Made to Order: Discovering monotonic temporal changes via self-supervised video ordering

Charig Yang[1], Weidi Xie[1,2], and Andrew Zisserman[1]

[1] Visual Geometry Group, University of Oxford
[2] CMIC, Shanghai Jiao Tong University
{charig,weidi,az}@robots.ox.ac.uk
https://charigyang.github.io/order/

Abstract. Our objective is to discover and localize monotonic temporal changes in a sequence of images. To achieve this, we exploit a simple proxy task of ordering a shuffled image sequence, with 'time' serving as a supervisory signal since only changes that are monotonic with time can give rise to the correct ordering. We also introduce a flexible transformer-based model for general-purpose ordering of image sequences of arbitrary length with built-in attribution maps. After training, the model successfully discovers and localizes monotonic changes while ignoring cyclic and stochastic ones. We demonstrate applications of the model in multiple video settings covering different scene and object types, discovering both object-level and environmental changes in unseen sequences. We also demonstrate that the attention-based attribution maps function as effective prompts for segmenting the changing regions, and that the learned representations can be used for downstream applications. Finally, we show that the model achieves the state of the art on standard benchmarks for ordering a set of images.

Keywords: Ordering · Change detection · Self-supervised learning

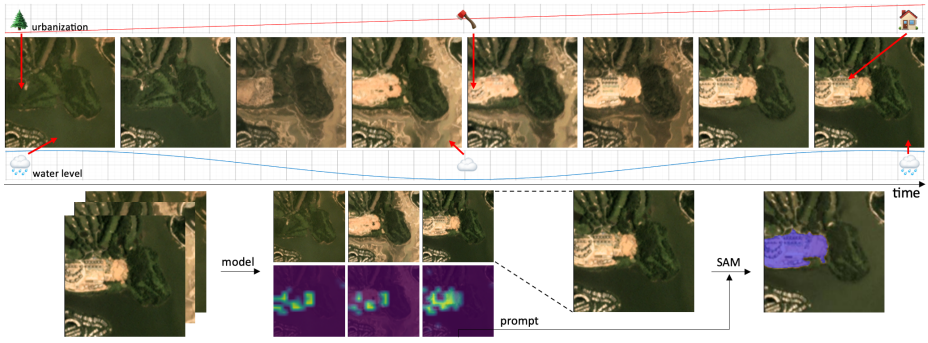


Fig. 1: Localizing monotonic temporal changes. Top: satellite images (ordered left to right) taken months apart, containing several changes – some are correlated monotonically with time (such as urbanization), while others are seasonal/cyclic (such as water level). Bottom: Our model's attribution map prediction on the sequence is able to localize the regions with monotonic time-correlated changes (in green), while being invariant to the seasonal and sporadic changes. The model is trained with no manual supervision, generalises to unseen sequences (as here), and the attribution map can also be used as a prompt to obtain segmentation.

1 Introduction

In the image sequence in Figure 1, there exists numerous changes over time, though many of these are seasonal, and hence distracting for long-term monitoring applications. As humans, we not only observe what is changing, but also reason about which changes are correlated monotonically with time and which ones are not. In this paper, we introduce a new task of automatically identifying temporally correlated changes in an image sequence, while being invariant to other changes. More specifically we wish to *discover* and *localize* the image regions where the change is correlated with time. Our motivation is to go beyond just detecting changes, but also discovering what changes are relevant over a period of time, and to explore the potential applications that this task could enable.

To achieve this, we propose a self-supervised proxy task, with time serving as a supervisory signal: the task is simply to order a shuffled sequence of video frames. The insight is that if a model can order the video frames, it would have learned to identify relevant (monotonic temporal-correlated) cues while disregarding other changes. The trained model can then be employed for video analysis applications where the goal is to identifying **changes** over time, such as developments or deforestation in satellite imagery (whilst ignoring seasonal variations) (see Figure 1) or aging signs in medical images. It can also be employed for detecting and tracking monotonic object **motion** over time, such as shadows caused by the movement of the sun or animals moving smoothly across the scene.

It is worth noting that temporal ordering has previously been used as a proxy task for self-supervised representation learning, with the learnt representations then finetuned for downstream tasks, such as video action recognition [18,39,46,64] (though due to the disparity between the proxy and downstream tasks, the effectiveness of learnt visual representation from temporal ordering has been unsatisfactory compared to other proxy tasks [14, 24, 27, 28]). In contrast, the objective in this paper is to use ordering as a proxy task to directly train a model for discovering and localizing monotonic changes in video sequences, without any subsequent supervised finetuning. In this sense, we are similar in spirit to previous works that use self-supervision to directly solve tasks, such as [8, 30, 34, 66] that targets tracking and segmentation by training on proxy tasks.

In order to harness the ordering proxy task, we introduce a transformer-based model that is able (i) to perform *ordering* on natural images sequences, and, more importantly, (ii) to provide *attribution* by localizing the evidence that gives rise to its prediction. Specifically, we use a DETR-like transformer encoder-decoder architecture where the queries in the decoder are cast as an *ordering index*. The architecture is designed to allow a attribution map to be obtained directly as part of the proxy training. Once the model has been trained for a particular setting, such as change detection in satellite image sequences, then it can generalize to unseen videos in the same setting, requiring only a forward pass to predict the localization of the monotonic temporal changes from the attribution map.

To summarize, in this paper, we make the following four contributions: (i) we introduce a new task of discovering and localizing monotonic temporal changes in image sequences, and use temporal ordering as a self-supervised proxy loss

for training. (ii) we introduce a flexible transformer-based model that can be used for ordering images of different sequence lengths, and also for localizing evidence. (iii) once trained on a setting (such as satellite images), the model is able to discover the changes correlating monotonically with time in unseen image sequences in the same setting, and we demonstrate several situations where this can be applied. Finally, (iv) we demonstrate that the trained model is able to order novel image sequences surpassing the performance of previous approaches for ordering on standard benchmarks.

2 Related Work

Video self-supervised learning has become increasingly popular in computer vision. Most research in this area focuses on representation learning, with an emphasis on downstream performance after supervised fine-tuning or linear probing. In contrast, there is a less explored direction that goes beyond representation learning to directly learning a useful task under the self-supervised learning setting, such as depth [22,70], optical flow [41,45], correspondence [63] and sound localization [2,3,13,40]. Our work in this paper builds upon such paradigm, that enables to deploy the model to downstream tasks without the need for labels.

Self-supervised learning from time. This work involves using temporal ordering as a supervisory signal. Alternative sources of supervision is to leverage other cues, such as speed [6], uniformity [67], and the arrow of time [64]. The closest kin to our work involves using temporal sequencing as supervision [18,19,39,46], though their primary focus is on representation learning. In this work, we focus on advocating temporal ordering as a useful task on its own, showing that localization can emerge by using time as supervisory signal. We highlight the differences between our work and others in the Supplementary Material.

Ordering has been a longstanding task in computer science, dating back to sorting algorithms. In machine learning, it is a relevant task in both language [15, 16] and vision [4, 46, 58, 71]. For images, ordering has also been treated as a pretext for self-supervised pretraining, such as jigsaw puzzles [49], or as a task of interest, for example, image sequencing [4,5,31,57,68]. There is also some interest in the literature that focuses on differentiable sorting algorithms [12,17,25,51,52], though they mostly focus on algorithmic developments, such as differentiable loss function and black-box combinatorial optimisation. While in this paper, we make contributions towards the architecture by introducing a transformer-based ordering model, which allows ordering of arbitrary-length image sequences, with built-in visualisation via attribution maps.

Attribution localization, specifically in the case where explicit supervision is not given, has been of interest in the vision community. In ConvNets, attribution methods attempt to look into the network to find out where it is seeing [21,69]. In transformers, several methods have been proposed to look into the attention on the `[CLS]` tokens [1]. Instead of these implicit localization, several methods have also carefully designed the architecture so that localization emerges explicitly despite not being trained on, such as in sound localization [2,3,13,40]. This work follows the latter paradigm, while using the self-supervision from ordering.

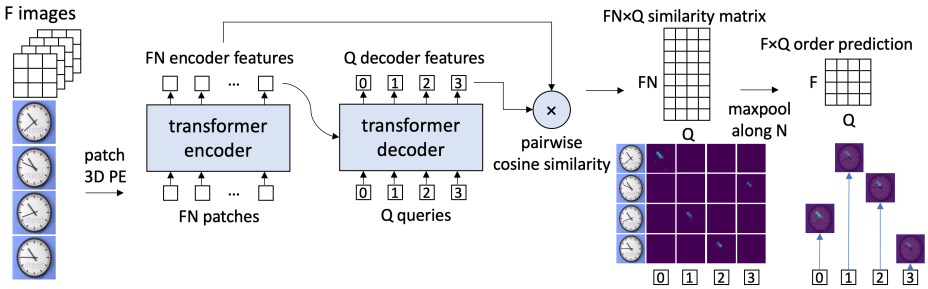


Fig. 2: Network architecture. For an unordered sequence of F frames each with N patches, the transformer encoder takes in all FN patches as input, and outputs FN features. The transformer decoder takes in Q learnable queries, each corresponding to an ordinal position, and the encoder output for cross-attention, resulting in Q features for output. A $FN \times Q$ cosine similarity matrix is constructed between all pairs of features from the encoder and decoder outputs, and the spatial max-pooling over this matrix reveals the $F \times Q$ order predictions. The ordering can simply then be obtained by taking an argmax along each query axis. In the example sequence, the hour hand is correlated monotonically with time, and appears in the attribution map.

