Capturing Temporal Information in a Single Frame: Channel Sampling Strategies for Action Recognition

Kiyoon Kim
kiyoon.kim@ed.ac.uk
Shreyank N Gowda
s.narayana-gowda@sms.ed.ac.uk
Oisin Mac Aodha
oisin.macaodha@ed.ac.uk
Laura Sevilla-Lara
lsevilla@ed.ac.uk

School of Informatics
University of Edinburgh
Edinburgh, UK

Abstract

We address the problem of capturing temporal information for video classification in 2D networks, without increasing their computational cost. Existing approaches focus on modifying the architecture of 2D networks (e.g. by including filters in the temporal dimension to turn them into 3D networks, or using optical flow, etc.), which increases computation cost. Instead, we propose a novel sampling strategy, where we re-order the channels of the input video, to capture short-term frame-to-frame changes. We observe that without bells and whistles, the proposed sampling strategy improves performance on multiple architectures (e.g. TSN, TRN, TSM, and MVFNet) and datasets (CATER, Something-Something-V1 and V2), up to 24% over the baseline of using the standard video input. In addition, our sampling strategies do not require training from scratch and do not increase the computational cost of training and testing. Given the generality of the results and the flexibility of the approach, we hope this can be widely useful to the video understanding community. Code is available on our website: https://github.com/kiyoon/channel_sampling.

1 Introduction

Understanding temporal information is crucial in order to understand video. While 2D convolutional filters are usually the standard approach to capture spatial information, there is a wider variety of methods for capturing temporal information. These include, for example, using the transformer architecture [1] 3D networks [2], computing optical flow and feeding it to a 2D convolutional network [3], using relational networks [4], or using Recurrent Neural Networks (RNNs) [5], among others. All of these methods require changes in the underlying network architectures, additional computational cost compared to simple 2D networks, as well as the time-consuming process of pre-training from scratch.

© 2022. The copyright of this document resides with its authors.
It may be distributed unchanged freely in print or electronic forms.
Inference Time (seconds), log scale

Accuracy (%)

TSN
TRN
TSM
MVFNet
TDN
Swin Transformer
RGB
TC+2
GrayST

Figure 1: We show that it is possible to significantly increase the performance of lightweight action recognition networks on the challenging Something-Something-V1 dataset by simply adapting how the individual image channels are sampled. Our approaches, TC+2 (in green) and GrayST (in orange), improve accuracy across several networks without increasing inference time. This enables us to narrow the gap between these efficient baselines and much more computationally demanding methods such as TDN [18] and Video Swin Transformers [12]. Note that we use a log scaling on the horizontal axis.

In this paper, we propose two simple and novel channel sampling strategies that improve the ability of a given 2D network to capture temporal information without changing the architecture. In particular, we re-order the channels of the input video in two ways: in the first, we re-order the channels of the video, so that each frame is composed of three channels: one that belongs to the frame before, one that belongs to the current frame, and one that belongs the frame after. These three channels are concatenated in the channel dimension as if it were a single frame. This procedure incorporates temporal information from neighboring frames, while keeping the dimensions of the input frame. In the second strategy, we compute a grayscale version of three neighboring frames at a time, and again concatenate them in the channel dimension, as if they were a single RGB image.

We test these extremely simple re-ordering strategies on five widely used 2D networks (TSN [17], TRN/MTRN [24], TSM [11], and MVFNet [21]). We observe that without any additional engineering these re-ordering strategies improve results up to 24% compared to the standard RGB channel ordering, across multiple challenging video datasets and different networks. We also observe that the improvement is particularly large in the datasets where temporal information is more important, in particular on CATER [6] and Something-Something [8]. Figure 1 illustrates the performance improvement which does not add additional significant computational overhead.

Our main contributions are:

• We improve the performance of simple 2D CNN-based action recognition models while incurring no additional computational complexity and with no modification to the underlying models. Our solutions can be used with most existing network architectures.

• Through extensive experimental evaluation on several common models and datasets, we show the efficiency of our proposed sampling methods and report superior performance compared to standard image sampling.
2 Related Work

Representations of Input Video. The most common representation of video frames for deep learning is 8-bit raw RGB, where each pixel has three color channels – red, green, and blue, each represented by 8 bits. This representation does not encode any temporal information in a frame. Another approach is to compute differential images by simply subtracting each pair of consecutive frames [17]. This gives the network a more explicit representation of temporal high-frequency changes. Unfortunately, this is not robust to spatial shifting, so as the camera pans, all pixel values change, which defeats the purpose of identifying the moving objects by subtracting the frames.

