Words in Motion: Extracting Interpretable Control Vectors for Motion Transformers

Ömer Şahin Taş* **Royden Wagner***
FZI Research Center for Information Technology
Karlsruhe Institute of Technology

Abstract

Transformer-based models generate hidden states that are difficult to interpret. In this work, we aim to interpret these hidden states and control them at inference, with a focus on motion forecasting. We leverage the phenomenon of neural collapse and use linear probes to measure interpretable features in hidden states. Our experiments reveal meaningful directions and distances between hidden states of opposing features, which we use to fit control vectors for activation steering. Consequently, our method enables controlling transformer-based motion forecasting models with interpretable features, providing a unique interface to interact with and understand these models.[1]

1 Introduction

Accurately predicting sequential data while maintaining interpretability is crucial for many real-world applications. However, these two objectives often conflict, as achieving higher accuracy frequently comes at the cost of lower interpretability. This trade-off is primarily linked to the representational capacity of the underlying model: methods achieving higher accuracy tend to rely on the increased complexity of their underlying models [20, 4]. This, in turn, renders them difficult to understand and interpret in terms of semantically meaningful patterns.

Deep learning models employ loss functions that encourage the clustering of data samples based on their patterns [28]. Together with regularizers that prevent overfitting, clusters become more distinct over the course of training [15, 49]. We leverage this phenomenon to analyze the learned representations of transformer-based models with respect to human-interpretable features during training. Specifically, we use linear probing [1] to measure the degree to which these features are embedded in hidden states. In this way, we identify that interpretable features are embedded in the hidden states of transformer-based models. Building on these insights, we fit control vectors to opposing features, enabling the control of forecasts at inference.

Our application focuses on recent transformer-based motion forecasting models [26, 52, 43]. They process features of past motion sequences (i.e., past positions, orientation, acceleration, speed) and environment context (i.e., map data and traffic light states), and transform them into future motion sequences. Like other transformer models, they rely on learned representations of these features, resulting in hidden states that are difficult to interpret and control.

The contributions of this work are:

- We leverage the phenomenon of neural collapse and use linear probes to measure interpretable features in hidden states of recent motion transformers. Our analysis reveals that the collapse of these hidden states creates meaningful directions and distances in the latent space.

*Equal contribution. Corresponding author tas@fzi.de.
[1] Our implementation is available at https://github.com/kit-mrt/future-motion.

Interpretable AI: Past, Present and Future Workshop at NeurIPS 2024

- We exploit these latent space properties to fit control vectors for each interpretable feature. Our control vectors enable adjusting motion forecasts of transformer models at inference.

2 Related Work

2.1 Natural language as an interface for model interaction and control

Linking learned representations to natural language has gained significant attention (e.g., [31, 2, 22]). Broadly, approaches incorporating language in models can be categorized into four types.

Conditioning. Numerous works leverage natural language to condition generative models in diverse tasks such as image synthesis [32, 51], video generation [8], and 3D modeling [41, 48].

Prompting. Some works use language as an interface to interact with models, enabling users to request assistance or information. This includes obtaining explanations of underlying reasoning, and human-centric descriptions of model behavior [10, 34].

Enriching. Another line of work leverages LLMs' generalization abilities to enrich context embeddings, providing additional information for better prediction and planning [16].

Instructing. Natural language can be used to issue explicit commands for specific tasks, distinct from conditioning [27, 9]. The main challenge is connecting the abstractions and generality of language with environment-grounded actions [30].

In the appendix, we provide further examples of these language incorporation approaches in robotics and self-driving applications. While these works align learned text representations with embeddings of other modalities, they do not measure the degree to which interpretable features are embedded within hidden states.

2.2 Methods for model interpretability and explainability

Latent space regularities. Mikolov et al. [25] show that consistent regularities naturally emerge from the training process of word embeddings. This phenomenon, commonly referred to as the "word2vec hypothesis", suggests that learned embeddings capture both semantic and syntactic relationships between words through consistent vector offsets in latent space. Many multimodal models, including CLIP [31], use contrastive learning to align embeddings across different modalities and maximize their cosine similarity. It should be emphasized that these works measure the similarity of embeddings of all features per sample, rather than focusing on distinct features within the latent space.

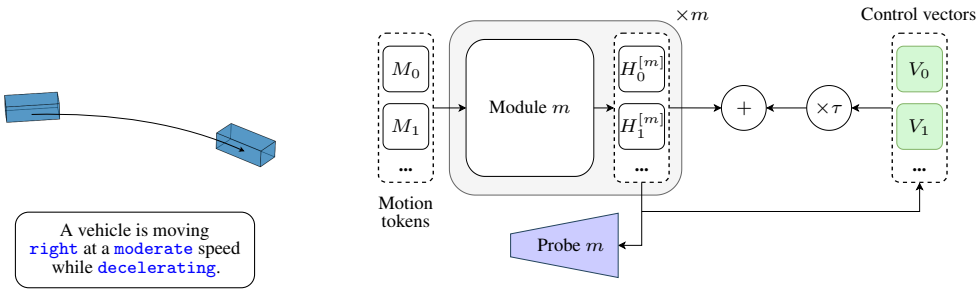
Neural collapse. A recent line of work [28, 15, 49] introduces the term *neural collapse* to describe a desirable learning behavior of deep neural networks for classification.[2] It refers to the phenomenon that learned top-layer representations form semantic clusters, which collapse to their means at the end of training. In addition, the cluster means transform progressively into equidistant vectors when centered around the global mean. Therefore, neural collapse facilitates classification tasks and is considered a desirable learning behavior for both supervised [28] and self-supervised learning [7].

