A new neural network feature importance method: Application to mobile robots controllers gain tuning

Ashley Hill, Eric Lucet, Roland Lenain

▶ To cite this version:

Ashley Hill, Eric Lucet, Roland Lenain. A new neural network feature importance method: Application to mobile robots controllers gain tuning. ICINCO 2020, 17th International Conference on Informatics in Control, Automation and Robotics, Jul 2020, Paris, France. cea-03314585

HAL Id: cea-03314585
https://cea.hal.science/cea-03314585

Submitted on 5 Aug 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

A new neural network feature importance method: Application to mobile robots controllers gain tuning.

Ashley Hill[1], Eric Lucet[1] and Roland Lenain[2]
[1] *CEA, LIST, Interactive Robotics Laboratory, Gif-sur-Yvette, F-91191, France*
[2] *Université Clermont Auvergne, Inrae, UR TSCF, Centre de Clermont-Ferrand, F-63178 Aubière, France*
{ashley.hill, eric.lucet}@cea.fr, roland.lenain@inrae.fr

Keywords: Machine Learning, Neural Network, Robotics, Mobile Robot, Control Theory, Gain Tuning, Adaptive Control, Explainable Artificial Intelligence.

Abstract: This paper proposes a new approach for feature importance of neural networks and subsequently a methodology to determine useful sensor information in high performance controllers, using a trained neural network that predicts the quasi-optimal gain in real time. The neural network is trained using the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) algorithm, in order to lower a given objective function. The important sensor information for robotic control are determined using the described methodology. Then a proposed improvement to the tested control law is given, and compared with the neural network's gain prediction method for real time gain tuning. As a results, crucial information about the importance of a given sensory information for robotic control is determined, and shown to improve the performance of existing controllers.

1 Introduction

In robotic control, the search for more accurate and more adaptive controllers have been the foundation of research in the field. Where the controller must be able to adapt to varying known conditions or to react with respect to observed changes in the environment. However, it is not trivial to know a priori which aspects of the perception need to be included into the control law. As such many paths have been and are being explored for improving control law, such as computer vision (Ha and Schmidhuber, 2018) or state estimators and observes (Lenain et al., 2017).

More recent papers have shown the use of neural networks for gain prediction (Hill. et al., 2019). Where the gain changes with respect to the perception quality, allowing the controller to adapt to information that was underused. Indeed, due to their nature as universal function approximators (Hornik et al., 1990), neural networks can use most of the available sensor information; in order to predict a quasi-optimal gain that adapts to the changes in the perception.

However, many tasks cannot use neural networks, for safety reasons or in some cases for performance reasons. Indeed, they are considered black-boxes due to their mathematical complexity and high number of internal parameters (LeCun et al., 2015), making them hard to analyze and predict consistent behavior.

As such, the natural question that follows, is how to analyze trained neural controllers in order to understand their behavior, and possibly improve current control laws. This is an important question in the field of machine learning and artificial intelligence in general, and currently heavily researched (Gunning, 2017). In this paper, the application of existing and novel analysis methods are used, in order to understand the how a trained gain prediction method reacts to its input information, and how this knowledge can be used in order to improve classic control law.

At first, a few methods for neural network analysis will be described, including a novel analysis method. Then, an experimental setup, where the gain prediction method is described, along with a dynamic simulation of a robotic car-like bicycle model. Followed by experiments showing how to integrate the analysis information into improving the control law. And finally, a discussion of the results and in depth analysis before concluding.

2 Analysis Method

2.1 Preliminaries

The analysis is based on *feature importance*, it describes how important each input feature is in order to obtain a good prediction.

This is usually used in the context of decision trees (Liaw et al., 2002), where each node on the tree has a score determining the quality of its split. Which in some cases is the Gini impurity (Suthaharan, 2016). The feature importance is described as the input parameters that lead to a low Gini impurity for each node that use the input feature for its split. When sorted, these feature importance show which inputs where the most useful in order to obtain a good prediction.

Unfortunately, decision trees struggle to outmatch the performance of neural networks, due to neural networks strengths as dimensional reducers and being universal function approximators for non-linear functions (LeCun et al., 2015). This means that in most cases neural networks must be used in order to obtain the desired performance.

