Benchmarking the Robustness of CNN-based Spatial-Temporal Models (Supplementary material)

Chenyu Yi[1*] **Siyuan Yang**[1,2*] **Haoliang Li**[3] **Yap-Peng Tan**[1] **Alex C. Kot**[1]
[1]School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore
[2]Interdisciplinary Graduate Programme, Nanyang Technological University, Singapore
[3]Department of Electrical Engineering, City University of Hong Kong, China
{yich0003,siyuan005}@e.ntu.edu.sg haoliang.li@cityu.edu.hk {eyptan,eackot}@ntu.edu.sg

1 Creating Benchmark Datasets

Since our work synthesizes corruptions on existing Kinetics400 and Something-Something-V2 datasets which are available online, we provide complete code for generating all types of corruptions on them. The code is available at `https://github.com/Newbeeyoung/Video-Corruption-Robustness`. The parameters for corruptions with different severity levels are shown in corresponding code functions. In this URL, we list the detail of samples extracted from the official Kinetics400 and Something-Something-V2 datasets. It includes the class name of data, the index of data, and the number of frames in each video data. We also provide code for pre-processing existing datasets Kinetcs and Something-Something-V2, including converting video to HDF5 file and creating JSON files for dataloading. Furthermore, we provide a structured framework of training and evaluating the spatial-temproal models. With the code, researchers can recreate the same benchmark datasets with following steps:

1. Download Kinetics400 and Something-Something-V2 from the public websites
2. Extract Mini Kinetics and Mini SSV2 based on class name list
3. Preprocess video data for data loading
4. Apply the proposed corruptions on the Mini Kinetics and Mini SSV2 validation datasets.

2 Implementation of Proposed Corruptions

We have provided complete code in the Github repository: `create_mini_kinetics_c.py` and `create_mini_ssv2_c.py` for researchers to generate our proposed corruptions. For the simulation of Packet Loss, we added an extra folder: `packet-loss-simulation`. We also present the implementation detail of each corruption in Table 1. Only the parameters of Motion Blur and Frame Rate are different for these two datasets, because Kinetics has fps of 24 but SSV2 has fps of 12. When we determine the parameters of each corruption, we make sure the corrupted videos are still human recognisable even if the corruptions have the highest level of severity.

3 Benchmark Maintenance

We will maintain our code for generating the benchmark in this link: `https://github.com/Newbeeyoung/Video-Corruption-Robustness`. Any enquiry on the benchmark creation and evaluation can be sent to yich0003@ntu.edu.sg. Besides, we will update the benchmark leaderboard in the link for any work beating the state-of-the-art performance on benchmark. The leaderboard will consist of approach, reference, backbone, input length, sampling method and the approach

*equal contribution

35th Conference on Neural Information Processing Systems (NeurIPS 2021) Track on Datasets and Benchmarks.

Table 1: Implementation detail of corruptions

Dataset	Corruption	Parameter	Severity				
			1	2	3	4	5
Mini Kinetics-C	Shot Noise	Photons number	60	25	12	5	3
	Rain	(Density, Length)	(65,10)	(65,30)	(65, 60)	(100,60)	(125,80)
	Fog	(Thickness, Smoothness)	(1.5, 2)	(2., 2)	(2.5, 1.7)	(2.5, 1.5)	(3., 1.4)
	Contrast	Difference portion	0.5	0.4	0.3	0.2	0.1
	Brightness	Addition in HSV space	0.1	0.2	0.3	0.4	0.5
	Saturate	Manipulation in HSV space	(0.3, 0)	(0.1, 0)	(2, 0)	(5, 0.1)	(20, 0.2)
	Motion Blur	Number of correlated frames	3	5	7	9	11
	ABR	Rate of bit rate	2	4	8	16	32
	CRF	Sane value	27	33	39	45	51
	Bit Error	Bit error ratio	1/100000	1/50000	1/30000	1/20000	1/10000
	Packet Loss	Packet loss ratio(%)	1	2	3	4	6
	Frame Rate	FPS	20	16	12	9	6
Mini SSV2-C	Shot Noise	Photons number	60	25	12	5	3
	Rain	(Density, Length)	(65,10)	(65,30)	(65, 60)	(100,60)	(125,80)
	Fog	(Thickness, Smoothness)	(1.5, 2)	(2., 2)	(2.5, 1.7)	(2.5, 1.5)	(3., 1.4)
	Contrast	Difference portion	0.5	0.4	0.3	0.2	0.1
	Brightness	Addition in HSV space	0.1	0.2	0.3	0.4	0.5
	Saturate	Manipulation in HSV space	(0.3, 0)	(0.1, 0)	(2, 0)	(5, 0.1)	(20, 0.2)
	Motion Blur	Number of correlated frames	1	2	3	4	6
	ABR	Rate of bit rate	2	4	8	16	32
	CRF	Sane value	27	33	39	45	51
	Bit Error	Bit error ratio	1/100000	1/50000	1/30000	1/20000	1/10000
	Packet Loss	Packet loss ratio (%)	1	2	3	4	6
	Frame Rate	FPS	10	8	6	4	2

performance, including clean accuracy, mPC and rPC. Any result can be submitted to us via pull request in the link.

4 Sample of Mini SSV2-C

We show the visualization samples from Mini SSV2-C in Figure 1. From the samples shown, Mini SSV2 consists of first-person view human action videos, while the Mini Kinetics is constructed by third-person view human action videos. With the variety in Mini Kinetics and Mini SSV2, we can evaluate the corruption robustness of models in video classification comprehensively.

5 List of Mini Kinetics Video Classes

The complete list of 200 classes and their corresponding Parent-Child Group grouping is shown in Table 2. The number of videos for each action class is given by the number in brackets following each class name. The number of child-actions of each action group is shown by the number in brackets following each action group name. The class label in blue color means that it belongs to multiple action groups and has occurred above. The class label in red color means that it doesn't belong to any action group. The class label with * sign means that it belongs to **40**-class Mini Kinetics.

Table 2: List of Mini Kinetics Video Classes.

