Token Boosting for Robust Self-Supervised Visual Transformer Pre-training (Supplementary Material)

Tianjiao Li[1†] Lin Geng Foo[1†] Ping Hu[2] Xindi Shang[3] Hossein Rahmani[4]
Zehuan Yuan[3] Jun Liu[1‡]
[1]Singapore University of Technology and Design
[2]Boston University [3]ByteDance [4]Lancaster University
{tianjiao_li,lingeng_foo}@mymail.sutd.edu.sg, pinghu@bu.edu, shangxindi@bytedance.com
h.rahmani@lancaster.ac.uk, yuanzehuan@bytedance.com, jun_liu@sutd.edu.sg

1. More Experiments

Visualization of corrupted samples and features. Here, in order to qualitatively show the improvements from using TBM, we visualize the image reconstruction quality during pre-training. In Fig. 1, we plot the original corrupted image and the reconstructed images of ViT-Huge with the decoder, with and without the use of TBM. When TBM is not used, reconstruction of the masked parts of the image is challenging, and the reconstruction looks blurred and inaccurate. This is because the corruptions in the unmasked parts of the image make it difficult to predict the masked parts. However, when TBM is used, there is a visible improvement in the quality of the reconstructed images, where the image looks sharper and less blurred, and much of the corruptions have been smoothened out.

Next, in Fig. 2 we visualize the effects of corruptions on the extracted features. On the left, we see the produced features when the input image is clean. When the same input is perturbed with a corruption and fed to the baseline ViT-Huge, the features undergo significant observable changes, showing that the features are not very robust to added corruptions. However, when TBM is applied, the features show minimal changes when fed with the same corrupted image, and look similar to features obtained under a clean setting. This shows that TBM helps to make the output features of VTs more robust against input corruptions.

Investigation on the impact of the number of layers using TBM modules. Firstly, we train a single model to deal with an individual type of corruption in a self-supervised manner, i.e., we train multiple models (with ViT-Huge as the encoder) to handle the various types of corruptions, and report the average performance of the models over all the corruption types in the top row of Fig. 3. We find that, after adding our TBM module to a single layer, we obtain a significantly improved accuracy. However, inserting TBM modules to more layers does not lead to further improvement, which suggests that inserting our TBM module to a single layer is sufficient in this case. Next, we train one model to handle all types of corruptions in ImageNet-C at the same time. The results are plotted in the bottom row of Fig. 3. As the number of layers using our TBM module increases, the performance increases by a large margin

Figure 1. Visualization of corrupted image and its reconstruction during pre-training. Here, we visualize a corrupted input image (left), the reconstruction from ViT-Huge (middle), and the reconstruction with token boosting from our ViT-Huge+TBM (right). Best viewed in color.

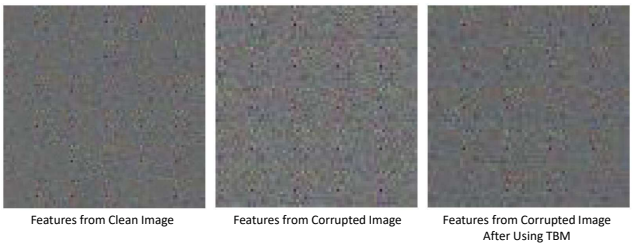

Figure 2. Visualization of features. We visualize the features obtained when a clean image is input to ViT-Huge (left), when a corrupted image is input to ViT-Huge (middle), and when the same corrupted image is input to ViT-Huge+TBM (right). Best viewed in color.

