Violence Detection

These violent delights have violent ends…

- W.S.

Team Bittah Ninja

Jen, Lance, Alex, Ahsen

Disclaimer: Viewers might see images or videos that contain violence

What Are We Doing?

- Classifying human behavior as violent or non-violent in real-time on edge enabled CCTV devices using action recognition models
- Shorten the time between violent action and deployment of de-escalation measures
- POC model detects violent or non-violent punches

Why Are We Doing It?

- Security professionals, judged on response times, spend hours with eyes glued to camera feed
- Responders need “who, what, when, where” ASAP for de-escalation efforts
- Delays in identification or notification can result in increased victim count/injury

Why Punch Detection?

- Build confidence with customer (security personnel) that our recognition model is fast and accurate for a single use case
  - Open doors for buy-in on models for other actions (e.g. firearm detection)
  - Build business case for gaining access to security team's footage to expand dataset
- POC deployment in locations where weapons have been screened
- Potential to act as evidence in assault situations

Violent Punch

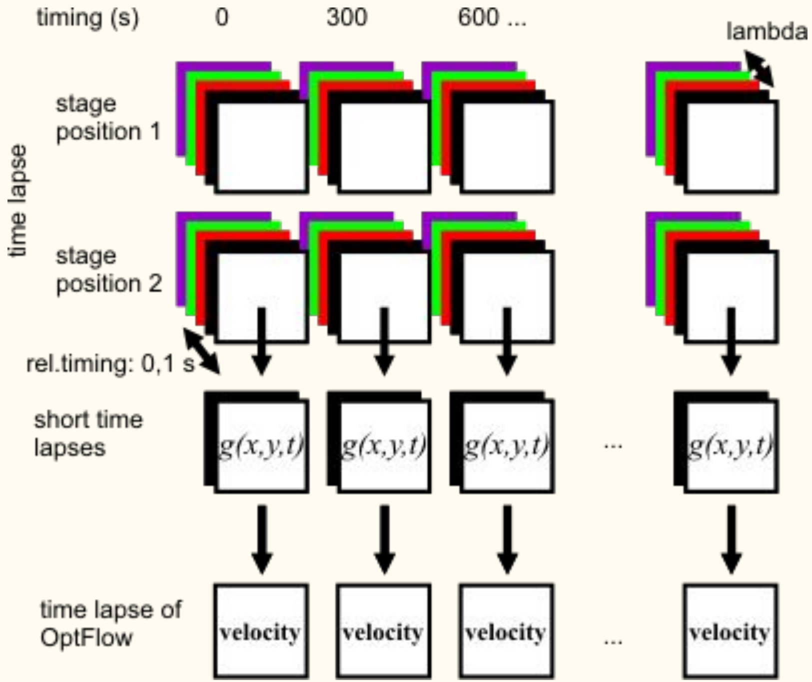
No Punch

Data Engineering

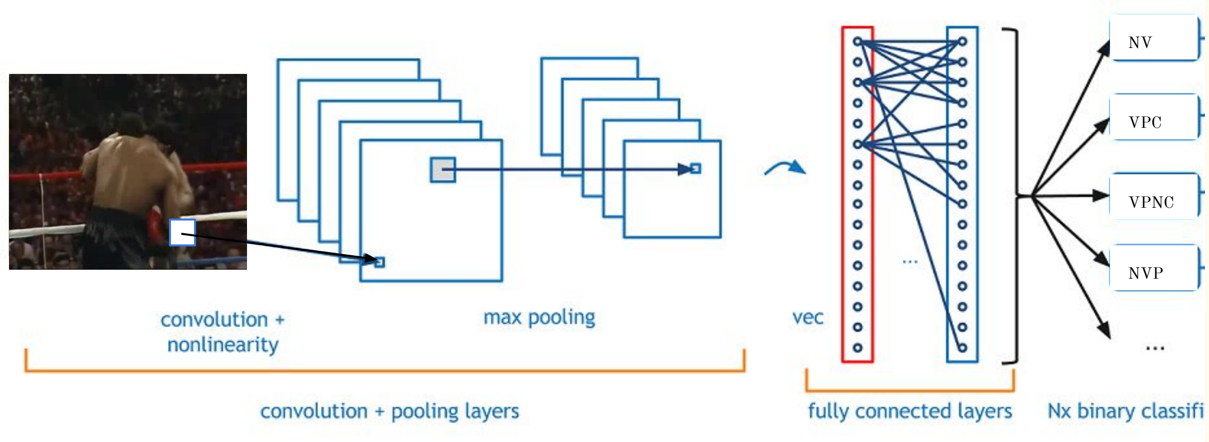
- Datasets for pretraining: UCF101 and Kinetics datasets
- Custom dataset for fine-tuning: YouTube videos, Violent Flows, and Kaggle
  - 5400 clips of street fights, crowd riots, dancing, sports events, training, etc.
- Slice videos into 5 sec (151 frames) maximum segments using ffmpeg
- Scale frames to 224x224 pixels prior to training
- Videos hand-labeled by the bittah-ninja team into 5 classes as well as an exclusion class
  - 0 - No punch, 1 - Violent punch, contact, 2 - Violent punch, no contact, 3 - Non-violent punch, contact, 4 - Non-violent punch, no contact (shadowboxing)
- Exclusion rules
  - Video does not contain human subjects
  - Poor video quality