Class Labels	**Action Groups**
arranging flowers* (583) blowing glass (1145) brush painting (532) carving pumpkin (711) getting a tattoo (737)	arts and crafts (**5**)
high jump (954) parkour (504)	athletics - jumping (**3**)

pole vault (984)	
catching or throwing frisbee (1060) disc golfing* (565) javelin throw* (912) throwing axe (816) throwing ball (634)	athletics - throwing + launching (**5**)
pumping gas (544)	auto maintenance (**1**)
catching or throwing softball (842) dunking basketball* (1105) golf driving* (836) golf putting (1081) hitting baseball (1071) juggling soccer ball* (484) passing American football (not in game) (1045) playing kickball (468) playing squash or racquetball (980) playing tennis (1144) playing volleyball (804) shooting basketball (595) shooting goal (soccer) (444)	ball sports (**13**)
applauding* (411) bending back* (635) drumming fingers (409) finger snapping (825) headbanging (1090) pumping fist (1009) shaking head (885) swinging legs* (409)	body motions (**8**)
cleaning floor (874) cleaning pool (447) cleaning shoes (706) cleaning toilet (576) cleaning windows (695) making bed (679) setting table (478)	cleaning (**7**)
folding clothes (695) folding napkins (874) ironing (535) making bed (679) tying bow tie (387)	cloths (**5**)
answering questions (478) bartending (601) crying* (1037) giving or receiving award (953) news anchoring (420) presenting weather forecast (1050)	communication (**6**)
baking cookies (927) barbequing (1070) breading or breadcrumbing (454) cooking chicken (1000) cutting pineapple (712) cutting watermelon (767) grinding meat* (415)	cooking (**7**)
belly dancing (1115) country line dancing (1015) dancing ballet (1144) dancing charleston (721) dancing gangnam style (836) dancing macarena (958)	dancing (**11**)

marching* (1146) swing dancing (512) tango dancing (1114) tap dancing (947) zumba (1093)	
bartending (601) dining* (671) drinking beer (575) eating burger (864) eating cake (494) eating carrots (516) eating chips (749) eating hotdog* (570) eating spaghetti* (1145) eating watermelon (550)	eating + drinking (**10**)
assembling computer (542) texting (704) using computer (937) using remote controller (not gaming) (549)	electronics (**4**)
blowing leaves (405) carving pumpkin (711) mowing lawn* (1147) watering plants (680)	garden + plants (**4**)
golf driving* (836) golf putting (1081)	golf (**2**)
curling hair* (855) fixing hair (676) waxing back (537)	hair (**3**)
applauding* (411) cutting nails (560) drumming fingers (409) finger snapping (825) pumping fist (1009) washing hands* (916)	hand (**6**)
balloon blowing (826) beatboxing (943) blowing nose (597) blowing out candles* (1150) gargling (430) headbanging (1090) shaking head (885) singing (1147) smoking hookah (857)	head + mouth (**9**)
abseiling (1146) bungee jumping* (1056) climbing ladder (662) diving cliff* (1075) ice climbing (845) jumping into pool (1133) paragliding (800) slacklining (790) springboard diving (406) swinging on something (482) trapezing (786)	heights (**11**)
bee keeping (430) feeding goats (1027) holding snake (430) ice fishing (555) milking cow (980)	interacting with animals (**9**)

riding elephant (1104) riding or walking with horse (1131) training dog (481) walking the dog (1145)	
contact juggling (1135) hula hooping (1129) juggling soccer ball* (484) spinning poi (1134)	juggling (**4**)
filling eyebrows (1085) getting a tattoo (737)	makeup (**2**)
high kick (825) punching bag* (1150) side kick (991) sword fighting (473) tai chi (1070) wrestling (488)	martial arts (**6**)
garbage collecting (441) laying bricks* (432) moving furniture (426) unloading truck (406)	miscellaneous (**4**)
driving car (1118) faceplanting (441) jogging (417) motorcycling* (1142) parkour (504) pushing car (1069) riding a bike (476) skateboarding (1139) surfing crowd (876) using segway (387)	mobility - land (**10**)
crossing river (951) diving cliff* (1075) jumping into pool (1133) scuba diving* (968) snorkeling (1012) springboard diving (406) swimming backstroke (1077) swimming butterfly stroke* (678) water sliding (420)	mobility - water (**9**)
beatboxing (943) playing accordion* (925) playing bagpipes (838) playing bass guitar (1135) playing cello (1081) playing didgeridoo (787) playing guitar (1135) playing harp (1149) playing keyboard* (715) playing organ* (672) playing piano* (691) playing recorder (1148) playing saxophone (961) playing trumpet (989) playing ukulele* (1146) playing violin* (1142) playing xylophone (746) singing (1147)	music (**18**)
bookbinding (914) counting money (674)	paper (**6**)

folding napkins (874) ripping paper (605) shredding paper* (403) writing (735)	
taking a shower (378) washing hands* (916)	personal hygiene (**2**)
flying kite (1063) playing cards (737) playing chess* (850) playing poker (1134) rock scissors paper (424) shuffling cards (828) skipping rope (488)	playing games (**7**)
catching or throwing softball (842) hitting baseball (1071) playing squash or racquetball (980) playing tennis (1144)	racquet + bat sports (**4**)
biking through snow (1052) bobsledding (605) hockey stop (468) ice climbing (845) ice fishing (555) making snowman (756) shoveling snow (879) ski jumping (1051) snowboarding (937) snowkiting (1145) snowmobiling (601) tobogganing (1147)	snow + ice (**12**)
swimming backstroke (1077) swimming butterfly stroke* (678)	swimming (**2**)
massaging legs (592) shaking hands* (640) tickling (610)	touching person (**3**)
bending metal* (410) pumping gas (544) sanding floor* (574) sharpening knives (424)	using tools (**4**)
kitesurfing (794) parasailing (762) windsurfing (1114)	water sports (**3**)
waxing back (537)	waxing (**1**)
bench pressing (1106) deadlifting* (805) exercising arm* (416) lunge* (759) pull ups (1121) yoga (1140)	**No Parent-Group (6)**

6 List of Mini SSV2 Video Classes

The complete list of 87 classes and their corresponding action-groups is shown in Table 3. The number of videos for each action class is given by the number in brackets following each class name. The number of child-actions of each action group is shown by the number in brackets following each action group name.

Figure 1: Our proposed corruption robustness benchmarks consists of 12 types of corruptions with five levels of severity for each video. In the visualization examples of Mini SSV2-C, we use uniform sampling to extract 3 frames from each corrupted video, the sampling interval is 20 frames.

Table 3: List of Mini SSV2 Video Classes.

