Deep Learning Optimization Algorithm for Efficient Automotive Powertrain Control

Eric M. Zou

November 7, 2024

Abstract

With heightened concerns over the environmental impacts of internal combustion engines (ICEs), many automobile manufacturers have curtailed development of internal combustion engines in favor of alternative fuel vehicles. Alternatively, manufacturers have devoted large amounts of resources to redesign existing ICEs for improved efficiency. Both of these solutions are costly and time-consuming. Meanwhile, the advancement of deep learning algorithms provides the opportunity for a new avenue of automotive optimization. The focus of this study is on the application of deep learning algorithms specifically in engine control units (ECUs), which can significantly improve performance, efficiency, and reliability alike.

The objective of this study is to demonstrate the potential for a machine learning model to optimize internal combustion engines. A Python script was written to simulate the essential qualities of a commercial engine; a PyTorch neural network was trained to operate its inputs to optimize efficiency. After several hundred training cycles, the model attempted to optimize engine performance given fictitious environment data. Engine performance was recorded in various formats, notably fuel consumption. These results were holistically evaluated against simulated data of human inputs. The results show promise for AI controlled ECU systems to improve fuel efficiency of ICEs, without the steep development costs of full engine redesign.

1 Introduction

At time of writing, there exist several proposed solutions to the environmental impacts of ICEs. These include but are not limited to: emission controls [1], forced induction systems [2], hybrid powertrains [3], and battery-electric vehicles (BEVs) [4]. Of these proposed solutions, many suffer from key drawbacks. These include time/resource consumption or infrastructure readiness [5]. For example, a point can be made that the broader consumer market is not yet fully equipped with the infrastructure (both economic and safety) to support the seamless function and integration of battery-electric vehicles.

In the mean time, advances in computer chip technology facilitated the advent of the modern automotive engine control unit (ECU), which handles all engine controls in contemporary fuel-injection engines. Tasks handled by these chips include fuel injection control, ignition timing, and throttle control. As of current, these units are largely hard-coded and do not contain degrees of flexibility sufficient for optimal function under a variety of environments, machinery states, or drivers.

The use of an adaptive machine learning algorithm in tandem with modern ECUs has been suggested [6], but has not been realized for hardware limitations. This study aims to emphasize the importance of this integration and the importance of future hardware developments to accelerate said integration. The research question of this study is thus as follows: "To what extent can deep learning algorithms optimize the efficiency and performance of an internal combustion engine?"

2 Methodology

2.1 Overview

In the broader sense, this project consists of three components; a software simulation, a neural network (the subject), and a simulated scenario using the simulation engine.

First, a software simulation is developed using Python to simulated the engine performance of a 1.6 liter 4-stroke, inline-4 cylinder fuel injected engine. The simulation will be able to accept environmental inputs and predict power and fuel consumption analytics, akin to a virtual dynamometer.

Second, a 3x5x1 neural network is developed using PyTorch. This feed-forward system emulates the installation of adaptive engine control software in the automotive ECU. It takes in environmental data and outputs an optimized throttle control value for fuel efficiency.

Third, the software simulation and neural network are combined in a field scenario to train and test the efficacy of fuel consumption optimization using the AI throttle control, against the performance of a constant throttle input (as is common in human drivers).

2.2 Software Simulation

The software simulation (See Figure 1) was developed using Python 3 in a Jupyter Notebook environment. The simulation simulates a traditional, approximately square 4-cylinder engine with a stroke of 76mm and bore of 75mm. The compression ratio is set to be 10:1, and the gas constant for air is set to be R = 287.05 J/(kg*K).

