ELECTRICAL ENGINEERING AND COMPUTER SCIENCE
UNIVERSITY OF MICHIGAN

BOCA: Body-Worn Camera Analytics

NIST PSIAP Stakeholders / PI Meeting

July 2019

PI: Jason Corso

http://web.eecs.umich.edu/~jjcorso
jjcorso@umich.edu

DISCLAIMER

This presentation was produced by guest speaker(s) and presented at the National Institute of Standards and Technology's 2019 Public Safety Broadband Stakeholder Meeting. The contents of this presentation do not necessarily reflect the views or policies of the National Institute of Standards and Technology or the U.S. Government.

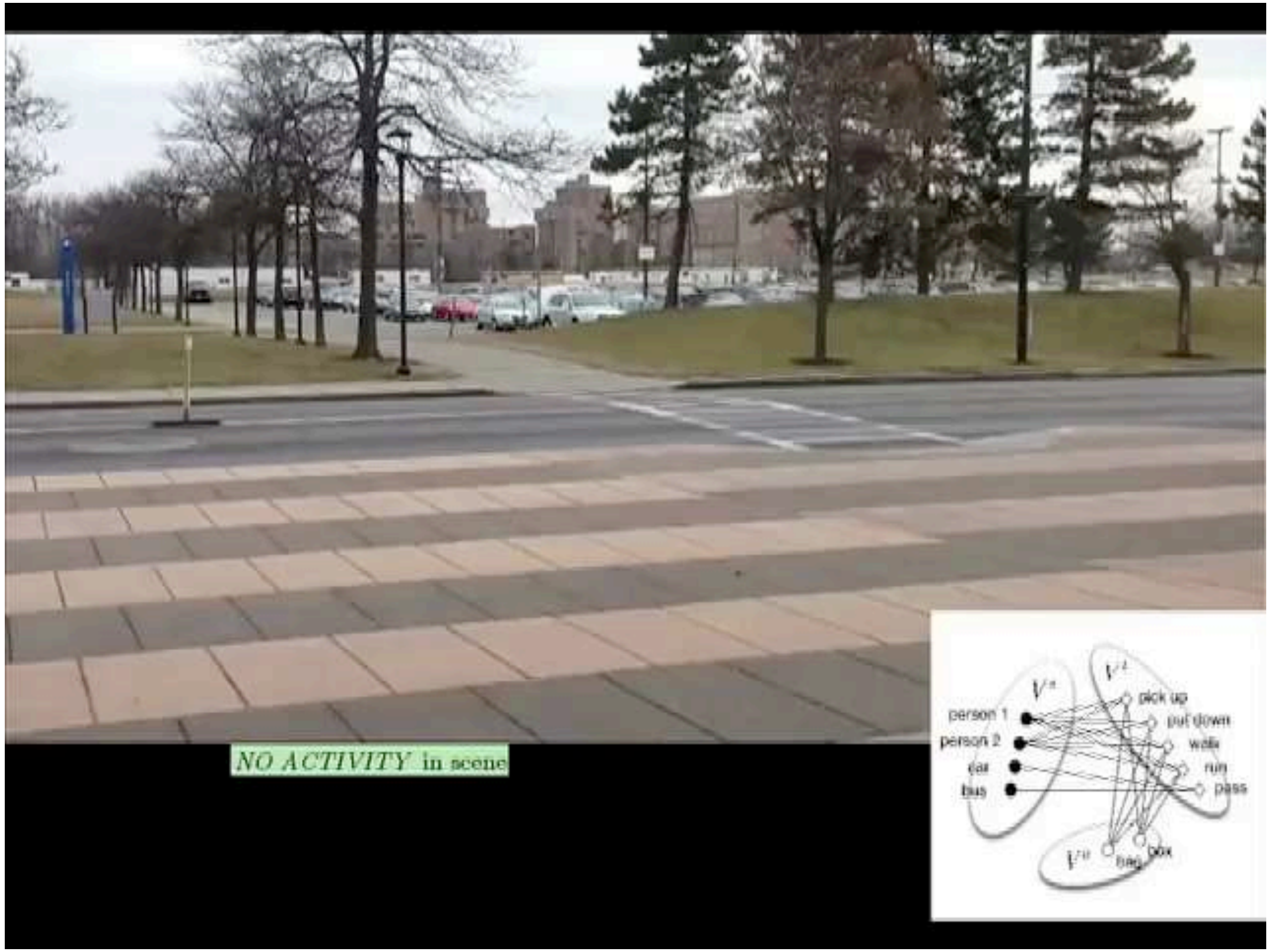
Posted with permission

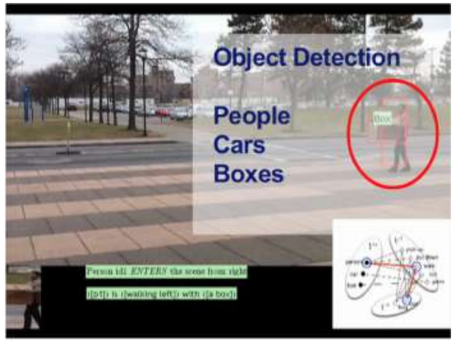
Team

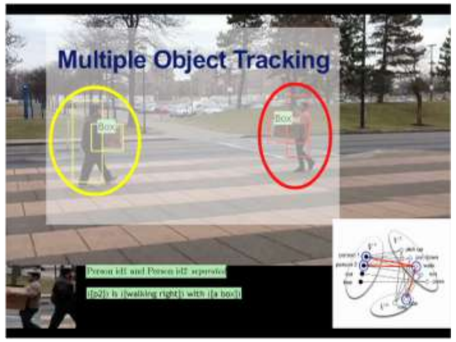
- University of Michigan
  - PI: Jason Corso (EECS)
  - Graduate Student: Kyle Byungsu Min

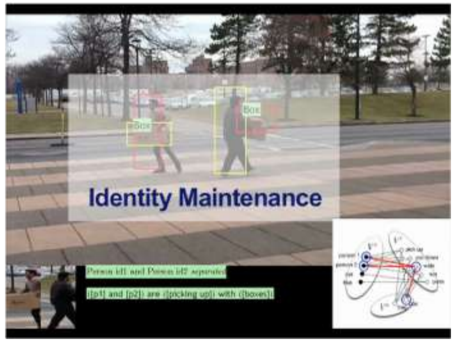
- Texas State University (Sub-Contract)
  - Faculty: Tom Yan (Computer Science)
  - Undergraduate students: Mario Delagarza, William Hunt and Kevin McNeff

What can *we* do now?





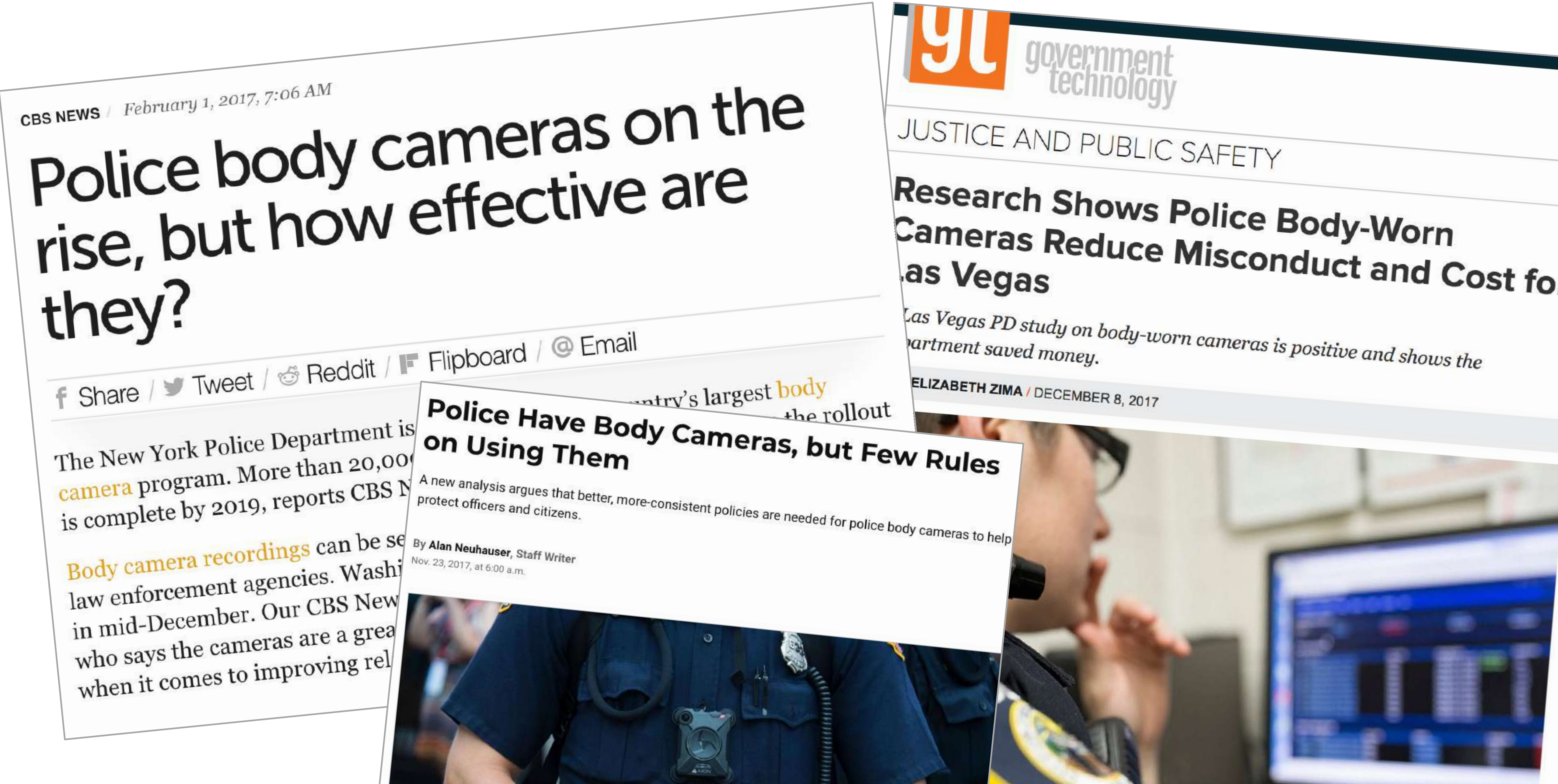




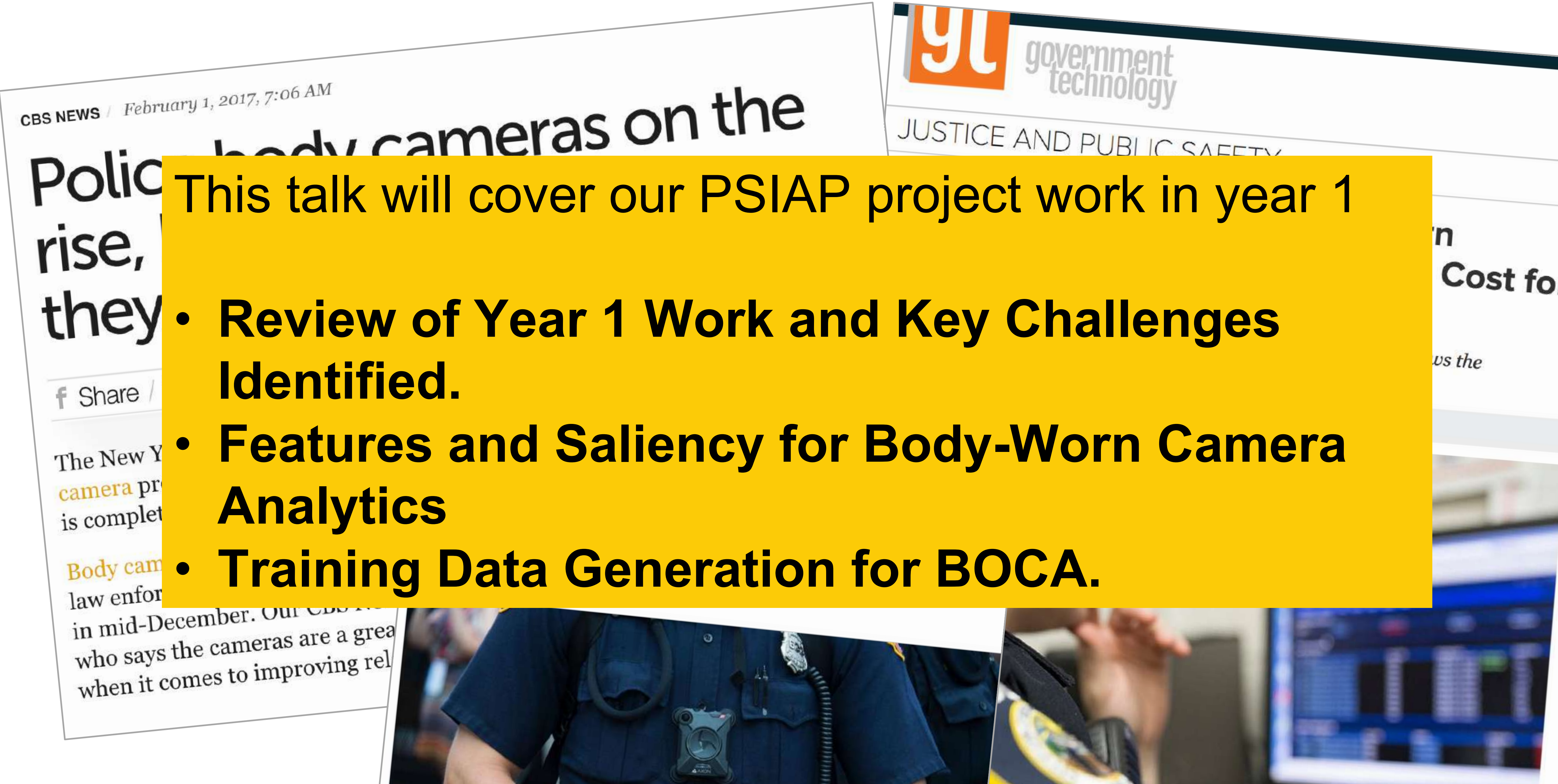
Video On an Index Card Engine

Demo Video Download https://youtu.be/DOmyl-UOkmc

BOCA - Body-Worn Camera Analytics



BOCA - Body-Worn Camera Analytics



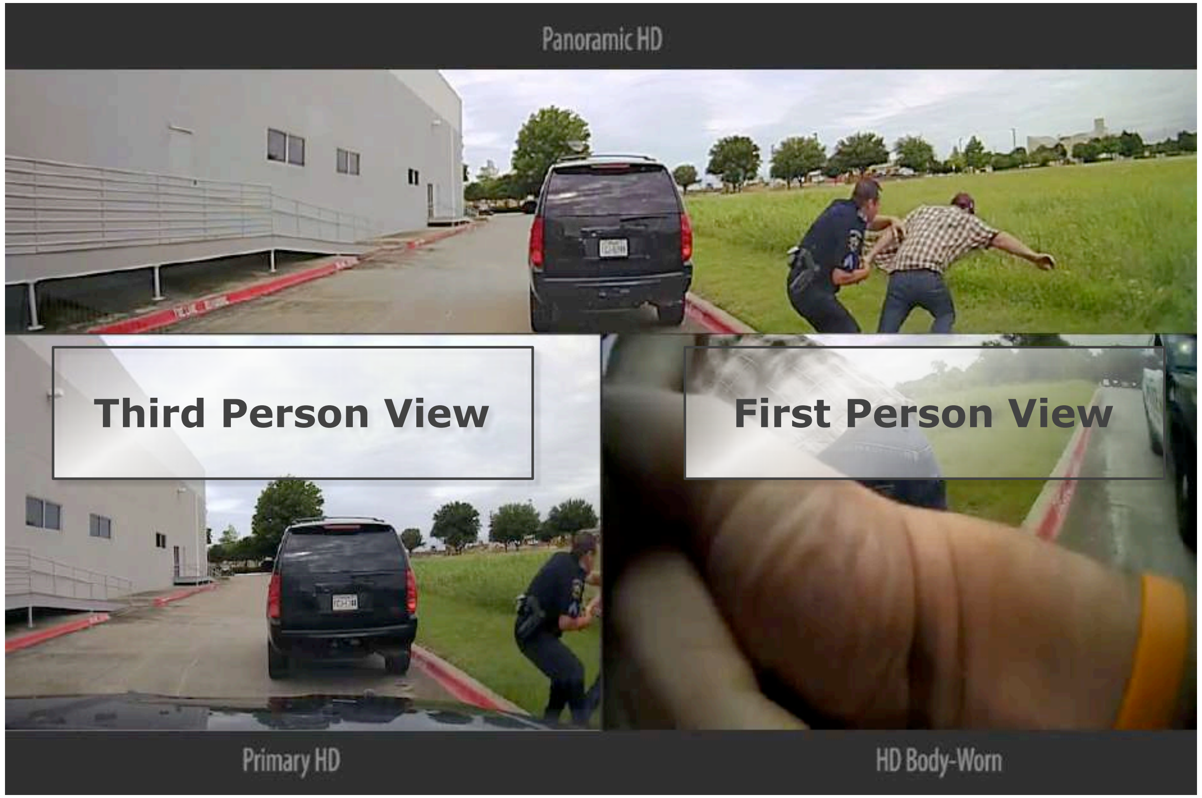
This talk will cover our PSIAP project work in year 1

- **Review of Year 1 Work and Key Challenges Identified.**
- **Features and Saliency for Body-Worn Camera Analytics**
- **Training Data Generation for BOCA.**

First-person vs Third-person

Both third and first person views are critical for fully understanding activities; often only one is available.

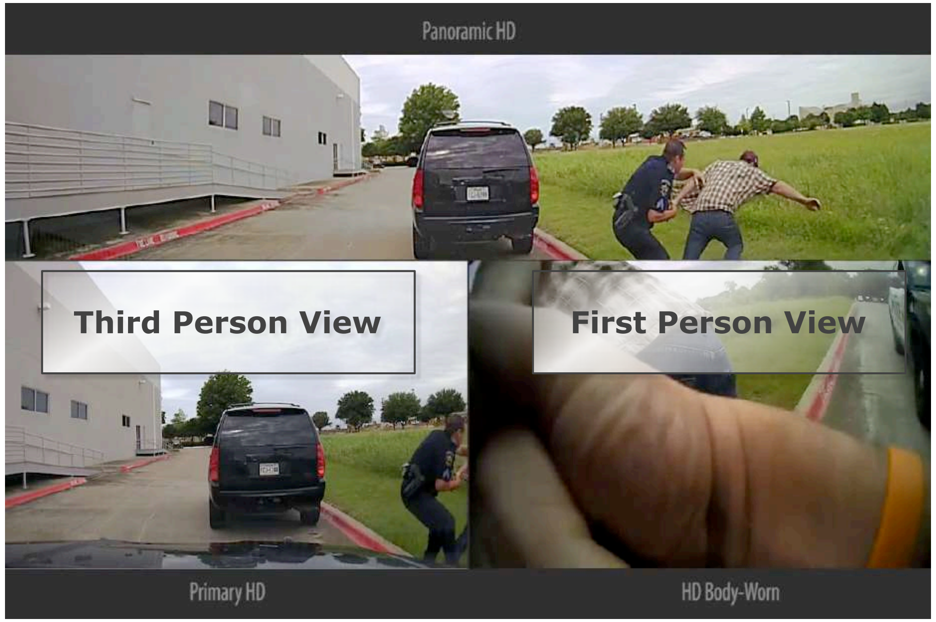
Activity Recognition



First-person vs Third-person

Both third and first person views are critical for fully understanding activities; often only one is available.

Activity Recognition



There is **no existing activity recognition dataset in the literature that supports body-worn activity recognition benchmarking**, nor with synchronized third-person view data. Therefore, a new body-worn camera activity dataset needs will be curated.

