Idempotent Unsupervised Representation Learning for Skeleton-Based Action Recognition (Supplementary Material)

Lilang Lin Lehong Wu Jiahang Zhang Jiaying Liu
Wangxuan Institute of Computer Technology, Peking University

Contents

1. Theoretical Proofs

1.1. Proof of Eq. (4)

Proof. First, the maximum entropy coding length is:

$$L = \mu \log\det\left(\mathbf{I} + \lambda \mathbf{Z}^T\mathbf{Z}\right), \tag{1}$$

where $\mu = \frac{m+d}{2}$ and $\lambda = \frac{d}{m\varepsilon^2}$. Next, we employ the following identical equation:

$$\begin{aligned}\log\det\left(\exp\left(\mathbf{A}\right)\right) &= \log\det\left(\sum_{n=0}^{\infty}\frac{\mathbf{A}^n}{n!}\right) = \log\det\left(\sum_{n=0}^{\infty}\frac{(\mathbf{U}\mathbf{D}\mathbf{U}^T)^n}{n!}\right)\\ &= \log\det\left(\mathbf{U}\sum_{n=0}^{\infty}\frac{\mathbf{D}^n}{n!}\mathbf{U}^T\right) = \log\prod_{i=1}^{m}\exp(\lambda_i) = \sum_{i=1}^{m}\lambda_i\\ &= \mathrm{Tr}\left(\mathbf{A}\right),\end{aligned} \tag{2}$$

where $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_m$ are the eigenvalues of $\mathbf{A}$. $\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{U}^T$ is the eigendecomposition. Then, we have:

$$\log\det\left(\mathbf{I} + \lambda\mathbf{Z}^T\mathbf{Z}\right) = \text{Tr}\left(\log\left(\mathbf{I} + \lambda\mathbf{Z}^T\mathbf{Z}\right)\right). \tag{3}$$

Finally, we apply a Taylor series expansion to the logarithm of the matrix to obtain:

$$L = \text{Tr}\left(\mu\sum_{n=1}^{\infty}\frac{(-1)^{n-1}}{n}\left(\lambda\mathbf{Z}^T\mathbf{Z}\right)^n\right). \tag{4}$$

□

1.2. Proof of Eq. (8)

Proof. The total loss of the idempotent generative model is expressed as:

$$\begin{aligned}
\mathcal{L} &= \mathcal{L}_{\text{ide}} - L = -2\sum_{\mathbf{x},\hat{\mathbf{x}}} p(\mathbf{x},\hat{\mathbf{x}}) f(\hat{\mathbf{x}}_i)^T f(\mathbf{x}_i) + \sum_{\mathbf{x},\mathbf{x}'} p(\mathbf{x})p(\mathbf{x}')\left(f(\mathbf{x})^T f(\mathbf{x}')\right)^2 + \mathbf{R} \\
&= -2\mathbb{E}_{(\mathbf{x},\hat{\mathbf{x}})\sim p(\mathbf{x},\hat{\mathbf{x}})}\left[f(\hat{\mathbf{x}})^T f(\mathbf{x})\right] + \mathbb{E}_{(\mathbf{x},\mathbf{x}')\sim p(\mathbf{x})p(\mathbf{x}')}\left[\left(f(\mathbf{x})^T f(\mathbf{x}')\right)^2\right] + \mathbf{R} \\
&= -2\text{Tr}\left(\mathbf{F}\mathbf{A}\mathbf{F}^T\right) + \text{Tr}\left(\left(\mathbf{F}^T\mathbf{F}\right)^2\right) + \mathbf{R} = 2\text{Tr}\left(\mathbf{F}\mathbf{L}\mathbf{F}^T\right) + \text{Tr}\left(\left(\mathbf{F}^T\mathbf{F}\right)^2\right) + \mathbf{R} + \mathbf{C} \\
&= \|\mathbf{A} - \mathbf{F}^T\mathbf{F}\|_F^2 + \mathbf{R} + \mathbf{C},
\end{aligned} \tag{5}$$

where $\mathbf{A} \in \mathbb{R}^{m\times m}$ is the adjacency matrix defined by the data generation and $\mathbf{C}$ is a constant. $\mathbf{F} = \mathbf{Z}\text{diag}(\sqrt{p(\mathbf{x})})$ The weights $\mathbf{A}_{\mathbf{x},\hat{\mathbf{x}}} = \frac{p(\mathbf{x},\hat{\mathbf{x}})}{\sqrt{p(\mathbf{x})p(\hat{\mathbf{x}})}}$. $\mathbf{L} = \mathbf{I} - \mathbf{A}$ is the Laplacian matrix. □