The simulation accepts the following parameters as input: throttle opening (0 ¡= x ¡= 1), surrounding air temperature, surrounding air pressure, engine speed in revolutions per minute (RPM), and fuel-air ratio. The simulation calculates the rate of air intake to the vehicle as follows (an idea air intake is assumed):

$$v_{air} = \frac{throttle * V_{cylinder} * v_{engine}}{60}$$

Following this, stoichiometric constants are used to calculate the total air intake per 4-stroke engine cycle:

$$m_{air} = \frac{\rho_{air} * V_{air}}{R * T_{air}}$$

Subsequently, the fuel-air ratio is used to determine the amount of fuel consumed during that combustion cycle:

$$m_{fuelcycle} = \frac{m_{air}}{ratio}$$

Traditionally, fuel is injected once every four cycles, and the engine rotates 720 degrees during that time. Therefore, to find the amount of fuel burned for every full rotation,

$$m_{fuelrotation} = 2 * m_{fuelcycle}$$

For this simulation, an adjustable "torque factor" is in place to adjust engine performance to best match dynamometer results of commonly known engines, such as the Honda D16A. In this case, the torque factor is set to

$$TF = 1 + 0.01 * m_{fuelcycle}$$

The torque is thus calculated as follows:

```
if engine_speed <= 3500:
    torque = 200 + ((engine_speed / 3500) ** 2) * 200 * torque_factor
    torque /= 3
else:
    torque = 400 - (((engine_speed - 3500) / 3100) ** 2) * 300 * torque_factor
    torque /= 3
```

This produced a torque dynamometer curve similar to that found in mass-produced consumer engines. The horsepower curve is calculated similarly:

```
HP = torque * engine_speed / (5252)
```

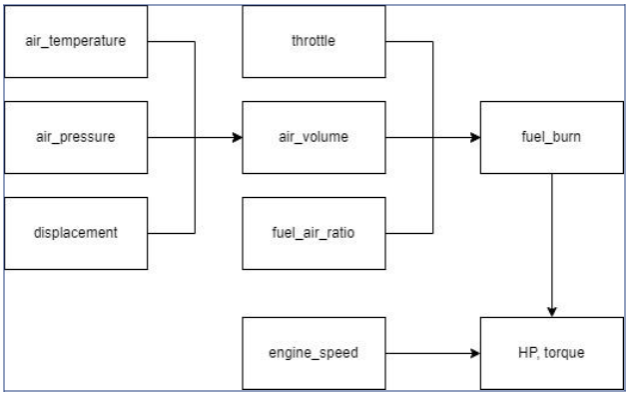

Figure 1: Flow chart of simulation setup; one can see that HP and torque are the primary outputs of the simulation, in addition to fuel burn. Figure created by student researcher.

where the torque peaks at 3500 RPM and horsepower peaks around 5252 RPM. This simulation does not take into consideration forced induction (turbocharges and superchargers) or variable valve timing/variable camshaft geometry systems (Honda VTEC, etc.).

2.3 Neural Network Design

The neural network is the driving subject of this investigation. It will be given relevant environmental data and asked to provide an optimal throttle control to minimize fuel consumption while providing the required torque. Its performance will be compared with a simulated constant throttle input of 70 percent.

The network is implemented in PyTorch, with a 3x5x1 architecture for simplicity and improved training speed (See Figure 2). The input layer accepts three inputs: air temperature, air pressure, and engine speed. These inputs are fed linearly to the hidden layer of five neurons. Subsequently, the five neurons feed forward to the output layer using the ReLU activation function for optimized training speed. Finally, since the final output must be a throttle control value between 0.0 and 1.0 exclusive, the sigmoid activation function is applied.

The neural network is defined as follows, using the PyTorch framework:

```
class ThrottleModel(nn.Module):
    def __init__(self):
        super(ThrottleModel, self).__init__()
        self.fc1 = nn.Linear(3, 5)  # Input layer: 3 neurons, Hidden layer: 5 neurons
        self.relu = nn.ReLU()  # ReLU activation function
        self.fc2 = nn.Linear(5, 1)  # Hidden layer: 5 neurons, Output layer: 1 neuron

    def forward(self, x):
        x = self.fc1(x)
```

