DeepCube: Transcribing Rubik's Cube Moves with Action Recognition

Junshen Kevin Chen
Stanford University
jkc1@stanford.edu

Wanze Xie
Stanford University
wanzexie@stanford.edu

Zhouheng Sun
Stanford University
sunz@stanford.edu

1. Introduction

Motivation Speedcubing, namely solving a Rubik's Cube as fast as possible, has been a long standing sport since its invention. Currently, it often takes human effort to watch videos of cube solves, and transcribe the moves manually into text, and submit the solution sequence to speedcubing communities such as CubeSolves [1], in order to conduct analysis of solves. With recent advance in video understanding and action recognition, we hypothesize that the process of manually transcribing Rubik's Cube moves can be automated using vision-based deep learning approaches.

Furthermore, while action recognition has been popular and successful in detecting various human activities [22, 14, 16, 21, 9], studies on the fine-grained atomic actions has been relatively scarce, mainly due to the lack of well-defined atomic activities. This motivates the problem of action recognition for Rubik's Cube operations, a subtle and atomic activity that is more challenging to detect.

With our work, we hope to facilitate speedcubing as a sport as well as push forward the frontier of vision-based fine-grained atomic action recognition.

Problem formulation We formulate this problem as a **time-distributed multi-class classification** task. In this project, we aim to produce a model that takes input of a video clip of arbitrary length, and produce an output sequence of the same length, each element denoting one action completed at that time step, including "no action".

To solve this problem, we introduce DeepCube, an egocentric video dataset of a person turning a Rubik's Cube under constrained environment. Our dataset is collected by 3 of us in well-lit environments with static monotonic background (see fig 1). It contains 3 hours of recordings and have well distributed variety of cube moves including rotation and layer turns. To demonstrate the utility of the DeepCube dataset, we explore various methods [10, 6, 15] to tackle the task of action classification.

2. Related Work

While there exists a few research that explores the marriage between rubik's cube and deep learning [3, 18], they are related to constructing solution based on cube's current state with reinforcement learning [18] and robotics [3]. Our work differs in that we want to detect cube's moves rather than constructing solution for it. Therefore our inspirations primarily draw from action recognition in videos.

Action recognition datasets. Previously, most action recognition datasets focus on video classification, such as hollywood2 [17], UCF101 [23], and YouTube-8M [2]. These datasets are suited for tasks that directly maps the whole video clip to an action class. More recently, more works shift toward to temporal localization for activities. The datasets like Charades [22] and ActivityNet [5] contains long videos with multiple actions occurring at different time period. AVA [14] steps further and introduces spatio-temporal localization of actions and highlights the concept of atomic actions. Although most of the aforementioned datasets are allocentric, there exist a few egocentric video datasets like Something-something [12] and EPIC-Kitchens [8], but none focuses on atomic visual activities. This is where our work fits in - an egocentric video datasets for temporal atomic action localization in a specific task.

Action classification and localization. Despite some non deep learning-based works [20, 7], a vast majority of action models with impressive results come from architecture that leverages CNN and RNN modules. One popular structure involves using CNN to extract visual features and feed into an RNN module for sequence learning, which allows variable-length visual input and class label output, known as LRCN [10]. Some modern state-of-the-art models [14, 11] often combines a "backbone" 3D CNN, such as I3D [6] and R3D [15], with a region detector using Faster R-CNN [19], followed by a classifier to predict the action. One distinction in our task is that we put further emphasis on the order of the predicted action sequence, as one mistake in the order causes significant difference in the cube state of next moves.

3. Data

We collect our own original dataset. We first present the format of our data, and then describe the collection process.

Format The entire dataset is a collection of annotated video clips. For each video clip, we annotate each frame with a single label describing the move, from the following three categories:

1. *Zero label*, which means there is no action.

2. *Layer-move labels*. These describe a 90-degree turn of one of the six outer layers of the cube, clockwise or counterclockwise. There are 12 possible labels in total.

3. *Rotation-move labels*. These describe a whole-cube rotation relative to a base orientation which we set during calibration. We assume rotations only happen around the x, y or z axis, and could be 90 deg clockwise or counterclockwise, or 180 deg. There are 9 possible labels in total.

Collection We record moves using a camera facing the cube from the general direction of the solver, such that only the up and front faces are visible most of the time. The solver proceeds to do regular scrambles and solves at medium speed, so that motion blur is minimized.