1.3. Proof of Eq. (11)

Proof. We compose the marginal distribution of $\mathbf{x}$ as a matrix $\mathbf{D}$. $\mathbf{D}_{\mathbf{x}} = p(\mathbf{x}) = d_{\mathbf{x}}$. We normalize the feature $\mathbf{z} = f(\mathbf{x})$ as $\mathbf{U}_{\mathbf{x}} = \sqrt{d_{\mathbf{x}}}f(\mathbf{x})$. $\mathbf{U} = \mathbf{Z}\mathbf{D}^{\frac{1}{2}} \in \mathbb{R}^{d\times m}$. We also normalize the adjacency matrix $\hat{\mathbf{A}}_{\mathbf{x},\hat{\mathbf{x}}} = \frac{p(\mathbf{x},\hat{\mathbf{x}})}{\sqrt{d_{\mathbf{x}}d_{\hat{\mathbf{x}}}}}$. Then we reformulate the downstream error,

$$\begin{aligned}
\mathbb{E}_{\mathbf{x},\mathbf{y}}\|\mathbf{y} - \mathbf{W}f(\mathbf{x})\|^2 &= \sum_{(\mathbf{x},\mathbf{y})} d_{\mathbf{x}}\|\mathbf{y} - \mathbf{W}f(\mathbf{x})\|^2 \\
&= \|\mathbf{Y}\mathbf{D}^{\frac{1}{2}} - \mathbf{W}\mathbf{U}\|^2 \\
&= \|\mathbf{Y}\mathbf{D}^{\frac{1}{2}} - \mathbf{C}\hat{\mathbf{A}}^T + \mathbf{C}\hat{\mathbf{A}}^T - \mathbf{W}\mathbf{U}\|^2,
\end{aligned} \tag{6}$$

where $\mathbf{C}_{j,\mathbf{x}} = \sqrt{d_{\mathbf{x}}1[\mathbf{y}(\mathbf{x}) = j]}$. Then we consider the relationship between the downstream error and the augmentation graph:

$$(\mathbf{Y}\mathbf{D}^{\frac{1}{2}})_{j,\mathbf{x}} = \sqrt{d_{\mathbf{x}}1[\mathbf{y}(\mathbf{x}) = j]}, \quad (\mathbf{C}\hat{\mathbf{A}}^T)_{j,\mathbf{x}} = \sum_{\hat{\mathbf{x}}}\frac{p(\mathbf{x},\hat{\mathbf{x}})}{\sqrt{d_{\mathbf{x}}d_{\hat{\mathbf{x}}}}}\sqrt{d_{\hat{\mathbf{x}}}1[\mathbf{y}(\hat{\mathbf{x}}) = j]}. \tag{7}$$

When $j = \mathbf{y}(\mathbf{x})$,

$$\begin{aligned}
(\mathbf{Y}\mathbf{D}^{\frac{1}{2}} - \mathbf{C}\hat{\mathbf{A}}^T)_{j,\mathbf{x}} &= \sqrt{d_{\mathbf{x}}1[\mathbf{y}(\mathbf{x}) = j]} - \sum_{\hat{\mathbf{x}}}\frac{p(\mathbf{x},\hat{\mathbf{x}})}{\sqrt{d_{\mathbf{x}}d_{\hat{\mathbf{x}}}}}\sqrt{d_{\hat{\mathbf{x}}}1[\mathbf{y}(\hat{\mathbf{x}}) = j]} \\
&= \sqrt{d_{\mathbf{x}}} - \sum_{\hat{\mathbf{x}}}\frac{p(\mathbf{x},\hat{\mathbf{x}})}{\sqrt{d_{\mathbf{x}}}}1[\mathbf{y}(\hat{\mathbf{x}}) = j] \\
&= \sum_{\hat{\mathbf{x}}}\frac{p(\mathbf{x},\hat{\mathbf{x}})}{\sqrt{d_{\mathbf{x}}}} - \sum_{\hat{\mathbf{x}}}\frac{p(\mathbf{x},\hat{\mathbf{x}})}{\sqrt{d_{\mathbf{x}}}}1[\mathbf{y}(\hat{\mathbf{x}}) = j] \\
&= \sum_{\hat{\mathbf{x}}}\frac{p(\mathbf{x},\hat{\mathbf{x}})}{\sqrt{d_{\mathbf{x}}}}1[\mathbf{y}(\hat{\mathbf{x}}) \neq j].
\end{aligned} \tag{8}$$