BOCA Dataset

first-person | third-person

Surfing

Ping-pong

Volleyball

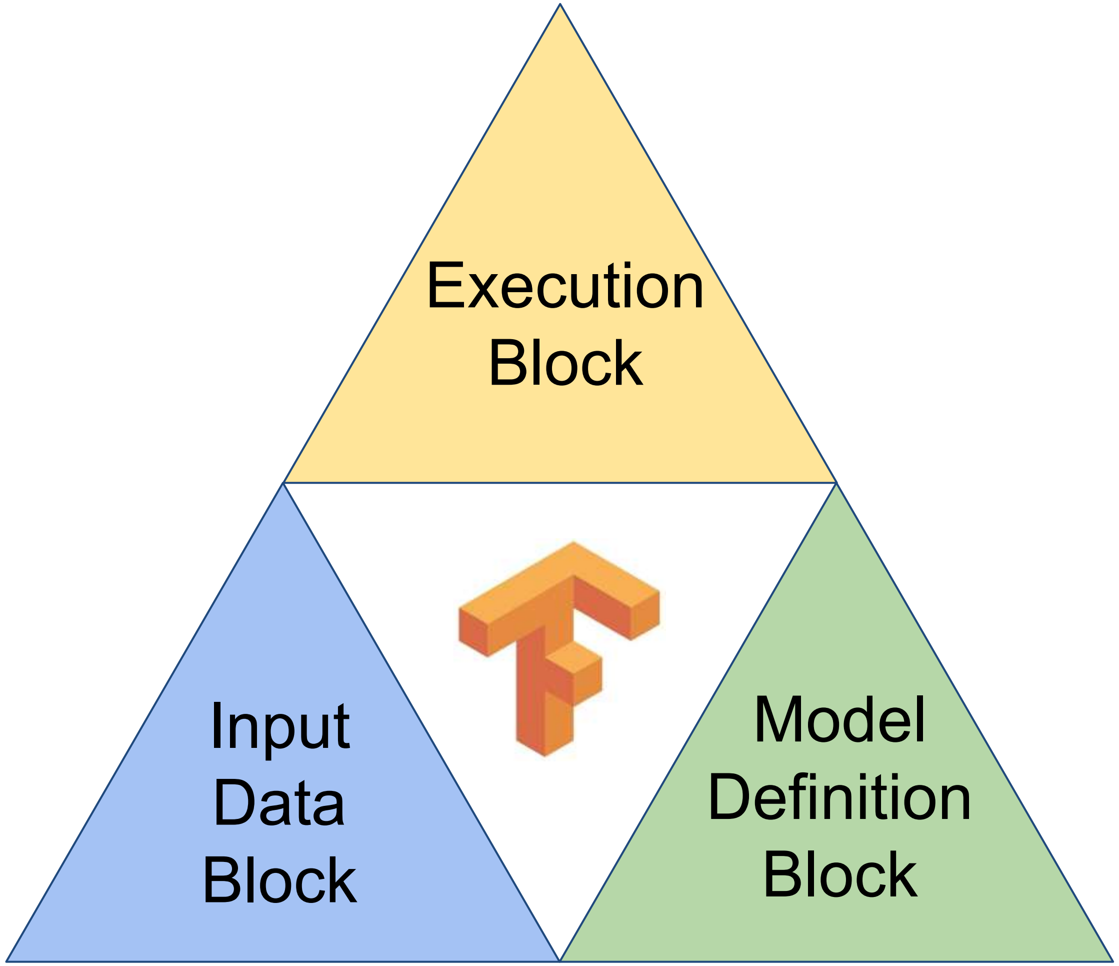
FPV dataset			
Name	Year	Video	Activity
VINST	2011	31	9
ADL	2012	20	18
GTEA+	2012	30	100
Disney Social	2012	8	12
JPL Interaction	2013	57	7
Huji EgoSeg	2014	122	7

BOCA Dataset Statistics

Activity	First-person	Third-person	Total
Horseback-riding	139	144	283
Surfing	146	124	270
Ping-pong	134	155	289
Running-a-marathon	148	143	291
Playing-racquetball	146	142	288
Playing-lacrosse	144	141	285
Volleyball	141	153	294
Playing-squash	144	156	300
Playing-badminton	144	115	259
Windsurfing	142	158	300
Snowboarding	140	158	298
Playing-water-polo	138	186	324
Playing-ice-hockey	148	148	306
Hammer-throw	23	155	178
Dodgeball	130	144	274
ALL	2007	2222	4229

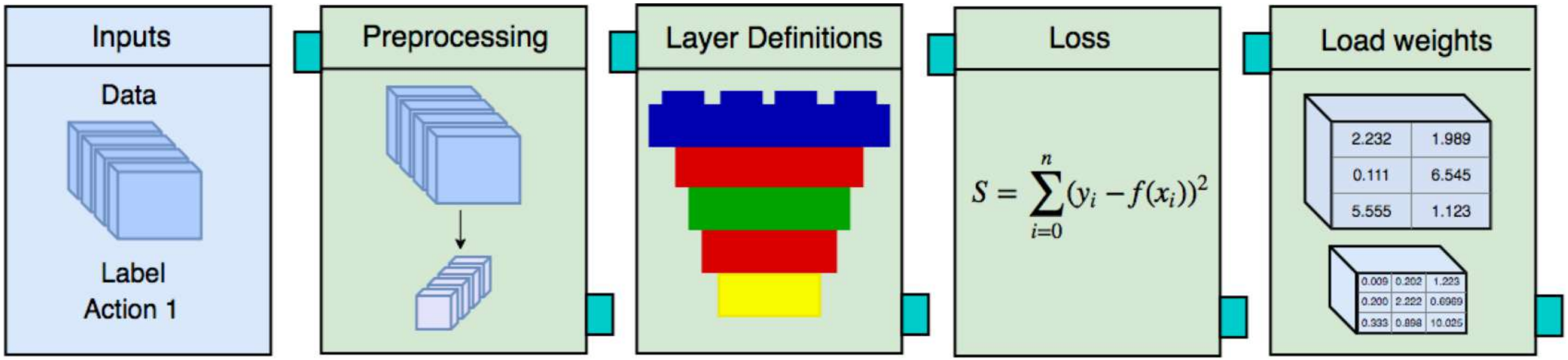
Table 2: Statistics for the collected sport activity dataset.

M-PACT: Michigan Platform for Activity Classification in Tensorflow



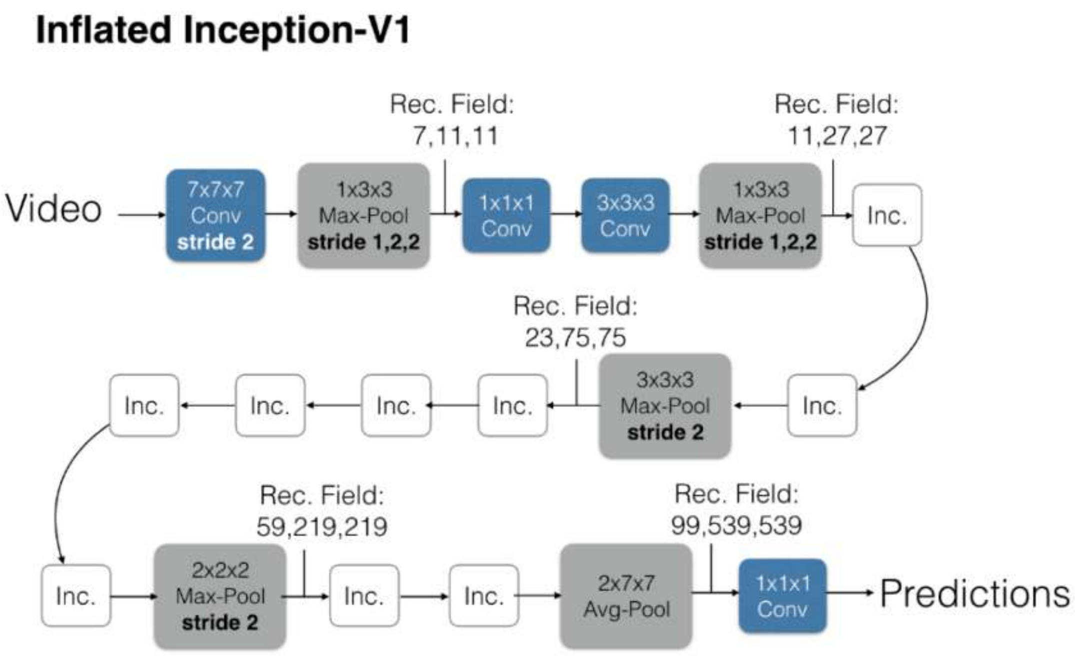
https://github.com/MichiganCOG/M-PACT

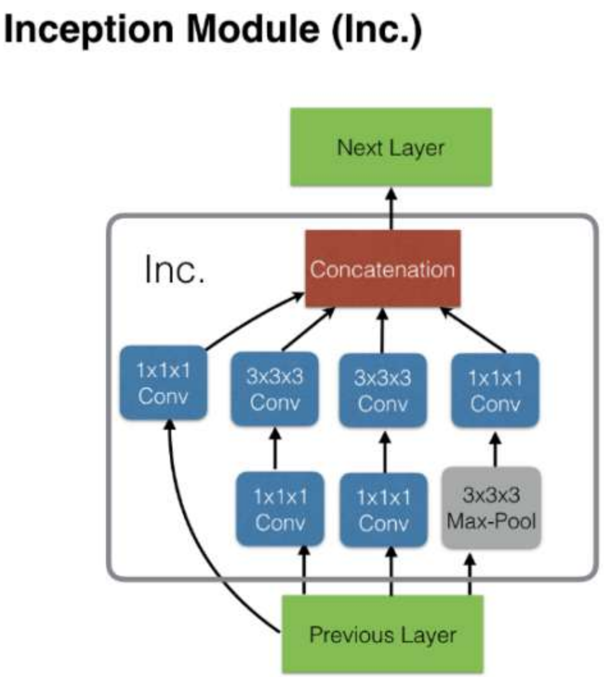
Model Definition Block



https://github.com/MichiganCOG/M-PACT

Implemented Model: I3D





Model	HMDB51 Accuracy (%)		UCF101 Accuracy (%)	
	Orig. Authors	M-PACT	Orig. Authors	M-PACT
I3D	74.80	68.10	95.60	92.55

Carreira J. et al. CVPR 2017

https://github.com/MichiganCOG/M-PACT

Where?

- https://github.com/MichiganCOG/M-PACT

M-PACT: Michigan Platform for Activity Classification in Tensorflow

This python framework provides modular access to common activity recognition models for the use of baseline comparisons between the current state of the art and custom models.
This README will walk you through the process of installing dependencies, downloading and formatting datasets, testing the framework, and expanding the framework to train your own models.

This repository holds the code and models for the paper
M-PACT: Michigan Platform for Activity Classification in Tensorflow, Eric Hofesmann, Madan Ravi Ganesh, and Jason J. Corso, arXiv, April 2018.

ATTENTION: Please cite the arXiv paper introducing this platform when releasing any work that used this code.
Link: https://arxiv.org/abs/1804.05879

Implemented Model's Classification Accuracy:

Model Architecture	Dataset (Split 1)	M-PACT Accuracy (%)	Original Authors Accuracy (%)
I3D	HMDB51	68.10	74.80*
C3D	HMDB51	51.90	50.30*
TSN	HMDB51	51.70	54.40
ResNet50 + LSTM	HMDB51	43.86	43.90
I3D	UCF101	92.55	95.60*
C3D	UCF101	93.66	82.30*
TSN	UCF101	85.25	85.50
ResNet50 + LSTM	UCF101	80.20	84.30

(*) Indicates that results are shown across all three splits

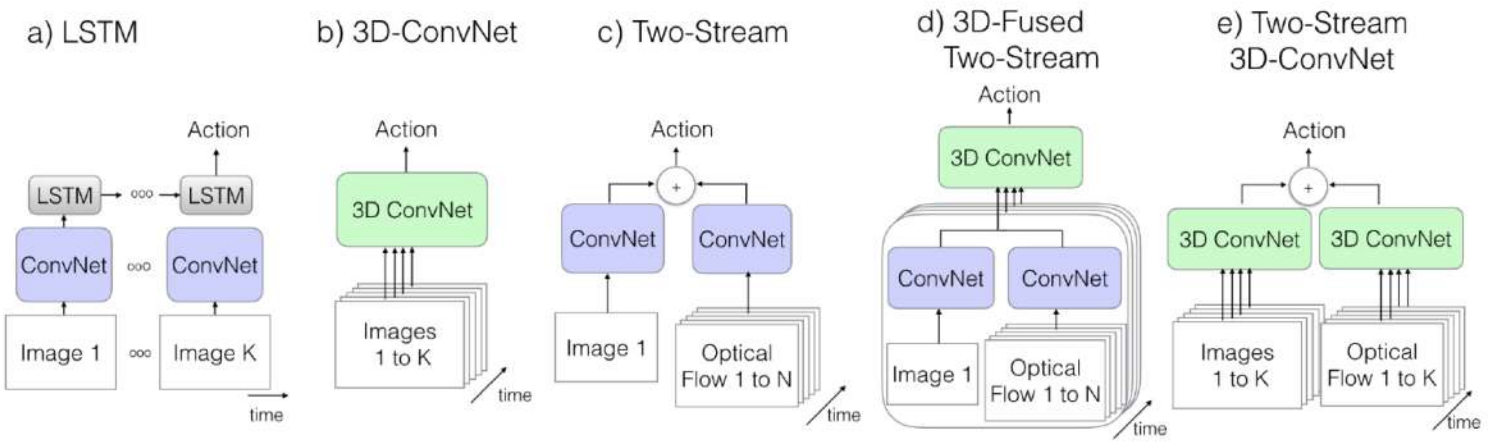
Table of Contents

Framework File Structure

```
/tf-activity-recognition-framework
   train.py
   test.py
   create_model.py
   load_a_video.py

   /models
        /model_name
            modelname_model.py
            default_preprocessing.py
            model_weights.npy shortcut to ../weights/model_weights.npy (Optional)

        /weights
            model_weights.npy

   /results
        /model_name
            /dataset_name
                /preprocessing_method
                    /experiment_name
                            /checkpoints
                                checkpoint
                                checkpoint-100.npy
                                checkpoint-100.dat
                            /metrics_method
                                testing_results.npy

    /logs
        /model_name
            /dataset_name
                /preprocessing_method
                    /metrics_method
                        /experiment_name
                            tensorboard_log

    /scripts
        /shell
            download_weights.sh

    /utils
        generate_tfrecords_dataset.py
        convert_checkpoint.py
        checkpoint_utils.py
        layers_utils.py
        metrics_utils.py
        preprocessing_utils.py
        sys_utils.py
        logger.py
```

SOTA Video Classification Pipeline

ConvNet + LSTM and two-stream network are usually used for video classification.



We use I3D model [1] trained from third-person and evaluated on our collected BOCA dataset.
We observe that there is actually distribution gap between first-person and third-person activities.
(74.2% vs 60.9%, 54.4% vs 50.4%)

	top1_acc	*top5_acc*
First-person	50.4 %	69.4 %
First-person (removing classes doesn't overlap with third-person)	60.9 %	85.1 %
Third-person	54.4 %	66.8 %
Third-person (removing classes doesn't overlap with first-person)	74.2 %	91.3 %

Table 1: I3D model baselines on our collected dataset.

[1] "Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset" by Joao Carreira and Andrew Zisserman, CVPR 2017

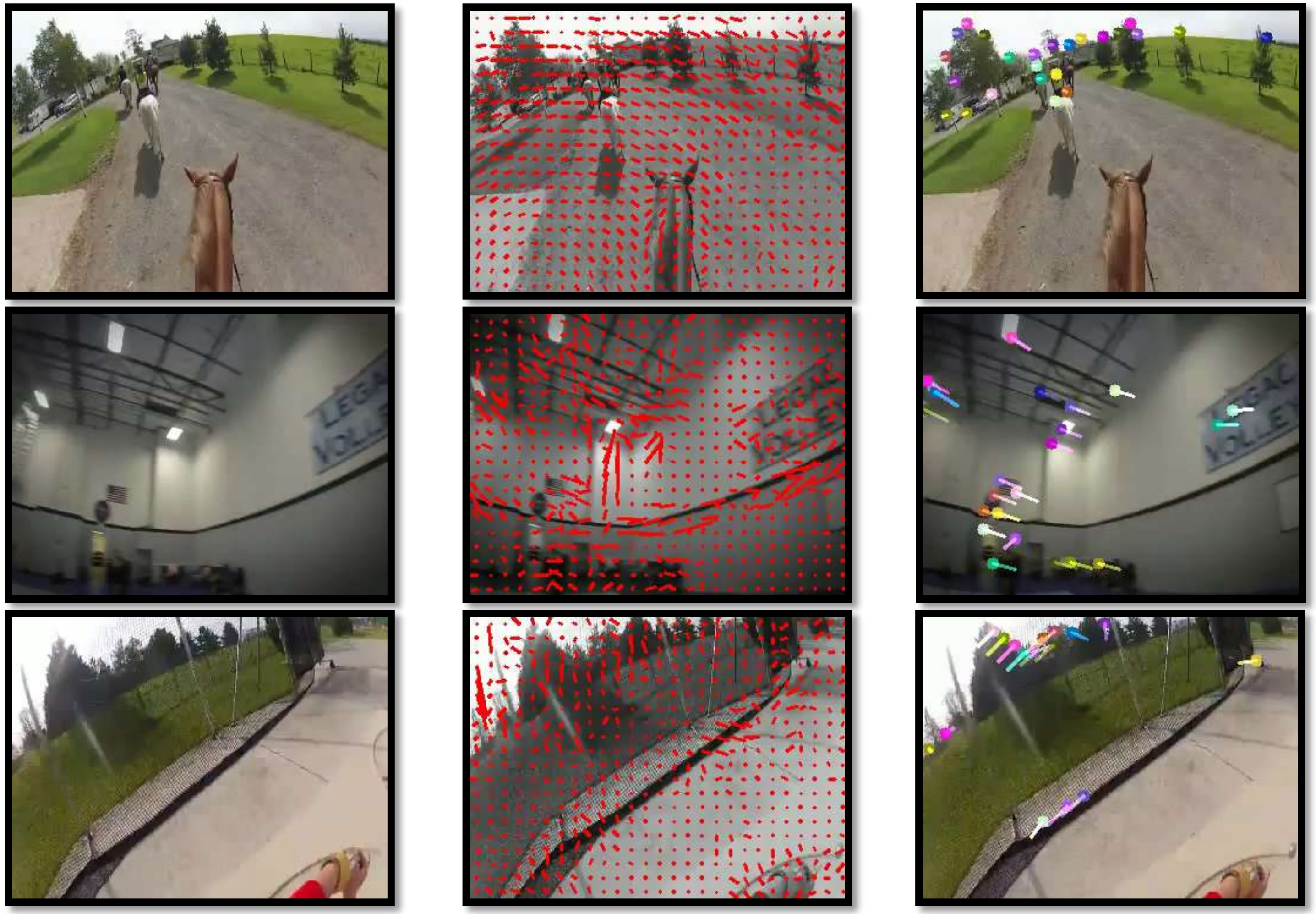
BOCA Dataset (cont.)

What makes FPV activity understanding so hard?

