
TROVE: Discovering Error-Inducing Static Feature Biases in Temporal Vision-Language Models

Maya Varma¹ Jean-Benoit Delbrouck^{1,2} Sophie Ostmeier¹
Akshay Chaudhari¹ Curtis Langlotz¹
¹Stanford University ²HOPPR
mayavarma@cs.stanford.edu

Abstract

Vision-language models (VLMs) have made great strides in addressing temporal understanding tasks, which involve characterizing visual changes across a sequence of images. However, recent works have suggested that when making predictions, VLMs may rely on *static feature biases*, such as background or object features, rather than dynamic visual changes. Static feature biases are a type of shortcut and can contribute to systematic prediction errors on downstream tasks; as a result, identifying and characterizing error-inducing static feature biases is critical prior to real-world model deployment. Existing approaches for identifying such systematic failure modes in trained models (i) are typically designed for non-temporal settings and (ii) are challenging to evaluate in temporal settings due to the lack of quantitative evaluation frameworks. In this work, we address these challenges by introducing TROVE, an automated approach for discovering error-inducing static feature biases learned by temporal VLMs. Given a trained VLM and an annotated validation dataset associated with a downstream classification task, TROVE extracts candidate static features from the dataset and scores each feature by (i) the effect of the feature on classification errors as well as (ii) the extent to which the VLM relies on the feature when making predictions. In order to quantitatively evaluate TROVE, we introduce an evaluation framework consisting of 101 trained temporal VLMs paired with ground-truth annotations for learned static feature biases. We use this framework to demonstrate that TROVE can accurately identify error-inducing static feature biases in VLMs, achieving a 28.6% improvement over the closest baseline. Finally, we apply TROVE to 7 off-the-shelf VLMs and 2 temporal understanding tasks, surfacing previously-unknown static feature biases and demonstrating that knowledge of learned biases can aid in improving model performance at test time. Our code is available at <https://github.com/Stanford-AIMI/TroVe>.

1 Introduction

Vision-language models (VLMs) capable of jointly processing visual and textual data have been shown to possess state-of-the-art reasoning abilities [1–7]. In particular, given an input sequence with multiple images collected across varying timepoints, *temporal VLMs* can effectively characterize visual changes over time, a capability known as temporal understanding [8–15]. For example, temporal VLMs can recognize human actions given a sequence of video frames [11–13] and characterize disease progression given longitudinal medical images [8–10].

Models designed to perform temporal understanding tasks often demonstrate high overall performance; however, recent works have demonstrated that such models may be affected by *static feature biases*, a phenomenon where models utilize static patterns (e.g. image background or a particular

object in the scene) as shortcuts when making predictions rather than analyzing dynamic visual changes occurring across the image sequence [16–20]. As an illustrative example, consider videos from an activity recognition dataset with the class label “climbing tree”, which depict people climbing up or down trees (Figure 1). A temporal VLM is tasked with accepting a sequence of video frames as input and then classifying the action being performed by the person in the scene. In this scenario, a VLM that relies on static feature biases may base predictions for the class label “climbing tree” solely on the presence of trees and foliage, rather than analyzing the true motion patterns associated with a person climbing a tree. At inference time, the performance of this VLM will depend heavily on whether the static feature is present; consequently, as illustrated in Figure 1, the VLM is likely to make systematic prediction errors when classifying videos from other classes (e.g. “swinging on something”) with prominently visible trees.

Identifying learned static feature biases that contribute to systematic prediction errors is critical prior to real-world model deployment [21]. Traditional approaches for detecting such failure modes, which typically involve a combination of manual analysis and pixel-wise interpretability algorithms (e.g. GradCAM), require extensive human effort and are time-consuming to implement at scale, particularly as the length of the input sequence increases [22, 23]. This suggests the need for automated approaches; however, performing automated identification of error-inducing static feature biases is challenging for the following two reasons. First, existing automated approaches for discovering systematic errors are designed for non-temporal (e.g. single image) settings [24–27]. Such approaches, which typically operate by analyzing model predictions on a labeled validation dataset and surfacing coherent groups of misclassified samples, are not adequate for discovering static feature biases in settings where each data sample consists of a sequence of multiple, temporally-linked images. Second, performing quantitative evaluations of automated approaches in the temporal setting is complicated by the fact that the ground-truth static feature biases of pretrained models are typically unknown; as a result, it is difficult to ascertain whether biases extracted by automated methods are indeed accurate.

In this work, we address these challenges by introducing TROVE, an automated approach for improving Temporal **R**obustness of **V**ision-**L**anguage models. Given a pretrained VLM, our goal is to discover learned static feature biases that contribute to systematic prediction errors on downstream temporal understanding tasks. Knowledge of such static feature biases (e.g. trees in the previously-discussed example) can enable a developer to better understand and address model failure modes prior to real-world deployment. To this end, TROVE operates on a labeled validation dataset by first decomposing each input multi-image sequence into constituent images and grouping visually-similar images into clusters. Here, each cluster represents a particular feature occurring consistently throughout the dataset. We then introduce a scoring function that ranks each feature by (i) the effect of the feature on classification errors as well as (ii) the extent to which the VLM relies on the feature when making predictions. As output, TROVE generates a list of identified static feature biases paired with affected class labels.

In order to assess the utility of our approach, we design an evaluation framework consisting of 101 temporal VLMs trained on synthetic data. We pair each VLM with annotations for ground-truth error-inducing static feature biases, enabling rigorous quantitative analyses. Across this suite of models, TROVE accurately discovers error-inducing static feature biases, achieving a 28.6% improvement over the closest baseline. We find that TROVE operates effectively across a range of static feature bias types (background biases, object biases, and attribute biases) and input sequence lengths.

Given the strong performance of TROVE on synthetic experimental settings, we then extend TROVE to real-world temporal VLMs. Across a suite of seven state-of-the-art VLMs and two temporal

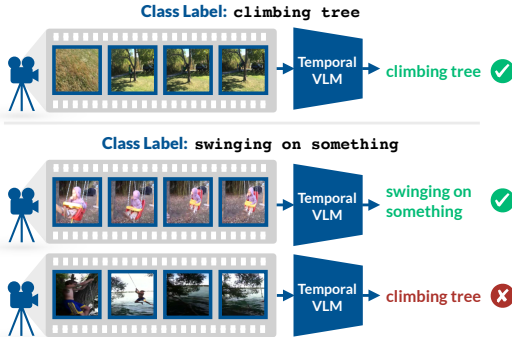


Figure 1: Temporal models often rely on the presence of static feature biases, such as background features or objects in the scene, when making predictions. In this example, the VideoCLIP-XL model [12] makes systematic prediction errors on the class swinging on something when trees are present.

understanding tasks (activity recognition in videos and disease progression classification in medical images), TROVE accurately surfaces previously-unknown static feature biases. We further demonstrate that knowledge of TROVE-discovered features can aid in improving test-time VLM performance; we present an approach for mitigating prediction errors without the need for data augmentation or VLM retraining, resulting in performance improvements on an activity recognition task of up to 111% on sequences containing static features.

Ultimately, TROVE demonstrates strong practical utility and can serve as an effective tool for evaluating and improving robustness of temporal VLMs.

2 Related Work

Discovering systematic errors in non-temporal settings: Since subgroup labels are typically unavailable in datasets, identifying and understanding critical failure modes of models can be challenging. Early approaches paired visualization techniques with humans in the loop to identify model failures [28, 22]; however, this approach is time-consuming and difficult to scale effectively across large numbers of models and tasks. To address this challenge, a recent line of work has proposed automated approaches for the task of identifying systematic prediction errors made by models; notable methods in this domain include Domino [24], Distilling Failures [25], and George [26], among others [29–31]. Such methods typically analyze model predictions on a labeled validation set and identify features in the data (e.g. a specific visual cue in image settings) that systematically contribute to mispredicted labels. Model developers can then use these identified error patterns to update models prior to deployment [32].

Although such approaches have been shown to be effective, these methods are predominantly designed for non-temporal settings, where each input sample (e.g. image, text) can be represented as a single entity. In contrast, input samples in the temporal setting take the form of multi-image sequences, and collapsing an entire sequence to a single entity provides inadequate granularity for detecting errors resulting from static feature biases. As we demonstrate in Section 4, naive extensions of such methods to temporal settings fail to work effectively, particularly when error-inducing features are visible in only a subset of the sequence.

Our approach also builds upon recent fine-grained approaches in the image setting that leverage region-level information for detecting systematic errors [27, 33]. However, these approaches are designed and evaluated solely on non-temporal, single-image settings.

Discovering systematic errors in temporal settings: Prior works have noted that temporal models often rely on static feature biases as shortcuts when making predictions [20, 34–36]. In such settings, using just a single frame as input to a model can result in high performance on multi-image temporal understanding tasks [17, 20]. A range of approaches have been proposed for reducing model reliance on static feature biases, predominantly in the context of background biases in video-based activity recognition tasks [37, 16, 18, 19, 38, 39]; such approaches typically involve novel optimization procedures or data augmentation strategies. We draw a key distinction between these works and our approach: whereas this line of work focuses explicitly on mitigating the influence of static feature biases during the model training procedure, our work instead aims to accurately *discover* learned biases given a pretrained temporal model. We also extend beyond the human activity recognition setting, including evaluations on both a synthetic task as well as a medical imaging task.

3 Our Approach: TROVE

We now introduce TROVE, an approach for improving temporal robustness of VLMs by discovering learned static feature biases that contribute to systematic prediction errors. In Section 3.1, we formally describe our problem setting. We then present methodological details for TROVE in Section 3.2. An overview of TROVE is provided in Figure 2.

3.1 Preliminaries

VLMs designed to perform temporal understanding tasks are generally trained on large-scale datasets of the form $\mathcal{D} = \{(S_i, T_i)\}_{i=1}^m$, where S_i represents a multi-image sequence and T_i represents paired text in the form of a caption or description. Each input sequence S_i can be expressed as

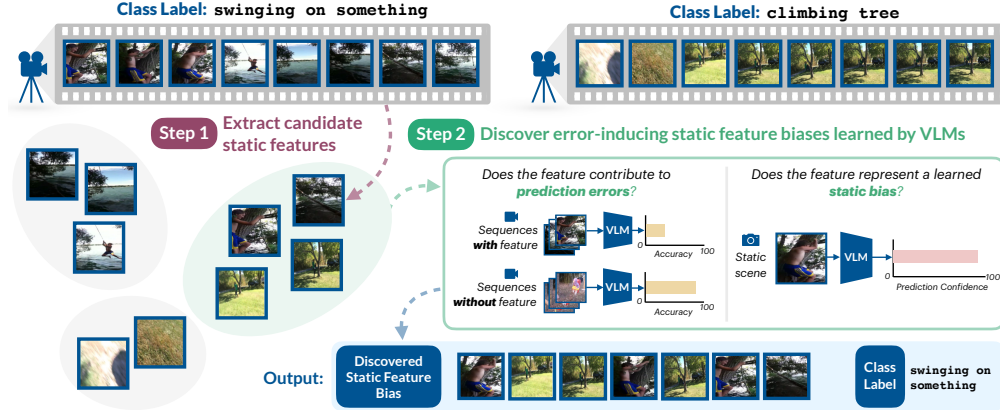


Figure 2: TROVE is an automated approach for discovering error-inducing static feature biases learned by temporal VLMs. In this example, TROVE identifies a static feature bias associated with trees, which results in degraded performance on the class label `swinging on something` [40].

$S_i = (I_i^1, I_i^2, \dots, I_i^{n_i})$, where each I_i represents a single image and n_i represents the total number of images in sequence S_i . At inference time, temporal VLMs are evaluated using downstream tasks that assess the ability of the model to understand visual changes over time (e.g. activity recognition, disease progression classification). By definition, an effective downstream temporal understanding task will require the model to parse a multi-image sequence and analyze dynamic visual patterns.