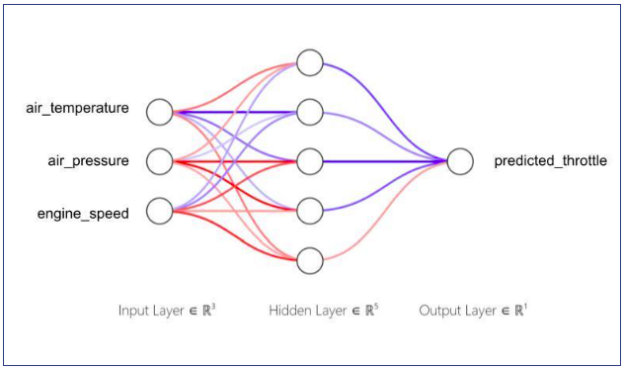

Figure 2: Visualization of 3x5x1 neural network. One can see three input neurons, a hidden layer of five neurons, and a single output. Figure created by student researcher via publicly available software.

```
x = self.relu(x)
x = self.fc2(x)
return torch.sigmoid(x)  # Apply sigmoid activation function for throttle input
```

2.4 Training Setup

The training procedure is as follows:

1. Randomly generated environment data are fed to both the engine simulation (putting it on standby) and the neural network
2. The neural network attempts to find an optimal throttle control
3. The neural network feeds its predicted throttle control into the engine simulation, finding a torque vs fuel burn ratio
4. The additive inverse of the torque vs fuel burn ratio is used as the loss function as a feedback to the prediction algorithm
5. The prediction algorithm is trained through its neural network setup

The above process is repeated with 500 epochs of 250 sets of randomly generated environment data. Environment data are randomized as follows:

- Air temperature between 20 and 30 degrees Celsius (deg C)
- Air pressure between 50 and 105 kilopascals (kPa)
- Engine speed between 500 and 7000 revolutions per minute (RPM)

Training inputs are normalized to better fit PyTorch tensor requirements.

2.5 Testing Parameters and Scenarios

After model training has been completed, it is tested in a fictitious scenario of a 2-hour drive up a hypothetical mountain. To accomplish this, a multi-dimensional array of length 120 (one entry per minute) is created, and environmental data are entered as follows:

- Linearly decreasing temperature from 30 deg C to 20 deg C
- Linearly decreasing engine speed from 6000 RPM to 2000 RPM
- Linearly decreasing air pressure from 103.15 kPa to 95 kPa

Then, for each entry, the machine learning model is provided the environmental data and asked to output a predicted optimal throttle control value. This value is recorded in the array.

After all algorithm predictions are complete, the engine simulation as outlined in Section 2.2 runs through the array and finds the torque to fuel burn ratio (used as a criterion for measuring engine fuel efficiency) for each minute entry. A secondary array measures the torque to fuel burn ratio if the throttle control was at a constant 0.7. The data are recorded in results.

3 Results

3.1 Engine Simulation Configuration

Based on the adjustment of the engine simulation as outlined in Section 2.2, the simulation was given a control run to output torque, horsepower, and fuel efficiency curves for the following data:

- Air temperature 25 deg C
- Air pressure 101.3 kPa
- Fuel-air-ratio 14.7:1
- Throttle input 0.8

The simulation was able to provide horsepower, torque, and fuel efficiency curves similar to that seen by traditional physical dynamometers (See Figure 3). Under the aforementioned conditions, the engine produced a peak horsepower at a little under 5000 RPM, a peak torque at 3500 RPM, and linearly increasing fuel economy. Given the engine's relatively square design, these outputs are fairly plausible.

3.2 Engine Simulation Virtual Dynamometer

The simulation also provided a dynamometer reading similar to that as outlined in Section 3.1. This can be seen in Figure 4.

