# A. Appendix

## A.1. Foundations of compositional contrastive learning

In this section, we develop more formally a basic theory of compositional contrastive learning formulation, providing rigorous grounds for the approach described in Sec. 3.

Consider the problem of learning a function $f : \mathcal{X} \to \mathcal{Y}$. In a contrastive setting, we are not given information about the values of $f$; instead, we are given a *contrast function* $c : \mathcal{X} \times \mathcal{X} \to \{0, 1\}$ which only tells for which pairs of points $x_1$ and $x_2$ $f$ is the same and for which it differs:

**Definition 1.** *The function $f$ is* compatible *with the contrast $c$ if, and only if, for all $x_1, x_2 \in \mathcal{X}$:*

$$c(x_1, x_2) = \delta_{f(x_1)=f(x_2)}.$$

A contrast function cannot be arbitrary:

**Lemma 1.** *The predicate $c(x_1, x_2) = 1$ is an equivalence relation if, and only if, there exists a function $f$ compatible with $c$.*

*Proof.* If $c(x_1, x_2) = 1$ defines an equivalence relation on $\mathcal{X}$, then such a function is given by the projection on the quotient $\hat{f} : \mathcal{X} \to \mathcal{X}/c = \mathcal{Y}$. On the other hand, setting $c(x_1, x_2) = \delta_{f(x_1)=f(x_2)} = 1$ for any given function $f$ is reflexive, symmetric and transitive because the equality $f(x_1) = f(x_2)$ is. □

**Definition 2.** *The contrast function $c$ is admissible if, and only if, $c(x_1, x_2) = 1$ defines an equivalence relation.*

Full knowledge of the contrast function $c$ only specifies the level sets of the function $f$:

**Lemma 2.** *Let $f$ be any function compatible with the admissible contrast $c$. Then, we can write $f = \iota \circ \hat{f}$ as the composition of an injection $\iota : \mathcal{X}/f \to \mathcal{Y}$ and the (unique) projection $\hat{f} : \mathcal{X} \to \mathcal{X}/c$ of $\mathcal{X}$ onto the equivalence classes $\mathcal{X}/c$ of the equivalence relation $c(x_1, x_2) = 1$.*

*Proof.* From elementary algebra, we can decompose any function $f : \mathcal{X} \to \mathcal{Y}$ as

$$f : \mathcal{X} \xrightarrow{\hat{f}} \mathcal{X}/f \xrightarrow{\iota} \mathcal{Y}$$

where $\iota$ is an injective function and $\hat{f}$ projects $\mathcal{X}$ to the quotient $\mathcal{X}/f$, i.e. the collection of subsets $X \subset \mathcal{X}$ where $f(x)$ is constant (level sets). The latter are also the equivalence classes of the relation $f(x_1) = f(x_2)$. Due to definition 1, this is the same equivalence relation given by the contrast $c$, so that $\mathcal{X}/f = \mathcal{X}/c$. □

Note that, in our contrastive learning formulation, we do *not* define the contrast $c$ on the sample space $\mathcal{X}$, but rather on the transformation space $\mathcal{T}$. The following lemma suggests that defining a contrast $c(T, T')$ on transformations instead of data samples is usually acceptable:

**Lemma 3.** *Let $c : \mathcal{T} \times \mathcal{T} \to \{0, 1\}$ be an admissible contrast function defined on a set (e.g., a batch) of generalized data transformations $\mathcal{T}$. Furthermore, let $x$ be a dataset and let $x(T) \in \mathcal{X}$ be the sample indexed by transformation $T$. If $x(T) = x(T') \Rightarrow T = T'$ (i.e. different transformations output different samples), then setting $\tilde{c}(x(T), x(T')) = c(T, T')$ defines part of an admissible sample contrast function $\tilde{c}$ on $\mathcal{X}$.*

*Proof.* The expression above defines $\tilde{c}$ on the sample space $\tilde{\mathcal{X}} = \{x(T) : T \in \mathcal{T}\} \subset \mathcal{X}$. Reflexivity, symmetry and transitivity are inherited from $c$. However, if the same data point $x(T) = x(T')$ can be obtained from two different transformations $T$ and $T'$, the definition is ill posed. The hypothesis in the lemma guarantees that this is not the case. □

### A.1.1 Compositional transformations

Next, we consider the case in which $T = (t_1, \ldots, t_M)$ is a composition of individual transformations $t_m$, each with its own contrast $t_m$:

**Definition 3.** *We say that a contrast function $c(t_m, {t'}_m)$ is distinctive if it is given by $\delta_{t_m={t'}_m}$. We say that it is invariant if it is identically one.*

The following lemma provides a formula for the overall contrast function $c(T, T)$ given the contrasts for the individual factors.