Hidden state activations. Transformers consist of attention blocks, followed by simple feed-forward networks, whose hidden state activations are analyzed for interpretability. Elhage et al. [12] explore two key hypotheses that describe how these activations capture meaningful structures: the linear representation hypothesis [29] and the superposition hypothesis [3]. These hypotheses essentially state that the neural networks represent features as directions in their activation space, and that representations can be decomposed into independent features [12].

Control vectors. In natural language processing [56, 39, 42], control vectors allow targeted adjustments to model outputs by steering hidden state activations without the need for fine-tuning or prompt engineering. Control vectors are a set of vectors that capture the difference of hidden states from opposing concepts or features [33]. This approach requires a well-structured latent space, where samples are clustered according to classes or features (e.g., a high degree of neural collapse, see Section 3.2).

[2] Neural collapse is not to be confused with representation collapse [17, 5], where learned representations across all classes collapse to redundant or trivial solutions (e.g., zero vectors).

3 Method



(a) Interpretable motion features

(b) Controllable motion encoder

Figure 1: **Words in Motion. (a)** We classify motion features in an interpretable way, as in natural language. **(b)** We measure the degree to which these interpretable features are embedded in the hidden states H_i of transformer models with linear probes. Furthermore, we use our discrete features to fit interpretable control vectors V_i that allow for controlling motion forecasts at inference.

3.1 Motion feature classification using natural language

In contrast to natural language, where words naturally carry semantic meaning, motion lacks predefined labels. Therefore, we identify human-interpretable motion features by quantizing them into discrete subclasses as in natural language. Our classes of motion features are based on insights from [35].

Initially, we classify motion direction using the cumulative sum of differences in yaw angles, assigning it to either `left`, `straight`, or `right`. Additionally, we introduce a `stationary` class for stationary objects, where direction lacks semantic significance. We define further classes for speed, dividing the speed values into four intervals: `high`, `moderate`, `low`, and `backwards`. Lastly, we analyze the change in acceleration by comparing the integral of speed over time to the projected displacement with initial speed. Accordingly, we classify acceleration profiles as either `accelerating`, `decelerating`, or `constant` (see Figure 1a). The threshold values used for classification are detailed in the appendix.

3.2 Neural collapse as a metric of interpretability

We propose to measure neural collapse as a metric of interpretability. Specifically, we focus on interpreting hidden states (i.e., latent representations) and evaluate whether hidden states embed human-interpretable features. We measure how close abstract hidden states are related to interpretable semantics using linear probing accuracy [1].[3] We train linear probes (i.e., linear classifiers detached from the overall gradient computation) on top of hidden states (H_i in Figure 1). During training, we track their accuracy in classifying our interpretable features on validation sets. Adapted to motion forecasting, we choose the aforementioned motion features as interpretable semantics.

Besides linear probing accuracy, following Chen and He [11], we use the mean of the standard deviation of the ℓ_2-normalized embedding to measure representation collapse. Representation collapse refers to an undesirable learning behaviour where learned embeddings collapse into redundant or trivial representations [17, 5]. Redundant representations have a standard deviation close to zero. In a way, representing the opposite of neural collapse. As shown in [11], rich representations have a standard deviation close to $1/\sqrt{\text{dim}}$, where dim is the hidden dimension.

3.3 Interpretable control vectors

We use interpretable features to build pairs of opposing features. For each pair, we build a dataset and extract the corresponding hidden states. Afterwards, we compute the element-wise difference

[3] Ben-Shaul et al. [7] show that linear probing accuracy is consistent with the accuracy of nearest class center classifiers, which are typically used to measure neural collapse.

between the hidden states of opposing classes. Finally, we follow Zou et al. [56] and use principal component analysis (PCA) with one component to reduce the computed differences to one scalar per hidden dimension and to generate control vectors (V_i in Figure 1b). At inference, we scale the vectors with a temperature parameter (τ in Figure 1b) to control the strength of the corresponding features.

4 Experimental Setup

4.1 Motion forecasting models

We study three recent transformer models for motion forecasting in self-driving vehicles. Wayformer [26] and RedMotion [43] models employ attention-based scene encoders to learn agent-centric embeddings of past motion, map, and traffic light data. To efficiently process long token sequences, Wayformer uses latent query attention [19] for subsampling, RedMotion lowers memory requirements via local-attention [6] and redundancy reduction. HPTR [52] models learn pairwise-relative environment representations via kNN-based attention mechanisms. For Wayformer, we use the implementation by Zhang et al. [52] and the early fusion configuration. Therefore we analyze the hidden states generated by MLP-based input projectors for motion data, which consists of three layers. For RedMotion, we use the publicly available implementation with a late fusion encoder for motion data [43]. For HPTR, we use the implementation by Zhang et al. [52] and a custom hierarchical fusion setup with a modality-specific encoder for past motion and a shared encoder for environment context. Further details on fusion mechanisms, training setups, and hyperparameters are presented in the appendix.

4.2 Linear probes

We add linear probes for our quantized motion features to each hidden state of all models ($H_i^{[m]}$ in Figure 1, where $m \in \{0, 1, 2\}$ is the module number and i is the token index). These classifiers are learned during training using regular cross-entropy loss to classify speed, acceleration, direction, and the agent classes from hidden states. We decouple this objective from the overall gradient computation. Therefore, these classifiers do not contribute to the alignment of latent representations, but exclusively measure the corresponding neural collapse into interpretable clusters.