When $j \neq \mathbf{y}(\mathbf{x})$,

$$\begin{aligned}(\mathbf{Y}\mathbf{D}^{\frac{1}{2}} - \mathbf{C}\hat{\mathbf{A}}^T)_{j,\mathbf{x}} &= \sqrt{d_{\mathbf{x}} 1[\mathbf{y}(\mathbf{x}) = j]} - \sum_{\hat{\mathbf{x}}} \frac{p(\mathbf{x}, \hat{\mathbf{x}})}{\sqrt{d_{\mathbf{x}} d_{\hat{\mathbf{x}}}}} \sqrt{d_{\hat{\mathbf{x}}} 1[\mathbf{y}(\hat{\mathbf{x}}) = j]} \\ &= -\sum_{\hat{\mathbf{x}}} \frac{p(\mathbf{x}, \hat{\mathbf{x}})}{\sqrt{d_{\mathbf{x}}}} 1[\mathbf{y}(\hat{\mathbf{x}}) = j].\end{aligned} \tag{9}$$

Then, we define $\beta_{\mathbf{x}} = \sum_{\hat{\mathbf{x}}} p(\mathbf{x}, \hat{\mathbf{x}}) 1[\mathbf{y}(\hat{\mathbf{x}}) \neq \mathbf{y}(\mathbf{x})]$. We have:

$$\begin{aligned}\|(\mathbf{Y}\mathbf{D}^{\frac{1}{2}} - \mathbf{C}\hat{\mathbf{A}}^T)_{\mathbf{x}}\|^2 &= \left(\sum_{\hat{\mathbf{x}}} \frac{p(\mathbf{x}, \hat{\mathbf{x}})}{\sqrt{d_{\mathbf{x}}}} 1[\mathbf{y}(\hat{\mathbf{x}}) \neq \mathbf{x}]\right)^2 + \sum_{j \neq \mathbf{y}(\mathbf{x})} \left(\sum_{\hat{\mathbf{x}}} \frac{p(\mathbf{x}, \hat{\mathbf{x}})}{\sqrt{d_{\mathbf{x}}}} 1[\mathbf{y}(\hat{\mathbf{x}}) = j]\right)^2 \\ &\leq \left(\sum_{\hat{\mathbf{x}}} \frac{p(\mathbf{x}, \hat{\mathbf{x}})}{\sqrt{d_{\mathbf{x}}}} 1[\mathbf{y}(\hat{\mathbf{x}}) \neq \mathbf{x}]\right)^2 + \left(\sum_{\hat{\mathbf{x}}} \frac{p(\mathbf{x}, \hat{\mathbf{x}})}{\sqrt{d_{\mathbf{x}}}} \sum_{j \neq \mathbf{y}(\mathbf{x})} 1[\mathbf{y}(\hat{\mathbf{x}}) = j]\right)^2 \\ &= \left(\sum_{\hat{\mathbf{x}}} \frac{p(\mathbf{x}, \hat{\mathbf{x}})}{\sqrt{d_{\mathbf{x}}}} 1[\mathbf{y}(\hat{\mathbf{x}}) \neq \mathbf{x}]\right)^2 + \left(\sum_{\hat{\mathbf{x}}} \frac{p(\mathbf{x}, \hat{\mathbf{x}})}{\sqrt{d_{\mathbf{x}}}} 1[\mathbf{y}(\hat{\mathbf{x}}) \neq \mathbf{y}(\mathbf{x})]\right)^2 \\ &= \frac{2\beta_{\mathbf{x}}^2}{d_{\mathbf{x}}}.\end{aligned} \tag{10}$$

And we assume $\beta_{\mathbf{x}} = \sum_{\hat{\mathbf{x}}} p(\mathbf{x}, \hat{\mathbf{x}}) 1[\mathbf{y}(\hat{\mathbf{x}}) \neq \mathbf{y}(\mathbf{x})] \leq d_{\mathbf{x}} \alpha$.