**Lemma 4.** *Let $c(t_m, t'_m) = 1$ be admissible contrast functions, either distinctive or invariant. Then, the product $c(T, T') = \prod_{m=1}^{M} c(t_m, t'_m)$ is also admissible.*

*Proof.* The reflexive and symmetric properties are obviously inherited. For the transitive property, note that $c(T, T') = 1$ if, and only if, $\forall m : c(t_m, t'_m) = 1$. Hence:

$$\begin{aligned} c(T, T') = c(T', T'') = 1 \\ \Rightarrow \quad \forall m : c(t_m, t'_m) = c(t'_m, t''_m) = 1 \\ \Rightarrow \quad \forall m : c(t_m, t''_m) = 1 \quad \Rightarrow \quad c(T, T'') = 1. \end{aligned}$$

□

Finally, we show that, essentially, the formula above is the only reasonable one. For this, we only require $c(T, T')$ to be monotonic in the individual factors; i.e., if more factors become 1, then $c(T, T')$ can only grow:

**Definition 4.** *We say that $c(T, T')$ is monotonic in the individual factors if, and only if, for any three transformations $T, T', T''$ such that $c(t_m, t'_m) \leq c(t_m, t''_m)$ for all the factors, then we also have $c(T, T') \leq c(T, T'')$.*

Next, we show that $c$ can only have a very limited form:

**Lemma 5.** *Suppose that the admissible monotonic contrast $c(T, T')$ is expressible solely as a function of the individual admissible contrasts $c(t_m, t'_m)$ for $m = 1, \ldots, M$. Then, up to a permutation of the transformations, we can always write*

$$c(T, T') = \prod_{i=1}^{m} c(t_i, t'_i)$$

*where $0 \leq m \leq M$. In particular, $m = M$ is the only option that is guaranteed not to ignore some of the factors.*

*Proof.* From the assumptions, we can write

$$c(T, T') = h \circ v(T, T')$$

where $h$ is a function of the binary vector

$$v(T, T') = (c(t_1, t'_1), \ldots, c(t_M, t'_M)) \in \mathbb{B}^M.$$

Furthermore, since invariant factors are constant, they do not affect the function; hence, without loss of generality we can assume that all factors are distinctive.

Since all factors are distinctive, we can construct two transformations $T' = (t'_1, \ldots, t'_M)$ and $T'' = (t''_1, \ldots, t''_M)$ such that $v(T', T'') = (0, \ldots, 0)$ (i.e., all the contrasts $c(t'_m, t''_m)$ are null). If $c(T', T'') = 1$, then, due to monotonicity, $c(T', T'')$ is identically 1 and the lemma is proved for $m = 0$.

If not, let $c(T', T'') = h(0, \ldots, 0) = 0$. Then, for any given binary vector $v$, we can construct a transformation $T = (t_1, \ldots, t_M)$ such that $v(T, T') = v$ and $v(T, T'') = \bar{v}$ as follows:

$$t_m = \begin{cases} t'_m, & \text{if } v_m = 1, \\ t''_m, & \text{otherwise.} \end{cases}$$

We cannot have $c(T, T') = h(v) = h(\bar{v}) = c(T, T'') = 1$; otherwise, due to the transitivity of $c$, we would have $c(T', T'') = h(0, \ldots, 0) = 1$, which contradicts our assumption. Hence, $h$ must partition the space of binary vectors in two halves, the ones for which $h(v) = 1$ and their complements $h(\bar{v}) = 0$.

