Video Transformers for Autonomous Driving

Sounak Mondal
Department of Computer Science
Stony Brook University
somondal@cs.stonybrook.edu
sounak.mondal@stonybrook.edu

Jongwoo Park
Department of Computer Science
Stony Brook University
jongwopark@cs.stonybrook.edu
jongwoo.park@stonybrook.edu

Abstract

Recently, pure Transformer-based architectures have proved to be a more efficient alternative to image-based tasks. However, most state transition models using the video sequence information in behavior cloning methods still adopt Recurrent Neural Networks (RNNs) or attention layers in conjunction with convolutional networks. Hence, we explore the efficacy of pure Transformers for the behavior cloning model using video input - particularly when applied to the case of self-driving in autonomous vehicles. We propose a novel transformer-based model called Spatiotemporal Video Transformer (VidT) which captures the spatiotemporal relationships between image patches of several consecutive frames using self-attention layers along with spatial embedding and temporal embeddings. The proposed Transformer provides a more accurate prediction of acceleration than the baseline LSTM with a small amount of Waymo Open Dataset.

1. Introduction

Autonomous vehicle (AV in short) or self driving vehicles have been in the limelight in recent times. Recent advances in the field of autonomous driving has prompted significant investment in terms of financial and human resources from the tech industry. Most popular autonomous driving systems employ behavior cloning as a technique to reproduce the human driving skills. An integral part of many sophisticated behavior cloning models is a Recurrent Neural Network (RNN)-based sequence model whose function is to capture history of an array of contiguous frames. Since Transformer-based models are supplanting RNN-based architectures in text tasks and have recently proved to excel at several vision tasks as well, we seek to exploit the efficacy of transformers to behavior cloning applied to autonomous driving. We propose a novel Video Transformer model called Spatiotemporal Video Transformer (VidT) based on Visual Transformers[3]. Additionally, we leverage the recently released large scale Waymo Open Dataset[9] for training our Video Transformer model. Our model proves to perform marginally better than the baseline LSTM-based model on a subset of the Waymo Open Dataset.

2. Related Work

2.1. Behavior Cloning

Behavior cloning can mimic the behavior of human experts without a reward function. Bojarski et al. [1] presented that the model with CNN and image input successfully learns the behavior of human experts. The learned model could drive in complex scenarios, such as areas with blurred vision or unpaved roads.

2.2. Transformer

A Transformer is a type of neural network that uses a self-attention mechanism to find the relation between input sequences and enrich the representation of embeddings. Researchers have recently applied Transformer to computer vision (CV) tasks, and ViT[3] showed high performance in image recognition. However, most researchers still use LSTM on behavior cloning with video input [4]. It inspired us to design a Spatiotemporal Video Transformer (VidT) for the behavior cloning of AV.

3. Spatiotemporal Video Transformer (VidT) for Behavior Cloning

We propose a transformer based architecture called Spatiotemporal Video Transformer (VidT) for processing a sequence of frames in order to estimate driving parameters in a self-driving setting. Additionally, to provide a comparable baseline for the same, we implement an LSTM-based model for the same task.