Data Engineering

- Extract and store raw frames for spatial stream in TSN model
- Calculate x and y-axis optical flows on raw frames for temporal stream in TSN model
  - gradient of pixel intensity wrt time

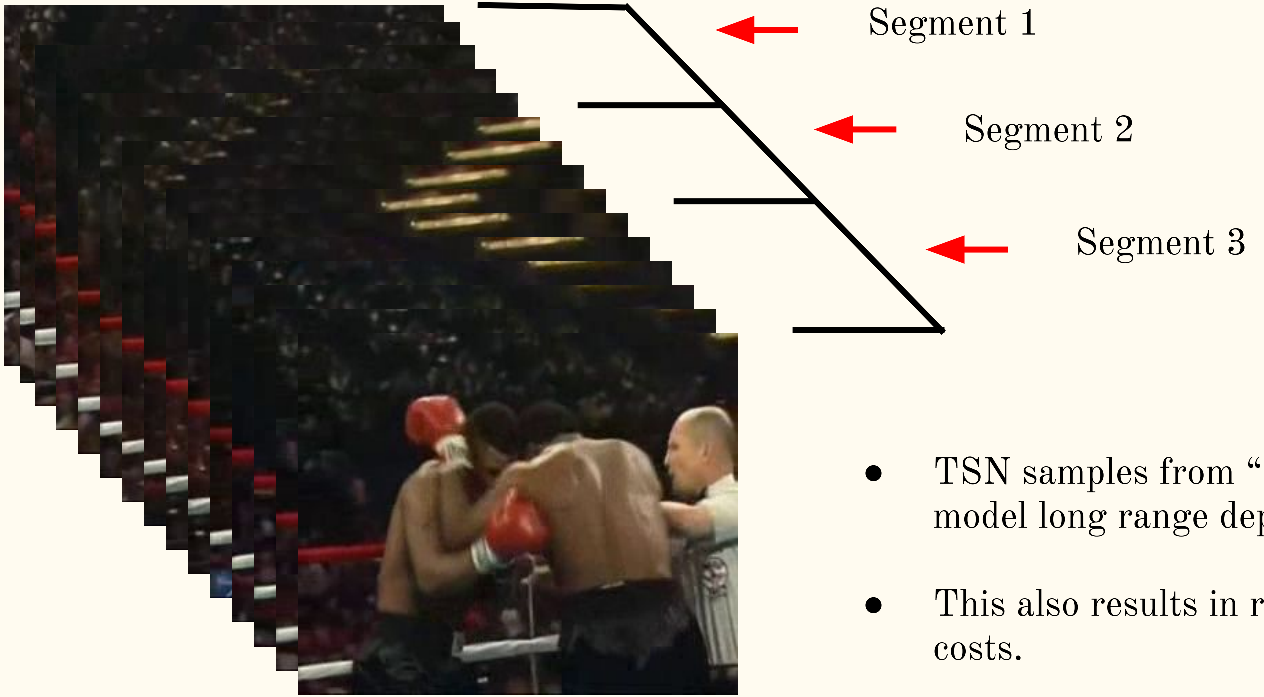


CNN For Naive Baseline



https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/

TSN Models Long Range Dependencies



- TSN samples from “segments”, allowing to model long range dependencies.
- This also results in reduced computation costs.