In this work, we focus explicitly on downstream temporal understanding tasks formulated as classification problems, where inference datasets are expressed as $\mathcal{D}_V = \{(S_i, y_i)\}_{i=1}^p$ for sequences S_i and class labels $y \in \mathcal{Y}$. Here, $\mathcal{Y}$ represents the ground-truth label set associated with the task, and we assume that $n_i > 1$ for all sequences $S_i \in \mathcal{D}_V$. Recent works have suggested that when making predictions at inference time, models trained to perform temporal understanding tasks may rely heavily on static feature biases, such as background features or objects in the scene, rather than rely on the true dynamic visual changes [16, 34]. For example, as shown in Figure 1, the recently-introduced VideoCLIP-XL model relies on the presence of trees, a static feature, when assigning predictions for the class label $y = \text{climbing tree}$ [12]. Static feature biases are a type of shortcut and can ultimately result in systematic prediction errors at inference time; in Figure 1, this manifests as low performance on other classes in $\mathcal{Y} \setminus \{y\}$ when the static feature is present, such as the class label `swinging on something`.

3.2 Discovering Static Feature Biases

Our key goal in this work is to discover static features that meet the following two criteria. First, identified static features should be *error-inducing*, meaning that the presence of the feature within a sequence directly contributes to prediction errors on the downstream temporal understanding task of interest. Second, identified static features should reflect a *learned model bias*, suggesting that the model relies on the presence of the feature when making predictions. However, designing an automated algorithm that satisfies these two criteria is challenging, since image-level static features are typically not annotated in temporal datasets and may only occur in a subset of images within a sequence. For instance, as shown in Figure 1, image-level annotations for trees are not available, and trees are only visible in a portion of the sequence; this complicates the process of discovering the association between the static feature and observed prediction errors.

We now introduce TROVE, which identifies static feature biases learned by temporal VLMs. In line with the criteria described above, TROVE aims to identify static features that both contribute directly to downstream prediction errors and represent learned model biases. Given a pretrained VLM F and a validation dataset $\mathcal{D}_V$ associated with a downstream temporal understanding task, TROVE operates by (1) extracting candidate static features that occur consistently throughout sequences in $\mathcal{D}_V$ and (2) scoring each feature by both its effect on prediction errors made by VLM F and the extent to which the feature represents a learned bias by VLM F . Importantly, our proposed approach does not require

image-level static feature annotations and can operate effectively even when static features occur in only a subset of the sequence.

Extracting candidate static features. Given the labeled validation dataset $\mathcal{D}_V$, the first step in our approach is to extract candidate static features. To this end, we begin by retrieving all multi-image sequences S_i in $\mathcal{D}_V$. Since our goal is to identify static feature biases that manifest at the image level, we compute image-level embeddings for each image I_i contained within input sequence $S_i = (I_i^1, I_i^2, \dots, I_i^{n_i})$. To generate an image-level embedding for I_i , we create a new *static sequence* consisting of the single image I_i replicated n_i times, effectively removing all temporal variation; we then use the vision encoder of VLM F to compute an embedding for this sequence.

In order to identify static features that occur consistently within dataset $\mathcal{D}_V$, we cluster the computed image-level embeddings using spherical K-means with cosine distance. The optimal number of clusters is selected automatically by sweeping across a range of potential values and selecting the number that maximizes the Silhouette score. At the end of this step, we obtain a collection of clusters $\mathcal{C}$, where each cluster $C \in \mathcal{C}$ represents a set of images with a shared feature; for instance, in the example from Figure 2, one cluster in $\mathcal{C}$ may consist of frames with prominently-visible trees.

Discovering error-inducing static feature biases. The second step in our approach is to determine the extent to which a candidate feature represented by cluster C both (i) contributes to prediction errors and (ii) represents a static bias learned by temporal VLM F . To this end, we introduce a two-pronged scoring function designed to characterize each of these factors. First, for a given cluster C , the *error contribution score* (ECS) evaluates whether VLM F makes systematic prediction errors on one or more classes when static features associated with C are present. Second, the *static bias score* (SBS) evaluates whether VLM F has learned a bias associated with static features in C ; this involves determining the extent to which model F relies on static features in C when making predictions. We discuss these components in detail below.

For each cluster $C \in \mathcal{C}$ representing a candidate static feature, we first identify all multi-image sequences $S_i \in \mathcal{D}_V$ with at least one constituent image in cluster C . Let $\mathcal{Y}_c$ represent the set of ground-truth class labels associated with these sequences. For cluster C and class label $y \in \mathcal{Y}_c$, we compute the error contribution score ECS_C^y as follows:

$$ECS_C^y = acc_{-C}^y - acc_C^y \quad (1)$$

Here, acc_C^y represents classification accuracy on all multi-image sequences $S_i \in \mathcal{D}_V$ with at least one constituent image in cluster C and ground-truth label y . Conversely, acc_{-C}^y represents classification accuracy on all multi-image sequences $S_i \in \mathcal{D}_V$ with no constituent images in cluster C and ground-truth label y . The error contribution score ECS_C^y ranges between -1 and 1. Large positive values of ECS_C^y suggest that when static features associated with C are present in a sequence, the VLM F is likely to demonstrate degraded performance on class label y .

Next, for cluster C and class label $y \in \mathcal{Y}_c$, we compute the static bias score SBS_C^y . Let $\hat{y}_i$ refer to the label predicted by model F for an input sequence $S_i \in \mathcal{D}_V$. We first filter cluster C to retain only images I_i where the corresponding sequence S_i (1) has ground-truth label y and (2) is mispredicted (i.e. $\hat{y}_i \neq y$). We refer to this set as $C_{wrong} \subseteq C$. We then use VLM F to classify each image $I_i \in C_{wrong}$ using the full label set $\mathcal{Y}$; in order to do so, we utilize the same procedure discussed previously, where we provide a static sequence consisting of image I_i repeated n_i times as input to the vision encoder associated with model F . Our insight here is that the downstream temporal classification task, by definition, requires a dynamic sequence with visual changes in order to be successfully solved. As a result, a model that generates high-confidence predictions when provided with only a static, unchanging sequence as input is likely relying on a learned static bias. Based on this insight, we compute the static bias score:

$$SBS_C^y = \frac{1}{|C_{wrong}|} \sum_{I_i \in C_{wrong}} \text{softmax}(F([I_i, I_i, \dots, I_i]))_{\hat{y}_i} \quad (2)$$

The static bias score ranges between 0 and 1, with large values of SBS_C^y suggesting that model F has learned to rely on the static feature when making predictions. We also calibrate model confidences via temperature scaling prior to computing the static bias score [41].

Finally, for each cluster C and label y , we compute a sum of the error contribution score and the static bias score as follows: $ECS_C^y + SBS_C^y$. This quantity, which we refer to as the TROVE score,

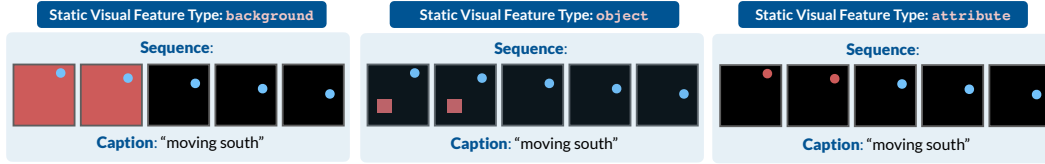


Figure 3: Example five-image sequences with injected static visual features. Our evaluation framework considers three types of static visual features - background, object, and attribute - that can contribute to biased model predictions on temporal understanding tasks.

can be used to measure the extent to which each static feature (i) contributes to prediction errors and (ii) represents a static bias learned by VLM F . An ablation on the role of the two components of the TROVE score is provided in Appendix B.

4 Evaluating TROVE in Synthetic Settings

Evaluating TROVE is complicated by the fact that ground-truth biases learned by a VLM are typically unknown a priori; thus, it is challenging to assess whether discovered biases are indeed accurate. In order to address this challenge, we introduce a large-scale quantitative evaluation framework that leverages synthetic data. Our approach is motivated by prior works [24, 27, 42] yet introduces a novel setup that focuses on temporal settings with multi-image sequence inputs. In Section 4.1, we discuss details related to the construction of our evaluation framework. Then, in Section 4.2, we demonstrate quantitatively that TROVE can effectively surface error-inducing static feature biases learned by VLMs, achieving a 28.6% improvement over the closest baseline.

We emphasize here that the use of synthetic data provides several key advantages, chief among them the ability to perform large-scale evaluations (we consider 101 temporal VLMs in this analysis) as well as support for precisely controlling key parameters of the input dataset. We follow up our synthetic evaluations with additional analyses on real-world settings in Section 5.

4.1 Designing an Evaluation Framework

In this section, we introduce our approach for quantitatively evaluating TROVE. Our insight is to predefine a static feature b ; then, we train a temporal VLM such that it learns a bias with respect to b , resulting in classification errors at inference time on class label $\tilde{y}$. This approach allows us to pair trained VLMs with ground-truth annotations for error-inducing static feature biases b and associated class labels $\tilde{y}$. Consequently, we can evaluate TROVE by measuring its ability to identify biases that align with the ground-truth annotations. Given this setup, we design a suite of evaluation configurations, with each configuration consisting of the following components:

1. **A vision-language training dataset with an injected predefined static feature bias.** The vision-language training dataset $\mathcal{D} = \{(S_i, T_i)\}_{i=1}^m$ consists of multi-image sequences S_i paired with textual captions T_i . We construct sequences using synthetic images, where each constituent image in S_i depicts a blue circle placed on a black background. The sequence is paired with a textual caption T_i , indicating the direction of movement of the circle across the sequence; namely, the circle may be “moving north”, “moving south”, “moving west”, or “moving east”. Given this setup for dataset $\mathcal{D}$, we then select one of the following static visual features: (i) background features, where all pixels located outside the circle are colored red, (ii) object features, where a red rectangle is inserted at a random position in the image, and (iii) attribute features, where the color of the circle is changed to red. We then insert the selected static visual feature of interest into dataset $\mathcal{D}$ such that the feature is highly prevalent in sequences where the circle is “moving south” and low in prevalence otherwise; this procedure injects a bias into the training dataset, contributing to errors at inference time when the static feature appears in sequences from other classes. Examples of sequences with injected static visual features are provided in Figure 3.
2. **A temporal VLM trained on this dataset.** We train a temporal VLM F using training dataset $\mathcal{D}$. Since the dataset exhibits a strong bias with respect to the predefined static feature, the model is likely to pick up on this shortcut during training.

Table 1: TROVE reliably demonstrates strong performance across all three feature categories.

Method	Background				Object				Attribute			
	P@10	P@25	P@100	R-Prec	P@10	P@25	P@100	R-Prec	P@10	P@25	P@100	R-Prec
Random	20.7	19.6	18.8	18.3	14.3	15.0	16.3	16.2	16.2	19.1	19.4	18.8
Domino	48.6	47.0	32.4	19.7	52.2	47.1	34.8	23.1	63.1	57.8	35.1	19.1
George	45.5	44.7	42.0	35.6	45.0	41.1	44.8	35.7	46.2	46.8	44.4	38.9
Dist. Failures	62.9	62.0	59.9	49.0	60.4	58.3	55.9	49.9	69.2	68.0	66.4	62.3
Confidence	61.0	60.6	64.6	59.5	97.8	97.0	95.6	78.0	58.5	59.4	57.9	52.5
TROVE	100.0	100.0	100.0	95.0	97.8	97.8	97.8	92.8	100.0	100.0	99.3	89.3

3. **A downstream temporal understanding task.** We establish a downstream temporal understanding task that aligns closely with the training data; namely, given a multi-image sequence depicting blue circles in each image, the task involves classifying the motion of the circle as one of four classes: moving north, moving south, moving east, and moving west.