Horseback-riding

Volleyball

Hammer-throw



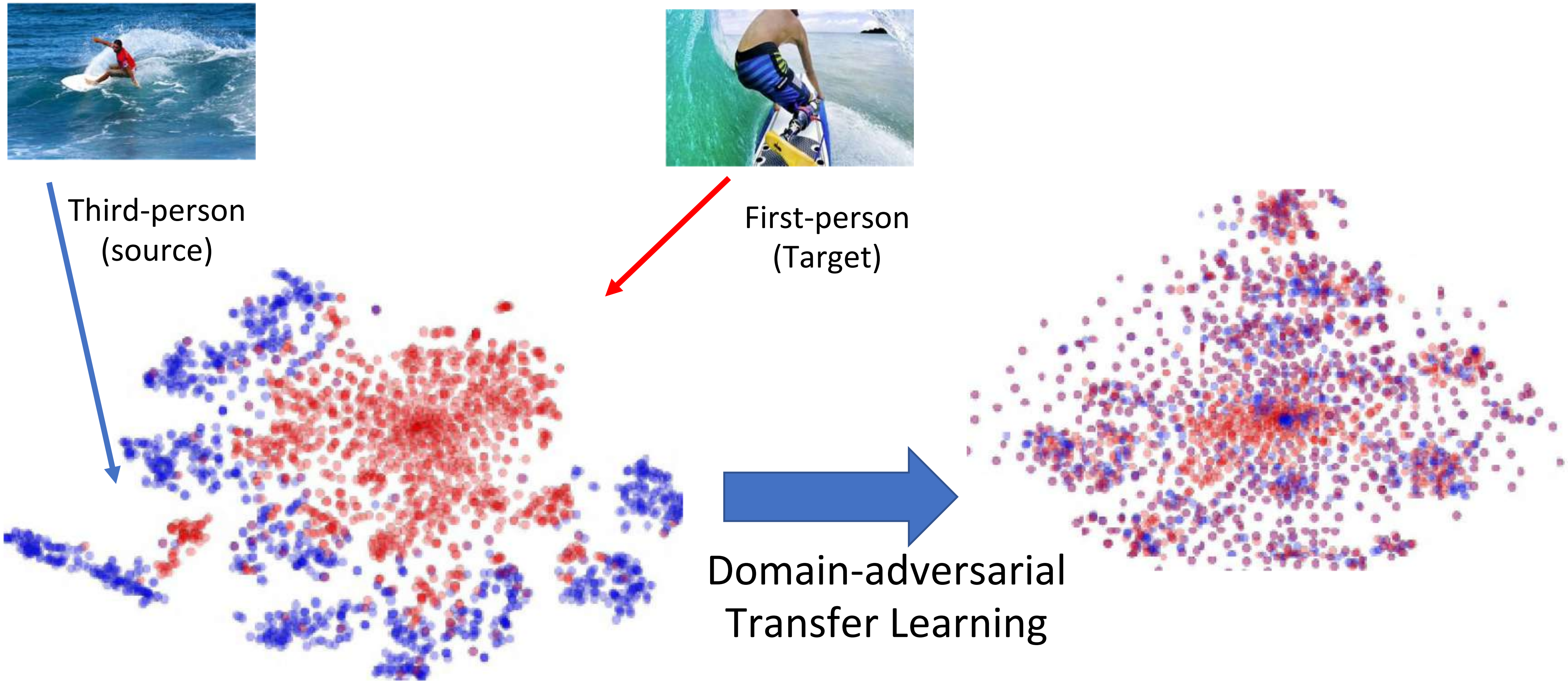
Original video | Dense Optical Flow Field | Point tracking using OF

Camera motion

Summary of Findings from Year 1

- Third-Person v First-Person Activities
  - First-Person activity recognition is more difficult because the range of motion of higher.
- Third-Person v First-Person Data
  - The available data resources are limited for first-person-based learning models, unlike third-person.
- Suggests
  - Transfer learning approaches are important.
  - Different handling of features and content in first-person video is necessary.

Domain-adversarial Transfer Learning



Year 2 Emphasis 1:
Feature-Saliency for Body-Worn Cameras

Challenge: Conventional feature descriptors and models do not perform well on first-person videos from a body-worn camera largely due to a high range of motion.

To overcome this problem, we propose a saliency-based approach as a different, effective way of analyzing first-person videos.

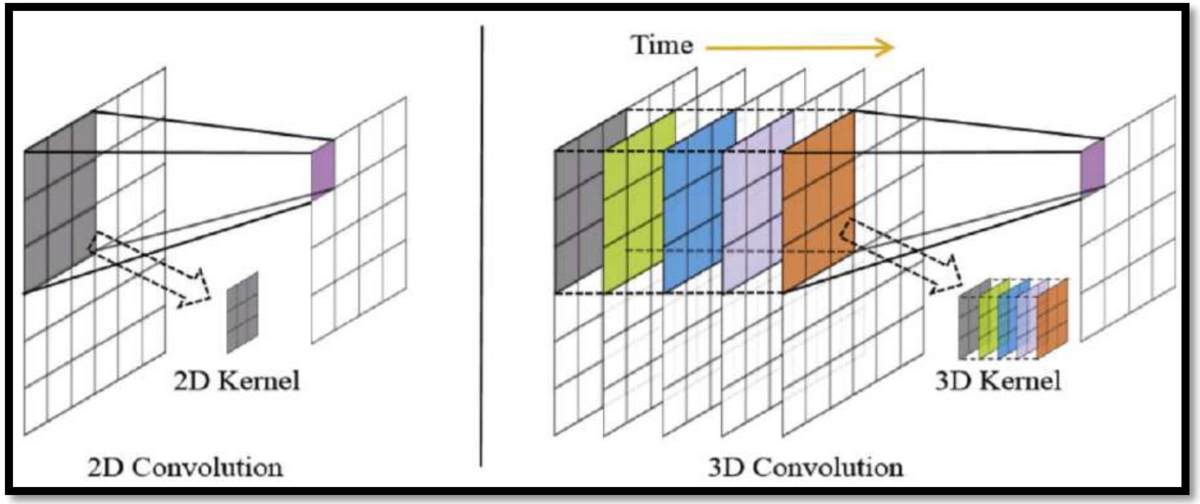
Video Saliency

- It prioritize the information across space and time
- There no longer is a need for an explicit detection or tracking of objects.
- In this sense, we study video saliency detection models to develop an implicit approach which can boost the performance of the activity recognition model for first-person videos.

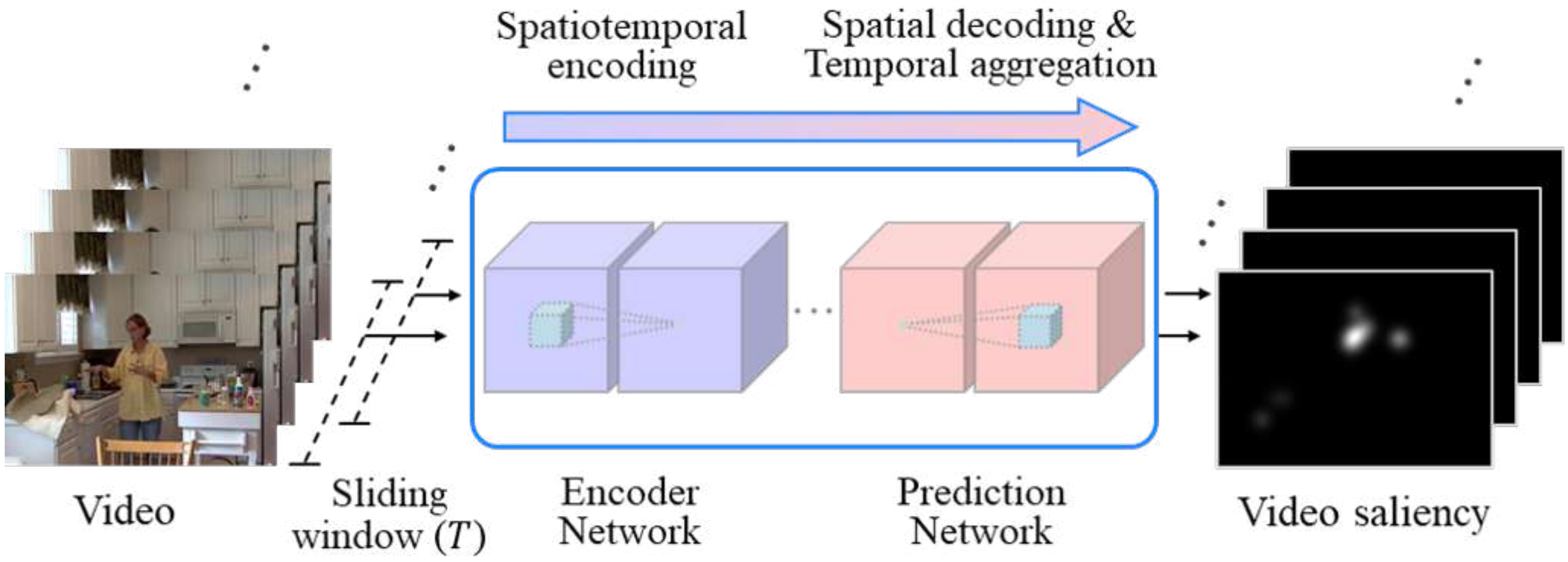
Video Saliency

We found that all the previous approaches fail to jointly process the spatiotemporal information, which is expected to be important to video saliency detection; that is, The existing works are unable to leverage the collective spatiotemporal variation.

To this end, we propose TASED-Net, which is a novel 3D ConvNet architecture for video saliency detection.

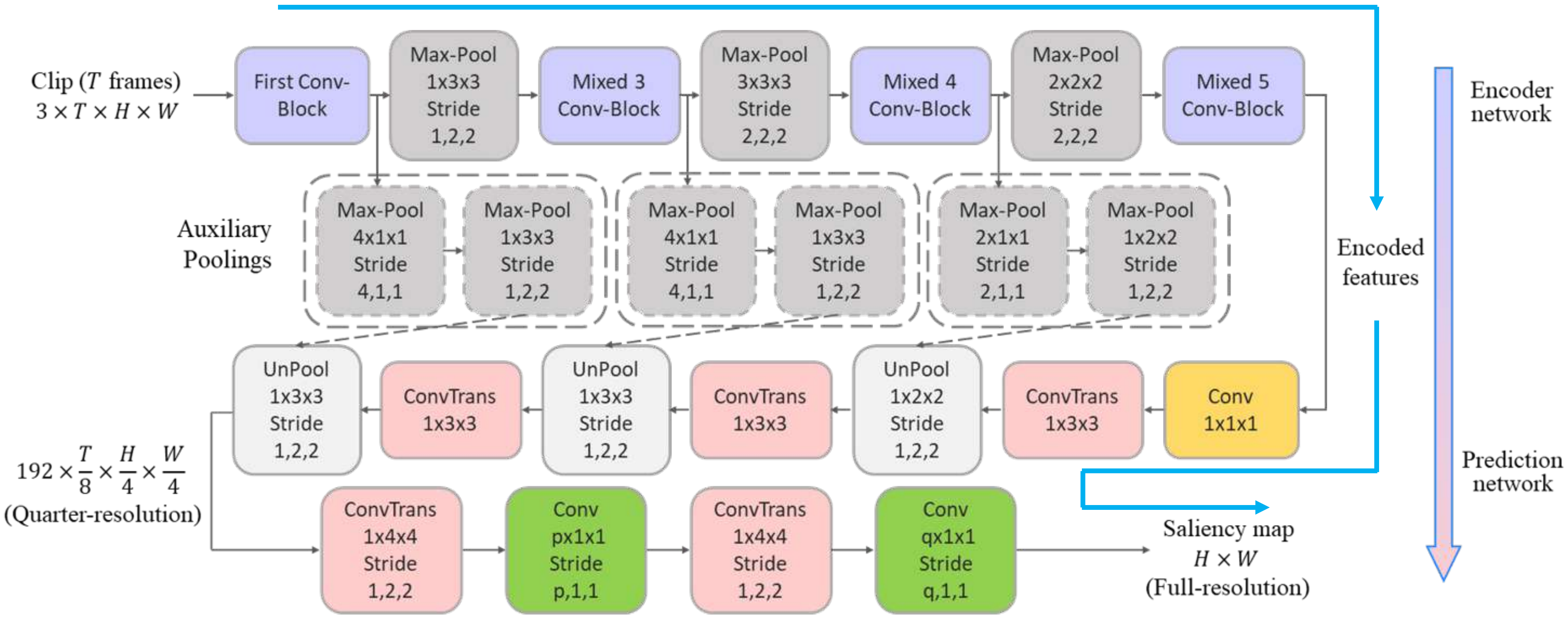


TASED-Net



In TASED-Net (Temporally-Aggregating Spatial Encoder-Decoder Network), an input clip of multiple frames is spatiotemporally encoded. The encoded features are then decoded spatially while all the temporal information of it is aggregated by the following decoder to produce a saliency map.

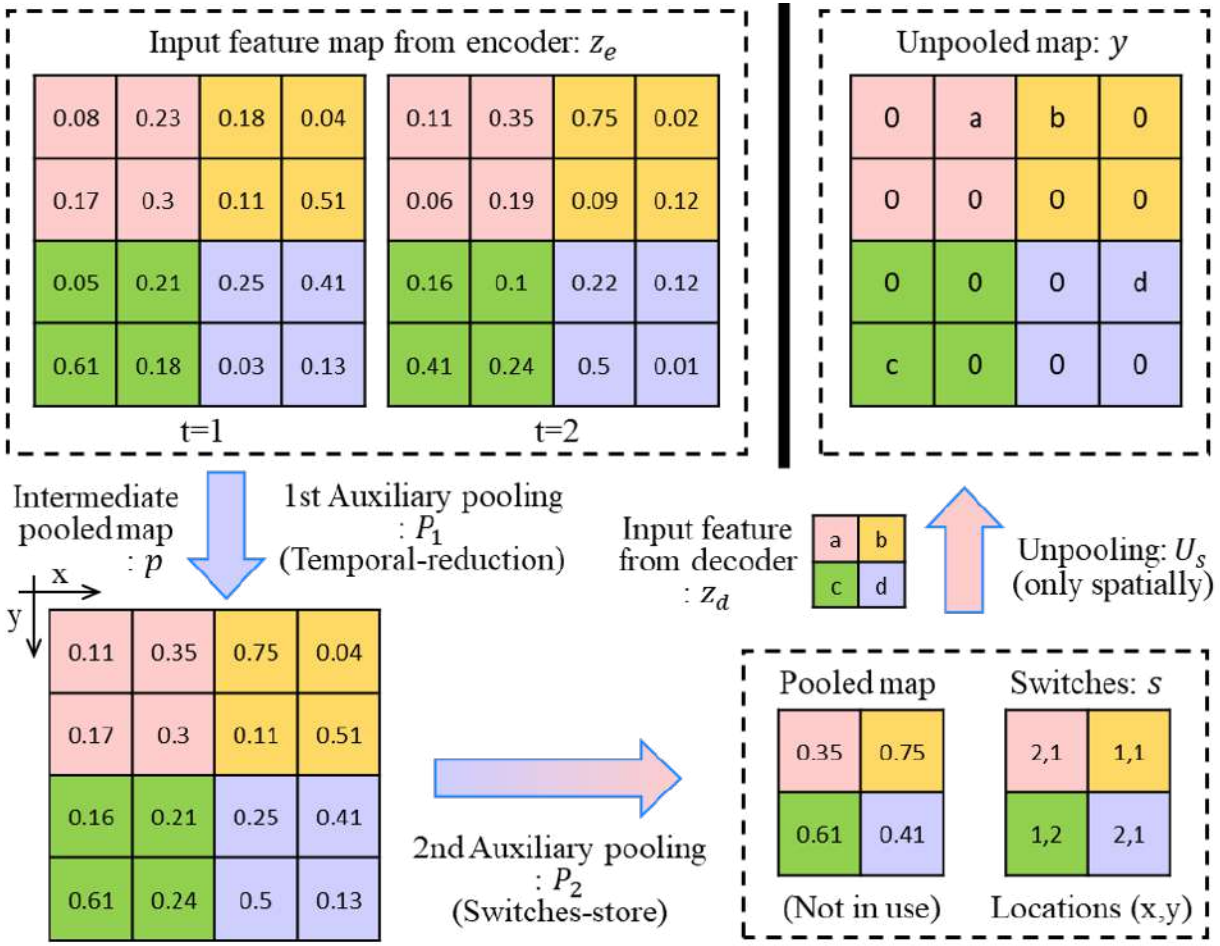
TASED-Net



Using 3D convolutional networks for the decoding purpose is non-trivial.

In order to resolve the tricky problem, we propose Auxiliary poolings.

Auxiliary Poolings



In mathematical notation,

Normal pooling: $[p, s] = P(z)$

1st Auxiliary pooling: $[p, -] = P_1(z_e)$

Normal pooling: $[-, s] = P_2(p)$

Normal pooling: $y = U_s(z_d)$

Auxiliary Poolings

Method \ Metric	NSS	CC	SIM	AUC-J	s-AUC
TASED-Net-tri	2.452	0.448	0.337	0.891	0.702
TASED-Net-trp	2.598	0.470	0.353	0.894	0.707
TASED-Net	2.706	0.481	0.362	0.894	0.718

TASED-Net-tri and TASED-Net-trp do not utilize Auxiliary pooling because they replace unpooling layers with trilinear upsampling and transposed convolution, respectively.

TASED-Net perform better, which demonstrates the effectiveness of Auxiliary pooling.

How many frames should we use?

Method \ Metric	NSS	CC	SIM	AUC-J	s-AUC
TASED-Net (4)	2.434	0.441	0.327	0.887	0.689
TASED-Net (8)	2.585	0.460	0.348	0.889	0.696
TASED-Net (16)	2.622	0.469	0.349	0.892	0.713
TASED-Net (32)	**2.706**	**0.481**	**0.362**	**0.894**	**0.718**
TASED-Net (48)	2.636	0.472	0.348	**0.894**	0.708
TASED-Net (64)	2.554	0.459	0.336	0.893	0.702

In order to decide how many frames we use to aggregate at one pass, we performed many experiments to optimize T.

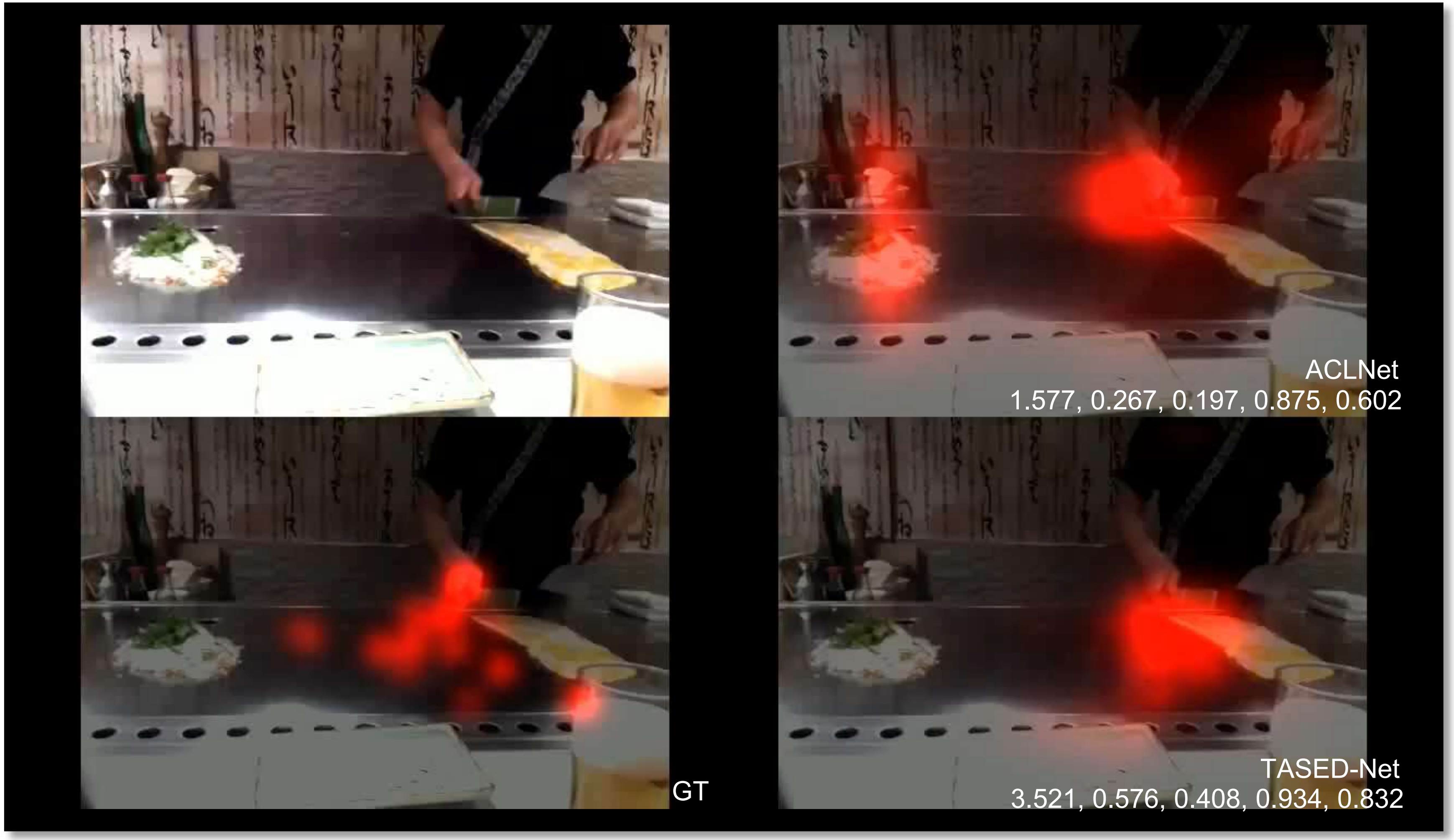
We observe that a clip with a duration of about one second (32 frames) produces the best performance.