† equal contribution
‡ corresponding author

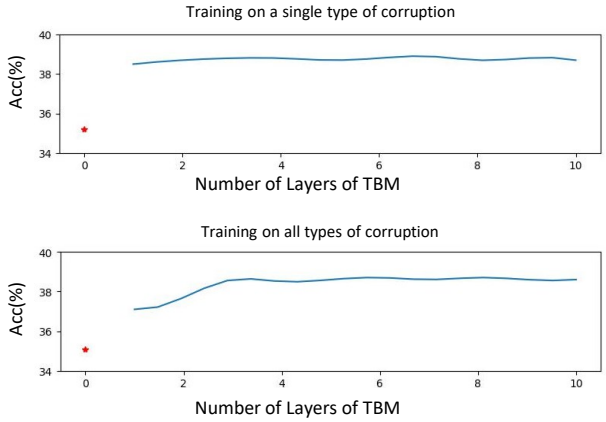

Figure 3. Evaluation of the impact of the number of layers using TBM modules. At the top, we plot the results where a single model is trained to deal with an individual type of corruption in ImageNet-C. At the bottom, we plot the results where the model is trained to handle all types of corruptions. The results of the baseline (i.e., ViT-Huge) without using any TBM modules are indicated in red stars ($\star$).

and then becomes stable after 3 layers. The results suggest that, our model, which boosts the features at multiple levels (3 levels in our experiments) with TBM, is capable of handling various types of corruptions simultaneously, which is a complicated task.

Performance after fine-tuning our pre-trained model. In the main paper, we evaluate on the self-supervised and supervised settings, and here we evaluate on a third setting, where we conduct fine-tuning on the entire model (i.e., both the ViT-Huge+TBM encoder and the linear layer). Specifically, we fine-tune the model for 40 epochs using the AdamW optimizer with a learning rate of 0.001. The results are reported in Tab. 1. We observe that we outperform the baseline on all three settings.

Table 1. Performance comparison (%) of ViT-Huge+TBM after fine-tuning on ImageNet-C.

Methods	Supervised	Self-supervised	Fine-tuned
ViT-Huge [3]	50.1	35.2	51.0
ViT-Huge+TBM	**53.2**	**38.6**	**54.1**

Comparison against alternative settings with similar model size. Here, we compare ViT-Huge+TBM against other settings with similar model size. In **ViT-Huge+TBM** ($\lambda = 0$), we train our ViT-Huge+TBM model as usual, except that the weight of the TBM's reconstruction loss λ is set to 0. This means that the TBM modules are still inserted into ViT-Huge, but there is no explicit boosting, and the TBM parameters are just a part of our model (which are involved in the end-to-end training). Alternatively, another way to add more parameters to approximately match the number of parameters with our ViT-Huge+TBM model is to have slightly more layers (**ViT-Huge + more layers**) or slightly wider layers (**ViT-Huge + wider layers**). As shown in Tab. 2, our method outperforms other methods with a similar model size.

Table 2. Performance comparison against alternative settings with similar model size on ImageNet-C. ViT-Huge+TBM outperforms other variants with the same number of parameters.

Settings	Accuracy (%)
ViT-Huge + more layers	36.0
ViT-Huge + wider layers	36.1
ViT-Huge+TBM ($\lambda = 0$)	35.7
ViT-Huge+TBM	38.6

Impact of the design of g. Next, we investigate different designs for the module g, and the results are reported in Tab. 3. Firstly, we ablate over the depth of g on all three tasks, and find that when we increase the depth of g, the performance improves, and then keeps stable. We find that using 3 layers for all tasks reaches optimal performance in the stable region. Next, we also replace the 3 simple fc-layers with 3 self-attention layers, and find that the results are similar. Thus, overall, we use 3 fc-layers in our design.

2. More Implementation Details

2.1. Training on Images

Detailed image pre-training procedure. For the training on corrupted RGB and depth images, we employ the random sampling strategy as outlined in [9]. Each image is split into 196 non-overlapping patches of approximately similar size. Then, patches are randomly selected to be masked until 75% of them have been masked, i.e., 147 patches are masked, and 49 patches remain visible. When patches are masked, the corresponding tokens are *removed* and are not fed as input to the VT encoder. Instead, masked tokens are introduced at the decoder for reconstruction purposes. Following [9], we only apply MSE reconstruction loss on the masked patches, and do not apply it on visible patches. During pre-training, following [9], our decoder consists of 8 stacked Transformer blocks with a width of 512. We also employ a data augmentation strategy of random resized cropping.

Detailed image linear probe hyperparameters. After adding the linear layer, the VT encoder is frozen and the final layer is trained using the cross entropy loss for 90 epochs with a batch size of 512. We use a LARS optimizer and set the learning rate to 0.1. We also augment our data using random resized cropping.