Class Labels	**Action Groups**
Trying but failing to attach [something] to [something] because it doesn't stick (660) Attaching [something] to [something] (1227)	Attaching/Trying to attach (**2**)
Approaching [something] with your camera (1349) Moving away from [something] with your camera (1199) Turning the camera right while filming [something] (1239) Turning the camera left while filming [something] (1239) Moving [something] away from the camera (986)	Camera motions (**6**)

Turning the camera downwards while filming [something] (976)	
[Something] colliding with [something] and both are being deflected (653)	Collisions of objects (**1**)
Uncovering [something] (3004) Covering [something] with [something](3530)	Covering (**2**)
Putting [something similar to other things that are already on the table] (2339) Taking [one of many similar things on the table] (2969)	Crowd of things (**2**)
Dropping [something] onto [something] (1623) Dropping [something] next to [something] (1232) Dropping [something] in front of [something] (1222)	Dropping something (**3**)
Showing [something] behind [something] (2315)	Filming objects, without any actions (**1**)
Folding [something] (1542)	Folding something (**1**)
Hitting [something] with [something] (2234)	Hitting something with something (**1**)
Holding [something] behind [something] (1374) Holding [something] over [something] (1804)	Holding something (**2**)
Lifting up one end of [something], then letting it drop down (1850) Lifting up one end of [something] without letting it drop down (1613) Lifting [something] up completely, then letting it drop down (1851) Lifting [something] up completely without letting it drop down (1906)	Lifting and (not) dropping something (**4**)
Tilting [something] with [something] on it until it falls off (1272)	Lifting/Tilting objects with other objects on them (**1**)
Moving [something] closer to [something] (1426)	Moving two objects relative to each other (**1**)
Touching (without moving) [part] of [something] (1763)	Moving/Touching a part of something (**1**)
Opening [something] (1869) Pretending to open [something] without actually opening it (1911)	Opening or closing something (**2**)
Picking [something] up (1456)	Picking something up (**1**)
Plugging [something] into [something] but pulling it right out as you remove your hand (1176) Plugging [something] into [something] (2252)	Plugging something into something (**2**)
Poking [something] so it slightly moves (1599) Poking [something] so lightly that it doesn't or almost doesn't move (2430) Poking a stack of [something] without the stack collapsing (276) Poking a hole into [something soft] (258) Pretending to poke [something] (754) Poking a stack of [something] so the stack collapses (367) Poking a hole into [some substance] (115)	Poking something (**7**)
Pouring [something] onto [something] (403) Pouring [something] into [something] (1530) Pouring [something] into [something] until it overflows (352)	Pouring something (**3**)
Pulling [something] from right to left (1886) Pulling [something] from left to right (1908)	Pulling something (**2**)
Pulling two ends of [something] so that it separates into two pieces (313)	Pulling two ends of something (**1**)
Pushing [something] with [something] (1804) Pushing [something] off of [something] (687) Pushing [something] so that it slightly moves (2418)	Pushing something (**3**)
Laying [something] on the table on its side, not upright (950)	Putting something upright/on its side (**1**)
Putting [something] onto [something else that cannot support it] so it falls down (442) Taking [something] out of [something] (2259) Pretending to put [something] onto [something] (740) Putting [something] and [something] on the table (1353) Putting [something] onto [something] (1850) Pretending to put [something] underneath [something](373) Putting [something] next to [something] (2431) Putting [something] behind [something] (1428) Putting [something] on the edge of [something] so it is not supported and falls down (638) Removing [something], revealing [something] behind (1069) Pretending to put [something] next to [something] (1297)	Putting/Taking objects into/out of/next to other objects (**11**)
Letting [something] roll along a flat surface (1163) Rolling [something] on a flat surface (1773) Putting [something] that can't roll onto a slanted surface, so it slides down (442) Lifting a surface with [something] on it but not enough for it to slide down (268) Putting [something] on a flat surface without letting it roll (553)	Rolling and sliding something (**5**)
Showing [something] to the camera (1061) Showing a photo of [something] to the camera (916)	Showing objects and photos of objects (**2**)
Showing that [something] is empty (2209)	Showing that something is full/empty (**1**)
[Something] falling like a feather or paper (1858)	Something falling (**1**)
Moving [something] and [something] so they collide with each other (577)	Something passing/hitting another thing (**1**)
Spinning [something] so it continues spinning (1168)	Spinning something (**1**)
Spreading [something] onto [something] (535) Pretending to spread 'air' onto [something] (225)	Spreading something onto something (**2**)
Pretending to sprinkle 'air' onto [something] (543)	Sprinkling something onto something (**1**)
Putting number of [something] onto [something] (1180)	Stacking or placing N things (**1**)
Stuffing [something] into [something] (1998)	Stuffing/Taking out (**1**)
Pretending to take [something] from somewhere (1437)	Taking something (**1**)
Tearing [something] just a little bit (2025)	Tearing something (**1**)
Throwing [something] (2626) Throwing [something] in the air and catching it (1177)	Throwing something (**5**)

Pretending to throw [something] 1019 Throwing [something] in the air and letting it fall (1038) Throwing [something] onto a surface (1035)	
Tipping [something] over (896)	Tipping something over (**1**)
Twisting (wringing) [something] wet until water comes out (408) Twisting [something] (1131)	Twisting something (**2**)

7 More Training Details

We follow the training protocol in [2], as shown in Table 4, for I3D, S3D, 3D-ResNet 18, Inception V1 and Resnet 18 in our experiments. We progressively train the model with different input frames. We first train a starter model using 8 frames. The model is either inflated with (3D models) or initialized from (2D models) its corresponding ImageNet pre-trained weight. We then finetune the model using more $N(N = 16, 32, 64)$ frames from the model using $N/2$ frames. For the started model, we trained 75 epochs with cosine annealing learning rate schedule starting with 0.01. For models with more input frames, the learning rate is also set as 0.01 and is divided by 10 at every 15 epochs. The training process is ended at the 50_{th} epoch. For TAM, 3D-ResNet 50, SlowFast, and X3D models, we only trained these models using 32 frames. For TimeSformer, we use 32 frames to train on Kinetics while using 16 frames to train on SSV2. Because the performance using 16 frames is better than 32 frames. We use synchronized SGD with momentum 0.9 and weight decay 0.0001 for all models mentioned above.

In video classification, the model can be evaluated at clip level and video level settings using uniform sampling and dense sampling. For experiments in this supplementary material, we use both uniform sampling and dense sampling strategies. At clip level, we divide a video into multiple segments of equal duration, randomly choose one frame from the first video segment, and extract frames with fixed stride for uniform sampling. For dense sampling, we randomly choose one frame from the whole video and use the following frames as input. At the video level, we extract the first N frames and use each frame as the first frame of each clip for uniform sampling, where N is the number of clips used in video level uniform sampling. For dense sampling, we crop the video evenly and extract M clips with continuous frames. In our experiments, we use $N = 4$ and $M = 8$ because the performance of uniform sampling saturates faster. For all types of corruption, we use the same settings to evaluate the models.