Now let $v$ be a vector with the minimal number of 1 such that $h(v) = 1$. Again without loss of generality, we can assume this is of the type $v = (1, \ldots, 1, 0, \ldots, 0)$ with $m$ ones in front. Due to monotonicity, all vectors of type $v' = (1, \ldots, 1, v_{m+1}, \ldots, v_M)$ must also have $h(v') = 1$; by taking their complement, the previous result shows that all vectors $v'' = (0, \ldots, 0, v_{m+1}, \ldots, v_M)$ must have $h(v'') = 0$. This is also the case for any vector of the type $(v_1, \ldots, v_m, 0, \ldots, 0)$ where any $v_i = 0$ for $1 \leq i \leq m$ (because $m$ is the minimum number of ones required for $h(v) = 1$). We conclude that $h(v) = 1$ if, and only if, $(v_1, \ldots, v_m) = (1, \ldots, 1)$. □

### A.1.2 Forming batches

Let $\hat{\mathcal{T}}_1 \times \cdots \times \hat{\mathcal{T}}_M$ be a composite space of generalized data transformations, so that data points are indexed as $x(t_1, \ldots, t_M)$. Furthermore, let $c(t_m, t'_m)$ be corresponding admissible contrast functions and let $c(T, T')$ be their product, as in lemma 4. As before, we assume that the functions are of two kinds:

- invariant: $c(t_m, t'_m) = 1$.
- distinctive: $c(t_m, t'_m) = \delta_{t_m = t'_m}$.

Let $I \subset \{1, \ldots, M\}$ be the subset of indices $m$ corresponding to the invariant transformations and $D = \{1, \ldots, M\} \backslash I$ the distinctive ones.

Let $\text{sample}(\hat{\mathcal{T}}_m; K_m)$ be a stochastic operator that samples $K_m \leq |\hat{\mathcal{T}}_m|$ transformations from $\hat{\mathcal{T}}_m$ without replacement. We sample a batch recursively:

- Let $\mathcal{T}_1 = \text{sample}(\hat{\mathcal{T}}_1; K_1)$
- Let $\mathcal{T}_m = \bigcup_{T \in \mathcal{T}_{m-1}} T \circ \text{sample}(\hat{\mathcal{T}}_m; K_m)$

At each level of the recursion, each transformation is extended by sampling $K_m$ more transformations (note that no two identical transformations can be generated in this manner). Hence $|\mathcal{T}_M| = K_1 \cdots K_M$.

**Lemma 6.** *There are exactly $(\prod_m K_m)(\prod_{m \in I} K_m)$ pairs of transformations $(T, T') \in \mathcal{T}_M \times \mathcal{T}_M$ for which $c(T, T') = 1$. Of these, exactly $\prod_m K_m$ are trivial pairs ($T = T'$). Hence, there are $(\prod_m K_m)(\prod_{m \in I} K_m - 1)$ non-trivial pairs for which $c(T, T') = 1$.*

**Lemma 7.** *For each $T \in \mathcal{T}_M$, there are exactly $(\prod_m K_m) - (\prod_{m \in I} K_m)$ pairs $(T, T'')$ such that $c(T, T'') = 0$.*

For example, in SimCLR $M = 2$, $D = \{1\}$, $I = \{2\}$, $K_1 = B/2$, $K_2 = 2$, $|\mathcal{T}_2| = B$, there are $B(2 - 1) = B$ non-trivial pairs of transformations for which $c(T, T') = 1$, and, for each $T$, there are $B - 2$ pairs of transformations for which $c(T, T'') = 0$.

The lemmas above suggest that we should pick $K_m \geq 2$ for at least one invariant factor and at least $K_m \geq 2$ for at least one distinctive factor, as otherwise eq. (1) is degenerate.

### A.1.3 Limitations

In general, we want more restrictive requirements than the one described above. When learning $f$, difficult (and therefore interesting) cases amount to: learning to be sensitive

to 'small' variations in the distinctive factors and learn to be insensitive to 'large' variations in the invariant factors. For the former, we would like $f$ to observe variations in a single distinctive factor at a time, as these are the 'smallest'. For these individual variations to exist at all in the batch, we should choose $K_m \geq 2$ for all distinctive factors $m \in D$.

Even so, the hierarchical scheme in general *prevents* us from observing all individual variations. In fact, suppose that two transformations $T$ and $T'$ in $\mathcal{T}_M$ differ for factor $m$ (i.e. $t_m \neq t'_m$). Then, the remaining factors $t_{m+1} \neq t'_{m+1}, \ldots$ also differ in general because successive transformations are sampled independently in different branches of the tree. This means that we cannot, in general, observe a change in $t_m$ *alone*, so the function $f$ may not learn to be distinctive to this 'minimal' change in isolation.

