Spline-based Transformers

Prashanth Chandran[1*], Agon Serifi[2,3*],
Markus Gross[1,3], and Moritz Bächer[2]

[1] DisneyResearch|Studios, Switzerland
[2] Disney Research, Switzerland
{prashanth.chandran, moritz.baecher}@disneyresearch.com
[3] ETH Zurich, Switzerland
{agon.serifi, grossm}@inf.ethz.ch

Abstract. We introduce Spline-based Transformers, a novel class of Transformer models that eliminate the need for positional encoding. Inspired by workflows using splines in computer animation, our Spline-based Transformers embed an input sequence of elements as a smooth trajectory in latent space. Overcoming drawbacks of positional encoding such as sequence length extrapolation, Spline-based Transformers also provide a novel way for users to interact with transformer latent spaces by directly manipulating the latent control points to create new latent trajectories and sequences. We demonstrate the superior performance of our approach in comparison to conventional positional encoding on a variety of datasets, ranging from synthetic 2D to large-scale real-world datasets of images, 3D shapes, and animations.

Keywords: transformer · latent splines · positional encoding

1 Introduction

Positional encoding is an essential component in transformer models, introduced in the seminal work by Vaswani *et al.* [43]. It infuses positional information into input tokens to help transformers learn position-agnostic token embeddings. Positional encoding works by (1) pre-assigning sinusoids of different frequencies and phases to every position an input token can take on in a sequence, and (2) by adding this sinusoid to the token embedding that appears at the corresponding position in the sequence. Injecting a token with positional information, also referred to as *absolute* position encoding in later work, has evolved into several variants that address shortcomings and improve generalization. For example, several works have shown that absolute position encoding limits the ability of transformers to handle longer sequences at inference time and proposed *relative* position encoding schemes where a fixed or learned bias is added to the attention matrix [19, 31, 34]. Irrespective of their exact arrangement, today's state-of-the-art transformer architectures employ a combination of absolute and relative position encoding schemes [31, 37, 38].

---

[*] equal contribution.

Positional encoding assumes that token embeddings represent elemental data in a collection, *e.g.*, individual words in a sentence, images in a video, or poses in an animation, and that an additional notion of position is required to model a collection of such elements, such as sequences of words, images, or animation frames. This thought process conceptually decouples elemental and collective datatypes and forces that separate representations for the elements and the collection as a whole are learned. This becomes even more evident when we consider that most existing architectures that learn compact neural representations for collections do not leverage the fact that individual elements, traversed in a particular order, make up a collection.

In this work, we argue that learned neural representations for elemental and collective datatypes do not have to be decoupled from one another. Instead, they can be effectively represented in a single, shared latent space. At the heart of our approach is the idea that a collection can be represented by learning to traverse a trajectory in the latent space of elemental data. Inspired by animation workflows where splines are commonly used to describe a temporal sequence of poses, we introduce a new class of transformer models based on splines that we call *Spline-based Transformers*. They do not require absolute position encoding.

At a high level, our approach uses a transformer-based encoder with additional learned control tokens to reduce an input collection of elements to a fixed number of latent control points $\in \mathbb{R}^d$. These control points are interpreted as the control points of a d-dimensional spline in latent space, representing a continuous latent space trajectory. The trajectory encapsulates the fundamental characteristics of the elements constituting the input collection. Uniformly sampling and processing the trajectory through the transformer-based decoder reconstructs the original input sequence. Our Spline-based Transformers require no sinusoidal positional encoding and, therefore, completely circumvent the downsides of absolute position encoding, including poor extrapolation and overfitting. A conceptual overview of our approach is illustrated in Fig. 1.
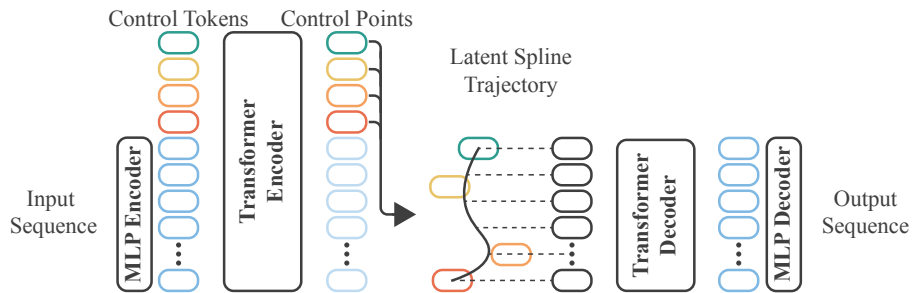

Fig. 1: Spline-based Transformers. Our Spline-based Transformers encode an input sequence, together with learnable control tokens, into a trajectory in latent space defined by the latent control points of a spline curve.

We demonstrate the superior performance of our Spline-based Transformers over transformers with conventional positional encoding on several datasets and applications, including synthetic data (Sec. 4.1), images (Sec. 4.2), animation data (Sec. 4.3), and in representing challenging geometry like hair strands (Sec. 4.4). Additionally, our Spline-based Transformers allow users to manipulate a given collection by directly interacting with corresponding latent controls, thereby introducing a new means of interacting with this architecture. With transformers gaining significant attention in recent years as general purpose architectures [2, 5, 9, 29, 30, 33], we believe that our simple yet effective approach has the potential to be leveraged across multiple disciplines for a wide variety of tasks.