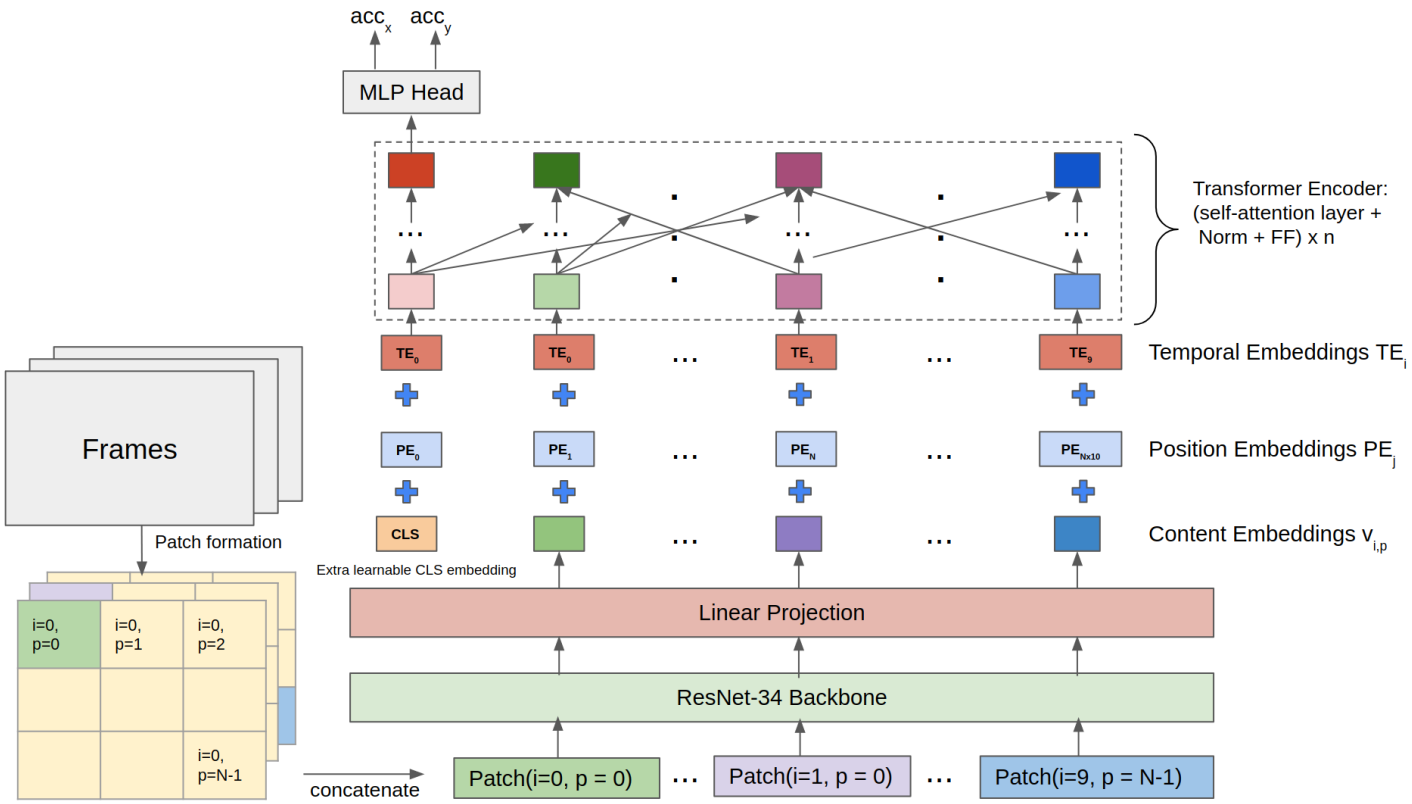

Figure 1. Spatiotemporal Video Transformer

3.1. Proposed Architecture

For our behavior cloning task, we are supposed to estimate the accelerations of the car along longitudina(x) and lateral(y) directions for each frame F_i where i is an index of a frame. To give us a sense of the history we collect an additional 9 preceding frames $\{F_{i-9}, F_{i-8}, ..., F_{i-1}\}$ along with current frame F_i.

We leverage the patch-wise processing used in Visual Transformers (VIT) proposed by Dosovitskiy et al[3] and extend it to a sequence of frames instead of a single image. Essentially, we partition each frame $F_i \in \mathbb{R}^{H\times W\times C}$ into N congruent patches $x_{i,p} \in \mathbb{R}^{h\times w\times C}$ where p is an index of a patch. To better facilitate the learning process of the transformer architecture, we replace the single linear projection layer of VIT by a frozen pre-trained ResNet-34[5] encoder to get high level embeddings $r_{i,p} \in \mathbb{R}^{512}$ for each patch $x_{i,p}$. We follow this ResNet layer by a linear projection layer for dimensionality reduction to obtain patch token content embeddings $v_{i,p} \in \mathbb{R}^d$ corresponding to $x_{i,p}$ where d is the latent dimensionality that the linear projection layer projects the $r_{i,p}$ to.

Now, to extend the VIT to a sequence of frames, we append the patch token embeddings $v_{i,p}$ in order of the frames. Essentially, the resultant concatenation $V \in \mathbb{R}^{(N\times 10+1)\times d}$ is of the following form:

$$
\begin{aligned}
V &= [v_{cls}; v_{i-9,0}; ...; v_{i-9,N-1}; v_{i-8,0}; ...; v_{i,0}; ...; v_{i,N-1}] \\
v_{i,p} &= r_{i,p} * W_{linear}, \quad W_{linear} \in \mathbb{R}^{512\times d} \\
r_{i,p} &= ResNet34(x_{i,p})
\end{aligned}
\tag{1}
$$

Note that v_{cls} is a special token embedding for pooling the sequence level information.