Table 4: Training protocol

	8-frame	16-frame	32-frame	64-frame
Weight Init.	ImageNet	8-frame	16-frame	32-frame
Epochs	75	50	50	50
Learning rate	0.01			
LR scheduler	cosine	multisteps	multisteps	multisteps
Weight decay	0.0005			
Optimizer	Synchronized SGD with moment 0.9			

8 Trade off between Robustness, Generalization and Efficiency of 3D CNN-Based Models

In our benchmark study, we have plot mPC and rPC versus clean accuracy and FLOPs for the SOTA approaches. The correlation between mPC, rPC and generalization is obvious. However, the correlation between rPC and FLOPs is not significant. In this section, we only use 3D CNN-based approaches for visualization in Figure 2 and Figure 3, where these approaches have similar architectures. We discard the results of TimeSformer(transformer-based) and TAM(2D CNN-based). Figure 3 shows the correlation of rPC and FLOPs is more significant. Combined with the result of mPC, it indicates there is trade off between robustness and efficiency of 3D CNN-based spatial-temporal models.

Table 5: Corruption robustness of 3D ResNet-18 on Mini Kinetics-C with additive Gaussian noise.

				Spatial						Temporal					
Model	Clean	mPC	rPC	Shot	Rain	Fog	Contrast	Brightness	Saturate	Motion	Frame Rate	ABR	CRF	Bit Error	Packet Loss
Shot Noise	71.5	**62.3**	**87.1**	**74.0**	**68.5**	**33.2**	**45.0**	**64.2**	**59.4**	**66.1**	**71.3**	**69.3**	**66.7**	**63.3**	**66.6**
Rain	73.7	48.2	65.4	47.3	55.0	15.7	26.0	53.3	48.9	53.4	62.2	58.6	55.4	49.3	53.7
Fog	41.6	27.9	67.1	27.4	32.1	22.3	18.2	26.7	20.6	35.8	35.1	31.6	30.8	26.4	27.9
Contrast	49.6	31.3	63.1	26.3	39.5	20.5	25.4	31.2	26.3	39.7	36.7	35.5	33.8	29.4	30.7
Brightness	71.2	45.1	63.3	41.6	44.5	14.0	19.4	61.5	49.0	49.0	58.7	54.6	51.6	46.9	50.7
Saturate	75.4	47.1	62.5	44.3	47.5	12.2	18.7	58.4	54.3	51.0	63.1	58.0	54.5	49.8	53.9
Motion Blur	71.3	41.7	58.5	38.4	38.8	12.5	16.6	49.5	42.4	56.0	55.3	52.2	49.3	43.3	46.4
Frame Rate	**80.3**	41.7	51.9	38.3	35.9	10.7	14.4	53.8	42.8	43.6	59.6	52.8	48.0	48.3	52.0
ABR	79.4	44.6	56.2	43.2	41.5	10.8	15.0	53.9	48.3	49.9	60.0	56.9	53.2	48.8	53.2
CRF	78.8	43.6	55.3	40.0	37.7	8.7	15.0	55.5	47.5	48.1	60.5	55.7	51.8	48.5	54.1
Bit Error	77.8	43.2	55.5	39.1	38.2	8.0	12.7	56.6	49.0	43.2	60.8	53.6	49.4	52.2	55.9
Packet Loss	77.3	44.5	57.6	44.9	41.4	11.4	14.9	55.5	48.7	45.8	61.4	54.0	49.4	51.1	55.6
Clean	78.9	46.0	58.3	44.9	47.2	12.9	18.3	54.1	46.9	49.3	63.9	57.3	52.2	49.9	54.6

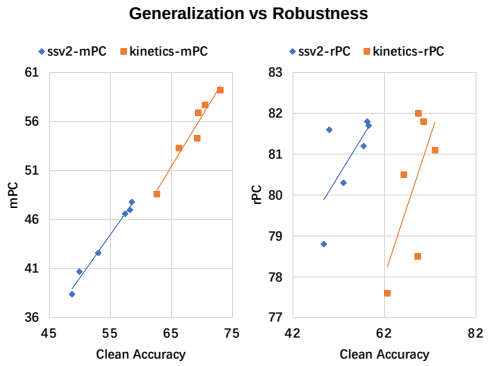

Figure 2: The generalization and robustness of the 3D CNN-based approaches.

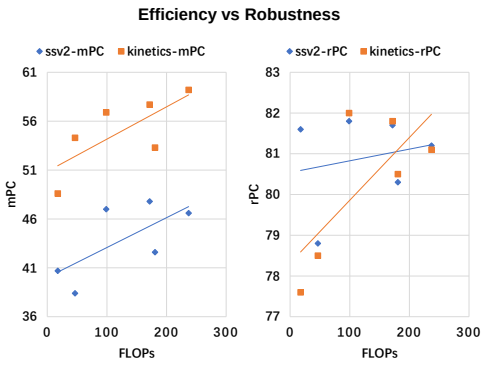

Figure 3: The efficiency and robustness of the 3D CNN-based approaches.

9 Ablation Study

9.1 2D-3D Module

For comparing the corruption robustness of 2D and 3D models, we use InceptionV1 [6] and ResNet18 [5] as backbone, and input length of 8 and 16 at the clip level. With the same backbone, input length and training approach, the 3D models consistently outperform the 2D models in terms of rPC on both datasets, as shown in Table 6. On Mini Kinetics-C, the clean accuracy and mPC of 2D models remain the same when the input length increase, but the accuracy of 3D models increase by 4% $\sim$ 6%. On Mini SSV2-C, the 3D models improve both clean accuracy and mPC by 18% $\sim$ 22%.

9.2 Input Sampling Strategy

As we introduced in the evaluation protocol, uniform sampling and dense sampling are widely used in video classification tasks. To assess the effect of sampling strategy on model corruption robustness, we train three classic video classification models I3D [1], 3D ResNet-18 [4] and S3D [7] using uniform sampling and dense sampling on both Mini Kinetics and Mini SSV2. In each setting, we use 8,16,32,64 frames as input. We also evaluate the generalization and robustness of models at clip and video levels. The clean accuracy and mPC of models on the benchmark are shown in Figure 4 and Figure 5.