We create a large suite of evaluation configurations by varying the following hyperparameters: (i) the type of static visual feature (background, object, or attribute), (ii) the sequence length n_i , (iii) the prevalence of sequences in the training dataset with the static feature, and (iv) the number of images per sequence displaying the static feature. We then verify the quality of each configuration by evaluating (i) the suitability of the proposed task and (ii) the suitability of the trained VLM. After the quality verification stage, our framework yields a total of 101 temporal VLMs paired with ground-truth annotations indicating the predefined static feature bias b and the downstream class label $\tilde{y}$ on which the bias induces errors. Additional details are provided in Appendix A.

4.2 TROVE Accurately Discovers Error-Inducing Biases in Synthetic Settings

We now evaluate TROVE using the framework from Section 4.1. We provide the trained VLM F and dataset $\mathcal{D}_V$ as input. Then, for each class label in $\mathcal{D}_V$, TROVE outputs a list of image clusters ranked by TROVE scores; each cluster represents an identified error-inducing static feature bias.

Recall from Section 4.1 that our framework annotates VLM F with both the ground-truth static feature bias b (namely, the red background, red rectangle, or red circle) and the downstream class label $\tilde{y}$ on which the bias induces errors. In order to score the output of TROVE, we compute Precision@K, defined as the proportion of the top-K images in the generated ranked list for class $\tilde{y}$ that depict b . In line with prior works on error discovery [24, 27], large Precision@K values suggest that a human user can easily understand the TROVE-identified bias by simply inspecting the top-K returned images.

We compare TROVE with five methods for systematic error detection. Three state-of-the-art approaches for systematic error discovery in non-temporal settings are considered: Domino [24], George [26], and Distilling Failures [25]. Since these methods were designed for non-temporal settings, each input sequence is represented as a single unit; thus, these methods generate ranked lists of *sequences* as output rather than ranked lists of images. We naively adapt these methods to the temporal setting by first generating a ranking of sequences and then sorting images from each sequence in temporal order. In addition to these methods, we also compare TROVE with a previously-developed temporal approach that we refer to as Confidence. Confidence, which is an application of the method proposed in Li et al. [16], ranks images from sequences in $\mathcal{D}_V$ by their maximum image-level prediction confidence. Finally, we consider a random baseline, where we pool together images from all sequences in $\mathcal{D}_V$ and then generate a random ordering.

Results are summarized in Table 1, where we report Precision@K for $K = 10, 25, 100$ as well as R-precision, a variant of Precision@K where K is equal to the total number of images in $\mathcal{D}_V$ annotated with the ground-truth static bias. We demonstrate that TROVE outperforms all other evaluated methods, achieving a 28.6% improvement over the closest baseline (namely, confidence). Existing non-temporal systematic error detection methods (Domino, George, and Distilling Failures) demonstrate low performance when directly extended to the temporal setting due to their inability to retrieve the specific images containing the static feature from a sequence. Across all three static feature categories explored in our framework (background, object, and attribute), TROVE demonstrates superior performance compared to the other methods. We note that whereas baselines exhibit significant variations in performance across the three static feature categories, TROVE consistently achieves strong performance. Extended results and ablations are provided in Appendix B.

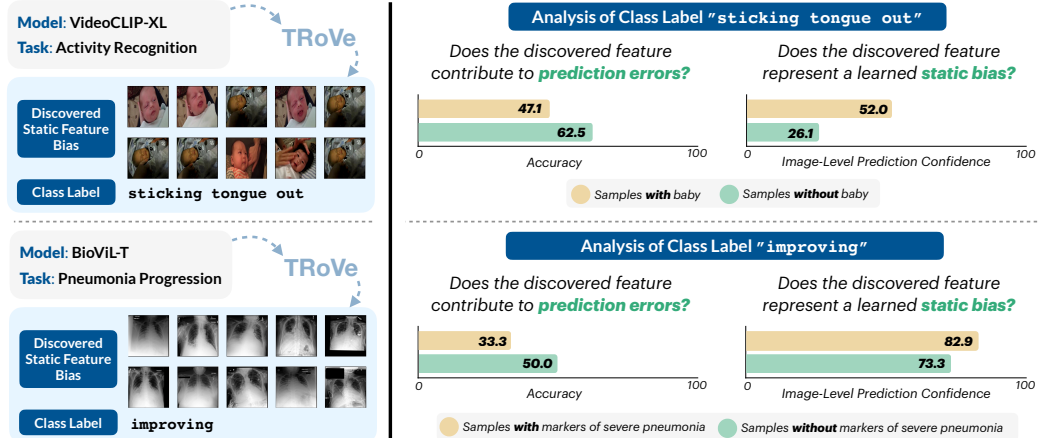


Figure 4: [Left Panel] Qualitative examples of static feature biases discovered by TROVE across various temporal VLMs and downstream tasks. [Right Panel] We demonstrate that TROVE-discovered biases satisfy desired properties.

5 Evaluating TROVE in Real-World Settings

Given the strong performance of TROVE on our synthetic experimental settings, we now utilize TROVE to discover error-inducing static feature biases learned by real-world temporal VLMs. As discussed in Section 4, evaluating the accuracy of discovered biases in real-world settings is challenging. Consequently, we validate TROVE-identified biases in two ways. First, in Section 5.1, we utilize image-level pseudolabels to verify that static feature biases surfaced by TROVE exhibit desired properties. Second, in Section 5.2, we demonstrate that knowledge of TROVE-discovered features can aid with mitigating prediction errors at test time, yielding substantial performance improvements without significant model training or data augmentation. Extended implementation details and results are in Appendix C and D.

5.1 TROVE Accurately Discovers Error-Inducing Biases in Real-World Settings

Analyzing Pretrained VLMs with TROVE: We analyze a suite of pretrained contrastive VLMs with temporal understanding capabilities [12, 11, 8, 43] across two temporal understanding tasks - 400-class activity recognition on Kinetics400 and 2-class pneumonia progression classification on MS-CXR-T [40, 8]. For the six pretrained VLMs evaluated on activity recognition, TROVE identifies between 36 and 116 learned static feature biases per model. For the one pretrained VLM evaluated on pneumonia progression classification, TROVE identifies 4 learned static feature biases.

In Figure 4 (left panel), we provide examples of static feature biases and associated class labels surfaced by TROVE. For the VideoCLIP-XL model, TROVE surfaces a feature cluster consisting of babies; this suggests that when static features associated with babies are present in a sequence, VideoCLIP-XL is likely to exhibit degraded performance on the class *sticking tongue out*. Similarly, on a pneumonia disease progression classification task, TROVE discovers a cluster of chest X-rays depicting features such as bilateral opacities, medical devices, and low lung volumes, which are indicative of severe pneumonia. This suggests that when a chest X-ray in a multi-image sequence depicts such features, BioViL-T is likely to exhibit degraded performance on the class *improving* due to a learned static feature bias. We provide additional qualitative examples in Figure 8.

Validating Discovered Biases: In Figure 4 (right panel), we validate the accuracy of biases discovered by TROVE. For the activity recognition task, we utilize an open-vocabulary object detector [44] to annotate the presence of babies in all constituent images for sequences with class label *sticking tongue out*. We find that (i) classification accuracy of VideoCLIP-XL is significantly lower on this class label (by 15.4 points) when babies are present, and (ii) VideoCLIP-XL demonstrates high prediction confidence when classifying static sequences with babies, suggesting a learned bias. Predicted labels for incorrectly-classified sequences in this class include *baby waking up* and *carrying baby*, further corroborating our finding that the model is focusing on the presence of the

baby rather than the true dynamic visual changes associated with the action sticking tongue out. Similarly, for the pneumonia progression classification task, we utilize radiology reports as well as a domain-specific entity extraction tool [45] in order to annotate each image with the presence of markers associated with severe pneumonia (i.e. bilateral lung involvement, devices, and low lung volumes). We find that (i) classification accuracy of BioViL-T is lower (by 16.7 points) when these markers are present, and (ii) BioViL-T demonstrates high prediction confidence when classifying static sequences with these markers. Additional validation is provided in Appendix D.

Comparing Temporal and Non-Temporal VLMs: Non-temporal VLMs (e.g. CLIP [1]) can be applied to temporal tasks by encoding each sequence as the mean of its constituent image-level embeddings. Intuitively, since non-temporal VLMs are trained only on static image-level features, we should expect to see high rates of static feature biases. Indeed, we find that this intuition holds. Across four non-temporal VLMs [1, 46] evaluated on activity recognition, TROVE discovers an average of 134.5 ± 38.8 static feature biases per model. This is considerably larger than the six temporal VLMs [12, 11, 43], which exhibit an average of 84.5 ± 27.3 static feature biases per model.

5.2 TROVE Improves Downstream VLM Classification Performance

We now demonstrate that knowledge of TROVE-identified static feature biases can aid with mitigating prediction errors on downstream tasks. We specifically consider contrastive temporal VLMs as a case study, which have demonstrated state-of-the-art performance on many temporal understanding tasks [12, 11].

We first run TROVE on a validation dataset $\mathcal{D}_V$, which generates as output a ranked list of image clusters (representing learned static feature biases) and associated class labels (on which the presence of the static feature induces errors). Let C represent an identified image cluster, such as the cluster of trees in Figure 2, and let $\tilde{y}$ represent the associated error-prone class label, such as the label swinging on something in Figure 2. Due to the learned bias, sequences with the static feature represented by C are particularly difficult for the VLM to correctly classify.

Prior works in non-temporal settings have suggested that VLM classification accuracy can be improved by injecting text prompts with additional fine-grained detail in order to maximize class-level separation [47, 48]. We aim to improve VLM performance on sequences with feature C by leveraging CoOp, an approach for automatically learning effective prompts [49]. We use CoOp to learn prompts that achieve the best possible classification accuracy among sequences in $\mathcal{D}_V$ with at least one image in cluster C . All parameters in the VLM are frozen, avoiding significant training costs. At test time, given an input sequence with an unknown label, we first use the trained clustering model from Section 3 to determine if the sequence contains at least one image in cluster C . If so, we use the learned prompts to perform classification; otherwise, we use default prompts.

We apply our mitigation approach to improve the performance of three contrastive temporal VLMs on an activity recognition task (Kinetics400 [40]). In Table 2, we report classification performance (Accuracy@5) across test set sequences with at least one image assigned to the top-20 TROVE-identified static feature clusters. Across this set (denoted as ‘‘Overall’’ in Table 2), we observe strong performance improvements when applying our mitigation approach. Notably, on sequences in this set with ground-truth labels $\tilde{y}$ that are particularly impacted by static feature biases (denoted as ‘‘Label $\tilde{y}$ ’’ in Table 2), we observe performance improvements of up to 111%. We note that the ‘‘Label $\tilde{y}$ ’’ and ‘‘Overall’’ categories in Table 2 are analogous to worst-group and average analyses performed in robustness literature. Our results show that knowledge of TROVE-identified biases can aid in improving test-time VLM performance by correcting errors induced by learned static feature biases.

Table 2: We show that classification accuracy of VLMs can be improved given knowledge of TROVE-identified static feature biases. This table reports performance (Accuracy@5) on a subset of videos in Kinetics400 [40] containing the top 20 static features identified by TROVE.

Model	Label $\tilde{y}$	Overall
VideoCLIP-XL	51.7	82.2
+ TROVE	94.4	86.7
ViCLIP-B	45.3	73.4
+ TROVE	95.8	77.7
ViCLIP-L	71.4	77.1
+ TROVE	96.9	80.7

6 Conclusion

In this work, we introduced TROVE, an automated approach for improving robustness of temporal VLMs. Given a temporal VLM, TROVE discovers learned static feature biases that contribute to prediction errors on downstream tasks. Ultimately, our work can help enable users to discover and mitigate important failure modes in temporal VLMs prior to deployment in real-world settings.