$$\begin{aligned}\|(\mathbf{Y}\mathbf{D}^{\frac{1}{2}} - \mathbf{C}\hat{\mathbf{A}}^T)\|^2 &= \sum_{\mathbf{x}} \frac{2\beta_{\mathbf{x}}^2}{d_{\mathbf{x}}} \\ &= \sum_{\mathbf{x}} 2 d_{\mathbf{x}} \alpha^2 \\ &= 2\alpha^2 \leq 2\alpha.\end{aligned} \tag{11}$$

Then we obtain

$$\begin{aligned}\mathbb{E}_{\mathbf{x},\mathbf{y}} \|\mathbf{y} - \mathbf{W} f(\mathbf{x})\|^2 &= \|\mathbf{Y}\mathbf{D}^{\frac{1}{2}} - \mathbf{C}\hat{\mathbf{A}}^T + \mathbf{C}\hat{\mathbf{A}}^T - \mathbf{W}\mathbf{U}\|^2 \\ &\leq c_2 \alpha + c_1 \|\mathbf{C}\hat{\mathbf{A}}^T - \mathbf{W}\mathbf{U}\|^2 \\ &\leq c_1 \|\hat{\mathbf{A}}^T - \mathbf{U}^T\mathbf{U}\|^2 + c_2\alpha + \text{const} \\ &= c_1 \sum_{i=d+1}^{m} \lambda_i^2 + c_2 \alpha + \text{const}.\end{aligned} \tag{12}$$

□

2. Gradient Analysis

In this section we specifically analyse the effect of the gradient of maximum entropy coding and fusion modules on the feature space to specify the phenomenon of dimensionality collapse of the generative model.

2.1. Gradient Dynamics for Maximum Entropy Coding

First, the maximum entropy coding length is:

$$L = \mu \log\det\left(\mathbf{I} + \lambda \mathbf{Z}^T \mathbf{Z}\right), \tag{13}$$

where $\mu = \frac{m+d}{2}$ and $\lambda = \frac{d}{m\varepsilon^2}$. $\mathbf{Z} = \mathbf{V}\mathbf{W}\mathbf{U}^T$.

$$\begin{aligned}L &= \mu \log\det\left(\mathbf{I} + \lambda \mathbf{Z}^T \mathbf{Z}\right) \\ &= \mu \log\det\left(\mathbf{U}\left(\mathbf{I} + \lambda \mathbf{W}^{\mathbf{2}}\right)\mathbf{U}^T\right) \\ &= \mu \sum_{i=1}^{m} \log(1 + \lambda \omega_i^2).\end{aligned} \tag{14}$$

So for maximum entropy coding with constraints, the optimized dynamics of the derived eigenvalues are

$$\begin{aligned}\omega_i(t+1) &= \omega_i(t) + \eta\nabla_{\omega_i(t)}L - \kappa\nabla_{\omega_i(t)}\|\mathbf{W}\|_1 \\ &= \omega_i(t) + \frac{2\eta\mu\lambda\omega_i(t)}{1+\lambda\omega_i^2(t)} - \kappa.\end{aligned} \tag{15}$$

2.2. Equilibrium State and Complete Collapse

After reaching equilibrium, singular values converge to a fixed value:

$$\omega^* = \frac{\eta\mu}{\kappa} \pm \sqrt{\frac{\eta^2\mu^2}{\kappa^2} - \frac{1}{\lambda}}, \tag{16}$$

where we assume $\frac{\eta^2\mu^2}{\kappa^2} > \frac{1}{\lambda}$. Otherwise, if $\kappa^2 \geq \lambda\eta^2\mu^2$, a complete collapse occurs, at which point the singular values all converge to zero. We can infer from this observation that as the feature dimension increases and the dataset size grows, the likelihood of a complete collapse decreases.