Figure 4 shows that uniform sampling outperforms dense sampling on all settings at clip levels on Mini Kinetics-C. The mPC of models using uniform sampling are 4$\sim$13% higher than dense sampling. Because uniform sampling at clip level extracts the frame from the whole video evenly, it enables the model to capture longer-term temporal information than dense sampling at the clip level. At the video level, uniform sampling yields similar results to clip level. Dense sampling boosts by 9$\sim$11% on mPC, comparing to clip level. However, all the models using uniform sampling perform better than

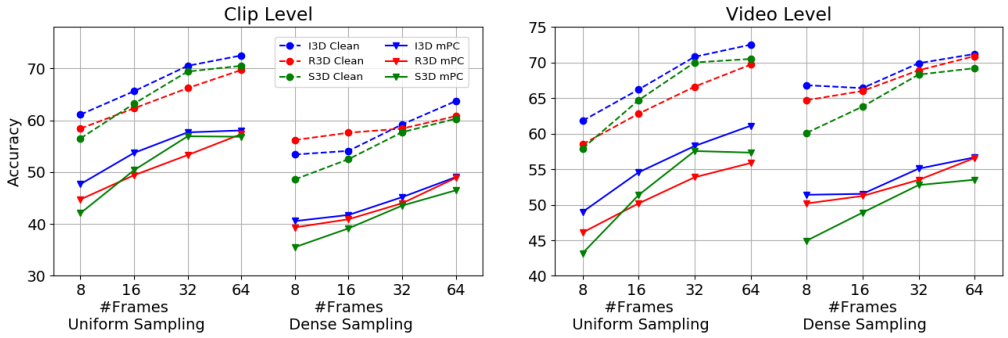

Figure 4: Robustness comparison of uniform sampling and dense sampling at clip level and video level on Mini Kinetics-C. For each setting, we use 8,16 ,32 , 64 frames as input. We use I3D, S3D and R3D(3D-ResNet) for training and evaluation. R3D use a backbone of ResNet18. The dash line indicates the accuracy of models on clean validation data. The solid line indicates the mPC of models on corrupted data.

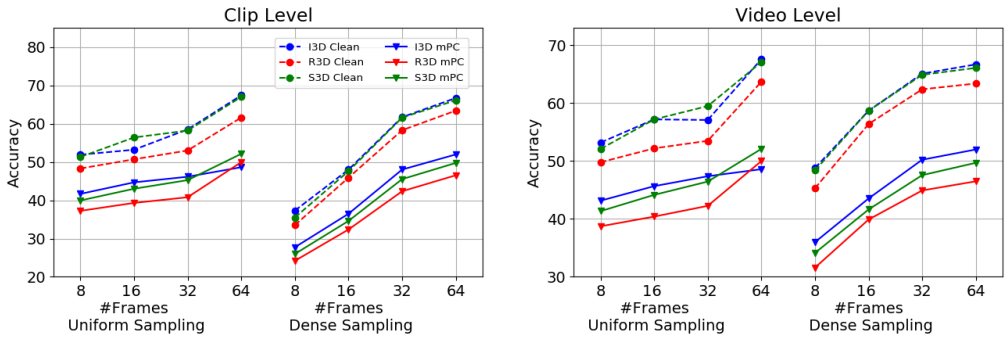

Figure 5: Robustness comparison of uniform sampling and dense sampling at different input frames on Mini SSV2-C. We use the same setting and networks as comparison on Mini Kinetics-C.

dense sampling at frames of 32 and 64, except 3D ResNet-18 at 64 frames. On Mini SSV2-C, the uniform sampling of input makes model more robust than dense sampling at 8 frames and 16 frames at both clip and video levels. The mPC of models using different sampling methods are similar at 32 frames and 64 frames. Most of the videos from SSV2 dataset have 30 to 60 frames, both sampling methods will extract similar frames at input length larger than 32.

9.3 Input Length

From Figure 4 and Figure 5, we find the mPC of models increase with input frame length on both datasets. We also find the same trend of rPC from Table 6. The rPC of models using 16 frames is larger than models using 8 frames under all settings. It indicates that models using longer frames input with more temporal information can improve the robustness of models.

9.4 Impact of Batch Size and Training Epochs

Besides 2D-3D module, input sampling strategy, and input length, we study the impact of more general hyper-parameters including batch size (16 and 32) and training epochs (25 and 50). We show the results in Table 7 and Table 8. We find these two general hyper-parameters have less impact on the performance comparing to the hyper-parameters which are specific for video classification. The clean accuracy and mPC of models are similar while the hyper-parameters differ by large.

Table 6: Clean Accuracy and mPC of 2D and 3D Models

Frames	Backbone	Approach	Mini Kinetics-C			Mini SSV2-C		
			Clean Acc	mPC	rPC	Clean Acc	mPC	rPC
8	InceptionV1	3D	61.1	47.7	**78.1**	51.9	41.7	**80.3**
		2D	68.1	52.0	76.4	33.4	22.1	66.2
8	ResNet18	3D	58.4	44.9	**76.9**	48.3	37.2	**77.0**
		2D	66.9	50.5	75.5	30.3	18.6	61.4
16	InceptionV1	3D	65.6	53.7	**81.9**	53.2	44.7	**84.0**
		2D	67.8	52.7	77.7	34.6	22.7	65.6
16	ResNet18	3D	62.4	49.4	**79.2**	50.7	39.3	**77.5**
		2D	67.4	51.6	76.6	30.6	19.9	65.0

Table 7: Corruption robustness of 3D ResNet-18 trained with different batch size. The bs stands for batch size.

				Spatial						Temporal					
Batch Size	Clean	mPC	rPC	Shot	Rain	Fog	Contrast	Brightness	Saturate	Motion	Frame Rate	ABR	CRF	Bit Error	Packet Loss
bs=16	65.2	54.2	83.1	52.4	52.1	49.2	44.7	45.0	58.4	53.3	64.9	59.8	56.1	56.0	59.0
bs=32	66.2	53.3	80.5	47.7	45.8	40.5	43.0	58.9	53.5	53.4	65.1	60.1	56.4	55.8	59.3

9.5 Number of Clips in Video Level Evaluation

The evaluation on Mini Kinetics-C and Mini SSV2-C requires 60 times of computation cost on original validation datasets. It is necessary to find the proper number of clips at the video level in practice, which minimizes the trade-off between performance and efficiency. To find the number of clips, we train the models on Mini Kinetics using 8 frames input and evaluate them on Mini Kinetics-C at the video level. Figure 6 shows that the models using uniform sampling saturate faster than dense sampling. The performance improvement is subtle after 4 clips. Dense sampling requires more number of clips at the video level. The model performance saturates at 8 clips. As a result, we use 4 clips for uniform sampling and 8 clips for dense sampling for all the video level evaluations.

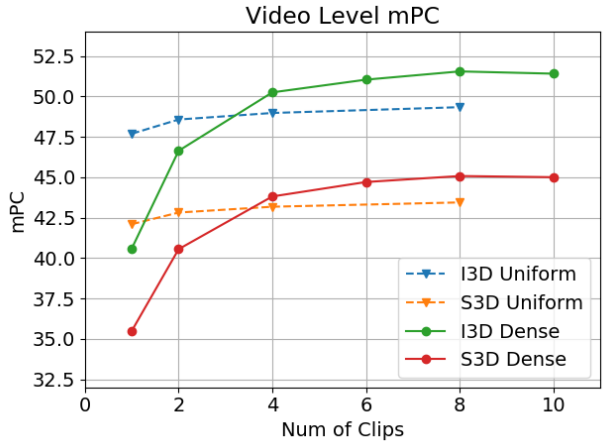

Figure 6: mPC of I3D and S3D using different number of clips at video level evaluation.