Succinctly, our contributions are:

– We introduce Spline-based Transformers; Transformer models that use a spline-based latent space to encode temporal information without requiring additional positional encoding.
– We show that simple control mechanisms to manipulate the latent space are automatically learned by our models and allow for rapid manipulation of the output sequence.
– We demonstrate superior performance of Spline-based Transformers over Transformers with positional encoding on a variety of data modalities, including synthetic data, images, 3D shapes, and motion datasets.

2   Related Work

Before we discuss related work, we would like to clarify our use of the term positional encoding. The term positional encoding is widely used in the literature of coordinate-based neural networks [27, 28] to refer to a frequency encoding scheme for improvement of network training [39] wherein a low dimensional input (such as a 3D position) is mapped to a higher dimension using a collection of sinusoids of different frequencies. In our work, we refer to positional encoding as a mechanism to *introduce* positional information into inputs and outputs that are otherwise devoid of any positional information, as is common in the literature on transformers.

Transformers [43] were introduced as an alternative to traditional sequence models such as RNNs [36] and LSTMs [16]. While they initially showed remarkable performance on language tasks, their effectiveness as a general purpose neural architecture led to their quick adoption as foundational image models [9, 22, 23], in speech recognition [33], 3D and 4D modeling [2, 4, 5, 30], and more recently as backbone architectures for diffusion models [29].

The original transformers [43] relied on absolute position encoding with sinusoids to inject positional and temporal information into input tokens. Soon after, researchers identified sequence length extrapolation and overfitting as limitations of absolute positional encoding and introduced several extensions to combat them. Su *et al.* proposed *RoPE* [38], where absolute positions are encoded as a rotation matrix and relative token positions are explicitly taken into

account during attention computations for better performance and generalization. Raffel *et al*. introduced the T5 transformer [34] where a learned bias that depends on the relative distance between tokens is added to the attention matrix. They achieved a performance boost on a variety of natural language tasks. Press *et al*. proposed *ALiBi* [31] and showed that a fixed bias with a predetermined slope that depends on the attention head can improve the performance for unseen sequence lengths. An extension to ALiBi [1], which applied ideas from RoPE, further improved the performance of ALiBi in language tasks. To specifically tackle the extrapolation problem, Ruoss *et al*. proposed positional encoding with a randomized ordering of sinusoids [37] to account for longer test positions by augmenting the training distribution. A more recent and highly relevant finding in the context of our work is described by Kazemnejad *et al*. [18]. They showed that transformers with no positional encoding (NoPE) outperform most commonly used forms of position encoding in *decoder* only tasks.

However, for transformers used in learning condensed latent spaces of sequential data [4, 5, 10, 30, 32, 40], almost all current methods make use of a transformer autoencoder with relative position encoding [31, 34] and an additional [CLS] (short for classification) token [8] as input. In these applications, the transformer encoder aggregates information from the input data tokens into the [CLS] token, which is then interpreted as a condensed latent descriptor of the input sequence at the encoder's output. This latent descriptor token is appropriately position-encoded and passed through a transformer decoder to reconstruct the input. For such scenarios, the use of no positional encoding (NoPE) is not a viable solution as it reduces to an n-fold duplication of the latent descriptor, therefore passing an identical or static latent sequence to the decoder. Without any variation in its input or absolute positional encoding, the decoder fails to reconstruct the original input sequence.

Our Spline-based Transformers present a new approach to learning such condensed latent spaces for sequential data using a transformer autoencoder that does not require a positional encoding. In addition to providing significant performance benefits, our approach provides a novel control mechanism to navigate the latent spaces without any additional complexity.

3   Spline-based Transformers

The core of our architecture is constructed around a transformer autoencoder model, which incorporates a latent space between the encoder and decoder components. In the following, we first introduce the theory behind the modifications that result in our Spline-based Transformers and later describe the architectural details of the new transformer autoencoder. See Fig. 1 for an illustration.

3.1   Background: Splines

Splines have seen widespread use in function approximation, computer-aided design, and the specification and editing of animation curves in computer graphics.

They provide a means to define a curve or a trajectory with a discrete set of control points and have many desirable properties. Adjustments to control points have only a local effect, and the degree of the polynomial basis provides users with control over the smoothness of a curve.

While our modeling is agnostic to the specific spline representation, we use B-Splines in our current transformer model as they provide a good trade-off between ease of implementation and fine-grained control over shape and smoothness. A B-Spline curve is a linear combination of control points, $\mathbf{p}_i$, and basis functions, $N_{i,k}(t)$,

$$\mathbf{s}(t) = \sum_{i=0}^{n} N_{i,k}(t)\mathbf{p}_i \quad \text{for} \quad t \in [t_{k-1}, t_{n+1}], \tag{1}$$

and describes a piecewise polynomial curve where each segment has degree k [11]. The smoothness at the interface of pairs of segments is determined by the knot vector

$$T = (t_0, t_1, \ldots, t_{k-1}, t_k, t_{k+1}, \ldots, t_{n-1}, t_n, t_{n+1}, \ldots, t_{n+d}). \tag{2}$$

Note that a B-Spline curve does, in general, *not* pass through the two end control points. Only if a knot has multiplicity $k-1$, the corresponding control point will lie on the curve, reducing the continuity at that point to $\mathcal{C}^0$. If we increase the multiplicity of a knot to k, the curve is $\mathcal{C}^{-1}$ and therefore discontinuous. In more general terms, a knot with multiplicity m results in a curve that is $k - m - 1$-differentiable, and hence C^{k-m-1}, at the knot. We can always normalize the time interval so that $t \in [0, 1]$.