As authors in VIT[3] propose, we element-wise add positional embedding (PE) to the each content embeddings (V) pertaining to $(N\times 10+1)$ unique positions in 10 frames. Additionally, we propose a temporal segment embedding (TE) resembling the segment embedding used in BERT[2] to distinguish between paired input sequences in tasks like Textual Similarity and Textual Entailment estimation. The temporal segment embedding denotes which frame indexed between 0 and 9 the patch embedding belongs to. This temporal segment embedding is also element-wise added to the content embeddings. So, in essence, the positional embeddings captures the spatial information of the patch while the segment embedding captures the time step of the frame the

patch belongs to. This disentangling of position and segment embeddings has been found to yield good results in case of text summarization[8] where there is only one input sequence - but segment embeddings are used to demarcate sentences in text.

$$\begin{aligned}
&\mathbb{V} = V + PE + TE, \text{ where } V, PE, TE \in \mathbb{R}^{(N\times 10+1)\times d} \\
&PE = [PE_0; PE_1; ...; PE_{(N\times 10+1)}] \text{ where } PE_i \in \mathbb{R}^d \\
&TE = [TE_0^{N+1}; TE_1^N; ...; TE_9^N] \text{ where } TE_i \in \mathbb{R}^d
\end{aligned} \tag{2}$$

Note that, $X^y \in \mathbb{R}^{k\times y}$ is the vector $X \in \mathbb{R}^k$ repeated y times and concatenated.

Hence, $\mathbb{V}$ is the input to a multi layer transformer encoder. We use the transformer encoder described by Vaswani et al[10] which uses a cascade of self-attention, normalization and feed forward layers with residual connections to compute contextual representation of an input sequence. After passing through the multi-layer transformer encoder, we achieve a contextualized representation corresponding to each vector in V. We choose the representation corresponding to v_{cls}, add a two layer MLP on top to regress the acceleration values along x and y directions. Figure 1 depicts the overall architecture of the Spatiotemporal Video Transformer model.

4. Baseline LSTM-based model for Behavior Cloning

To establish a baseline for behavior cloning on the same amount of dataset that we train and evaluate on our Video Transformer model, we also created an LSTM-based model. This is done primarily due to the fact that most behavior cloning algorithms (Gu et al[4]) employ RNN-based architectures to capture history information through a sequence of frames.

For our LSTM model, we obtain the same content embeddings $v_{i,p}$. However, since the input to each time step in the LSTM is a composite frame-level embedding, we construct an embedding $v_i \in \mathbb{R}^{N\times d}$ for frame i by concatenating $[v_{i,0}; ...; v_{i,N-1}]$. The LSTM we implement is a multi-layer LSTM where several LSTM layers are stacked on top of each other. The final layer embedding of the final time step is subsequently passed through a two-layer MLP to regress the acceleration values along x and y directions.

5. Experiments

5.1. Generating Ground Truth labels

The Waymo Open Dataset[9] contains 800GB of training and 200GB of validation dataset, and they include images from 5 directions. We use only the front images of the training and validation dataset because front images are the most useful information to predict acceleration, as observed in the work done by Gu et al[4] using multi-layer LSTM models. Therefore, We use only the front images of 13 training tars (32.5GB) and 3 validation tars (7.5GB) to predict acceleration.

The velocity of AV is provided in a global coordinate system. We need to transform the velocity data into a vehicle coordinate system so that we can calculate the acceleration in a vehicle coordinate system. The dataset provides a vehicle pose that transforms variables from vehicle to global coordinate(VP_v^g in short). We can calculate a vehicle pose that transforms the variables from global to vehicle coordinate(VP_g^v) by taking a matrix inversion of VP_v^g.

$$VP_g^v = VP_v^{g-1} \tag{3}$$

VP_v^g is a 4x4 matrix and we can use rotation elements R_g^v to transform velocity from global to vehicle coordinate. Here, v is a linear velocity and w is an angular velocity.

$$\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} = R_g^v, \quad \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} = VP_g^v \tag{4}$$

$$\begin{bmatrix} v_x \\ v_y \\ v_z \\ w_x \\ w_y \\ w_z \end{bmatrix}_v = \begin{bmatrix} r_{11} & r_{12} & r_{13} & 0 & 0 & 0 \\ r_{21} & r_{22} & r_{23} & 0 & 0 & 0 \\ r_{31} & r_{32} & r_{33} & 0 & 0 & 0 \\ 0 & 0 & 0 & r_{11} & r_{12} & r_{13} \\ 0 & 0 & 0 & r_{21} & r_{22} & r_{23} \\ 0 & 0 & 0 & r_{31} & r_{32} & r_{33} \end{bmatrix} \cdot \begin{bmatrix} v_x \\ v_y \\ v_z \\ w_x \\ w_y \\ w_z \end{bmatrix}_g \tag{5}$$

We calculate the acceleration of AV in a vehicle coordinate by subtracting two velocities in a vehicle coordinate and divide it by the time difference.

$$a_t = \begin{cases} a_1 & t = 0 \\ \frac{v_t - v_{t-1}}{\Delta t} & t > 0 \end{cases} \tag{6}$$

where t is the frame index, and a_t and v_t are the acceleration and velocity of the AV in the vehicle coordinate at the time frame t, respectively. We randomly select the trajectory of scenes in the dataset and compare it to the generated ground truth labels. We confirm that generated ground truth accelerations match the pattern of trajectories.