Input Data for Two Stream Action Recognition

RGB Frames - Spatial

Optical Flows - Temporal

Flows on x-axis

Flows on y-axis

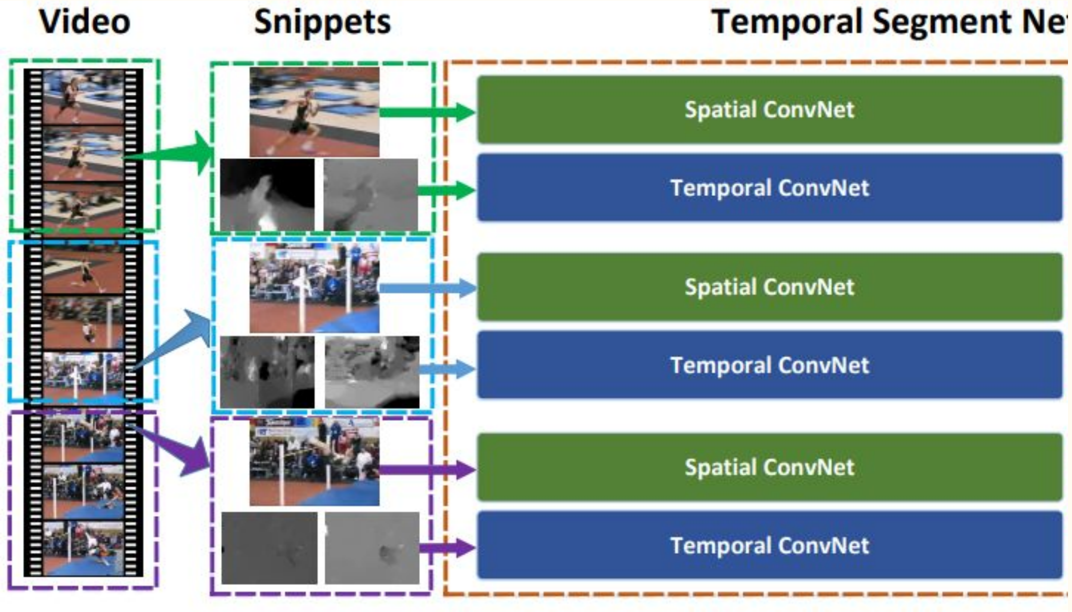
TSN Evenly Segments Videos

https://arxiv.org/pdf/1608.00859.pdf

Snippets Sampled From Segments

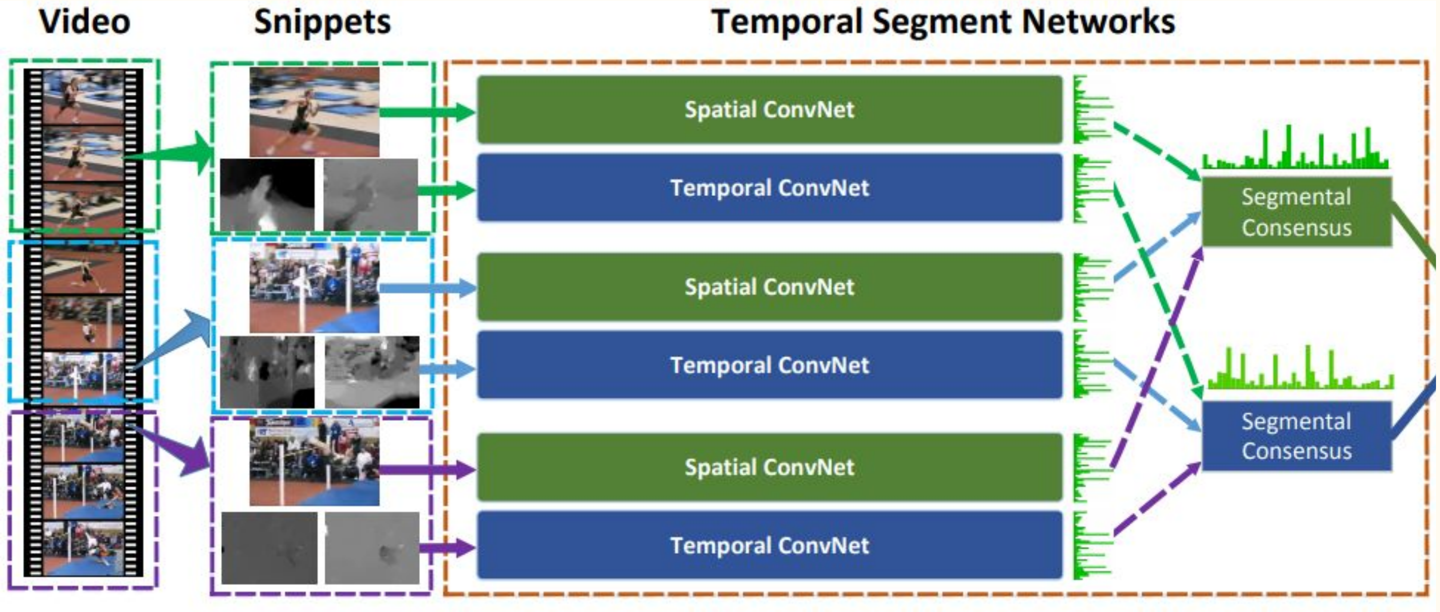
https://arxiv.org/pdf/1608.00859.pdf

Snippets Passed Through Deep Neural Nets



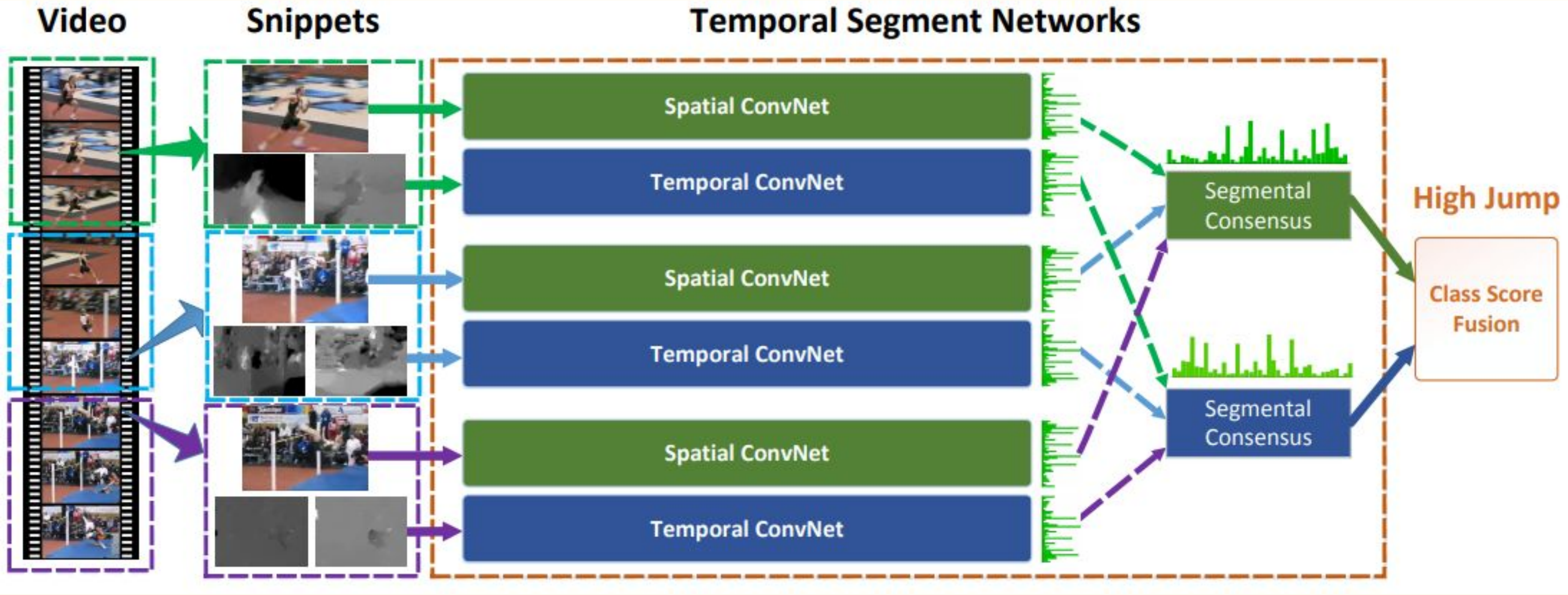
https://arxiv.org/pdf/1608.00859.pdf

Consensus Reached For Each Modality



https://arxiv.org/pdf/1608.00859.pdf

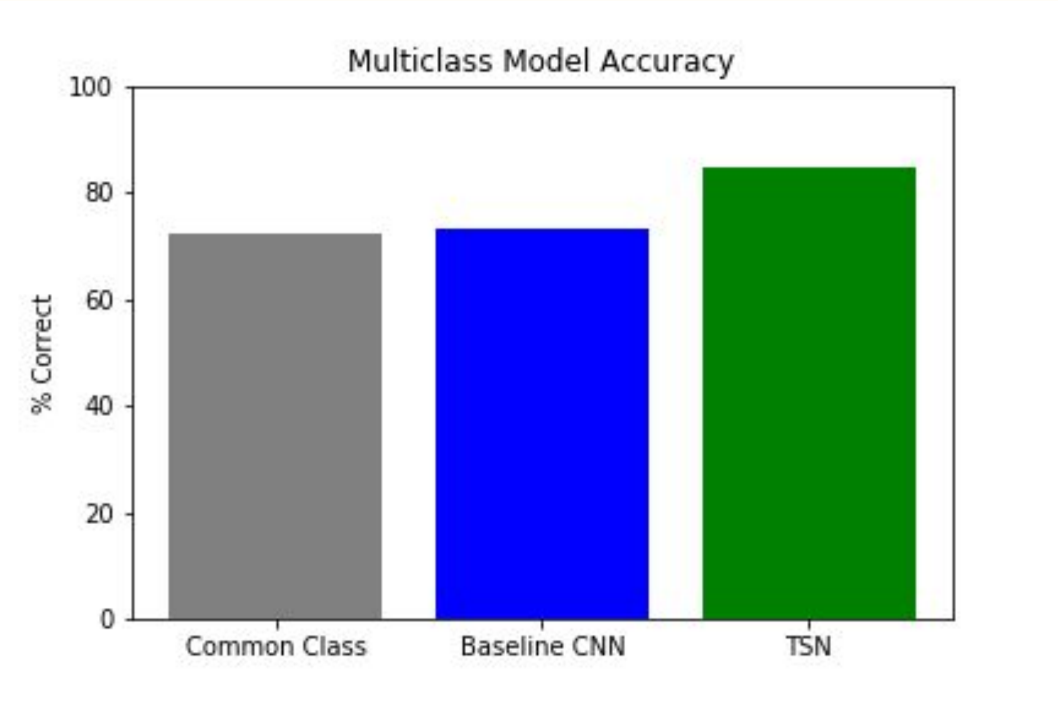
Final Class Predicted

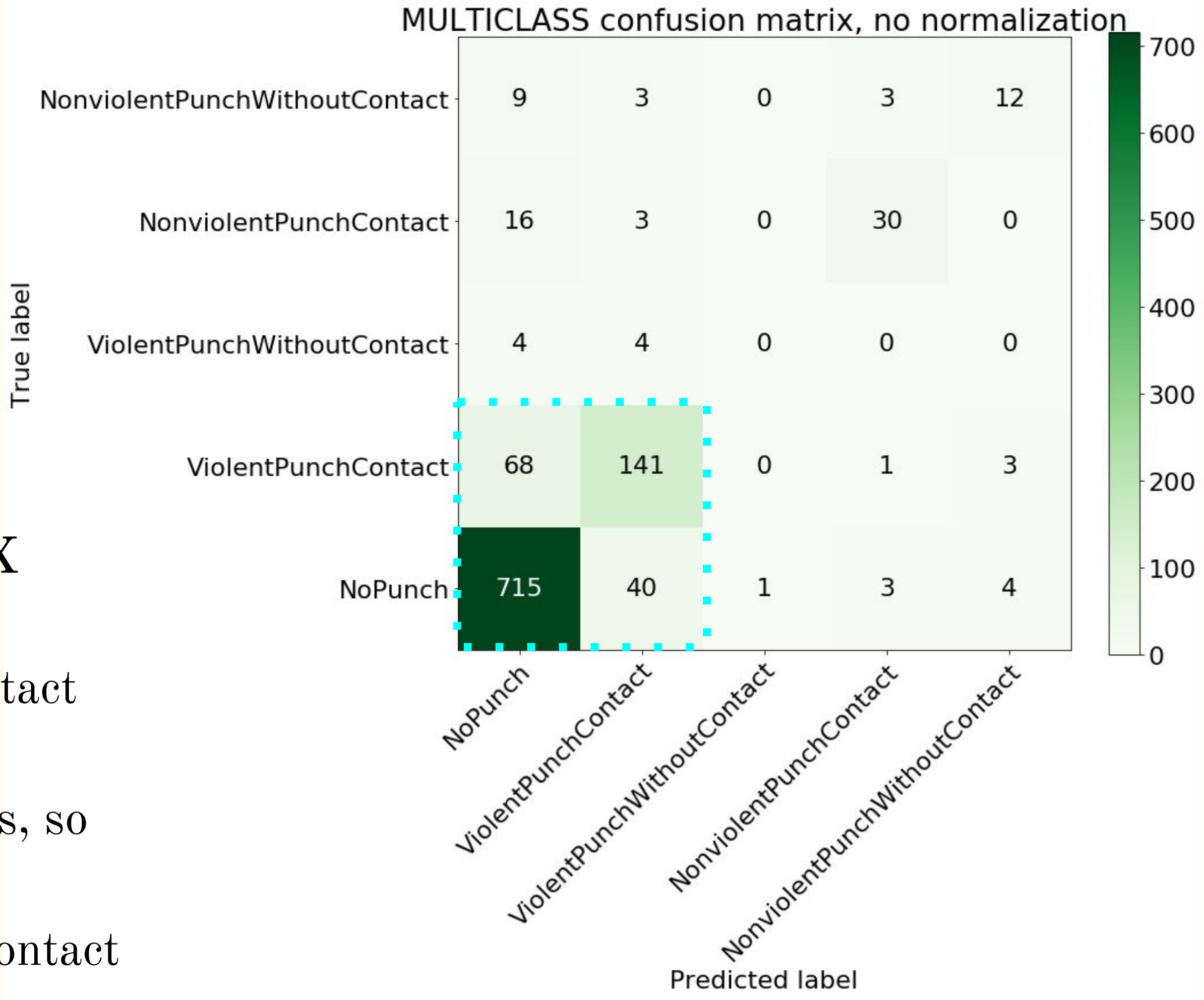


https://arxiv.org/pdf/1608.00859.pdf