Acknowledgments and Disclosure of Funding

MV is supported by graduate fellowship awards from the Knight-Hennessy Scholars program at Stanford University, the Quad program, and the United States Department of Defense (NDSEG). AC is supported by NIH grants R01 HL167974, R01HL169345, R01 AR077604, R01 EB002524, R01 AR079431, P41 EB027060, AY2 AX000045, and 1AYS AX0000024-01; ARPA-H grants AY2AX000045 and 1AYSAX0000024-01; and NIH contracts 75N92020C00008 and 75N92020C00021. AC has provided consulting services to Patient Square Capital, Chondrometrics GmbH, and Elucid Bioimaging; is co-founder of Cognita; has equity interest in Cognita, Subtle Medical, LVIS Corp, Brain Key. CL is supported by NIH grants R01 HL155410, R01 HL157235, by AHRQ grant R18HS026886, and by the Gordon and Betty Moore Foundation. CL is also supported by the Medical Imaging and Data Resource Center (MIDRC), which is funded by the National Institute of Biomedical Imaging and Bioengineering (NIBIB) under contract 75N92020C00021 and through the Advanced Research Projects Agency for Health (ARPA-H).

This research was funded, in part, by the Advanced Research Projects Agency for Health (ARPA-H). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government.

References

- [1] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR, 18–24 Jul 2021. URL <https://proceedings.mlr.press/v139/radford21a.html>.
- [2] Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 4904–4916. PMLR, 18–24 Jul 2021. URL <https://proceedings.mlr.press/v139/jia21b.html>.
- [3] Sheng Zhang, Yanbo Xu, Naoto Usuyama, Jaspreet Bagga, Robert Tinn, Sam Preston, Rajesh Rao, Mu Wei, Naveen Valluri, Cliff Wong, Matthew P. Lungren, Tristan Naumann, and Hoifung Poon. Large-scale domain-specific pretraining for biomedical vision-language processing, 2023. URL <https://arxiv.org/abs/2303.00915>.
- [4] Maya Varma, Jean-Benoit Delbrouck, Sarah Hooper, Akshay Chaudhari, and Curtis Langlotz. Villa: Fine-grained vision-language representation learning from real-world data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023.
- [5] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation, 2022. URL <https://arxiv.org/abs/2201.12086>.
- [6] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning, 2023. URL <https://arxiv.org/abs/2304.08485>.
- [7] Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. Qwen2-vl: Enhancing vision-language model’s perception of the world at any resolution, 2024. URL <https://arxiv.org/abs/2409.12191>.
- [8] Shruthi Bannur, Stephanie Hyland, Qianchu Liu, Fernando Pérez-García, Maximilian Ilse, Daniel C. Castro, Benedikt Boecking, Harshita Sharma, Kenza Bouzid, Anja Thieme, Anton Schwaighofer, Maria Wetscherek, Matthew P. Lungren, Aditya Nori, Javier Alvarez-Valle, and Ozan Oktay. Learning to exploit temporal structure for biomedical vision-language processing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15016–15027, June 2023.
- [9] Zhihong Chen, Maya Varma, Jean-Benoit Delbrouck, Magdalini Paschali, Louis Blankemeier, Dave Van Veen, Jeya Maria Jose Valanarasu, Alaa Youssef, Joseph Paul Cohen, Eduardo Pontes Reis, Emily B. Tsai, Andrew Johnston, Cameron Olsen, Tanishq Mathew Abraham, Sergios Gatidis, Akshay S. Chaudhari, and Curtis Langlotz. Chexagent: Towards a foundation model for chest x-ray interpretation, 2024. URL <https://arxiv.org/abs/2401.12208>.
- [10] Shruthi Bannur, Kenza Bouzid, Daniel C. Castro, Anton Schwaighofer, Anja Thieme, Sam Bond-Taylor, Maximilian Ilse, Fernando Pérez-García, Valentina Salvatelli, Harshita Sharma, Felix Meissen, Mercy Ranjit, Shaury Srivastav, Julia Gong, Noel C. F. Codella, Fabian Falck, Ozan Oktay, Matthew P. Lungren, Maria Teodora Wetscherek, Javier Alvarez-Valle, and Stephanie L. Hyland. Maira-2: Grounded radiology report generation, 2024. URL <https://arxiv.org/abs/2406.04449>.
- [11] Yi Wang, Kunchang Li, Xinhao Li, Jiashuo Yu, Yinan He, Guo Chen, Baoqi Pei, Rongkun Zheng, Zun Wang, Yansong Shi, Tianxiang Jiang, Songze Li, Jilan Xu, Hongjie Zhang, Yifei Huang, Yu Qiao, Yali Wang, and Limin Wang. Internvideo2: Scaling foundation models for multimodal video understanding. In Aleš Leonardis, Elisa Ricci, Stefan Roth, Olga Russakovsky, Torsten Sattler, and Gül Varol, editors, *Computer Vision – ECCV 2024*, pages 396–416, Cham, 2025. Springer Nature Switzerland. ISBN 978-3-031-73013-9.

- [12] Jiapeng Wang, Chengyu Wang, Kunzhe Huang, Jun Huang, and Lianwen Jin. VideoCLIP-XL: Advancing long description understanding for video CLIP models. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 16061–16075, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.898. URL <https://aclanthology.org/2024.emnlp-main.898/>.
- [13] Yi Wang, Yinan He, Yizhuo Li, Kunchang Li, Jiashuo Yu, Xin Ma, Xinhao Li, Guo Chen, Xinyuan Chen, Yaohui Wang, et al. Internvid: A large-scale video-text dataset for multi-modal understanding and generation. In *The Twelfth International Conference on Learning Representations*, 2023.
- [14] Hang Zhang, Xin Li, and Lidong Bing. Video-llama: An instruction-tuned audio-visual language model for video understanding. *arXiv preprint arXiv:2306.02858*, 2023. URL <https://arxiv.org/abs/2306.02858>.
- [15] Kunchang Li, Yali Wang, Yinan He, Yizhuo Li, Yi Wang, Yi Liu, Zun Wang, Jilan Xu, Guo Chen, Ping Luo, Limin Wang, and Yu Qiao. Mvbench: A comprehensive multi-modal video understanding benchmark, 2024. URL <https://arxiv.org/abs/2311.17005>.
- [16] Haoxin Li, Yuan Liu, Hanwang Zhang, and Boyang Li. Mitigating and evaluating static bias of action representations in the background and the foreground. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 19911–19923, October 2023.
- [17] Shyamal Buch, Cristobal Eyzaguirre, Adrien Gaidon, Jiajun Wu, Li Fei-Fei, and Juan Carlos Niebles. Revisiting the “Video” in Video-Language Understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.
- [18] Jinpeng Wang, Yuting Gao, Ke Li, Yiqi Lin, Andy J. Ma, Hao Cheng, Pai Peng, Feiyue Huang, Rongrong Ji, and Xing Sun. Removing the background by adding the background: Towards background robust self-supervised video representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11804–11813, June 2021.
- [19] Shuangrui Ding, Maomao Li, Tianyu Yang, Rui Qian, Haohang Xu, Qingyi Chen, Jue Wang, and Hongkai Xiong. Motion-aware contrastive video representation learning via foreground-background merging. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9716–9726, June 2022.
- [20] Jie Lei, Tamara Berg, and Mohit Bansal. Revealing single frame bias for video-and-language learning. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 487–507, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.29. URL <https://aclanthology.org/2023.acl-long.29/>.
- [21] Luke Oakden-Rayner, Jared Dunnmon, Gustavo Carneiro, and Christopher Ré. Hidden stratification causes clinically meaningful failures in machine learning for medical imaging. September 2019.
- [22] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. *International Journal of Computer Vision*, 128(2):336–359, October 2019. ISSN 1573-1405. doi: 10.1007/s11263-019-01228-7. URL <http://dx.doi.org/10.1007/s11263-019-01228-7>.
- [23] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks, 2013. URL <https://arxiv.org/abs/1311.2901>.
- [24] Sabri Eyuboglu, Maya Varma, Khaled Kamal Saab, Jean-Benoit Delbrouck, Christopher Lee-Messer, Jared Dunnmon, James Zou, and Christopher Re. Domino: Discovering systematic errors with cross-modal embeddings. In *International Conference on Learning Representations*, 2022. URL <https://openreview.net/forum?id=FPCMqjIOjXN>.

- [25] Saachi Jain, Hannah Lawrence, Ankur Moitra, and Aleksander Madry. Distilling model failures as directions in latent space. In *The Eleventh International Conference on Learning Representations*, 2023. URL <https://openreview.net/forum?id=99RpBVpLiX>.
- [26] Nimit Sohoni, Jared Dunnmon, Geoffrey Angus, Albert Gu, and Christopher Ré. No subclass left behind: Fine-grained robustness in coarse-grained classification problems. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 19339–19352. Curran Associates, Inc., 2020. URL <https://proceedings.neurips.cc/paper/2020/file/e0688d13958a19e087e123148555e4b4-Paper.pdf>.
- [27] Maya Varma, Jean-Benoit Delbrouck, Zhihong Chen, Akshay Chaudhari, and Curtis Langlotz. Ravl: Discovering and mitigating spurious correlations in fine-tuned vision-language models, 2024. URL <https://arxiv.org/abs/2411.04097>.
- [28] Christoph Feichtenhofer, Axel Pinz, Richard P Wildes, and Andrew Zisserman. Deep insights into convolutional networks for video recognition. *Int. J. Comput. Vis.*, 128(2):420–437, February 2020.
- [29] Younghyun Kim, Sangwoo Mo, Minkyu Kim, Kyungmin Lee, Jaeho Lee, and Jinwoo Shin. Discovering and mitigating visual biases through keyword explanation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11082–11092, June 2024.
- [30] Rakesh R Menon and Shashank Srivastava. DISCERN: Decoding systematic errors in natural language for text classifiers. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 19565–19583, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.1091. URL <https://aclanthology.org/2024.emnlp-main.1091/>.
- [31] Greg d’Eon, Jason d’Eon, James R. Wright, and Kevin Leyton-Brown. The spotlight: A general method for discovering systematic errors in deep learning models, 2021. URL <https://arxiv.org/abs/2107.00758>.
- [32] Nari Johnson, Ángel Alexander Cabrera, Gregory Plumb, and Ameet Talwalkar. Where does my model underperform? a human evaluation of slice discovery algorithms, 2023.
- [33] Yu Yang, Besmira Nushi, Hamid Palangi, and Baharan Mirzasoleiman. Mitigating spurious correlations in multi-modal models during fine-tuning. In *International Conference on Machine Learning*, 2023.
- [34] Yuanxin Liu, Shicheng Li, Yi Liu, Yuxiang Wang, Shuhuai Ren, Lei Li, Sishuo Chen, Xu Sun, and Lu Hou. Tempcompass: Do video llms really understand videos?, 2024. URL <https://arxiv.org/abs/2403.00476>.
- [35] Takumi Fukuzawa, Kensho Hara, Hirokatsu Kataoka, and Toru Tamaki. *Can Masking Background and Object Reduce Static Bias for Zero-Shot Action Recognition?*, page 366–379. Springer Nature Singapore, 2025. ISBN 9789819620715. doi: 10.1007/978-981-96-2071-5_27. URL http://dx.doi.org/10.1007/978-981-96-2071-5_27.
- [36] Jihoon Chung, Yu Wu, and Olga Russakovsky. Enabling detailed action recognition evaluation through video dataset augmentation. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 39020–39033. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/ff52407b80dde0f45814db2738464c-Paper-Datasets_and_Benchmarks.pdf.
- [37] Jinwoo Choi, Chen Gao, Joseph C. E. Messou, and Jia-Bin Huang. Why can’t i dance in the mall? learning to mitigate scene bias in action recognition. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/ab817c9349cf9c4f6877e1894a1faa00-Paper.pdf.