Optical flow is another common representation in action recognition [3, 15, 17]. Given two consecutive video frames, the optical flow is the direction and magnitude of the motion at each pixel. Using optical flow as a separate stream [15] generally improves performance, although it is often computationally expensive. Recent work has used the motion vectors from compressed video (e.g., H.264) to avoid the computational cost of optical flow [20, 23]. Given any of these representations, one of the standard approaches is to simply feed them as input to a 2D convolutional network, similar to those used in the image domain, where frames are processed independently, and the predictions are aggregated in the end. This family of approaches is called 2D, or frame-based networks. Despite their simplicity, they often work well on many datasets, and are fast to train and test.

Dynamic images [2] are a compact video representation that encode spatial and the entire motion information in a single image, so any image classification network can perform action recognition using the generated images. In contrast, our methods encode short-term temporal information per frame, resulting in space-time sequential representation (i.e., they do not compress an entire video into a single frame) and can thus use off-the-shelf 2D action recognition networks to better capture temporal information compared to only using a single image classification model as in [2].

3D Architectures. The main issue with 2D networks is that they are limited in their temporal receptive field size. 3D networks [16] on the other hand, benefit from learning spatio-temporal representations explicitly as they apply convolutions in both space and time. One of the most popular approaches is the I3D model [3] that inflates 2D models to 3D. As a result, they are able to make use of well established 2D architectures and pre-trained weights. Extensions to I3D include adding a non-local (NL) module [19] to help capture pixel-level correlations, resulting in improved performance. The SlowFast network [5] is also a widely used 3D network, and it uses two pathways, a slow and a fast one, to capture motion and fine temporal information. While 3D networks are helpful, they are much more expensive than their 2D counterparts. In this work we aim to introduce temporal information into 2D backbone-based models without requiring the computation of optical flow or without expensive 3D convolutions.

2D Architectures for Temporal Modeling. While 3D action recognition networks can perform better than their 2D counterparts, they have larger computational requirements for training and testing. This in turn, makes them challenging to deploy in resource-constrained settings, e.g. mobile and online scenarios. As a result, 2D networks for action recognition tasks is still common practice for certain application domains.

Temporal Segment Networks (TSN) [17] is an early and widely used 2D method. TSN simply extracts predictions for each frame sampled from the input video using a 2D backbone
network and then averages these predictions for the final output. Frames are sparsely sampled from longer videos so that the model can reason over long time spans. With an added optical flow stream, TSN achieves competitive performance on action datasets, even defeating some 3D networks at the time of release. This high performance is more obvious in datasets where there is a strong correlation between actions and static objects and scenes [10]. In contrast, TSN performs worse on datasets that require explicit temporal reasoning [8, 14].

Temporal Relation Networks (TRN) [24] aim to capture relational information across frames. For this, it also uses a 2D backbone to extract features from sparsely sampled frames, but then aggregates the frame-level features using a relational module [25]. The authors proposed a multi-scale relational module that learns 2-frame, 3-frame, and up to $T$-frame relationships (TRN-multiscale, or MTRN in short). This slightly improves performance over the single-scale TRN. This is an improvement over TSN, which is agnostic to the temporal ordering of the input frames. As a result, TRN improves performance on temporal datasets, e.g., [8]. However, their model is not very memory-efficient, as it requires $T - 1$ relational modules (from 2 to $T$ frame) using fully-connected layers, making it challenging to process many frames.

The Temporal Shift Module (TSM) [11] follows the same strategy as the TSN but modifies the ResNet [9] backbone so that the network can partially access information from one past and one future frame (i.e., temporal shift). In order to maintain the spatial feature learning capability, the temporal shift happens inside a residual branch, so the backbone choice is limited to those with residual connections. TSM achieved improved performance on the Something-Something-V2 dataset [8], showing that 2D models are not obsolete and are still competent in action recognition.

Multi-View Fusion Network (MVFNet) [21] adds channel-wise convolutions to model height-width, height-time, and width-time views instead of treating height-width-time video frames as a space-time signal. This approach performs better than TSM while still remaining computation efficient.

Finally, Temporal Difference Networks (TDN) [18] explicitly compute motion information by sampling five times as many input frames (e.g., 8-frame TDN uses 40 frames) and computes RGB differences for capturing short-term information and uses multi-scale attention for capturing long-term temporal information. TDN presented a new state-of-the-art on the Something-Something datasets, but at the cost of using significantly more input frames.