Model Evaluation

- Baseline CNN accuracy 73%
  - (Applied to 1 single frame sampled at random per clip)
  - Barely outperforms guessing common (no-punch) class of 72%
- TSN accuracy 84.7%





Confusion Matrix

- 74% precision for violent-punch-with-contact predictions
- 66% recall for this class, so we're missing 1/3rd of violent-punches-with-contact

Correct:
True negative

Correct:
True positive

Wrong:
False positive

Wrong:
False negative

Mock-Up of the Punch Detection Web App

<demo prototype>

Key Challenges

- Being frugal
  - Video data is expensive to work with
- Data engineering
  - How to manipulate video data?
- Model development
  - Which model?
- Website development
  - Minimalist but expressive

Retro

- Crowd source labeling (e.g., Mechanical Turks)
- Additional features for data
    - Video quality
    - Audio
- Fixed video inputs (e.g., surveillance camera versus phone camera)
- Optimize video clip length and resolution

Next Steps

- Generalize classes and enrich dataset (kicks, shoves, etc.)
- Team up with a security company to get more data
- Scalable implementation - lightweight model for edge (TSM), heavy duty for decision support (TSN or others?)

Thank You

http://www.thumos.info/download.html

References

- https://github.com/open-mmlab/mmaction
- https://github.com/qijiezhao/py-denseflow