Change detection has also been studied in computer vision. Many works look at changes in the image domain [53, 55], and across different applications from construction monitoring [59], satellite imaging [44], to medical imaging [50]. Other works associate short-term changes with motion, and use motion as a cue to discover moving objects [9, 10, 35, 36, 65, 66]. We differ from these lines of work in that we are mostly concerned with temporally coherent changes at different timescales, which may go beyond object level and not associated with motion.

3 Method

3.1 Problem Formulation

Our goal is to train a vision model to localize the changes in an image sequence that correlate monotonically with time. As a subsidiary goal, the model should also be able to order the image sequence.

Formally, given set of images, the model should output an *attribution map* $\mathbf{S}_{\text{att}} \in \mathbb{R}^{F \times H \times W}$ and an ordering $y_{\text{order}} \in \mathbb{Z}^F : y_{\text{order},i} \in \{0, 1, 2, ..., F-1\}$ as:

$$y_{\text{order}}, \mathbf{S}_{\text{att}} = \Phi(\mathcal{I}_0, \mathcal{I}_1, ..., \mathcal{I}_{F-1})$$

where $\mathcal{I} \in \mathbb{R}^{C \times H \times W}$ represents the input images. We show that we can train the model Φ via self-supervised learning on a proxy task, namely, ordering an arbitrary sequence of F images shuffled from a temporal sequence.

3.2 Ordering architecture

To address this problem, we propose a simple yet novel transformer-based architecture, as shown in Figure 2. The architecture comprises a transformer encoder (Φ_{enc}) that encodes the image patches, and a transformer decoder (Φ_{dec})

that encodes the ordering. To obtain an attribution map, we simply compute the pairwise cosine similarity between features from the encoder and queries from the decoder. We can then perform a max pooling operation across patches of the same image to get the ordering prediction.

Transformer Encoder (Φ_{enc}). To process an unordered sequence of F images, *i.e.*, $\mathcal{X} = \{\mathcal{I}_0, \mathcal{I}_1, ..., \mathcal{I}_{F-1}\}$, we start by dividing each frame $\mathcal{I} \in \mathbb{R}^{C \times H \times W}$ into 2D patches of size (P, P), resulting in $N = HW/P^2$ patches per frame and FN patches in total. Following the vision transformer approach, we flatten each patch using a learnable projection layer to D dimensions and add 3D positional encoding (spatial and frame) to each patch.

It is important to note that the frame positional encoding does not contain absolute temporal information since the frames are unordered, but it allows the patches to identify whether they belong to the same frame. As a result, after patchifying the input sequence, it ends up with a tensor of $\mathbf{x} \in \mathbb{R}^{F \times N \times D}$, which is then fed into a transformer encoder. The key difference to the standard vision transformer is that we output all the features, *i.e.*, $\mathbf{x}_{\text{enc}} \in \mathbb{R}^{F \times N \times D}$ instead of using a `[CLS]` token. In summary, we can express this procedure as $\mathbf{x}_{\text{enc}} = \Phi_{\text{enc}}(\mathcal{I}_0, \mathcal{I}_1, ..., \mathcal{I}_{F-1})$.

Transformer Decoder (Φ_{dec}). The transformer decoder is composed of Q learnable queries $\mathbf{q} \in \mathbb{R}^{Q \times D}$, with each corresponding to an ordering position $(0, 1, ..., Q-1)$. The task for the transformer decoder is to align the query vector with the encoder feature that demonstrates the correct temporal order. These queries iteratively attend the visual outputs from the encoder with cross-attention in the standard transformer decoder. We denote the output of the decoder as, $\mathbf{x}_{\text{dec}} \in \mathbb{R}^{Q \times D} = \Phi_{\text{dec}}(\mathbf{q}, \mathbf{x}_{\text{enc}})$. In practice, $Q = F$.

Cosine similarity matrix (S). Recall that we now possess two sets of features: encoder features $\mathbf{x}_{\text{enc}} \in \mathbb{R}^{F \times N \times D}$ and decoder features $\mathbf{x}_{\text{dec}} \in \mathbb{R}^{Q \times D}$. We then compute the pairwise cosine similarity matrix $\mathbf{S} \in \mathbb{R}^{F \times N \times Q} : [\mathbf{S}]_{i,j} = \cos(\mathbf{x}_{\text{enc},i}, \mathbf{x}_{\text{dec},j}) \in [-1, 1]$ between each i of the $F \times N$ features in $\mathbf{x}_{\text{enc}}$ and each j of the Q features in $\mathbf{x}_{\text{dec}}$, where $\cos(\cdot, \cdot)$ denotes the cosine similarity function.

Given the similarity matrix, we want to obtain (i) the ordering of the frames and (ii) the attribution map that indicates the spatial evidence within each frame that gives rise to ordering. The matrix $\mathbf{S} \in \mathbb{R}^{F \times N \times Q}$ consists of $F \times Q$ different spatial maps of size N, each indicating the attention between each pair of queries ($j \in Q$) and images ($i \in F$).

Order prediction. To obtain the order predictions, we perform spatial max-pooling over the patches of each frame (along the N dimension), to obtain $\hat{y} \in \mathbb{R}^{F \times Q} = \max_{i \in N} \mathbf{S}_i$. This max-pooling is designed to create an information bottleneck – the query has to attend to the correct token(s) within the correct image in order to predict the order correctly. The resulting matrix serves as a predictor for the position of each query in the ordering. We then apply a softmax along the query axis of the matrix to get the probability scores for each query.

Attribution map. Among the $F \times Q$ different spatial maps, we are only interested in the ones that correspond to the correct ordering. For each query $j \in Q$, we select one map $i \in F$ that has the maximum activation, *i.e.* $i = \arg\max \hat{y}_j$

resulting in Q maps. We then rearrange and resize each map of N patches back to the original resolution, resulting in $\mathbf{S}_{\text{att}} \in \mathbb{R}^{Q \times H \times W}$. Notably, this localization can be achieved without the need for additional fine-tuning, supervision, or post-hoc attribution methods [1, 21].

3.3 Training and inference

Temporal loss. Given the ground-truth order $y \in \mathbb{Z}^Q : y_i \in \{0, 1, ..., F-1\}$, the model can be trained via binary cross-entropy loss. Specifically, we convert the ground-truth order into a matrix where each column is a one-hot vector indicating its position, e.g. $(1, 0, 2)$ as $\left(\begin{smallmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{smallmatrix}\right)$. With some notation abuse, we still call this matrix $y \in \mathbb{Z}^{F \times Q}$. The forward loss is then simply the elementwise binary cross-entropy between the two matrices: $\mathcal{L}_f(y, \hat{y}) = \frac{1}{FQ} \sum_{i \in F} \sum_{j \in Q} \text{cross-entropy}(\hat{y}_{ij}, y_{ij})$.

In practice, we find that allowing reversibility in the loss aids with training, as many changes are reversible in nature without prior knowledge of the arrow of time [64] (the sequence could equally be ordered from first to last, or last to first). To allow this, we calculate the loss as the minimum of the loss for both forward and backward sequences, *i.e.* $\mathcal{L}_r = \min(\mathcal{L}_f(y, \hat{y}), \mathcal{L}_f(\text{reverse}(y), \hat{y}))$. The loss is minimized when the sequence is in the correct order, regardless of the direction.

Inference. At inference time, we simply take the argmax along each query axis as the order prediction, that is, $y_{\text{order}} \in \mathbb{Z}^Q : y_{\text{order},j} = \arg\max_{i \in F} \hat{y}_{i,j}$ In other words, each query picks the image that contains the maximum activation for its query, as illustrated in the bottom-right corner of Figure 2.

3.4 Discussion

Generalization to different sequence lengths. Our architecture is designed to handle sequences of arbitrary, possibly unequal length during training and inference, without the need to re-design or train separate models for each sequence length. At training time, we assume there is a maximum number of images, thus initialize a total of $F_{\max}$ learnable queries in the transformer, *i.e.*, $\mathbf{q} \in \mathbb{R}^{F_{\max} \times D}$. While the model handles a sequence of F images, with $F \leq F_{max}$, it only uses the first F queries as input to the decoder, ignoring the rest. This approach enables each query to represent its positioning $(0, 1, ..., F-1)$, making it generalizable to different lengths during both training and testing. However, the model will not generalize to lengths above $F_{\max}$.

Avoiding trivial solutions. There are two factors that we need to account for: camera motion and video compression artifacts. Camera motion can be smooth or uniform over a short time gap, which can result in an uninteresting cue. To address this, we apply a small random cropping on each frame in settings where the time gap between frames is small (*i.e.* < 1s). This slight jittering helps to prevent the model from learning trivial solutions. We note that this does not degrade the performance even if camera motion is absent. Another factor that can give rise to trivial solutions is inter-frame video compression artifacts. To address this, we follow conventional wisdom [29, 64] and use H.264 formatting for all videos, thus minimizing compression artifacts and preventing trivial solutions.