In this work, we take advantage of the efficiency and simplicity of 2D video models by proposing image channel sampling strategies that improve a model’s ability to capture temporal information. This increases accuracy, all without introducing any additional computation during training or testing.

3 Method

3.1 Overview

Given an input video $v$, the task of action recognition is to predict the action class label $y$ of the video. Typically, the input video $v$ can be sampled sparsely, which means that each video is evenly divided into $T$ segments. Then, in the standard 2D paradigm, a frame $x_i$ is chosen at random for each segment at training time. We call the set of sampled frames $X$, i.e., $X = \{x_1, x_2, \ldots, x_T\}$.

We let $f(x_i)$ denote the 2D backbone model that takes a single image as input, and
outputs a feature map. We compute \( f(X) = \{f(x_1), f(x_2), \ldots, f(x_T)\} \). Finally, \( g(\cdot) \) is a temporal aggregation module that takes the \( T \) feature maps and returns the output category prediction \( \hat{y} \). The aggregation function \( g(\cdot) \) can simply be an average of the individual predictions as in TSN \([11, 17]\),
\[
g(f(X)) = \frac{\sum_{t=1}^{T} h_{\theta}(f(x_t))}{T},
\]
where \( h_{\theta} \) is a per-frame classifier. In other cases, like TRN \([24]\), the aggregation function reasons about the relationship among multiple frames by concatenating the features and applying a multi-layer perceptron (MLP) \( h_{\phi} \),
\[
g(f(X)) = h_{\phi}(\text{concat}(f(X))).
\]
The multi-scale version of TRN (MTRN) works similarly but has several aggregation modules, each accounting for reasoning between 2-frame, 3-frame, \ldots, \( T \)-frame relationships.

Across variants of the 2D paradigm, the frames of a video are typically sampled sequentially with all color channels intact. Instead, we propose two alternative sampling strategies, Time-Color Reordering and Grayscale Short-Term Stacking that enables 2D backbone-based models to exploit temporal information by means of reordering the input channels. Despite their conceptual simplicity, these two sampling strategies significantly improve action recognition performance without requiring any structural changes to the backbone model.
3.2 Time-Color Reordering

We propose a simple yet powerful sampling technique for video frames called Time-Color (TC) Reordering. We allow the 2D backbone model \( f(x_i) \) to see temporal information by re-sampling the three color channels of the model. We do this by taking one color channel from the input clip (i.e. red) of 3 consecutive frames, and concatenating in the channel dimension to form an input “image”. Then we repeat the process with the next color channel (i.e. green), and so on. Note that this is merely a different representation of the input video by means of changing the channel order, and does not require any modification to the backbone model or include additional information or processing. More specifically, the process is as follows. Let \( X^{TC} = \{x^{TC}_1, x^{TC}_2, \ldots, x^{TC}_T\} \) denote a video clip sampled following the proposed TC Reordering sampling strategy, where

\[
 x^{TC}_i = \begin{cases} 
 (x^R_i, x^R_{i+1}, x^R_{i+2}), & \text{if } i \mod 3 = 1, \\
 (x^G_i, x^G_{i+1}, x^G_{i+2}), & \text{if } i \mod 3 = 2, \\
 (x^B_i, x^B_{i+1}, x^B_{i+2}), & \text{otherwise}.
\end{cases}
\]  

(3)

Here, \( x^R_i \) is the red channel from the \( i \)-th frame, \( x^G_i \) for green, and \( x^B_i \) for blue. Since the last two TC frames try to access future frames (i.e. \( x_{T+1} \) and \( x_{T+2} \)) that are not available, we just use \( x_T \) with the corresponding color channel for this case. This simple process yields frames that contain information about the neighboring frames, and it is the core reason for the large advantage we observe in temporal tasks. Fig. 2(a) outlines the procedure in detail.

We alternate color channels to expose the network to more varied data. That is, different channels depict a particular object with different brightness and contrast, while keeping the original shape of the object. We observe that this color alternation produces better results than a single one, and speculate that it may work as a form of data augmentation.

We also propose an alternative method TC+2 where we sample just two more frames to avoid the duplication of the last frames. The cost of sampling two more frames is almost negligible in many practical scenarios even with long-input networks, and we observe significant improvement of performance as the way the input data is formed is consistent throughout the frames.
3.3 Grayscale Short-Term Stacking

Here we introduce another sampling technique that we call Grayscale Short-Term Stacking (GrayST). This approach is designed to use more source frames with the same compact 2D networks by using grayscale images instead of RGB. The motivation is that in semantic understanding tasks, color information can be redundant, and we can use that capacity to include temporal information instead, by inputting more frames. Previous work [22] showed that there is only a 0.5% drop in accuracy on ImageNet [4], when training on grayscale images compared to RGB images. Thus, we replace the three color channels that are normally fed into a 2D backbone network with three grayscale frames, from three different sequential time steps. In effect, this allows the backbone model to see short temporal information at the expense of forgoing the ability to reason about color. This sampling strategy is visualized in Fig. 2(b).