- [38] De-An Huang, Vignesh Ramanathan, Dhruv Mahajan, Lorenzo Torresani, Manohar Paluri, Li Fei-Fei, and Juan Carlos Niebles. What makes a video a video: Analyzing temporal information in video understanding models and datasets. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.
- [39] Yating Yu, Congqi Cao, Yifan Zhang, and Yanning Zhang. Learning to generalize without bias for open-vocabulary action recognition, 2025. URL <https://arxiv.org/abs/2502.20158>.
- [40] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. The kinetics human action video dataset, 2017. URL <https://arxiv.org/abs/1705.06950>.
- [41] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks, 2017. URL <https://arxiv.org/abs/1706.04599>.
- [42] Weixin Liang and James Zou. Metashift: A dataset of datasets for evaluating contextual distribution shifts and training conflicts. In *International Conference on Learning Representations*, 2021.
- [43] Bolin Ni, Houwen Peng, Minghao Chen, Songyang Zhang, Gaofeng Meng, Jianlong Fu, Shiming Xiang, and Haibin Ling. Expanding language-image pretrained models for general video recognition, 2022. URL <https://arxiv.org/abs/2208.02816>.
- [44] Matthias Minderer, Alexey Gritsenko, and Neil Houlsby. Scaling open-vocabulary object detection, 2024. URL <https://arxiv.org/abs/2306.09683>.
- [45] Jean-Benoit Delbrouck, Pierre Chambon, Zhihong Chen, Maya Varma, Andrew Johnston, Louis Blankemeier, Dave Van Veen, Tan Bui, Steven Truong, and Curtis Langlotz. RadGraph-XL: A large-scale expert-annotated dataset for entity and relation extraction from radiology reports. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics: ACL 2024*, pages 12902–12915, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.765. URL <https://aclanthology.org/2024.findings-acl.765/>.
- [46] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. *arXiv preprint arXiv:2303.15343*, 2023.
- [47] Sachit Menon and Carl Vondrick. Visual classification via description from large language models, 2022. URL <https://arxiv.org/abs/2210.07183>.
- [48] Sarah Pratt, Ian Covert, Rosanne Liu, and Ali Farhadi. What does a platypus look like? generating customized prompts for zero-shot image classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15691–15701, October 2023.
- [49] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Learning to prompt for vision-language models. *International Journal of Computer Vision*, 130(9):2337–2348, July 2022. ISSN 1573-1405. doi: 10.1007/s11263-022-01653-1. URL <http://dx.doi.org/10.1007/s11263-022-01653-1>.
- [50] Huaxiu Yao, Yu Wang, Sai Li, Linjun Zhang, Weixin Liang, James Zou, and Chelsea Finn. Improving out-of-distribution robustness via selective augmentation. In *Proceeding of the Thirty-ninth International Conference on Machine Learning*, 2022.
- [51] Joao Carreira, Eric Noland, Andras Banki-Horvath, Chloe Hillier, and Andrew Zisserman. A short note about kinetics-600, 2018. URL <https://arxiv.org/abs/1808.01340>.
- [52] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzyńska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, Florian Hoppe, Christian Thureau, Ingo Bax, and Roland Memisevic. The "something something" video database for learning and evaluating visual common sense, 2017. URL <https://arxiv.org/abs/1706.04261>.

- [53] Binglu Wang, Yongqiang Zhao, Le Yang, Teng Long, and Xuelong Li. Temporal action localization in the deep learning era: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(4):2171–2190, 2024. doi: 10.1109/TPAMI.2023.3330794.

Appendix

Contents

A	Implementation Details for Synthetic Evaluations	16
B	Extended Results for Synthetic Evaluations	18
C	Implementation Details for Real-World Evaluations	18
D	Extended Results for Real-World Evaluations	21
	D.1 Analysis of Pretrained Temporal VLMs	21
	D.2 Improving Downstream Classification Performance	23
	D.3 Extending to Additional Tasks	23
E	Additional Applications of TROVE	25
F	Extended Related Work	26
G	Extended Discussion	27

A Implementation Details for Synthetic Evaluations

In this section, we expand on Section 4.1 by providing additional implementation details for our evaluation framework.

Implementation details for vision-language training datasets with injected predefined static feature biases: Sequences in the vision-language training dataset $\mathcal{D}$ are composed of synthetic images, where each image depicts a blue circle placed on a black background. To generate images, we first create an empty image of size 60 pixels $\times$ 60 pixels. We then place a blue circle with a diameter of 10 pixels in a random location on the image. The position of the blue circle will vary across the sequence; specifically, the circle will be "moving north" (towards the upper border of the image), "moving south" (towards the lower border of the image), "moving west" (towards the left border of the image), or "moving east" (towards the right border of the image). Textual captions paired with each sequence indicate the direction of the circle’s movement.

As stated in Section 4.1, we create a large suite of evaluation configurations by varying several hyperparameters associated with the training dataset. We define each configuration by selecting a single value for each hyperparameter. Hyperparameters and possible values are described in detail below:

- *Type of static visual feature.* Motivated by the types of static feature biases that emerge in real-world settings [16], we consider three categories of static features: (i) background features, where all pixels outside the circle are colored red, (ii) object features, where a red rectangle with dimensions 15 pixels $\times$ 15 pixels is inserted in a random location, and (iii) attribute features, where the color of the circle is changed to red. For object features, we ensure that the red rectangle does not overlap with the blue circle when placed in the image. By design, these three categories of static features vary in visual subtlety, with background features resulting in the most pixel-level changes and attribute features the least.
- *Sequence length.* We consider four options for the sequence length n_i : 2 images, 3 images, 5 images, and 10 images.
- *Prevalence of sequences in the training set with the static feature.* In line with prior work [50], we use the Cramer’s V metric to ensure that the presence of the static feature in the training set is

strongly associated with the group of sequences in which the circle is "moving south". We consider the following values of Cramer’s V: 0.7, 0.8, 0.9, and 0.95.

- *Number of images per sequence displaying the static feature.* We select a nonzero integer value v_i less than or equal to the value of n_i . When injecting the static feature into a selected sequence in $\mathcal{D}$, we randomly select a contiguous subsequence of v_i images to depict the feature. For example, in Figure 3, $v_i = 2$ and $n_i = 5$.

Implementation details for trained temporal VLMs: As part of our evaluation framework, we train a temporal VLM F using training dataset D . Model F is implemented in the form of a simple contrastive VLM where the vision and text encoders are based on the CLIP ViT-L/14 architecture. For each input sequence S_i , constituent images are passed through the vision encoder followed by a trainable projection head consisting of two linear layers interspersed with a ReLU activation. Our architecture includes a total of n_i projection heads, and the appropriate projection head for each image is selected based on its position in the sequence. We assume that n_i remains constant for all sequences in the dataset. The resulting embeddings for constituent images are concatenated together (in order to preserve temporal information) and then passed through a projection head consisting of three linear layers interspersed with ReLU activations. The output of this projection head is a single embedding characterizing the sequence.

Training is performed on a single NVIDIA V100 GPU using a batch size of 256, an initial learning rate of $1e-4$, and a total of 100 epochs with early stopping based on validation set performance. All parameters associated with the vision and text encoder remain frozen, whereas parameters associated with the projection heads are learnable. At inference time, classification is performed by computing the cosine similarity between the sequence-level embedding and text embeddings associated with each class; the class with the highest cosine similarity is selected as the prediction.

Implementation details for downstream temporal understanding tasks: The downstream temporal understanding task takes the form of a classification task in which the motion of the circle must be classified in one of four categories: moving north, moving south, moving east, and moving west. In order to reflect real-world settings, we assume that the dataset $\mathcal{D}_V$ used in the downstream task is drawn from the same distribution as the training dataset, where the injected static feature is highly prevalent for the class “moving south” and less prevalent on other classes. For each evaluation configuration in our suite, we use the same instantiations of hyperparameters for both the training dataset and the downstream task dataset.

Quality verification: We verify the quality of each evaluation configuration across two axes: (i) the suitability of the proposed task and (ii) the suitability of the trained VLM.

First, in order to evaluate the suitability of the proposed task, we train a temporal VLM on a version of the training dataset with no inserted static feature, and we verify that the downstream inference task can be successfully solved by this model. We also train a standard, non-temporal VLM on this dataset to perform the task using only a single selected image per sequence, and we verify that the downstream task cannot be solved by this model; here, we adopt the approach introduced by Buch et al. [17]. In combination, this analysis confirms that in an unbiased setting, the task can only be addressed by a temporal VLM capable of parsing visual changes across multiple images. In practice, we retain evaluation configurations where the temporal, unbiased VLM exhibits at least a 20 point improvement over the non-temporal, unbiased VLM; the remainder are filtered out.

Second, in order to evaluate the suitability of the trained VLM, we verify that (i) the VLM has learned the predefined static feature bias and (ii) the presence of the static feature contributes to mispredictions. To evaluate whether the VLM has learned the predefined static feature bias, we compute the difference in image-level classification performance between images without the predefined static feature and images with the predefined static feature. We retain evaluation settings where the gap in performance is greater than 20 points on at least one class $\tilde{y}$; the remainder are filtered out. To evaluate whether the presence of the static feature bias contributes to mispredictions, we compute the difference in sequence-level classification performance between sequences without the predefined static feature and sequences with the predefined static feature. Again, we retain all evaluation configurations where the gap in performance is greater than 20 points on class $\tilde{y}$. In combination, this analysis confirms that the trained VLM has learned the static feature bias of interest and that the static feature bias contributes to errors on at least one class associated with the downstream task.

Summary statistics: After the quality verification stage, our framework yields a total of 101 valid evaluation configurations, each consisting of a temporal VLM paired with ground-truth annotations indicating the learned static feature bias b and the downstream class label $\tilde{y}$ on which the bias induces errors. Below, we provide a breakdown of these configurations across various factors:

- *Downstream class label $\tilde{y}$:* Out of the 101 valid evaluation configurations, the value of $\tilde{y}$ equals moving north for 17 configurations, moving west for 36 configurations, and moving east for 48 configurations.
- *Type of static visual feature:* Out of the 101 valid evaluation configurations, 42 exhibit a background static feature bias, 46 exhibit an object static feature bias, and 13 exhibit an attribute static feature bias.
- *Sequence length:* Out of the 101 valid evaluation configurations, 13 have sequences consisting of 2 images, 17 have sequences consisting of 3 images, 34 have sequences consisting of 5 images, and 37 have sequences consisting of 10 images.
- *Prevalence of sequences in the training set with the static feature:* Out of the 101 valid evaluation configurations, 1 has a Cramer’s V score of 0.7, 12 have a Cramer’s V score of 0.8, 38 have a Cramer’s V score of 0.9, and 50 have a Cramer’s V score of 0.95.
- *Proportion of images per sequence displaying the static feature:* Out of the 101 valid evaluation configurations, 25 have between 20% and 49% of the frames per sequence depicting the static feature, 38 have between 50% and 79% of the frames per sequence depicting the static feature, and 38 have between 80% and 100% of the frames per sequence depicting the static feature.

B Extended Results for Synthetic Evaluations

In this section, we extend the results provided in Section 4.2 with additional performance breakdowns. In Figure 5, we provide a breakdown of TROVE performance by the number of images per sequence. As part of our suite of 101 trained VLMs, we consider VLMs trained on datasets with varying numbers of images n_i per sequence; in particular, we consider $n_i \in \{2, 3, 5, 10\}$. As shown in Figure 5, TROVE demonstrates strong performance across all four categories. On the other hand, we see considerable declines in performance for the Confidence baseline as the number of images per sequence increases.

In Figure 6, we analyze TROVE performance with respect to the proportion of images per sequence displaying the static feature. As part of our suite of 101 trained VLMs, we consider VLMs trained on datasets with varying proportions of images per sequence containing the static visual feature; for instance, in a sequence consisting of five images, a proportion of 0.4 indicates that two images in the sequence display the static feature, as depicted in Figure 3. TROVE demonstrates strong performance across the spectrum, whereas baselines again show considerable variation. The ability of TROVE operate effectively even when the static feature is only visible in a portion of the sequence is a key advantage over the non-temporal systematic error detection methods (Domino, Distilling Failures, and George) evaluated in this work.

In Figure 7, we provide overall performance metrics across four evaluation metrics: Precision@10, Precision@25, Precision@100, and R-Precision. Across all metrics, TROVE demonstrates superior performance to baselines, demonstrating that our approach is effective at generating accurate ranked lists of identified static feature biases.