Results

Method \ Metric	NSS	CC	SIM	AUC-J	s-AUC
GBVS	1.474	0.283	0.186	0.828	0.554
STSConvNet	1.632	0.325	0.197	0.834	0.581
Deep Net	1.775	0.331	0.201	0.855	0.592
SALICON	1.901	0.327	0.232	0.857	0.590
OM-CNN	1.911	0.344	0.256	0.856	0.583
DVA	2.013	0.358	0.262	0.860	0.595
SalGAN	2.043	0.370	0.262	0.866	0.709
ACLNet	2.354	0.434	0.315	0.890	0.601
TASED-Net	**2.667**	**0.470**	**0.361**	**0.895**	**0.712**

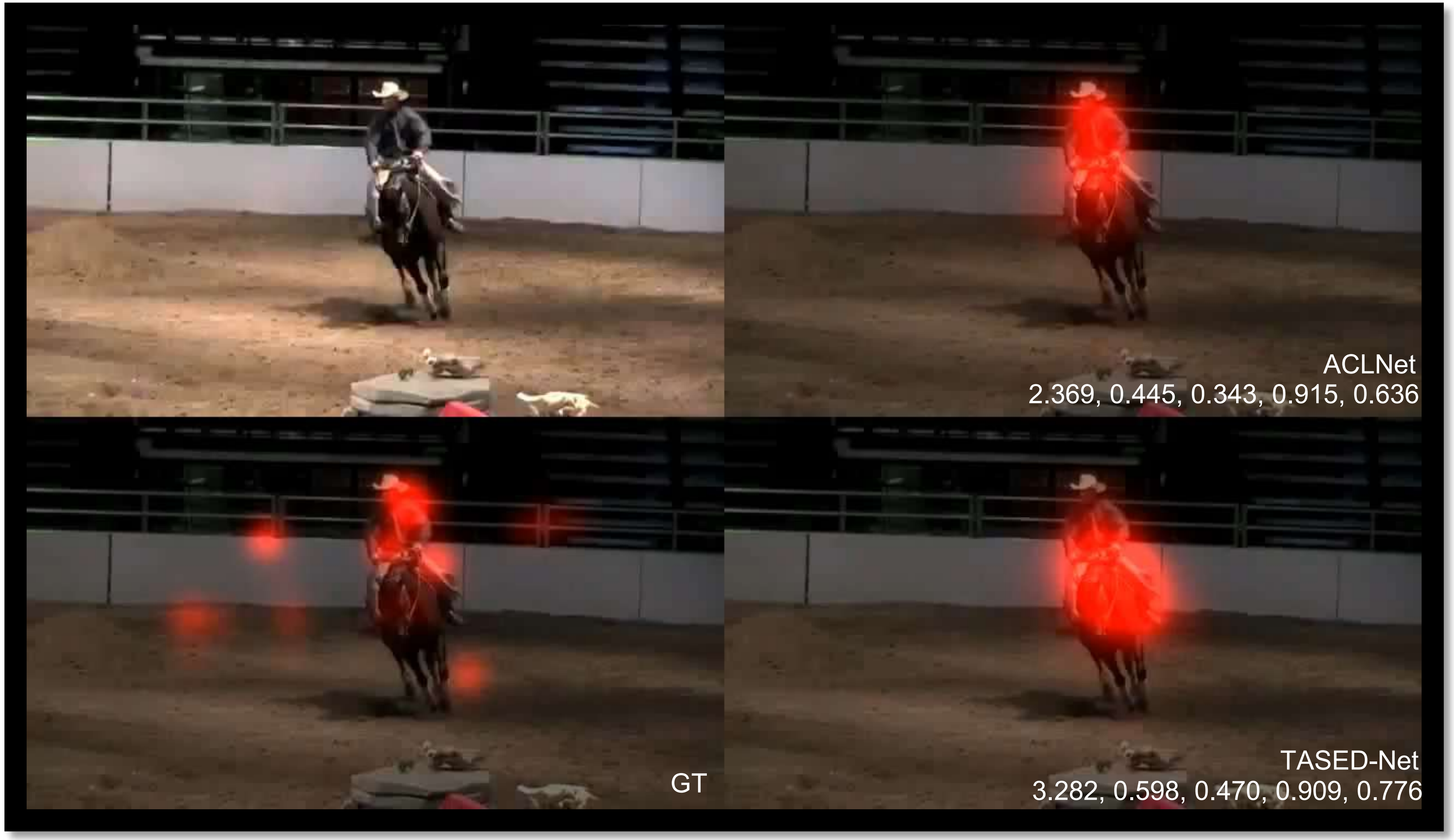
	Method \ Metric	NSS	CC	SIM	AUC-J	s-AUC
Hollywood2	STSConvNet	1.748	0.382	0.276	0.863	0.710
	SALICON	2.013	0.425	0.321	0.856	0.711
	Deep Net	2.066	0.451	0.300	0.884	0.736
	OM-CNN	2.313	0.446	0.356	0.887	0.693
	DVA	2.459	0.482	0.372	0.886	0.727
	ACLNet	3.086	0.623	**0.542**	0.913	0.757
	TASED-Net	**3.302**	**0.646**	0.507	**0.918**	**0.768**
UCFSports	GBVS	1.818	0.396	0.274	0.859	0.697
	Deep Net	1.903	0.414	0.282	0.861	0.719
	OM-CNN	2.089	0.405	0.321	0.870	0.691
	DVA	2.311	0.439	0.339	0.872	0.725
	ACLNet	2.567	0.510	0.406	0.897	0.744
	TASED-Net	**2.920**	**0.582**	**0.469**	**0.899**	**0.752**

Results

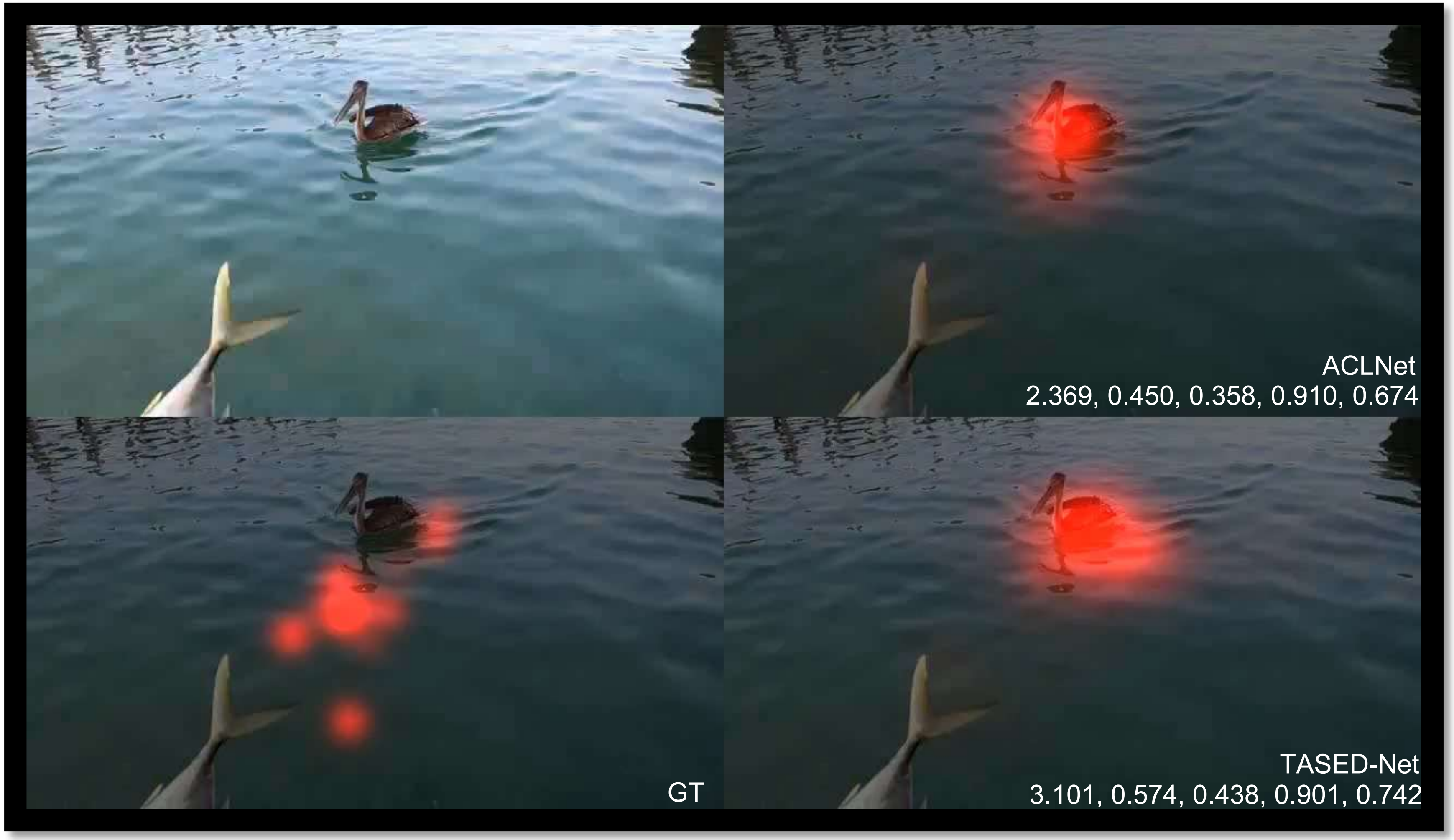

ACLNet
1.577, 0.267, 0.197, 0.875, 0.602
GT
TASED-Net
3.521, 0.576, 0.408, 0.934, 0.832

BOCA - Body-Worn Camera Analytics

Results



Results


ACLNet
2.369, 0.450, 0.358, 0.910, 0.674
GT
TASED-Net
3.101, 0.574, 0.438, 0.901, 0.742

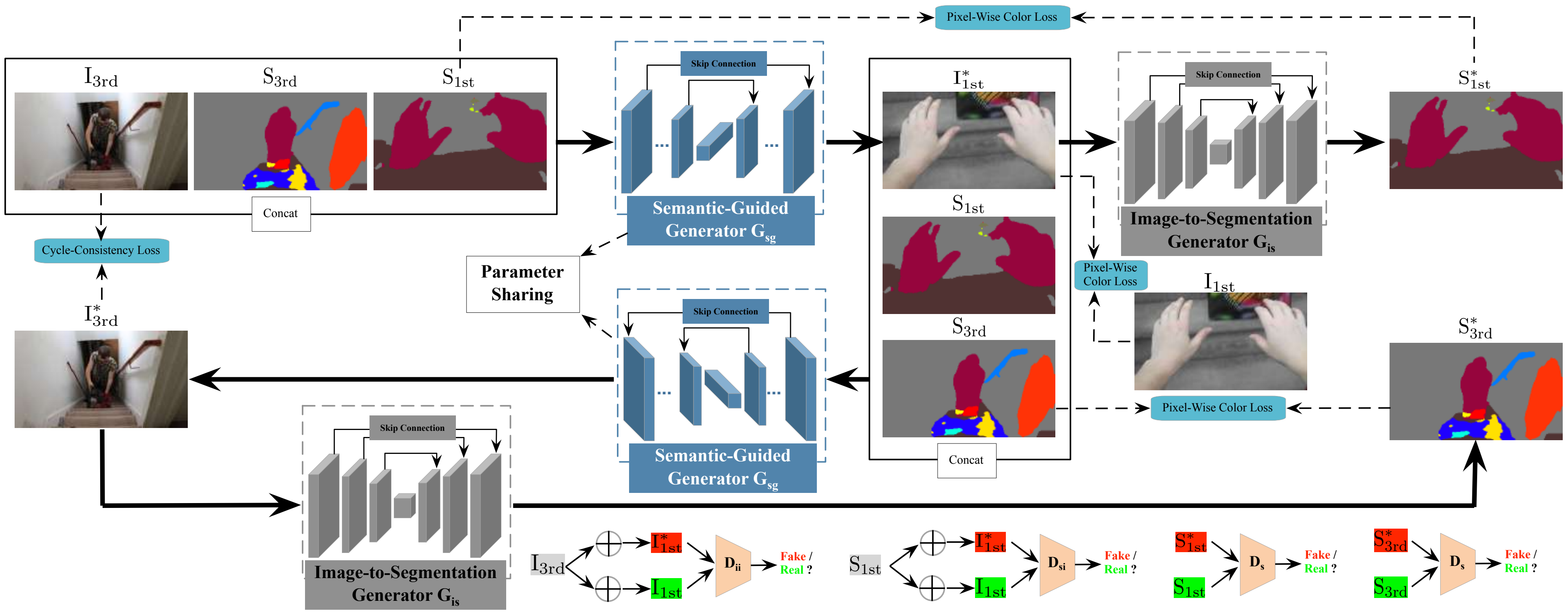
Year 2 Emphasis 2: Training Data Generation for Body-Worn Cameras

Challenge: Body-worn cameras produce a huge amount of data—unannotated data the annotation of which would require a massive human effort.

We target to generate first-person videos from third-person videos via Segmentation Map Guided Cycle-Consistent Generative Adversarial Network.

BOCA - Body-Worn Camera Analytics

We target to generate first-person videos from third-person videos via Segmentation Map Guided Cycle-Consistent Generative Adversarial Network.



BOCA - Body-Worn Camera Analytics

UNIVERSITY OF MICHIGAN

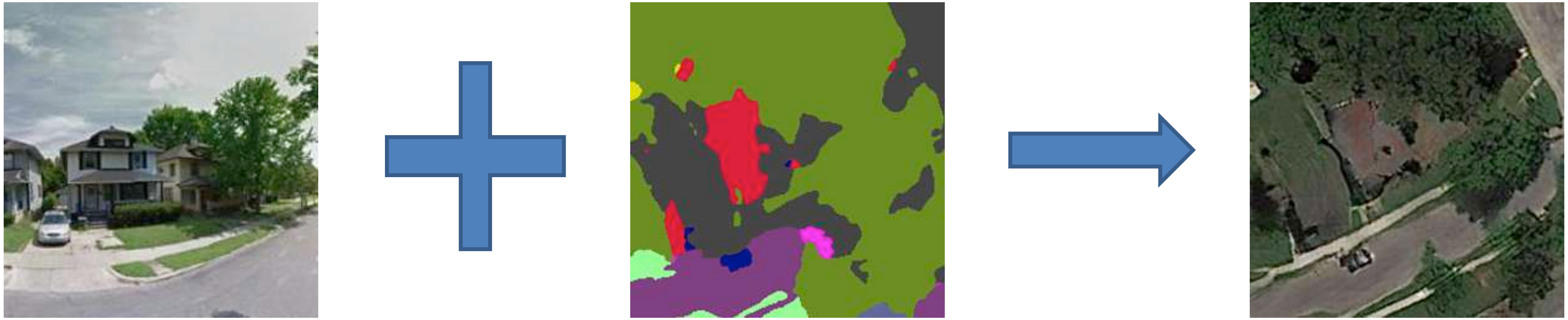
- **Goal:** generate new images from one viewpoint to another.

- **Problem:**

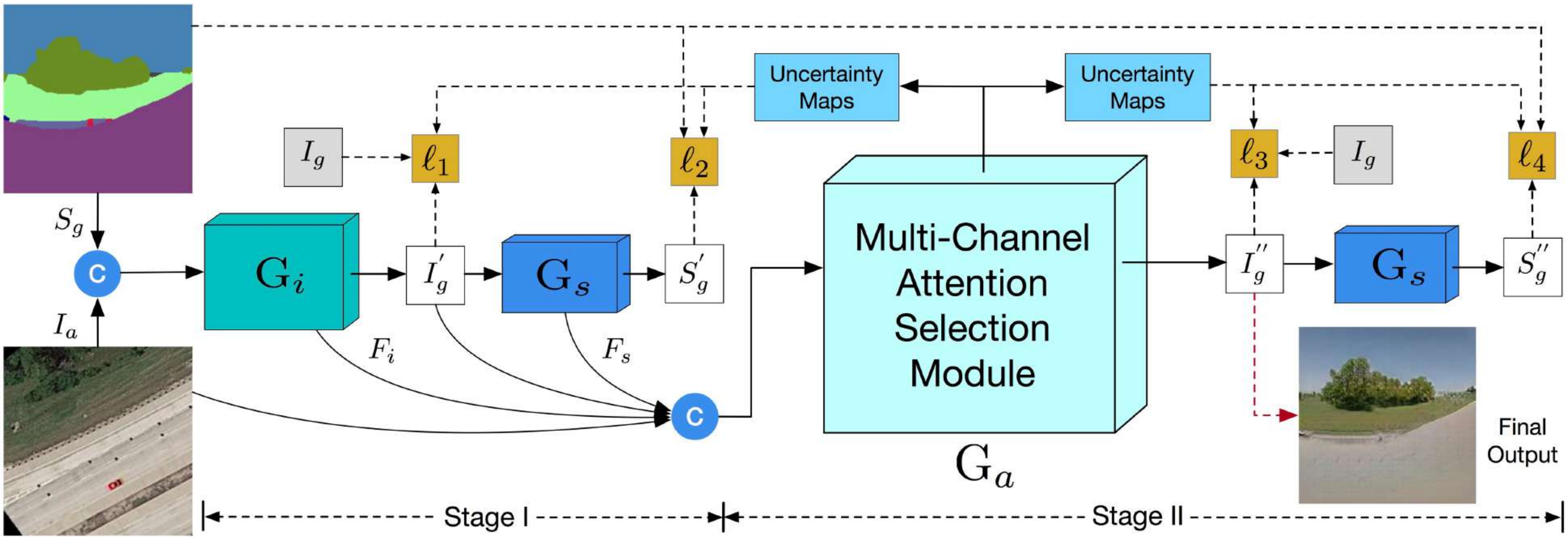
1. Pretrained semantic models
2. Single phase generation
3. Three-channel generation space

- **Key idea:** generate scene images based on an image of the scene and a novel semantic map.