4.3 Control vectors

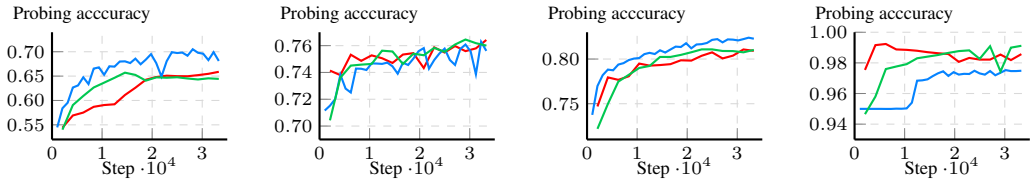
Using our interpretable motion features (see Section 3.1), we build pairs of opposing features. Specifically, we generate *speed* control vectors representing the transition from low to high speed, *acceleration* control vectors representing the transition from decelerating to accelerating, and *direction* control vectors representing the transition from the left and right direction, and *agent* control vectors representing the transition from pedestrian to vehicle (see Section 3.3). For each pair, we use the hidden states $H_i^{[m]}$ from module $m = 2$ and the last token per motion sequence (with $i = -1$), as it is closest to the start of the prediction.

5 Results

5.1 Extracting interpretable features for motion

Our approach relies on a well-structured latent space, where samples are clustered with respect to classes of features. Before evaluating the clustering behaviour, we ensure that our features are not highly correlated, as confirmed by the Spearman feature correlation analysis in the appendix. We report linear probing accuracy for interpretable features during training on both datasets.

Figure 2 shows the linear probing accuracies for our interpretable motion features on the Argoverse 2 Forecasting (abbr. AV2F) dataset. The scores are computed on the validation split over the course of training. All models achieve similar accuracy scores, while the Wayformer model achieves a slightly higher acceleration and lower agent accuracies. Overall, high linear probing accuracies for all motion features are achieved. This shows that all models likely exhibit neural collapse towards interpretable motion features.



(a) Acceleration (b) Speed (c) Direction (d) Agent

Figure 2: Probing accuracies for RedMotion, Wayformer, and HPTR on the validation split of the AV2F dataset.

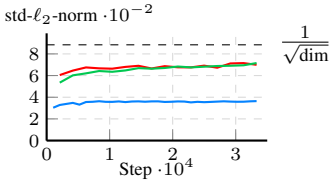
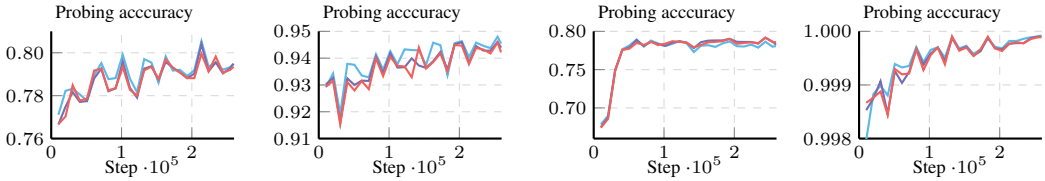


Figure 3: Normalized standard deviation representation quality metric for RedMotion, Wayformer, and HPTR.

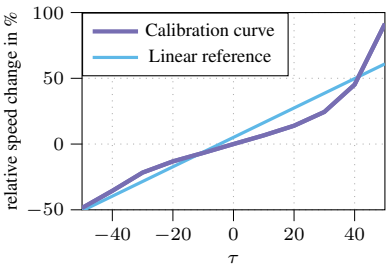
The representation quality metric normalized standard deviation of embeddings is shown in Figure 3. Both HPTR and RedMotion learn to generate embeddings with a normalized standard deviation close to the desired value of $1/\sqrt{\text{dim}}$. While the Wayformer model achieves lower scores in this metric. This reflects differences between attention-based and MLP-based motion encoders.

Figure 4 shows the linear probing accuracies for our interpretable motion features on the Waymo Open Motion (abbr. Waymo) dataset. Here, we report the scores for each of the three hidden states H_i in the RedMotion model (i.e., after each module m in the motion encoder, see Figure 1). Similar accuracy scores are reached for all features at all three hidden states. The accuracies for the acceleration and speed features progressively improve, while the direction feature reach a score of 80% early on. Compared to the direction scores on the AV2F dataset, the scores on the Waymo dataset "jump" earlier. We hypothesize that this is linked to the shorter input motion sequence on Waymo (1.1 s vs. 5 s), which limits the amount possible movements and thus simplifies classifying direction. In contrast to the AV2F dataset, higher accuracies for the speed class are achieved. Overall, the highest scores are reached for the agent features, alike on the AV2F dataset.



(a) Acceleration (b) Speed (c) Direction (d) Agent

Figure 4: Probing accuracies at module 0, module 1 and module 2 for acceleration, direction, and speed on the validation split of the Waymo dataset.



Pearson	R^2	S-idx
0.948	0.821	0.957

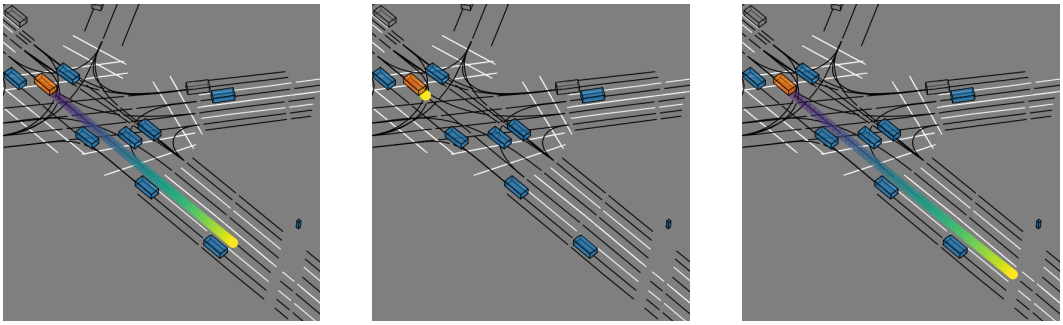
Figure 5: Linearity measures for controlling with our speed control vector along with their calibration curves.