In Table 3, we provide an ablation with respect to the two components that make up the TROVE score: the Static Bias Score (SBS) and the Error Contribution Score (ECS). We demonstrate that using both components in tandem yields significantly higher performance across our synthetic evaluations than using SBS alone. We do not perform an ablation with the ECS alone, since using ECS alone will not specifically target *static feature biases*, which are the topic of this work.

C Implementation Details for Real-World Evaluations

In this section, we provide additional implementation details associated with the analysis in Section 5.

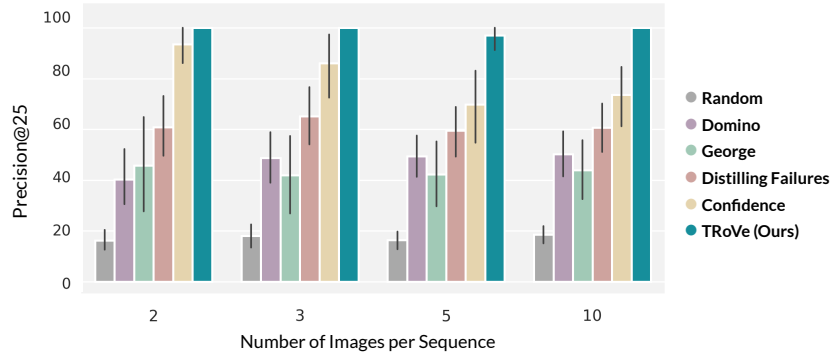


Figure 5: TRoVe demonstrates strong performance across evaluation configurations with varying numbers of images per sequence.

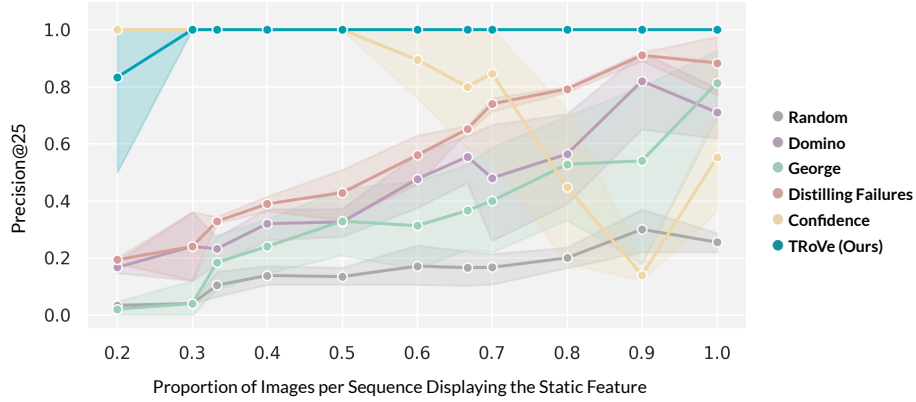


Figure 6: TRoVe consistently demonstrates strong performance regardless of the proportion of images per sequence displaying the static feature.

We utilize TRoVe to analyze seven pretrained contrastive VLMs with temporal understanding capabilities using two temporal understanding tasks: activity recognition on Kinetics400 [40] and pneumonia progression classification on MS-CXR-T [8].

Activity Recognition on Kinetics400: The activity recognition task on Kinetics400 involves classifying a video into one of 400 possible classes that represent human actions, such as `welding` or `climbing tree`. The model is presented with a sequence of video frames as input; for our analysis, we use 8 frames per video obtained via uniform sampling. We split the validation set of Kinetics400 into a development set (used for the analysis in this section) and a test set (used for the analysis in Section 5.2 where we mitigate biases). We compute logits and predictions for each model using a standard zero-shot classification approach, performed by computing the cosine similarity between the sequence embedding and text embeddings representing each class label. In line with prior work [1], we compute text embeddings by ensembling the following prompt templates for each class label: {"a photo of [LABEL].", "a photo of a person [LABEL].", "a photo of a person using [LABEL].", "a photo of a person doing [LABEL].", "a photo of a person during [LABEL].", "a photo of a person performing [LABEL].", "a photo of a person practicing [LABEL].", "a video of [LABEL].", "a video of a person [LABEL].", "a video of a person using [LABEL].", "a video of a person doing [LABEL].", "a video of a person during [LABEL].", "a video of a person performing [LABEL].", "a video of a person practicing [LABEL].", "a example of [LABEL].", "a example of a person [LABEL].", "a example of a person using [LABEL].", "a example of a person doing [LABEL].", "a example of a

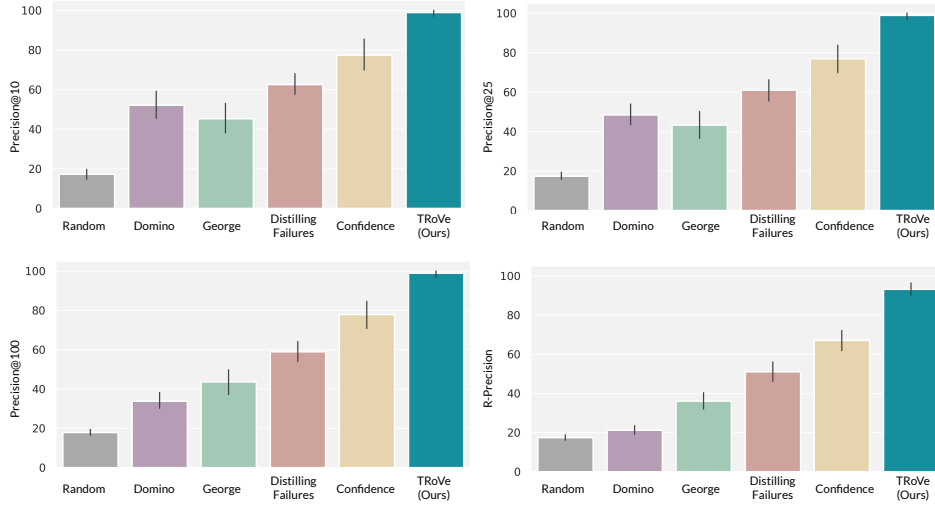


Figure 7: We compute four metrics to characterize overall performance of TROVE on our evaluation framework: Precision@10, Precision@25, Precision@100, and R-Precision.

Table 3: Ablations with respect to the two components of the TROVE score.

Method	P@10	P@25	P@100	RP
Static Bias Score (SBS) Only	76.2	76.2	75.9	78.2
Static Bias Score (SBS) & Error Contribution Score (ECS)	99.1	99.1	98.9	93.2

person during [LABEL].", "a example of a person performing [LABEL].", "a example of a person practicing [LABEL].", "a demonstration of [LABEL].", "a demonstration of a person [LABEL].", "a demonstration of a person using [LABEL].", "a demonstration of a person doing [LABEL].", "a demonstration of a person during [LABEL].", "a demonstration of a person performing [LABEL].", "a demonstration of a person practicing [LABEL].". Kinetics400 is open-source.

We analyze six contrastive VLMs on the activity recognition task (VideoCLIP-XL [12], ViCLIP-L [11], ViCLIP-B [11], XCLIP-B/16 [43], XCLIP-B/32 [43], and XCLIP-L/14 [43]). As described in Section 3, TROVE includes a clustering stage. The optimal number of clusters is selected automatically by sweeping across a range of potential values ($|\mathcal{Y}| * 2, |\mathcal{Y}| * 6$) at increments of 400; here, the bounds of the range evaluate to $[800, 2400)$, given the fact that $|\mathcal{Y}| = 400$. We classify a prediction as correct if the ground-truth label appears in the top-5 predicted classes (i.e. Accuracy@5). We exclude any identified static feature biases where (1) the error contribution score is low (defined as below a predefined threshold of 0.1) and (2) the static bias score is less than or equal to random chance (defined as $1/|\mathcal{Y}| = 0.0025$ in this case). VideoCLIP-XL is available under CC-BY-NC-SA-4.0. ViCLIP and XCLIP are available under MIT licenses. We implement TROVE using a single NVIDIA V100 GPU.

Pneumonia Progression Classification on MS-CXR-T: The pneumonia progression classification task on MS-CXR-T involves classifying a sequence of chest X-rays collected at varying timepoints into one of two possible categories: *improving*, which suggests that pneumonia is improving over the course of the sequence, and *worsening*, which suggests that pneumonia is worsening over the course of the sequence. Each sequence contains two chest X-rays. We compute logits and predictions using a standard zero-shot classification approach. In line with prior work [8], we compute text embeddings by ensembling across prompt templates. For the class label *improving*, we utilize the following prompts: {"pneumonia is better", "pneumonia is cleared", "pneumonia is decreased", "pneumonia is decreasing", "pneumonia is improved", "pneumonia is improving", "pneumonia is reduced", "pneumonia is resolved", "pneumonia is resolving", "pneumonia is smaller"}. For the

class label worsening, we utilize the following prompts: {"pneumonia is bigger", "pneumonia is developing", "pneumonia is enlarged", "pneumonia is enlarging", "pneumonia is greater", "pneumonia is growing", "pneumonia is increased", "pneumonia is increasing", "pneumonia is larger", "pneumonia is new", "pneumonia is progressing", "pneumonia is progressive", "pneumonia is worse", "pneumonia is worsened", "pneumonia is worsening"}. MS-CXR-T is available under PhysioNet Credentialed Health Data License 1.5.0.

We analyze one contrastive VLM on the pneumonia progression classification task (BioViL-T [8]). As described in Section 3, TROVe includes a clustering stage. The optimal number of clusters is selected automatically by sweeping across a range of potential values $[|\mathcal{Y}| * 2, |\mathcal{Y}| * 6)$ at increments of 1; here, the bounds of the range evaluate to $[4, 12)$, given the fact that $|\mathcal{Y}| = 2$. We exclude any identified static feature biases where (1) the error contribution score is low (defined as below a predefined threshold of 0.1) and (2) the static bias score is less than or equal to random chance (defined as $1/|\mathcal{Y}| = 0.5$ in this case). BioViL-T is available under an MIT license. We implement TROVe using a single NVIDIA V100 GPU.

D Extended Results for Real-World Evaluations

D.1 Analysis of Pretrained Temporal VLMs

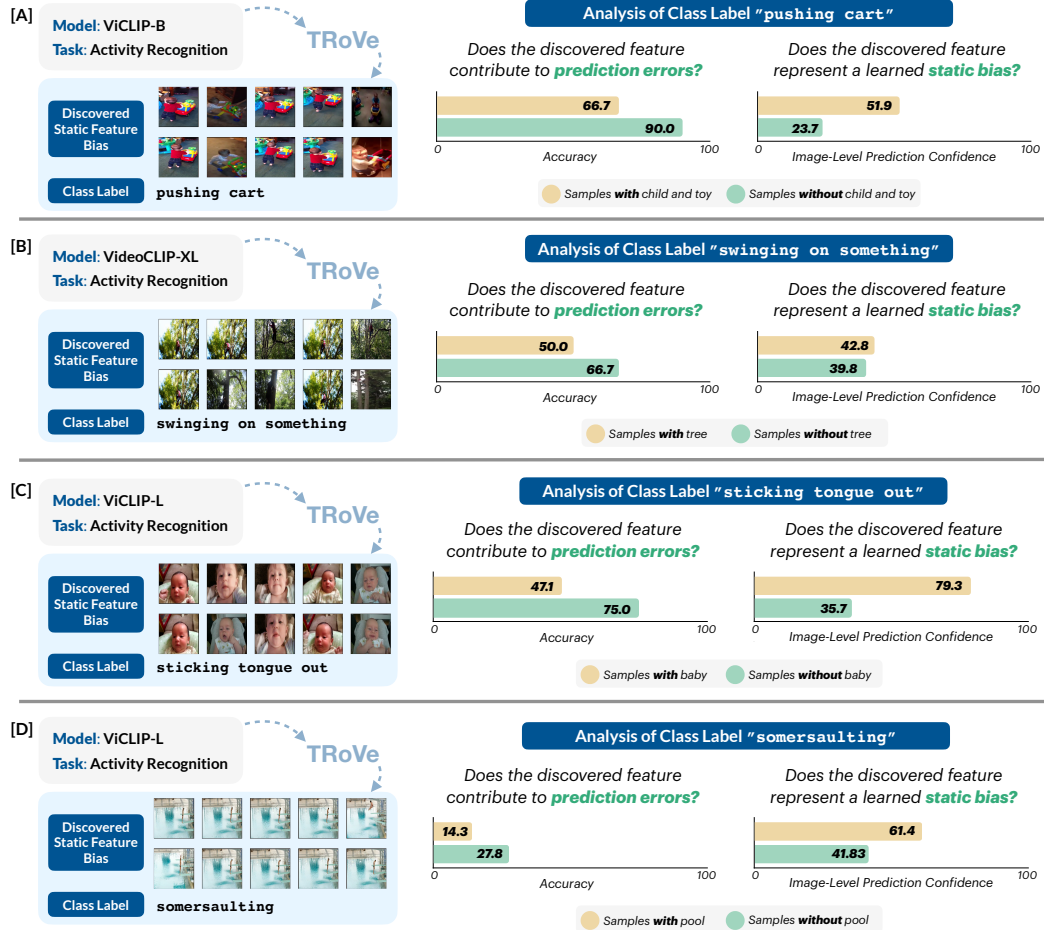


Figure 8: Additional qualitative examples of static feature biases discovered by TROVe across various temporal VLMs.