Input Image | Semantic Map | Generated Image

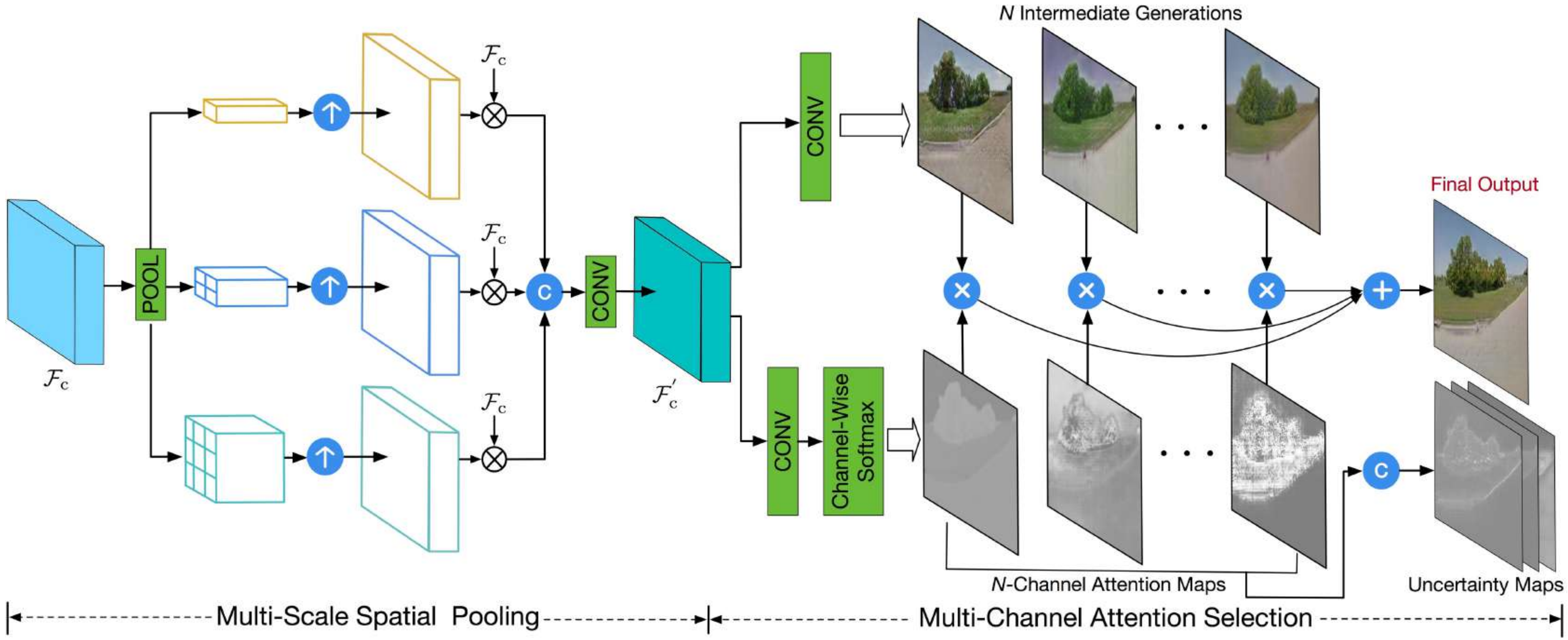


SelectionGAN Framework



Our Proposed Multi-Channel Attention Selection GAN (SelectionGAN) consisting of two stages

Multi-Channel Attention Selection Module



Our Proposed Multi-Channel Attention Selection Module consists of a multi-scale spatial pooling and a multi-channel attention selection component

Ablation Analysis

Baseline	Setup	SSIM	PSNR	SD
A	$I_a \stackrel{G_i}{\rightarrow} I_g^{'}$	0.4555	19.6574	18.8870
B	$S_g \stackrel{G_i}{\rightarrow} I_g^{'}$	0.5223	22.4961	19.2648
C	$[I_a, S_g] \stackrel{G_i}{\rightarrow} I_g^{'}$	0.5374	22.8345	19.2075
D	$[I_a, S_g] \stackrel{G_i}{\rightarrow} I_g^{'} \stackrel{G_s}{\rightarrow} S_g^{'}$	0.5438	22.9773	19.4568
E	D + Uncertainty-Guided Pixel Loss	0.5522	23.0317	19.5127
F	E + Multi-Channel Attention Selection	0.5989	23.7562	20.0000
G	F + Total Variation Regularization	0.6047	23.7956	20.0830
H	G + Multi-Scale Spatial Pooling	**0.6167**	**23.9310**	**20.1214**

SelectionGAN has 8 baselines

Results – Aerial2Ground

Input	Pix2pix	X-Fork	X-Seq	Ours	GT

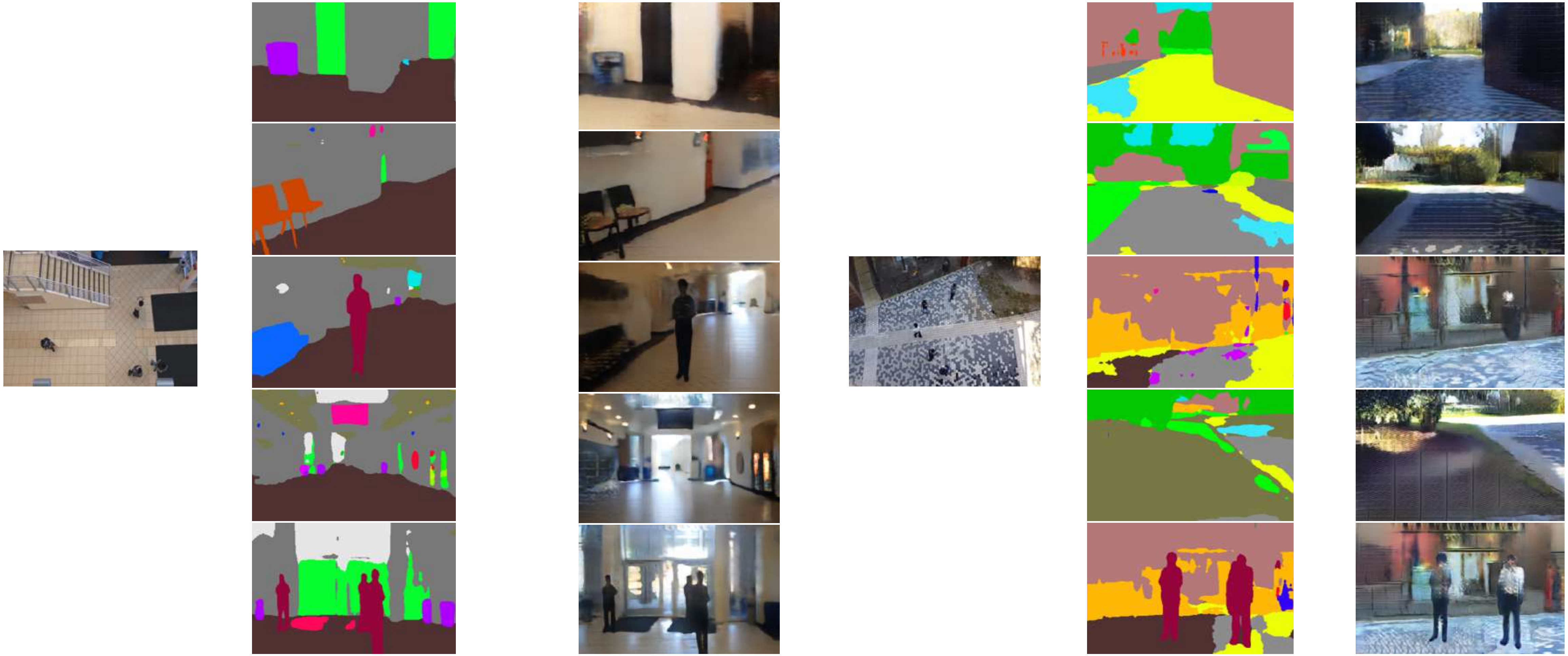
Results – Ground2Aerial

Input	Pix2pix	X-Fork	X-Seq	Ours	GT

BOCA - Body-Worn Camera Analytics

Arbitrary Cross-View Image Translation

Input | Semantic Map | SelectionGAN | Input | Semantic Map | SelectionGAN



BOCA - Body-Worn Camera Analytics

- **Publications:**

[1] H. Tang, W. Wang, D. Xu, Y. Yan, **J. J. Corso**, and N. Sebe. "Attribute-guided Sketch Generation". In Proceedings of IEEE International Conference on Automatic Face and Gesture Recognition. 2019. https://arxiv.org/abs/1901.0974

[2] H. Tang, D. Xu, N. Sebe, Y. Wang, **J. J. Corso**, and Y. Yan. "Multi-Channel Attention Selection GAN with Cascaded Semantic Guidance for Cross-View Image Translation". In Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition. 2019. https://arxiv.org/abs/1904.06807

- **Publications (in submission):**

[3] H. Tang, D. Xu, N. Sebe, **J. J. Corso**, and Y. Yan. "Joint Adversarial Learning Local Class-Specific and Global Image-Level Generation for Cross-View Image Translation". In Proceedings of IEEE International Conference on Computer Vision. 2019.

[4] Y. Yan, C. Xu, D. Cai, **J. J. Corso**. "A Weakly Supervised Multi-task Ranking Framework for Actor-Action Semantic Segmentation", In International Journal of Computer Vision, 2019

[5] K. B. Min and **J. J. Corso**. "TASED-Net: Temporally-Aggregating Spatial Encoder-Decoder Network for Video Saliency Detection", In Proceedings of IEEE International Conference on Computer Vision. 2019.

- **Software:**

[1] E. Hofesman, M. R. Ganesh and **J. J. Corso**. M-PACT. https://github.com/MichiganCOG/M-Pact. 2019.

Conclusions and Summary

- BOCA focuses on two core goals

1. Catalyzing a broader research effort in the challenging problem of video analytics in BWC.
2. Developing advances in understanding activity in BWC.

- Acknowledging NIST PSIAP 60NANB17D191.

Break for

Lunch

BACK AT

1:00PM

Backup

Examples of Third-person Activity

Playing-badminton

Playing-squash

Playing-water-polo

Examples of First-person Activity

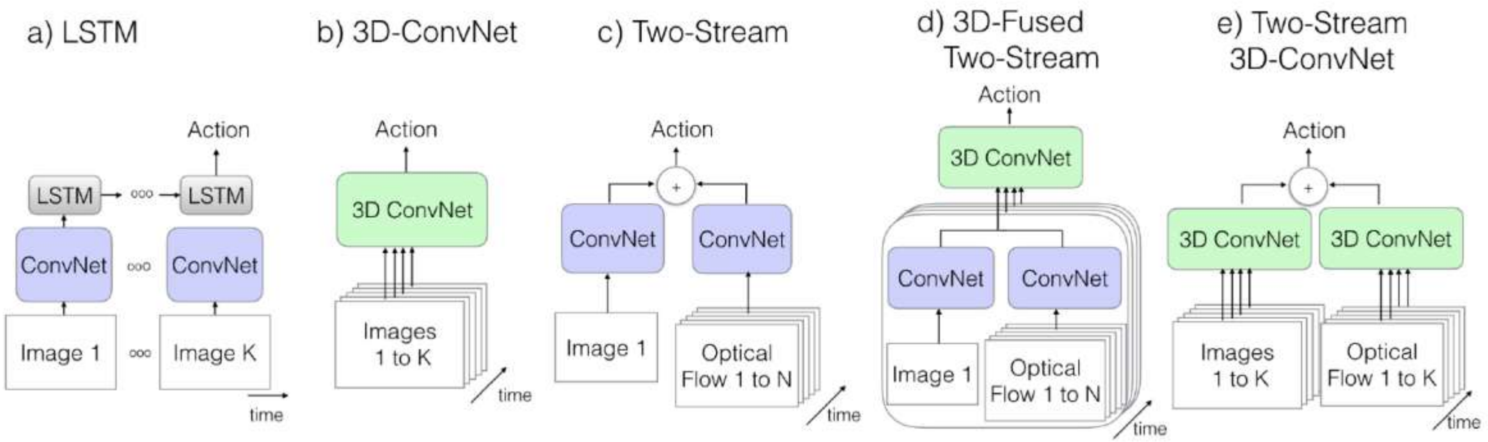
Windsurfing

Playing-squash

Running-a-marathon

SOTA Video Classification Pipeline

ConvNet + LSTM and two-stream network are usually used for video classification.



We use I3D model [1] trained from third-person and evaluated on our collected BOCA dataset.
We observe that there is actually distribution gap between first-person and third-person activities.
(74.2% vs 60.9%, 54.4% vs 50.4%)

	top1_acc	*top5_acc*
First-person	50.4 %	69.4 %
First-person (removing classes doesn't overlap with third-person)	60.9 %	85.1 %
Third-person	54.4 %	66.8 %
Third-person (removing classes doesn't overlap with first-person)	74.2 %	91.3 %

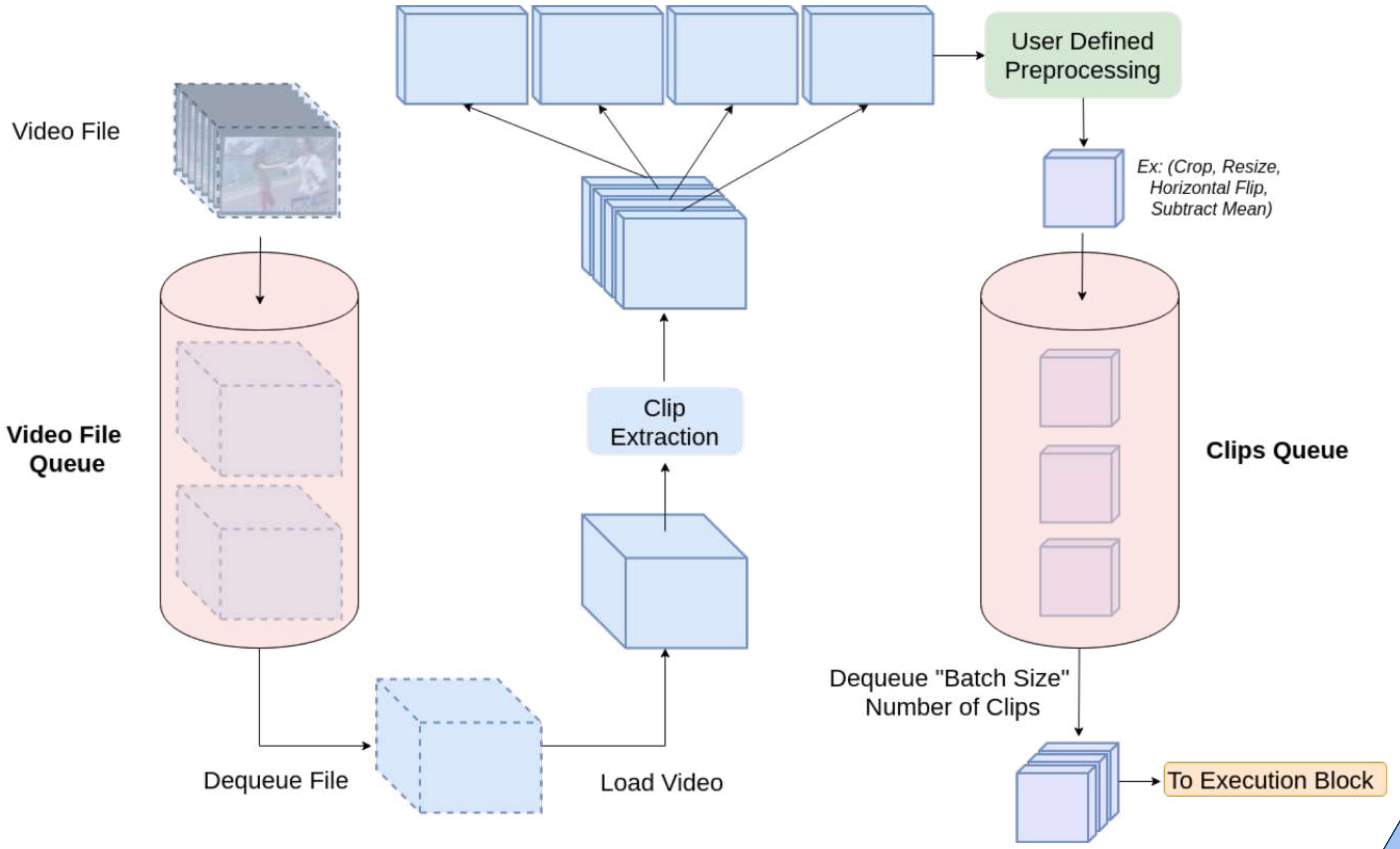
Table 1: I3D model baselines on our collected dataset.

[1] "Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset" by Joao Carreira and Andrew Zisserman, CVPR 2017

We have BOCA Dataset + TPV algorithm. What is Next?

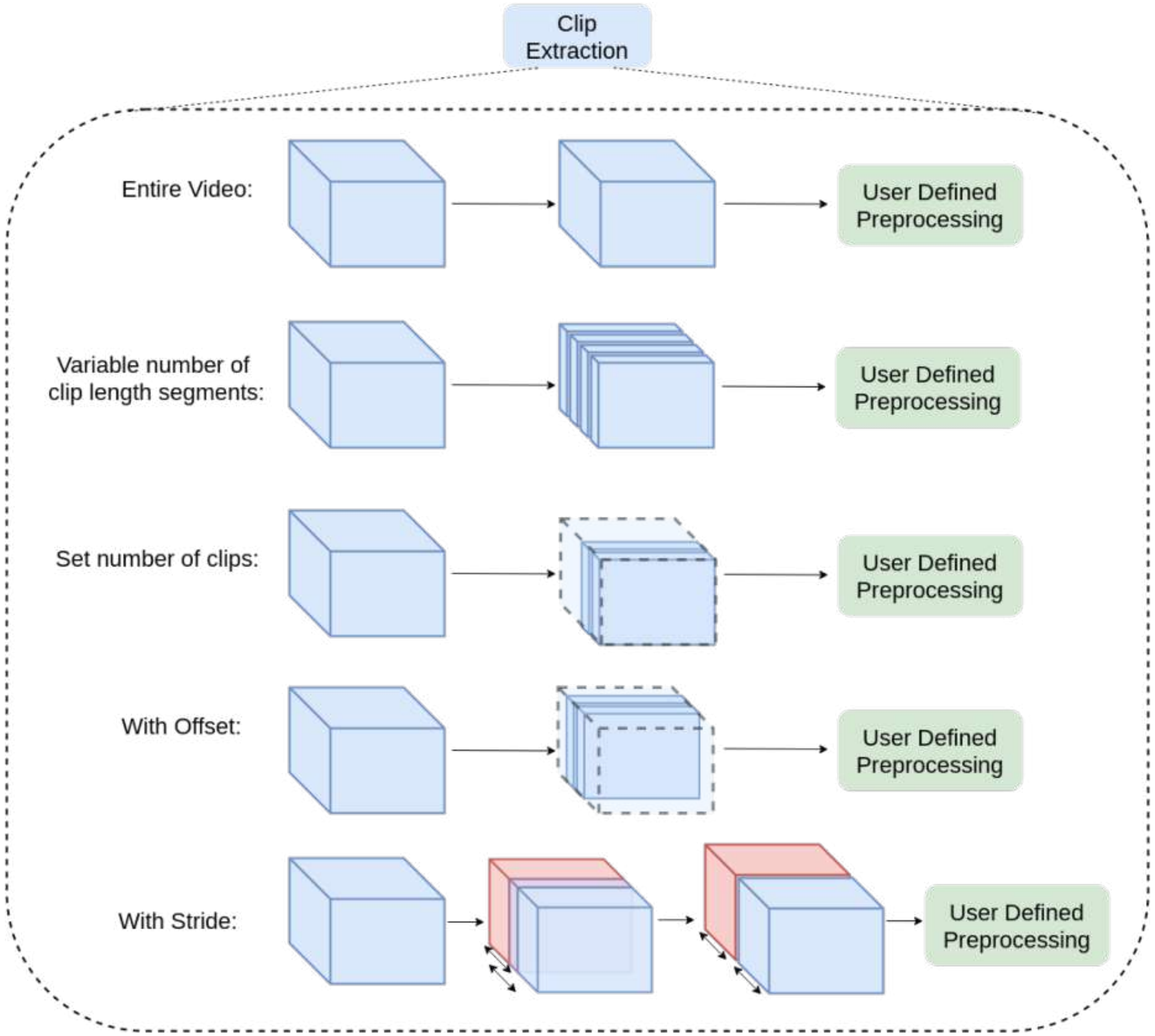
How do we leverage existing knowledge and well-developed models of third-person to assist immature technique first-person activity recognition?