Additionally, we measure a class-distance normalized variance (CDNV) [15] value of 0.9499 on the Waymo dataset and 2.4598 on the AV2F dataset for RedMotion. We hypothesize that the higher CDNV value on AV2F is caused by the longer past motion sequence (i.e., 5 s vs. 1.1 s on Waymo), allowing for a greater range of potential movements.

Figure 5 presents the calibration curve and linearity measures for controlling with our speed control vector. We evaluate control temperatures in the interval $\tau \in [-50, 50]$. We use the Pearson correlation coefficient, the coefficient of determination (R^2), and the straightness index (S-idx) as linearity measures. Higher linearity implies improved controllability. These plots reinforce the observation that control vectors enable linear controls, highlighting their suitability for control in forecasting tasks.

5.2 Controlling motion forecasts at inference

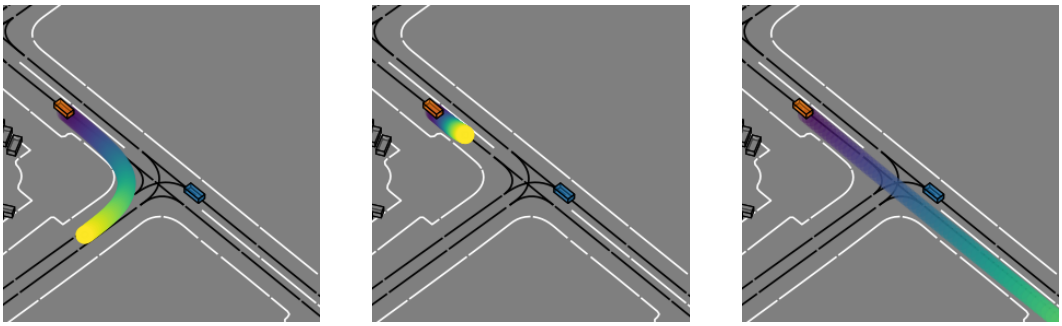
Figure 6 shows a qualitative example from the AV2F dataset. Subfigure (a) shows the default (i.e., non-controlled) top-1 (i.e., most likely) motion forecast. In subfigure (b) and (c), we apply our acceleration control vector with $\tau = -20$ and $\tau = 100$, respectively.



(a) Default motion forecast (b) Enforced strong deceleration (c) Enforced acceleration

Figure 6: **Controlling a vehicle at an intersection.** In subfigure (b) and (c), we apply our acceleration control vector with $\tau = -20$ and $\tau = 100$. The focal agent is highlighted in orange, dynamic agents are blue, and static agents are grey. Lanes are black lines and road markings are white lines.

Figure 7 shows qualitative results on the Waymo dataset. In subfigure (b) and (c), we apply our speed control vector to de- and increase the driven speed of a vehicle. Both controls affect the future speed analogously, while increasing the speed also changes the route to fit the given environment context.



(a) Default motion forecast (b) Speed controlled $\tau = -32$ (c) Speed controlled $\tau = 100$

Figure 7: **Controlling a vehicle before a predicted right turn.** In subfigure (b) and (c), we apply our speed control vector to de- and increase the driven speed of a vehicle. Both controls affect the future speed analogously, while increasing the speed also changes the route to fit the context.

6 Conclusion

We analyze "words in motion" by examining the representations associated with motion features in transformer models. First, we quantize motion features into discrete subclasses as in natural language. Our experiments on large-scale motion datasets include models with varying environment representations and fusion mechanisms. Furthermore, we analyze the hidden states of attention and MLP-based motion encoders. Specifically, we show that neural collapse towards human-interpretable classes of features occurs in recent motion transformers. Building on these insights, we fit control vectors to opposing features, and control forecasts at inference.

Our findings not only improve the practical applicability of recent motion transformer models, but also enable interpreting and manipulating internal representations of transformer models. Possible applications in self-driving vehicles include applying control vectors to motion planning and adjusting planned trajectories, whenever required. Future work can explore using features from other modalities by incorporating both static and dynamic scene elements.

Acknowledgments and Disclosure of Funding

The research leading to these results is funded by the German Federal Ministry for Economic Affairs and Climate Action within the project "NXT GEN AI METHODS – Generative Methoden für Perzeption, Prädiktion und Planung".

References

[1] G. Alain and Y. Bengio. Understanding intermediate layers using linear classifier probes. In *ICLR*, 2017.

[2] J.-B. Alayrac, J. Donahue, P. Luc, A. Miech, I. Barr, Y. Hasson, K. Lenc, A. Mensch, K. Millican, M. Reynolds, et al. Flamingo: a Visual Language Model for Few-Shot Learning. *NeurIPS*, 2022.

[3] S. Arora, Y. Li, Y. Liang, T. Ma, and A. Risteski. Linear Algebraic Structure of Word Senses, with Applications to Polysemy. *Transactions of the Association for Computational Linguistics*, 6:483–495, 2018.

[4] Y. Bahri, E. Dyer, J. Kaplan, J. Lee, and U. Sharma. Explaining neural scaling laws. *Proceedings of the National Academy of Sciences*, 2024.

[5] F. Barbero, A. Banino, S. Kapturowski, D. Kumaran, J. G. Araújo, A. Vitvitskyi, R. Pascanu, and P. Veličković. Transformers need glasses! Information over-squashing in language tasks. *arXiv:2406.04267*, 2024.

[6] I. Beltagy, M. E. Peters, and A. Cohan. Longformer: The Long-Document Transformer. *arXiv:2004.05150*, 2020.

[7] I. Ben-Shaul, R. Shwartz-Ziv, T. Galanti, S. Dekel, and Y. LeCun. Reverse Engineering Self-Supervised Learning. In A. Oh, T. Neumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *NeurIPS*, 2023.