The notion of *feature importance* is still available to neural networks, however they are not as clear as for the decision trees. The most known method is the Temporal Permutation method, described in (Molnar, 2019), and detailed in the following section.

2.2 Feature Importance Using Temporal Permutation

Neural network predicts from a vector inputs, a desired output vector. By varying the inputs and observing the change in the output, a correlation can be established between each input and amount of change for the output. This correlation shows if a given input were to change, by how much would the output change.

If the assumption that the neural network predicts a quasi-optimal output is given. Then the change between the original predicted output, and the predicted output when the input was altered, should give the influence each input has on the output. And this is turn gives describes how important each input is to predicting the quasi-optimal output.

However, the changes that are made to each input must be in such a way, that the input values must remain consistent and realistic. For this, the approach proposed in (Molnar, 2019), use a temporal permutation method. Where for a list of input vectors recorded over time and a given input to analyze, the given input will be shuffled across all the input vectors over time. This causes a given input to be randomly permuted over time, meaning the input value will not longer hold any meaning with respect to the other inputs in the input vector. This input will then be similar to a random distribution of the same type as observed in the initial unshuffled input vectors.

When this method is applied to each input, a difference of the output, relative to each input can be achieved. Where the difference of the output is directly translated to the error of the output, if the assumption that the neural network predicts a quasi-optimal output is given.

2.3 Feature Importance Using The Gradient

Feed forward multilayer perceptron neural networks, consist of a sequence of matrix multiplications, adds, and activation functions, from the given input to the given output (LeCun et al., 2015):

$$y = a(b^{(n)} + w^{(n,n-1)}a(...b^{(1)} + w^{(1,0)}X))$$

Where y is the output, X is the input, a is the activation function, $b^{(n)}$ is the bias at the layer n, and $w^{(n,n-1)}$ is the weight matrix between the layer n and the layer $n-1$.

From this, the gradient between the output, and any component of the neural network can be achieved using the chain rule. Indeed this is the exact method that is used in backpropagation (LeCun et al., 2015) for gradient descent in supervised learning methods applied to neural networks.

However, backpropagation is only used to tune the parameter of the neural network in order to minimize an error between the predicted output and a desired output. A different method can be used with the chain rule to calculate the rate of change of the output value with respect to the input vector:

$$\frac{\partial y}{\partial X} = \frac{\partial y}{\partial a}\frac{\partial a}{\partial z^{(n)}}\frac{\partial z^{(n)}}{\partial s^{(n-1)}}\cdots\frac{\partial s^{(1)}}{\partial a}\frac{\partial a}{\partial z^{(1)}}\frac{\partial z^{(1)}}{\partial X}$$

$$\frac{\partial y}{\partial X} = a'(z^{(n)})w^{(n,n-1)}\ldots a'(z^{(1)})w^{(1,0)}$$

A variant of this method was previously used in image modification with neural networks, in order to change an input image that maximized a cost function (Mordvintsev et al., 2015). And for determining a Saliency map in image classifiers (Simonyan et al., 2013).

Using this methods a jacobian matrix between each output component and each input component, can be obtained at each feed forward prediction of the neural network. With this, the average and variance of the rate of change for each input with respect to the output can be achieved over a given task. The contribution that allow the method to return the feature importance, is the assumption that the neural network predicts a quasi-optimal output. Where the rate of change of the output is directly translated to the error of the output. Meaning if a low rate of change for a given input is obtained, then the input does not contribute much to the quasi-optimal output, and as such is not considered to be an important feature for the prediction method.

3 Experimental Setup

Before any analysis of the neural network can be done, a training environment must first be established. For this, a robotic car-like model in a dynamic simulation is used to generate training samples. Which are then used as input to a covariance matrix adaptation evolution strategy (CMA-ES) method (Hansen, 2016) in order to optimize a neural network's weights and biases, to minimize an objective function. This neural network has as a goal to output in real time, the quasi-optimal gains for the steering controller, in a similar fashion to previous works (Hill. et al., 2019).

3.1 Robotic Model & Simulation

The robotic model is a dynamic bicycle model, which takes into account the slide slip angle and lateral forces applied to the rear and front axle. This model is show in the figure 1.

