Pre-training time series models of truck behavior - Volvo Group 1 Final Report

Weilong Chen, Jadd Ujam, K Jai Ganesh Singh Dadwal

December 19, 2023

Abstract

This document is a report on the project undertaken by the MPENM students from Chalmers at the Volvo Group Truck, during study period 2 (Autumn Semester) 2023-2024. The project is on pre-training time series model which classifies as stochastic data analysis and deep learning. The document explains the nature of our work as well as the theory, in concise, behind the project. We will discuss data prepocessing, model construction, training of hyper-parameters, and the theory of BERT model, one of the primary models used in natural language processing, as well as the transformer architecture(used by BERT) and its mathematical theory. The results of our project, to predict fuel-consumption after pre-training, gave a mean absolute error (MAE) less than 0.5%. For further research, other model architectures could be deployed such as the GPT model (Generative Pre-trained Transformer). Some of us will also be pursuing our master's thesis in that direction.

Table of Contents

Introduction

Volvo Group Trucks has made substantial investments in AI technologies to enhance its capabilities in developing top-tier products. This initiative aligns with a project led by Chalmers MPENM students, focusing on a time series model based on a mathematically derived architecture known as the transformer.

In simpler terms, the project involves training the model to accurately predict one or more output signals based on a set of data, also referred to as signals. These signals, in our case, are numerical data, although they could encompass various data types such as images or audio. The transformer architecture plays a crucial role in converting raw data into a numerical vector, facilitating subsequent processing.

For individuals familiar with recurrent neural networks (RNNs), it's noteworthy that transformers overcome several limitations associated with RNNs. Unlike RNNs, transformers excel in handling sequential data, offering improved efficiency and performance in learning complex patterns from the given signals.

In summary, the project aims to leverage transformer architecture to pre-train a time series model that can effectively predict output signals from numeric data, showcasing the potential of A.I. technologies in advancing product development within the Volvo Group Trucks.

Theory

The Transformer is a deep learning architecture built upon the self-attention mechanism. The attention mechanism plays a crucial role in calculating soft weights for each piece of data (token), aiding in its embedding. Data embedding involves representing data in a real-valued vector, ensuring that data with similar meanings are closer together in the vector space.

The architectural foundation of the Transformer consists of a mathematically constructed encoder and decoder. The document's current focus is exclusively on the encoder, thus limiting the scope of the discussion to this component.

Before delving into the details, let's briefly define and explain some key mathematical concepts integral to the transformer architecture [2].
A transformer is a parameterized function class $f_\theta : \mathbb{R}^{n\times d} \longrightarrow \mathbb{R}^{n\times d}$. If $x \in \mathbb{R}^{n\times d}$ is the input, then $z = f_\theta(x)$ is the output where,

$$Q^{(h)}(x_i) = W_{h,q}^T x_i,\ K^{(h)}(x_i) = W_{h,k}^T x_i,\ V^{(h)}(x_i) = W_{h,v}^T x_i, \text{ where } W_{h,q}, W_{h,k}, W_{h,v} \in \mathbb{R}^{d\times k} \tag{1}$$

Q, K, V are matrices which are known as query, key and values. The triple is similar to a retrieval system, where in a search engine will map the query (say, text in the search bar) against a set of keys (video title, description, etc.) associated with candidate videos in their database, then the retrieval system presents the best matched videos (values). The triple is generated with the help of input (data with which the model is trained) and some weights (which are initially randomly generated, say, normally distributed). Note here, that input data is embedded, that is, converted to a vector of numbers. Input data can be anything such as sound, picture etc. In our project, the input data is numeric.