2.3. Stability Analysis and Dimensional Collapse

We determine the gradient of the equilibrium point based on the gradient of the equilibrium point:

$$\dot{\omega} = f(\omega) = \frac{2\eta\mu\lambda\omega}{1+\lambda\omega^2} - \kappa, \tag{17}$$

$$f'(\omega) = \frac{2\eta\mu\lambda(1-\lambda\omega^2)}{(1+\lambda\omega^2)^2}, \tag{18}$$

$$\begin{aligned} f'(\frac{\eta\mu}{\kappa} + \sqrt{\frac{\eta^2\mu^2}{\kappa^2} - \frac{1}{\lambda}}) < 0, \\ f'(\frac{\eta\mu}{\kappa} - \sqrt{\frac{\eta^2\mu^2}{\kappa^2} - \frac{1}{\lambda}}) > 0. \end{aligned} \tag{19}$$

Therefore, larger equilibrium point represents stable fixed point, whereas smaller equilibrium point indicates unstable fixed point. Consequently, all singular values smaller than the smaller equilibrium converge to 0, resulting in dimensional collapse. Likewise, increased dimensions and larger datasets, coupled with reduced regularization constraints, can help alleviate dimensionality collapse.

3. Implementation Details

3.1. Preliminary: Diffusion Models

Diffusion models implement generative processes by reversing a pre-defined forward diffusion process, commonly expressed as a linear stochastic differential equation (SDE). Formally, the data trajectory $\{\mathbf{x}(t) \in \mathbb{R}^n\}_{t\in[0,1]}$ follows the forward SDE given by:

$$\mathrm{d}\mathbf{x} = \mu(t)\mathbf{x}\mathrm{d}t + \nu(t)\mathrm{d}\mathbf{w}. \tag{20}$$

Here, $\mu(t)\mathbf{x} \in \mathbb{R}^n$ and $\nu(t) \in \mathbb{R}$ represent the drift and diffusion coefficients, while $\mathbf{w}$ is a standard Wiener process.

The affine drift coefficients ensure the presence of analytically tractable Gaussian perturbation kernels. These are denoted by $p_{0,t}(\mathbf{x}_t \,|\, \mathbf{x}) = \mathcal{N}(\mathbf{x}_t; \alpha_t\mathbf{x}, \sigma_t^2\mathbf{I})$, where exact coefficients α_t and σ_t can be obtained using standard techniques. Through appropriately designed α_t and σ_t, this facilitates the transformation of the data distribution $\mathbf{x}_0 \sim p_{\text{data}}$ into a tractable isotropic Gaussian distribution $\mathbf{x}_1 \sim \mathcal{N}(0, \mathbf{I})$ via forward diffusion. Furthermore, $\mu(t)$ and $\nu(t)$ must be dependent on α_t and σ_t in the following form:

$$\mu(t) = \frac{\mathrm{d}\log\alpha_t}{\mathrm{d}t}, \quad g^2(t) = \frac{\mathrm{d}\sigma_t^2}{\mathrm{d}t} - 2\frac{\mathrm{d}\log\alpha_t}{\mathrm{d}t}\sigma_t^2. \tag{21}$$

To recover the data distribution p_{data} from the Gaussian distribution $\mathcal{N}(0, \mathbf{I})$, Anderson established a pivotal theorem stating that the forward process has an equivalent reverse-time diffusion process, as represented by the following equation, thereby equating the generating process to solving the diffusion SDE:

$$\mathrm{d}\mathbf{x}_t = \left[\mu(t)\mathbf{x}_t - \nu(t)^2\nabla_{\mathbf{x}_t}\log p_t(\mathbf{x}_t)\right]\mathrm{d}t + \nu(t)\mathrm{d}\bar{\mathbf{w}}, \tag{22}$$

where $\bar{w}_t$ represents the Wiener process in reverse time, and $\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)$ is the score function.

Moreover, Song *et al*. [6] also demonstrated the existence of a corresponding deterministic process whose trajectories possess the same marginal probability densities $p_t(\mathbf{x}_t)$, thereby establishing the basis for efficient sampling using numerical ODE solvers.

$$\mathrm{d}\mathbf{x}_t = \left[\mu(t)\mathbf{x}_t - \nu(t)^2 \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)\right] \mathrm{d}t. \tag{23}$$

Typically, we train a score network $s_\theta(\mathbf{x}_t, t)$, parameterized by θ, to approximate the score function $\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)$ by optimizing the denoising score matching loss:

$$\mathcal{L} = \mathbb{E}_t \left\{ \omega(t) \mathbb{E}_{\mathbf{x}_0, \mathbf{x}_t} [s_\theta(\mathbf{x}_t, t) - \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t|\mathbf{x}_0)]^2 \right\}, \tag{24}$$

In this equation, $\omega(t)$ represents a weighting function. In practical applications, there are two common methods for reparameterizing the score network. The first approach involves using a noise prediction network ϵ_θ, where $\epsilon_\theta(\mathbf{x}_t, t) = -\sigma_t s_\theta(\mathbf{x}_t, t)$. The second approach employs a data prediction network x_θ, where $x_\theta(\mathbf{x}_t, t) = (\sigma_t^2 s_\theta(\mathbf{x}_t, t) + \mathbf{x}_t)/\alpha_t$.

