Words in Motion: Extracting Interpretable Control Vectors for Motion Transformers

Ömer Şahin Taş* Royden Wagner*
*equal contribution

ICLR

https://github.com/KIT-MRT/future-motion

Karlsruhe Institute of Technology
FZI

Research questions

1. How do we know that our control vectors are **interpretable?**
2. How can we optimize the linearity of our control vectors **without increasing inference time?**
3. Can we apply control vectors to **robotics and self-driving**?
4. Is there any **structure in the hidden states** of motion transformer models, as in large language models?

Contributions

1. We argue that, to fit control vectors, latent space regularities with **separable features are necessary**. We use **linear probing** and show that neural collapse toward interpretable features occurs in hidden states of motion transformers, indicating a structured latent space.
2. We fit control vectors using hidden states with opposing features. By modifying hidden states at inference, we show that control vectors describe **functionally important directions**. Similar to the vector arithmetic in *word2vec*, we obtain predictions consistent with the current driving environment.
3. We use sparse autoencoders to **optimize our control vectors**. Notably, enforcing sparsity leads to more linear changes in predictions when scaling control vectors. We use linearity measures to compare these results against a **Koopman autoencoder** and evaluate the effects of various layers and activation functions, including **convolutional and MLPMixer layers**.

Neural collapse and control vectors

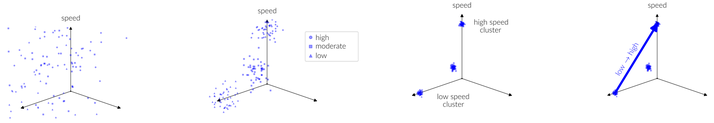
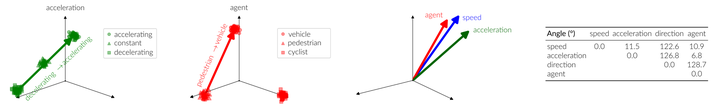


Figure 1. Clusters become more distinct over the course of training, i.e. *neural collapse*.



Angle (°)	speed	acceleration	direction	agent
speed	0.0	11.5	122.6	10.9
acceleration		0.0	126.8	6.8
direction			0.0	128.7
agent				0.0

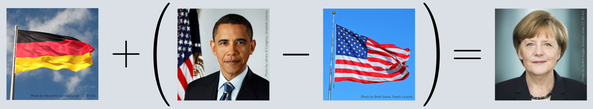
Figure 2. Comparison of control vectors.



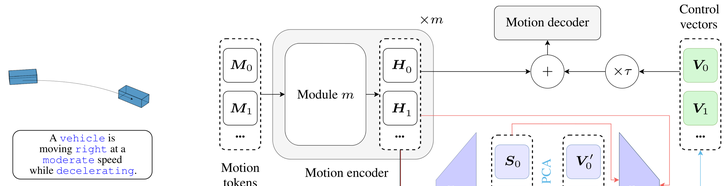
(a) Speed (b) Acceleration (c) Direction (d) Agent

Figure 3. Linear accuracies (Alain and Bengio, 2017) for RedMotion, Wayformer, and HPTR forecasting models.

Latent space arithmetic



Method



(a) Interpretable motion features (b) Controllable motion encoder

Figure 4. **(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 $\boldsymbol{H}_i$ of transformer models with linear probes. Furthermore, we use our discrete features and sparse autoencoding to fit interpretable control vectors $\boldsymbol{V}_i$ that allow for controlling motion forecasts at inference. The training of the sparse autoencoder is shown with red arrows (→) and the fitting of control vectors with blue arrows (→).

Equations

We optimize our control vectors using SAEs (Bricken et al., 2023).

$$\boldsymbol{S}^{\text{pos}}_{i,:} = \text{ReLU}\Big(\boldsymbol{W}_{\text{enc}}\big(\boldsymbol{H}^{\text{pos}}_{i,:} - \boldsymbol{b}_{\text{dec}}\big) + \boldsymbol{b}_{\text{enc}}\Big),$$

where $\boldsymbol{W}$ and $\boldsymbol{b}$ denote weights and biases of the SAE.

We compute $\boldsymbol{S}^{\text{neg}}_{i,:}$ and obtain the intermediate control vectors as

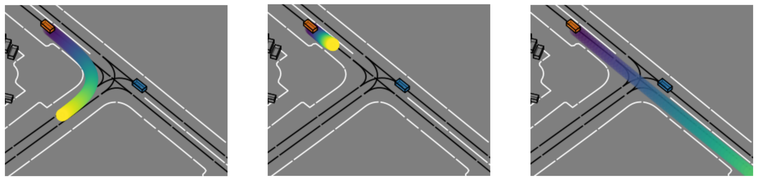
$$\boldsymbol{V}'_{i,:} = \text{PCA}\big(\boldsymbol{S}^{\text{pos}}_{i,:} - \boldsymbol{S}^{\text{neg}}_{i,:}\big).$$

Leveraging the Johnson-Lindenstrauss Lemma,[1] we decode them back to the hidden dimension of the transformer encoder

$$\boldsymbol{V}_{i,:} = \boldsymbol{W}_{\text{dec}}\boldsymbol{V}'_{i,:} + \boldsymbol{b}_{\text{dec}}.$$

At inference, we scale the control vectors with a temperature parameter τ to control the strength of the corresponding features.