[8] A. Blattmann, R. Rombach, H. Ling, T. Dockhorn, S. W. Kim, S. Fidler, and K. Kreis. Align your Latents: High-Resolution Video Synthesis with Latent Diffusion Models. In *CVPR*, 2023.

[9] T. Brooks, A. Holynski, and A. A. Efros. InstructPix2Pix: Learning to Follow Image Editing Instructions. In *CVPR*, 2023.

[10] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, et al. Language Models are Few-Shot Learners. *arXiv:2005.14165*, 2020.

[11] X. Chen and K. He. Exploring Simple Siamese Representation Learning. In *CVPR*, 2021.

[12] N. Elhage, T. Hume, C. Olsson, N. Schiefer, T. Henighan, S. Kravec, Z. Hatfield-Dodds, R. Lasenby, D. Drain, C. Chen, R. Grosse, et al. Toy models of superposition. *Transformer Circuits Thread*, 2022.

[13] S. Ettinger, S. Cheng, B. Caine, C. Liu, H. Zhao, S. Pradhan, Y. Chai, B. Sapp, C. R. Qi, Y. Zhou, et al. Large Scale Interactive Motion Forecasting for Autonomous Driving: The Waymo Open Motion Dataset. In *ICCV*, 2021.

[14] D. Fu, X. Li, L. Wen, M. Dou, P. Cai, B. Shi, and Y. Qiao. Drive Like a Human: Rethinking Autonomous Driving with Large Language Models. In *WACV*, 2024.

[15] T. Galanti, A. György, and M. Hutter. On the Role of Neural Collapse in Transfer Learning. In *ICLR*, 2022.

[16] L. Guan, K. Valmeekam, S. Sreedharan, and S. Kambhampati. Leveraging Pre-trained Large Language Models to Construct and Utilize World Models for Model-based Task Planning. In *NeurIPS*, 2023.

[17] T. Hua, W. Wang, Z. Xue, S. Ren, Y. Wang, and H. Zhao. On feature decorrelation in self-supervised learning. In *ICCV*, 2021.

[18] Y. Huang, J. Sansom, Z. Ma, F. Gervits, and J. Chai. DriVLMe: Enhancing LLM-based Autonomous Driving Agents with Embodied and Social Experiences. In *First Vision and Language for Autonomous Driving and Robotics Workshop*, 2024.

[19] A. Jaegle, F. Gimeno, A. Brock, O. Vinyals, A. Zisserman, and J. Carreira. Perceiver: General perception with iterative attention. In *ICML*, 2021.

[20] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei. Scaling laws for neural language models. *arXiv:2001.08361*, 2020.

[21] Y.-L. Kuo, X. Huang, A. Barbu, S. G. McGill, B. Katz, J. J. Leonard, and G. Rosman. Trajectory prediction with linguistic representations. In *ICRA*, 2022.

[22] H. Liu, C. Li, Q. Wu, and Y. J. Lee. Visual Instruction Tuning. *NeurIPS*, 2024.

[23] I. Loshchilov and F. Hutter. Decoupled Weight Decay Regularization. In *ICLR*, 2019.

[24] J. Mao, J. Ye, Y. Qian, M. Pavone, and Y. Wang. A Language Agent for Autonomous Driving. *arXiv:2311.10813*, 2023.

[25] T. Mikolov, W.-t. Yih, and G. Zweig. Linguistic Regularities in Continuous Space Word Representations. In *Proceedings of the North American Association for Computational Linguistics*, 2013.

[26] N. Nayakanti, R. Al-Rfou, A. Zhou, K. Goel, K. S. Refaat, and B. Sapp. Wayformer: Motion Forecasting via Simple & Efficient Attention Networks. In *ICRA*, 2023.

[27] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, et al. Training language models to follow instructions with human feedback. *NeurIPS*, 2022.

[28] V. Papyan, X. Y. Han, and D. L. Donoho. Prevalence of neural collapse during the terminal phase of deep learning training. *PNAS*, 2020.

[29] J. Pennington, R. Socher, and C. D. Manning. GloVe: Global Vectors for Word Representation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

[30] M. A. Raad, A. Ahuja, C. Barros, F. Besse, A. Bolt, A. Bolton, B. Brownfield, G. Buttimore, M. Cant, S. Chakera, et al. Scaling Instructable Agents Across Many Simulated Worlds. *arXiv:2404.10179*, 2024.

[31] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021.

[32] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever. Zero-shot text-to-image generation. In *ICML*, 2021.

[33] N. Rimsky, N. Gabrieli, J. Schulz, M. Tong, E. Hubinger, and A. M. Turner. Steering Llama 2 via Contrastive Activation Addition. *arXiv:2312.06681*, 2023.

[34] V. Sanh, A. Webson, C. Raffel, S. H. Bach, L. Sutawika, Z. Alyafeai, A. Chaffin, A. Stiegler, T. L. Scao, A. Raja, et al. Multitask Prompted Training Enables Zero-Shot Task Generalization. *arXiv:2110.08207*, 2021.

[35] A. Seff, B. Cera, D. Chen, M. Ng, A. Zhou, N. Nayakanti, K. S. Refaat, R. Al-Rfou, and B. Sapp. MotionLM: Multi-Agent Motion Forecasting as Language Modeling. In *ICCV*, 2023.

[36] H. Shao, Y. Hu, L. Wang, S. L. Waslander, Y. Liu, and H. Li. LMDrive: Closed-Loop End-to-End Driving with Large Language Models. In *CVPR*, 2024.

[37] M. Shridhar, L. Manuelli, and D. Fox. CLIPort:What and Where Pathways for Robotic Manipulation. In *CoRL*, 2021.

