Recent Works on Computer Vision Applications

Prof. Chien-Chang Chen
Tamkang University
Taiwan

about me

Chien-Chang Chen, TKU

* Professor, Department of Computer Science and Information Engineering, Tamkang University, Taiwan
* 2016.08 - 2020.07: Chair of Department of Computer Science and Information Engineering, Tamkang University, Taiwan
* TEEP students:
  * 2023 spring(3)
  * 2022 autumn(6)
  * 2022 spring(2)
  * 2018 autumn(2)

about me

Chien-Chang Chen, TKU

* Research interests
  * Secret image sharing
  * Artificial intelligence
  * Texture analysis
* Recent interests in collaborations across fields
  * Sports(Pose applications)
  * air calligraphy
  * fabric defect detection(Anomaly Detection)

Secret Image Sharing

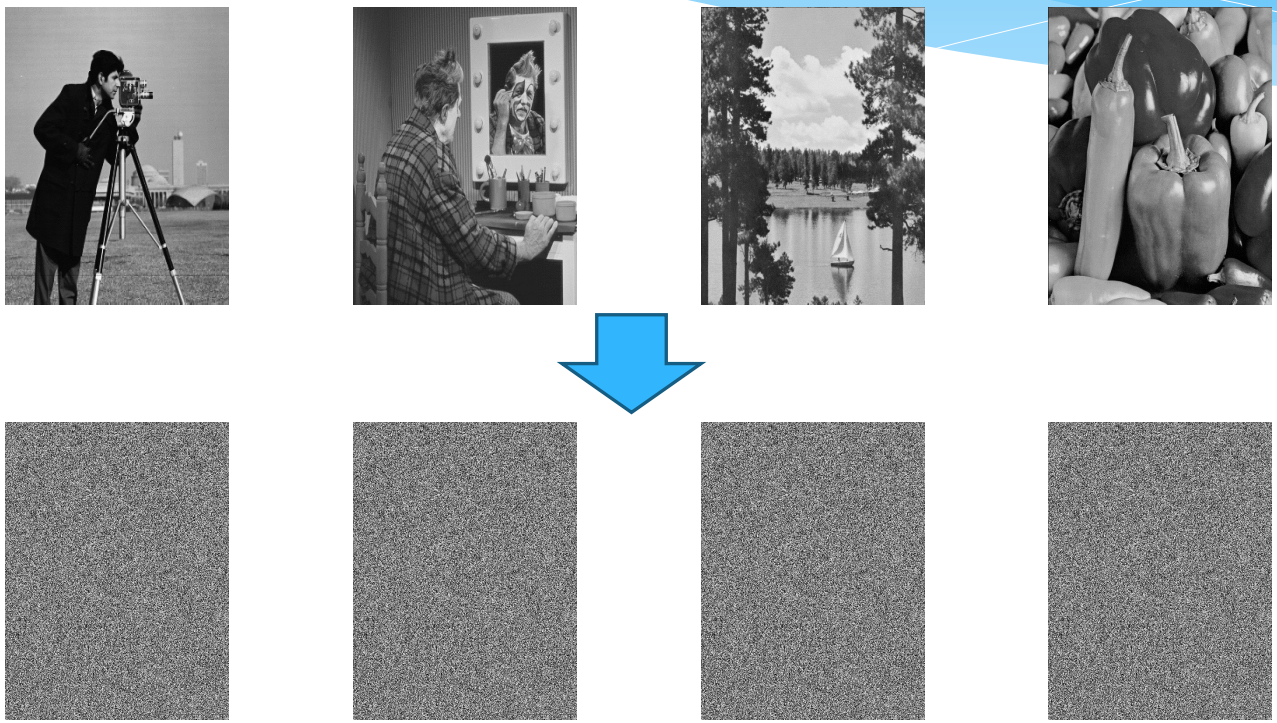
shared shadows

secret Image

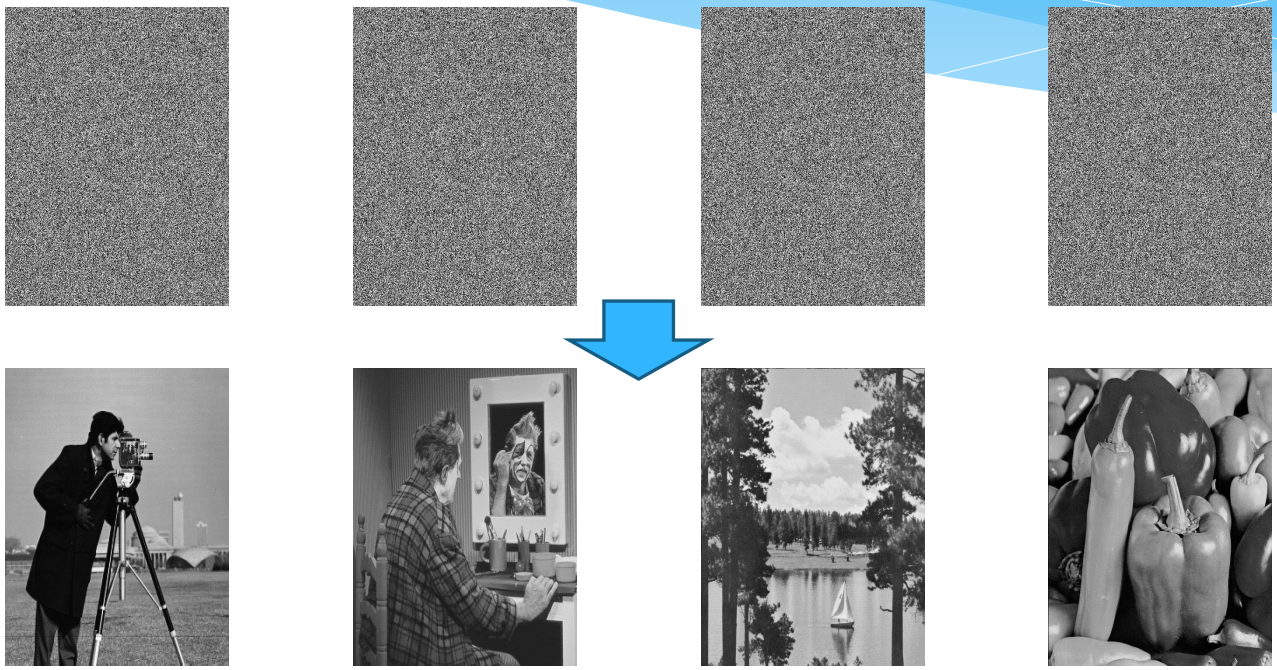
recovered Image

(t, n) threshold
$t < n$

Multi-Secret Image Sharing



Multi-Secret Image Sharing



Publications

* Chien-Chang Chen and Yu-Wei Chien, "Sharing Numerous Images Secretly with Reduced Possessing Load", FUNDAMENTA INFORMATICAE, vol. 86, no. 4, 447-458, 2008. (SCI)