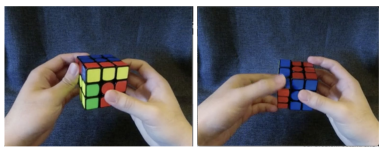


Figure 1: Sample frames from our video dataset, and the right image is performing a move labeled as 'L'.

To obtain the timestamp of each turn, we use commercial bluetooth-connected Rubik's Cube that emits signals for cube moves and orientations, sending a packet at ˜100ms intervals as soon as it detects an action is complete. We write a program that captures the signal while we are recording, and processes the signals to (timestamp, label) pairs. After recording finishes, we play the video and the annotation side by side, and manually align annotation timestamps with the video frames, so that the labels are aligned with the frame at which the move is considered finished by human viewers.

In total there are more than 3 hours of video footage recorded at 30 fps, consisting of more than 20,000 layer and rotation moves in total.

4. Methods

4.1. Input and Output

Input The input to the model are sequences of frames from the recorded video, which we down-sample to be 90 pixels wide and 68 pixels high, in 3 channels of RGB.

Output The output of the model is a sequence of labels corresponding to each frame from the input, semantically denoting "which action **has finished** performing at this frame". Possible actions are:

- No action $\emptyset$
- Cube rotation, denoted as `x,y,z`, their counterclockwise counterparts `x',y',z'`, and their 180 degree counterparts `x2,y2,z2`
- Layer turns to the cube, denoted as `L,R,U,D,F,B`, their counterclockwise counterparts, while their 180 degree counterparts are simply consecutive repeats of itself requiring no new classes

As such, we formulate the problem as a multi-class classification task, with 22 classes.

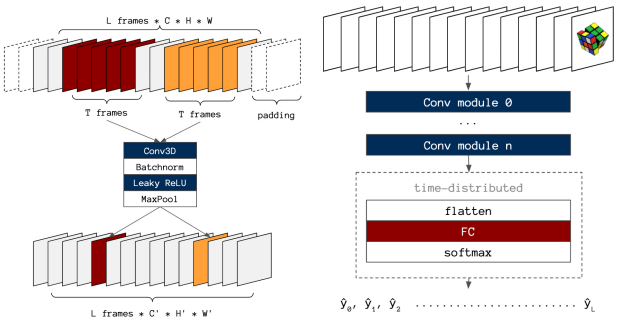
4.2. Baseline

For the baseline model we propose a statistical approach based on frame differences, drawing inspiration from the model [20], which was originally designed to recognize human body actions. For each frame, we get the pixel-by-pixel frame differences from the previous frame, and calculate statistical features such as the center of motion. Features from all the frames are concatenated and passed into a fully connected network. This method does not handle segmentation of moves on its own, so for the purpose of evaluation, we supply readily segmented sequences of 5 frames with a single label denoting the only move (or blank) within these 5 frames. We were not able to get a satisfying result, and we attribute this to the fact that the model was aimed at coarse-grained whole-body actions, and that their object of interest is single-colored so its contour is easily identified. In our model, we aim at hand movement which is much more fine-grained, so a few statistical features may be insufficient for the task. Moreover, this approach does not distinguish between frame difference caused by the shift of the cube pieces, or the shift of the hands. Therefore, we do not report result for this baseline, as we will switch to the more feature-rich, CNN-based approaches as baseline instead.

4.3. 3D Convolutional Action Classifier

Intuitively, because each turn of the cube is spans only several dozen frames, and the motion itself is entirely local to these frames, while being completely independent to any motion before of after it, the model only need to predict any motion given a range of frames.

Figure 2a depicts one convolution module consisting of a 3D convolutional layer, batch normalization, leaky ReLU activation, and max pooling. Before applying the convolution, we always zero-pad the input sequence in the time



(a) Single 3D-conv module (b) 3D-conv Classifier

Figure 2: 3D convolutional action classifier

dimension, depending on the kernel size in the time dimension, such that the output sequence (extracted features) has exactly as many frames as input.

After one or more 3D convolutional modules (figure 2b), we pass the extracted features into a module that flattens each frame, forward through a fully-connect network with ReLU activation, and finally a softmax activation to produce one predicted action per frame. This classification module is time-distributed, meaning that the same weights of the feed-forward network is applied equally to each frame over the time dimension.