[38] C. Sima, K. Renz, K. Chitta, L. Chen, H. Zhang, C. Xie, P. Luo, A. Geiger, and H. Li. DriveLM: Driving with Graph Visual Question Answering. *arXiv:2312.14150*, 2023.

[39] N. Subramani, N. Suresh, and M. E. Peters. Extracting Latent Steering Vectors from Pretrained Language Models. In *Findings of the Association for Computational Linguistics*, pages 566–581, 2022.

[40] S. Tan, B. Ivanovic, X. Weng, M. Pavone, and P. Kraehenbuehl. Language Conditioned Traffic Generation. In *CoRL*, 2023.

[41] G. Tevet, B. Gordon, A. Hertz, A. H. Bermano, and D. Cohen-Or. MotionCLIP: Exposing Human Motion Generation to CLIP Space. In *ECCV*, 2022.

[42] A. Turner, L. Thiergart, D. Udell, G. Leech, U. Mini, and M. MacDiarmid. Activation Addition: Steering Language Models Without Optimization. *arXiv:2308.10248*, 2023.

[43] R. Wagner, Ö. Ş. Taş, M. Klemp, C. Fernandez, and C. Stiller. RedMotion: Motion Prediction via Redundancy Reduction. *TMLR*, 2024.

[44] T.-H. Wang, A. Maalouf, W. Xiao, Y. Ban, A. Amini, G. Rosman, S. Karaman, and D. Rus. Drive Anywhere: Generalizable End-to-end Autonomous Driving with Multi-modal Foundation Models. *arXiv:2310.17642*, 2023.

[45] Wayve Technologies Ltd. LINGO-1: Exploring Natural Language for Autonomous Driving, 2023. URL https://wayve.ai/thinking/lingo-natural-language-autonomous-driving/. Blog Post.

[46] L. Wen, D. Fu, X. Li, X. Cai, T. MA, P. Cai, M. Dou, B. Shi, L. He, and Y. Qiao. DiLu: A Knowledge-Driven Approach to Autonomous Driving with Large Language Models. In *ICLR*, 2024.

[47] B. Wilson, W. Qi, T. Agarwal, J. Lambert, J. Singh, S. Khandelwal, B. Pan, R. Kumar, A. Hartnett, J. K. Pontes, et al. Argoverse 2: Next Generation Datasets for Self-Driving Perception and Forecasting. *arXiv:2301.00493*, 2023.

[48] M. Wu, H. Zhu, L. Huang, Y. Zhuang, Y. Lu, and X. Cao. High-fidelity 3D Face Generation from Natural Language Descriptions. In *CVPR*, 2023.

[49] R. Wu and V. Papyan. Linguistic Collapse: Neural Collapse in (Large) Language Models. *arXiv:2405.17767*, 2024.

[50] Z. Xu, Y. Zhang, E. Xie, Z. Zhao, Y. Guo, K. K. Wong, Z. Li, and H. Zhao. DriveGPT4: Interpretable End-to-end Autonomous Driving via Large Language Modell. *arXiv:2310.01412*, 2023.

[51] L. Zhang, A. Rao, and M. Agrawala. Adding Conditional Control to Text-to-Image Diffusion Models. In *ICCV*, 2023.

[52] Z. Zhang, A. Liniger, C. Sakaridis, F. Yu, and L. Van Gool. Real-Time Motion Prediction via Heterogeneous Polyline Transformer with Relative Pose Encoding. In *NeurIPS*, 2023.

[53] X. Zheng, L. Wu, Z. Yan, Y. Tang, H. Zhao, C. Zhong, B. Chen, and J. Gong. Large Language Models Powered Context-aware Motion Prediction. *arXiv:2403.11057*, 2024.

[54] Z. Zhong, D. Rempe, Y. Chen, B. Ivanovic, Y. Cao, D. Xu, M. Pavone, and B. Ray. Language-Guided Traffic Simulation via Scene-Level Diffusion. In *CoRL*, 2023.

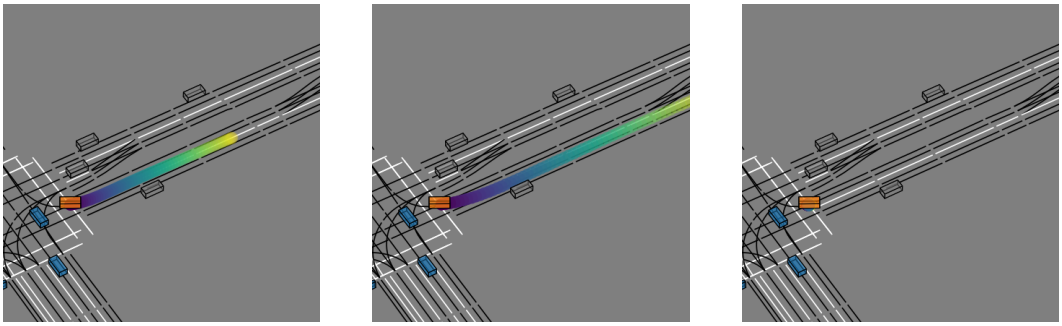
[55] B. Zitkovich, T. Yu, S. Xu, P. Xu, T. Xiao, F. Xia, J. Wu, P. Wohlhart, S. Welker, A. Wahid, Q. Vuong, V. Vanhoucke, et al. RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control. In *CoRL*, 2023.

[56] A. Zou, L. Phan, S. Chen, J. Campbell, P. Guo, R. Ren, A. Pan, X. Yin, M. Mazeika, A.-K. Dombrowski, et al. Representation Engineering: A Top-Down Approach to AI Transparency. *arXiv:2310.01405*, 2023.