- Khurram Soomro, et al. "UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild." (2012), https://arxiv.org/abs/1212.0402
- Joao Carreira, et al. "Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset." (2017), https://arxiv.org/abs/1705.07750
- Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang: “Temporal Segment Networks: Towards Good Practices for Deep Action Recognition”, (2016), http://arxiv.org/abs/1608.00859
- Ji Lin, Chuang Gan: “TSM: Temporal Shift Module for Efficient Video Understanding”, (2018), http://arxiv.org/abs/1811.08383

BACKUP

Preliminary Results Using UCF-101 Promising

Subset of UCF-101 Classes:

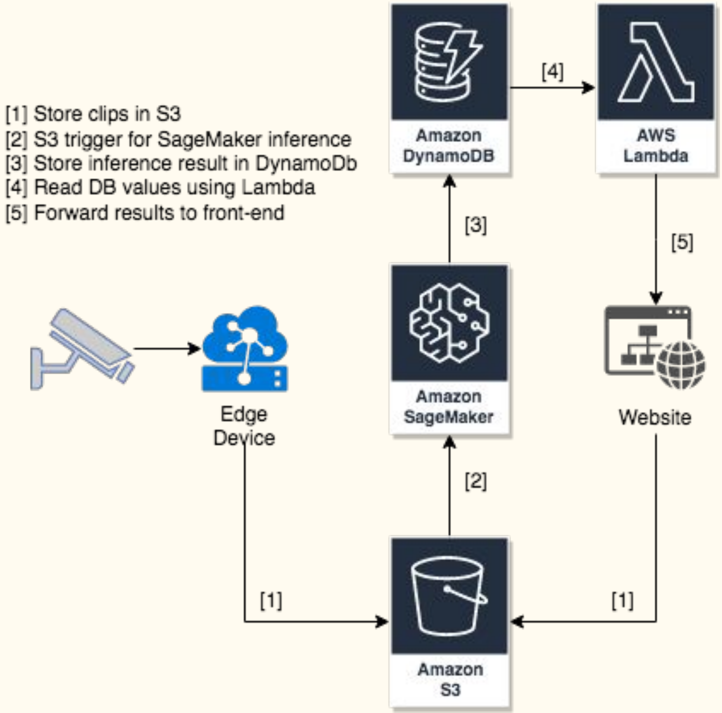
- Punch
- Baseball Pitch
- Bench Press
- Breaststroke
- Tai Chi
- etc.