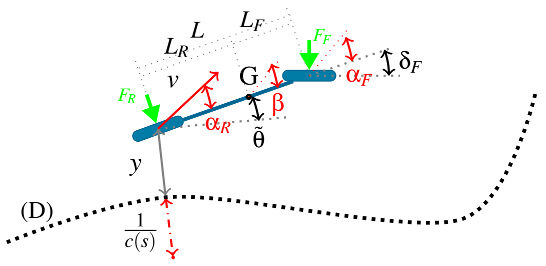

Figure 1: The dynamic robot model.

The notation of the model being defined as follows: (D) is the trajectory being followed, s is the curvilinear abscissa along D, L is the wheel base length of the robot, v is the speed vector of the robot at the middle of the rear axle, $\tilde{\theta}$ is the angular error, y is the lateral error, $c(s)$ is the curvature at the point s, δ_F is the front steering angle, G is the center of mass of the robot, L_R and L_F are the distance from the center of mass to the rear and front axle respectively, F_R and F_F are the lateral force on the rear and front axle respectively, β is the vehicle sliding angle, α_F and α_R are the front and rear axle sliding angle respectively, I_z is the moment of inertia across the Z axis.

From this modeling, the following system of equations can be derived:

$$\left\{\begin{array}{lcl}
\dot{s} & = & v\,\frac{\cos(\tilde{\theta})}{1-c(s)y} \\
\dot{y} & = & v\,\sin(\tilde{\theta}) \\
\ddot{\theta} & = & \frac{1}{I_z}\left(-L_F F_F\cos(\delta_F)+L_R F_R\right) \\
\dot{\beta} & = & -\frac{1}{v_2 m}\left(F_F\cos(\beta-\delta_F)+F_R\cos(\beta)\right)-\dot{\theta} \\
\beta_R & = & \arctan(\tan\beta-\frac{L_R\dot{\theta}}{v_2\cos(\beta)}) \\
\beta_F & = & \arctan(\tan\beta+\frac{L_F\dot{\theta}}{v_2\cos(\beta)})-\delta_F \\
v_2 & = & \frac{v\cos(\beta_R)}{\cos(\beta)}
\end{array}\right.$$

The lateral forces applied to the rear and front axle, follow the Pacejka magic formula (Bakker et al., 1987), in order to obtain a more realistic and dynamic environment.

An extended Kalman filter (Welch and Bishop, 1995) is used in order to determine the robotic state from the sensor input, along with the covariance.

The steering actuators are modeled using an action delay of approximately $0.5s$.

This robot's steering is controlled using the following control equation:

$$\delta_F = \arctan\left(\frac{L\cos^3\varepsilon_\theta}{\alpha}\left(k_\theta(e_\theta)+\frac{\kappa}{\cos^2(\varepsilon_\theta)}\right)\right)$$

with $e_\theta = \tan\varepsilon_\theta - \left(\frac{k_l\,\varepsilon_l}{\alpha}\right)$ is the relative orientation error of the robot to reach its trajectory (i.e. ensuring the convergence of ε_l to 0). This control detailed in (Lenain et al., 2017) guaranties the stabilization of the robot to its reference trajectory, providing a relevant choice for the gains k_θ and k_l.

3.2 Gain Prediction Model & Training

The gain prediction model used, is based on previous works (Hill. et al., 2019), where a neural networks predicts the quasi-optimal gain that minimize a given objective function. This neural network predict this gain using information that is underused in the control loop, such as the perception quality. This neural network is then optimized in order to minimize the objective function using the CMA-ES method (Hansen, 2016). The full control loop is shown on the figure 2:

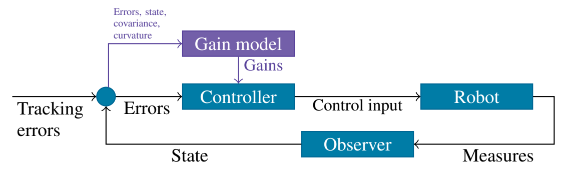

Figure 2: The control loop of the mobile robot steering task, with a gain predictor

The objective function used is the following:

$$ob_1 = \frac{1}{T}\sum_{n=0}^{N}\left[|y(t_n)| + L|\tilde{\theta}(t_n)| + k_{\text{steer}}L|\delta_F(t_n)|\right] dt \quad [m]$$

Where T is the total time taken to follow the path, N is the number of measured timesteps, t_n is the time at the timestep n, y is the lateral error, $\tilde{\theta}$ is the angular error, δ_F is the front steering, and dt is the time step between two samples. k_{steer} is set to 0.5, as it showed to be the ideal compromise between minimizing the control errors, and keeping a low steering energy which minimizes oscillations.