When the input sequence length to a network is intended to be \( T \) frames, we simply sample \( 3 \times T \) frames in grayscale, and stack the three consecutive frames into one image, containing three grayscale channels. In offline scenarios, we can utilize higher temporal resolution per video clip without introducing any latency for training or testing. Some online scenarios may have a limited number of frames, so we also experiment with matching the number of input RGB frames, i.e., 8-frame GrayST (that sees 24 frames) vs 24-frame RGB.

We let \( X^{GST} = \{x_1^{GST}, x_2^{GST}, \ldots, x_T^{GST}\} \) denote a GrayST video clip. We first sample \( 3 \times T \) grayscale frames following the same sparse sampling strategy as before, 

\[
X^g = \{x_1^g, x_2^g, \ldots, x_{3T}^g\},
\]  

where \( x_i^g \) is a grayscale image. Then, a GrayST frame is made of three neighboring temporal frames in \( X^g \) which is defined as

\[
x_i^{GST} = (x_{3i-2}^g, x_{3i-1}^g, x_{3i}^g).
\]

A comparison between the two sampling techniques is visualized in Fig. 3.

4 Evaluation

In this section we evaluate our two channel sampling strategies on multiple different action recognition datasets using several different network architectures.

4.1 Experiment Details

Datasets. We experiment with challenging datasets that require extensive temporal reasoning. The datasets are chosen to span a wide range of situations, e.g., short and long range temporal reasoning, static and moving cameras, etc. CATER [6] is a synthetic action recognition dataset involving long-term temporal reasoning. CATER has two versions of the videos, with camera motion and without, and we experiment with both of them. CATER also consists of three different tasks: primitive multi-label action recognition (task 1), compositional multi-label action recognition (task 2), and localization of object of interest (task 3). We evaluate task 2, compositional action recognition, as it requires the longest temporal reasoning (from the beginning to the end of the videos consisting of 301 frames). This provides the biggest challenge for action recognition methods that only focus on short-term clips.
Table 1: Performance on CATER task 2 using 32-frame models ($T = 32$). We report last epoch’s validation mAP using one clip. Note, GrayST uses three times as many frames as RGB or TC, which are then converted to grayscale, and TC+2 uses just two frames more. The different sampling strategies do not impact the size of the network, i.e. in the end, they all get the same number of input frames.

Specifically, CATER task 2 has 301 classes. Each class represents the ordering of two action pairs: an object performing an action before/during/after another object performing another action, for example, “cone slides before cylinder rotates”. Furthermore, CATER allows containment of objects, and even recursion of it. That is, the action models have to remember which object is contained by which carrier, during which period, to successfully keep track of all actions happening in the scene.

We also evaluate on datasets that require temporal understanding such as Something-Something-V1 and Something-Something-V2 [8]. The Something-Something datasets consist of videos of pre-defined actions being performed using everyday objects. To focus on the action, and not the objects being used, these datasets removed object and scene bias by grouping the same action using various objects. The labels include “pushing something from right to left”, “putting something on a surface”, etc., which ensures the focus is on the action. Something-Something-V1 contains 108,499 videos with 174 class labels, while Something-Something-V2 consists of 220,847 videos of the same 174 classes.

Networks. We use a ResNet50 [9], pre-trained on ImageNet [4], as our 2D backbone model for all experiments. Both sampling strategies would likely benefit further from the backbone being pre-trained for their specific channel sampling strategies, but we leave this for future work. We mainly present results for five popular 2D models: TSN [17], TRN/MTRN [24], TSM [11], and MVFNet [21] with some additional experiments using I3D [1] pre-trained on Kinetics-400 [10] for completeness. Note, that we were not able to perform 32-frame MTRN experiments due to GPU VRAM limitations, as it has 31 temporal relational modules consisting of dense fully-connected layers. More details on the training settings can be found in the supplementary material.
Table 2: Validation accuracy on Something-Something with 8-frame models ($T = 8$). Models marked with $\ast$ use 30 clips for testing (3 spatial $\times$ 10 temporal), otherwise we report 1-clip accuracy. Our focus is on making simple 2D networks better, and thus we only report the performance of the expensive I3D on Something-Something-V1 for comparison.