Splines have many desirable properties, notably:

- *Local support:* A knot span, $t_i \leq t \leq t_{i+1}$, is only affected by k control points, and a control point only has an effect on k spans. Adjustments to a control point, $\mathbf{p}_i$, have an effect on the curve between t_i and t_{i+k}.
- *Smoothness:* If the multiplicity of a knot is zero, a B-Spline curve is $\mathcal{C}^{k-1}$ and $k - 1$-differentiable. To increase the smoothness of the curve, we can always increase the degree of the polynomial basis.
- *Numerical Stability:* The theory behind B-Splines is well-understood, and numerically stable and efficient algorithms exist to evaluate them [11].

3.2   Network Architecture

Typically, the encoder of a transformer autoencoder reduces an input sequence of tokens into a single latent code. However, because transformers are sequence-to-sequence architectures, they require additional pooling mechanisms to condense information from the entire input sequence into a single latent token [8, 42]. This is usually accomplished by concatenating an additional learned token to the input sequence, and by using only the latent representation of this token as input to subsequent neural networks (*e.g.*, a decoder) or by directly using it in a

training objective [32]. The other outputs of the transformers are discarded. In classification tasks, the learned token is often referred to as the [CLS] token [8]. Finally, in order to decode the latent [CLS] token into an output sequence, it is duplicated, positional encoded appropriately, and passed through a transformer decoder that predicts the output sequence.

Instead of only appending a single [CLS] token to the input, Spline-based Transformers append a collection of ordered control tokens to the input sequence. Specifically, Spline-based Transformers append $n+1$ control tokens to the input sequence to obtain $n+1$ control points, $\mathbf{p}_i$, that will be used to evaluate a latent spline at the output of the encoder with polynomial basis of order k. Latent codes corresponding to each output token are produced by evaluating the spline at the token's position according to Eq. (1).

The resulting trajectory, $\mathbf{s}(t)$, in latent space, has several advantages compared to a traditional positional encoding. First, the latent code is not perturbed by positional information, meaning the decoder does not need to learn to distinguish between positional and contextual information. Second, when using sinusoidals to encode the position of tokens, the contextual part of the token remains fixed and therefore provides a form of redundancy; our latent spline trajectories encode the temporal information implicitly, *e.g.*, they can traverse the latent space faster in certain points and slower in others, making better use of the latent space. In Fig. 2, we show an overview of how our spline-based latent trajectories are derived from the control points and how they differ from commonly used schemes like ALiBi [31].

Architecture Details As seen in Fig. 1, an input sequence is encoded using an MLP that is shared across input tokens, leading to an embedded sequence. Learnable control tokens are concatenated to the embedded sequence and sent through a seq2seq transformer encoder block. We add a linear layer after the last transformer encoder block to map the encoded tokens to the latent space dimension d. The exact number of evaluations of the spline depends on the number of output tokens expected at the decoder's output. Our transformer decoder uses the same structure as our encoder. The encoder and decoder have n-layers, each layer has h heads, and c feature dimensions. In every layer of the transformer, an ALiBi attention bias is added. Each layer, except the last MLP of the decoder, uses the GeLU activation [15]. While our transformer blocks follow the structure of the T5 transformer model [34], any transformer block could be used in combination with the spline-based latent space. Depending on the complexity of the data type, we use transformer blocks of varying feature dimensions and capacities. For training our Spline-based Transformer autoencoder, we use the RAdam optimizer [21] with a cosine annealing learning rate scheduler [25].

4   Experiments

We now present experiments on a number of datasets to demonstrate the effectiveness of the proposed Spline-based Transformers when applied to multiple
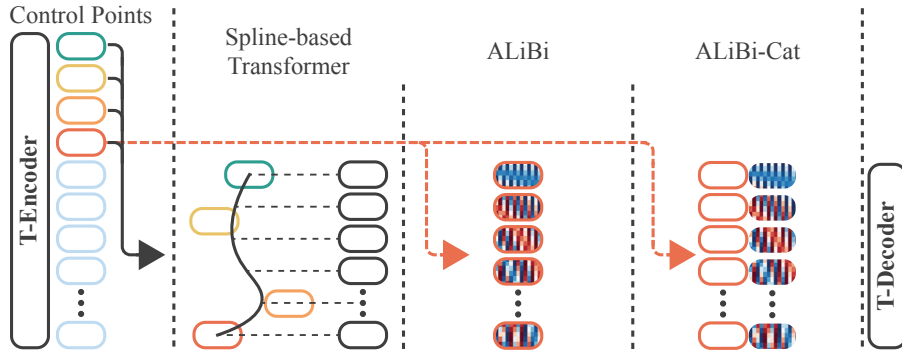
Fig. 2: Variations of Latent Spaces. Our *Spline-based Transformers* use multiple control points to evaluate a latent B-Spline and to create a d dimensional trajectory in the model's latent space. On the other hand, *ALiBi* duplicates a single control point and adds positional information to the duplicated points, while the positional information is concatenated to the duplicated control point in *ALiBi-Cat*.

modalities of sequential data. We specifically compare our method against ALiBi [31], a state-of-the-art transformer model that uses a combination of both absolute and relative positional encoding. Because ALiBi adds sinusoids to the input token embedding, which could create an ambiguity between the token's content and position for the transformer decoder to disentangle, we also compare against a variation of ALiBi, where the sinusoids are concatenated with the token embedding, effectively doubling the size of the transformer decoder blocks. We refer to this concatenated variation as ALiBi-Cat. Fig. 2 illustrates the differences between our spline-based latent space and the two baselines; ALiBi uses a single control point and adds positional information on top to create a latent sequence [4, 7], while ALiBi-Cat concatenates the control point and the positional information instead.