3.3 Feature Importance for robotic control

In robotics, there can be many sensors that can measure a wide variety of different kinds information. However, it is not always clear what kind of information or sensors will be of use in order to develop a performant control equation.

As such, the goal of the following experiments, is to use a trained neural network that can predict the quasi-optimal gain. And from this, determine which inputs of the neural network are used the most in order to minimize the given objective function.

This will give a list of features that are needed in order to improve the performance of the control equation. Some of which will then be integrated into the control law.

4 Results

The following results were obtained using the previously described methods, with a line following task and the trajectory shown in the figure 3, at $2.0m.s^{-1}$, $1.5m.s^{-1}$, and $1.0m.s^{-1}$. Midway though the trajectory, a GPS noise of $1m$ is applied to simulate a perception quality loss. Two change lanes occurs along the trajectory, in order to simulate a GPS constellation jump or a sudden change in the target set-point.

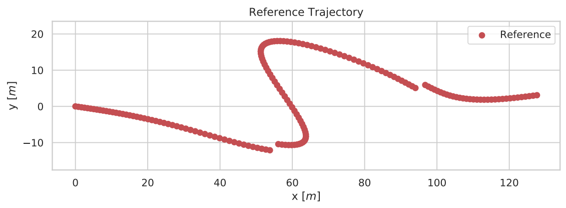

Figure 3: The trajectory

4.1 Baseline

The baseline method that will be used to compare in the following sections, is the expert tuned constant gain method. Using a gain of $k_l = 0.2$, and $k_\theta = 1.0$.

The results for the experiment can be see in the figure 4. Where the baseline method reached an end objective function value of $0.389m$, $0.336m$, and $0.308m$ for $2.0m.s^{-1}$, $1.5m.s^{-1}$, and $1.0m.s^{-1}$ respectively.

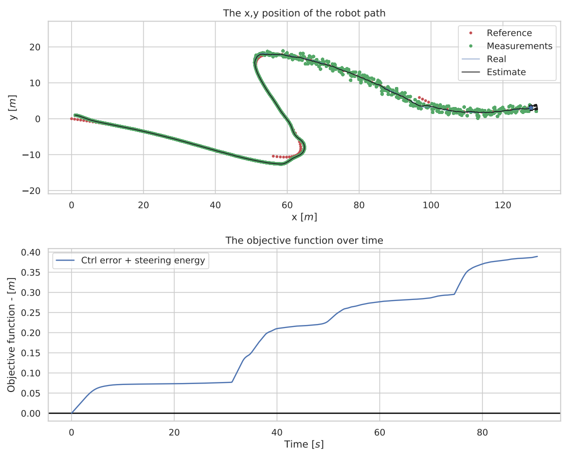

Figure 4: Above: The method line following. Below: the objective function over time.

This method has some obvious shortcomings, as it is not adaptive to the changes in speed or sensor accuracy. Which can be observed in the noisy region, where the controller is reacting to the GPS noise.

4.2 Gain Adaptation

The neural network gain method uses the information in the control loop, in order to minimize the objective function. As such, it is able to predict the quasi-optimal gain along the trajectory at any given time.

The results for the experiment can be see in the figure 4. Where the neural network gain method reached an end objective function value of $0.372m$, $0.325m$, and $0.291m$ for $2.0m.s^{-1}$, $1.5m.s^{-1}$, and $1.0m.s^{-1}$ respectively.