Table 3. Evaluation of various designs of g on three different tasks.

# fc layers in g	1	2	3	4	5	3 self-attention
Accuracy (%) on ImageNet-C	37.5	38.2	38.6	38.7	38.6	38.5
Accuracy (%) on NTU60	78.6	79.0	79.1	79.1	79.0	79.0
Accuracy (%) on WRGBD	71.5	71.4	71.5	71.4	71.5	71.4

Detailed image fully-supervised training hyperparameters. For the supervised setting on images, the full VT + linear layer is trained using the cross entropy loss for 400 epochs. ViT backbones are optimized using LARS optimizer with a learning rate of 0.1, while DEiT and Swin backbones are optimized using AdamW optimizer with a learning rate of 0.001. To augment our data, we use random resized cropping.

2.2. Training on Skeleton Sequences

Detailed skeleton sequence tokenization procedure. Following DSTA-Net [15], each input frame of the skeleton sequence is split into P patches, where the P presents the number of joints which is 25 for NTU RGB+D datasets. Each patch is converted to a token embedding using a mapper comprising of a Conv2D layer with width of 256, a BatchNorm layer and a LeakyReLU activation function. We follow [12] to process all the skeleton sequences.

Detailed skeleton sequence masking procedure. Here, we describe how we select the patches to mask during pre-training. We follow the Spatial Temporal Masking strategy in [14]. This strategy combines both Spatial Masking and Temporal Masking, where the model is forced to 1) learn to use relevant information from other (unmasked) joints, which improves the model's spatial awareness of the skeletal structure and 2) learn plausible motions of skeletal joints through an extended period of time, which improves the model's understanding of the temporal aspect of skeletal sequences. Specifically, following [14], we set the temporal masking ratio to 80%, and spatial masking to involve 2 joints per frame.

Detailed skeleton sequence pre-training procedure. When doing the masking procedure, we follow the same strategy in [14] where the masked joints are *removed* and not fed to the following encoding process. Masked tokens are added before the skeleton decoder to be reconstructed into the original skeleton sequence. For the skeleton decoder, we follow the basic structure of Kinetic-GAN [2] that leverages 7 stacked basic ST-GCN blocks as the decoder network. We train it for 200 epochs with batch size of 256.

Detailed skeleton sequence linear probe hyperparameters. After adding the linear layer, the DSTA-Net encoder is frozen and the final layer is trained using the cross entropy loss for 90 epochs with a batch size of 256. We use a SGD optimizer and set the learning rate to 0.1.

Detailed skeleton sequence fully-supervised training hyperparameters. For the supervised setting on skeleton sequences, the DSTA-Net+linear layer is end-to-end trained using the cross entropy loss for 90 epochs with a batch size of 256. Here, we adopt an SGD optimizer with a learning rate of 0.1.

2.3. More TBM Details

In order to set $\alpha > 0$ in practice while optimizing α using gradient descent, we apply the ReLU function onto our α parameters before using them to scale the noise. The ReLU function will have the effect of mapping elements of α into values ≥ 0 for scaling the noise. Moreover, we add a skip connection over the TBM module, which helps with convergence during training.

3. Future Work

In this work, we explore the scenario where training and testing corruption types are the same, which is commonly seen in many real-world tasks, such as skeleton action recognition and depth image recognition. As shown in the main paper, our TBM can handle these tasks well and achieve state-of-the-art results. In the future, we plan to explore the adaptation to unseen corruption types, i.e., Unsupervised Domain Adaptation (UDA) to new corruptions, which is an even more challenging scenario. One possible solution is to modify TBM to generate α based on the input, which allows TBM to adapt flexibly according to the input data, i.e., similar to a dynamic network [5, 10] that adapts its parameters according to the input data.

Other future work includes experimentation on other tasks which also involve noisy input data, such as pose estimation [4,7,13,18,20,21]. We can also investigate applying TBM to unified models [1,6,8,17] that tackle multiple tasks simultaneously.