The input $x \in \mathbb{R}^{n\times d}$, here, can be interpreted as a collection of n objects each with d features. The output, $z = f_\theta(x)$, is also of the same dimension, in terms of matrices. The parameters θ consists of the entries of the weight matrices i.e. W's, along with the layer norm parameters γ and β, given in equations (4), (6) and (7).

$$\alpha_{i,j}^{(h)} = \text{softmax}_j\left(\frac{\langle Q^{(h)}(x_i)\ ,\ K^{(h)}(x_j)\rangle}{\sqrt{k}}\right) \tag{2}$$

The softmax is an activation function which converts the input embedding into a vector of probabilities. The notation softmax_j indicates that in equation (9) softmax function is taken over the d-dimensional vector indexed by j i.e. across features.

$$u_i' = \sum_{h=1}^{H} W_{c,h}^T \sum_{j=1}^{n} \alpha_{i,j}^{(h)} V^{(h)}(x_i), \text{ where } W_{c,h} \in \mathbb{R}^{k\times d} \tag{3}$$

The equation (1), (2) and (3) are H sets of equations indexed by $h = 1, ..., H$. These together constitute the multi-headed self attention. The $\alpha_{i,j}^{(h)}$ are attention weights, which control how much element x_i attends x_j in head h. This can be thought of as a mechanism that weights the importance of different inputs when producing an output.

$$u_i = \text{ LayerNorm}(x_i + u_i'; \gamma_1, \beta_1), \text{ where } \gamma_1, \beta_1 \in \mathbb{R}^d \tag{4}$$

A per-object fully connected layer is defined by equation (5). Equations (4) and (6) constitutes the layer normalization and residual connections between attention layers and fully connected layers, motivated by emperical observations about effectively optimizing deep learning models. Equation (7) and (8) defines what a layer normalization is .

$$z_i' = W_2^T \text{ReLU}(W_1^T u_i), \text{ where } W_1 \in \mathbb{R}^{d\times m},\ W_2 \in \mathbb{R}^{m\times d} \text{ and } \text{ReLU}(g) = \max(0, g) \tag{5}$$

$$z_i = \text{ LayerNorm}(u_i + z_i'; \gamma_2, \beta_2), \text{ where } \gamma_2, \beta_2 \in \mathbb{R}^d \tag{6}$$

$$\text{LayerNorm}(z; \gamma, \beta) = \gamma\frac{(z - \mu_z)}{\sigma_z} + \beta, \text{ where } \gamma, \beta \in \mathbb{R}^k \tag{7}$$

$$\mu_z = \frac{1}{k}\sum_{i=1}^{k} z_i,\ \sigma_z = \sqrt{\frac{1}{k}\sum_{i=1}^{k}(z_i - \mu_z)^2} \tag{8}$$

Also, ReLU is another activation function used in connected layer see equation (5). It is simple and has a faster convergence rate as compared to other activation functions like sigmoid function.

$$\hat{y} = \text{softmax}(W_z^T z) = \frac{\exp(W_z^T z)}{\sum_{k=1}^{m}\exp(W_z^T z)_k}, \text{ where } W_z \in \mathbb{R}^{d\times n} \tag{9}$$

A key feature of a transformer model is that it is oblivious to relational structure between its n inputs $x_i \in \mathbb{R}^d$ i.e. a transformer is a set of features model, operating on a collection of n unordered , d-dimensional features. To model position in a transformer, one of the approaches, is to positionally encode based on the sinusoidal position embedding $p \in \mathbb{R}^{n \times d}$ as shown in equation (10).

$$p_{k,2i} = \sin\left(\frac{k}{10000^{2i/d}}\right), \; p_{k,2i+1} = \cos\left(\frac{k}{10000^{2i/d}}\right) \tag{10}$$

The hyper-parameters of the transformer are d, k, m, H, and L as used in equations (1-10). L is the number of transformer blocks that form a composition: $f_{\theta_L} \circ ... \circ f_{\theta_1}(x) \in \mathbb{R}^{n \times d}$. Common settings of the hyper-parameters are d=512, k=64, m=2048, H=8, and L=6. This concludes the basic mathematical aspects of the transformer architecture.

One important point to note is that there are several variants of activation functions like ReLU, based on the requirements of the learning model. One such activation function which is used over ReLU is called gaussian error linear unit (GELU), equation (11). One important property of such an activation function over ReLU is that GELU is differentiable at zero value, whereas ReLU being a step wise function is not. Thus in the case of ReLU, it makes the probability of input being dropped higher as domain value decreases. This gives GELU an added advantage over ReLU, to be used in the feed forward pass.

$$\begin{aligned} \text{GELU}(g) = \frac{g}{2}\left(1 + \text{erf}(g/\sqrt{2})\right), \text{ where erf denoted as the error function:} \\ \text{erf}(g) = \frac{2}{\sqrt{\pi}} \int_0^g \exp^{-r^2} dr \end{aligned} \tag{11}$$