For our experiments, we parameterize the latent space between the encoder and decoder blocks with cubic Bézier curves, with four control points per segment. Bézier curves are one instance of the B-Spline family, and provide sufficient smoothness for the applications we have studied so far. We uniformly sample the latent spline trajectory in the range $t \in [0, 1]$. We summarize the network parameters for each experiment in our supplemental material.

4.1   Synthetic Datasets

We first evaluate our Spline-based Transformer, ALiBi, and ALiBi-Cat in representing parametric 2D curves that have a known latent space size. For this task, we use three different parametric curve families: (1) Lissajous ($d = 3$), (2) Hypotrochoids ($d = 4$), (3) Bézier curves (with $d = 2$, and $d = 64$). For each curve type, we create three different transformer autoencoders for the Spline-based Transformer, ALiBi, and ALiBi-Cat, respectively. As seen in Fig. 2, the

network architectures for the different autoencoders are identical, with the only difference being the mechanism used to derive latent token trajectories. The dimensionality of latent token embedding is decided based on the known latent space of the curve family. We train the three transformer autoencoders independently on each curve family. For training, we randomly sample curve parameters according to the parameterization of the curve in a pre-determined domain. Using the sampled parameters, we evaluate the curve to create a sequence of 256 2D tokens that contain the (x, y) coordinates of the curve, which are then fed as input to the transformer encoder. The three transformer autoencoders are trained end-to-end using a simple L2 reconstruction objective. In Tab. 1, we show the reconstruction error of each of the transformer models when presented with 10,000 unseen curves from the family it was trained on. Our Spline-based Transformer outperforms ALiBi and ALiBi-Cat, especially on low dimensional latent spaces. Some qualitative comparisons are shown in Fig. 3.

Table 1: Average reconstruction error (MSE) of 10000 test curves in 2D

Method	Lissajous (3D)	Hypotrochoids (4D)	Bézier (2D)	Bézier (64D)
ALiBi	1e-4	2e-3	1.76e-2	3.88e-3
ALiBi-Cat	8e-4	5.3e-3	1.78e-2	3.89e-3
Spline (Ours)	**3e-5**	**1.4e-3**	**2e-6**	**3.87e-3**

4.2   Images

We continue by showing the effectiveness of Spline-based Transformers in reconstructing real image datasets. For the following experiment, we divide an image into distinct non-overlapping patches to create a sequence of patches similar to (Masked) Vision Transformers [9, 14]. The sequence of 2D image patches is then the input to our transformer autoencoder. We replace the MLP-Encoder in Fig. 1 with a CNN-Encoder that maps each patch of size $(PS, PS, 3)$ to a d-dimensional latent token $(1, d)$. An image (H, W) is therefore represented with a sequence of size $(HW/PS^2, d)$. The rest of the transformer autoencoder remains identical to what was described above. We train the transformer autoencoder using a simple L2 reconstruction loss to recover the input image from the patch sequence and compare the performance of the Spline-based Transformer against ALiBi and ALiBi-Cat.

We present results on three different image datasets: CIFAR-10 [20] (32x32), AFHQ [6] (128x128), and a dataset containing facial images [3] (128x128). For each dataset and method, we train three transformers with three different latent sizes: 32D, 64D, and 128D. Tab. 2 summarizes the results. The spline-based latent space significantly outperforms the baselines by a factor of 2. Fig. 4 shows examples of the reconstructed images and their corresponding error maps; the spline-based latent space results in sharper and more detailed images. We ob-
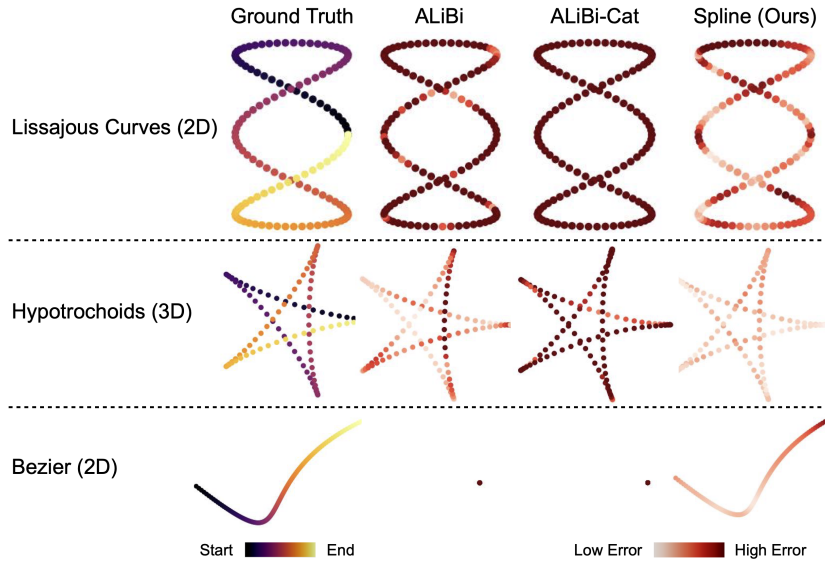
Fig. 3: Our Spline-based Transformer can successfully reconstruct curves of different families with consistently better performance than ALiBi and ALiBi-Cat. In certain scenarios (third row), reconstructions from ALiBi and ALiBi-Cat can collapse to a single point, while our Spline-based Transformer successfully manages to recover the input curve.