### 4.2 Results

We illustrate results for CATER Static/Camera Motion task 2 in Table 1 and Something-Something-V1/V2 in Table 2. Note that the GrayST method gets to use three times more frames and TC+2 uses just two more frames than RGB or TC, while maintaining the same computational cost of the model. We also report I3D results on Something-Something-V1, to show the impact of our sampling strategies when applied to 3D networks, see Table 2(a).

#### TC Reordering Performance.
Using our TC Reordering, we observe that even the simple TSN model obtains much better performance compared to using conventional RGB frames with TRN and MTRN on all datasets. On TRN and MTRN, we observe significant improvement on most datasets, except on CATER Camera Motion task 2. However, it has less impact when combined with TSM and MVFNet, and sometimes hurts performance.

TC Reordering shifts the input video through its color channels in order to provide temporal information to 2D backbones. As a result, if a model already captures temporal information (e.g. TSM and I3D), TC Reordering is less effective, but still helps in many cases.

#### GrayST Performance.
We observe consistent improvement on GrayST over RGB on most datasets and all models. The first obvious reason may be that it gets to see more
frames, i.e. three times as many as the other strategies. By combining grayscale information from different points in time it enables the 2D backbones to see temporal information. Despite utilizing more frames, this method does not increase training and inference times, with the exception of the time associated with loading the extra frames and converting them to grayscale. In comparison, using more RGB frames would make training and testing slower.

8-frame GrayST vs 24-frame RGB One might wonder if GrayST is better just because it ‘sees’ more frames. However, it has been reported that simply increasing the number of RGB frames does not necessarily improve performance or it is very marginal [24], and can even decrease performance [7] depending on the dataset. For example, [19] showed a baseline I3D-ResNet50 network with a 32-frame input obtained 73.3% on the Kinetics-400 dataset [10]. [5] later conducted an experiment with an 8-frame input following the same recipe for training and obtained 73.4%. We also conducted an ablation experiment in Table 3 to show that in many cases, the GrayST 8-frame (that looks at 24 frames and generates 8-frame 3-channel representation) is better than the RGB 24-frame. A GrayST frame not only increases the number of frames without increasing training and inference cost, but also allows 2D models to see more temporal information by discarding redundant color information. Note that TSM can make use of the information contained in the extra frames (see 8 versus 24 frames), but this benefit comes with increasing the training/testing cost by a factor of three.

<table>
<thead>
<tr>
<th>Model</th>
<th>Sampling</th>
<th>Model Size (T)</th>
<th>Top-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSN</td>
<td>RGB</td>
<td>8</td>
<td>17.2</td>
</tr>
<tr>
<td></td>
<td>RGB</td>
<td>24</td>
<td>18.3</td>
</tr>
<tr>
<td></td>
<td>GrayST</td>
<td>8</td>
<td><strong>35.5</strong></td>
</tr>
<tr>
<td>TRN</td>
<td>RGB</td>
<td>8</td>
<td>29.7</td>
</tr>
<tr>
<td></td>
<td>RGB</td>
<td>24</td>
<td>31.7</td>
</tr>
<tr>
<td></td>
<td>GrayST</td>
<td>8</td>
<td><strong>37.0</strong></td>
</tr>
<tr>
<td>TSM</td>
<td>RGB</td>
<td>8</td>
<td>45.4</td>
</tr>
<tr>
<td></td>
<td>RGB</td>
<td>24</td>
<td><strong>51.1</strong></td>
</tr>
<tr>
<td></td>
<td>GrayST</td>
<td>8</td>
<td>47.6</td>
</tr>
</tbody>
</table>

Table 3: Validation accuracy on Something-Something-V1 for 8 vs 24 frames. $T$ refers to the temporal size of the model, and note that the GrayST gets to access three times as many frames without increasing the model size.

5 Conclusion

We presented two novel video channel sampling strategies: TC Reordering and GrayST. The former re-orders RGB channels to increase temporal information, and the latter uses grayscale images to use more frames resulting in an increased temporal receptive field. In spite of the simplicity of our approaches, we observe a significant boost in performance on multiple challenging datasets. Importantly, these sampling strategies allow us to significantly increase the performance of existing lightweight video networks without increasing the computational cost or requiring any modifications to the underlying network architectures. Our hope is that this will pave the way for alternative sampling strategies. Future work includes developing a temporal aggregation module that is compatible with our sampling strategies for reasoning over much longer time scales.
References