Input Data Block



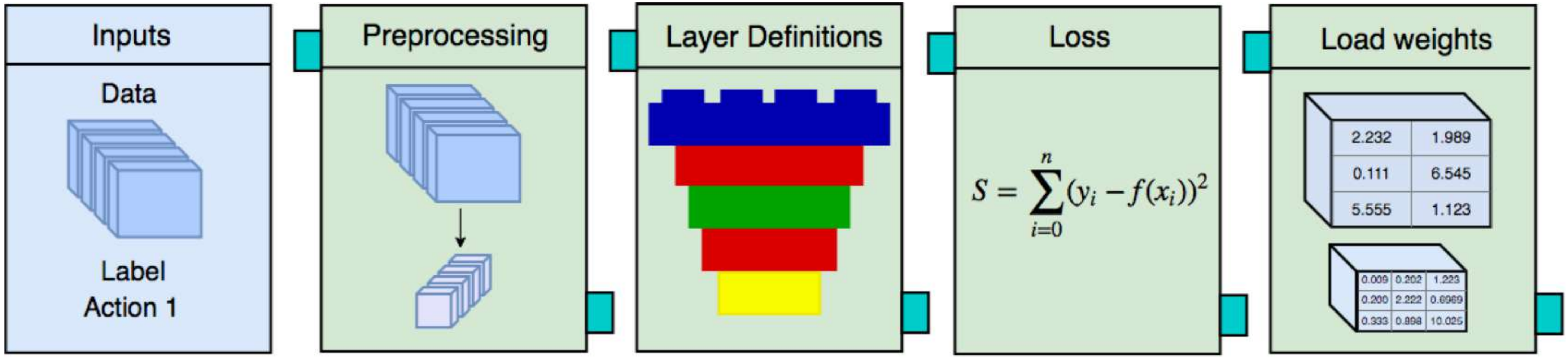
https://github.com/MichiganCOG/M-PACT

Input Data Block



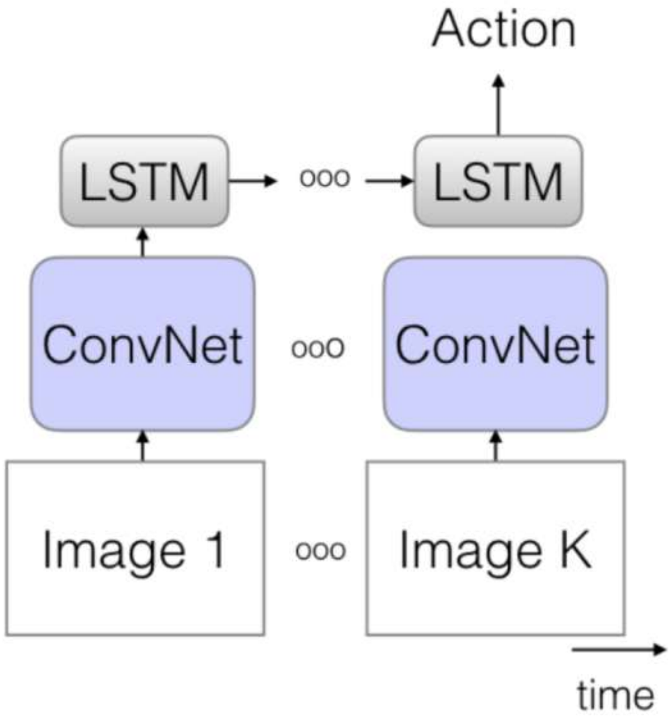
https://github.com/MichiganCOG/M-PACT

Model Definition Block



https://github.com/MichiganCOG/M-PACT

Implemented Model: ResNet50 + LSTM

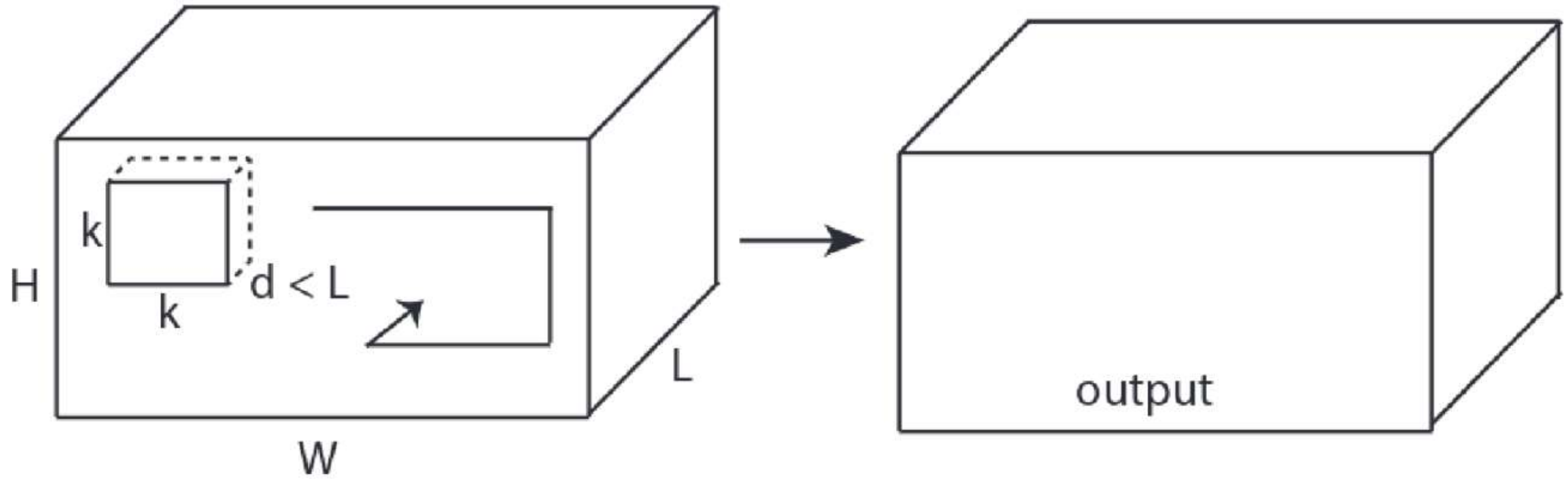


Model	HMDB51 Accuracy (%)		UCF101 Accuracy (%)	
	Orig. Authors	M-PACT	Orig. Authors	M-PACT
ResNet50 + LSTM	43.90	43.86	84.30	80.20

Donahue J. et al. CVPR 2015

https://github.com/MichiganCOG/M-PACT

Implemented Model: C3D

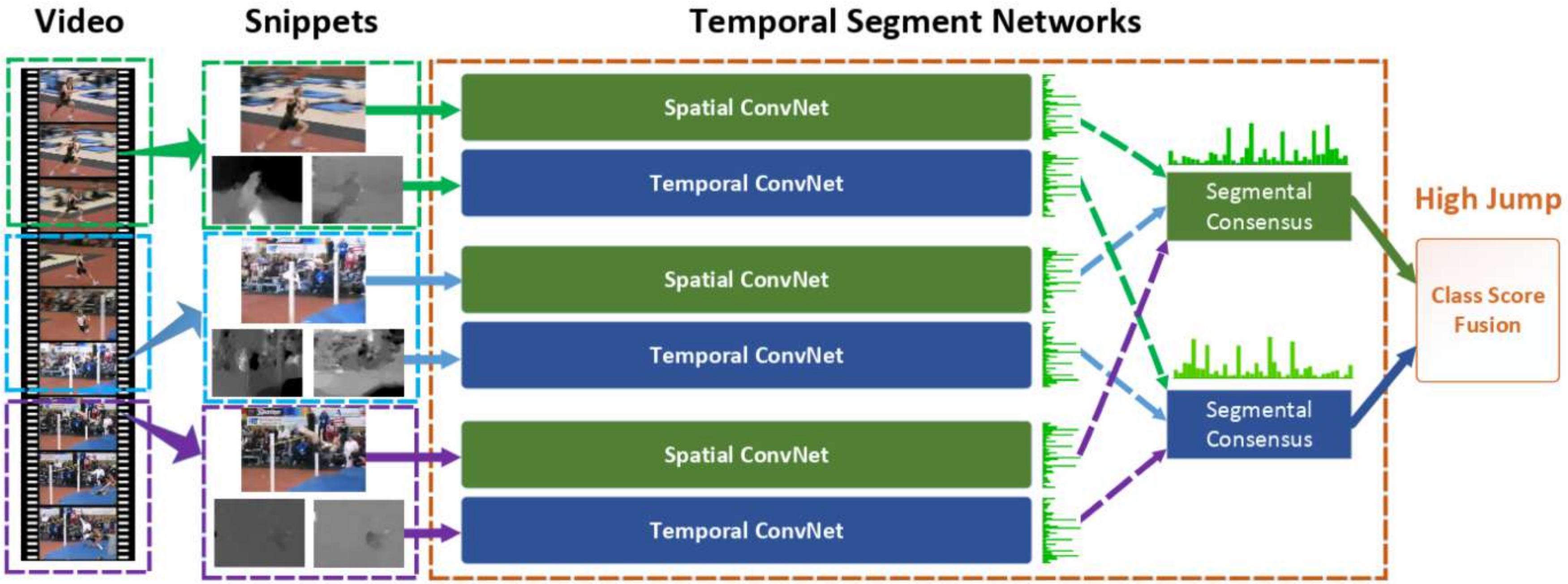


Model	HMDB51 Accuracy (%)		UCF101 Accuracy (%)	
	Orig. Authors	M-PACT	Orig. Authors	M-PACT
C3D	50.30	51.90	82.30	93.66

Tran D. et al. ICCV 2015

https://github.com/MichiganCOG/M-PACT

Implemented Model: TSN



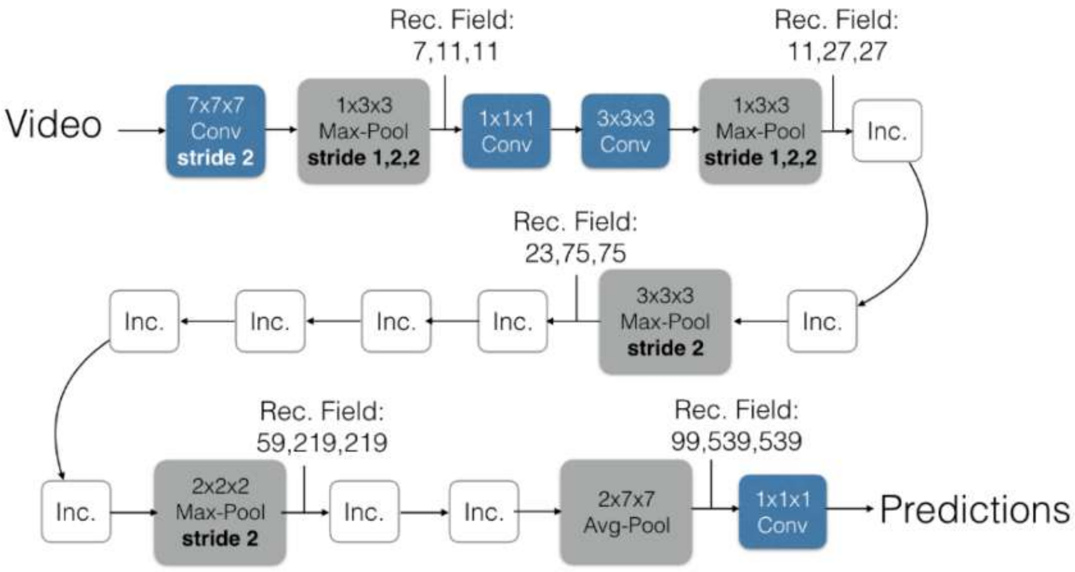
Model	HMDB51 Accuracy (%)		UCF101 Accuracy (%)	
	Orig. Authors	M-PACT	Orig. Authors	M-PACT
TSN	54.40	51.70	85.50	85.25

Wang L. et al. ECCV 2016

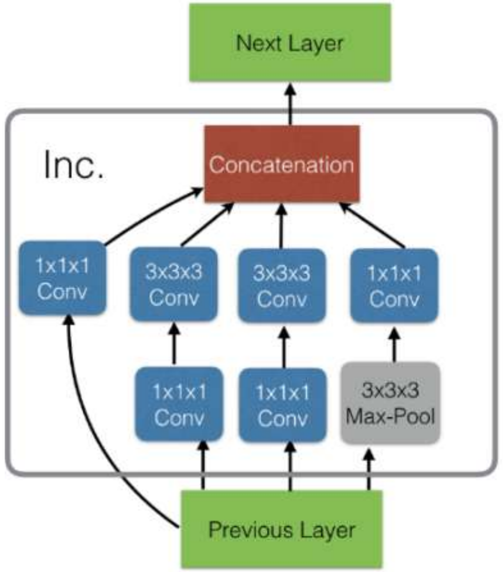
https://github.com/MichiganCOG/M-PACT

Implemented Model: I3D

Inflated Inception-V1



Inception Module (Inc.)

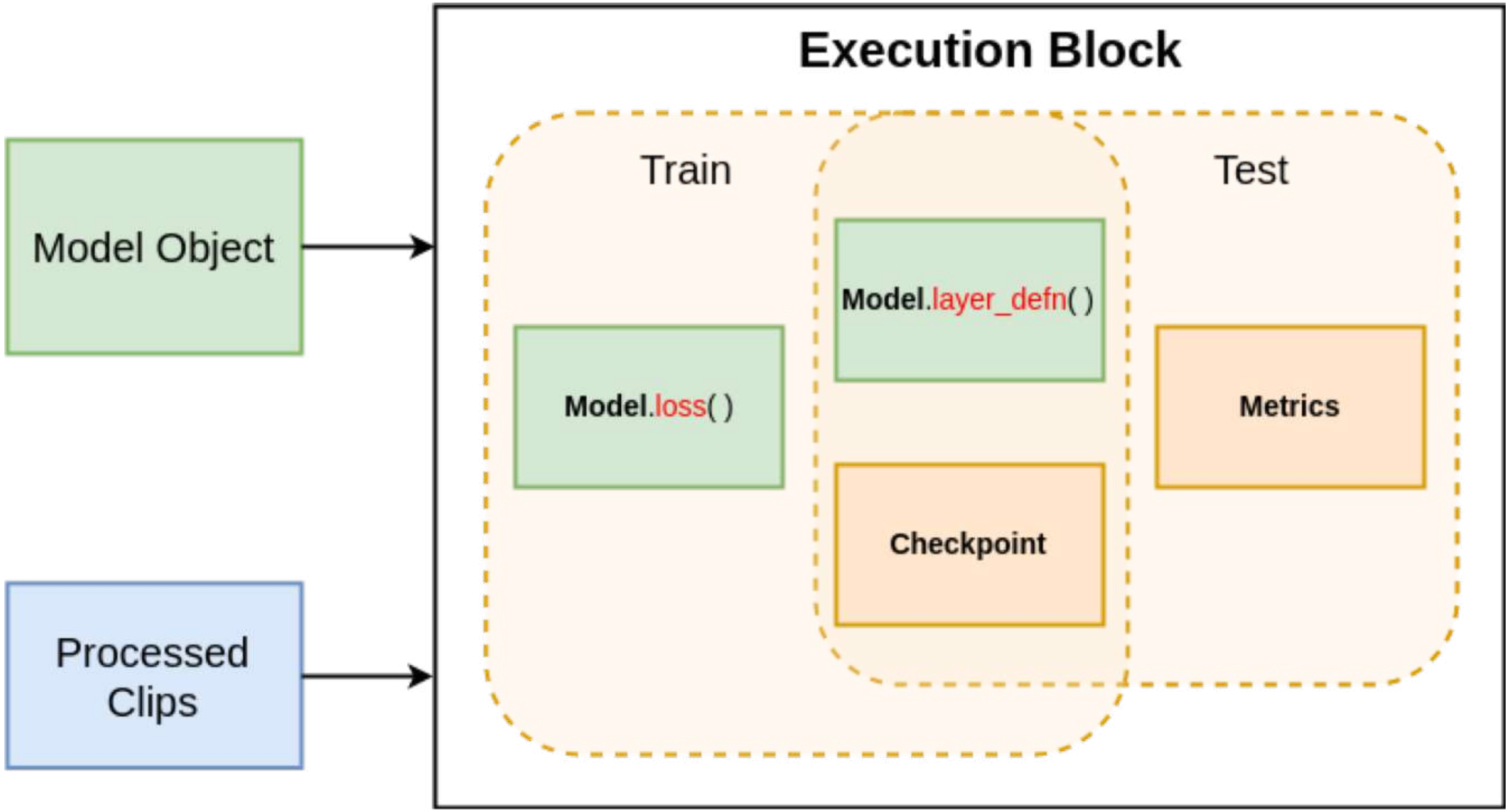


Model	HMDB51 Accuracy (%)		UCF101 Accuracy (%)	
	Orig. Authors	M-PACT	Orig. Authors	M-PACT
I3D	74.80	68.10	95.60	92.55

