Lab4 (b): CSC_52002_EP Human motion generation

Advanced deep learning

February 2, 2026

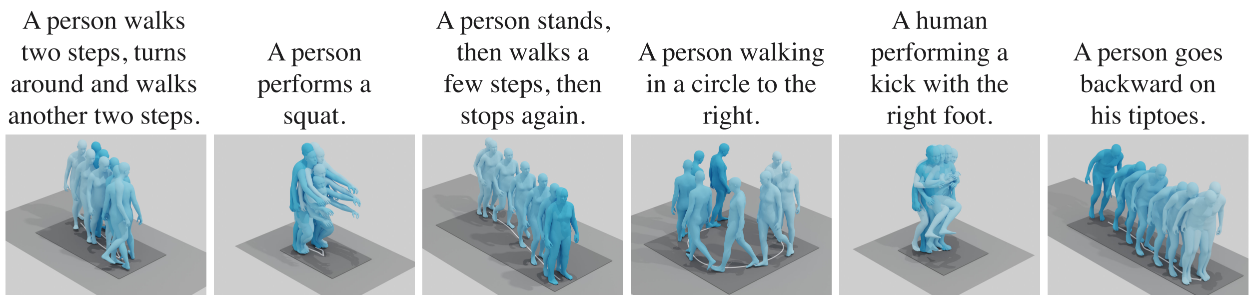

Figure 1: Examples of text-to-motion generation results [7].

In this lab, we explore the field of **text-to-human motion generation**. The goal is to generate a human motion sequence that corresponds to a given text caption, as illustrated in Figure 1.

The lab is structured as follows: first, we examine the **dataset and human motion representation** (**Section 1**); next, we study the **diffusion framework** (**Section 3**); then, we explore **different architectures** (**Section 2**); and finally, we **analyze the results** (**Section 4**).

Code 1: Upload the notebook in colab, set a GPU runtime and run cells of the `Setup` section to setup the following GitHub repository.

1 Human motion dataset and representation

1.1 HumanML3D dataset

HumanML3D dataset [1] is a 3D human motion-language dataset built from motion capture (mocap) acquisitions. It encompasses a wide range of human actions, including daily activities (e.g., "walking," "jumping"), sports (e.g., "swimming," "playing golf"), acrobatics (e.g., "cartwheel"), and artistic movements (e.g., "dancing").

Question 1: How many motions and descriptions are included in the HumanML3D dataset, and what are their average lengths?

Answer: HumanML3D dataset consists of 14,616 motions and 44,970 descriptions composed. The total length of motions amounts to 28.59 hours. The average motion length is 7.1 seconds, while the average description length is 12 words.

1.2 SMPL representation

The SMPL [5] representation is a realistic 3D mesh model of the human body, learned from thousands of 3D body scans. It is based on a set of trainable parameters that capture body shape and pose variations.

Given a set of input parameters, the SMPL model infers a detailed 3D mesh consisting of vertices (points in 3D space that define the shape) and faces (triangular surfaces connecting the vertices to form the body structure).

Question 2: What are the different input parameters of the SMPL model used to infer the vertices, and what do they represent? **Hint:** Check the official code of SMPL models.
Answer: The different SMPL input parameters are: **body shapes** (betas), **global orientation** (global rotation of the body), **body pose** (local rotations of the body in the global orientation framework), and **translation** (root translation of the body).

We cannot directly learn to generate the SMPL parameters directly, we need to use more compact features. We represent a human motion sequence $\mathbf{x} \in \mathbb{R}^{F \times d}$ where F is the number of frames and d the feature dimension. We have $\mathbf{x} = [r_z, \dot{r}_x, \dot{r}_y, \dot{\alpha}, \theta, \mathbf{j}]$, where r_z is the Z (up) coordinate of the pelvis, $\dot{r}_x$, and $\dot{r}_y$ are the linear velocities of the root, $\dot{\alpha}$ is the angular velocity of the Z angle of the body, θ are the SMPL pose parameters, and $\mathbf{j}$ are the joints positions (computed with the SMPL layer).

Code 2: Complete the missing part in `visualize_smpl.py` to print the dimensions F and d. Then, run the following command to visualize the output video in `smpl.mp4`:

```
python src/visualize_smpl.py
```

2 Model architectures

In this section, we implement three distinct architectures, as shown in Figure 2, inspired by the DiT paper [6]. The paper presents three different methods for conditioning a diffusion model: incontext, AdALN and cross attention. In the code, we implement these architectures using the base class `BaseDiT` located in `src/models/modules/base.py`. The main inference method is `forward`. In the following section, we will implement parts of `cond_mapping`, `backbone` and `output_projection` of each architecture.

2.1 Config A: Incontext

Incontext conditioning is the simplest approach for incorporating conditioning into a transformer-based diffusion model. It consists in appending the conditioning tokens to the main token stream and letting the self-attention blocks mixing them.