5.2. Training details

Both Spatiotemporal Video Transformer (VidT) and baseline LSTM basd model are implemented purely in PyTorch. We chose hyperparameters for both architecture so that both configurations have approximately the same number of parameters in order to aid a fair comparison. Since

both models require considerable time to train, we did little hyperparameter tuning - most of the hyperparameter choices are similar for both models, thus making comparison easier and fairer.

5.2.1 Patch formation

Each frame has spatial dimensions 1080×1920. We resize it to 1120×1792 and subsequently partition it into 5×8 grid (40 patches) with each patch of dimension 224×224. Hence the value of N is 40. These patches are then processed through the ResNet-34 encoder as shown in Figure 1.

5.2.2 Spatiotemporal Video Transformer (VidT)

The value of d - which is the latent dimension of the linear projection layer following the ResNet34 transformation of each patch - is set to 128. The model has 6 transformer layers with 8 self-attention heads each. Each self-attention head have dimensionality of 128. The dimensionality of the Feed Forward layers' projection is also set to 128. The top hidden layer of the 2-layer MLP head on top projects 128-dimensional vector to 64-dimensional space, while the last layer regresses the 64-dimensional projection to a 2-dimensional vector which are the acceleration along x and y directions. Additionally, for training - batch size was selected as 64 and an Adam[7] optimizer was used with learning rate of 7e-5. Dropout of 0.4 was applied to the transformer layers and a dropout of 0.4 was applied to the classification layers. The total number of parameters of this model with the above hyperparameters is 3.4 million.

5.2.3 Baseline LSTM-based architecture

Similar to the VidT model, the value of d is set to 128. The model has 6 stacked LSTM layers. The 2-layer MLP on top of the last LSTM layer's last time step has the same architecture and hyperparameter choices as VidT's MLP head. Additionally, for training - batch size was selected as 64 and an Adam[7] optimizer was used with learning rate of 1e-5. Dropout of 0.4 was applied to the LSTM layers and a dropout of 0.4 was applied to the classification layers. The total number of parameters of this model with the above hyperparameters is 3.4 million.

5.3. Results

We use Mean Average Error (MAE) of acceleration values along x direction and y direction denoted by MAE_x and MAE_y respectively to evaluate the performance of our models.

The experimental results on both our VidT model and LSTM-based model are tabulated in Table 1. As we can see, our VidT model outperforms, albeit marginally, the baseline LSTM model. Since both architectures have the same number of parameters, we can infer that our novel proposal parallels the performance of LSTM - one of the most popular architectures for behavior cloning. We posit that with more hyperparameter tuning and more data to train on, our VidT model should provide far superior performance since transformers are known to be data-hungry models.

Method	MAE_x	MAE_y
VidT	**0.53106**	**0.08348**
LSTM baseline	0.53114	0.08395

Table 1. Experimental results comparing MAEs along x direction - MAE_x and along y direction - MAE_y

6. Conclusion

We observed that our Spatiotemporal Video Transformer (VidT) performs better than the LSTM-based model to predict the acceleration of the AV, which supports our assumption that the Transformer with both spatial and temporal embedding would capture the history of AV motion well.

Our Spatiotemporal Video Transformer (VidT) can be improved in several ways. One limitation was we could not utilize the full amount of training data even though we preprocessed them due to limited time and resources. There are 2.5 times more training data in Waymo open dataset than we used in this work. If we use them all, we may achieve better performance by using all of them. Another idea is that the time and position embeddings can be injected in a more sophisticated way to better undertand the complex relationship between a long sequence of patches. For example, we can try to inject them in every self-attention layer as described in Deberta[6] or use the 2D location of a patch to represent positional embedding as described in ViT[3].

7. Contributions

Both authors contributed equally towards this work. The ideation and development of the architecture scheme was done together through literature survey and discussion. Code for data cleaning and generation was written by Jongwoo Park while the code for the model architectures was written by Sounak Mondal. Both also have equal contributions towards experimentation, analysis and writing this report.

References

[1] Mariusz Bojarski, Davide D Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D, Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016. 1

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 2

[3] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 1, 2, 4

[4] Zhicheng Gu, Zhihao Li, Xuan Di, and Rongye Shi. An lstm-based autonomous driving model using a waymo open dataset. *Applied Sciences*, 10(6):2046, 2020. 1, 3

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 2

[6] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*, 2020. 4

[7] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 4

[8] Yang Liu and Mirella Lapata. Text summarization with pretrained encoders. *arXiv preprint arXiv:1908.08345*, 2019. 3

[9] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2446–2454, 2020. 1, 3

[10] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017. 3