Carreira J. et al. CVPR 2017

https://github.com/MichiganCOG/M-PACT

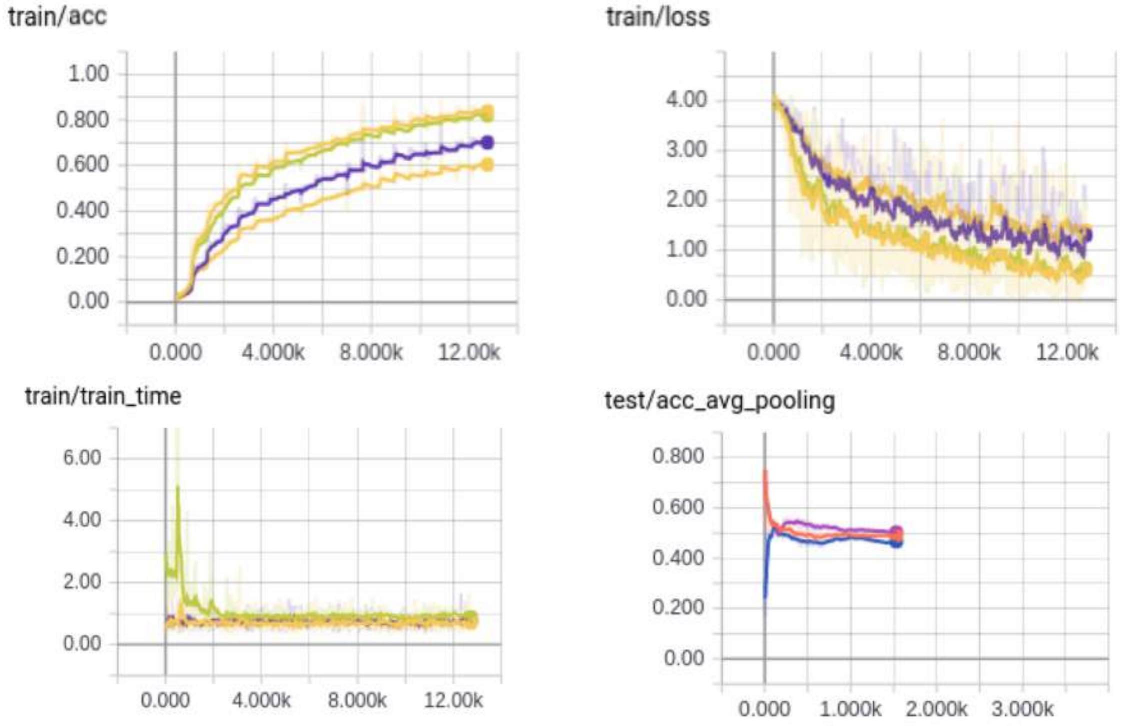
Execution Block



https://github.com/MichiganCOG/M-PACT

Execution Block - Metrics

- Classification metrics include:
  - Average pooling
  - Classification using the last frame of the input
  - Linear SVM
- Internal scalar tensorboard logging
- Feature extraction and storage



https://github.com/MichiganCOG/M-PACT

Results

Model	HMDB51 Accuracy (%)		UCF101 Accuracy (%)	
	Orig. Authors	M-PACT	Orig. Authors	M-PACT
I3D	74.80	68.10	95.60	92.55
C3D	50.30	51.90	82.30	93.66
TSN	54.40	51.70	85.50	85.25
ResNet50 + LSTM	43.90	43.86	84.30	80.20

https://github.com/MichiganCOG/M-PACT

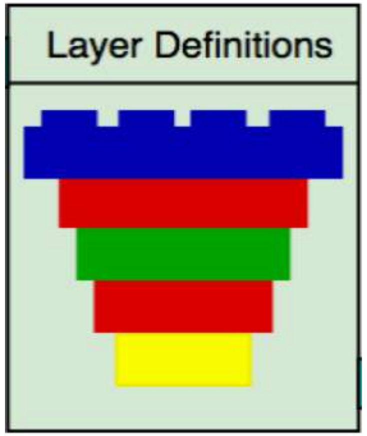
Easy to use

- Download Dataset
- Format Dataset using tfrecords
  - python utils/generate_tfrecords_dataset.py
  - --videos_dir /dir/to/dataset/vids
    - --save_dir /dir/to/save/tfrecords_dataset
- Download trained model weights
  - sh scripts/shell/download_weights.sh
- Train models
  - python train.py --model I3D --dataset UCF101 --inputDims 64 --outputDims 101 --seqLength 1 --size 224 --expName i3d_train --numVids 9537 --baseDataPath /tfrecords_dataset --fName trainlist
- Test models
  - python test.py --model resnet --dataset HMDB51 --loadedDataset HMDB51 --inputDims 50 --outputDims 51 --seqLength 50 --size 224 --expName resnet_test --numVids 1530 --baseDataPath /tfrecords_dataset --fName testlist

https://github.com/MichiganCOG/M-PACT

Easy to use - Add model using template

```
class MODELNAME(Abstract_Model_Class):

    def __init__(self, **kwargs):
        """
        Args:
            Pass all arguments on to parent class, you may not add additional arguments without modifying abstract_model_class.py,
 Models.py, train.py, and test.py. Enter any additional initialization functionality here if desired.
        """
        super(MODELNAME, self).__init__(**kwargs)
```

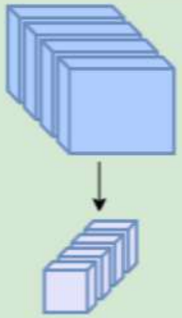



```
    def inference(self, inputs, is_training, input_dims, output_dims, seq_length, batch_size, scope, dropout_rate = 0.5, return_la
yer=['logits'], weight_decay=0.0):

        with tf.name_scope(scope, 'MODELNAME', [inputs]):
            layers = {}

            ##########################################################################################
            #       TODO: Add any desired layers from layers_utils to this layers dictionary          #
            #                                                                                         #
            #       EX: layers['conv1'] = conv3d_layer(input_tensor=inputs,                           #
            #           filter_dims=[dim1, dim2, dim3, dim4],                                         #
            #           name=NAME,                                                                    #
            #           weight_decay = wd)                                                            #
            ##########################################################################################


            ##########################################################################################
            #       TODO: Final Layer must be 'logits'                                                #
            #                                                                                         #
            #  EX:  layers['logits'] = [fully_connected_layer(input_tensor=layers['previous'],         #
            #                                out_dim=output_dims, non_linear_fn=None,                 #
            #                                name='out', weight_decay=weight_decay)]                  #
            ##########################################################################################

            layers['logits'] = # TODO Every model must return a layer named 'logits'

            layers['logits'] = tf.reshape(layers['logits'], [batch_size, seq_length, output_dims])

        # END WITH

        return [layers[x] for x in return_layer]
```

https://github.com/MichiganCOG/M-PACT

Easy to use - Add preprocessing and loss

Preprocessing



Loss

$$S = \sum_{i=0}^{n} (y_i - f(x_i))^2$$

```
    def preprocess_tfrecords(self, input_data_tensor, frames, height, width, channel, input_dims, output_dims, seq_length, size, l
abel, istraining, video_step):
        """
        Args:
            :input_data_tensor:     Data loaded from tfrecords containing either video or clips
            :frames:                Number of frames in loaded video or clip
            :height:                Pixel height of loaded video or clip
            :width:                 Pixel width of loaded video or clip
            :channel:               Number of channels in video or clip, usually 3 (RGB)
            :input_dims:            Number of frames used in input
            :output_dims:           Integer number of classes in current dataset
            :seq_length:            Length of output sequence
            :size:                  List detailing values of height and width for final frames
            :label:                 Label for loaded data
            :is_training:           Boolean value indication phase (TRAIN OR TEST)
            :video_step:            Tensorflow variable indicating the total number of videos (not clips) that have been loaded
        """

        ####################################################
        # TODO: Add more preprcessing arguments if desired #
        ####################################################

        return preprocess(input_data_tensor, frames, height, width, channel, input_dims, output_dims, seq_length, size, label, ist
raining, video_step, self.input_alpha)


    """ Function to return loss calculated on given network """
    def loss(self, logits, labels, loss_type):
        """
        Args:
           :logits:     Unscaled logits returned from final layer in model
           :labels:     True labels corresponding to loaded data
           :loss_type:  Allow for multiple losses that can be selected at run time. Implemented through if statements
        """

        ####################################################################################
        #  TODO: ADD CUSTOM LOSS HERE, DEFAULT IS CROSS ENTROPY LOSS                       #
        #                                                                                  #
        #   EX: labels = tf.cast(labels, tf.int64)                                         #
        #       cross_entropy_loss = tf.losses.sparse_softmax_cross_entropy(labels=labels, #
        #                                                    logits=logits)                #
        #        return cross_entropy_loss                                                 #
        ####################################################################################
                                                                       85,1          Bot
```

https://github.com/MichiganCOG/M-PACT

Where?

- https://github.com/MichiganCOG/M-PACT

M-PACT: Michigan Platform for Activity Classification in Tensorflow

This python framework provides modular access to common activity recognition models for the use of baseline comparisons between the current state of the art and custom models.
This README will walk you through the process of installing dependencies, downloading and formatting datasets, testing the framework, and expanding the framework to train your own models.

This repository holds the code and models for the paper
M-PACT: Michigan Platform for Activity Classification in Tensorflow, Eric Hofesmann, Madan Ravi Ganesh, and Jason J. Corso, arXiv, April 2018.

ATTENTION: Please cite the arXiv paper introducing this platform when releasing any work that used this code.
Link: https://arxiv.org/abs/1804.05879

Implemented Model's Classification Accuracy:

Model Architecture	Dataset (Split 1)	M-PACT Accuracy (%)	Original Authors Accuracy (%)
I3D	HMDB51	68.10	74.80*
C3D	HMDB51	51.90	50.30*
TSN	HMDB51	51.70	54.40
ResNet50 + LSTM	HMDB51	43.86	43.90
I3D	UCF101	92.55	95.60*
C3D	UCF101	93.66	82.30*
TSN	UCF101	85.25	85.50
ResNet50 + LSTM	UCF101	80.20	84.30

(*) Indicates that results are shown across all three splits

Table of Contents

Framework File Structure

```
/tf-activity-recognition-framework
   train.py
   test.py
   create_model.py
   load_a_video.py

   /models
        /model_name
            modelname_model.py
            default_preprocessing.py
            model_weights.npy shortcut to ../weights/model_weights.npy (Optional)

        /weights
            model_weights.npy

   /results
        /model_name
            /dataset_name
                /preprocessing_method
                    /experiment_name
                            /checkpoints
                                checkpoint
                                checkpoint-100.npy
                                checkpoint-100.dat
                            /metrics_method
                                testing_results.npy

    /logs
        /model_name
            /dataset_name
                /preprocessing_method
                    /metrics_method
                        /experiment_name
                            tensorboard_log

    /scripts
        /shell
            download_weights.sh

    /utils
        generate_tfrecords_dataset.py
        convert_checkpoint.py
        checkpoint_utils.py
        layers_utils.py
        metrics_utils.py
        preprocessing_utils.py
        sys_utils.py
        logger.py
```

M-PACT Additions: T-RECS

Training for Rate-Invariant Embeddings by Controlling Speed

Original Video Speed

α = 0.6 | α = 0.8 | α = 1.0 | α = 1.2

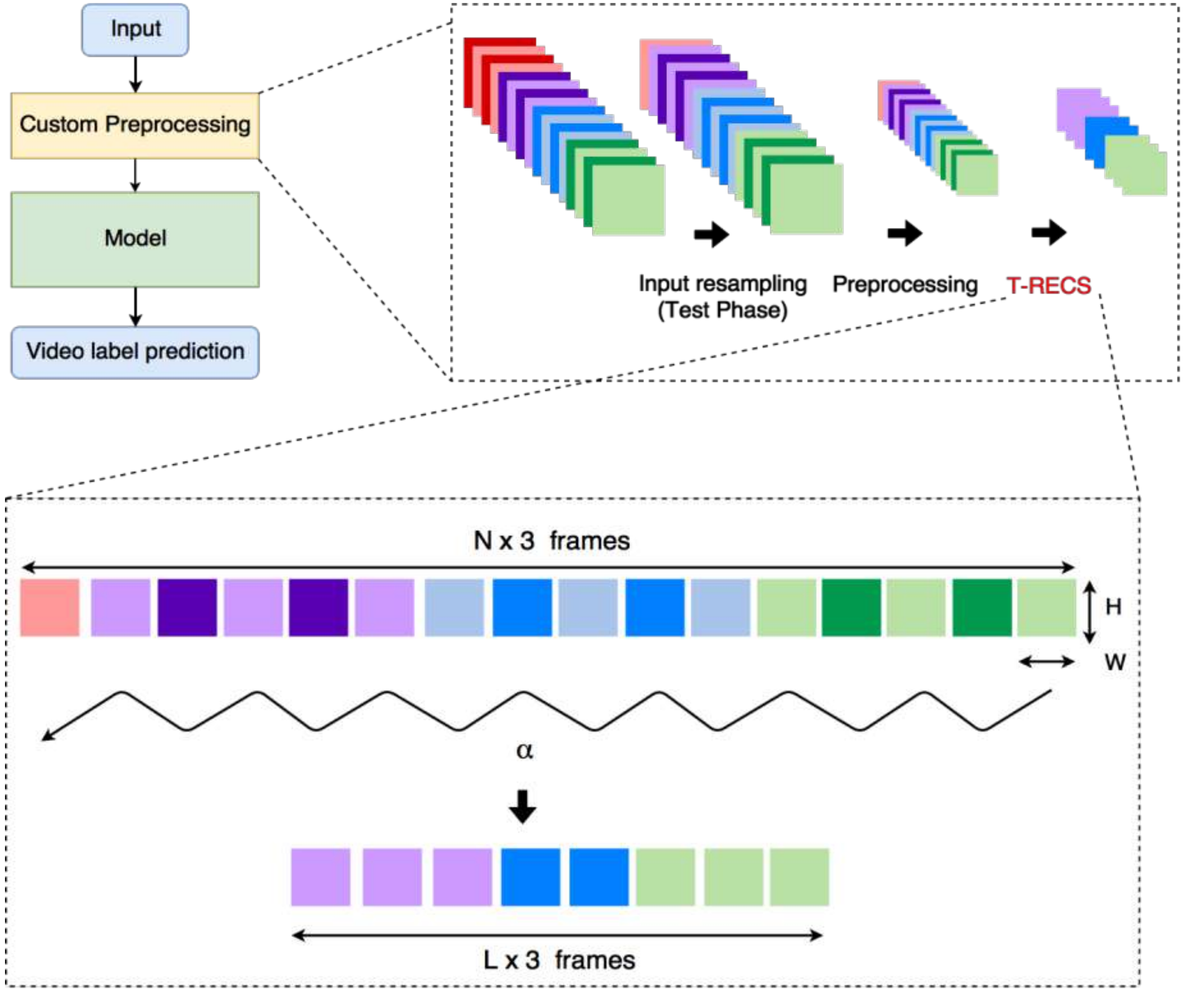
I3D Prediction:

Shoot Bow | Shoot Bow | Hug | Shoot Ball

Ravi Ganesh M., Hofesmann E., Min K., Gafoor N., Corso J., ArXiv 2018

M-PACT Additions: T-RECS

Training for Rate-Invariant Embeddings by Controlling Speed

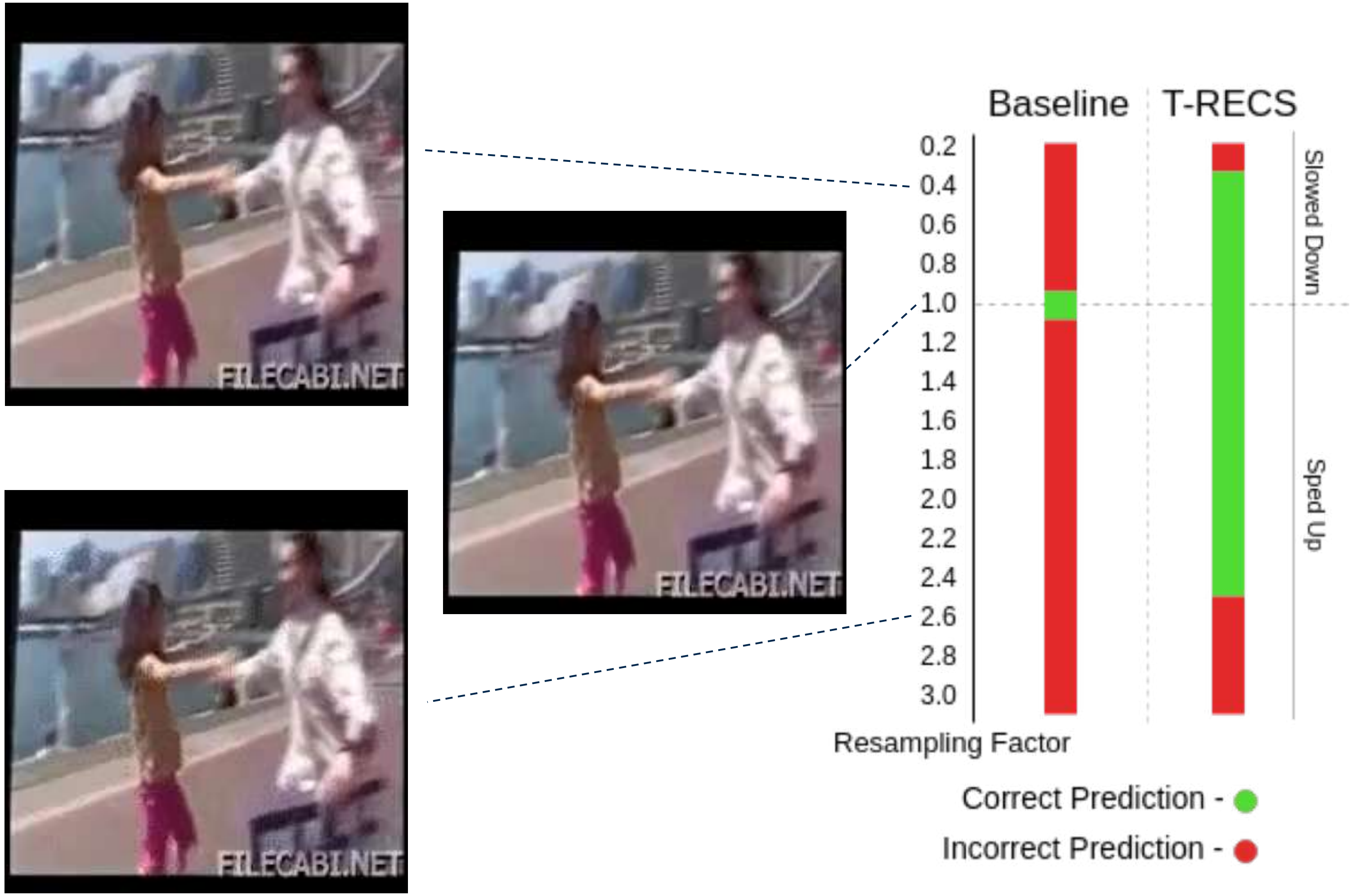


Ravi Ganesh M., Hofesmann E., Min K., Gafoor N., Corso J., ArXiv 2018

BOCA - Body-Worn Camera Analytics

M-PACT Additions: T-RECS

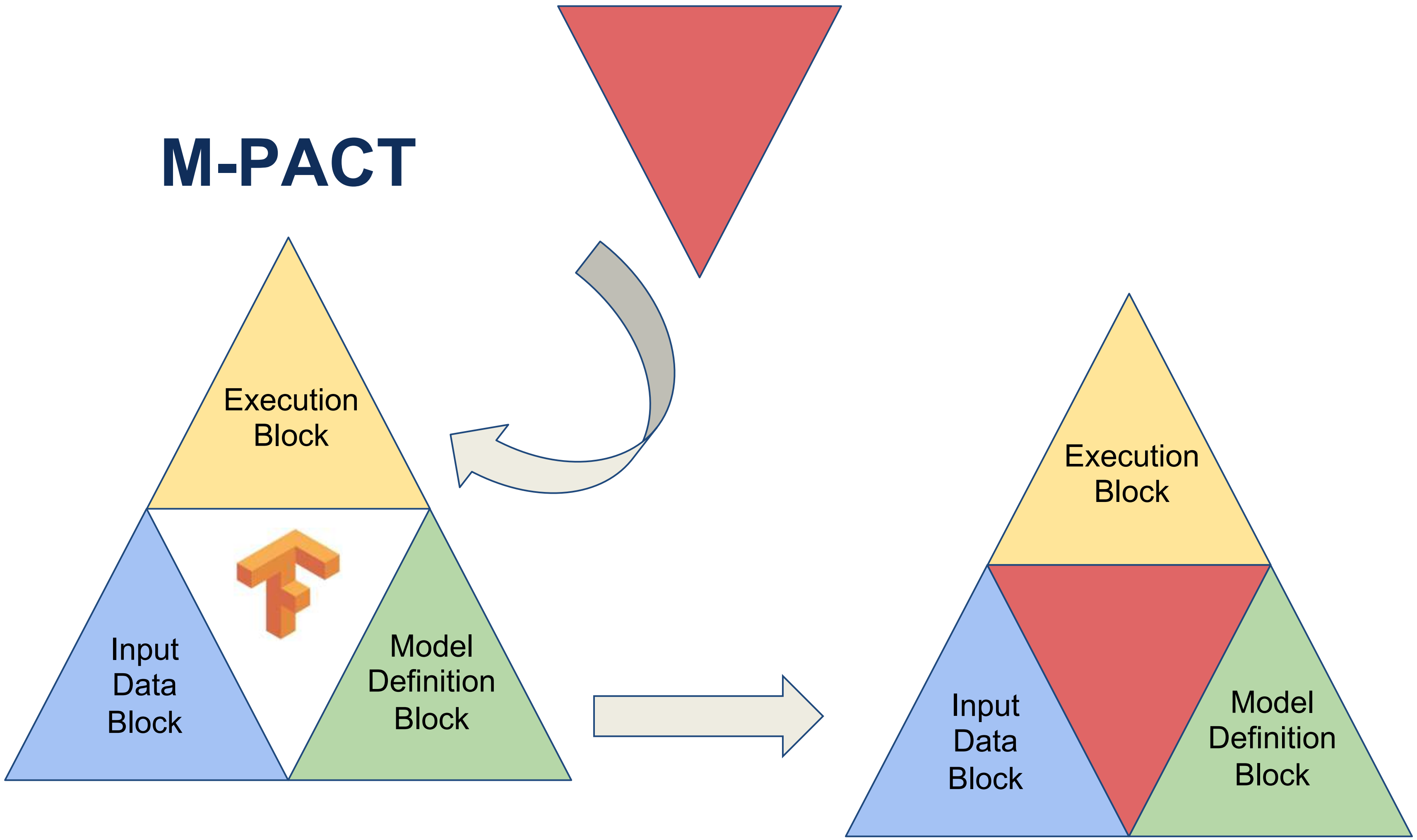
Training for Rate-Invariant Embeddings by Controlling Speed