Control vectors for self-driving

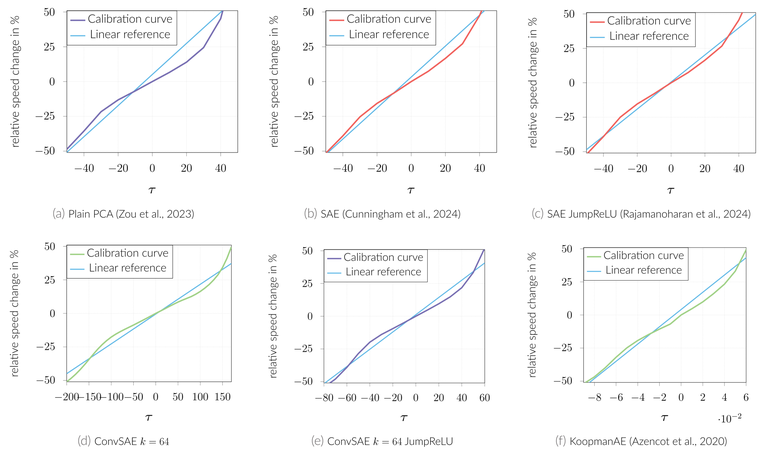


(a) Default motion forecast (b) Speed controlled $\tau = -32$ (c) Speed controlled $\tau = 100$

Figure 5. Controlling a vehicle with various temperatures.

[1] Johnson and Lindenstrauss (1984) state that a set of points in high-dimensional space can be projected into a lower-dimensional space while approximately preserving the pairwise distances between points.

How linear is the relationship between control vector scaling and model output?



(a) Plain PCA (Zou et al., 2023) (b) SAE (Cunningham et al., 2024) (c) SAE JumpReLU (Rajamanoharan et al., 2024)

(d) ConvSAE $k = 64$ (e) ConvSAE $k = 64$ JumpReLU (f) KoopmanAE (Azencot et al., 2020)

Figure 6. JumpReLU compensates feature shrinkage as reflected in a smaller range of τ values for the same range of change.

Linearity measures for control vectors

Autoencoder	Activation function	Pooling	Patch/kernel size	Pearson	R^2	S-idx
–	–	PCA	–	0.988	0.969	0.981
SAE (Bricken et al., 2023)	ReLU	PCA	–	**0.993**	0.984	0.988
SAE (Rajamanoharan et al., 2024)	JumpReLU	PCA	–	**0.993**	**0.986**	0.988
SAE (Cunningham et al., 2024)	ReLU	PCA	–	0.987	0.971	0.980
Sparse MLPMixer	ReLU	PCA	64	0.992	0.980	0.986
Sparse MLPMixer	JumpReLU	PCA	64	0.992	0.981	0.986
Sparse MLPMixer	ReLU	PCA	32	0.990	0.978	0.985
Sparse MLPMixer	JumpReLU	PCA	32	0.991	0.980	0.986
ConvSAE	ReLU	PCA	64	0.986	0.383	0.991
ConvSAE	JumpReLU	PCA	64	0.987	0.861	0.978
ConvSAE	ReLU	PCA	32	0.988	0.622	0.986
ConvSAE	JumpReLU	PCA	32	0.989	0.623	0.986
KoopmanAE (Azencot et al., 2020)	tanh	PCA	–	0.991	−0.057	**1.000**

References

Alain and Bengio. Understanding intermediate layers using linear classifier probes. In *ICLR*, 2017.
Azencot et al. Forecasting Sequential Data Using Consistent Koopman Autoencoders. In *ICML*, 2020.
Bricken et al. Towards Monosemanticity: Decomposing Language Models With Dictionary Learning. *Transformer Circuits Thread*, 2023.
Cunningham et al. Sparse Autoencoders Find Highly Interpretable Features in Language Models. In *ICLR*, 2024.
Johnson and Lindenstrauss. Extensions of Lipschitz maps into a Hilbert space. *Contemporary Mathematics*, 26:189–206, 01 1984.
Rajamanoharan et al. Jumping Ahead: Improving Reconstruction Fidelity with JumpReLU Sparse Autoencoders. In *NeurIPS*, 2024.
Zou et al. Representation Engineering: A Top-Down Approach to AI Transparency. *arXiv:2310.01405*, 2023.