serve that the performance improvements are larger for lower dimensional latent spaces. More results are reported in our supplementary.

Table 2: Image Reconstruction. Comparison across different datasets and bottleneck dimensions. **Bold** indicates the best overall performance, and <u>underline</u> the best in each category. Performance is measured in Mean Squared Error (MSE).

Method	CIFAR-10			AFHQ			Faces [3]		
	32D	64D	128D	32D	64D	128D	32D	64D	128D
ALiBi	0.266	0.178	0.107	0.064	0.050	0.038	10.65e-3	8.56e-3	6.71e-3
ALiBi-Cat	0.264	0.174	0.108	0.064	0.049	0.038	10.87e-3	8.56e-3	7.14e-3
Spline (Ours)	<u>0.107</u>	<u>0.056</u>	**0.042**	<u>0.038</u>	<u>0.030</u>	**0.025**	<u>6.77e-3</u>	<u>5.27e-3</u>	**4.52-3**

4.3   Animation

A commonly encountered modality of sequential data is 3D animation, with interest in using transformers for learning motion manifolds garnering significant attention in recent years [2, 4, 7, 30]. A Spline-based Transformer, when used to
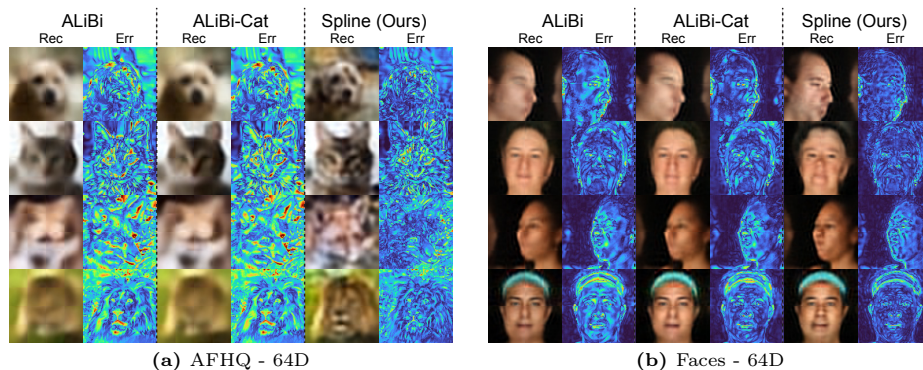
(a) AFHQ - 64D   **(b)** Faces - 64D

Fig. 4: Visual Image Reconstruction. Comparison between ALiBi, ALiBi-Cat, and Spline (ours) using a 64D bottleneck on AFHQ and Faces [3].

represent 4D data, like a sequence of 3D meshes from a facial animation or a sequence of joint poses describing human motion, can lead to notable performance benefits.

Faces We compare the performance of the Spline-based Transformer autoencoder against ALiBi, and ALiBi-Cat, training the three models on a database of 3D facial animations [4]. Each animation is represented by a sequence of registered 3D meshes. We decimate them to meshes with around 5,000 vertices, and flatten the vertices to a vector. A flattened animation sequence is thereafter split into windows of size 30 ($\sim$1 second of animation) and used to train three variations of the transformer autoencoder. In Tab. 3, we show the reconstruction error on six performances from three unseen identities. Irrespective of the dimensionality of the latent space, the Spline-based Transformer outperforms both ALiBi and ALiBi-Cat. A qualitative visualization of a reconstructed test performance is shown in Fig. 5 for the 64D Spline-based Transformer model. We note that the ALiBi transformer decoder used in these experiments is conceptually similar to the ones used in previous works [2, 4, 7, 30], indicating that Spline-based Transformers could lead to improved performance in several downstream tasks.

Table 3: Face Performance Reconstruction. Comparison across different latent dimensions. **Bold** indicates the best overall performance, and <u>underline</u> the best in each category. Performance is measured in Mean Squared Error (MSE).