3.2. Network Architecture

Data Transformation: We describe the adopted transformations in detail. We select the following transformations as the basic transformation set to the input sequences of the encoder $f(\cdot)$, which have been widely adopted in previous works [3, 7]:

• *Shearing:* This transformation slants the human body 3D coordinates to a random angle by using a shear transformation matrix:

$$\mathbf{S} = \begin{bmatrix} 1 & a_{12} & a_{13} \\ a_{21} & 1 & a_{23} \\ a_{31} & a_{32} & 1 \end{bmatrix}, \tag{25}$$

where a_{ij} is the shear factor randomly sampled from [-1, 1].

• *Joint Jittering:* This transformation randomly selects j out of the 25 joints in the skeleton data. Subsequently, these selected joints are either masked to zero or perturbed by a uniformly distributed random matrix. By default, we set j to 15.

• *Gaussian Noise:* Gaussian noise $\mathcal{N}(0, 0.005)$ is added to the skeleton data.

• *Random Mask:* Mask refers to random masking in the time and space domain:

$$\tilde{\mathbf{x}} = \mathbf{x} \odot \mathbf{M}, \tag{26}$$

where $\mathbf{M}$ is a mask tenser. We adopt a more extensive masking ratio, inspired by the approach in MAMP [2], where we mask 90% of the data.

Patchify and Embedding: Given the input skeleton $\mathbf{x} \in \mathbb{R}^{c \times t \times v}$, we initially segment it into non-overlapping patches with equal lengths along the temporal dimension, resulting in $\mathbf{x}' \in \mathbb{R}^{(s \times c) \times t' \times v'}$, where t represents the number of frames, v denotes the number of joints, s signifies the patch length, and $t' = t/s$. These patches are then flattened and embedded into d dimensions via a trainable linear projection:

$$\mathbf{E} = \text{LinearProj}(\mathbf{x}') \in \mathbb{R}^{d \times (t' \times v')}, \tag{27}$$

yielding $l = t' \cdot v'$ tokens of dimension d as the input for the Transformer.

Transformer Architecture: Our encoder and decoder adhere to a vanilla Transformer architecture. Initially, trainable spatial and temporal positional embeddings, $\mathbf{P}_v \in \mathbb{R}^{d \times 1 \times v'}$ and $\mathbf{P}_t \in \mathbb{R}^{d \times t' \times 1}$, are incorporated into the tokens $\mathbf{E}$ through broadcasting.

$$\mathbf{z} = \mathbf{E} + \mathbf{P}_v + \mathbf{P}_t. \tag{28}$$

The Transformer network comprises alternating layers of multi-head self-attention (MSA) and multi-layer perceptron (MLP) with residual connections. Layer Norm (LN) is applied before each layer and after the last layer. The encoder output, $\mathbf{z} \in \mathbb{R}^{d \times l}$, represents the dense local representations. The global representation, $\bar{\mathbf{z}}$, is obtained by pooling all tokens. For the decoder output, it is linearly projected to yield the final prediction of the same shape as $\mathbf{x}$.

For enhancing network training, all skeleton sequences undergo temporal downsampling to 120 frames. The encoder $f(\cdot)$ and generator $g(\cdot)$ are built using the Transformer architecture [8], employing hidden channels configured to a dimension of 256. To assess performance, we employ a fully connected layer $\phi(\cdot)$.

To refine our network, we employ the Adam optimizer [4]. Training is executed on a single NVIDIA GeForce RTX 4090, employing a batch size of 128, and the network undergoes training for 400 epochs.

3.3. Training Strategy

Here we introduce the details of unsupervised, supervised learning, transfer learning, and zero-shot adaptation learning. Through comprehensive experiments, we can fully demonstrate the superiority of our method and obtain solid conclusions.