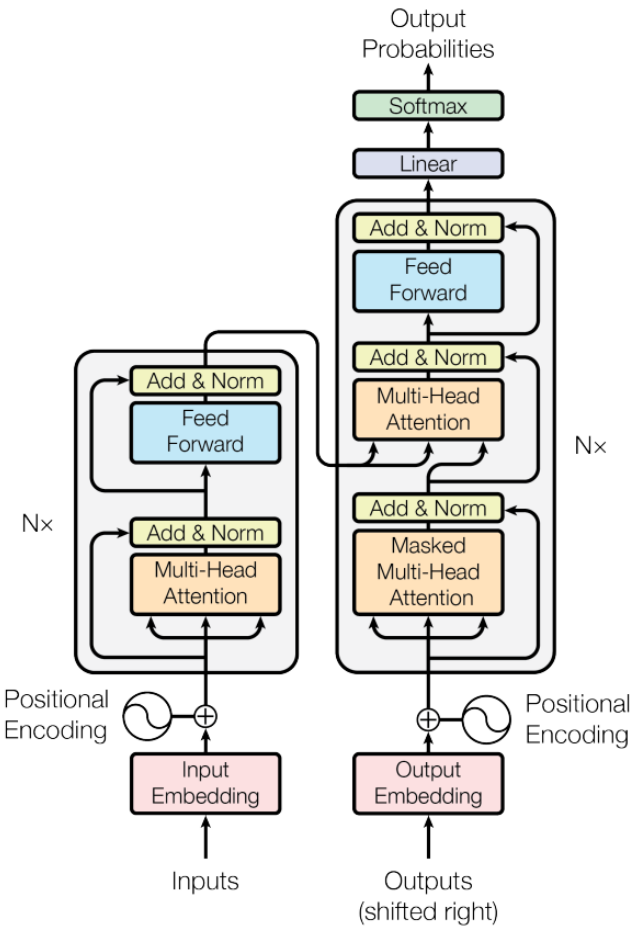


Figure 1: The diagram is sourced from the original paper “Attention is all you need".[3]

BERT, short for Bidirectional Encoder Representations from Transformers, is a language model developed by Google researchers [1]. It was introduced in their 2018 paper, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. BERT marks a significant advancement in the realm of machine language

understanding, exerting a profound influence on Natural Language Processing (NLP). The model employs a masked language model objective where It randomly masks words in a sentence, and the model is trained to predict the masked words using bidirectional context. This bidirectional training allows BERT to grasp context from both preceding and following words, enhancing its understanding of context. BERT represents an advancement in the domain of machine language understanding and has been influential in the field of Natural Language Processing (NLP). Unlike previous models that processed text in one direction, BERT analyzes text bidirectionally. This method allows for a more comprehensive understanding of the context within sentences, aiding in various language processing tasks like text summarization and question answering.

A key aspect of BERT's design is its architecture, which is based on a multi-layer bidirectional Transformer encoder. This architecture is central to its ability to understand language context. The architecture of BERT is composed of multiple layers of Transformer encoders. Each encoder layer processes the input text in parallel, which allows the model to efficiently handle long texts and further understand the context from both directions. Within these encoder layers, BERT uses a self-attention mechanism. This mechanism enables the model to weigh the importance of different words in a sentence, regardless of their position, providing a more nuanced understanding of language.

Implementation

The implementation is structured into three key phases: Data Preprocessing, where raw data undergoes transformation and exploratory analysis; Model Construction, involving the definition of model architecture and selection based on the problem's nature; and Training Hyperparameter tuning, focusing on optimizing model performance through parameter adjustments.

Data Preprocessing

In the data preprocessing phase, relevant signals were extracted from both the test field and simulated data. The process involved a three-step approach: clean, extraction, and second-round checks. Metadata was used to identify files containing the required signals, and extraction was performed to validate and retrieve the necessary information. The extracted signals were then organized into 1024-second segments and saved as separate pickle files, with each file representing a portion of a driving session.