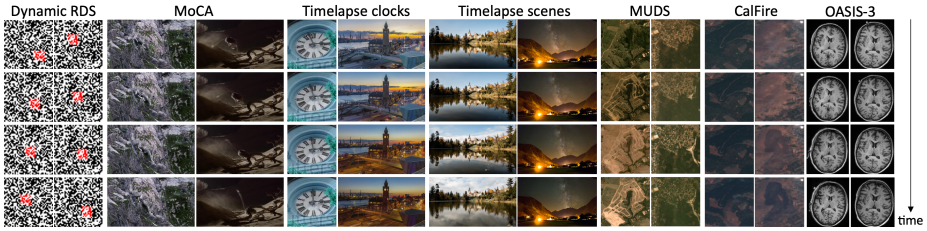


Fig. 3: Sequence datasets. From left to right: dynamic Random Dot Stereograms (RDS) (moving dots colored only for illustration), moving camouflaged animals (MoCA), timelapse clocks (cropped/full), timelapse scenes, MUDS, CalFire, OASIS-3.

From localization to segmentation. While the attribution map is useful, some applications may benefit from going beyond just localization. Here, we propose two solutions. We can directly obtain segmentation at patch-level granularity by thresholding the attribution map, together with minimal post-processing namely averaging across frames and removing small contours. We note that this method is crude as it does not distinguish pixels within a patch. Alternatively, we can use the highest activation points (*i.e.* centre pixels of patches) as prompt for the Segment Anything model (SAM) [33], to obtain pixel-level segmentation masks. We use the pretrained SAM model without fine-tuning.

4 Experiments

4.1 Video datasets

To leverage the inherent temporal information in videos, specifically, we sample a sequence from a video, shuffle the frames, then train the model with the original ordering as groundtruth. This approach enables the attribution map to identify temporally coherent changes that contribute to the ordering while disregarding other changes. Our study explores various sequence types across multiple domains, each with distinct cues of interest, which we summarize in Table 1 and illustrate with sample sequences in Figure 3.

Dynamic random dot stereograms are a type of image sequence that features a box of random dots moving smoothly over a background of random dots, originally used as a pair to demonstrate stereoscopic motion [47]. While the individual images may appear random, the box is visible when viewed in sequence. We generate a synthetic dataset of controllable dynamic random dot stereograms (Dynamic RDS) to test the model's ability to detect subtle relative cues. Since this a synthetic dataset, we know the ground-truth of the box's motion, so can compare this with the predictions.

Moving camouflaged animals (MoCA) [36] was constructed from videos of camouflaged animals. We use this dataset to investigate the use of short-term object locomotion as a cue, particularly where it is challenging to distinguish the object from the background. To accomplish this, we follow [66] and focus on the subset of 88 videos in which the animals are in motion. To evaluate on localisation, we assume that the change is object-level due to animal motion, and use the annotated object bounding box as the ground-truth for localization.

dataset	Δt	cue	seq (trn/test)	EM	EW
Dynamic RDS	<1s	motion	∞	99.8	99.9
MoCA	<1s	motion	75/13	82.0	90.6
Clocks (cropped)	∼1m	clock	2011/500	62.5	73.0
Clocks (full)	∼1h	clock/scene	210/20	55.0	74.3
Timelapse scenes	∼1h	scene	130/50	61.8	79.4
MUDS	1mo	landscape	60/20	56.4	69.6
CalFire	1mo	landscape	800/276	76.6	87.5
OASIS-3	1y	brain	100/34	84.3	89.1

Table 1: Dataset attributes and ordering results. This table shows different datasets and their attributes, as well as the ordering results on the test (unseen) sequences on exact match (EM) and elementwise (EW) metrics.

Timelapse analog clocks. Our study examines real-world scenes that feature both absolute cues (time on clocks) and relative cues (scene changes). To accomplish this, we utilize the Timelapse dataset [67] in two ways. Firstly, we use the entire dataset of 2,511 videos that features cropped clocks. Secondly, we create a subset consisting of 260 outdoor scenes with static cameras where the clock occupies only a small area of the scene.

Timelapse scenes. We have gathered an outdoor dataset of timelapse videos from various online sources. The dataset comprises 180 static videos, and our aim is to investigate the cues that the model can extract from the scene to learn its order, as there are no specific absolute cues present.

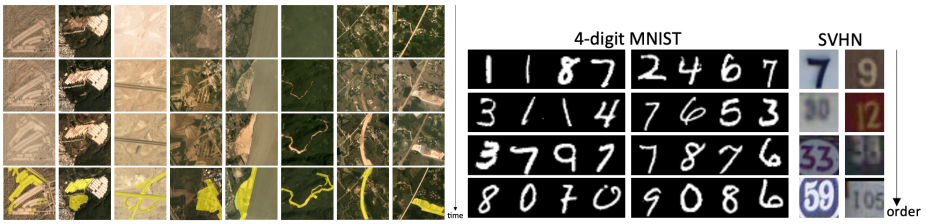
4.2 Temporal image sequences

We also show effectiveness of our approach for image sequences that are collected over a period of time, including satellite images and longitudinal medical data.

Multi-temporal urban development dataset (MUDS) [61] is a dataset that contains 80 aerial satellite image sequences captured monthly over a two-year period. We aim to identify geographical changes that occur over time, including deforestation and urbanization, while ignoring other changes. As there are no existing datasets and benchmarks for this task, we hand-label 60 sub-sequences on the test set for segmentation masks where changes are monotonic, and call this evaluation set **Monotonic MUDS**. Samples are shown in Figure 4a. We use this dataset to evaluate localization and segmentation performance of the model trained on the MUDS dataset.

CalFire [43] is a satellite dataset tracking wildfires in California. It also contains other events, such as snowfall, new construction, changes in water level, that we aim to discover. We pre-process by removing scenes with significant cloud cover.

OASIS-3 [37] contains longitudinal MRI scans taking 1-4 years apart to study how brains age. We select sequences with 3 or more scans, resulting in 134 sequences. Following [31], we perform affine registration and use the centre slice.



(a) Monotonic MUDS benchmark. (b) Image ordering datasets.

Fig. 4: (a) To evaluate localization and segmentation performance, we manually annotate the monotonically changing regions (shown in yellow) on the MUDS test set. Each sequence contains four frames, and the monotonic changes between the first and last frames are annotated. (b) 4-digit MNIST [38] (left) and SVHN [48] (right). The task is to order the images by the numbers they contain in increasing order (top to bottom).

4.3 Image ordering datasets

In addition, we showcase our general ordering capability by evaluating our method on standard benchmarks. We compare our ordering performance with related works [17, 25, 51, 52] on the task of sorting images of numbers in ascending order, as shown in Figure 4b. What the model has to learn here is different from the previous datasets, as the ordering is *absolute*, and not understanding changes.
Multi-digit MNIST [38] dataset is a modified version of the MNIST dataset in which four digits are concatenated to form a four-digit number. The goal is to order the image sequences in increasing order. To construct this dataset, we synthetically combine examples from the corresponding train and test sets of the MNIST dataset, resulting in a total of 50,000 training and 10,000 testing images.
Street view house numbers (SVHN) [48], was collected from Google Street View and includes images of house numbers. Similarly, the task is to order these numbers in increasing order. The dataset consists of 33,402 images for training and 13,068 for testing. To ensure consistency with previous studies, we followed the data preprocessing and augmentation methods described in [23].

4.4 Evaluation metrics

Localization. We use a pointing-game evaluation method, this is to follow the convention of the localization literature in other domains, including audio-visual localization [2, 3] and saliency methods [20, 21], that is, if the pixel with maximum activation in the attribution map contains the change of interest, then it is positive (1), otherwise it is negative (0). As our method only outputs patch-level attribution, we simply select the centre pixel of the patch as the highest activation. The overall accuracy is then calculated as the average over all sequences.
Segmentation. We use the standard metric for segmentation, *i.e.*, mean intersection over union (mIoU), where the mean is the average across all sequences.
Ordering. We use the evaluation metrics for sorting as outlined in [51, 52]. These metrics include exact match (EM) and elementwise (EW) accuracy. EM is considered correct if the entire sequence is ordered correctly, while EW considers the order accuracy per element. Following previous benchmarks we evaluate these

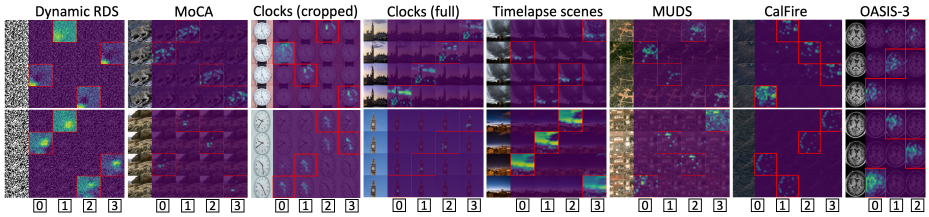


Fig. 5: Ordering and Localization results across various datasets, where the model is able to discover and localize various cues across different domains, including object motion, clocks, scenery, landscape and biological aging. The left column shows the input (unordered) images. Each column of the similarity matrix represents the model's prediction of each individual order (0, 1, 2, 3), where the image in the red box is chosen, and the attention heat map within the box localizes the change.

at sequence lengths 5, 9 and 16. To test the generalization to different sequence lengths, we also evaluate the exact match accuracy at a fixed sequence length of 5 at test-time (EM5), regardless of the training sequence length.