* Chien-Chang Chen and W. Y. Fu, "A Geometry-Based Secret Image Sharing Approach", JOURNAL OF INFORMATION SCIENCE AND ENGINEERING, vol. 24, no. 5, 1567-1577, 2008. (SCI)
* Chien-Chang Chen, Chaur-Chin Chen and Yun-Cheng Lin, "Weighted Modulated Secret Image Sharing Method", Journal of Electronic Imaging, vol.18, 043011, 1-6, 2009. (SCI IF=0.563)
* Chien-Chang Chen and Chong-An Liu, "A Tamper-Proof Secret Image Sharing Scheme for Identifying Cheated Secret Keys and Shared Images", Journal of Electronic Imaging, vol.22, no.1, 013008, 2013. (SCI IF=1.061)
* Chien-Chang Chen and Wei-Jie Wu, "A Secure Boolean-based Multi-Secret Image Sharing Scheme," The Journal of Systems and Software, vol.92, no.6, 107-114, 2014. (SCI IF=1.135)
* Chien-Chang Chen, Shih-Chang Chen, "Two-Layered Structure for Optimally Essential Secret Image Sharing Scheme," Journal of Visual Communication and Image Representation, vol.38, 595-601, 2016. (MOST 103-2221-E-032 -051)(2014 SCI IF=1.218 Computer Science, Information Systems 57/139)(2014 SCI IF=1.218 Computer Science, Software Engineering 39/104)(2015 SCI IF=1.530 Computer Science, Information Systems 49/143)