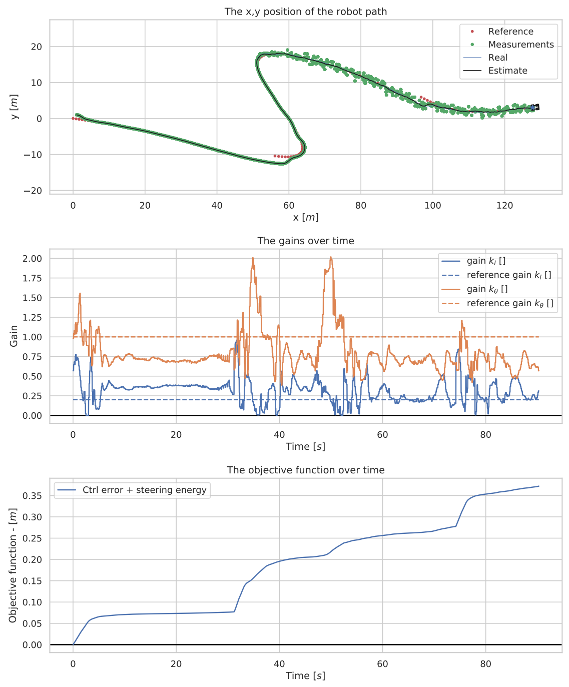

Figure 5: Above: The method line following. Middle: The predicted gain, where the reference gain is the baseline constant gain. Below: the objective function over time.

The neural network is adapting the gain with respect to changes in the speed, sensor accuracy, curvature, and error. This allows the method to lower it's objective function substantially when compared to the baseline method.

The feature importance can now be done over the trained neural network. For this, both the temporal permutation feature importance and the novel gradient feature importance are done.

On figure 6, the temporal permutation feature importance can be observed. It is shown as the absolute mean difference between the original predicted gain, and the predicted gain when the given input is shuffled over time. From this, the most important features from highest to lowest are the Kalman covariance matrix denoted C_{xy}, the speed and target speed denoted v and v_{target} respectively, the lateral error denoted y, the angular error denoted $\tilde{\theta}$, and the curvature denoted $c(s)$.

The temporal permutation feature importance describes the absolute change in the gain, with respect to the input feature. However is it does not show the rate of change of the gain, with respect to the values of the input features. For this the gradient feature importance is used.

On figure 7, the gradient feature importance can be observed. It is shown as the mean and the variance of the jacobian matrix of the gain for each input. Similarly to the temporal permutation feature importance, the most important features from highest to lowest are the Kalman covariance matrix denoted C_{xy}, the speed and target speed denoted v and v_{target} respectively, the lateral error denoted y, the angular error denoted $\tilde{\theta}$, the curvature denoted $c(s)$, and the future curvature (the predicted curvature at $t+1s$) denoted future $c(s)$

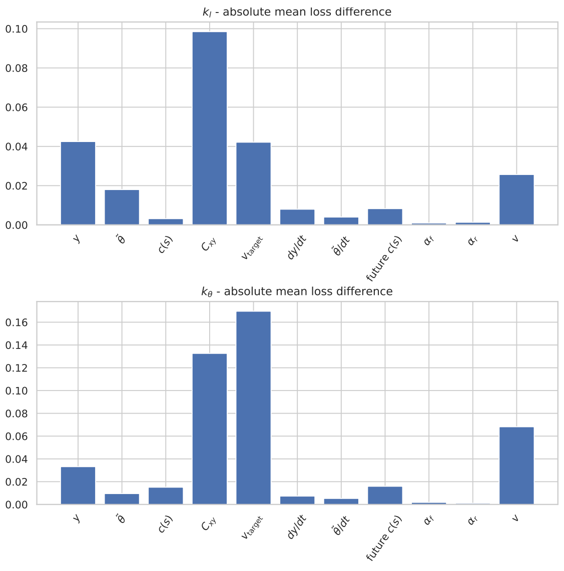

Figure 6: The temporal permutation feature importance. Above: for the k_l gain. Below: for the k_θ gain.