Note that this is a limitation that affects our sampling scheme as well as existing methods such as SimCLR. Fortunately, in practice this is often not an issue. There are in fact two mitigating factors, which apply to most existing formulations, including the new ones presented here.

First, some transformations spaces $\hat{\mathcal{T}}_m$ are very small, and in fact binary (e.g., modality splitting, time reversal). In this case, $K_m = 2$ means that transformations are sampled exhaustively, so for level $m$ the hierarchical sampling scheme does extract all possible combinations of transformations.

Second, in other cases the issue is moot due to the nature of the transformations and the data. For instance, in SimCLR the first transformation $t_1$ amounts to sampling a certain image $x_i$, and the second transformation $t_2$ amounts to sampling two data augmentation $g_{1i}(x_i)$ and $g_{2i}(x_i)$, different for each image. The issue here is that we cannot observe a change in the index $i$ for the same augmentation $g(x_1)$ and $g(x_2)$, as these data points do not exist in the batch. This means that the representation $f$ can only learn to distinguish two different images $x_i$ that also have two different augmentations applied to them. Because of the particular nature of the training data (ImageNet) this is likely irrelevant since different images $x_i$ are unrelated in any case, so applying transformations does not significantly alter their statistical relationships.

However, note that this is not *always* the case. For instance, if SimCLR was applied to a dataset of pre-aligned faces (for the purpose of learning face recognition), then being unable to contrast different faces $g(x_1)$ and $g(x_2)$ under the same transformation $g$ would make negative pairs far to easy to discriminate.

## A.2. Reduction in variance theorem

### A.2.1 Proof of theorem 1

For ease of notation, we will express eq. 1 as the expected value of a loss function $\ell$, which subsumes the weight ($w$), contrast ($c$), feature extractor ($\Phi$) and log-softmax functions:

$$\mathcal{L} = \mathbb{E}_{T,T' \sim \hat{\mathcal{T}}} \left[ \ell \left( x(T), x(T') \right) \right]. \tag{2}$$

The expectation is over pairs of transformations in $\hat{\mathcal{T}} = \hat{\mathcal{T}}_1 \times \ldots \times \hat{\mathcal{T}}_M$, the space of all compositions of transformations, which can be applied to the data $x$. Note that eq. 1 contains a sum over a third transformation ($T''$) to compute the softmax's normalization, which is also subsumed by $\ell$ in eq. 2 as this third transformation is not essential for the rest of the proof. We will separate each transformation into invariant and distinctive parts, $T = (T^I, T^V)$ respectively with $T^I \in \hat{\mathcal{T}}_I$ and $T^V \in \hat{\mathcal{T}}_V$ (see sec. A.1.2). Note that this separation is merely a notational convenience; the individual transformations can be applied to the data in *any* order, with $x(T^I, T^V) = x(t_1 \circ \ldots \circ t_M)$, and each $t_i$ belonging to either $T^I$ or $T^V$. Then, eq. 2 becomes:

$$\mathcal{L} = \mathbb{E}_{T^I, T'^I \sim \hat{\mathcal{T}}_I,\, T^V, T'^V \sim \hat{\mathcal{T}}_V} \left[ \ell \left( x(T^I, T^V), x'(T'^I, T'^V) \right) \right].$$

Consider a mini-batch of data sample pairs and their associated transformation compositions, $\mathcal{B}_{\text{direct}} = \left\{ T^I_i, T^V_i, {T'}^I_i, {T'}^V_i \right\}_{i=1}^{K_I^2 K_V^2}$, sampled as $T^I_i, {T'}^I_i \sim \hat{\mathcal{T}}_I$ and $T^V_i, {T'}^V_i \sim \hat{\mathcal{T}}_V$. The batch size is a function of $K_I = \prod_{j \in I} K_j$ and $K_V = \prod_{j \in V} K_j$, the number of sampled invariant and distinctive transformations in our method, respectively. The batch size of $K_I^2 K_V^2$ was chosen to allow a direct comparison. The expected value of the loss over this batch is then the simple empirical average:

$$\hat{\mathcal{L}}_{\text{d}} = \frac{1}{K_I^2 K_V^2} \sum_{i}^{K_I^2 K_V^2} \ell \left( x(T^I_i, T^V_i), x({T'}^I_i, {T'}^V_i) \right). \tag{3}$$