Additionally, to incorporate vehicle variants into the analysis, information from approximately 200 vehicles was extracted from an SQL database. This data, detailing variants for each vehicle, was saved in CSV format. The overall objective of these preprocessing steps was to ensure the relevance and validity of the extracted information, addressing potential errors or anomalies in the data.

This comprehensive data preprocessing approach is crucial for optimizing the dataset and laying the foundation for subsequent analysis and model training.

Model Construction

The model architecture deployed in this study is rooted in the transformative capabilities of transformer encoder blocks, a prevailing force in contemporary deep learning landscapes. The architecture is dissected into three key components: the stem, serving as the embedding hub with three linear layers to convert input data into a multidimensional space; the body, featuring a six-layer transformer encoder block for processing input sequences and generating context-rich hidden representations; and the head, comprising convolutional layers and a linear layer for final predictions. The stem, crucially, accommodates both the primary sequence embedding for signals and an additional parallel embedding block for variants, enhancing modularity and catering to the distinct data modalities of categorical variants and discrete signals.

Furthermore, the task-specific models, tailored for predicting truck fuel consumption and engine power output, employ input signals encompassing road slope, weight, cruise control engagement, and vehicle velocity. The supervised learning process guides model training, where sizable data batches are fed into the system, allowing it to learn from examples and make predictions for fuel consumption or engine power output in 1024-second slices. The use of the L2 Loss function underscores the precision required in predicting the nuanced interplay of road and vehicle dynamics, encapsulating the intricate relationship between input features and the targeted output variables.

Training Hyperparameter

Generally, we use PyTorch and also the Lightning library to implement our end-to-end training pipeline. The code is private at Volvo on Enterprise GitHub and has been continuously worked on by several other colleagues.

Below is a summary of the most important hyperparameters for training and inference, which will be read from a JSON config file.

The training mainly runs on Nvidia A100 GPU and takes around 2 to 3 hours to complete.

Name	**Key Configuration**
burnx-main	Type: MaskedSignalDataset, Val Ratio: 0.01, Mask: bert-masking
burnx-pred-fuel	Type: SignalDataset, Val Ratio: 0.01, Encode/Decode Steps: 1024
mvpred	Type: Signet, Stem: Seq Emb Patch Length: 32, Var Emb Width: 32, Body: VanillaTransformerEncoder, Depth: 6, Width: 256, Subsequent Mask: True
base	Type: Signet, Stem: Seq Emb Patch Length: 32, Var Emb Width: 32, Body: VanillaTransformerEncoder, Depth: 6, Width: 256, Subsequent Mask: True, Bottleneck: ConvBlock, Dimension: 1, Structures: In Channels: 32, List Out Channels: [64, 128, 256], Kernel Size: 3, Stride: 2
train runner	Batch Size: 128, Log Steps: 32, Epoch Size: 4096, Max Epochs: 128, Num Val Batches: 32, Optim: Init LR: 0.0001, Schedule LR: False, Early Stopping: Monitor: Loss, Mode: Min, Patience: 16, Visualization: Do: True, N Samples: 8
train dry runner	Batch Size: 64, Log Steps: 32, Epoch Size: 256, Max Epochs: 8, Num Train Batches: 256, Num Val Batches: 2, Optim: Init LR: 0.0001, Schedule LR: False, Early Stopping: Monitor: Loss, Mode: Min, Patience: 16, Visualization: Do: True, N Samples: 8

Results

We tested our model on 2000 samples, and the mean absolute error we obtained is around 0.20%. In Figure 3, we selected two cases to illustrate—one where the mean absolute error is slightly higher but still within an acceptable range, and another where the relative error is lower. Instead of focusing solely on the fuel consumption task, we have the flexibility to choose arbitrary signals as input and output. This provides the advantage of generalizability to different tasks.

Discussion/Conclusion

The benefit of our deep learning model is its ability to process large amounts of data faster than traditional physics-based models, making it an attractive choice for industries that require real-time predictions. By leveraging the power of deep learning, our model can extract hidden patterns and relationships from complex datasets that may not be immediately evident with traditional models.

While there are still limitations to current vanilla transformer architectures, further research will focus on how to encode physical knowledge to enhance the model's accuracy and explainability. Data augmentation is also an avenue to explore. Since the model currently relies heavily on data, exploring the possibility of utilizing equivariant neural network architecture to reduce the dimensionality of the required data is a potential avenue for improvement. Another future direction is to explore the possibility of combining the current model with multimodal models.