4. Analysis of $E[Q|I] = E[P|I]$

In Section 3.2 of our paper, we mention that $\mathbb{E}[Q|I] = \mathbb{E}[P|I]$, given our assumptions that P and Q are drawn from the same distribution. Here we give theoretical justifications of this.

$$\mathbb{E}[Q|I] = \mathbb{E}[Q|R+P+Q=I] \tag{1}$$
$$= \int_q q \cdot \frac{\text{Prob}(Q=q, R+P+Q=I)}{\text{Prob}(R+P+Q=I)} \tag{2}$$
$$= \int_q q \cdot \frac{\text{Prob}(Q=q, P=I-R-q)}{\text{Prob}(R+P+Q=I)} \tag{3}$$
$$= \frac{1}{d}\int_q q \cdot \text{Prob}(Q=q, P=I-R-q) \tag{4}$$

where $d = \text{Prob}(R+P+Q=I)$ is a constant.

On the other hand, we also get:

$$\mathbb{E}[P|I] = \mathbb{E}[P|R+P+Q=I] \tag{5}$$
$$= \int_p p \cdot \frac{\text{Prob}(P=p, R+P+Q=I)}{\text{Prob}(R+P+Q=I)} \tag{6}$$
$$= \int_p p \cdot \frac{\text{Prob}(P=p, Q=I-R-p)}{\text{Prob}(R+P+Q=I)} \tag{7}$$
$$= \frac{1}{d}\int_p p \cdot \text{Prob}(P=p, Q=I-R-p) \tag{8}$$

When P and Q have the same distribution, and thus the same support, we can conclude that Eq. 4 = Eq. 8, and $\mathbb{E}[Q|I] = \mathbb{E}[P|I]$.

Note that above, we have the assumption that P and Q have the same distribution. Thus, in the next section, we analyse theoretically why our method can learn to enable the synthetic corruptions Q to have the same distribution as P.

5. Analysis that distribution of synthetic corruptions Q approximate distribution of natural corruptions P

In the section above, we assume that Q and P come from the same distribution. Here, we show why we can expect this in practice, and show theoretical analysis to support this. In short, it is because α can be meaningfully trained to be similar to the underlying corruption distribution. Specifically, the achieved loss will be higher, if α does not learn to model the underlying corruption distribution well.

Following [11, 16, 19], the "natural" corruption distribution $\mathcal{P}$ is modeled as Gaussian with a mean of 0 and some unknown standard deviations ω. In our TBM, the synthetic corruption Q is drawn from Gaussian with standard deviations α. We next show that if $\alpha \neq \omega$, the produced boosted features $\hat{R}$ will be biased, and so there will be a higher loss incurred.

To show that $\mathbb{E}[Q|I] = \frac{\alpha^2}{\omega^2}\mathbb{E}[P|I]$, we first start by analyzing the conditional pdf of P.