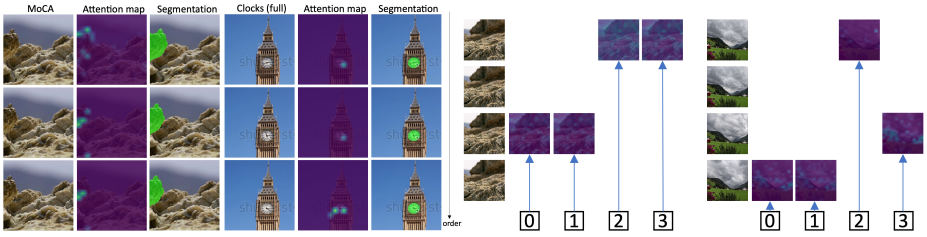
4.5 Implementation details

We split each dataset into disjoint training and testing subsets, and then randomly sample frames from each video. We keep the time gap between sampled frames constant within each video, but vary this across videos to train a robust model. For image sequences (MUDS, CalFire, OASIS-3) where data collection is less regular, we relax these constraints and simply randomly sample between all frames within the sequence. We train each model separately for each dataset.

Architecture. For the encoder, we use a smaller version of TimeSFormer [7] with a divided space-time attention architecture, consisting of 256 dimensions, 4 heads, 6 layers, and 512 MLP dimensions. For the decoder, we use a standard transformer decoder with the same parameters, except for 64 dimensions and 3 layers. As a result, the model is lightweight, with only 4M parameters. We use Adam optimizer [32] with learning rate 1e-4 in all experiments, and batch size 32 sequences with 4 frames per sequence for all video datasets, except 3 frames for OASIS-3 as this is the minimum MRI scan sessions per subject. For image ordering, we use batch size 100 with varying numbers of frames. All experiments are run on a single GPU. The code, datasets, and models will be released.

5 Results

We present ordering results in Section 5.1. Then, as our work cuts across several domains, we consider several avenues for comparison. In Section 5.2, we compare our method with existing change detection methods, and show that discovering monotonic temporal changes is unattainable by previous methods. In Section 5.3, we compare with other self-supervised proxy tasks and show that (i) our method works better than previous works in change localization, and (ii) our method learns better representations that can be finetuned to downstream tasks. Lastly, we show in Section 5.4 that our model can serve as a general-purpose ordering method that outperforms existing methods in image ordering benchmarks.



(a) SAM Prompting. **(b) Failure cases.**

Fig. 6: (a) The attribution map is used to prompt SAM to obtain segmentation masks. (b) Unorderable sequences, one being too static and the other being too stochastic.

5.1 Results on ordering video frames or image sequence

Video ordering results. The results for ordering as well as the main signals that the model can pick up, are shown by EM and EW in Table 1, with qualitative results in Figure 5. The model is able to successfully order sequences across different datasets, particularly in cases where changes are significant. It is expected that the scores are not perfect, as sampled data from videos is not guaranteed to contain ordering cues thus rendering the sequence simply unorderable, as illustrated in the Supplementary Material.

Temporal image sequence ordering results. For satellite image sequences, the model is able to discover cues that are relevant to ordering, including road building and forest fires. This illustrates our model's application on remote sensing imagery. In MRI scans, we explore the cues for age changes. Our results show that there are cues in the posterior part of the brain. This is consistent with the literature [11] that suggest that ventricular enlargement is a prominent feature, and causes the posterior horn to inflate in response to tissue loss. There are also some cues along the outline. This is in line with the literature [56, 60], which suggest that brain volume also decreases with old age.

Failure cases. A limitation of our model is that we do not force a one-to-one matching between queries and images, and this may result in some images being claimed by multiple different queries or by none at all, as seen in Figure 6b. This problem can easily be resolved by allowing each image to be predicted once. However, not being able to order also provides valuable information as not all real sequences can be ordered – for example, sequences where everything is static for a period, or very stochastic. Therefore, we treat invalid orderings as a means to provide information on whether particular frames can be ordered or not.

5.2 Comparison with change detection methods

We compare against three other previous approaches for their ability to detect monotonic temporal changes on the Monotonic MUDS dataset, namely, a supervised Siamese Networks [26] trained on MUDS dataset for the task of *urban development tracking*, by highlighting the differences between buildings; and the Change Vector Analysis [42], which is a baseline for the task of *change detection*, highlighting all changes in an image pair without knowledge their nature. Deep

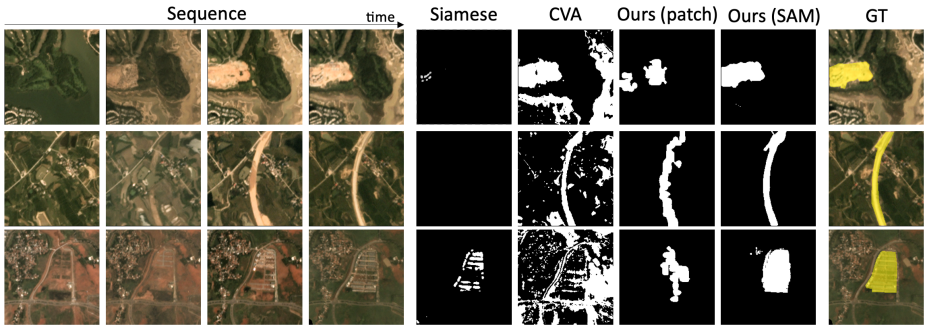


Fig. 7: Segmentation comparison with other methods: Siamese networks [26] and CVA [42]. Supervised methods ignore changes other than building, and pixel-based methods over-segments non-monotonic regions.

Method	Mono-MUDS loc (acc) ↑	Mono-MUDS seg (mIoU) ↑	RDS seg (mIoU) ↑	MoCA loc (acc) ↑
Siamese [26]	73.3	11.1	–	–
CVA [42]	71.7	34.6	22.3	69.6
DCVA [54]	70.0	35.5	–	–
Ours	**83.3**	**37.9** / **45.1** (SAM)	**34.2**	**75.0**

Table 2: Localization and segmentation results via the pointing game accuracy for localization, and mIoU for segmentation. The methods for segmentation (patch and SAM) are described in Section 3.4. Methods labelled as "–" are only trained on satellite images, hence do not generalize to other domains.

Change Vector Analysis [54] is a learned version of the above CVA that has been trained specifically on satellite images.

The performance comparison is given in Table 2, and illustrated in Figure 7. As can be seen, the urban development tracking method [26] under-segments changes that are correlated with urbanisation, as it is only trained to look at buildings. The change detection methods of [42, 54] over-segments changes that are non-monotonic as it has no concept of time. Both prior methods have shortcomings in detecting urban development: the former method (and even the ground-truth provided with the original dataset) misses changes like road building and deforestation, and the latter includes many erroneous regions such as the seasonal changes in vegetation and water level. Our model is able to highlight correctly the monotonic changes while being invariant to other changes. We further note that our model *discovers* such changes without any prior information on what to look for. We also evaluate on two other datasets: RDS and MoCA, where we have ground-truth for the moving objects in the video; and show that we obtain favourable results.

Quantitatively, in Table 2, we find that (i) our pointing-game localization and patch-level segmentation results outperform other methods despite operating

methods	loc (acc) ↑	seg (mIoU) ↑	finetuning (F1) ↑
scratch	—	—	18.2
S&L [46]	23.3	20.9	15.8
OPN [39]	25.0	24.1	17.1
AoT [64]	did not converge		—
Ours (AoT proxy)	63.3	26.9	25.5
Ours	**83.3**	**37.9 / 45.1 (SAM)**	**30.2**

Table 3: Comparison with other self-supervised methods (left) on localization and segmentation on Monotonic MUDS. (right) on fine-tuning performance on building change detection on MUDS.

only on patch-level (7×7) and not pixel-level granularity, and (ii) segmentation via SAM prompting further improves the results.

Qualitatively, the results of our localization experiments on various video datasets are presented in Figure 5. The model is capable of accurately identifying monotonic changes while remaining invariant to unrelated cues. Notably, these results were achieved on sequences unseen during training. We additionally show that our localization map works as a good prompt for the Segment Anything (SAM) model to obtain object-level changes, as in Figure 6a.

5.3 Comparison on self-supervised proxy tasks

Here, we compare to other self-supervised methods based on *time*. Please refer to the Supplementary Material for a discussion on the subtle differences between the proxy tasks. We include results for training baselines from scratch on MUDS, and testing on Mono-MUDS in localizing and segmenting monotonic changes. The results are shown in Table 3, where we conduct three sets of experiments, as detailed below.