In conclusion, the validation of our model by field experts highlights the effectiveness of deep learning in predictive modeling. Our model's ability to generate accurate forecasts at a faster speed than traditional physics-based models makes it a valuable tool for industries that require quick and precise predictions for decision-making.

Acknowledgments

We would like to extend our deepest gratitude to our supervisor, Parthasarathy Dhasarathy, architect at machine learning team at Volvo Group Trucks, for his exceptional mentorship and support during the development of this project. His guidance, encouragement, and expertise have been invaluable throughout the entire process. We are grateful for his insightful feedback, which has greatly enriched our work and helped us navigate through challenges. His support to our work has been truly encouraging, and we are thankful for the opportunity to have worked under his supervision. Also, we would like to thank Alexey Geynts, for taking out time to understand

our work in detail during our final presentation, which we presented from Volvo Group Truck office. We would also like to thank the whole machine learning team at Volvo group truck for their support and guidance in completing our project.

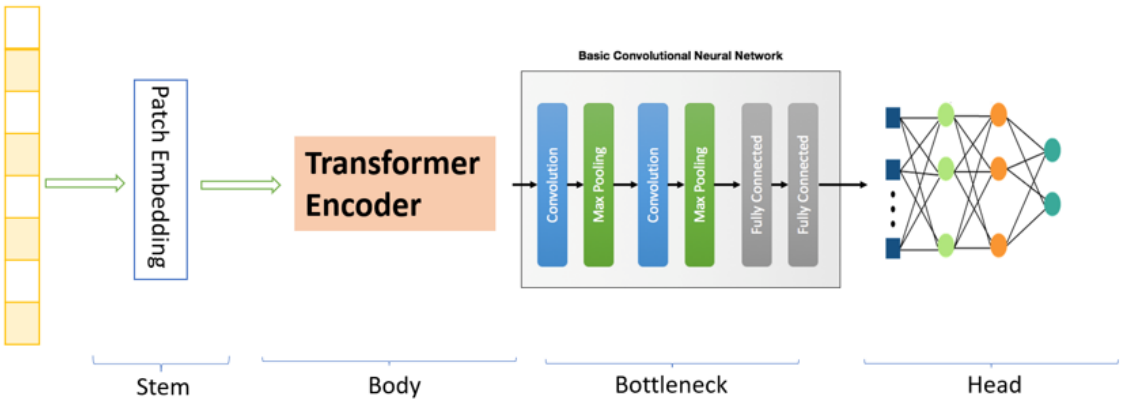
References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, 2019.

[2] John Thickstun, *The transformer model in equations*, University of Washington: Seattle, WA, USA (2021).

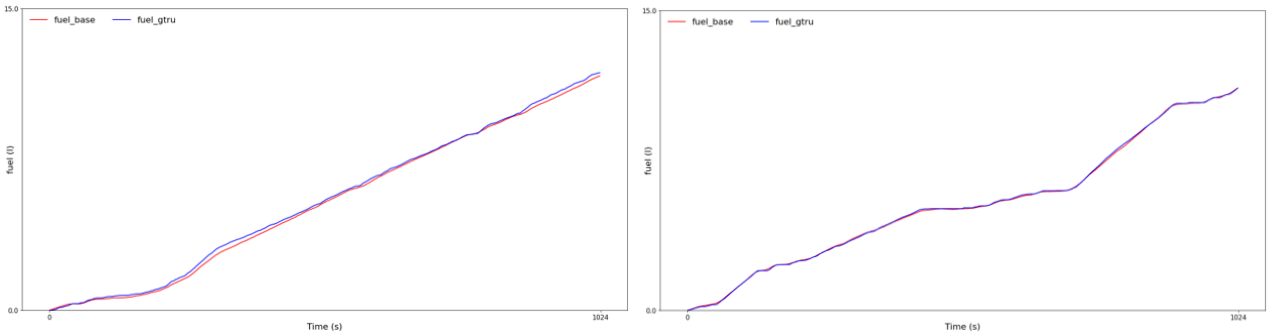
[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin, *Attention is all you need*, Advances in neural information processing systems, 2017, pp. 5998–6008.