Top 1 Accuracy = 94%
Top 5 Accuracy = 100%

Infrastructure/Deployment

Technologies

- AWS
- NodeJS
- React
- IBMCloud
- Jupyter Notebook
- Docker



Models Used for Action Recognition

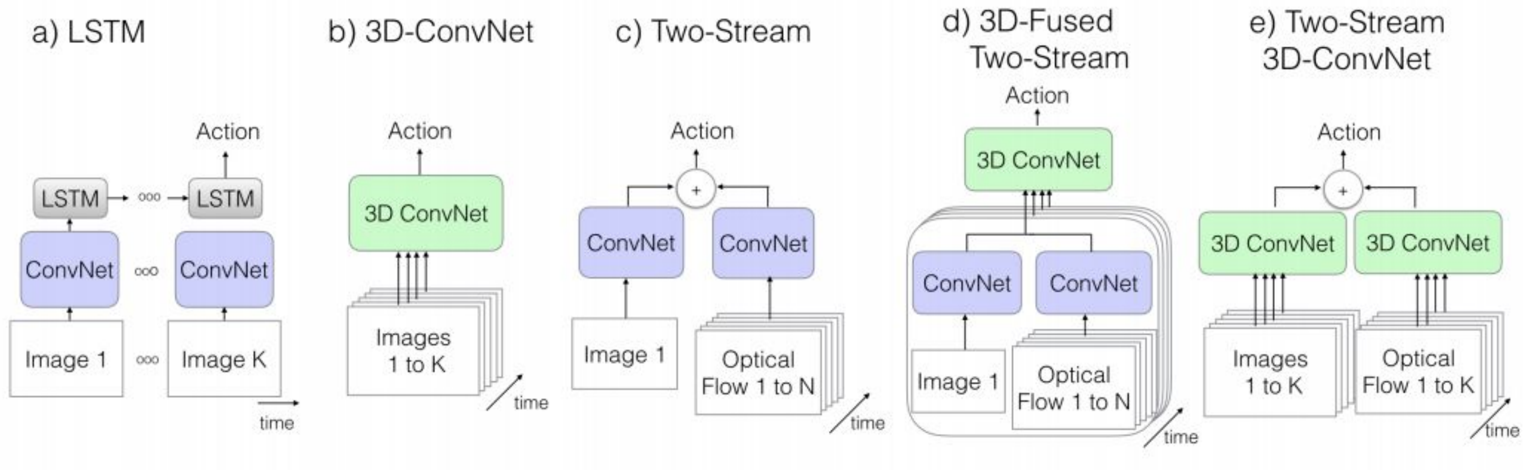
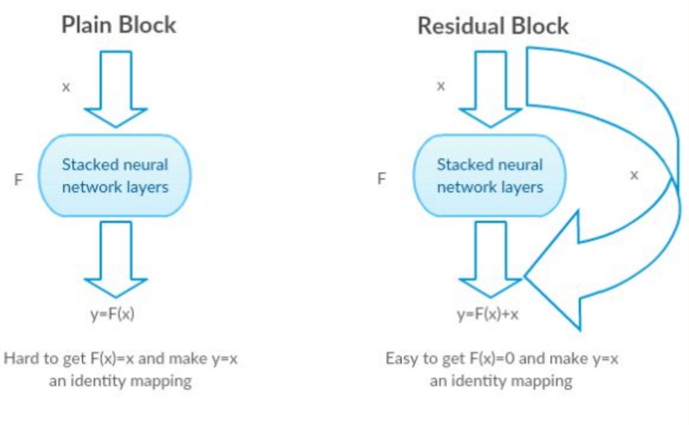


Figure 2. Video architectures considered in this paper. K stands for the total number of frames in a video, whereas N stands for a subset of neighboring frames of the video.