First, we observe that previous methods are extremely crude in localization; this is expected, as they are all based on conv5 feature (even AoT, via CAM). Given a 224p input, conv5 has a 13×13 feature map with 195p receptive field. Our architecture handles localization by design, and is hence more capable than other methods that use post-hoc attribution methods on top of standard backbones. We also note that AoT does not train, which is also expected as it ingests optical flow as input (and in satellite images using flow does not make sense (change≠motion)), our method is more flexible in this regard.

Second, we ask if our method is still superior if the architectural gap is closed. To achieve this, we also compare the proxy task in AoT (time direction) with ours (ordering) under a fairer comparison (using our architecture and RGB input), and show this in the table under "Ours (AoT proxy)". We conclude that both our architecture and proxy task leads to significant improvements.

Third, we investigate representation learning for a target task that requires both time and localization: change detection of buildings on satellite images (like [26]). For each method, we pre-train the encoder on MUDS, freeze it, then train a lightweight head (3× deconvolutions) on top of the same dataset, and

dataset	MNIST			SVHN		
frames	5	9	16	5	9	16
DSort [51]	83.4|92.6	56.3|86.7	30.5|80.7|86.6	64.1|82.1	24.2|69.6	3.9|59.6|66.8
DSv2 [52]	84.9|93.2	63.8|89.1	31.1|82.2|—	68.5|84.1	39.9|75.8	12.2|65.6|—
Ptr-Net [62]	91.9|95.6	87.7|95.0	68.9|90.0|1.1	76.3|87.6	48.7|79.4	9.8|63.2|0.1
Ours	**93.9|96.7**	**87.9|95.2**	**72.2|91.2|92.9**	**77.3|88.2**	**53.9|81.0**	**19.4|67.9|67.6**

Table 4: Ordering on image datasets on two standard benchmarks (MNIST and SVHN) where the task is to order images of numbers in increasing order. Metrics are (EM|EW|EM5). EM and EW are evaluated at the sequence length the model has been trained on (5/9/16), whereas EM5 tests generalisation to test length 5.

test on unseen sequences. To keep this fair, we keep the number of parameters roughly the same across methods. The results (Table 3, right column) show that our method learns good representations as compared to previous self-supervised methods in localization tasks.

5.4 Comparison with image ordering methods

We compare to two previous methods on image ordering benchmarks where the task is to arrange the images in increasing order.
Differentiable sorting networks such as DiffSort [51] and its successor DSortv2 [52] employ a parameter-free sorting network to rank scalars. We compare with these sorting models, and show that our model is capable as a general ordering network with added functionality.
Pointer Networks [62] ranks features by using a recurrent encoder-decoder network with attention. We note that Ptr-Net is not initially designed for such a task, but for arranging a set of coordinates. We simply extend pointer networks by adding an image encoder and task the model to rank the image features from small to large. We then jointly train this encoder and the pointer network. For fair comparison, we use the same transformer encoder as our model, and use the pointer network as the decoder with similar size to our transformer decoder.

The quantitative results are presented in Table 4. Our results demonstrate that (i) we compare favourably in ordering performance – on both MNIST and SVHN, our method has the best performance of the four (ii) Ptr-Net does not automatically generalise to testing with different sequence lengths, while our method does (as reflected by the poor EM5 accuracy), and (iii) our method also has the added benefit of having an attribution map.

6 Conclusion

In this paper, we explore using time as a proxy loss for self-supervised training of models to discover and localize monotonic temporal changes in image sequences. Possible extensions include discovering more complex temporal changes (seasonal/periodic), or object state and attribute changes. It would also be interesting to investigate how the model scales with larger datasets and compute, and what new applications this task can enable. Overall, we hope this paper presents a valuable starting point for future research and applications in this area.

Acknowledgements. We thank Tengda Han, Ragav Sachdeva, and Aleksandar Shtedritski for suggestions and proofreading. This research is supported by the UK EPSRC CDT in AIMS (EP/S024050/1), and the UK EPSRC Programme Grant Visual AI (EP/T028572/1).

References

1. Abnar, S., Zuidema, W.: Quantifying attention flow in transformers. arXiv preprint arXiv:2005.00928 (2020) 3, 6
2. Afouras, T., Owens, A., Chung, J.S., Zisserman, A.: Self-supervised learning of audio-visual objects from video. In: European Conference on Computer Vision (ECCV) (2020) 3, 9
3. Arandjelovic, R., Zisserman, A.: Objects that sound. In: European Conference on Computer Vision (ECCV) (2018) 3, 9
4. Basha, T., Moses, Y., Avidan, S.: Photo sequencing. In: European Conference on Computer Vision (ECCV) (2012) 3
5. Basha, T.D., Moses, Y., Avidan, S.: Space-time tradeoffs in photo sequencing. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV) (December 2013) 3
6. Benaim, S., Ephrat, A., Lang, O., Mosseri, I., Freeman, W.T., Rubinstein, M., Irani, M., Dekel, T.: Speednet: Learning the speediness in videos. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2020) 3
7. Bertasius, G., Wang, H., Torresani, L.: Is space-time attention all you need for video understanding? In: Proceedings of the International Conference on Machine Learning (ICML) (July 2021) 10, 7
8. Bian, Z., Jabri, A., Efros, A.A., Owens, A.: Learning pixel trajectories with multi-scale contrastive random walks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2022) 2
9. Bideau, P., Learned-Miller, E.: A detailed rubric for motion segmentation. arXiv preprint arXiv:1610.10033 (2016) 4
10. Bideau, P., Learned-Miller, E.: It's moving! a probabilistic model for causal motion segmentation in moving camera videos. In: European Conference on Computer Vision (ECCV) (2016) 4
11. Blinkouskaya, Y., Weickenmeier, J.: Brain shape changes associated with cerebral atrophy in healthy aging and alzheimer's disease. Frontiers in Mechanical Engineering (2021) 11
12. Brown, A., Xie, W., Kalogeiton, V., Zisserman, A.: Smooth-ap: Smoothing the path towards large-scale image retrieval. In: European Conference on Computer Vision (ECCV) (2020) 3
13. Chen, H., Xie, W., Afouras, T., Nagrani, A., Vedaldi, A., Zisserman, A.: Localizing visual sounds the hard way. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2021) 3
14. Chen, T., Kornblith, S., Norouzi, M., Hinton, G.: A simple framework for contrastive learning of visual representations. In: Proceedings of the International Conference on Machine Learning (ICML) (2021) 2
15. Chen, X., Qiu, X., Huang, X.: Neural sentence ordering. arXiv preprint arXiv:1607.06952 (2016) 3
16. Cui, B., Li, Y., Chen, M., Zhang, Z.: Deep attentive sentence ordering network. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing (2018) 3

17. Cuturi, M., Teboul, O., Vert, J.P.: Differentiable ranking and sorting using optimal transport. Advances in Neural Information Processing Systems (2019) 3, 9
18. Fernando, B., Bilen, H., Gavves, E., Gould, S.: Self-supervised video representation learning with odd-one-out networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2017) 2, 3
19. Fernando, B., Gavves, E., Oramas, J.M., Ghodrati, A., Tuytelaars, T.: Modeling video evolution for action recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 5378–5387 (2015) 3
20. Fong, R., Patrick, M., Vedaldi, A.: Understanding deep networks via extremal perturbations and smooth masks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2019) 9
21. Fong, R.C., Vedaldi, A.: Interpretable explanations of black boxes by meaningful perturbation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2017) 3, 6, 9
22. Godard, C., Mac Aodha, O., Firman, M., Brostow, G.J.: Digging into self-supervised monocular depth estimation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (2019) 3
23. Goodfellow, I.J., Bulatov, Y., Ibarz, J., Arnoud, S., Shet, V.: Multi-digit number recognition from street view imagery using deep convolutional neural networks. arXiv preprint arXiv:1312.6082 (2013) 9
24. Grill, J.B., Strub, F., Altché, F., Tallec, C., Richemond, P.H., Buchatskaya, E., Doersch, C., Pires, B.A., Guo, Z.D., Azar, M.G., Piot, B., Kavukcuoglu, K., Munos, R., Valko, M.: Bootstrap your own latent: A new approach to self-supervised learning. In: Advances in Neural Information Processing Systems (2021) 2
25. Grover, A., Wang, E., Zweig, A., Ermon, S.: Stochastic optimization of sorting networks via continuous relaxations. arXiv preprint arXiv:1903.08850 (2019) 3, 9
26. Hafner, S., Ban, Y., Nascetti, A.: Urban change detection using a dual-task siamese network and semi-supervised learning. In: IGARSS 2022-2022 IEEE International Geoscience and Remote Sensing Symposium (2022) 11, 12, 13
27. Han, T., Xie, W., Zisserman, A.: Self-supervised co-training for video representation learning. In: Advances in Neural Information Processing Systems (2020) 2
28. He, K., Chen, X., Xie, S., Li, Y., Dollár, P., Girshick, R.: Masked autoencoders are scalable vision learners. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2022) 2
29. Iashin, V., Xie, W., Rahtu, E., Zisserman, A.: Sparse in space and time: Audio-visual synchronisation with trainable selectors. arXiv preprint arXiv:2210.07055 (2022) 6
30. Jabri, A., Owens, A., Efros, A.A.: Space-time correspondence as a contrastive random walk. Advances in Neural Information Processing Systems (2020) 2
31. Kim, H., Sabuncu, M.R.: Learning to compare longitudinal images. arXiv preprint arXiv:2304.02531 (2023) 3, 8
32. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014) 10
33. Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A.C., Lo, W.Y., Dollár, P., Girshick, R.: Segment anything. arXiv:2304.02643 (2023) 7
34. Lai, Z., Lu, E., Xie, W.: Mast: A memory-augmented self-supervised tracker. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2020) 2
35. Lamdouar, H., Xie, W., Zisserman, A.: Segmenting invisible moving objects. In: British Machine Vision Conference (2021) 4