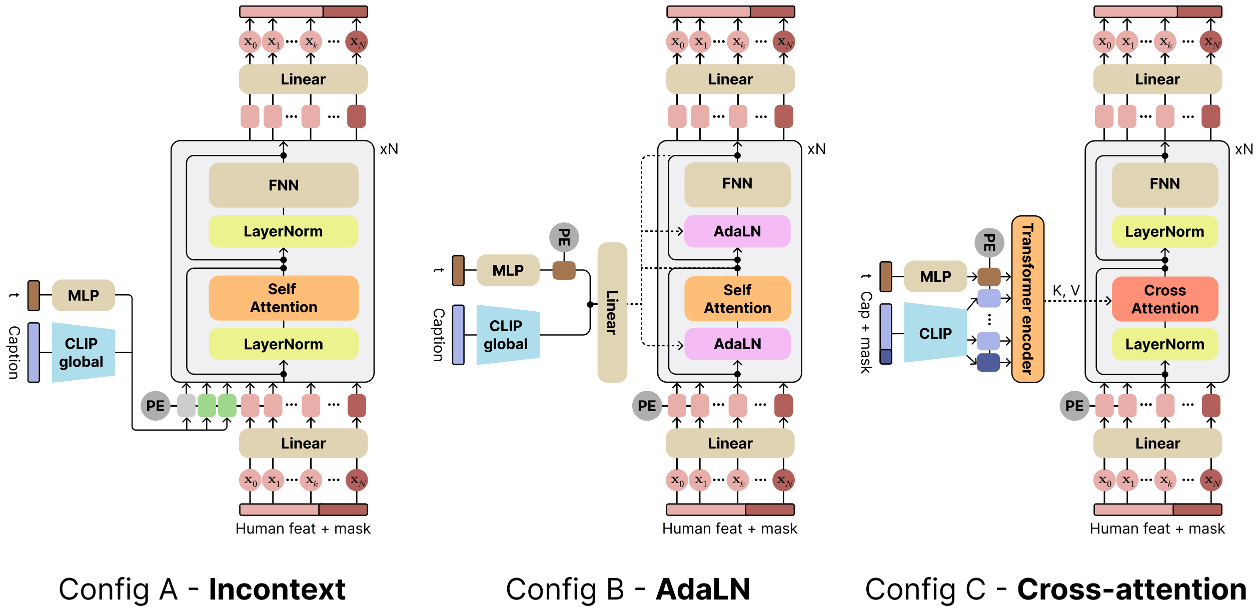

Figure 2: Human generation DiT-like architectures.

Code 3: As shown in Figure 2.A, fill in the missing parts in `src/models/modules/incontext.py`, then run the following command:

```
python src/models/modules/incontext.py
```

2.2 Config B: AdaLN

Adaptive Layer Normalization (adaLN) is a more sophisticated conditioning method. It replaces the standard layer normalization layers in transformer blocks with adaptive layer normalization (adaLN). The goal of adaLN is to regress a shift and a scale parameter from the conditioning tokens t and c.

Question 3: Given γ and β the shift and scale parameters, write the AdaLN operation:

$$adaLN(\mathbf{x}, \gamma, \beta) = \dots \tag{1}$$

Answer: $adaLN(\mathbf{x}, \gamma, \beta) = (1 + \gamma)LN(\mathbf{x}) + \beta.$

Code 4: As shown in Figure 2.B, fill in the missing parts in `src/models/modules/adaln.py`, then run the following command:

```
python src/models/modules/adaln.py
```

2.3 Config C: Cross attention

Question 4: Given $\mathbf{W}_Q$, $\mathbf{W}_K$, $\mathbf{W}_V$,as the query, key, and value weights, respectively, and d the token dimension, write the cross-attention operation between the input $\mathbf{x}$ and the condition $\mathbf{c}$:

$$\mathrm{CA}(\mathbf{x}, \mathbf{c}) = \dots \tag{2}$$

Answer:

$$\mathrm{CA}(\mathbf{x}, \mathbf{c}) = \mathrm{softmax}\left(\frac{(\mathbf{x}\mathbf{W}_Q).(\mathbf{c}\mathbf{W}_K)^\top}{\sqrt{d}}\right).(\mathbf{c}\mathbf{W}_V) \tag{3}$$

Code 5: As shown in Figure 2.C, fill in the missing parts in `src/models/modules/cross_attention.py`, then run the following command:

```
python src/models/modules/cross_attention.py
```

Question 5: What is the advantage of using AdaLN for conditioning, particularly in comparison to the cross-attention approach?
Answer: Among the three configs we explore, AdaLN introduces the fewest Gflops and is therefore the most compute-efficient, whereas the cross-attention approach requires significantly more computational resources.

3 Diffusion framework

In this section, we implement the diffusion loss and the sampling process. **Hint:** You can refer to this diffusion codebook (from page 15) to easily retrieve the relevant equations.

3.1 DDPM

Code 6: DDPM loss. Complete the missing part in `src/training/losses/ddpm.py` to compute the DDPM [4] loss. **Hint:** check Equation 14 of the original DDPM paper [4].

```
python src/training/losses/ddpm.py
```

Code 7: DDPM sampling. Complete the missing part in `src/training/sampler/ddpm.py` to perform the DDPM sampling. **Hint:** check Equation 84 of the diffusion codebook.