Appendix



(a) Model construction

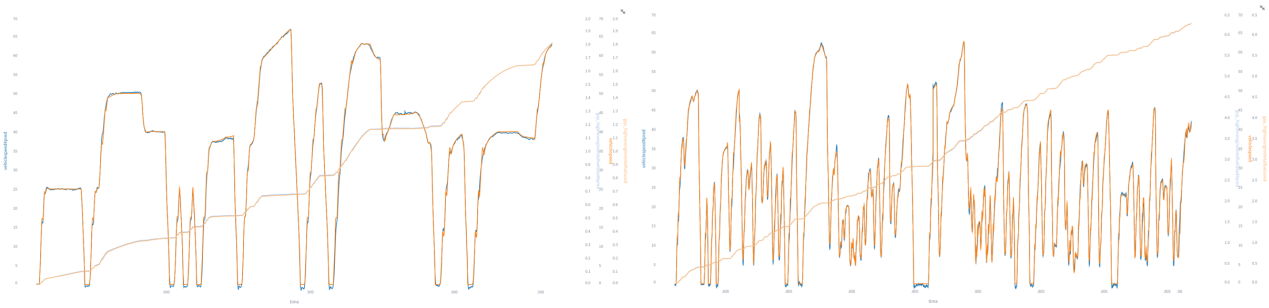
Figure 2: Model Constructions



(a) Larger absolute error case

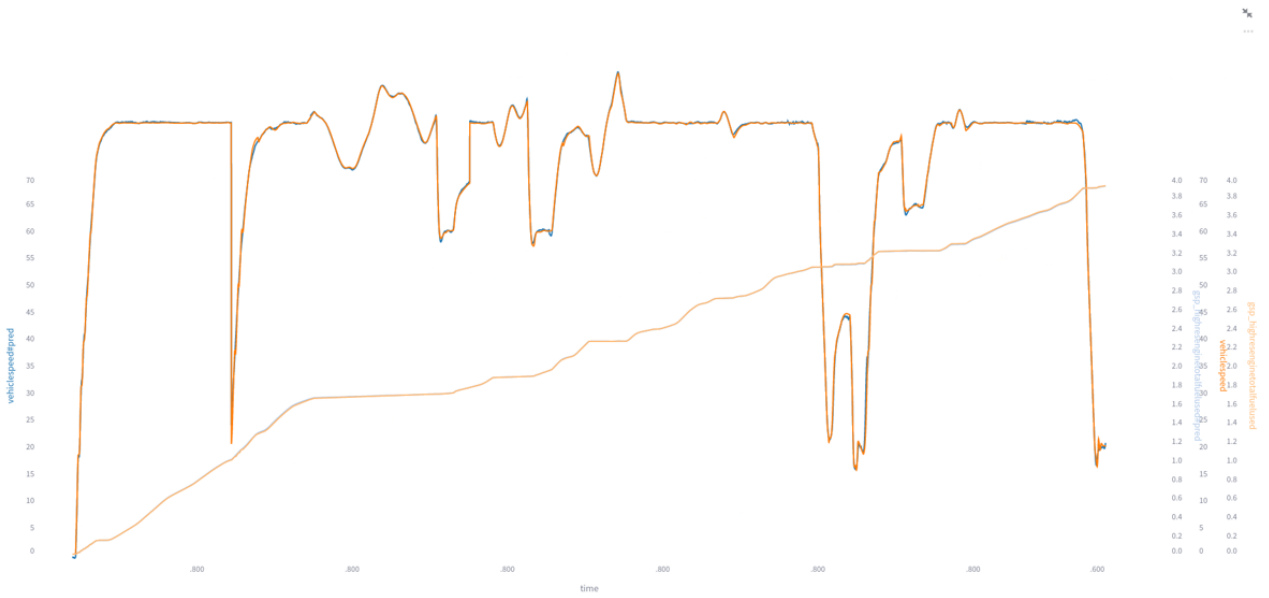
(b) Lower absolute error case

Figure 3: Model Predictions for fuel consumption



(a) Trip1

(b) Trip2



(c) Trip3

Figure 4: Model predictions with vehicle speed information