Publications

* Chien-Chang Chen, Wei-Jie Wu and Jun-Long Chen," Highly Efficient and Secure Multi-Secret Image Sharing Scheme," Multimedia Tools and Applications, vol.75, 7113-7128, June, 2016. (2014 SCI IF=1.346 Computer Science, Information Systems 52/139)(2015 SCI IF=1.331 Computer Science, Information Systems 62/143)
* Chien-Chang Chen* and Jun-Long Chen, "A New Boolean-based Multiple Secret Image Sharing Scheme to Share Different Sized Secret Images," Journal of Information Security and Applications, vol.33, 45-54, April, 2017. (2018 5-years SCI IF=not available Computer Science, Information Systems)
* Chien-Chang Chen* "Essential Secret Image Sharing Scheme with Equal-sized Shadows Generation," Journal of Visual Communication and Image Representation, vol.52, 143-150, April, 2018. (MOST 105-2221-E-032-053) (2017 5-year SCI IF=2.028 Computer Science, Information Systems 65/148) (2016 SCI IF=2.028 Computer Science, Software Engineering 34/104)
* Cheng-Shian Lin, Chien-Chang Chen*, Yu-Cheng Chen, "XOR-Based Progressively Secret Image Sharing," Mathematics, vol. 9, no. 6, 612, 2021. (2020 5-years SCI IF=2.165 Mathematics 32/330)
* Chien-Chang Chen, Cheng-Shian Lin, and Jia-Zhan Chen, "Boolean-Based (k, n, m) Multi-Secret Image Sharing," Axioms, vol. 11, no. 5, pp. 1-21, 2022. (2021 SCI IF=1.824 Mathematics 98/267)

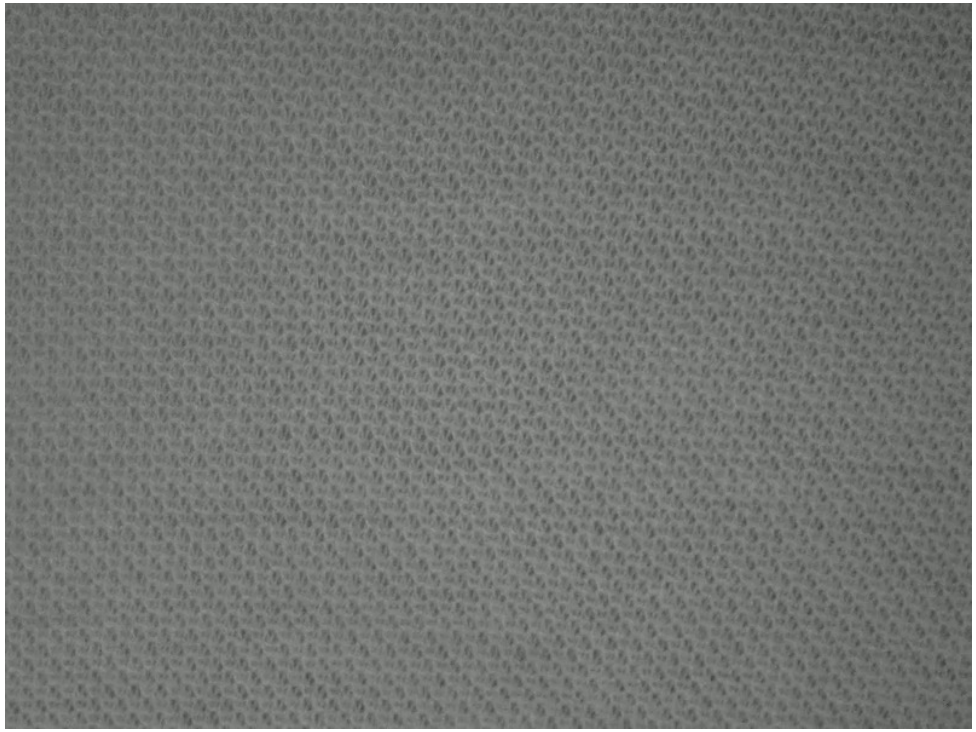
Contents

* Anomaly Fabric Defect Detection
  * Autoencoder
  * One-shot Siamese Network
* Sports(Pose applications)
  * Basketball Shooting Prediction
  * Rowing Match
  * Automatic rowing cycle detection
* Air Writing