Run the following command to generate human motion using DDPM sampling, and visualize the output video in `generation_ddpm_incontext.mp4`: **Hint:** change the value of the random seed to get a different result.

```
python src/generate.py batch_size=1 seed=2 \
diffuser/sampler@diffuser.test_sampler=ddpm
```

3.2 DDIM

Question 6: What are the advantages of DDIM over DDPM sampling, and what is the key difference between them?
Answer: DDIM enables significantly faster sampling by skipping steps in the denoising process.

Code 8: DDIM sampling. Complete the missing part in `src/training/sampler/ddim.py` to perform the DDIM sampling. **Hint:** check Equation 12 of the original DDIM paper [9].

Run the following command to generate human motion using DDIM sampling, and visualize the output video in `generation_ddim_incontext.mp4`: **Hint:** change the value of the random seed to get a different result.

```
python src/generate.py batch_size=1 seed=2 \
diffuser/sampler@diffuser.test_sampler=ddim
```

Question 7: Are there any qualitative differences between the generated samples from the sampling methods? If yes, what are they?
Answer: In theory: incontext < cross attention < adaLN.

4 Result analysis

4.1 Qualitative analysis

Code 9: Run the following commands to generate a sample using each architecture:

```
python src/generate.py batch_size=1 diffuser/network=incontext \
checkpoint_path=./humanml3d-data/checkpoints/incontext.ckpt

python src/generate.py batch_size=1 diffuser/network=adaln \
checkpoint_path=./humanml3d-data/checkpoints/adaln.ckpt

python src/generate.py batch_size=1 diffuser/network=cross_attention \
checkpoint_path=./humanml3d-data/checkpoints/cross_attention.ckpt
```

Question 8: Are there any qualitative differences between the generated samples from the different architectures? If yes, what are they?
Answer: In theory: incontext < cross attention < adaLN.

4.2 Quantitative analysis

To quantitatively evaluate the generated motion, we adopt the same approach used in image generation: computing metrics between reference and generated features from an external model. Specifically, we use the TMR [8] feature encoder to calculate the Fréchet Distance (FD_{TMR}) which assesses overall generation quality (similar to FID [3]), and the TMR-Score, which evaluates text-motion coherence (similar to CLIP-Score [2]).

Question 9: What is the main assumption about the reference and generated feature spaces when computing the Fréchet Distance?
Answer: The reference and generated feature spaces are assumed to follow a Gaussian distribution.

Code 10: Complete the missing part in `src/metrics/frechet.py` to compute the FD_{TMR}.

Code 11: Complete the missing part in `src/metrics/similarity.py` to compute the TMR-Score.

Bonus 1: Review the code in `src/metrics/motion_text.py` and `src/metrics/prdc.py`. Then, explain the meaning of the following metrics: R1, R2, R3, and PRDC.
Answer: R1, R2, and R3 represent the motion text retrieval scores at ranks 1, 2, and 3, respectively. PRDC stands for precision, recall, density, and coverage. It is similar to traditional precision and recall but is computed in a feature space using k-nearest neighbors to determine the span of a data point.

Code 12: Run the following commands to evaluate each architecture:

```
python src/evaluate.py diffuser/network=incontext \
checkpoint_path=./humanml3d-data/checkpoints/incontext.ckpt

python src/evaluate.py diffuser/network=adaln \
checkpoint_path=./humanml3d-data/checkpoints/adaln.ckpt

python src/evaluate.py diffuser/network=cross_attention \
checkpoint_path=./humanml3d-data/checkpoints/cross_attention.ckpt
```

Question 10: Given the computed scores, what can you say?
Answer: In theory: incontext < cross attention < adaLN. However here the number of evaluation sample is too low to conclude anything.

Bonus 2: Here we have evaluated our models on 10×64 samples. How reliable do you think each metric is, particularly FD_{TMR}, and why? What should we do?
Answer: The number of evaluation samples is insufficient, especially for FD_{TMR} which requires a large sample size due to its statistical foundation. To improve reliability, we should evaluate FD_{TMR} on using the entire training dataset. For the other metrics, the test is enough.

References

[1] Chuan Guo, Shihao Zou, Xinxin Zuo, Sen Wang, Wei Ji, Xingyu Li, and Li Cheng. Generating diverse and natural 3d human motions from text. In *CVPR*, 2022.

[2] Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. Clipscore: A reference-free evaluation metric for image captioning. In *EMNLP*, 2021.

[3] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *NeurIPS*, 2017.

[4] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *NeurIPS*, 2020.

[5] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. SMPL: A skinned multi-person linear model. *ACM TOG*, 2015.

[6] William S Peebles and Saining Xie. Scalable diffusion models with transformers. In *ICCV*, 2022.

[7] Mathis Petrovich, Michael J Black, and Gül Varol. Temos: Generating diverse human motions from textual descriptions. In *ECCV*, 2022.

[8] Mathis Petrovich, Michael J Black, and Gül Varol. Tmr: Text-to-motion retrieval using contrastive 3d human motion synthesis. In *ICCV*, 2023.

[9] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *ICLR*, 2021.