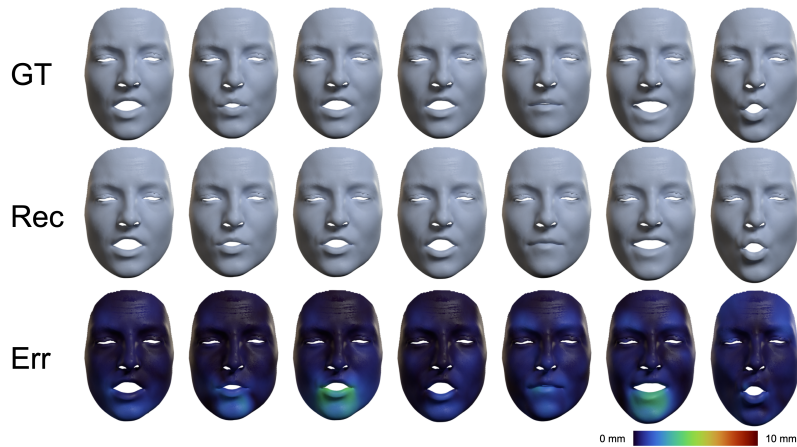| Method	32D	64D	128D	256D
ALiBi	1.58	1.55	1.48	1.54
ALiBi-Cat	1.60	1.54	1.53	1.50
Spline (Ours)	<u>1.43</u>	**1.35**	<u>1.47</u>	<u>1.47</u>

Fig. 5: Our Spline-based Transformer is able to successfully represent facial animations, preserving both the identity and expression of the subject throughout the performance.

Full-Body Motion We evaluate our method on the full-body human motion dataset HumanML3D [12], a combination of the HumanAct12 [13] and AMASS dataset [26]. In Tab. 4, we report the mean squared reconstruction error of per-frame joint positions measured in degrees. Spline-based Transformers show reconstruction improvements of at least a factor two, reducing the mean joint error of the smallest model (16D) from $\sim 0.4°$ to $\sim 0.2°$, and up to $\sim 0.07°$ for the largest model (64D). We use the joint rotations to compute SMPL body model parameters [24] and visualize the reconstruction in Fig. 6 along with the reconstruction error in mm. Positional encoded transformers have recently received a lot of attention in full-body motion reconstruction and synthesis [10, 30, 41]. We believe that Spline-based Transformers can help to increase the performance of various state-of-the-art models designed for these applications.

Table 4: Human Motion Reconstruction. Comparison across different latent dimensions. **Bold** indicates the best overall performance, and <u>underline</u> the best in each category. We report the Mean Squared Error (MSE) between joint angles [deg].

Method	16D	32D	64D
ALiBi	0.151	0.103	0.059
ALiBi-Cat	0.153	0.103	0.051
Spline (Ours)	<u>0.054</u>	<u>0.022</u>	**0.006**

Motion Editing We also show that motions can be modified by applying simple operations to the control points. In Fig. 6, we visualize two resulting motions where the control points have been modified to be closer or further away from the end-points of the spline. The motions preserve the overall style but change in speed and detail. This experiment suggests that latent splines behave smoothly in a neighborhood and edits result in plausible motions. We observe that motions can easily be toned down or amplified as we show with more results in the accompanying video. The spline-based latent space further allows us to super-sample motions. By sampling the latent spline more densely before decoding, we can achieve up to a 4x upsampling of a motion clip. This method effectively preserves the original motion characteristics, as demonstrated in the video. Multiple splines can be combined to represent longer sequences of motions. Each of the segments can then be modified individually.
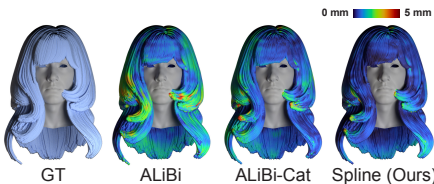

Fig. 6: Full Body Reconstruction and Modification. Upper rows: Reconstruction quality on a motion. Lower rows: Two reconstruction results after modifying the control tokens.

4.4   Geometric Representation

Finally, we show how Spline-based Transformers can also be used to model complex geometry like hair strands [35, 44], which present themselves as 3D curves in space. For this experiment, we use a dataset of 343 unique 3D hairstyles [17], where each hairstyle contains 10,000 strands, and each strand has 100 points. We are interested in representing only the strand geometry in our experiment, so we consider the strands across the hairstyles as individual 3D curves in space. We normalize the root position of each strand by translating it to the origin. Each normalized strand is therefore a sequence of a 100 vertices and is used to train a transformer autoencoder as before with an L2 reconstructive loss. We report the reconstruction error of strands from 10 test hairstyles in Tab. 5. While a thorough comparison to state-of-art methods [35, 44] is required to demonstrate the real effectiveness of Spline-based Transformers for this task, our initial tests indicate that they could be an interesting architectural alternative. For the purpose of visualizing the reconstructed strands as a coherent hairstyle, we apply the ground truth root position to the reconstructed strands in Tab. 5.

Table 5: Strand Reconstruction. Comparison across different latent dimensions. **Bold** indicates the best overall performance, and <u>underline</u> the best in each category. Performance is measured in Mean Squared Error (MSE).

Method	8D	16D	32D
ALiBi	4.8e-3	2.0e-3	1.5e-3
ALiBi-Cat	4.6e-3	1.9e-3	1.2e-3
Spline (Ours)	<u>1.09e-3</u>	<u>1.06e-3</u>	**9.4e-4**

4.5   Practical Limitations

Spline-based Transformers not only achieve a better performance than conventional transformers as demonstrated by our experiments, but can also achieve this performance improvement much faster than the traditional positional encoded models. Fig. 7a shows the validation loss on an image experiment.

