model. In *In IEEE Intelligent Vehicles Symposium (IV)*, 341–346 (2017).
14. Bai, Z., Cai, B., ShangGuan, W. & Chai, L. Deep learning based motion planning for autonomous vehicle using spatiotemporal LSTM network. 1–5 (2019).
15. Ji, L., Xu, W., Yang, M. & Yu, K. 3-d convolutional neural networks for human action recognition. In *International Conference on Machine Learning*, 495–502 (2010).
16. Elsken, T., Metzen, J. H. & Hutter, F. Neural architecture search. In *Automated Machine Learning*, 63–77 (2019).
17. Frazier, P. I. *Bayesian Optimization. Tutorial in Operation Research* 255–278 (2018).
18. Archetti, F. & Candelieri, A. *Bayesian Optimization and Data Science* (Springer, 2019).
19. Hutter, F., Kotthoff, L. & Vanschoren, J. *Automated Machine Learning* (Springer, 2019).
20. Bernardo, J. *et al.* Optimization under unknown constraints. *Bayesian Stat.* **9**, 1–19 (2011).
21. Hernández-Lobato, J. M., Gelbart, M. A., Hoffman, M. W., Adams, R. P. & Ghahramani, Z. Predictive entropy search for Bayesian optimization with unknown constraints. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning*, 1699-1707 (2015).
22. Sacher, M. *et al.* A classification approach to efficient global optimization in presence of non-computable domains. *Struct. Multidiscip. Optim.* **58**, 1537–1557 (2018).
23. Bachoc, F., Helbert, C. & Picheny, V. Gaussian process optimization with failures: classification and convergence proof. *HAL id: hal-02100819, version*, Vol. 1 (2019).
24. Candelieri, A. Sequential model based optimization of partially defined functions under unknown constraints. *J. Glob. Optim.* **79**, 281–303 (2019).
25. Nguyen, V., Gupta, S., Rane, S., Li, C. & Venkatesh, S. Bayesian optimization in weakly specified search space. In *2017 IEEE International Conference on Data Mining (ICDM)*, 347–356 (IEEE, 2017).
26. Nguyen, V., Gupta, S., Rana, S., Li, C. & Venkatesh, S. Filtering Bayesian optimization approach in weakly specified search space. *Knowl. Inf. Syst.* **60**, 385–413 (2019).
27. He, X., Zhao, K. & Chu, X. Automl: A survey of the state-of-the-art. *Knowl.-Based Syst.* **212**, 106622 (2021).
28. Liu, Y. *et al.* A survey on evolutionary neural architecture search. *IEEE Trans. Neural Netw. Learn. Syst.* (2021).
29. Wei, C. *et al.* Npenas: Neural predictor guided evolution for neural architecture search. *IEEE Trans. Neural Netw. Learn. Syst.* (2022).
30. White, C., Neiswanger, W. & Savani, Y. Bananas: Bayesian optimization with neural architectures for neural architecture search. In *Proceedings of the AAAI Conference on Artificial Intelligence* Vol. 35, 10293–10301 (2021).
31. Ma, L., Cui, J. & Yang, B. Deep neural architecture search with deep graph Bayesian optimization. In *2019 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, 500–507 (IEEE, 2019).
32. Deng, J., Dong, W., Socher, L.-J., Li, K. L. & Fei-Fei, L. Imagenet: a large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, 248–255 (2009).
33. Russakovsky, O. *et al.* Imagenet large scale visual recognition challenge. *Int. J. Comput. Vis.* **115**, 211–252 (2015).
34. Gidado, U. M. *et al.* A survey on deep learning for steering angle prediction in autonomous vehicles. *IEEE Access* **8**, 163797–163817 (2020).
35. Kim, J. & Canny, J. Interpretable learning for self-driving cars by visualizing causal attention. In *Proceedings of the IEEE International Conference on Computer Vision*, 2942–2950 (2017).
36. Nguyen, H.-P., Liu, J. & Zio, E. A long-term prediction approach based on long short-term memory neural networks with automatic parameter optimization by tree-structured parzen estimator and applied to time-series data of npp steam generators. *Appl. Soft Comput.* **89**, 106116 (2020).
37. Osmani, A. & Hamidi, M. Bayesian optimization of neural architectures for human activity recognition. In *Human Activity Sensing*, 171–195 (Springer, 2019).
38. Abeysirigoonawardena, Y., Shkurti, F. & Dudek, G. Generating adversarial driving scenarios in high-fidelity simulators. In *2019 International Conference on Robotics and Automation (ICRA)*, 8271–8277 (IEEE, 2019).
39. Zerwas, J. *et al.* Netboa: Self-driving network benchmarking. In *Proceedings of the 2019 Workshop on Network Meets AI & ML*, 8–14 (2019).
40. Gangopadhyay, B. *et al.* Identification of test cases for automated driving systems using bayesian optimization. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 1961–1967 (IEEE, 2019).
41. Kong, H., Yan, J., Wang, H. & Fan, L. Energy management strategy for electric vehicles based on deep q-learning using Bayesian optimization. *Neural Comput. Appl.* **32**, 14431–14445. https://doi.org/10.1007/s00521-019-04556-4 (2020).
42. Alizadeh, B. *et al.* A novel attention-based LSTM cell post-processor coupled with Bayesian optimization for streamflow prediction. *J. Hydrol.* **601**, 126526 (2021).
43. Chen, S. Sullychen dataset: driving dataset. https://github.com/SullyChen/driving-datasets (2018) (Accessed 17 Mar 2020).
44. Williams, C. K. & Rasmussen, C. E. *Gaussian Processes for Machine Learning* Vol. 2 (MIT Press Cambridge, 2006).
45. Tin Kam Ho. Random decision forests. In *Proceedings of 3rd International Conference on Document Analysis and Recognition*, Vol. 1, 278–282 (1995).

46. Srinivas, N., Krause, A., Kakade, S. M. & Seeger, M. W. Information-theoretic regret bounds for gaussian process optimization in the bandit setting. *IEEE Trans. Inf. Theory* **58**, 3250–3265 (2012).
47. Qian, D. *et al.* End-to-end learning driver policy using moments deep neural network. In *In Proceedings of the 2018 IEEE International Conference on Robotics and Biomimetics*, 1533–1538 (2018).

Acknowledgements
We greatly acknowledge the DEMS Data Science Lab for supporting this work by providing computational resources.

Author contributions
A.R. developed the software, did experimentation and was involved in writing the original draft, N.G. helped in the software implementation phase, A.C. conceived the experiments and was involved in writing the original draft, M.B. conceived the experiments, supervised the team and was involved in writing the original draft. All authors reviewed the manuscript.

Competing interests
The authors declare no competing interests.

Additional information
Correspondence and requests for materials should be addressed to M.B.