A Appendix

A.1 Additional qualitative results

Figure 8 shows a qualitative example for our direction control from the Argoverse 2 Forecasting dataset. The left control leads to accelerated future motion, which is consistent with the common driving style of slowing down in front of a curve and accelerating again when exiting the curve. A strong right control makes the focal agent stationary. We hypothesize that it cancels out the actually driven left turn, resulting in a virtually stationary past.



(a) Default motion forecast (b) Left control $\tau = 10$ (c) Right control $\tau = 100$

Figure 8: **Controlling a left turning vehicle.** In subfigure (b) and (c), we apply our left-direction control vector and right-direction control vector. The focal agent is highlighted in orange, dynamic agents are blue, and static agents are grey. Lanes are black lines and road markings are white lines.

A.2 Natural language as a modality in robotics and self-driving

We provide examples illustrating how natural language is utilized as a modality in robotics and autonomous driving, building upon the general approaches introduced in the main body of the text.

Conditioning. Tan et al. [40], Zhong et al. [54] generate dynamic traffic scenes based on user-specified descriptions expressed in natural language.

Prompting. Kuo et al. [21] generate linguistic descriptions of predicted trajectories during decoding, capturing essential information about future maneuvers and interactions. More recent works employ large language models (LLMs) to analyze driving environments in a human-like manner, providing explanations of driving actions and the underlying reasoning [50, 14, 38, 45]. This offers a human-centric description of the driving environment and the model's decision-making capabilities.

Enriching. Zheng et al. [53] integrate the enriched context information of LLMs into motion forecasting models. Wang et al. [44] use LLMs for data augmentation to improve out-of-distribution generalization. Others use pre-trained LLMs for better generalization during decision-making [24, 46, 36].

Instructing. Shridhar et al. [37] enable robotic control through language-based instruction. Zitkovich et al. [55] incorporate web knowledge, enriching vision-language-action models for more generalized task performance. Huang et al. [18] demonstrate the use of instructions to guide task execution in self-driving, with experiments in simulation environments.

Although these works create a language interfaces to interact with the underlying model, in contrast to our work, they do not measure the degree to which human-interpretable motion features are embedded within hidden states.

A.3 Parameters for classifying motion features

We classify motion trajectories with a sum less than 15° degrees as `straight`. When the cumulative angle exceeds this threshold, a positive value indicates `right` direction, while a negative value – exceeding the threshold in absolute terms – indicates a `left` direction. We classify speeds between $25\,\mathrm{km\,h^{-1}}$ and $50\,\mathrm{km\,h^{-1}}$ as moderate, speeds above this range as `high`, those below as `low`, and

negative speeds as `backwards`. For acceleration, we classify trajectories as `decelerating`, if the integral of speed over time to projected displacement with initial speed is less than 0.9 times. If this ratio is greater than 1.1 times, we classify them as `accelerating`. For all other values, we classify the trajectories as having `constant` speed. We determine all threshold values by analyzing the distribution of the dataset.

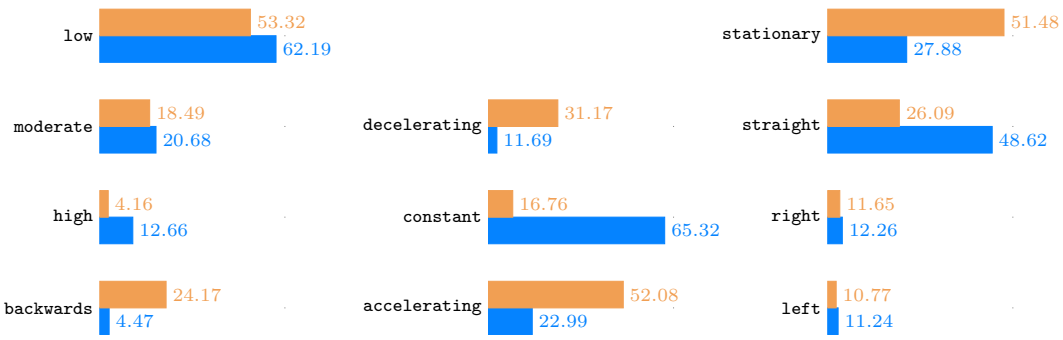
Figure 9 presents the distribution of motion subclasses across the datasets. Both datasets predominantly capture low-speed scenarios, with 62% of Waymo instances and 53% of AV2F instances falling into this category. Furthermore, a notable difference lies in the proportion of stationary vehicles, with AV2F exhibiting a significantly higher percentage (51%) compared to Waymo (28%). The Waymo dataset predominantly features vehicles with constant acceleration (65%) and traveling straight (49%), while the AV2F dataset showcases a higher proportion of accelerating instances (52%). Additionally, AV2F stands out with a much larger proportion of instances involving backward motion (24%) compared to Waymo (4%). This disparity in motion characteristics highlights that the two datasets capture different driving environments and scenarios, with Waymo potentially focusing on highway or structured urban driving, while AV2F encompasses more diverse traffic situations.

A.4 Training details and hyperparameters

We provide Wayformer and HPTR models with the nearest 512 map polylines, and RedMotion model with the nearest 128 map polylines. All models process a maximum of 48 surrounding traffic agents as environment context. For the AV2F dataset, we use past motion sequences with 50 timesteps (representing 5 s) as input. For the Waymo dataset, we use past motion sequences with 11 steps (representing 1.1 s) as input. As the main loss, we use a common combination of loss terms for motion forecasting. Following Zhang et al. [52], we use the unweighted sum of the negative log-likelihood loss for positions, cross-entropy for confidences, the cosine loss for the heading angle, and the Huber loss for velocities as the motion loss. We use AdamW [23] in its default configuration as optimizer and set the initial learning rate to 2×10^{-4}. All models have a hidden dimension of 128 and are configured to forecast $k = 6$ motion modes per agent. As post-processing, we do not perform trajectory aggregation, but follow Zhang et al. [52] and modify only the predicted confidences of redundant forecasts.