36. Lamdouar, H., Yang, C., Xie, W., Zisserman, A.: Betrayed by motion: Camouflaged object discovery via motion segmentation. In: Proceedings of the Asian Conference on Computer Vision (2020) 4, 7
37. LaMontagne, P.J., Benzinger, T.L., Morris, J.C., Keefe, S., Hornbeck, R., Xiong, C., Grant, E., Hassenstab, J., Moulder, K., Vlassenko, A.G., et al.: Oasis-3: longitudinal neuroimaging, clinical, and cognitive dataset for normal aging and alzheimer disease. MedRxiv (2019) 8
38. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE (1998) 9
39. Lee, H.Y., Huang, J.B., Singh, M., Yang, M.H.: Unsupervised representation learning by sorting sequences. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV) (2017) 2, 3, 13
40. Liu, J., Ju, C., Xie, W., Zhang, Y.: Exploiting transformation invariance and equivariance for self-supervised sound localisation. In: Proceedings of the ACM International Conference on Multimedia (2022) 3
41. Liu, P., Lyu, M., King, I., Xu, J.: Selflow: Self-supervised learning of optical flow. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2019) 3
42. Malila, W.A.: Change vector analysis: An approach for detecting forest changes with landsat. In: LARS symposia (1980) 11, 12
43. Mall, U., Hariharan, B., Bala, K.: Change event dataset for discovery from spatio-temporal remote sensing imagery. Advances in Neural Information Processing Systems (2022) 8
44. Mall, U., Hariharan, B., Bala, K.: Change-aware sampling and contrastive learning for satellite images. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2023) 4
45. Meister, S., Hur, J., Roth, S.: Unflow: Unsupervised learning of optical flow with a bidirectional census loss. In: Proceedings of the AAAI conference on artificial intelligence (2018) 3
46. Misra, I., Zitnick, C.L., Hebert, M.: Shuffle and learn: unsupervised learning using temporal order verification. In: European Conference on Computer Vision (ECCV) (2016) 2, 3, 13
47. Neff, R., Schwartz, S., Stork, D.G.: Electronics for generating simultaneous random-dot cyclopean and monocular stimuli. Behavior Research Methods, Instruments, & Computers (1985) 7
48. Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A.Y.: Reading digits in natural images with unsupervised feature learning (2011) 9
49. Noroozi, M., Favaro, P.: Unsupervised learning of visual representations by solving jigsaw puzzles. In: European Conference on Computer Vision (ECCV) (2016) 3
50. Patriarche, J., Erickson, B.: A review of the automated detection of change in serial imaging studies of the brain. Journal of digital imaging (2004) 4
51. Petersen, F., Borgelt, C., Kuehne, H., Deussen, O.: Differentiable sorting networks for scalable sorting and ranking supervision. In: International Conference on Machine Learning (ICML) (2021) 3, 9, 14
52. Petersen, F., Borgelt, C., Kuehne, H., Deussen, O.: Monotonic differentiable sorting networks. arXiv preprint arXiv:2203.09630 (2022) 3, 9, 14
53. Sachdeva, R., Zisserman, A.: The change you want to see. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV) (2023) 4

54. Saha, S., Bovolo, F., Bruzzone, L.: Unsupervised deep change vector analysis for multiple-change detection in vhr images. IEEE Transactions on Geoscience and Remote Sensing **57**(6), 3677–3693 (2019) 12
55. Sakurada, K., Okatani, T.: Change detection from a street image pair using cnn features and superpixel segmentation. In: British Machine Vision Conference (2015) 4
56. Scahill, R.I., Frost, C., Jenkins, R., Whitwell, J.L., Rossor, M.N., Fox, N.C.: A longitudinal study of brain volume changes in normal aging using serial registered magnetic resonance imaging. Archives of neurology **60**(7), 989–994 (2003) 11
57. Sevilla-Lara, L., Zha, S., Yan, Z., Goswami, V., Feiszli, M., Torresani, L.: Only time can tell: Discovering temporal data for temporal modeling. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (2021) 3
58. Shvetsova, N., Petersen, F., Kukleva, A., Schiele, B., Kuehne, H.: Learning by sorting: Self-supervised learning with group ordering constraints. International Conference on Computer Vision (ICCV) (2023) 3
59. Stent, S., Gherardi, R., Stenger, B., Cipolla, R.: Detecting change for multi-view, long-term surface inspection. In: British Machine Vision Conference (2015) 4
60. Svennerholm, L., Boström, K., Jungbjer, B.: Changes in weight and compositions of major membrane components of human brain during the span of adult human life of swedes. Acta neuropathologica (1997) 11
61. Van Etten, A., Hogan, D., Manso, J.M., Shermeyer, J., Weir, N., Lewis, R.: The multi-temporal urban development spacenet dataset. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2021) 8
62. Vinyals, O., Fortunato, M., Jaitly, N.: Pointer networks. Advances in Neural Information Processing Systems (2015) 14, 3
63. Wang, X., Jabri, A., Efros, A.A.: Learning correspondence from the cycle-consistency of time. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2019) 3
64. Wei, D., Lim, J.J., Zisserman, A., Freeman, W.T.: Learning and using the arrow of time. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2018) 2, 3, 6, 13
65. Xie, J., Xie, W., Zisserman, A.: Segmenting moving objects via an object-centric layered representation. In: Advances in Neural Information Processing Systems (2022) 4
66. Yang, C., Lamdouar, H., Lu, E., Zisserman, A., Xie, W.: Self-supervised video object segmentation by motion grouping. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (2021) 2, 4, 7
67. Yang, C., Xie, W., Zisserman, A.: It's about time: Analog clock reading in the wild. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2022) 3, 8
68. Zarrabi, N., Avidan, S., Moses, Y.: Crowdcam: Dynamic region segmentation. arXiv preprint arXiv:1811.11455 (2018) 3
69. Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., Torralba, A.: Learning deep features for discriminative localization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2016) 3
70. Zhou, T., Brown, M., Snavely, N., Lowe, D.G.: Unsupervised learning of depth and ego-motion from video. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2017) 3
71. Zhukov, D., Alayrac, J.B., Laptev, I., Sivic, J.: Learning actionness via long-range temporal order verification. In: European Conference on Computer Vision (ECCV) (2020) 3

The Supplementary Material includes the following sections:

- **Sect. A: Datasets in more detail**, where the datasets used in the paper are explained further.
- **Sect. B: Code**, where we present the pseudocode alongside the attached code.
- **Sect. C: Related work in more detail**, where we compare our method against previous ones and highligh the differences.
- **Sect. D: Unorderable sequences**, where we discuss the model's handling of unorderable sequences.
- **Sect. E: Experimental settings**, where we perform variations on the train/test set division.
- **Sect. F: Ablation studies**, where we perform variations on the architecture.
- **Sect. G: Qualitative results**, for both orderable and unorderable cases.

A Datasets in more detail

Table 5 expands on the main paper's dataset attributes (Main paper, Table 1) to give more detailed dataset statistics.

dataset	resolution	patch size	# patches	# train	# test
MNIST	28×112	7	4×16	50000	10000
SVHN	54×54	6	9×9	33402	13068
Dynamic RDS	42×42	7	6×6	∞	∞
MoCA	336×336	21	16×16	75	13
Clocks (cropped)	196×196	14	14×14	2011	500
Clocks (full)	320×480	20	16×24	210	50
Timelapse scenes	336×336	21	16×16	130	50
MUDS	196×196	7	28×28	60	20
CalFire	336×336	21	16×16	800	276
OASIS-3	224×224	16	14×14	50	19

Table 5: Dataset attributes. The different video datasets used, and their attributes in detail.