Now consider the domain of transformed samples $\mathcal{X} = \{ x(T^I, T^V) : T^I \in \hat{\mathcal{T}}_I, T^V \in \hat{\mathcal{T}}_V \}$. Due to the assumed injectivity of all $t \in T^I$, we may partition the domain using one partition $\mathcal{X}_j = \{ x(T^I, T^V_j) : T^I \in \hat{\mathcal{T}}_I \}$ per distinctive transformation $T^V_j$, i.e.: $\mathcal{X} = \cup_j^{K_V} \mathcal{X}_j$, with $\mathcal{X}_j \cap \mathcal{X}_{j'} = \emptyset$, $\forall j, j'$. The probability distribution of the samples has density $p(T^I, T^V)$, and the density in each partition is thus $p_j(T^I) = K_V p(T^I) \delta_{T^I \in \mathcal{X}_j}$, with $\delta$ the indicator function.

GDT can then be interpreted as a stratified sampling method, with one stratum (partition) per *pair* of distinctive transformations. The domain being sampled by the expectation in eq. 2 is $\mathcal{X}^2 = \cup_{jj'}^{K_V, K_V} \mathcal{X}_j \times \mathcal{X}_{j'}$, and stratified sampling consists of sampling an equal number of $K_I^2$ sample pairs from each of the $K_V^2$ partitions:

$$\hat{\mathcal{L}} = \frac{1}{K_V^2 K_I^2} \sum_{ii'jj'}^{K_I, K_I, K_V, K_V} \ell \left( x(T^I_i, T^V_j), x(T^I_{i'}, T^V_{j'}) \right). \tag{4}$$

Note the subtle difference from eq. 3 in the summation ranges, and that the *same* samples and transformations are

reused for both elements of each pair, instead of being sampled independently to fill a mini-batch. To make the following derivations easier, note that we can equivalently express eq. 4 as:

$$\hat{\mathcal{L}} = \frac{1}{K_V^2} \sum_{jj'}^{K_V,K_V} \hat{\mathcal{L}}_{jj'},$$

with $\hat{\mathcal{L}}_{jj'} = \frac{1}{K_I^2} \sum_{ii'}^{K_I,K_I} \ell\left(x(T_i^I, T_j^V), x(T_{i'}^I, T_{j'}^V)\right)$. We will first show that this pairwise stratified sampling is an unbiased estimate of eq. 2:

$$\begin{aligned}
\mathbb{E}[\hat{\mathcal{L}}] &= \frac{1}{K_V^2} \sum_{jj'}^{K_V,K_V} \mathbb{E}[\hat{\mathcal{L}}_{jj'}] \\
&= \frac{1}{K_V^2} \sum_{jj'}^{K_V,K_V} \mathcal{L}_{jj'} \\
&= \mathcal{L},
\end{aligned}$$

where we use the expectation $\mathcal{L}_{jj'}$ of the loss function evaluated on the partition $jj'$ (corresponding to distinctive transformations with indices $j$ and $j'$), as $\mathcal{L}_{jj'} = \mathbb{E}_{T^I \in \mathcal{X}_j, T'^I \in \mathcal{X}_{j'}} \left[\ell\left(x(T^I, T_j^V), x(T'^I, T_{j'}^V)\right)\right]$.

Similarly, we can also define each partition's loss variance $\sigma_{jj'}^2 = \mathbb{V}_{T^I \in \mathcal{X}_j, T'^I \in \mathcal{X}_{j'}} \left[\ell\left(x(T^I, T_j^V), x(T'^I, T_{j'}^V)\right)\right]$. Then, from eq. 4 we obtain directly

$$\begin{aligned}
\mathbb{V}[\hat{\mathcal{L}}] &= \frac{1}{K_V^4} \sum_{jj'}^{K_V,K_V} \mathbb{V}[\mathcal{L}_{jj'}] \\
&= \frac{1}{K_V^4 K_I^2} \sum_{jj'}^{K_V,K_V} \sigma_{jj'}^2.
\end{aligned}$$