A.5 Motion forecasting metrics

Following [47, 13], we use the average displacement error (minADE), the final displacement error (minFDE), and their respective Brier variants, which account for the predicted confidences. Furthermore, we compute the miss rate, and overlap rate to evaluate motion forecasts. All metrics are computed using the minimum mode for $k = 6$ modes. Accordingly, the metrics for the mode closest to the ground truth are measured.



(a) Speed (b) Acceleration (c) Direction

Figure 9: Distributions of our motion features for the Argoverse 2 Forecasting and the Waymo Open Motion datasets. All numbers are percentages.

A.6 Feature correlation

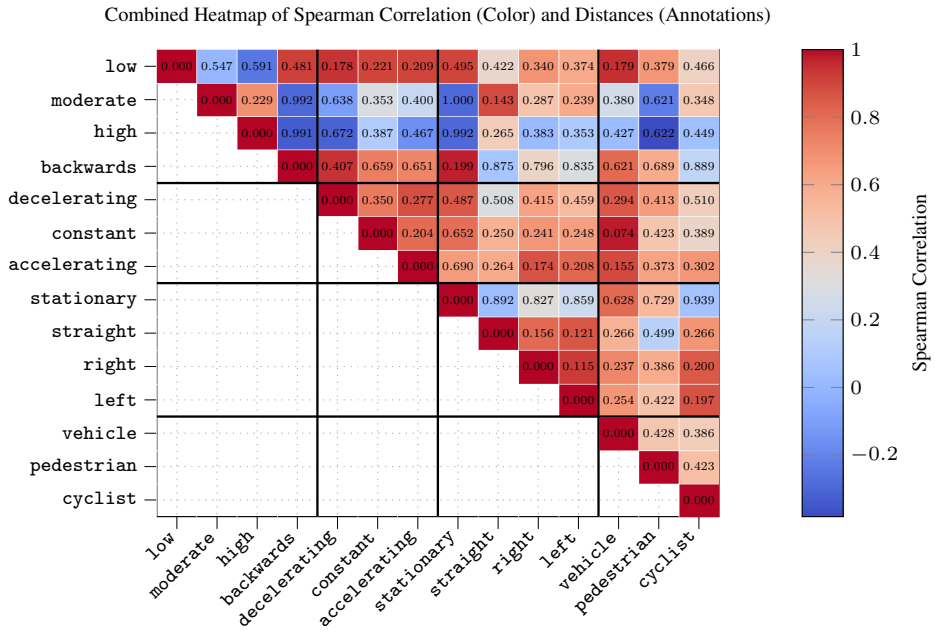


Figure 10: Heatmap representing Spearman correlation between feature cluster means for the Waymo Open Motion dataset. The values in the matrix indicate pairwise distances between clusters, normalized by the largest distance.

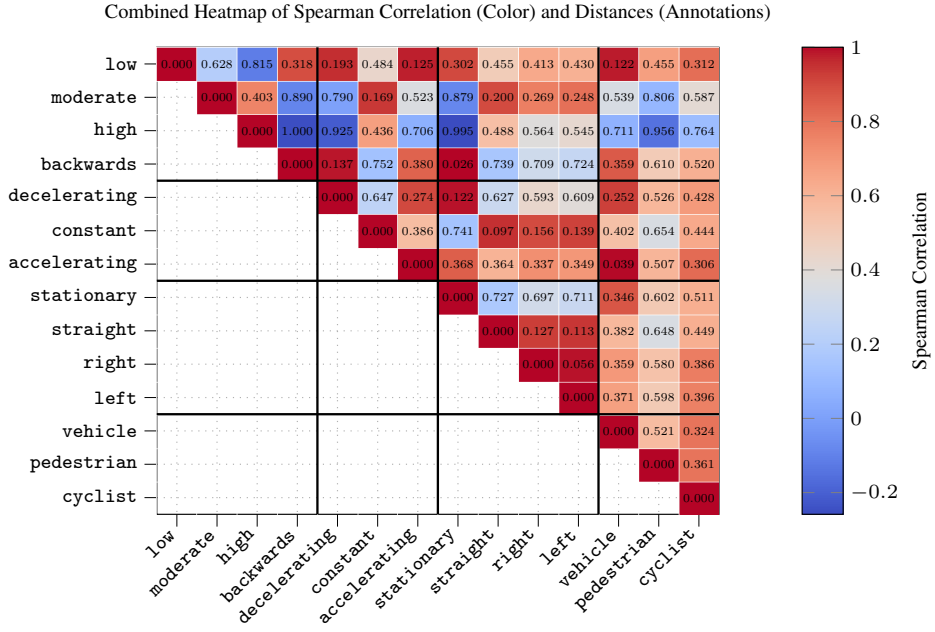


Figure 11: Heatmap representing Spearman correlation between feature cluster means for the Argoverse 2 Forecasting dataset. The values in the matrix indicate pairwise distances between clusters, normalized by the largest distance.

A.7 Early, hierarchical and late fusion in motion encoders

We define fusion types for motion transformers based on the information processed in the first attention layers within a model. In early fusion, the first attention layers process motion data of the modeled agent, other agents, and environment context. In hierarchical fusion, the first attention layers process motion data of the modeled agent, and other agents. In late fusion, the first attention layers exclusively process motion data of the modeled agent.

A.8 Title origin

The title of our work "words in motion" is inspired by our quantization method using natural language and a common notion in the computer architecture literature. In computer architecture, a word is a natural unit of data for a processing unit (e.g., CPU or GPU). In our work, words are classes of motion features, which are embedded in the hidden states of motion sequences processed by motion forecasting models.