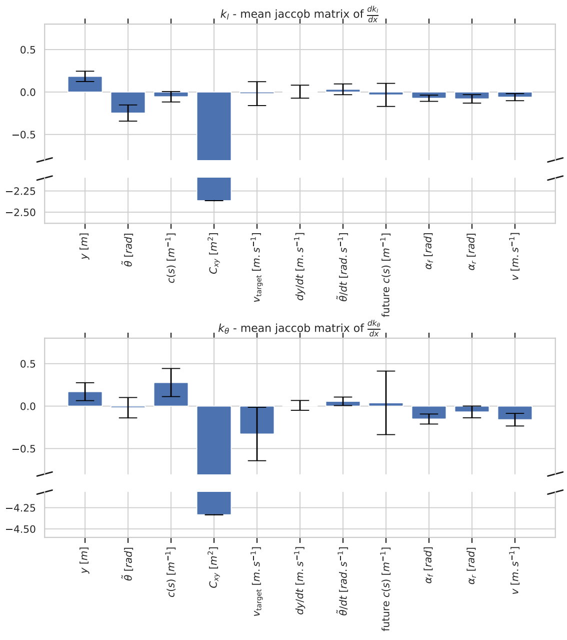

Figure 7: The gradient feature importance, as the mean jacobian matrix. The figure is cut midway to avoid scaling issue of outliers. The error bars, show the variance of the jacobian matrix. Above: for the k_l gain. Below: for the k_θ gain.

due to its high variance.

From this, the equivalence of the feature importance methods can be implied. However the gradient feature importance has some strong strengths to it: It does not need each input features to vary in order to obtain the feature importance. Indeed if a given input does not explore the span of values during the analysis, the temporal permutation feature importance will not return the correct feature importance. Furthermore the gradient feature importance can return the jacobian matrix for each input and output, at each feed forward of the neural network. Allowing for real time analysis and approximate prediction of the behavior of the neural network. And finally, the mean jacobian matrix for each input and output can be used, as an approximation of the neural network's behavior, that can be exploited to improve the control equation, as shown in the following section.

4.3 Improving Control Law

The ideal way to change the control law, is to derive it from the model while taking some of the important features into account (covariance, speed, sliding angles, ...). However, in this case study a simple modification of the gain will be used in order to show a proof of concept.

For this, the gains equation are augmented using the mean jacobian $\frac{\partial y}{\partial X}$ matrix for the speed. In this case study the covariance is not used, due to how the neural network is using it for non-linear adaptation of the gain. As such a linearization of the jacobian is not a valid approximation of the original gain behavior for the covariance, and in which during experimentation lead to unstable behavior.

Using the mean jacobian matrix, the following gain equations are derived:

$$k_l = v\frac{\partial \hat{k_l}}{\partial v} + k_{lv}$$

$$k_\theta = v\frac{\partial \hat{k_\theta}}{\partial v} + k_{\theta v}$$

Where $\frac{\partial \hat{k_l}}{\partial v}$, $\frac{\partial \hat{k_\theta}}{\partial v}$ are the mean jacobian of the predicted gains with respect to the speed, v is the longitudinal speed, and k_{lv}, $k_{\theta v}$ are the new tune-able gains for the control equation. This modification should allow the control law to change its reactivity to the error with respect to changes in the speed.

The results for the experiment can be see in the figure 8. Where the improved control equation method reached an end objective function value of $0.387m$, $0.327m$, and $0.288m$ for $2.0m.s^{-1}$, $1.5m.s^{-1}$, and $1.0m.s^{-1}$ respectively.

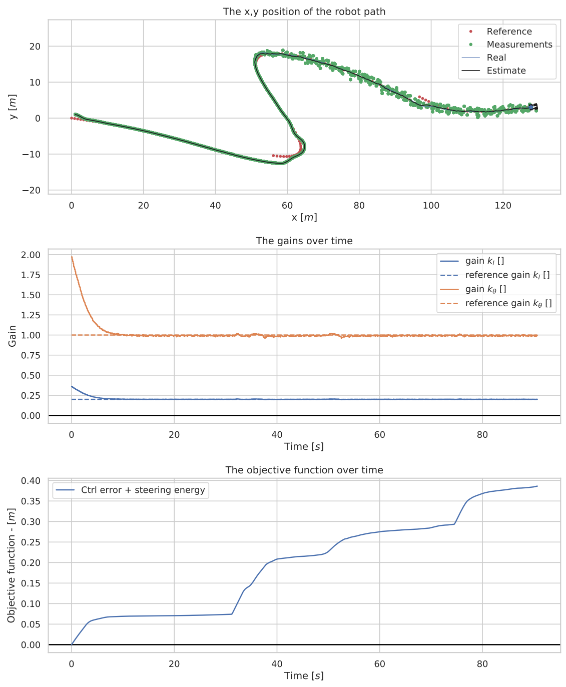

Figure 8: Above: The method line following. Middle: The predicted gain, where the reference gain is the baseline constant gain. Below: the objective function over time.