As a point of comparison, the variance of the direct sampling estimate is:

$$\begin{aligned}
\mathbb{V}[\hat{\mathcal{L}}_{\text{d}}] &= \frac{1}{K_V^2 K_I^2} \left( (\mathbb{E}_{(T^I, T'^I) \in \mathcal{X}^2} [\ell^2(x(T^I, T^V), x(T'^I, T'^V) \right. \\
&\qquad \left. x(T'^I, T'^V))] - \mathcal{L}^2) \right) \\
&= \frac{1}{K_V^2 K_I^2} \left( \frac{1}{K_V^2} \sum_{jj'}^{K_V,K_V} \mathbb{E}_{T^I \in \mathcal{X}_j, T'^I \in \mathcal{X}_{j'}} \right. \\
&\qquad \left. \left[\ell^2\left(x(T^I, T_j^V), x(T'^I, T_{j'}^V)\right)\right] - \mathcal{L}^2 \right) \\
&= \frac{1}{K_V^2 K_I^2} \left( \frac{1}{K_V^2} \sum_{jj'}^{K_V,K_V} \left(\mathcal{L}_{jj'}^2 + \sigma_{jj'}^2\right) - \mathcal{L}^2 \right) \\
&= \frac{1}{K_V^4 K_I^2} \sum_{jj'}^{K_V,K_V} \left(\sigma_{jj'}^2 + (\mathcal{L}_{jj'} - \mathcal{L})^2\right) \\
&\geq \frac{1}{K_V^4 K_I^2} \sum_{jj'}^{K_V,K_V} \sigma_{jj'}^2
\end{aligned}$$

completing the proof. □

## A.3. Additional experimental results

### A.3.1 Modality ablation

In table A.1, we provide the results of running our baseline model (sample-distinctiveness only) within-modally instead of across modalities and find a sharp drop in performance.

Table A.1: **Within vs cross-modal learning.** Results on action classification performance on HMDB-51 and UCF-101 is shown for finetuning accuracy (Acc) and frozen retrieval (recall@1) after pretraining on Kinetics-400 for 50 epochs.

| | HMDB | | UCF | |
|---|---|---|---|---|
| | **Acc.** | **R@1** | **Acc.** | **R@1** |
| Within-modal | 37.8 | 13.9 | 76.4 | 28.0 |
| Cross-modal | **52.4** | **21.8** | **87.6** | **54.8** |

### A.3.2 Dataset details

The Kinetics-400 dataset [39] is human action video dataset, consisting of 240k training videos, with each video representing one of 400 action classes. After filtering out videos without audio, we are left with 230k training videos, which we use for pretraining our model.

HT100M [50] is a large-scale instructional video collection of 1.2 million Youtube videos, along with automatic speech recognition transcripts. There are more than 100 million clips (ASR segments) defined in HowTo100M.

HMDB-51 [42] consists of 7K video clips spanning 51 different human activities. HMDB-51 has three train/test splits of size 5k/2k respectively.

UCF-101 [67] contains 13K videos from 101 human action classes, and has three train/test splits of size 11k/2k respectively.

IG65M [21] is a large-scale weakly supervised dataset collected from a social media website, consisting of 65M videos of human action events. We use the all the videos in the dataset for pretraining.

### A.3.3 Preprocessing details

The video inputs are 30 consecutive frames from a randomly chosen starting point in the video. These frames are resized such that the shorter side is between 128 and 160, and a center crop of size 112 is extracted, with color-jittering applied. A random horizontal flip is then applied with probability 0.5, and then the inputs' channels are z-normalized using

mean and standard deviation statistics calculated across each dataset.

One second of audio is processed as a $1 \times 40 \times 99$ image, by taking the log-mel bank features with 40 filters and 99 time-frames after random volume jittering between 90% and 110% is applied to raw waveform, similar to [4]. The spectrogram is then Z-normalized, as in [41]. Spec-Augment is then used to apply random frequency masking to the spectrogram with maximal blocking width 3 and sampled 1 times. Similarly, time-masking is applied with maximum width 6 and sampled 1 times.

For the text, we remove stop words from the narrations as in [50]. For each narration, we take a maximum of 16 consecutive words covering a max duration of 4 seconds as in [49].