While they converge faster, we observe that the Spline-based Transformers are sensitive to learning rates. Fig. 7b shows the same run with three different learning rates. A large learning rate can lead to a model collapse where the control points start to converge to the same point in latent space; leading to the same latent code for each token in the input sequence and the model not being able to recover from this state. We also observe that having a too small learning rate can harm performance significantly. We believe that a specialized scheduling strategy could significantly improve the stability and performance of Spline-based Transformers and leave this as future work.
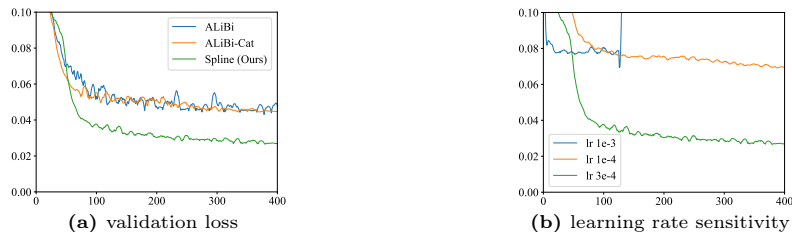
(a) validation loss

(b) learning rate sensitivity

Fig. 7: Training Performance (AFHQ). (a) shows the validation loss of the different methods. (b) shows the sensitivity of Spline-based Transformers to the learning rate.

Fig. 8: Latent Splines (AFHQ). Visualization of predicted latent spline trajectories in $d = 64$ and $d = 8$. x-axis: position along the spline; y-axis: value of the feature.

4.6   Latent Space Visualization

After training, we can visualize the latent spline trajectories (see Fig. 8). The trajectories show similar characteristics with sinusoidals but are more complex and asymmetric curves. In some parts, the change of the features is more rapid, while other dimensions propagate the same value over the whole sequence length.

5   Conclusion

In this work, we introduced Spline-based Transformers, a new class of Transformer models that eliminate the need for absolute positional encoding by combining temporal and contextual information into a single trajectory, represented by a latent spline curve. We presented the superior performance of Spline-based Transformers across a variety of datasets, from simple curves to complex animation data and images. The experiments show significant performance improvements over traditional positional encoded transformer models. Spline-based Transformers are trivial to implement yet effective and have no additional computational overhead. We identify improvements to the training stability as future work to reduce the sensitivity to training hyperparameters such as, *e.g.*, the learning rate. The spline-based latent space introduced by our method opens up a new way to interact with latent spaces using straightforward modifications of the latent control points. We hope that our work encourages research towards a new class of transformers with controllable latent spaces across a variety of applications.

References

1. Al-Khateeb, F., Dey, N., Soboleva, D., Hestness, J.: Position interpolation improves alibi extrapolation. CoRR **abs/2310.13017** (2023). `https://doi.org/10.48550/ARXIV.2310.13017`, `https://doi.org/10.48550/arXiv.2310.13017`
2. Aneja, S., Thies, J., Dai, A., Nießner, M.: Facetalk: Audio-driven motion diffusion for neural parametric head models (2023)
3. Chandran, P., Bradley, D., Gross, M., Beeler, T.: Semantic deep face models. In: 2020 International Conference on 3D Vision (3DV). pp. 345–354. IEEE Computer Society, Los Alamitos, CA, USA (nov 2020)
4. Chandran, P., Zoss, G., Gross, M., Gotardo, P., Bradley, D.: Facial animation with disentangled identity and motion using transformers. Computer Graphics Forum **41**(8), 267–277 (2022)
5. Chandran, P., Zoss, G., Gross, M., Gotardo, P., Bradley, D.: Shape transformers: Topology-independent 3d shape models using transformers. In: Computer Graphics Forum. vol. 41, pp. 195–207. Wiley Online Library (2022)
6. Choi, Y., Uh, Y., Yoo, J., Ha, J.W.: Stargan v2: Diverse image synthesis for multiple domains. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2020)
7. Daněček, R., Chhatre, K., Tripathi, S., Wen, Y., Black, M., Bolkart, T.: Emotional speech-driven animation with content-emotion disentanglement. ACM (Dec 2023). `https://doi.org/10.1145/3610548.3618183`, `https://emote.is.tue.mpg.de/index.html`
8. Devlin, J., Chang, M., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. CoRR (2018)
9. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale (2021)
10. Duan, Y., Shi, T., Zou, Z., Lin, Y., Qian, Z., Zhang, B., Yuan, Y.: Single-shot motion completion with transformer. arXiv preprint arXiv:2103.00776 (2021)
11. Farin, G.: Curves and surfaces for CAGD: a practical guide. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 5th edn. (2001)
12. Guo, C., Zou, S., Zuo, X., Wang, S., Ji, W., Li, X., Cheng, L.: Generating diverse and natural 3d human motions from text. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5152–5161 (June 2022)
13. Guo, C., Zuo, X., Wang, S., Zou, S., Sun, Q., Deng, A., Gong, M., Cheng, L.: Action2motion: Conditioned generation of 3d human motions. In: Proceedings of the 28th ACM International Conference on Multimedia. pp. 2021–2029 (2020)
14. He, K., Chen, X., Xie, S., Li, Y., Dollár, P., Girshick, R.: Masked autoencoders are scalable vision learners. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 16000–16009 (2022)
15. Hendrycks, D., Gimpel, K.: Gaussian error linear units (gelus). arXiv preprint arXiv:1606.08415 (2016)
16. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)
17. Hu, L., Ma, C., Luo, L., Li, H.: Single-view hair modeling using a hairstyle database. ACM Trans. Graph. **34**(4) (jul 2015)