4.4. 3D-LRCN

A 3D convolutional classifier has its field-of-view, including in the time dimension, defined as a hyperparameter of kernel size, stride, number of layers etc., so it is not suitable to learn the moves performed in different speed. For example, when the same move is conducted more quickly or slowly, the more than one convolution filter may need to be learned to capture the two types of fast and slow moves of the same kind. This is sub optimal, as outliers tend to cause incorrect predictions.

To get around this problem, we may consider attaching the convolutional modules (ones before the time-distributed feed forward network) from the previous model, and use its extracted features to train a recurrent structure, namely an LSTM, to produce the predicted label, as depicted in fig 3.

This is similar to LRCN [10], which uses a 2D convolution feature extractor. Ideally, a 3D-LRCN should learn to observe each move, fast or slow, while building up an "activation energy", and when the move is complete, release the energy and produce a label, or predict 0 for all other time.

4.5. Collapsed Edit Distance

The goal of this model is to transcribe the moves performed when given frames of videos, and therefore we should evaluate only on whether the predicted transcription is correct, i.e. the correct moves in the correct order. In our

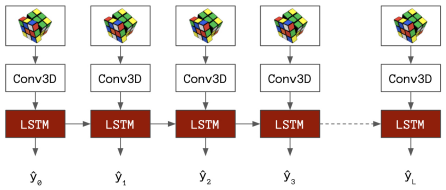


Figure 3: LSTM module with 3D convolutional feature extractor.

Table 1: 3D convolutional classifier, mean metrics

	Train	Dev
Cross Entropy	0.20408	0.68143
Collapsed Edit Distance	1.28571	2.28643

models, because we make a prediction in each timestep, and non-move is a valid prediction, we need to pre-process the output before evaluating the model.

The transcription of movement is **alignment-free**, meaning that it does not require an alignment between the input and the output strictly on each time step, it also does not require each move to only be predicted only one time step. We will collapse sequence by using the method identical to the one in Connectionist Temporal Classification (CTC) [13].

We perform this by merging repeats, and removing spaces, for both the prediction and the label [1]. Then, we take the edit distance, or how many insertion / modification / deletion it takes to edit one into another, between them as an evaluation metric of collapsed edit distance (CED).

For example:

$$\hat{y} = [0, 1, 2, 2, 0, 1], y = [0, 0, 0, 1, 0, 0, 0, 2, 1, 1, 0]$$
$$Collapsed(\hat{y}) = Collapsed(y) = [1, 2, 1], CED(\hat{y}, y) = 0$$

5. Results

At the time of drafting this report, our only working model is a 3D convolutional classifier, with 6 convolutional modules of conv3d-batchnorm-leakyrelu-maxpool3d, and a feed forward network of 512 hidden units. We evaluate on categorical cross entropy averaged for each time stamp, and collapsed edit distance averaged for each sequence of 100 frames. See table 1.

We observe some discrepancy between train and dev metrics, meaning that the model is not being descriptive enough to learn the pattern from training data, such that dev loss is significantly higher than train loss. This is also likely due to the abundance of saddle points in the optimization problem, as we observe the model takes more than 10 epochs before showing any significant loss decrease.

[1] For ground truth label, we do not merge repeats, as they accurately denote repeated actions, we simple remove the spaces.

Qualitatively, we dive into the predicted label and noticed a pattern. When the labeled action is for example $[0, 0, 2, 0, 0]$, the model has the tendency to predict $[0, 2, 2, 2, 0]$, which is accurate after collapsing. However, this signals that the 3D-conv model is not being very effective in learning "what movement has **finished**", but rather "what movement **is happening**". This is likely due to the limitation of a convolutional structure, and the drawback of stacking many modules on top of each other that increases the perceptive field across the time dimension.

This leads to a problem of incorrect collapse when handling consecutive motions, for example when the labeled action is $[0, 0, 2, 0, 2, 0]$ and the model predicts $[0, 0, 2, 2, 2, 0]$, the collapsed result are not identical. We hope that using a recurrence on top of the convolutional structure can help solve this problem.

6. Future Work

More evaluation metrics. So far, we mainly adopted edit distance as the quantitative metric for reasons we stated above. But in standard activity recognition tasks, mean average precision (mAP) is a more commonly used metric. Therefore, we look to evaluate our multi-class classification task using mAP. Similarly, since we also plan to implement temporal action localization using more modern architectures [4, 24], mAP will be the metric for this task as well.