Anomaly Fabric Defect Detection

The general way

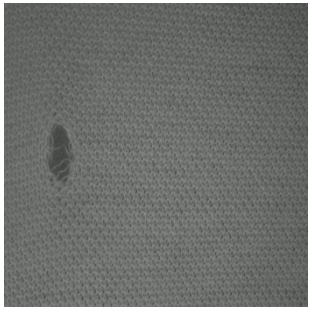
Fabric Video (from Pailung)



Fabric Defect Detection

* Extracted images



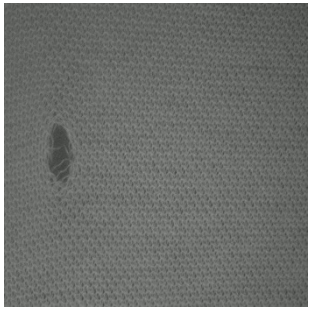



* Two kinds of research problems
  * Determine an image with defect region or not (texture analysis)
  * Detect the defect region (texture segmentation)

Texture Analysis(classify)

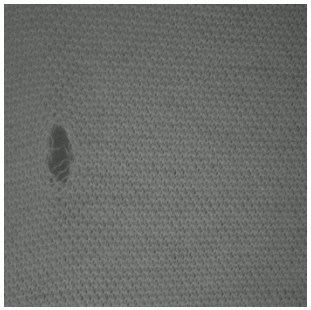
* Determine an image with defect region or not