$$\text{Prob}(P=p|I=i, R=r) = \frac{\text{Prob}(P=p, I=i, R=r)}{\text{Prob}(I=i, R=r)} \tag{9}$$
$$= \frac{\text{Prob}(P=p)\text{Prob}(Q=i-r-p)}{\text{Prob}(I=i, R=r)} \tag{10}$$
$$= \frac{1}{b_1}\text{Prob}(P=p)\text{Prob}(Q=i-r-p) \tag{11}$$
$$= \frac{1}{b_2} exp\left\{-\frac{p^2}{2\omega^2}\right\} exp\left\{-\frac{(i-r-p)^2}{2\alpha^2}\right\} \tag{12}$$
$$= \frac{1}{b_2} exp\left\{-\frac{\alpha^2p^2 + \omega^2(i-r-p)^2}{2\alpha^2\omega^2}\right\} \tag{13}$$
$$= \frac{1}{b_2} exp\left\{-\frac{(\alpha^2+\omega^2)p^2 - 2\omega^2p(i-r) + \omega^2(i-r)^2}{2\alpha^2\omega^2}\right\} \tag{14}$$
$$= \frac{1}{b_2} exp\left\{-\frac{p^2 - \frac{2\omega^2}{(\alpha^2+\omega^2)}p(i-r) + \frac{\omega^2}{(\alpha^2+\omega^2)}(i-r)^2}{2\frac{\alpha^2\omega^2}{(\alpha^2+\omega^2)}}\right\} \tag{15}$$
$$= \frac{1}{b_2} exp\left\{-\frac{p^2 - \frac{2\omega^2}{(\alpha^2+\omega^2)}p(i-r) + \left(\frac{\omega^2}{(\alpha^2+\omega^2)}\right)^2(i-r)^2}{2\frac{\alpha^2\omega^2}{(\alpha^2+\omega^2)}}\right\}$$
$$\cdot exp\left\{-\frac{\frac{\omega^2}{(\alpha^2+\omega^2)}(i-r)^2 - \left(\frac{\omega^2}{(\alpha^2+\omega^2)}\right)^2(i-r)^2}{2\frac{\alpha^2\omega^2}{(\alpha^2+\omega^2)}}\right\} \tag{16}$$
$$= \frac{1}{b_3} exp\left\{-\frac{\left(\frac{\omega^2}{\alpha^2+\omega^2}(i-r) - p\right)^2}{2\frac{\alpha^2\omega^2}{(\alpha^2+\omega^2)}}\right\} \tag{17}$$

where b_1, b_2, b_3 are constants. To get Eq. 10 by splitting the joint distribution into marginals, we use the fact that $I = R+P+Q$, and P and Q are independent of each other. We note that the conditional distribution in Eq. 17 has a mean of $\frac{\omega^2}{\alpha^2+\omega^2}(i-r)$. In other words, $\mathbb{E}[P|I=i, R=r] = \frac{\omega^2}{\alpha^2+\omega^2}(i-r)$. Doing the same for the conditional distribution of Q gives us $\mathbb{E}[Q|I=i, R=r] = \frac{\alpha^2}{\alpha^2+\omega^2}(i-r)$. Equating both of them gives us the resulting relationship: $\mathbb{E}[Q|I=i, R=r] = \frac{\alpha^2}{\omega^2}\mathbb{E}[P|I=i, R=r]$. As this result holds independently of R, we get $\mathbb{E}[Q|I] = \frac{\alpha^2}{\omega^2}\mathbb{E}[P|I]$.

In our TBM module, we will use Eq. 1 of the main paper, i.e., $\hat{R} = 2\hat{F} - I$ to reconstruct $\hat{R}$. Thus, our estimate becomes:

$$\hat{R} = 2\mathbb{E}[R+P|I] - (\mathbb{E}[R+P+Q|I] \tag{18}$$
$$= 2\mathbb{E}[R|I] + 2\mathbb{E}[P|I] - (\mathbb{E}[R|I] + \mathbb{E}[P|I] + \mathbb{E}[Q|I]) \tag{19}$$
$$= \mathbb{E}[R|I] + \mathbb{E}[P|I] - \mathbb{E}[Q|I] \tag{20}$$
$$= \mathbb{E}[R|I] + \mathbb{E}[P|I] - \frac{\alpha^2}{\omega^2}\mathbb{E}[P|I] \tag{21}$$
$$= \mathbb{E}[R|I] + (1 - \frac{\alpha^2}{\omega^2})\mathbb{E}[P|I] \tag{22}$$

If $\alpha \neq \omega$, this obtained term in Eq. 22 is not equal to $\mathbb{E}[R|I]$, which is not desirable as it means that our boosted features will be biased, i.e., $\hat{R} \neq \mathbb{E}[R|I]$. For example, when $\alpha << \omega$ and $\frac{\alpha^2}{\omega^2}$ is small and close to 0, we get an estimate $\hat{R} \approx \mathbb{E}[R|I] + \mathbb{E}[P|I] = F$, which means that almost no boosting is done. On the other hand, when $\alpha >> \omega$, our TBM module will over-compensate for the corruptions, and we get a case where the corruption changes signs (from $\mathbb{E}[P|I]$ to $(1 - \frac{\alpha^2}{\omega^2})\mathbb{E}[P|I]$) and still affect the performance of the task.