Self-Supervised Pretraining. We utilize diffusion-based generation to train the encoder $f(\cdot)$, generator $g(\cdot)$ and adapter $h(\cdot)$. We train the network for 400 epochs and the learning rate is set to 0.0001.
Unsupervised Learning. In the unsupervised setting, we assess the feature representation using a linear evaluation mechanism. This approach involves applying a linear classifier $\phi(\cdot)$ to the frozen pretrained weights of the encoder $f(\cdot)$. The classifier is trained to classify the features extracted from the encoder, and the action recognition accuracy serves as a metric to gauge the quality of the representation. We train the model for 100 epochs with a learning rate set to 0.1.
Supervised Learning. We utilize a K-nearest neighbor (KNN) classifier, a nonparametric supervised learning method, to directly assess the quality of the feature space learned by the encoder during the self-supervised pretraining stage. In this approach, we set K=1 to assign labels based on the cosine similarity distance, following a recent method called CMD [3].
Transfer Learning. To explore the generalization ability, we evaluate the performance of transfer learning. In transfer learning, we utilize the linear evaluation mechanism to evaluate the performance on the target dataset. To evaluate the transferability of learned features, we pretrain on NTU 60 dataset [5] and performs linear evaluation on the PKUMMD part II dataset [1]. We train the classifier $\phi(\cdot)$ for 100 epochs.
Zero-Shot Adaptation. In the zero-shot adaptation setting, first, we utilize the encoder to extract features from the noisy data. Then, these noise features serve as conditions for diffusion denoising. After several iterations of noise removal, we extract features from the denoised data to use as new conditions. This iterative process allows for the continuous removal of noise information from the conditions.

4. Visual Results

4.1. One-Step Denoising Visualisation

In the Figure 1, we show the results of one-step denoising results. First row of each group represents the ground truth data $\mathbf{x}$, the second row displays the noisy data $\mathbf{x}_t$ obtained after diffusion sampling, and the third row presents the predicted clean data $\mathbf{x}_0$.

4.2. Conditional Generation Visualisation

In Fig. 3, we present the results of conditional generation. By applying a series of data transformations to the conditions, we introduce noise, leading to the generation of skeleton data that exhibits some variation while preserving semantic information.

4.3. Mask Prediction Visualisation

4.4. Zero-Shot Generalization Visualisation

References

[1] Jiaying Liu, Sijie Song, Chunhui Liu, Yanghao Li, and Yueyu Hu. A benchmark dataset and comparison study for multi-modal human action analytics. *ACM Trans. Multimedia Comput. Commun. Appl.*, 16(2), 2020. 6

[2] Yunyao Mao, Jiajun Deng, Wengang Zhou, Yao Fang, Wanli Ouyang, and Houqiang Li. Masked motion predictors are strong 3d action representation learners. In *Proc. Int'l Conference on Computer Vision*, pages 10181–10191, 2023. 5

[3] Yunyao Mao, Wengang Zhou, Zhenbo Lu, Jiajun Deng, and Houqiang Li. CMD: Self-supervised 3d action representation learning with cross-modal mutual distillation. *Proc. European Conference on Computer Vision*, 2022. 5, 6

[4] Whitney K Newey. Adaptive estimation of regression models via moment restrictions. *Journal of Econometrics*, 1988. 5

[5] Amir Shahroudy, Jun Liu, Tian Tsong Ng, and Gang Wang. NTU RGB+D: A large scale dataset for 3d human activity analysis. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 6

[6] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020. 5

[7] Fida Mohammad Thoker, Hazel Doughty, and Cees GM Snoek. Skeleton-contrastive 3D action representation learning. In *Proc. ACM Int'l Conference on Multimedia*, 2021. 5

[8] Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *Proc. AAAI Conference on Artificial Intelligence*, 2018. 5

Figure 1. One-step denoising visualisation. Each group's first row represents the ground truth data $\mathbf{x}$, the second row displays the noisy data $\mathbf{x}_t$ obtained after diffusion sampling, and the third row presents the predicted clean data $\mathbf{x}_0$.

Figure 2. Conditional generation visualisation. Each group's first row represents the ground truth data $\mathbf{x}$, the second row presents the generated data $\mathbf{x}_0$.

Figure 3. Conditional generation visualisation. Each group's first row represents the ground truth data $\mathbf{x}$, the second row presents the generated data $\mathbf{x}_0$.