### A.3.4 Pretraining details

We use R(2+1)D-18 [72] as the visual encoder $f_v$ and ResNet [32] with 9 layers as the audio encoder $f_a$ unless otherwise noted; both encoders produce a fixed-dimensional output (512-D) after global spatio-temporal average pooling. For the text encoder, we use the Google News self-supervised pre-trained word2vec (d=300) embedding [51], that is linearly projected to 2048D and max-pooled as in [49]. After the inputs are encoded by their respective modality encoders, the vectors are then passed through two fully-connected layers with intermediate size of 512 to produce 256-D embeddings as in [8] which are normalized by their L2-norm [77]. The embedding is used for computing the contrastive loss, while for downstream tasks, a linear layer after the global spatio-temporal average pooling is randomly initialized. For NCE contrastive learning, the temperature $\rho$ is set as $1/0.07$. For optimizing these networks, we use SGD. The SGD weight decay is $10^{-5}$ and the SGD momentum is $0.9$. We use a mini-batch size of 8 on each of our 64 GPUs giving an effective batch size of 512 for distributed training. The initial learning rate is set to $0.01$ which we linearly scale with the number of GPUs, after following a gradual warm-up schedule for the first 10 epochs [24]. For Kinetics, we train for 100 epochs (3 days), while for HT100M, we train for 40 epochs (3 days).

### A.3.5 Ablation experiment details

For the ablations, we only pretrain for 50 epochs on the Kinetics-400 dataset, and 20 epochs on the HT100M dataset, since it is a much larger dataset.

For downstream evaluation, we only evaluate on the first fold of HMDB-51 but found the performance between folds to be close (within 1%).

### A.3.6 Evaluation details

All evaluation code is provided in the Supplementary Material.

**Video Action Recognition.** During training, we take 10 random clips of length 32 frames from each video. For video clip augmentations, we follow a standard protocol as in [41]. During evaluation, we uniformly sample 10 clips from each video, average softmax scores, and predict the class having the highest mean softmax score. We then measure the mean video top-1 accuracy across all videos and all official folds. During training, we use SGD with initial learning rate $0.0025$, which we gradually warm up to $2 \cdot 10^{-2}$ in the first 2 epochs. The weight decay is set to $5 \cdot 10^{-3}$ and momentum to $0.9$. We use a mini-batch size of 32 and train for 12 epochs with the learning rate multiplied by $5 \cdot 10^{-2}$ at 6 and 10 epochs. We compare our GDT pretrained model with both self-supervised methods, and supervised pretraining, and report average top-1 accuracies on UCF101 and HMDB-51 action recognition task across three folds in table 4.

**Few-shot classification** We follow the protocol in [38] and evaluate our our GDT pretrained network using few-shot classification on the UCF-101 dataset, and additionally on HMDB-51. We randomly sample $n$ videos per class from the train set, average the encoder's global average pooled features from ten clips per training sample and measure classification accuracy performance on the validation set using a $k$-nearest neighbor classifier, with $k$ set to 1.

**Video Retrieval.** We follow the standard protocol as outlined in [80]. We use the split 1 of UCF101, and additionally HMDB-51. We uniformly sample 10 clips per video, and average the max-pooled features after the last residual block for each clip per video. We use these averaged features from the validation set to query the videos in the training set. The cosine distance of representations between the query clip and all clips in the training set are computed. When the class of a test clip appears in the classes of $k$ nearest training clips, it is considered to be correctly predicted. We report accuracies for $k = 1, 5, 20$ and compare with other self-supervised methods on UCF101 and HMDB-51 in table 3.

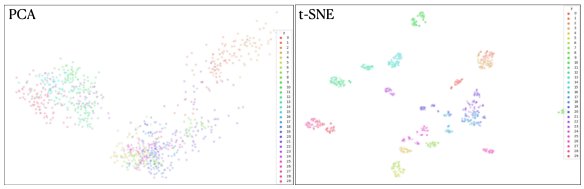


Fig. A.1: Feature visualizations with PCA and t-SNE on 30 videos of a single, random class of HMDB-51. For each video, we sample 10 temporal clips and encode video-ID with color. Embeddings are generated from our time-shift distinct model (Tab.1 (l)).

### A.3.7 Additional Qualitative analysis

In fig. A.1, we present a PCA and t-SNE [73] plots of the features obtained by our model (DS-d, TR-d, TS-d) (Tab. 1, row (l)). We observe that even comparing to videos of the same action category, the individual clips are well separated, showing that the model is learning to distinguish between different time intervals.