10 Robustness on Combination of Spatial and Temporal Corruptions

Besides single corruption, different types of corruption may happen simultaneously in nature. To mimic the random process that occurs in nature, we also evaluate the performance of models on each corruption with additive white Gaussian noise. The Gaussian noise has a standard deviation of 0.1. Table 5 shows that Gaussian noise can degrade the performance of models significantly. The model

Table 8: Corruption robustness of 3D ResNet-18 trained with different training epochs.

				Spatial						Temporal					
Epoch	Clean	mPC	rPC	Shot	Rain	Fog	Contrast	Brightness	Saturate	Motion	Frame Rate	ABR	CRF	Bit Error	Packet Loss
Epochs=25	65.2	54.6	83.7	50.1	54.6	48.1	43.2	44.7	59.2	54.5	65.8	60.7	57.3	56.9	59.8
Epochs=50	66.2	53.3	80.5	47.7	45.8	40.5	43.0	58.9	53.5	53.4	65.1	60.1	56.4	55.8	59.3

trained with shot noise obtains the best performance on all types of corruptions due to its similarity to Gaussian noise.

11 Full Corruption Robustness Results

The classification accuracy on each corruption with respect to each severity level in Mini Kinetics-C is in Figure 7, the accuracy on corruptions in Mini SSV2-C is in Figure 8. Both the figures shows the degrading performance when we increase the severity level of corruptions. When we compare the performance of models on Mini Kinetic-C and Mini SSV2-C horizontally, they present obvious difference in degradation on motion blur and frame rate conversion. The model trained on Mini SSV2 are more sensitive to these two temporal corruptions. Especially for frame rate conversion, the classification accuracy of models drops around 40% on Mini SSV2-C, while the performance of models on Mini Kinetics-C maintains the same.

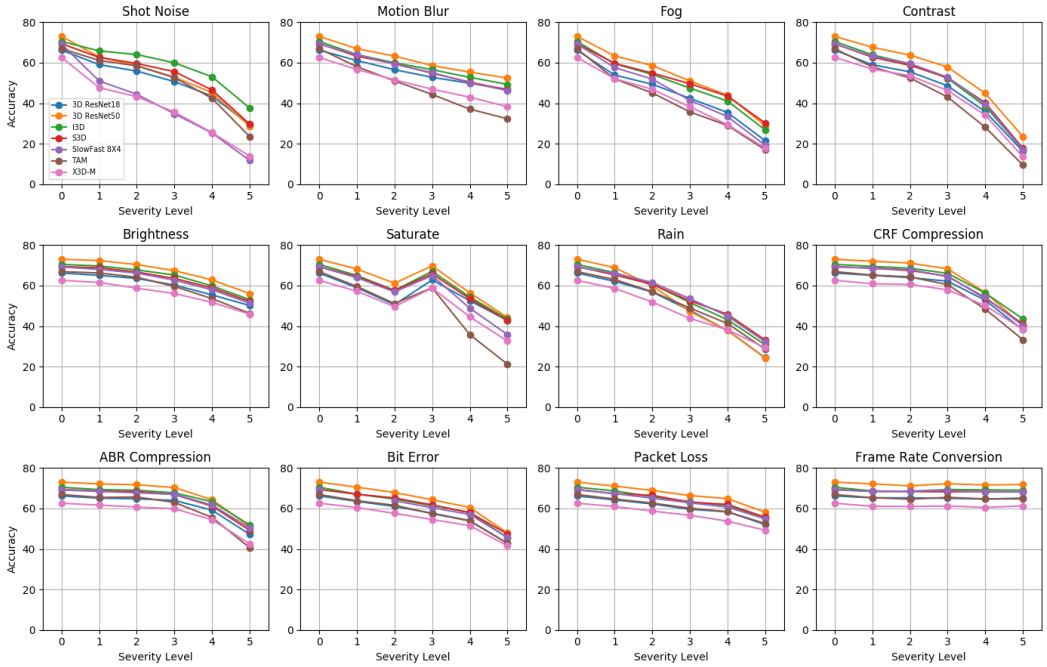

Figure 7: Model accuracy for SOTA approaches with respect to the video corruptions in Mini Kinetics-C. We evaluate the models at 5 level of severity. Severity level 0 indicates the accuracy on clean data.

12 Sample of Gaussian Noise Corrupted Video

We show the clean and Gaussian noise corrupted video samples in Figure 9. Following the setting of [3], we use standard deviation of 0.1 to generate Gaussian noise for training, because we apply corruptions with similar level of severity as ImageNet-C. To study the impact of Gaussian noise more comprehensively, we also apply the standard deviation of 0.2 to explore the trend.

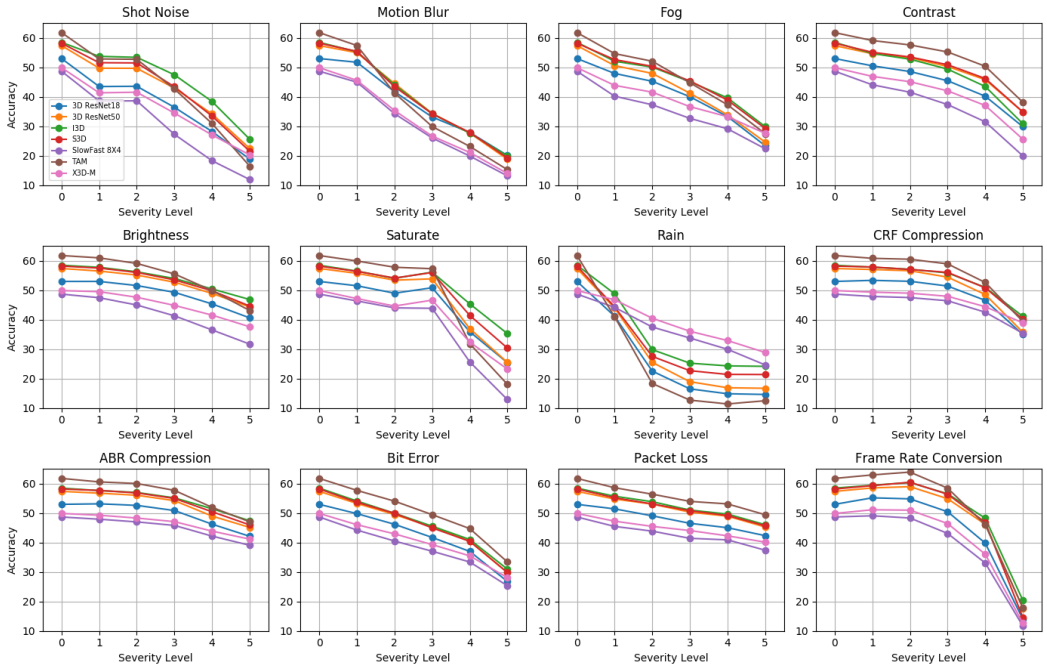

Figure 8: Model accuracy for SOTA approaches with respect to the video corruptions in Mini SSV2-C.