The improved control is adapting the gain with respect to changes in the speed. This allows the method to lower it's objective function substantially when compared to the baseline method, and to achieve similar performance to the neural network gain adaptation method. Furthermore, it is able to capture the fast initial convergence to the trajectory that the neural network gain adaptation method had, thanks to the initial speed up (visible from $t = 0$ to $t = 5$).

	$1.0m.s^{-1}$	$1.5m.s^{-1}$	$2.0m.s^{-1}$
Baseline	**0.308m**	**0.336m**	**0.389m**
Neural net gain	$0.291m$	**0.325m**	**0.372m**
Improved control	**0.288m**	$0.327m$	$0.387m$

Table 1: The objective function obtained for each method and speed over the trajectory.

In the table 1, the objective function for each method and speed can be observed. In all cases the baseline constant gain method had the highest objective function, meaning it had the worst performance. For the improved control and the neural network gain method, both results are comparable, as most of the performance gained is thanks to the speed adaption. However the improved control does not adapt to changes in the covariance, which in some cases allows the neural network gain method to outperform the improved control method.

5 Conclusion

A novel method for feature importance and a novel methodology to determine useful sensor information was proposed. This feature importance method allows the analysis of a neural network's behavior, to show the importance of each sensor information, and to potentially build a linear approximation of the neural network for a given input.

It has been applied to a steering controller of a car-like robot for a line following task in a highly dynamic simulated environment. In order to analyze a gain prediction method, and determine the optimal changes to the control equations to improve its performance. Indeed, the tested modification to the control law has been shown to reach comparable performance to the initial neural network gain prediction method.

This methodology can be applied to any given simulated model of a robot control task, in order to improve its control performance for a given criteria encoded as a objective function.

However, the sensor information must be ideally used to derive new control law from the robotic model, as using a linear approximation for a neural network will not encode the complete characteristic behavior to the neural network. As such this methodology far more powerful as a tool to describe what is important for control law, not how to derive a novel control law.

Future works include validating this methodology on varying control tasks in different field, and to use the novel feature importance method to assist in demystifying neural networks.

REFERENCES

Bakker, E., Nyborg, L., and Pacejka, H. B. (1987). Tyre modelling for use in vehicle dynamics studies. Technical report, SAE Technical Paper.

Gunning, D. (2017). Explainable artificial intelligence (xai). *Defense Advanced Research Projects Agency (DARPA), nd Web*, 2.

Ha, D. and Schmidhuber, J. (2018). World models. *arXiv preprint arXiv:1803.10122*.

Hansen, N. (2016). The CMA evolution strategy: A tutorial. *CoRR*, abs/1604.00772.

Hill., A., Lucet., E., and Lenain., R. (2019). Neuroevolution with cma-es for real-time gain tuning of a car-like robot controller. In *Proceedings of the 16th International Conference on Informatics in Control, Automation and Robotics - Volume 1: ICINCO,*, pages 311–319. INSTICC, SciTePress.

Hornik, K., Stinchcombe, M., and White, H. (1990). Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural Networks*, 3(5):551 – 560.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444.

Lenain, R., Deremetz, M., Braconnier, J.-B., Thuilot, B., and Rousseau, V. (2017). Robust sideslip angles observer for accurate off-road path tracking control. *Advanced Robotics*, 31(9):453–467.

Liaw, A., Wiener, M., et al. (2002). Classification and regression by randomforest. *R news*, 2(3):18–22.

Molnar, C. (2019). *Interpretable machine learning*. Lulu. com.

Mordvintsev, A., Olah, C., and Tyka, M. (2015). Deepdream-a code example for visualizing neural networks. *Google Research*, 2(5).

Simonyan, K., Vedaldi, A., and Zisserman, A. (2013). Deep inside convolutional networks: Visualising image classification models and saliency maps.

Suthaharan, S. (2016). Decision tree learning. In *Machine Learning Models and Algorithms for Big Data Classification*, pages 237–269. Springer.

Welch, G. and Bishop, G. (1995). An introduction to the kalman filter. Technical report, Chapel Hill, NC, USA.