3D-LRCN The 3D-LRCN model is under active development, and we will experiment with different training methods to evaluate the results.

3D-ResNet for action classification. Although 3D CNN has showed some preliminary result for classification, our current approach is frame-based, where each frame maps to an action label. We hope to explore the performance of our dataset, if we were to predict just one action label given the trimmed video. This intuitively should work better given the success of trimmed video action classification on many datasets.

SS-TAD/G-TAD for temporal localization. Temporal action localization is a more complex task than classification. Although our current frame-based approach is able to temporally localize the end point of the action, it is difficult to evaluate if the model has identified the frames that involves the action since collapsed edit distance removes all temporal signals. Therefore, we think it might be worth to explore the task of identifying start and end frames of the predicted action label using some modern temporal localization models like SS-TAD [4] and G-TAD [24].

References

[1] The cubesolves website. http://cubesolv.es/.

[2] Sami Abu-El-Haija, Nisarg Kothari, Joonseok Lee, Paul Natsev, George Toderici, Balakrishnan Varadarajan, and Sudheendra Vijayanarasimhan. Youtube-8m: A large-scale video classification benchmark. *arXiv preprint arXiv:1609.08675*, 2016.

[3] Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

[4] Shyamal Buch, Victor Escorcia, Bernard Ghanem, Li Fei-Fei, and Juan Carlos Niebles. End-to-end, single-stream temporal action detection in untrimmed videos. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2017.

[5] Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *Proceedings of the ieee conference on computer vision and pattern recognition*, pages 961–970, 2015.

[6] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308, 2017.

[7] Kyungeun Cho, Hyungje Cho, and Kyhyun Um. Human action recognition by inference of stochastic regular grammars. In Ana Fred, Terry M. Caelli, Robert P. W. Duin, Aurélio C. Campilho, and Dick de Ridder, editors, *Structural, Syntactic, and Statistical Pattern Recognition*, pages 388–396, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[8] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, et al. Scaling egocentric vision: The epic-kitchens dataset. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 720–736, 2018.

[9] Ali Diba, Mohsen Fayyaz, Vivek Sharma, Manohar Paluri, Jurgen Gall, Rainer Stiefelhagen, and Luc Van Gool. Holistic large scale video understanding. *arXiv preprint arXiv:1904.11451*, 2019.

[10] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2625–2634, 2015.

[11] Rohit Girdhar, João Carreira, Carl Doersch, and Andrew Zisserman. A better baseline for ava. *arXiv preprint arXiv:1807.10066*, 2018.

[12] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. The" something something" video database for learning and evaluating visual common sense. In *ICCV*, volume 1, page 5, 2017.

[13] Alex Graves and Jürgen Schmidhuber. 2005 special issue: Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Netw.*, 18(5–6):602–610, June 2005.

[14] Chunhui Gu, Chen Sun, David A Ross, Carl Vondrick, Caroline Pantofaru, Yeqing Li, Sudheendra Vijayanarasimhan, George Toderici, Susanna Ricco, Rahul Sukthankar, et al. Ava: A video dataset of spatio-temporally localized atomic visual actions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6047–6056, 2018.

[15] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Learning spatio-temporal features with 3d residual networks for action recognition. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 3154–3160, 2017.

[16] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

[17] Marcin Marszalek, Ivan Laptev, and Cordelia Schmid. Actions in context. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2929–2936. IEEE, 2009.

[18] Stephen McAleer, Forest Agostinelli, Alexander Shmakov, and Pierre Baldi. Solving the rubik's cube without human knowledge. *arXiv preprint arXiv:1805.07470*, 2018.

[19] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[20] Samy Sadek, Ayoub Al-Hamadi, Bernd Michaelis, and Usama Sayed. A fast statistical approach for human activity recognition. 2012.

[21] Dian Shao, Yue Zhao, Bo Dai, and Dahua Lin. Finegym: A hierarchical video dataset for fine-grained action understanding. *arXiv preprint arXiv:2004.06704*, 2020.

[22] Gunnar A Sigurdsson, Gül Varol, Xiaolong Wang, Ali Farhadi, Ivan Laptev, and Abhinav Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding. In *European Conference on Computer Vision*, pages 510–526. Springer, 2016.

[23] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[24] Mengmeng Xu, Chen Zhao, David S. Rojas, Ali Thabet, and Bernard Ghanem. G-tad: Sub-graph localization for temporal action detection, 2019.