B Code

The pseudocode is shown below, with the full code being available on the project webpage.

```
class Made2Order(nn.Module):
  def __init__(self):
    self.query = nn.Parameter([1,q,d]).repeat(b,1,1) # (q=queries, b=bsz)

  def forward(self, video):
    # input video of dimensions: b f c h w (f=frames, chw=image dimensions)
    x = patch_posemb(video) # b (f n) d (n=number of tokens)
    x_enc = self.TransformerEncoder(x) # b (f n) d
    x_dec = self.TransformerDecoder(self.query, x_enc) # b q d
    x_enc = F.normalize(x_enc, 2)
    x_dec = F.normalize(x_dec, 2)
    S = torch.einsum('bik,bjk->bij', x_enc, x_dec) # b (f n) q
    return einops.reduce(S, 'b (f n) q -> b f q', 'max')
```

C Related work in more detail

C.1 Self-supervised learning from time

	S&L [46]	OPN [39]	AoT [64]	Ours
Input	RGB	RGB	Flow	RGB
Filtering	Flow	Flow	Flow	no
Evidence	post-hoc (pool5)	post-hoc (pool5)	post-hoc (CAM)	built-in
Training task	is middle frame in-between	shuffle and order	forward or backward	shuffle and order
Goal	rep. learning	rep. learning	rep. learning+apps	change localization

Table 6: Comparison to related works. This table compares this paper with several related works, namely Shuffle and Learn [46], Order Prediction Network [39], and Arrow of Time [64]. Unlike previous work where the goal was self-supervised representation learning, our goal is to directly discover the change and predicts its localization.

On motivation. The main goal of Shuffle and Learn [46] and extensions such as Order Prediction Network [39] is representation learning using self-supervision as pre-training – the authors had the intuition that change was required and used a filter for this based on optical flow to select suitable frame-sets in order to learn good representations. Arrow of Time [64], too, focuses substantially on representation learning, though based on optical flow as input, and they too filter away sequences that lack motion using optical flow.

On attribution learning. All existing methods apply attribution methods post-hoc (S&L and OPN look at the activation on the spatial pooling after conv5

and AoT uses CAM on conv5 features). As mentioned in the main paper, both of these post-hoc methods have low accuracy as they rely on 7x7 conv5 features with very large receptive field. In contrast, we make a contribution in designing an architecture that has attribution built-in, enabling more precise attribution (please refer to the respective figures in each paper – Fig. 4 of [46], Fig. 8 of [39] and Fig. 4 of [64]), and no need for post-processing.

On pretext task. AoT asks the model to distinguish between forward and reverse order of an otherwise in-order sequence, while S&L fixes the start and end frames and asks if the middle frame is inside (see Sec. 3.2 of their paper for details). For our task it does not matter if the sequence is forwards or backwards – but that is the only goal of AoT, and the AoT goal would not be possible with shuffled frames. Our task is identical to OPN where the entire sequence is shuffled and the ordering is predicted, but we add (i) built-in attribution, and (ii) a more flexible architecture capable of general ordering across a variable number of frames. Without all these three add ons, our work would essentially boil down to the same as that of OPN.

C.2 Ordering methods

The other ordering architectures are shown in Figure 8, and their comparison is shown in Table 7. Our architecture allows for all of (i) ordering of absolute cues (e.g. images of numbers) (ii) ordering of relative cues (e.g. in natural images), (iii) ordering with flexible length during training and testing, and (iv) explicit localization via attribution map. Note that Ptr-Net is not initially designed for such a task, and we extend the architecture while preserving fair comparison.

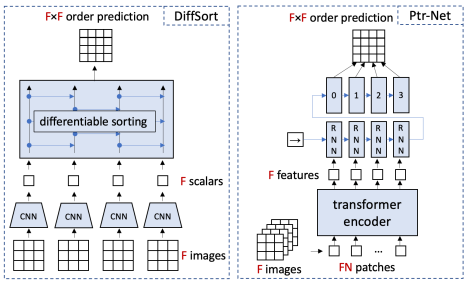


Fig. 8: Architectures of related works for ordering, including Differentiable Sorting [51] and Pointer Networks [62].

	DSort	Ptr	Ours
Absolute order	✓	✓	✓
Relative order	✗	✓	✓
Flexible length	✓	✗	✓
Localization	✗	✗	✓

Table 7: Related works. We show DiffSort [51], and Pointer Networks [62] are less versatile.

D Unorderable sequences

In real-world scenarios, sequences may not contain monotonic ordering cues for every frame (such as sequences that are static over several frames, or ones

where changes are cyclic), and hence it is impossible to order such sequences using monotonic changes alone – we denote such sequences as **unorderable**. We also observe that the model sometimes makes predictions that are **inconsistent** by predicting some ordering indices multiple times and others not at all. This happens when some frames are being claimed by multiple queries and others claimed by none at all. Here, we investigate the correlation between these two occurrences and ask whether the model is able to correctly identify unorderable sequences by making inconsistent predictions.

To do this, we create an unorderable set by including static frames on Mono-MUDS datset, resulting in 60 unorderable sequences (in addition to the 60 orderable ones) of 4 frames each. We experiment with these 120 sequences, and report the results in Table 8. We show that there is a strong correlation between the model predicting inconsistent ordering and the sequence being unorderable. We also note that this occurs as a result of a design during inference, hence training is unaffected.

In summary: an inconsistent prediction implies that the set of frames cannot be ordered, and the model is able to flag such sequences. In the case of videos, the ground-truth ordering is known, so we are able to easily detect when the prediction is incorrect or inconsistent. It is also possible to look at the (lack of) attention in the attribution map to observe the lack of ordering cues. We also show qualitative examples of these cases in Section G.2.

	correct	incorrect	inconsistent
gt orderable	88.3	10.0 ↑	1.7 ↓
gt unorderable	0.0	1.7 ↓	98.3 ↑

Table 8: Unorderable sequences. For an image sequence (0,1,2,3), the model either makes correct (0,1,2,3), incorrect *e.g.* (0,1,3,2), or inconsistent *e.g.* (0,1,3,3) predictions. We investigate the correlation between the model making inconsistent predictions and the sequence being unorderable. The model reliably gives inconsistent predictions for unorderable sequences.

E Experimental settings

In the main paper, we only explore the setting where the train and test sets are independent video clips within the same dataset, i.e. Figure 9, left. However, as a self-supervised task, we can explore other settings that are appropriate for applications.

Testing on the same clips, but at different times (Figure 9, centre). There may be cases in remote monitoring and surveillance where there exists past data that can be used for training, and the adaptation is mostly towards new sequences that are taken under the same or similar settings. In such cases, we can split the training and testing sets by time instead of by video clips.

Fig. 9: Experimental settings. The main paper only explores the case where the train and test videos are separate videos (left), all of the same class. Here we explore two other settings: testing on the same clips but at different time (centre), and testing on the training set (right).

Testing on the training set (Figure 9, right). Since no annotations are used, we can also test directly on the training set as an exploration tool to investigate what cues are used in determining the order.

Experiment. We conduct an experiment by having a common test set, and perform training on three different training settings, namely: (i) the training set does not include any videos from the test set, (ii) the training set includes the videos from the test set, but they do not overlap in time, and (iii) the training set includes the entire testing set. To keep the experiments fair we use the same number of clips for each training setting. We test this on MUDS dataset, and show the resutls in Table 9. It shows that the accuracy increases as the domain gap between training and test sets decreases.

setting	EM↑	EW↑
Base (split by video)	45.1	52.1
Testing on same clips, different time	49.7	59.6
Testing on training set	88.5	91.3

Table 9: Experiment on different settings. This table shows the accuracy for the common test set in MUDS dataset. Metrics are exact match and elementwise accuracies (higher is better).

F Ablation studies

In this section, we conduct ablation experiments on different aspects of the architectural design. As we introduce a new model in order to solve a new task, there are many variables that are interesting to investigate.

F.1 Transformer Architecture

To investigate the effect of model size on the ordering performance, we experiment by varying the sizes of the encoder and decoder. We do this by vary-

ing the numbers of (layers/feature dim/heads/feedforward dim) or both the encoder and decoder (in PyTorch's nn.Transformer, they are referred to as (num_layers/d_model/nhead/dim_feedforward). Feature dimensions refers to the number of features in self-attention or cross-attention within each transformer layer, while the feedforward dimension refers to the dimensionality of the feedforward network applied after attention within each transformer layer.

experiment	Encoder	Decoder	# params	EM↑	EW↑	EM5↑
Base (small)	6/256/4/512	3/64/4/512	3.6M	53.9	81.0	78.0
increase encoder size	2x base	base	19.4M	62.9	85.1	83.8
increase decoder size	base	2x base	6.4M	53.9	81.5	78.7
increase both	2x base	2x base	22.2M	62.7	84.9	83.0
decrease encoder size	0.5x base	base	816k	27.2	68.8	62.9
decrease decoder size	base	0.5x base	3.3M	51.7	80.1	76.8
decrease both	0.5x base	0.5x base	515k	25.6	68.0	61.8

Table 10: Ablation studies. We perform ablation studies varying the architecture of our model. The numbers for the encoder/decoder are (layers/feature dim/heads/feedforward dim). All ablations are relative to the base size, for example, 2x base refers to (12/512/8/1024). We evaluate on SVHN dataset with 9 frames. Base corresponds to the version implemented in the main paper.

The results are shown in Table 10. We can see from the results that the encoder size has a significant impact on the accuracy, while the decoder size does not matter as much. This is interesting, and we think that this arises from the fact that the encoder is doing most of the work, both in feature extraction and in ordering. We further investigate this in Section F.3.