Ravi Ganesh M., Hofesmann E., Min K., Gafoor N., Corso J., ArXiv 2018

M-PACT as a platform for BOCA

BOCA Transfer Learning Method



Transfer Learning

Dog/Cat Classifier

cat

dog

Data ***<u>not directly related to</u>*** the task considered

elephant

tiger

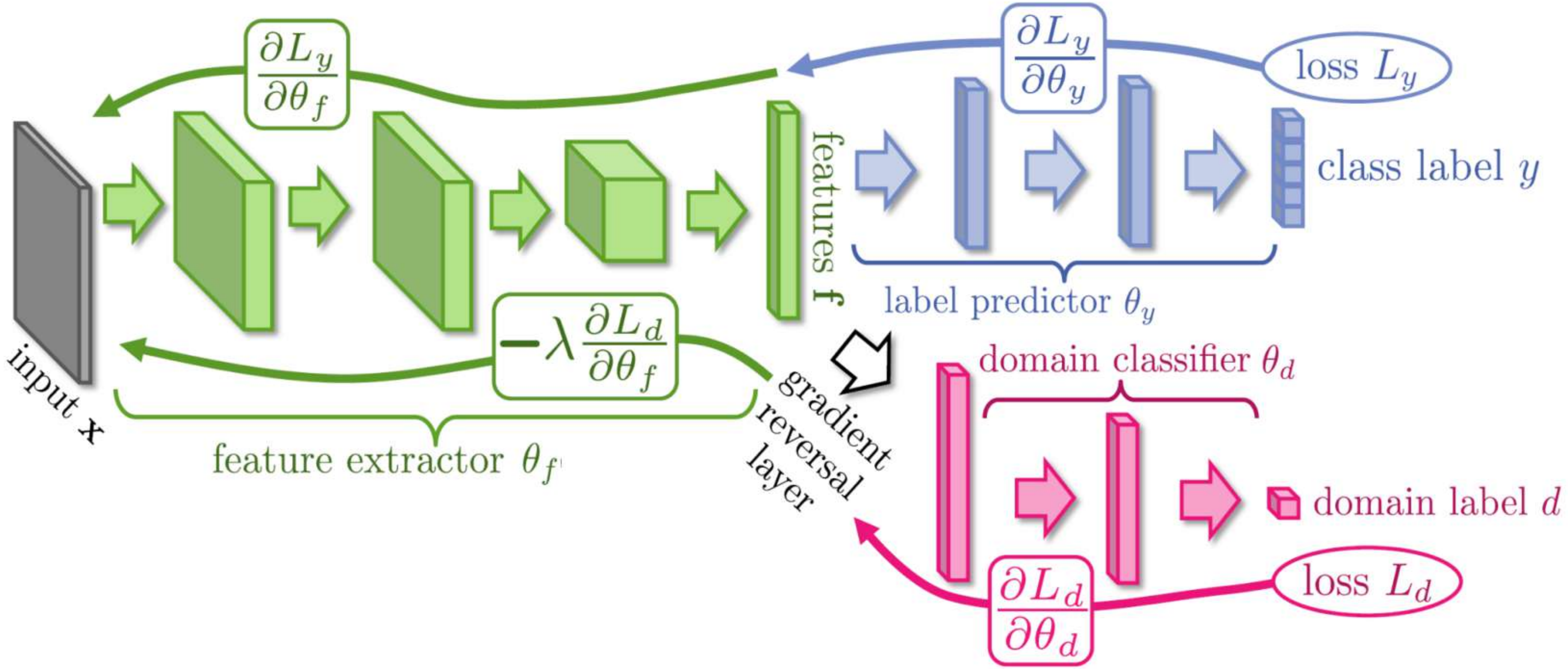
dog

cat

Similar domain, different tasks

Different domains, same task

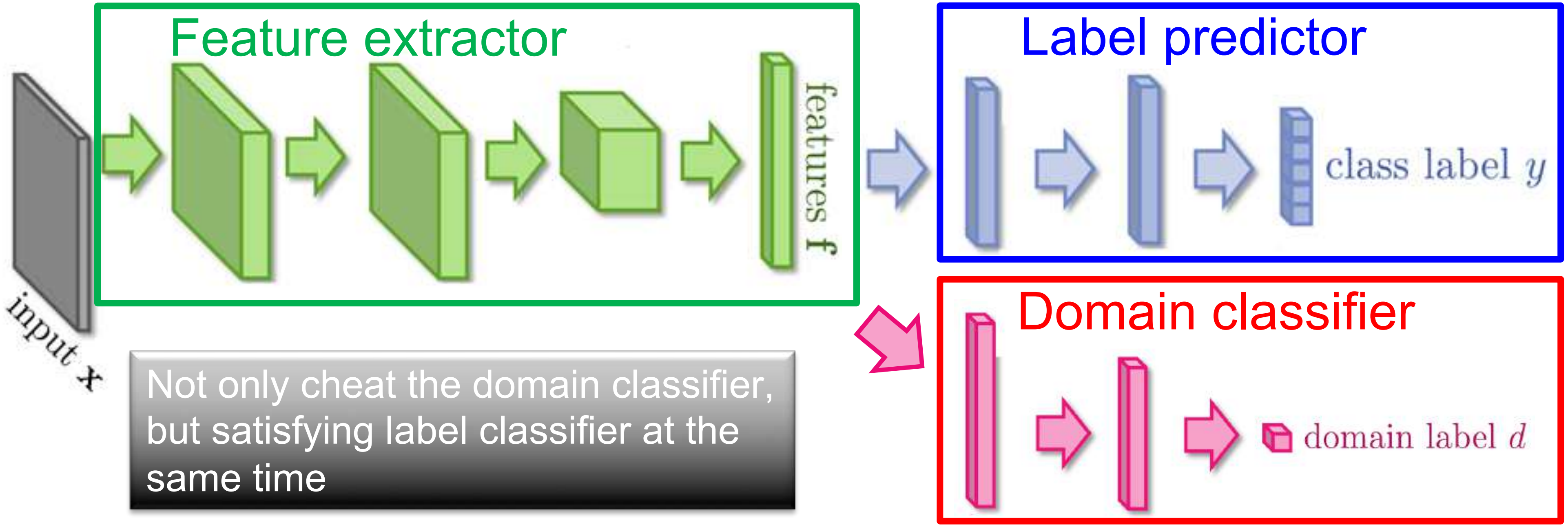
Domain-adversarial Transfer Learning



Domain-adversarial Transfer Learning

Maximize label classification accuracy + minimize domain classification accuracy

Maximize label classification accuracy



Not only cheat the domain classifier, but satisfying label classifier at the same time

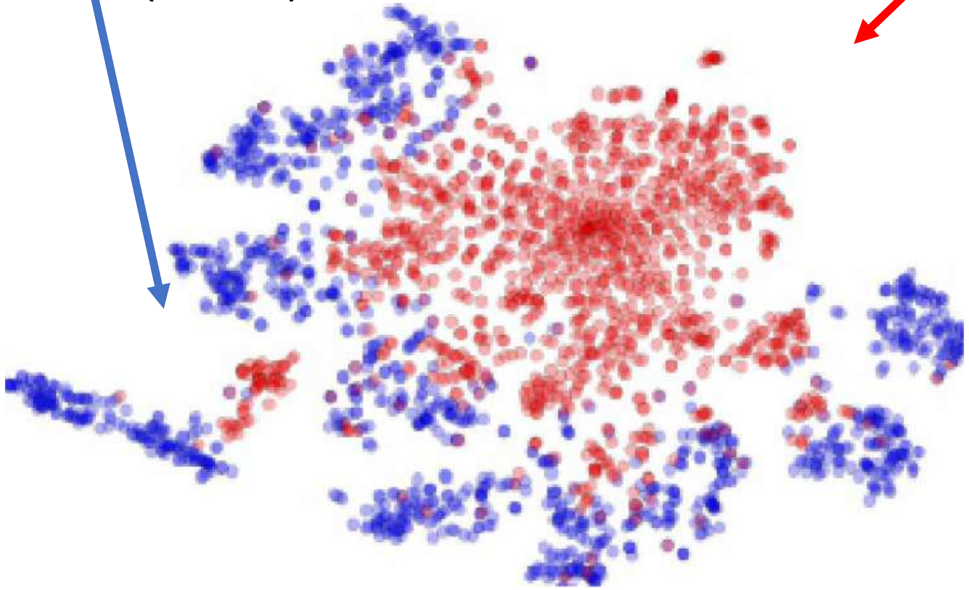
Maximize domain classification accuracy

This is a big network, but different parts have different goals.

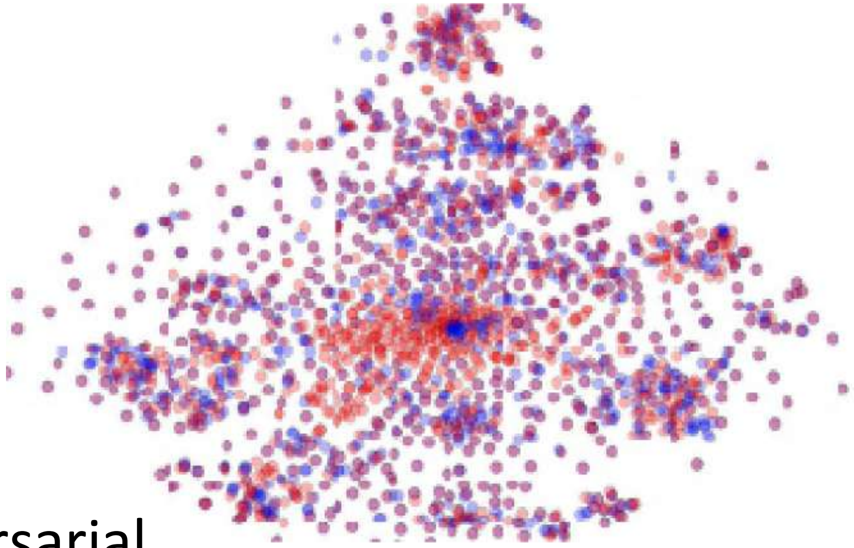
Domain-adversarial Transfer Learning

Third-person
(source)

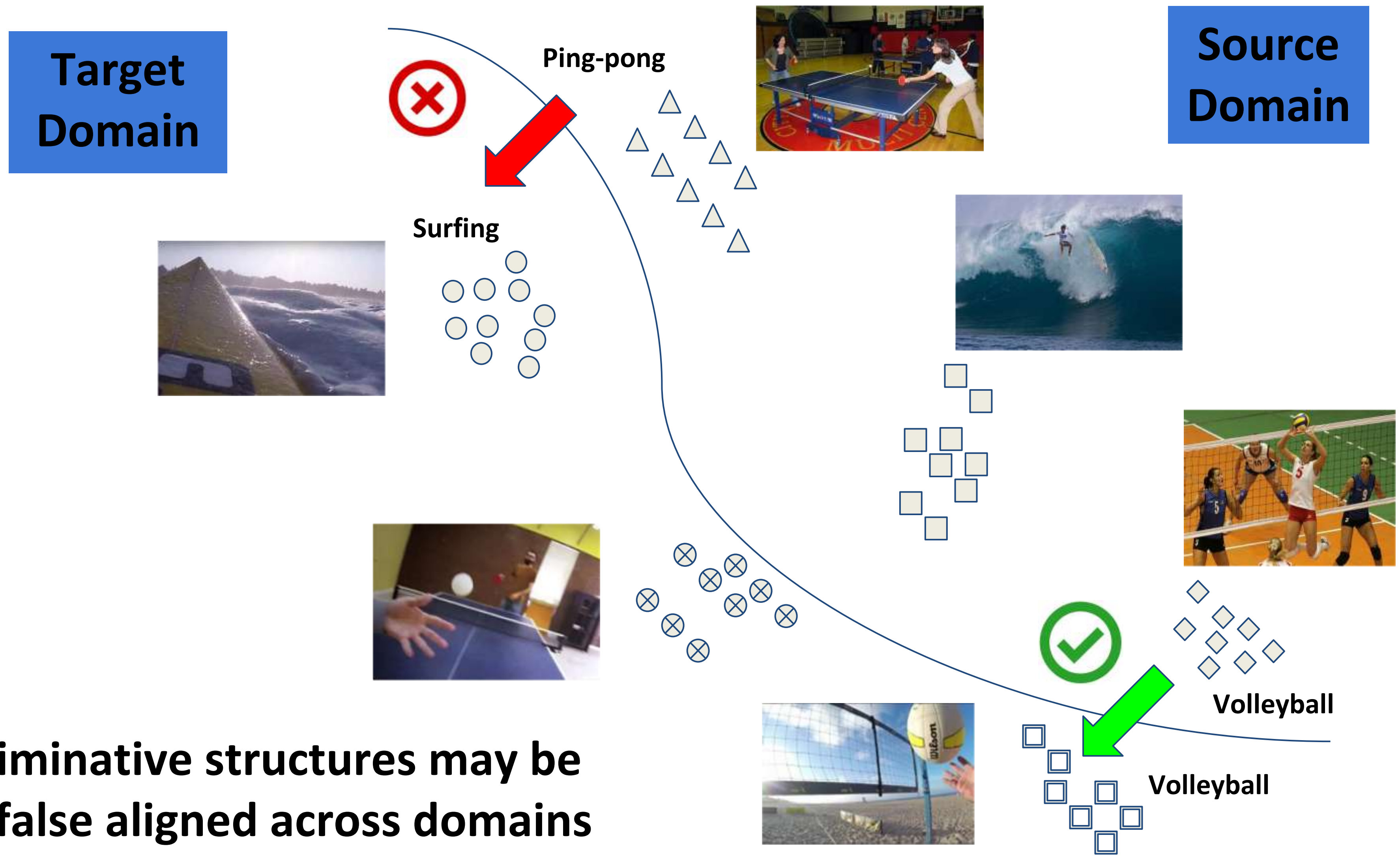
First-person
(Target)



Domain-adversarial
Transfer Learning

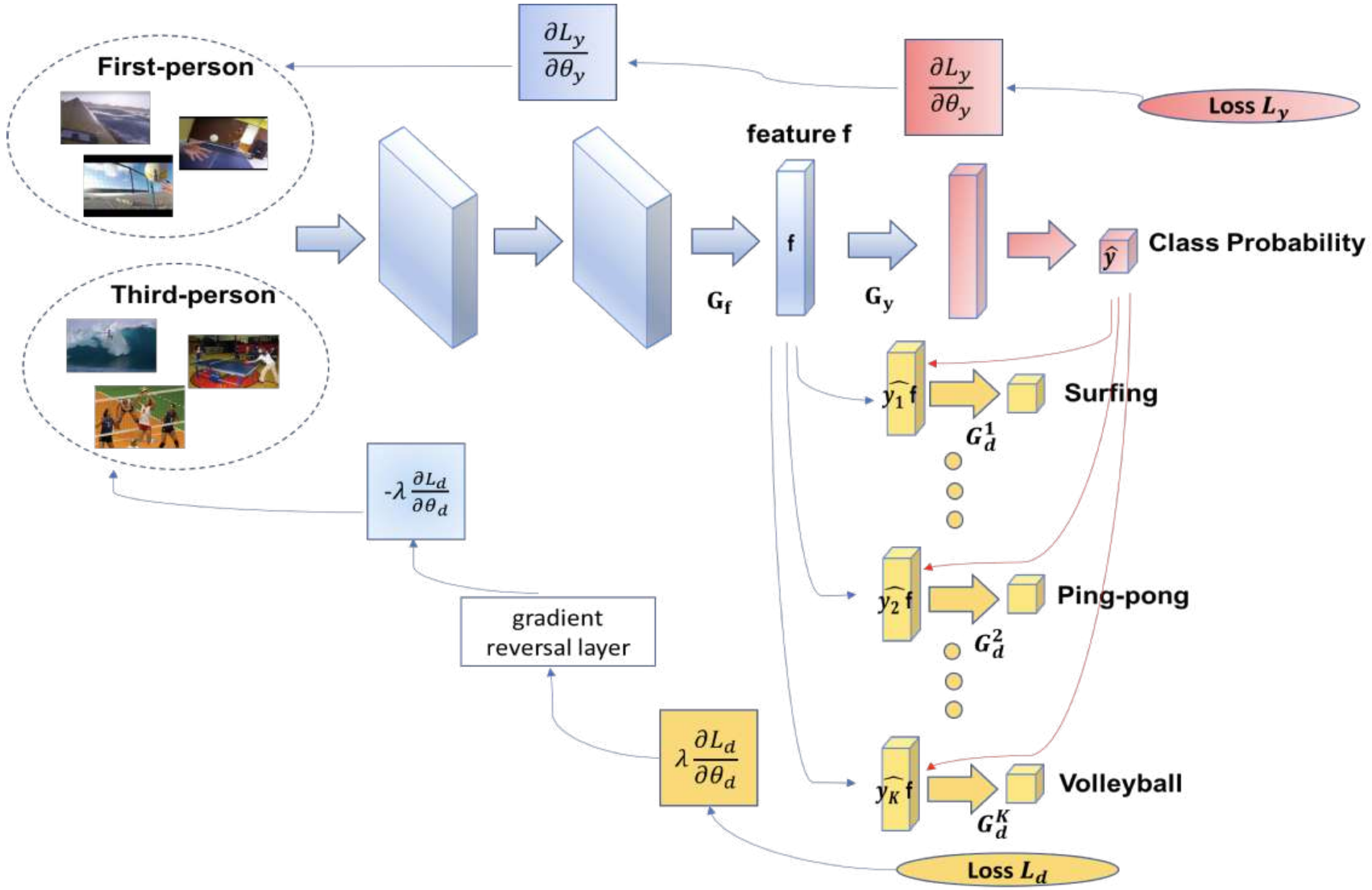


Transfer Learning Difficulty



The discriminative structures may be mixed or false aligned across domains

Multi-adversarial Unsupervised Domain Adaptation



We propose a multi-adversarial domain adaptation networks approach for unsupervised transfer learning by extracting transferable features that can reduce the distribution shift between the source third-person domain and the target first-person domain.

Results

	$top1\_acc$	$top5\_acc$
I3D: First-person	50.4 %	69.4 %
I3D: Third-person	54.4 %	66.8 %
3rd to 1st	51.6 %	72.3 %
1st to 3rd	53.5 %	64.6 %

Table 2: I3D model baselines and domain adversarial training performance on our collected dataset.

Some preliminary results of top-1 and top-5 accuracy are shown in the Table.

- We observe that the performance increases if we transfer knowledge from third-person to first-person.
- However, the performance drops if we adapt first-person to third-person which is a bit strange. We will investigate if there are some bugs or analyze the reason of this happening in the next step.

Successful Transfer case

Scene

Person Movement Patterns

Number of players in the activity

Third-person

First-person

Successful cases

Failure cases

Involved in less important parts for activity

Too much foreground occupied by human

Less information for the scene of marathon

Not engaged in the activity frequently