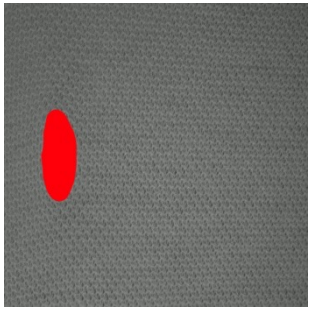


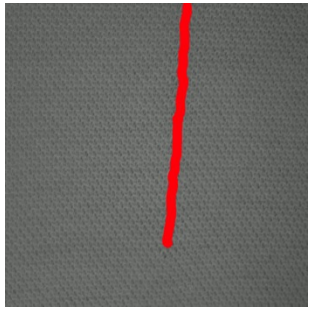

Texture Segmentation

* Detect the defect regions



* Ground truth



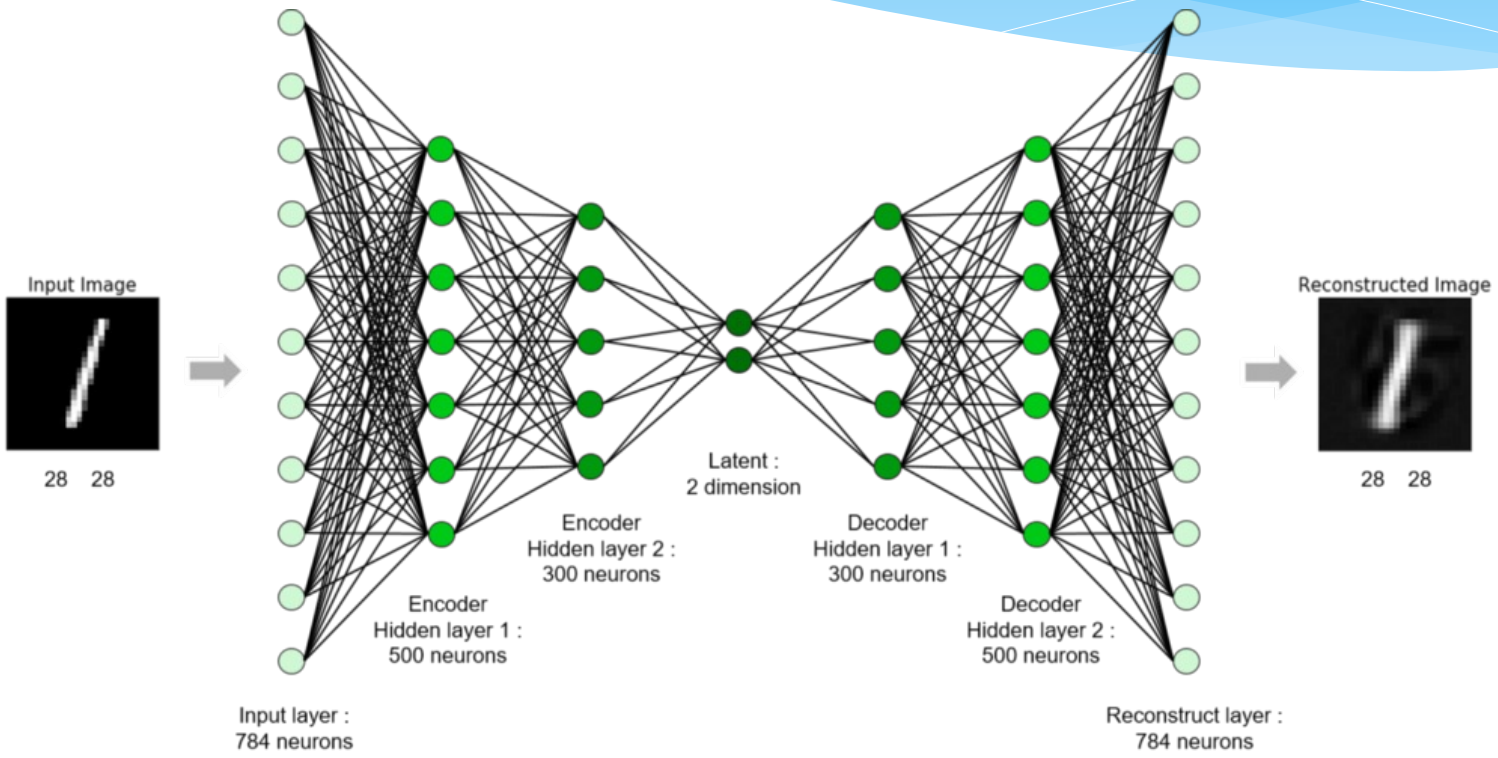


Proposed methods

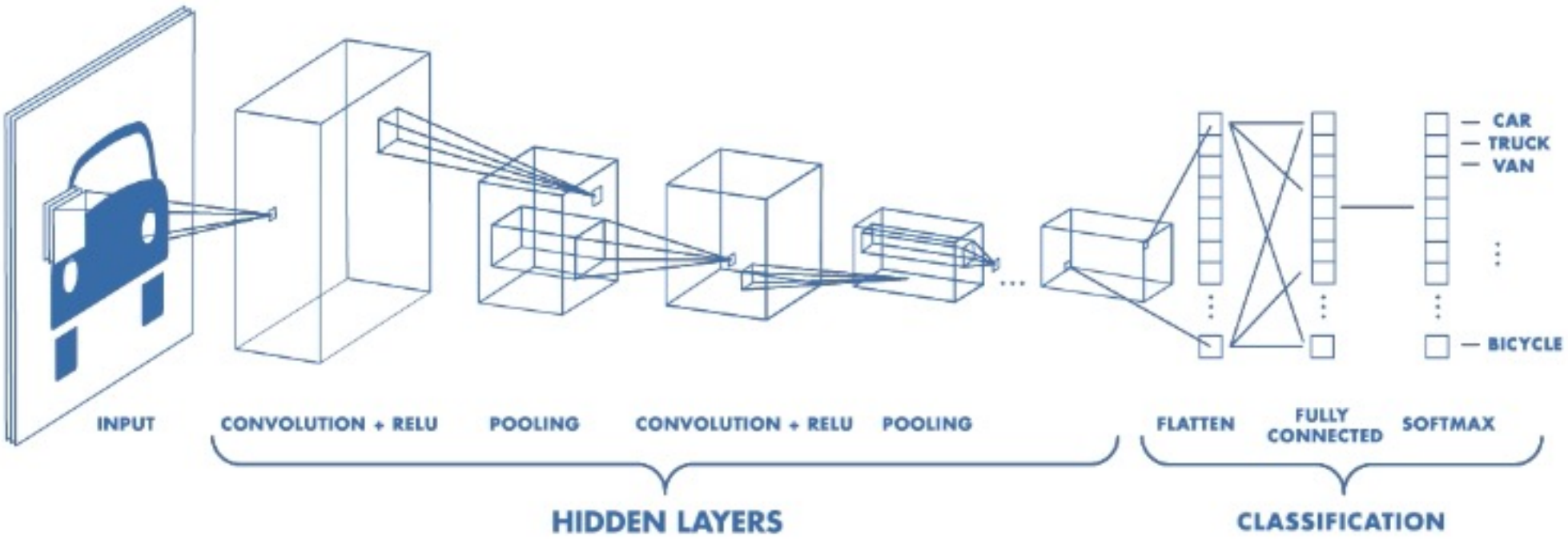
* Autoencoder model
* One-shot Siamese Network