13 Sample of Corruption Severity

In Figure 10, Figure 11 and Figure 12, we show different types of corruptions with five level of severity. When the severity level increase, the corruptions vary from subtle to obvious. The parameters for synthesizing the corruption are crucial for the benchmark, and they are listed in the code on `https://github.com/Newbeeyoung/Video-Corruption-Robustness`. With this range, the benchmark can represent each corruption type comprehensively.

References

[1] J. Carreira and A. 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.

[2] C.-F. Chen, R. Panda, K. Ramakrishnan, R. Feris, J. Cohn, A. Oliva, and Q. Fan. Deep analysis of cnn-based spatio-temporal representations for action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

[3] J. Gilmer, N. Ford, N. Carlini, and E. Cubuk. Adversarial examples are a natural consequence of test error in noise. In *International Conference on Machine Learning*, pages 2280–2289, 2019.

[4] K. Hara, H. Kataoka, and Y. Satoh. Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 6546–6555, 2018.

[5] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[6] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[7] S. Xie, C. Sun, J. Huang, Z. Tu, and K. Murphy. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 305–321, 2018.

(a) Clean video frames in Mini Kinetics

(b) Corrupted by Gaussian noise with std=0.1

(c) Corrupted by Gaussian noise with std=0.2

Figure 9: The visualization samples of Gaussian noise corrupted video in Mini Kinetics.

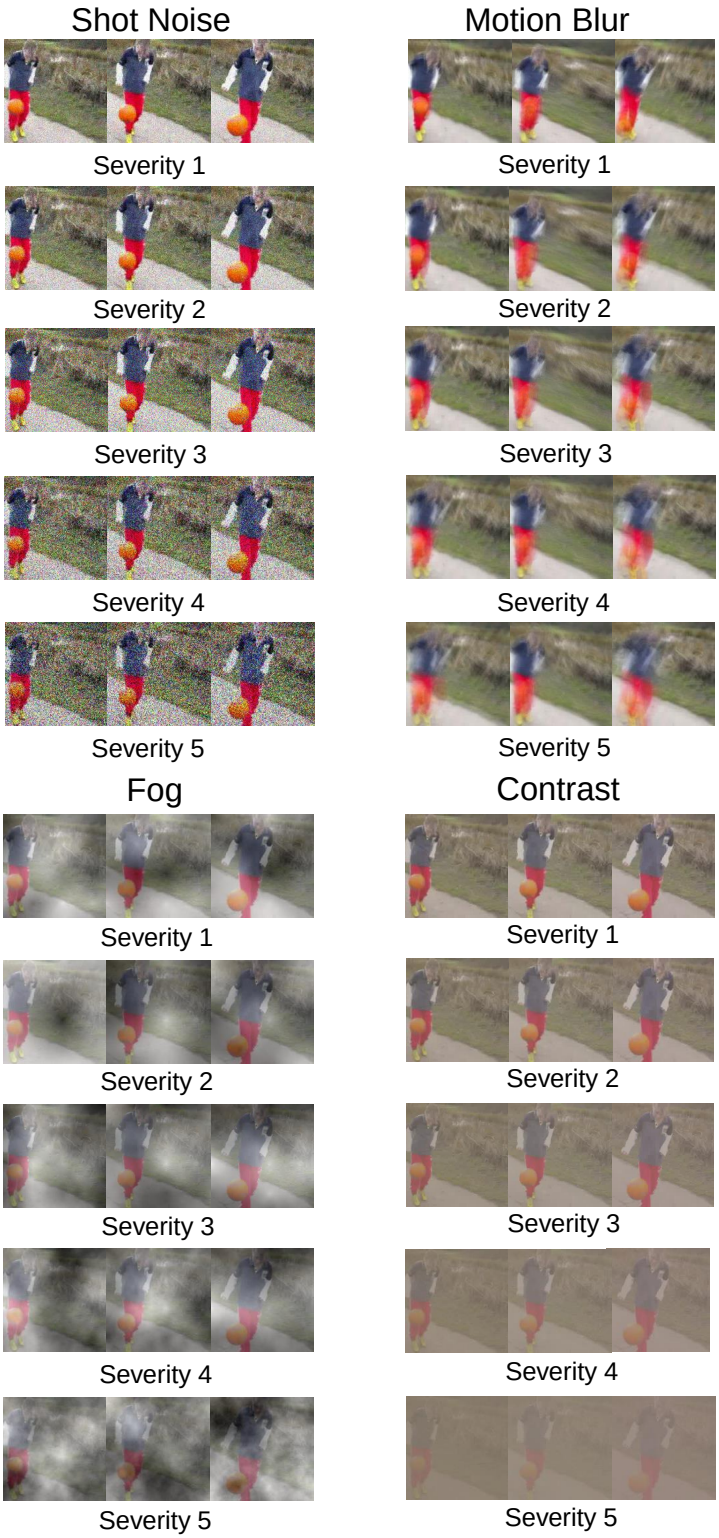

Figure 10: The visualization samples of shot noise, motion blur, fog and contrast with varying severities in Mini Kinetics-C

