SUPPLEMENTARY MATERIALS

A. Da-TACOS TRAINING SET

As mentioned in Section 3.2, we publicly share the Da-TACOS training set, which includes pre-computed crema-PCP features and the related metadata for a total of 97,904 songs in 17,999 unique works or cliques. The training and validation partitions include randomly-selected, disjoint sets of 14,499 and 3,500 cliques, respectively. The average duration of the songs is 218 s, with a standard deviation of 69 s. All audio files used for computing the features are encoded in MP3 format with varying bit rates, and their sample rate is 44.1 kHz. The metadata and the annotations are obtained using the API of `secondhandsongs.com`, and they are shared under the Creative Commons BY-NC 3.0 license. For each song, the shared metadata includes the song title and artist, the work title and artist, release year, SecondHandSongs.com performance and work IDs, and whether the song is instrumental or not.

Together with SHS-100K [23], this is one of the largest datasets available for training VI systems. An important advantage of the current dataset is that it contains a completely disjoint set of cliques with respect to the publicly-available evaluation set Da-TACOS [9]. Therefore, researchers can use it as their new training and validation datasets and still report their test results using Da-TACOS benchmark subset.

Figure S1 shows the distributions of songs per clique for the training (83.9 k songs, left) and the validation partitions (14.0 k songs, right). The number of songs per clique ranges from 2 to 109 in the training set, and from 2 to 11 in the validation set. Our intention was to mimic real-world data where it is more likely to have more unique works with less number of songs, rather than having a balanced dataset in terms of the number of songs per clique.

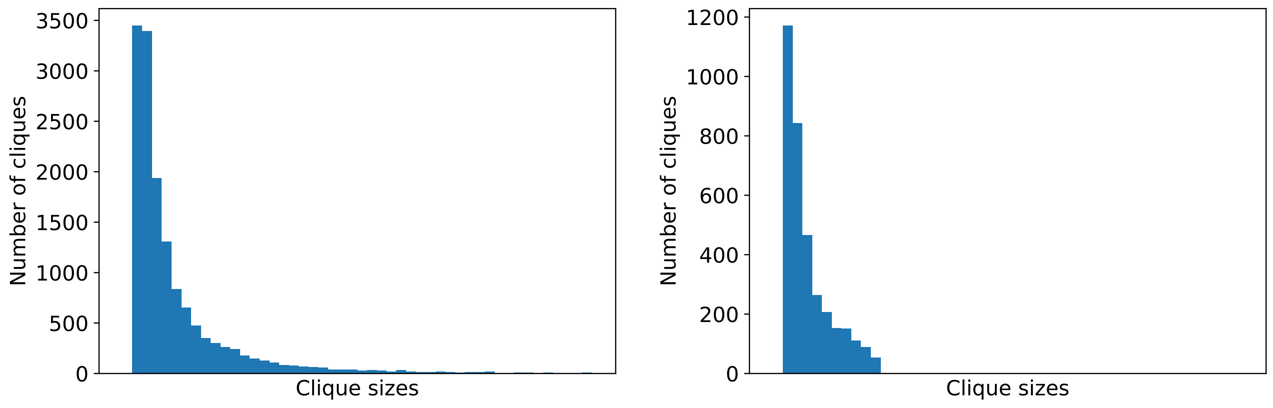

Figure S1: Distribution of the number of songs per clique for the training (left) and the validation (right) sets.

B. GENERALIZABILITY OF LATENT SPACE RECONFIGURATION

The experimental results reported in the main paper show that the latent space reconfiguration technique takes advantage of using a pre-trained feature extractor to reach high accuracies with fewer dimensions compared with using a randomly-initialized model. We claim that due to its model-agnostic nature, it can be used with any neural network-based VI system to demonstrate a similar performance gain.

To test our claim, we implement the encoder proposed by Doras and Peeters [10], and investigate whether we can obtain similar results compared to the ones in Section 4.2. For this, we extracted the dominant melody representations used in [10] for both our training data and Da-TACOS. To start with, we need a reliable base model (as MOVE-16k) for using its feature extractor. We train the network proposed in [10] with $d = 512$ and call it F0-512 for readability. To show the effect of using latent space reconfiguration, we later train the same network using $d = \{64, 128\}$ from scratch. Lastly, we use the feature extractor of F0-512, remove the linear layer and learn a new latent space with randomly-initalized linear layer using $d = \{64, 128\}$. Compared to the experiments in the main paper, F0-512 is the counterpart of MOVE-16k, and the networks with $d = \{64, 128\}$ correspond to the newly-trained MOVE models with $d < 2048$. For this set of experiments, we consider only the triplet loss in order to isolate the effect of using latent space reconfiguration from other factors such as the performance of different loss functions.

Instead of replicating their entire training setting, we use the training strategies explained for our other experiments, with a few exceptions: (1) we use Adam optimizer instead of SGD or Ranger, (2) the mini-batch size is 100, (3) we do not apply data augmentation to the samples, (4) the latent space reconfiguration experiments use an initial learning rate 0.1 while the others use 0.0001 (using 0.1 for training from scratch resulted in collapsed models).

Table S1 presents our findings. Firstly, the baseline results (top) appear counter-intuitive as using lower embedding sizes results in higher performances. We believe that this may be related to not optimizing the training strategies and hyper-

Method	d	
	64	128
Baselines (training from scratch)		
Triplet	0.283	0.279
Latent space reconfiguration		
Triplet	0.298	0.295

Table S1: MAP for different embedding sizes d when training from scratch (top) and when using latent space reconfiguration (middle-bottom). MAP for F0-512 is 0.277.

parameters for this specific architecture using this specific input representation. Secondly, the latent space reconfiguration results (bottom) seem consistently superior compared with the baseline results. The relative increases with respect to their baseline counterparts are 5.3% and 5.7% for $d = 64$ and $d = 128$, respectively. These results suggest that the performance increase achieved by using latent space reconfiguration is not limited to the network architecture or the input representation used for the experiments reported in the main paper. However, a more comprehensive analysis using several other network architectures and datasets is still needed to conclude that latent space reconfiguration technique can be applied to any embedding-based VI system.