On the activity recognition task, we identify 104 static feature biases for VideoCLIP-XL, 104 static feature biases for ViCLIP-L, 116 static feature biases for XCLIP-B/16, 81 static feature biases for XCLIP-B/32, and 36 static feature biases for XCLIP-L/14. On the pneumonia progression classification task, we identify 4 static feature biases for BioViL-T. We

observe a general trend that as the size of the model increases across the same family (for example, ViCLIP-B vs. ViCLIP-L and XCLIP-B/16 vs. XCLIP-L/14), the number of identified static feature biases decreases, potentially suggesting that larger models are less reliant on static feature biases. This follows logically from the fact that larger models also exhibit better overall performance with lower error rates on the activity recognition task, suggesting that these models are more likely to have learned true dynamic patterns. We also note here that the number of identified static feature biases is strongly correlated with the size of dataset and the number of classes; as a result, the smaller number of identified static feature biases for pneumonia progression classification in comparison to activity recognition is expected.

We validate the quality of each discovered bias by verifying whether the desired properties (namely, *error-inducing* and *learned model bias*) are satisfied. Here, we provided extended details on our human-in-the-loop validation procedure introduced in Section 5.1. First, given a TROVE-discovered cluster of images C and associated error-prone class label $\tilde{y}$, a human reader annotates the key feature shared among images in cluster C (e.g. "baby" in the activity recognition example provided in Figure 4). Then, for all sequences in the dataset with class label $\tilde{y}$, we assign image-level pseudolabels indicating whether the human-annotated feature (e.g. "baby" in Figure 4) is present or absent. For the activity recognition task, image-level pseudolabels are obtained by leveraging an off-the-shelf open-vocabulary object detector (OwlV2) [44], which is prompted to detect the presence of the human-annotated feature. The feature is considered to be present in an image if the confidence of OwlV2 is at least 0.3. For the pneumonia progression classification task, we obtain image-level pseudolabels by leveraging RadGraph-XL [45], a domain-specific entity recognizer, to annotate findings present in each image. Then, given image-level pseudolabels, we evaluate whether the TROVE-identified bias satisfies the two desired properties:

- *Does the discovered feature contribute to prediction errors?* We compute classification accuracy on sequences from class label $\tilde{y}$ with the human-annotated feature as well as sequences from class label $\tilde{y}$ without the human-annotated feature. If the TROVE-discovered bias is accurate, then we would expect to see lower performance on class label $\tilde{y}$ when sequences contain the human-annotated feature.
- *Does the discovered feature represent a learned static bias?* For each constituent image per incorrectly-classified sequence, we construct a static sequence consisting of the single image repeated n_i times. We then obtain softmax-normalized prediction logits when performing classification with the static sequence, and we extract the maximum value across all classes (i.e. the maximum prediction confidence). If the TROVE-discovered bias is accurate, then we would expect to see large confidence values, suggesting that the model is able to make highly-confident predictions without the use of any temporal information. In particular, we would expect to see confidence values significantly larger than the confidence values that would be expected by random chance (i.e. $1/|\mathcal{Y}|$). In our plots in Figures 4 and 8, we provide a comparison of the mean image-level maximum confidence values between images with the human-annotated feature and images without the human-annotated feature.

Figure 8 provides additional qualitative examples of error-inducing static feature biases and associated class labels discovered by TROVE, as well as associated validation results. Below, we analyze the results in each panel:

- **[Panel A]:** TROVE discovers a cluster of images depicting children and toy carts, suggesting that when features associated with children and toy carts are present in a sequence, ViCLIP-B is likely to exhibit lower performance on the class `pushing cart`. ViCLIP-B mispredicts these samples as `crawling baby` and `crying baby`, suggesting that the model is relying on static features associated with children when making predictions. In order to validate this finding, we use OwlV2 [44] to annotate the presence of a "child with toy" in all constituent images for sequences in class `pushing cart`. We find that (i) classification accuracy of ViCLIP-B is significantly lower on this class label (23.3 points) when children and toys are present and (ii) ViCLIP-B demonstrates high prediction confidence when classifying static sequences from this class label with children and toys.
- **[Panel B]:** TROVE discovers a cluster of images consisting of trees, suggesting that when static features associated with trees are present in a sequence, VideoCLIP-XL is likely to exhibit lower performance on the class `swinging on something`. VideoCLIP-XL mispredicts these samples as `climbing tree`, suggesting that the model is relying on static features associated with trees

when making predictions. This example is highlighted in Figure 2. In order to validate this finding, we use OwlV2 [44] to annotate the presence of a "tree" in all constituent images for sequences in class `swinging on something`. We find that (i) classification accuracy of VideoCLIP-XL is significantly lower on this class label (16.7 points) when trees are present and (ii) VideoCLIP-XL demonstrates high prediction confidence when classifying static sequences from this class label with trees. We note that the difference in mean image-level maximum confidence values between images with trees and images without trees is relatively small in this example (3 points); regardless, the prediction confidence is substantially larger than what would be expected by random chance (0.25 vs. 42.8), suggesting that the model is relying on a learned static bias.

- **[Panel C]:** TROVE discovers a cluster of images consisting of babies, suggesting that when static features associated with babies are present in a sequence, ViCLIP-L is likely to exhibit lower performance on the class `sticking tongue out`. ViCLIP-L mispredicts these samples as `baby waking up`, suggesting that the model is relying on static features associated with babies when making predictions. Interestingly, this example is similar to the example provided in Figure 4 (left panel), suggesting that ViCLIP-L and VideoCLIP-XL both learned a similar error-inducing static feature bias. In order to validate this finding, we use OwlV2 [44] to annotate the presence of a "baby" in all constituent images for sequences in the class `sticking tongue out`. We find that (i) classification accuracy of ViCLIP-L is significantly lower on this class label (27.9 points) when babies are present and (ii) ViCLIP-L demonstrates high prediction confidence when classifying static sequences from this class label with babies.
- **[Panel D]:** TROVE discovers a cluster of images consisting of divers with swimming pool backgrounds, suggesting that when these static features are present in a sequence, ViCLIP-L is likely to exhibit lower performance on the class `somersaulting`. Frequent mispredictions in this set of samples are the class labels `springboard diving` and `jumping into pool`, suggesting that the model is relying on static features associated with the pool when making predictions. Importantly, we note here that although these predictions do not align with the ground-truth label (`somersaulting`), they are not necessarily incorrect in this setting; this raises the possibility of another potential use case for TROVE in real-world settings: flagging potential labeling errors. To validate the discovered bias, we use OwlV2 [44] to annotate the presence of a "pool" in all constituent images for sequences in the class `somersaulting`. We find that (i) classification accuracy of ViCLIP-L is significantly lower on this class label (13.5 points) when pools are present and (ii) ViCLIP-L demonstrates high prediction confidence when classifying static sequences from this class label with pools.

For our analysis of non-temporal VLMs on the activity recognition task, we consider four models: CLIP-ViT/32 [1], CLIP-ViT/14 [1], CLIP-RN50 [1], and SigLIP [46]. We identify 195 static feature biases for CLIP-ViT/32, 92 static feature biases for CLIP-ViT/14, 111 static feature biases for CLIP-RN50, and 140 static feature biases for SigLIP.

D.2 Improving Downstream Classification Performance

We implement CoOp [49] using the default settings provided in the original implementation. For each considered cluster C , we utilize an SGD optimizer with a learning rate of 0.002 and train for 20 epochs. In Table 2, we reported mean classification accuracy across the test set sequences containing the top-20 TROVE-identified static features. In Table 4, we extend these results by reporting mean classification performance (evaluated with Accuracy@5) across the test set sequences containing the top-5 TROVE-identified static features and the top-10 TROVE-identified static features. In Table 5, we report Accuracy@1 metrics. Table 6 lists the number of test set sequences considered under each category when computing results in Tables 2, 4, and 5. Our mitigation approach leads to consistent performance improvements with minimal computational cost, demonstrating the practical utility of TROVE-identified biases.

D.3 Extending to Additional Tasks

We now extend our analysis on activity recognition to two additional tasks: 600-class activity recognition on Kinetics600 [51] and 174-class fine-grained activity recognition on Something-Something V2 [52]. Here, we specifically consider the VideoCLIP-XL model [12].

Table 4: We show that classification accuracy of VLMs can be improved given knowledge of TROVE-identified static feature biases. This table reports performance (Accuracy@5) on a subset of videos in Kinetics400 [40] containing the top-5 TROVE-identified static features, the top-10 TROVE-identified static features, and the top-20 TROVE-identified static features.

Model	Top 5 Identified Features		Top 10 Identified Features		Top 20 Identified Features	
	Label $\tilde{y}$	Overall	Label $\tilde{y}$	Overall	Label $\tilde{y}$	Overall
VideoCLIP-XL + TROVE	47.1 82.4	84.1 89.4	63.0 91.3	82.3 84.7	51.7 94.4	82.2 86.7
ViCLIP-B + TROVE	21.1 89.5	69.5 75.9	22.7 93.2	69.0 75.5	45.3 95.8	73.4 77.7
ViCLIP-L + TROVE	58.6 96.6	69.9 78.2	64.3 96.4	75.7 78.8	71.4 96.9	77.1 80.7

Table 5: We show that classification accuracy of VLMs can be improved given knowledge of TROVE-identified static feature biases. This table reports performance (Accuracy@1) on a subset of videos in Kinetics400 [40] containing the top-5 TROVE-identified static features, the top-10 TROVE-identified static features, and the top-20 TROVE-identified static features.

Model	Top 5 Identified Features		Top 10 Identified Features		Top 20 Identified Features	
	Label $\tilde{y}$	Overall	Label $\tilde{y}$	Overall	Label $\tilde{y}$	Overall
VideoCLIP-XL + TROVE	17.6 41.2	48.3 58.5	21.7 45.7	52.7 55.8	18.0 61.8	55.0 57.1
ViCLIP-B + TROVE	5.3 15.8	34.2 43.9	6.8 50.0	37.7 47.9	24.2 62.1	43.1 50.9
ViCLIP-L + TROVE	3.4 48.3	35.2 46.1	16.1 53.6	40.9 48.2	23.5 63.3	45.6 50.1

We first analyze the performance of VideoCLIP-XL on Kinetics600 using TROVE. In this setting, TROVE identifies 149 learned static feature biases; in comparison, TROVE had identified 104 learned static feature biases on Kinetics400. Intuitively, the larger number of static feature biases discovered with Kinetics600 is expected due to the fact that Kinetics600 is an approximate superset of Kinetics400.