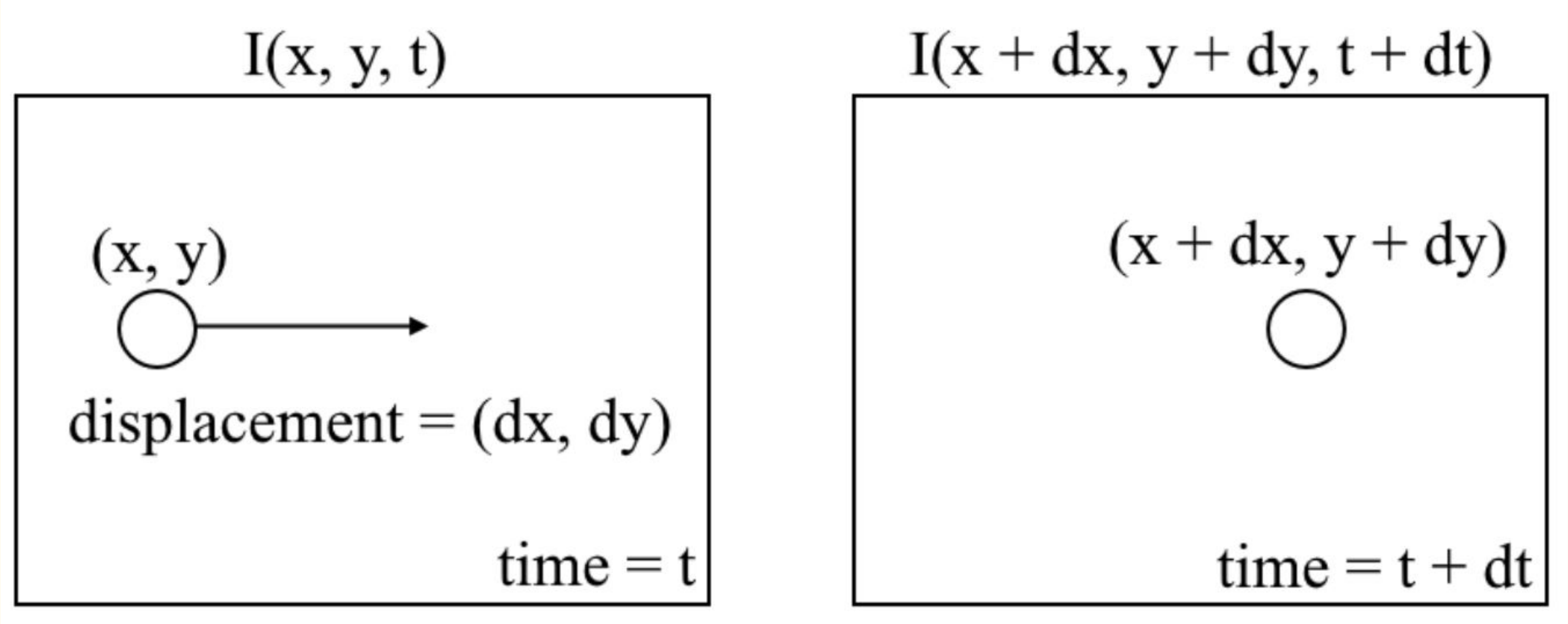
Model - Backbone

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		[3×3, 64 3×3, 64] ×2	[3×3, 64 3×3, 64] ×3	[1×1, 64 3×3, 64 1×1, 256] ×3	[1×1, 64 3×3, 64 1×1, 256] ×3	[1×1, 64 3×3, 64 1×1, 256] ×3
conv3_x	28×28	[3×3, 128 3×3, 128] ×2	[3×3, 128 3×3, 128] ×4	[1×1, 128 3×3, 128 1×1, 512] ×4	[1×1, 128 3×3, 128 1×1, 512] ×4	[1×1, 128 3×3, 128 1×1, 512] ×8
conv4_x	14×14	[3×3, 256 3×3, 256] ×2	[3×3, 256 3×3, 256] ×6	[1×1, 256 3×3, 256 1×1, 1024] ×6	[1×1, 256 3×3, 256 1×1, 1024] ×23	[1×1, 256 3×3, 256 1×1, 1024] ×36
conv5_x	7×7	[3×3, 512 3×3, 512] ×2	[3×3, 512 3×3, 512] ×3	[1×1, 512 3×3, 512 1×1, 2048] ×3	[1×1, 512 3×3, 512 1×1, 2048] ×3	[1×1, 512 3×3, 512 1×1, 2048] ×3
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9



Identity mapping in Residual blocks

Optical Flows - Calculation



Optical Flows - Calculations Continued..

Taylor Series Approximation

$$I(x+\delta x, y+\delta y, t+\delta t) = I(x,y,t) + \frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t + \ldots$$

$$\Rightarrow \frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t = 0$$

Dividing by Change in Time Gives Two Components (movement in the x and y direction)

$$\frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \frac{\partial I}{\partial t} = 0$$