18. Kazemnejad, A., Padhi, I., Natesan, K., Das, P., Reddy, S.: The impact of positional encoding on length generalization in transformers. In: Thirty-seventh Conference on Neural Information Processing Systems (2023), `https://openreview.net/forum?id=Drrl2gcjzl`

19. Ke, G., He, D., Liu, T.Y.: Rethinking positional encoding in language pre-training. In: International Conference on Learning Representations (2021), `https://openreview.net/forum?id=09-528y2Fgf`

20. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images (2009)

21. Liu, L., Jiang, H., He, P., Chen, W., Liu, X., Gao, J., Han, J.: On the variance of the adaptive learning rate and beyond. In: International Conference on Learning Representations (2020), `https://openreview.net/forum?id=rkgz2aEKDr`

22. Liu, Z., Hu, H., Lin, Y., Yao, Z., Xie, Z., Wei, Y., Ning, J., Cao, Y., Zhang, Z., Dong, L., Wei, F., Guo, B.: Swin transformer v2: Scaling up capacity and resolution. In: International Conference on Computer Vision and Pattern Recognition (CVPR) (2022)

23. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) (2021)

24. Loper, M., Mahmood, N., Romero, J., Pons-Moll, G., Black, M.J.: SMPL: A skinned multi-person linear model. ACM Trans. Graphics (Proc. SIGGRAPH Asia) **34**(6), 248:1–248:16 (Oct 2015)

25. Loshchilov, I., Hutter, F.: SGDR: Stochastic gradient descent with warm restarts. In: International Conference on Learning Representations (2017), `https://openreview.net/forum?id=Skq89Scxx`

26. Mahmood, N., Ghorbani, N., Troje, N.F., Pons-Moll, G., Black, M.J.: AMASS: Archive of motion capture as surface shapes. In: International Conference on Computer Vision. pp. 5442–5451 (Oct 2019)

27. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: representing scenes as neural radiance fields for view synthesis. Commun. ACM **65**(1), 99–106 (dec 2021). `https://doi.org/10.1145/3503250`, `https://doi.org/10.1145/3503250`

28. Park, J.J., Florence, P., Straub, J., Newcombe, R., Lovegrove, S.: Deepsdf: Learning continuous signed distance functions for shape representation. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019)

29. Peebles, W., Xie, S.: Scalable diffusion models with transformers. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 4195–4205 (October 2023)

30. Petrovich, M., Black, M.J., Varol, G.: Action-conditioned 3D human motion synthesis with transformer VAE. In: International Conference on Computer Vision (ICCV) (2021)

31. Press, O., Smith, N.A., Lewis, M.: Train short, test long: Attention with linear biases enables input length extrapolation. In: The Tenth International Conference on Learning Representations, ICLR. OpenReview.net (2022), `https://openreview.net/forum?id=R8sQPpGCv0`

32. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., Sutskever, I.: Learning transferable visual models from natural language supervision. CoRR **abs/2103.00020** (2021)

33. Radford, A., Kim, J.W., Xu, T., Brockman, G., Mcleavey, C., Sutskever, I.: Robust speech recognition via large-scale weak supervision. In: Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., Scarlett, J. (eds.) Proceedings

of the 40th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 202, pp. 28492–28518. PMLR (23–29 Jul 2023), `https://proceedings.mlr.press/v202/radford23a.html`

34. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P.J.: Exploring the limits of transfer learning with a unified text-to-text transformer. J. Mach. Learn. Res. **21**(1) (jan 2020)

35. Rosu, R.A., Saito, S., Wang, Z., Wu, C., Behnke, S., Nam, G.: Neural strands: Learning hair geometry and appearance from multi-view images. ECCV (2022)

36. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. nature **323**(6088), 533–536 (1986)

37. Ruoss, A., Delétang, G., Genewein, T., Grau-Moya, J., Csordás, R., Bennani, M., Legg, S., Veness, J.: Randomized positional encodings boost length generalization of transformers. In: 61st Annual Meeting of the Association for Computational Linguistics (2023)

38. Su, J., Lu, Y., Pan, S., Murtadha, A., Wen, B., Liu, Y.: Roformer: Enhanced transformer with rotary position embedding (2023)

39. Tancik, M., Srinivasan, P.P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J.T., Ng, R.: Fourier features let networks learn high frequency functions in low dimensional domains. NeurIPS (2020)

40. Tevet, G., Gordon, B., Hertz, A., Bermano, A.H., Cohen-Or, D.: Motionclip: Exposing human motion generation to clip space. In: Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXII. pp. 358–374. Springer (2022)

41. Tevet, G., Raab, S., Gordon, B., Shafir, Y., Cohen-or, D., Bermano, A.H.: Human motion diffusion model. In: The Eleventh International Conference on Learning Representations (2023), `https://openreview.net/forum?id=SJ1kSyO2jwu`

42. Touvron, H., Cord, M., El-Nouby, A., Bojanowski, P., Joulin, A., Synnaeve, G., Jégou, H.: Augmenting convolutional networks with attention-based aggregation. CoRR **abs/2112.13692** (2021)

43. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems, NeurIPS **30** (2017)

44. Zhou, Y., Chai, M., Pepe, A., Gross, M., Beeler, T.: Groomgen: A high-quality generative hair model using hierarchical latent representations. ACM Transactions on Graphics **42**(6), 1–16 (Dec 2023)