Autoencoder model

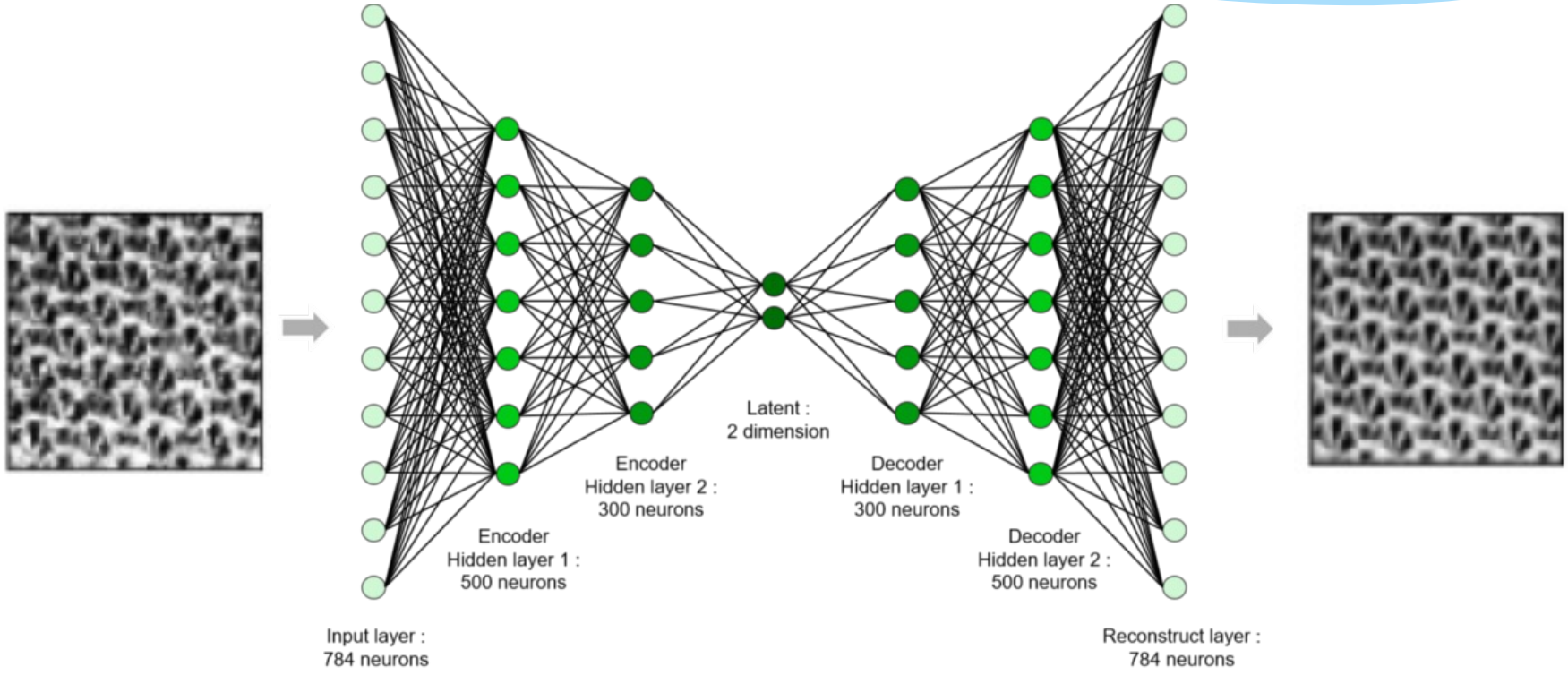
Autoencoder



Before autoencoder
CNN



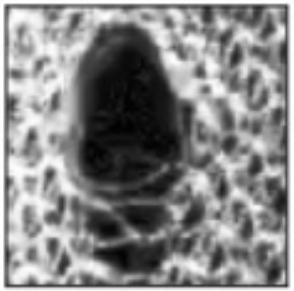
Autoencoder



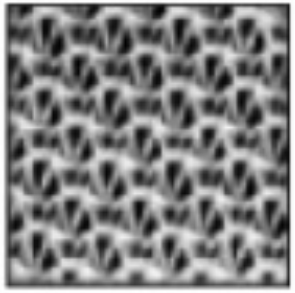