F.2 Spatial feature map size

We also experiment varying the spatial feature map size, to look at the trade-off between the granularity of localization and ordering performance. Small patch sizes require significantly higher memory consumption.

experiment	patch size	# patches	EM↑	EW↑	EM5↑
Base (M)	6	9×9	53.9	81.0	78.0
S	3	18×18	47.5	78.2	74.9
L	9	6×6	54.4	81.5	79.8
XL	18	3×3	44.1	76.1	73.1

Table 11: Patch size ablation. We perform ablation studies varying the patch size. We evaluate on SVHN dataset with 9 frames. Base is used in the main paper.

As shown in Table 13, while small patches are generally good for visualisation, as the changes will be more finely localized, it does come with a trade-off that ordering accuracy drops when the patches are too small.

F.3 Encoder attention

We also look at the choice of attention used in the encoder. In particular, in the model of the main paper we use divided space-time attention in the transformer encoder so that tokens of the same spatial position between frames are allowed to communicate.

The benefit of allowing communication between the frames in the encoder is that the ordering can be done within the encoder itself, easing the load on the decoder. Without this, the encoder will merely act as a feature extractor, and the decoder will have to find a way to rank these features. Another benefit particularly in video ordering is that the patches with the same spatial position across the frames are able to communicate and compare with each other, highlighting the differences more easily.

In this section we also experiment with two other settings, where (i) attention is restricted to within each frame, and (ii) all tokens are allowed to communicate with each other (we refer the reader to Figure 2 of [7] for illustration). We evaluate both on image set ordering (SVHN) and video temporal ordering (MUDS).

experiment	GPU mem (GB)↓	SVHN↑	MUDS↑
Base (divided space-time)	8.0/5.3	53.9 | 81.0	56.4 | 69.6
Space only	8.1/5.7	55.8 | 82.1	13.0 | 35.8
Full attention	14.8/9.2	51.5 | 80.6	7.6 | 30.7

Table 12: Encoder attention ablation. We perform ablation studies varying the encoder attention. We evaluate on SVHN and MUDS with 9 and 4 frames respectively. Metrics are (EM | EW). We also show the GPU memory usage in each respective dataset (SVHN/MUDS), and show that full attention significantly consumes more memory.

As shown in Table 12, we observe that the results vary significantly with the datasets. In the case of image set ordering on SVHN, inter-frame communication matters less presumably because there is no differences between patches of the same spatial position to highlight, unlike the case of video temporal ordering. We also observe that space-only attention (6 spatial layers) obtains slightly higher accuracy than that of divided space-time attention (3 spatial layers, 3 temporal layers), which reinforces the hypothesis that spatial attention within the image is more important than across images.

In the case of ordering video frames (MUDS), we found that (i) space-only attention hinders the model's ability to communicate with each other, and (ii) full attention is significantly more difficult to train due to a significantly larger number of tokens to attend to.

F.4 Number of frames

In this section, we investigate the trade-off for increasing the number of frames in a sequence. As shown in Table 13, having more frames allows the cues to be

highlighted more clearly, and allows the model to learn better what changes are correlated with time and what are not. However, the number of possible orders will also increase combinatorially which makes the task more difficult.

experiment	EM↑	EW↑
Base (4)	62.5	73.0
2	51.3	51.3
3	65.2	73.1
5	37.1	55.0

Table 13: Frame count ablation. We perform ablation studies varying the number of frames. We evaluate on Timelapse clocks (cropped).

F.5 Choice of objective function

We have two different loss functions (forward and reversible). Consider a scene of a sky where the sun is moving upwards. Without accounting for reversibility, it is difficult to distinguish between a sunrise and a sunset in reverse, hence unidirectional ordering becomes challenging, sometimes impossible. Since our goal is only to attribute the change, not to determine the forward or backwards direction, our intuition is that considering reversibility in the sequence allows the model to be trained better, as this prevents giving the model confusing training signals. In this case we find that the model is (i) harder to train and (ii) susceptible to overfitting if we do not allow reversibility in the model, especially in datasets where changes are reversible.

Table 14 experimental results for the MoCA dataset as an ablation. From the table, the results show that using reversible losses helps with both (i) training and (ii) generalisation.

reversible?	loss↓	train EM↑	train EW↑	val EM↑	val EW↑
yes	0.119	84.2	91.5	82.0	90.6
no	0.169	76.0	83.9	44.5	53.5

Table 14: Objective function ablation. Ablation on choice of objective function. We investigate whether the reversible loss helps with training and generalization. We evaluate on MoCA dataset.

G Qualitative Results

G.1 More qualitative results

We show more qualitative results in Figure 10. The model is able to find different cues and use them for ordering.

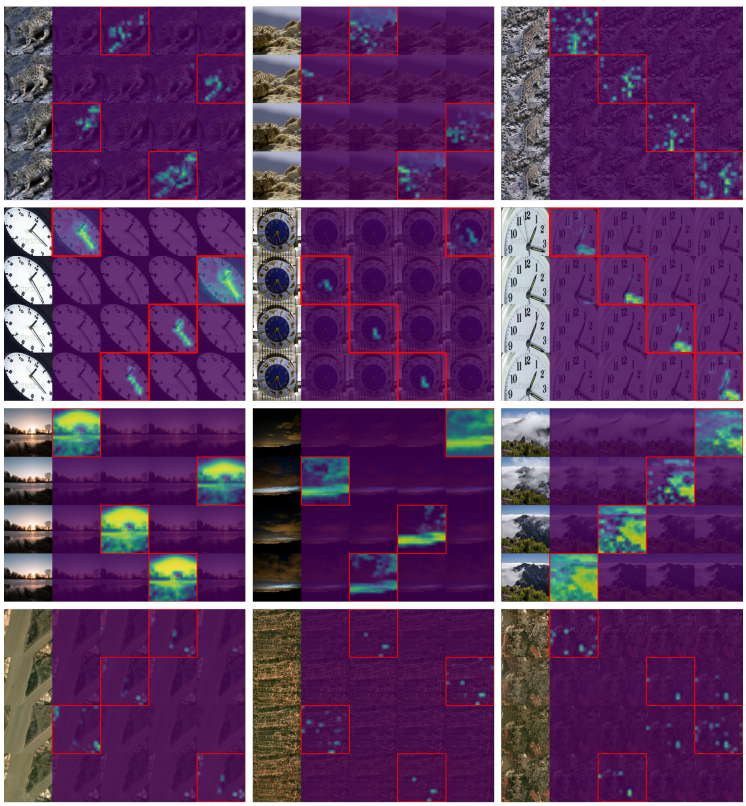


Fig. 10: Qualitative results. We show more qualitative results across different datasets. From top to bottom row: Camouflaged animals: the model uses motion as a cue in order to localize the moving subject; Timelapse clocks: the model correctly localizes the clock hands; Timelapse scenes: the model uses either the color of the sky or the objects such as clouds in ordering; Satellite images: the model is able to identify and localize structural changes in landscape such as new buildings and cleared land while being invariant to seasonal changes.

G.2 Unorderable sequences

We show qualitative examples of unorderable sequences in Figure 11. As we simply sample frames from video clips, there will be some sequences that cannot be ordered as they do not contain ordering cues. In particular, this can happen when: (i) there is insufficient change; (ii) changes are stochastic; or (iii) changes are seasonal (cyclic).

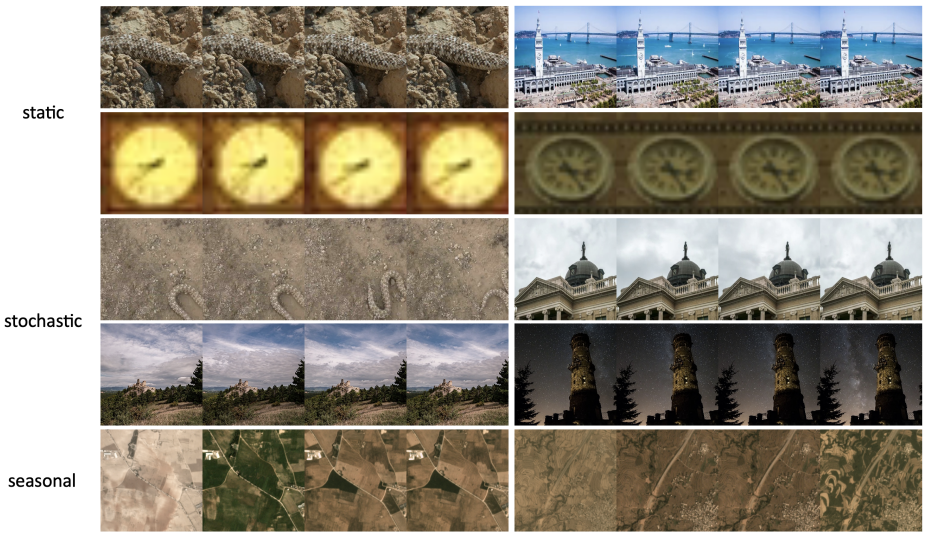


Fig. 11: Unorderable sequences. The figure shows examples of unorderable sequences across different datasets. Those selected are very reasonable: most contain insufficient cues for ordering to be determined, as the cues are either (i) too static, (ii) too stochastic, or (iii) uncorrelated with forward time (e.g. seasonal).