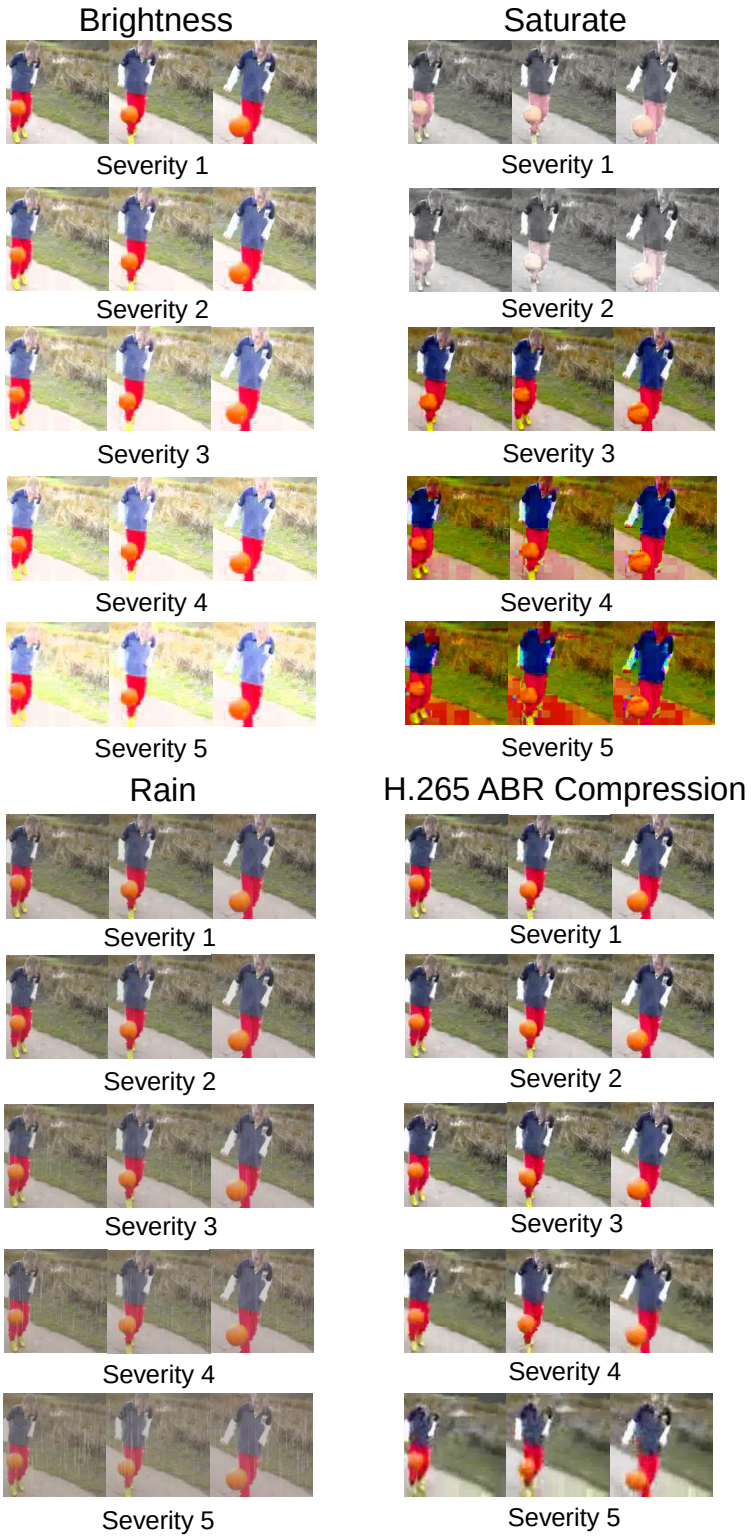

Figure 11: The visualization samples of brightness, saturate, rain and h.265 abr compression with varying severities in Mini Kinetics-C

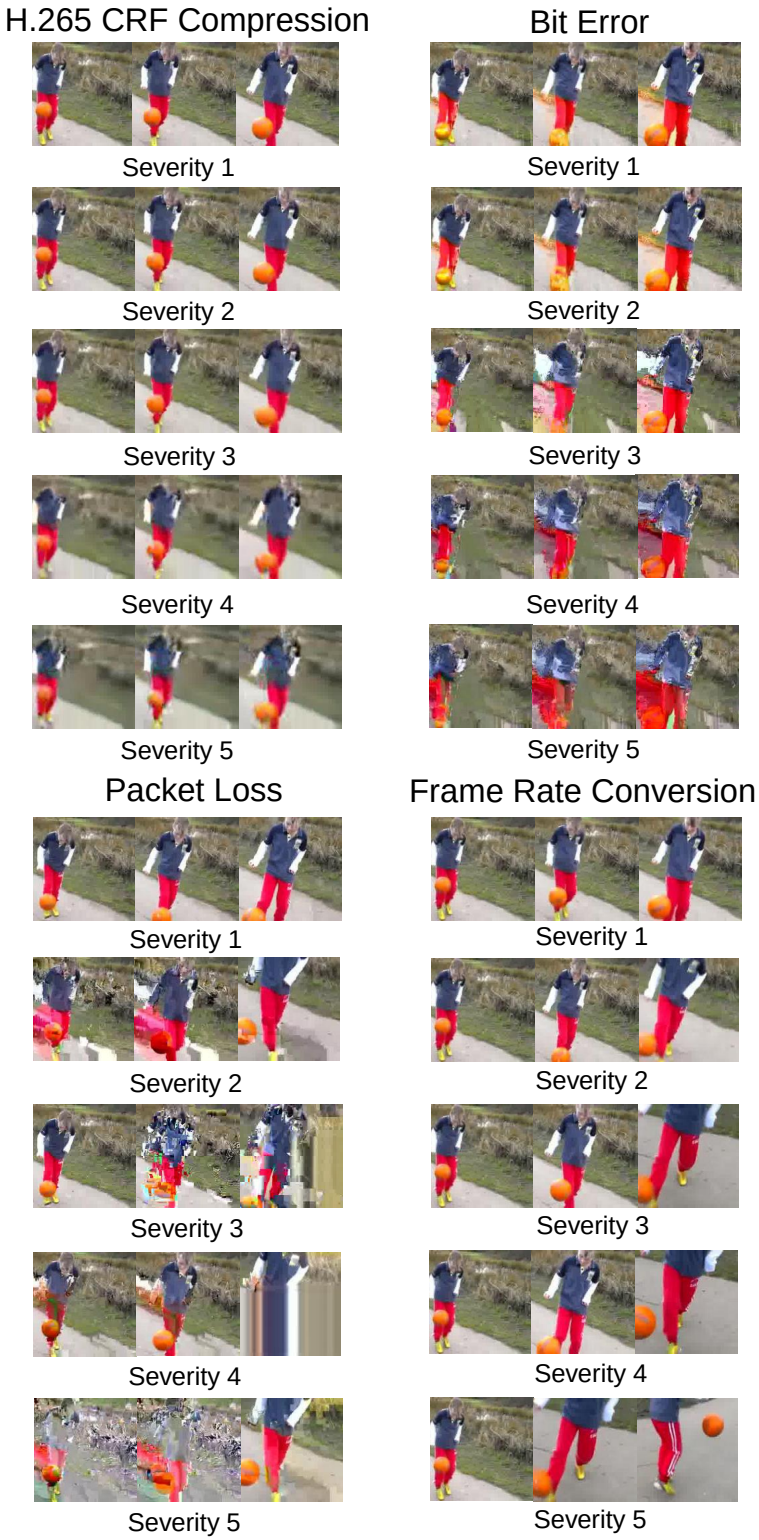

Figure 12: The visualization samples of h.265 crf compression, bit error, packet loss and frame rate conversion with varying severities in Mini Kinetics-C.