We find high consistency between results on Kinetics400 and Kinetics600. For instance, the top-ranked static feature identified by TROVE on Kinetics600 depicts a cluster of trees paired with the class label "swinging on something", suggesting that when static features associated with trees are present in a sequence, VideoCLIP-XL is likely to exhibit lower performance on the class "swinging on something". An identical static feature bias was identified when evaluating VideoCLIP-XL on Kinetics400, as shown in Figures 1, 2, and 8 [Panel B]. The ability of TROVE to yield consistent results across two distinct evaluation settings demonstrates its reliability. TROVE also uncovers new static feature biases not identified in Kinetics400, such as a link between static features associated with trampolines and errors on the class label "backflip (human)".

We then analyze VideoCLIP-XL on Something-Something V2. Something-Something V2 is a challenging task in zero-shot settings due to the fine-grained nature of class labels (e.g. "putting something on a surface", "pushing something from left to right", etc.). VideoCLIP-XL achieves an overall zero-shot performance (Accuracy@5) of just 18.1 on Something-Something V2. When analyzing VideoCLIP-XL with TROVE, we discover 14 error-inducing static feature biases in this setting. The low number of identified static feature biases in comparison to Kinetics400 and Kinetics600 aligns with expectations, since Something-Something V2 was specifically designed to ensure that reliance on single-frame, static content will not aid with predicting any of the classes; thus, models are unlikely to rely on static features as shortcuts on this task. Consequently, the observed low zero-shot

Table 6: Here, we list the number of test set sequences considered under each category when computing results in Tables 2, 4, and 5. The "Overall" column lists the number of test set sequences with at least one constituent image assigned to the top-K TROVE-identified static feature clusters. The "Label $\tilde{y}$ " column lists the number of test set sequences with ground-truth label $\tilde{y}$ and at least one constituent image assigned to the top-K TROVE-identified static feature clusters.

Model	Top 5 Identified Features		Top 10 Identified Features		Top 20 Identified Features	
	Label $\tilde{y}$	Overall	Label $\tilde{y}$	Overall	Label $\tilde{y}$	Overall
VideoCLIP-XL	17	207	46	419	89	669
ViCLIP-B	19	187	44	355	95	831
ViCLIP-L	29	193	56	452	98	866

Table 7: We show that classification accuracy of VLMs on Kinetics600 and Something-Something V2 can be substantially improved given knowledge of TROVE-identified static feature biases.

Model	Kinetics600		Something-Something V2	
	Label $\tilde{y}$	Overall	Label $\tilde{y}$	Overall
VideoCLIP-XL	54.4	80.1	7.3	30.0
+ TROVE	97.1	85.1	97.6	50.5

performance of VideoCLIP-XL on Something-Something V2 is likely a result of general reasoning limitations rather than learned static feature biases.

Next, we extend the analysis provided in Section 5.2 by mitigating prediction errors on Kinetics600 and Something-Something V2 at test time. Table 7 shows that classification accuracy of VideoCLIP-XL on these tasks can be substantially improved given knowledge of TROVE-identified static feature biases.

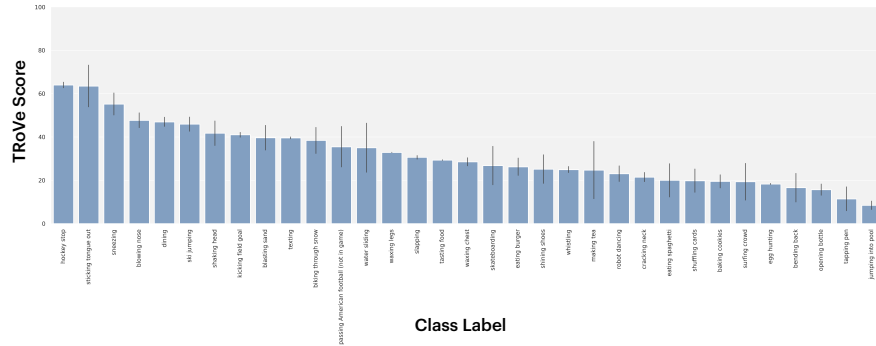
E Additional Applications of TROVE

In this section, we highlight another potential use-case of TROVE in real-world settings: analyzing the composition of temporal datasets. To analyze the composition of Kinetics400 [40], we construct an ensemble of six contrastive temporal VLMs; then, for each model, we use TROVE to identify static feature biases and corresponding class labels on which the bias induces errors. Our analysis finds that two classes in Kinetics400, namely `sneezing` and `cracking neck`, are identified as error-prone classes by all six models; this finding suggests that these two classes are likely to be consistently impacted by learned static feature biases. In contrast, 185 classes like `training dog` and `dancing ballet` are not affected by learned error-inducing static feature biases for any of the considered models.

In Figure 9, we vary the models included in the ensemble. Figure 9 [Panel A] considers an ensemble of two ViCLIP models (ViCLIP-B and ViCLIP-L) and identifies a total of 33 classes that are consistently impacted by learned static feature biases across both models. The 33 identified classes (sorted by their mean TROVE scores) include: `hockey stop`, `sticking tongue out`, `sneezing`, `blowing nose`, `dining`, `ski jumping`, `shaking head`, `kicking field goal`, `blasting sand`, `texting`, `biking through snow`, `passing American football (not in game)`, `water sliding`, `waxing legs`, `slapping`, `tasting food`, `waxing chest`, `skateboarding`, `eating burger`, `shining shoes`, `whistling`, `making tea`, `robot dancing`, `cracking neck`, `eating spaghetti`, `shuffling cards`, `baking cookies`, `surfing crowd`, `egg hunting`, `bending back`, `opening bottle`, `tapping pen`, and `jumping into pool`. Figure 9 [Panel B] considers an ensemble of three XCLIP models (XCLIP-B/16, XCLIP-B/32, and XCLIP-L/14) and identifies a total of 4 classes that are consistently impacted by learned static feature biases across all three models. The 4 identified classes include: `making a cake`, `sneezing`, `eating chips`, and `cracking neck`.

Ultimately, given a temporal dataset associated with a task of interest, TROVE can enable identification of challenging class labels that are particularly susceptible to systematic errors from static feature biases, aiding with model development and evaluation.

[A] Kinetics400 Classes Consistently Impacted by Static Feature Biases (ViCLIP model family)



[B] Kinetics400 Classes Consistently Impacted by Static Feature Biases (XCLIP model family)

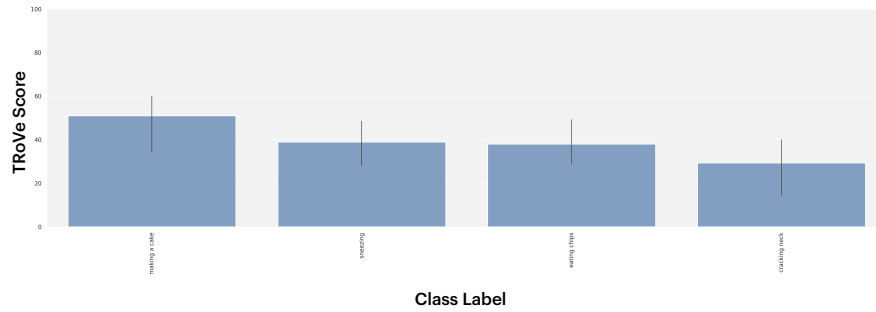


Figure 9: Several classes in Kinetics400 were found to be consistently impacted by learned static feature biases across multiple trained VLMs. Panel [A] depicts Kinetics400 classes consistently impacted by static feature biases across ViCLIP-L and ViCLIP-B. Panel [B] depicts Kinetics400 classes consistently impacted by static feature biases across XCLIP-B/16, XCLIP-B/32, and XCLIP-L/14.

F Extended Related Work

Here, we extend Section 2 by providing a discussion on the temporal action localization (TAL) task, which involves trimming videos to only include frames in which the action is directly visible [53]. TAL is a useful preprocessing step for reducing the effects of noisy or irrelevant frames and can be helpful in some cases for mitigating the influence of static feature biases; however, we note that TAL is not a universal solution for the following two reasons. First, in activity recognition settings, static features are not simply limited to noisy or irrelevant frames; rather, error-inducing static features are often directly observed in relevant action frames. As a representative example, consider Figure 8 (Panel A), where TROVE discovers that the presence of children and toy carts leads to prediction errors on the class "pushing cart". The learned static feature bias directly involves the subject performing the action, and trimming content will not be useful for addressing these prediction errors. Second, existing TAL methods are designed specifically for activity recognition and cannot be easily extended to other temporal settings. In our work, we extend beyond the traditionally-researched human activity recognition setting and include evaluations on both a synthetic task and a medical imaging task. TAL methods cannot be easily extended to these domains, as there is no clear analog to "trimming" content. Thus, TAL will have no effect on the presence of static feature biases here. As a result, static feature biases cannot be solved by only trimming videos to discard noisy or irrelevant frames, and methods like TROVE are necessary for identifying this important failure mode.

G Extended Discussion

Potential Societal Impacts: In this work, we demonstrate that static feature biases are a critical issue for temporal VLMs. Static feature biases are a type of spurious correlation and can contribute to prediction errors on downstream prediction tasks. We hope that our proposed approach can be utilized to detect and mitigate errors resulting from static feature biases prior to real-world deployment. Such an approach has the potential to improve robustness of temporal VLMs. This can be particularly advantageous in safety-critical settings like healthcare, where models capable of processing medical images across multiple timepoints are gaining increasing popularity.

Limitations and Future Work: Our work aims to discover and mitigate error-inducing static feature biases learned by temporal VLMs. We specifically focus on *image* sequences in this work, where each sequence represents a series of images collected over time (e.g. video frame sequences, medical images collected at varying timepoints, etc.). However, temporal data exists in many other modalities, such as audio and signals. Our work does not consider these settings, and determining the role of static feature biases in inducing prediction errors across non-image modalities would be an interesting direction for future work.

NeurIPS Paper Checklist

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [\[Yes\]](#)

Justification: The claims made in the abstract and introduction reflect the contributions of the paper.

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [\[Yes\]](#)

Justification: Limitations are discussed in Appendix Section G.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. Theory assumptions and proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [\[NA\]](#)

Justification: The paper does not include theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. Experimental result reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [\[Yes\]](#)

Justification: We have provided experimental details in Sections 4 and 5, with extended implementation details and hyperparameters in Appendix Sections A and C. Our code is available on GitHub.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
 - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
 - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
 - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
 - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Code is publicly available. All implementation details and hyperparameters are detailed in Appendix Section A and C.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. Experimental setting/details

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: We have provided implementation details and hyperparameters for all models used in this work. Details are in Appendix Section A and C.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. Experiment statistical significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We provide error bars (95% confidence intervals) when reporting performance of TROVE on our synthetic evaluation settings (see Figures 5, 6, and 7).

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments compute resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: Details on computational resources used when developing our evaluation framework are provided in Appendix Section A.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. Code of ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

Answer: [Yes]

Justification: This work conforms with the NeurIPS Code of Ethics.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. Broader impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: We provide a discussion of societal impact in Appendix Section G.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.

- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: This paper poses no such risks.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All existing data (Kinetics400, MS-CXR-T), models (VideoCLIP-XL, ViCLIP, XCLIP, CLIP, SigLIP, and BioViL-T), and code used in this paper have been cited. Kinetics400 is open source. MS-CXR-T is available under PhysioNet Credentialed Health Data License 1.5.0. VideoCLIP-XL is available under CC-BY-NC-SA-4.0. CLIP, ViCLIP, XCLIP, and BioViL-T are available under MIT licenses. SigLIP is available under an Apache-2.0 license.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.

- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [\[Yes\]](#)

Justification: Our evaluation framework and algorithm are documented in Sections 4 and 5 as well as Appendix Sections A and C.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [\[NA\]](#)

Justification: This paper does not involve crowdsourcing or research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [\[NA\]](#)

Justification: This paper does not involve crowdsourcing or research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: LLMs are not a component of the core methods in this research.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (<https://neurips.cc/Conferences/2025/LLM>) for what should or should not be described.