Failure to boost the tokens in the TBM module will lead to a higher loss in the end-to-end objective, as analyzed in Sec. 6 of the Supplementary. Thus, to minimize this loss, the gradients will optimize α to become an approximation of ω, i.e., $\alpha \approx \omega$. Thus, $\frac{\alpha^2}{\omega^2} \approx 1$ and Eq. 22 becomes $\mathbb{E}[R|I] + (1 - \frac{\alpha^2}{\omega^2})\mathbb{E}[P|I] \approx \mathbb{E}[R|I]$ as we intend, in order to minimize the loss. This means that α will be trained to be similar to ω, the parameters of the "natural" corruption distribution.

6. Detailed Analysis of Section 3.4 of Main Paper

The detailed derivation of Eq. 5 in the main paper is as follows:

$$\mathbb{E}\Big[\frac{1}{N_V}\sum_{j=1}^{N_V}[V_j - (\beta U + c)_j]^2\Big] \tag{23}$$
$$= \mathbb{E}\Big[\frac{1}{N_V}\sum_{j=1}^{N_V}[(\beta U + c + \epsilon)_j - (\beta U + c)_j]^2\Big] \tag{24}$$
$$= \mathbb{E}\big[\frac{1}{N_V}\sum_{j=1}^{N_V}\epsilon_j^2\big] \tag{25}$$
$$= \frac{1}{N_V}\sum_{j=1}^{N_V}\mathbb{E}[\epsilon_j^2] \tag{26}$$
$$= \frac{1}{N_V}\sum_{j=1}^{N_V}Var(\epsilon_j) + \mathbb{E}[\epsilon_j]^2 \tag{27}$$
$$= \gamma^2 \tag{28}$$

Next, the detailed steps to get Eq. 6-8 of the main paper is as follows:

$$\mathbb{E}\Big[\frac{1}{N_V}\sum_{j=1}^{N_V}[\tilde{V}_j - (\beta\tilde{U} + c)_j]^2\Big] \tag{29}$$
$$= \mathbb{E}\Big[\frac{1}{N_V}\sum_{j=1}^{N_V}[(V + S_V)_j - (\beta(U + S_U) + c)_j]^2\Big] \tag{30}$$
$$= \mathbb{E}\Big[\frac{1}{N_V}\sum_{j=1}^{N_V}[\epsilon_j + (S_V)_j - (\beta S_U)_j]^2\Big] \tag{31}$$
$$= \frac{1}{N_V}\sum_{j=1}^{N_V}\mathbb{E}\Big[[\epsilon_j]^2 + [(S_V)_j]^2 + [(\beta S_U)_j]^2 - 2[(S_V)_j][(\beta S_U)_j] - 2[\epsilon_j][(\beta S_U)_j] + 2[\epsilon_j][(S_V)_j]\Big] \tag{32}$$
$$= \frac{1}{N_V}\sum_{j=1}^{N_V}\mathbb{E}[\epsilon_j^2] + \mathbb{E}[(S_V)_j^2] + \mathbb{E}[(\beta S_U)_j^2] - 2\mathbb{E}[(S_V)_j \cdot (\beta S_U)_j] - 2\mathbb{E}[\epsilon_j \cdot (\beta S_U)_j] + 2\mathbb{E}[\epsilon_j \cdot (S_V)_j] \tag{33}$$
$$= \frac{1}{N_V}\sum_{j=1}^{N_V}\mathbb{E}[\epsilon_j^2] + \mathbb{E}[(S_V)_j^2] + \mathbb{E}[(\beta S_U)_j^2] - 2\mathbb{E}[\epsilon_j]\mathbb{E}[(\beta S_U)_j] - 2\mathbb{E}[(S_V)_j]\mathbb{E}[(\beta S_U)_j] + 2\mathbb{E}[\epsilon_j]\mathbb{E}[(S_V)_j] \text{ (By independence)} \tag{34}$$
$$= \frac{1}{N_V}\sum_{j=1}^{N_V}\mathbb{E}[\epsilon_j^2] + \mathbb{E}[(S_V)_j^2] + \mathbb{E}[(\beta S_U)_j^2] \quad (\text{as } \mathbb{E}[\epsilon_j] = \mathbb{E}[(S_V)_j] = 0) \tag{35}$$
$$= \frac{1}{N_V}\sum_{j=1}^{N_V}(Var(\epsilon_j) + \mathbb{E}[\epsilon_j]^2) + (Var((S_V)_j) + \mathbb{E}[(S_V)_j]^2) + (Var((\beta S_U)_j) + \mathbb{E}[(\beta S_U)_j]^2) \tag{36}$$
$$= \frac{1}{N_V}\sum_{j=1}^{N_V}Var(\epsilon_j) + Var((S_V)_j) + Var((\beta S_U)_j) \text{ (By independence)} \tag{37}$$
$$= \frac{1}{N_V}\sum_{j=1}^{N_V}\Big(\gamma^2 + \sigma_N^2 + \sum_{k=1}^{N_U}Var(\beta_{jk}(S_U)_k)\Big) \tag{38}$$
$$= \frac{1}{N_V}\sum_{j=1}^{N_V}\gamma^2 + \sigma_N^2 + \sum_{k=1}^{N_U}\beta_{jk}^2\sigma_N^2 \tag{39}$$
$$= \gamma^2 + \sigma_N^2 + \frac{1}{N_V}\sum_{j=1}^{N_V}\sum_{k=1}^{N_U}\beta_{jk}^2\sigma_N^2 \tag{40}$$