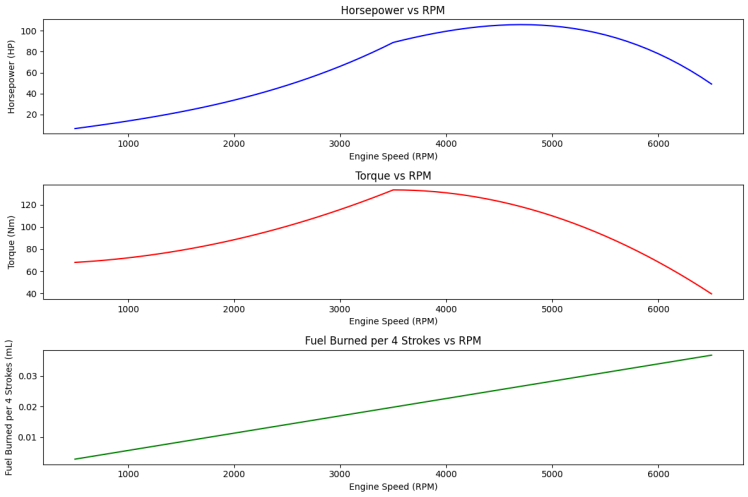

Figure 3: Virtual dynamometer and performance results of engine simulation in constant conditions. Figure created by student researcher.

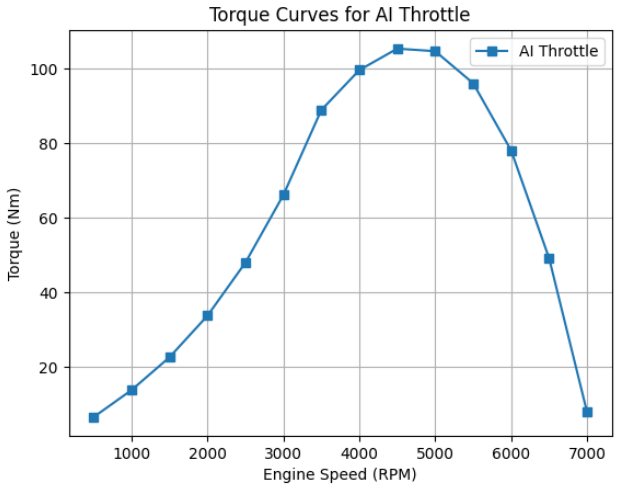

Figure 4: Virtual dynamometer results of engine simulation under AI throttle control. Figure created by student researcher.

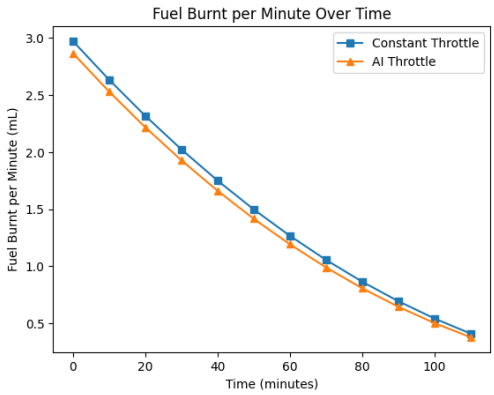

Figure 5: With time as the x-axis, figure shows simulated fuel burn over the 2-hour uphill drive, as piloted by both the AI throttle control (orange) and constant throttle input (blue). Figure created by student researcher.

3.3 Fuel Efficiency Metrics

Obtaining data on the fuel efficiency metrics of this new adaptive throttle control system is the primary objective of this study. Full results on the hour-by-hour fuel efficiency of the adaptive throttle system compared to a constant throttle input for the scenario devised in Section 2.5 can be seen in Figure 5.

To complete the full 2-hour drive, the constant 70 percent throttle input consumed 0.168 liters of fuel. By comparison, the adaptive throttle control only consumed 0.159 liters, making it 5.4 percent more fuel efficient within the parameters of this environment.

3.4 Miscellaneous Observations

Although the full training of the model takes several minutes, it can make small adjustments within milliseconds, as shown by its quickly decreasing loss curve (See Figure 6). This means that with additional parameters and more robust neural network structures, an adaptive algorithm running on an engine control unit can quickly adjust itself to the environment, vehicle condition, and even driver behaviors, greatly improving fuel economy and/or engine power as desired.