Lastly, the detailed steps corresponding to Eq. 9-11 of the main paper are as follows:

$$\mathbb{E}\left[\frac{1}{N_V}\sum_{j=1}^{N_V}[\tilde{V}_j - (\beta U + c)_j]^2\right] \tag{41}$$

$$= \mathbb{E}\left[\frac{1}{N_V}\sum_{j=1}^{N_V}[(V + S_V)_j - (\beta U + c)_j]^2\right] \tag{42}$$

$$= \mathbb{E}\left[\frac{1}{N_V}\sum_{j=1}^{N_V}[\epsilon_j + (S_V)_j]^2\right] \tag{43}$$

$$= \mathbb{E}\left[\frac{1}{N_V}\sum_{j=1}^{N_V}[\epsilon_j]^2 + [(S_V)_j]^2 + 2[\epsilon_j][(S_V)_j]\right] \tag{44}$$

$$= \frac{1}{N_V}\sum_{j=1}^{N_V}\mathbb{E}[\epsilon_j^2] + \mathbb{E}[(S_V)_j^2] + 2\mathbb{E}[\epsilon_j(S_V)_j] \tag{45}$$

$$= \frac{1}{N_V}\sum_{j=1}^{N_V}\mathbb{E}[\epsilon_j^2] + \mathbb{E}[(S_V)_j^2] + 2\mathbb{E}[\epsilon_j]\mathbb{E}[(S_V)_j] \tag{46}$$

(By independence)

$$= \frac{1}{N_V}\sum_{j=1}^{N_V}\mathbb{E}[\epsilon_j^2] + \mathbb{E}[(S_V)_j^2] \text{ (as } \mathbb{E}[\epsilon_j] = \mathbb{E}[(S_V)_j] = 0) \tag{47}$$

$$= \frac{1}{N_V}\sum_{j=1}^{N_V}(Var(\epsilon_j) + \mathbb{E}[\epsilon_j]^2) + (Var((S_V)_j) + \mathbb{E}[(S_V)_j]^2) \tag{48}$$

$$= \frac{1}{N_V}\sum_{j=1}^{N_V}Var(\epsilon_j) + Var((S_V)_j) \tag{49}$$

$$= \gamma^2 + \sigma_N^2 \tag{50}$$

References

[1] Zhiyang Chen, Yousong Zhu, Zhaowen Li, Fan Yang, Wei Li, Haixin Wang, Chaoyang Zhao, Liwei Wu, Rui Zhao, Jinqiao Wang, et al. Obj2seq: Formatting objects as sequences with class prompt for visual tasks. *arXiv preprint arXiv:2209.13948*, 2022. 3

[2] Bruno Degardin, João Neves, Vasco Lopes, João Brito, Ehsan Yaghoubi, and Hugo Proença. Generative adversarial graph convolutional networks for human action synthesis. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1150–1159, 2022. 3

[3] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. 2

[4] Lin Geng Foo, Jia Gong, Zhipeng Fan, and Jun Liu. System-status-aware adaptive network for online streaming video understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2023. 3

[5] Lin Geng Foo, Tianjiao Li, Hossein Rahmani, Qiuhong Ke, and Jun Liu. Era: Expert retrieval and assembly for early action prediction. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXIV*, pages 670–688. Springer, 2022. 3

[6] Lin Geng Foo, Tianjiao Li, Hossein Rahmani, Qiuhong Ke, and Jun Liu. Unified pose sequence modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2023. 3

[7] Jia Gong, Lin Geng Foo, Zhipeng Fan, Qiuhong Ke, Hossein Rahmani, and Jun Liu. Diffpose: Toward more reliable 3d pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2023. 3

[8] Tanmay Gupta, Amita Kamath, Aniruddha Kembhavi, and Derek Hoiem. Towards general purpose vision systems: An end-to-end task-agnostic vision-language architecture. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16399–16409, 2022. 3

[9] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. *arXiv preprint arXiv:2111.06377*, 2021. 2

[10] Tianjiao Li, Lin Geng Foo, Qiuhong Ke, Hossein Rahmani, Anran Wang, Jinghua Wang, and Jun Liu. Dynamic spatio-temporal specialization learning for fine-grained action recognition. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part IV*, pages 386–403. Springer, 2022. 3

[11] Xuelong Li, Zhenghang Yuan, and Qi Wang. Unsupervised deep noise modeling for hyperspectral image change detection. *Remote Sensing*, 11(3):258, 2019. 4

[12] Ziyu Liu, Hongwen Zhang, Zhenghao Chen, Zhiyong Wang, and Wanli Ouyang. Disentangling and unifying graph convolutions for skeleton-based action recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 143–152, 2020. 3

[13] Xun Long Ng, Kian Eng Ong, Qichen Zheng, Yun Ni, Si Yong Yeo, and Jun Liu. Animal kingdom: A large and diverse dataset for animal behavior understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19023–19034, 2022. 3

[14] Wenkang Shan, Zhenhua Liu, Xinfeng Zhang, Shanshe Wang, Siwei Ma, and Wen Gao. P-stmo: Pre-trained spatial temporal many-to-one model for 3d human pose estimation. In *ECCV*, page 461–478, 2022. 3

[15] Lei Shi, Yifan Zhang, Jian Cheng, and Hanqing Lu. Decoupled spatial-temporal attention network for skeleton-based action-gesture recognition. In *Proceedings of the Asian Conference on Computer Vision*, 2020. 3

[16] Yichun Shi and Anil K Jain. Probabilistic face embeddings. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6902–6911, 2019. 4

[17] Peng Wang, An Yang, Rui Men, Junyang Lin, Shuai Bai, Zhikang Li, Jianxin Ma, Chang Zhou, Jingren Zhou, and Hongxia Yang. Ofa: Unifying architectures, tasks, and modalities through a simple sequence-to-sequence learning framework. In *International Conference on Machine Learning*, pages 23318–23340. PMLR, 2022. 3

[18] Tianhan Xu and Wataru Takano. Graph stacked hourglass networks for 3d human pose estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16105–16114, 2021. 3

[19] Tianyuan Yu, Da Li, Yongxin Yang, Timothy M Hospedales, and Tao Xiang. Robust person re-identification by modelling feature uncertainty. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 552–561, 2019. 4

[20] Long Zhao, Xi Peng, Yu Tian, Mubbasir Kapadia, and Dimitris N Metaxas. Semantic graph convolutional networks for 3d human pose regression. In *IEEE CVPR*, pages 3425–3435, 2019. 3

[21] Weixi Zhao, Weiqiang Wang, and Yunjie Tian. Graformer: Graph-oriented transformer for 3d pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20438–20447, 2022. 3