Additionally, the full script, when downloaded as a Python Jupyter (.ipynb) file, only consumes 13.8 kilobytes (kB) of disk space, although Python is a compiled programming language and more space would likely be needed when compiling this program to a chip-level language to run on ECUs.

Figure 6: Sample of training loss curve of machine learning model. Loss fluctuates but drops quickly during training, leading to high adaptability in field scenarios. Figure created by student researcher.

4 Discussion

4.1 Potential Applications

The great fuel economy improvement (and therefore reduced emissions) observed by the adaptive throttle control algorithm within its simulated environment shows great potential for its implementation in the real world. Provided that (a) the simulation is a reasonable representation of real-world engine behaviors, (b) hardware limitations on existing ECUs can be overcome, and (c) the implementations can be developed with relatively low human and physical resources, AI engine control algorithms may serve as an easy-access, near-future solution to augment and complement alternative fuel technologies.

4.2 Hardware Implications

Hardware limitations outlined in Section 4.1 can and have been overcome. While current electronic control units are incapable of handling more computationally intensive machine learning software, external computational devices such as Raspberry Pis can and have been used to augment ECU abilities [7].

Further developments of computational hardware for automotive power trains are likely less cost and time intensive than alternatives such as engine redesign, forced induction [2], and alternative fuel vehicles [5]. Further, once the sufficient hardware is developed, implementation and deployment can be rapidly facilitated in existing vehicles due to the relative ease of installing a computer chip as opposed to modifying engine internals. This allow for existing vehicles to be optimized, reducing the economic strain of producing new vehicles with advanced emissions-reducing features.

4.3 Limitations/Future Studies

Several limitations exist to this study. Simulation software was developed by the researcher, limiting the accuracy of simulated results. More robust simulation software or even physical trials in a future study

would improve the accuracy of the results outlined in this study. Nonetheless, the results of this study point promise to the field of machine learning software in automotive ECUs and demonstrates the necessity of further research in the area.

5 Acknowledgments

All figures, software, and experimentation was conducted by the researcher unless otherwise noted. Google's Colab cloud computing service was used to run simulations and host machine learning model training. Figures are produced by a combination of Google Draw, Alex LeNail's NN-SVG, and Python Matplotlib. The efforts of various automotive forum (primarily honda-tech.com) to provide dynamometer readings are greatly appreciated.

References

[1] M. V. Twigg, "Progress and future challenges in controlling automotive exhaust gas emissions," *Applied Catalysis B Environment and Energy*, vol. 70, pp. 2–15, 7 2006.

[2] L. Schwitzer, "Forced-Induction possibilities for automotive vehicles," *SAE technical papers on CD-ROM/SAE technical paper series*, 1 1934.

[3] A. Balluchi, L. Benvenuti, M. di Benedetto, C. Pinello, and A. Sangiovanni-Vincentelli, "Automotive engine control and hybrid systems: challenges and opportunities," *Proceedings of the IEEE*, vol. 88, no. 7, pp. 888–912, 2000.

[4] J. Buekers, M. Van Holderbeke, J. Bierkens, and L. I. Panis, "Health and environmental benefits related to electric vehicle introduction in EU countries," *Transportation Research Part D Transport and Environment*, vol. 33, pp. 26–38, 10 2014.

[5] M. A. Ghadikolaei, P. K. Wong, C. S. Cheung, J. Zhao, Z. Ning, K.-F. Yung, H. C. Wong, and N. K. Gali, "Why is the world not yet ready to use alternative fuel vehicles?," *Heliyon*, vol. 7, p. e07527, 7 2021.

[6] N. Yokoyama and Y. Moriyama, "Technology development of AI application for in-vehicle control ECU," *DENSO TEN Technical Review*, vol. 3, no. 6.

[7] A. Vaughan and S. Bohac, V, "An extreme learning machine approach to predicting near chaotic HCCI combustion phasing in Real-Time," 10 2013.