Autoencoder

normal image

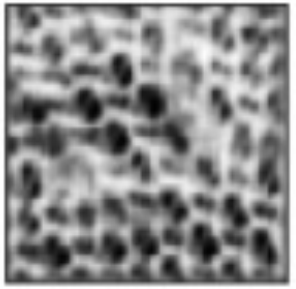
abnormal image



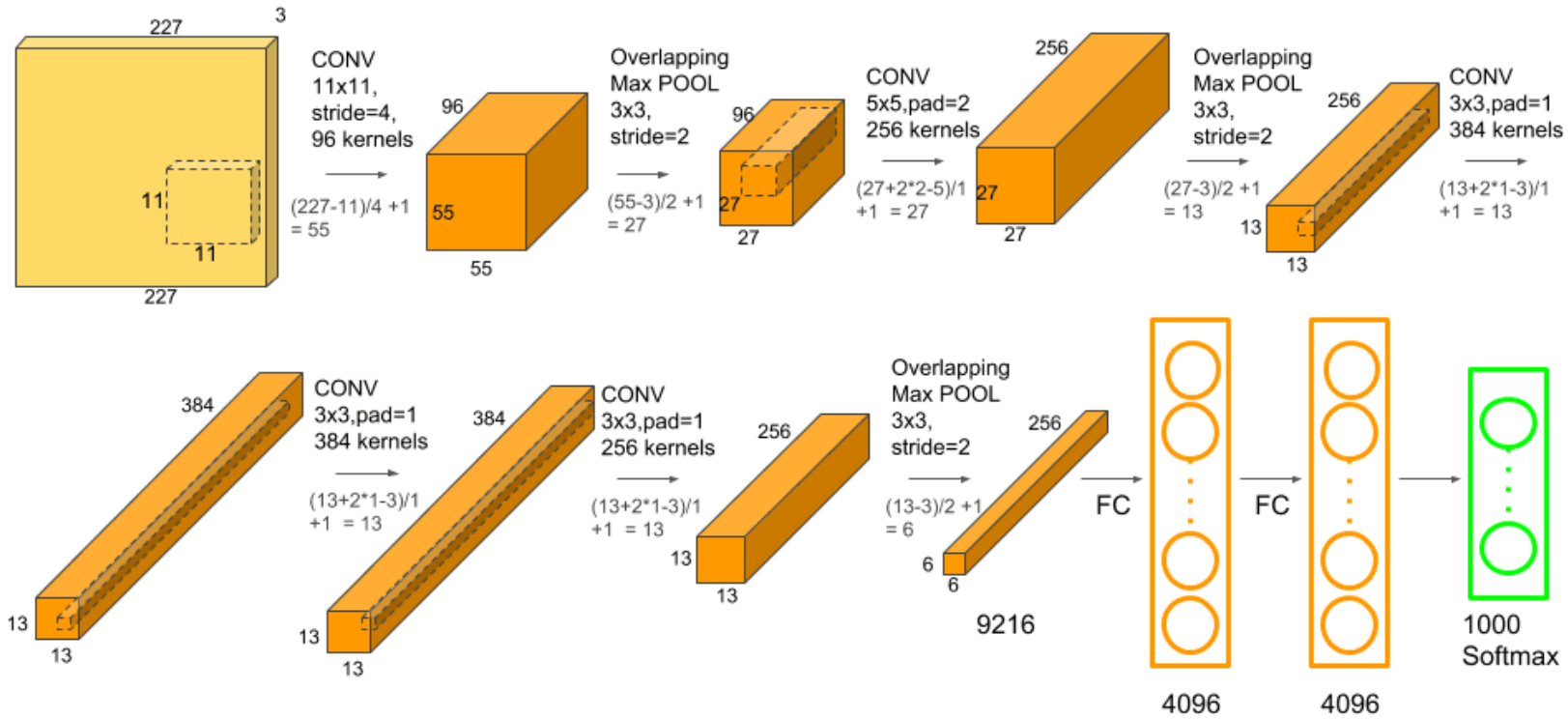
recovered image



recovered image



Alexnet

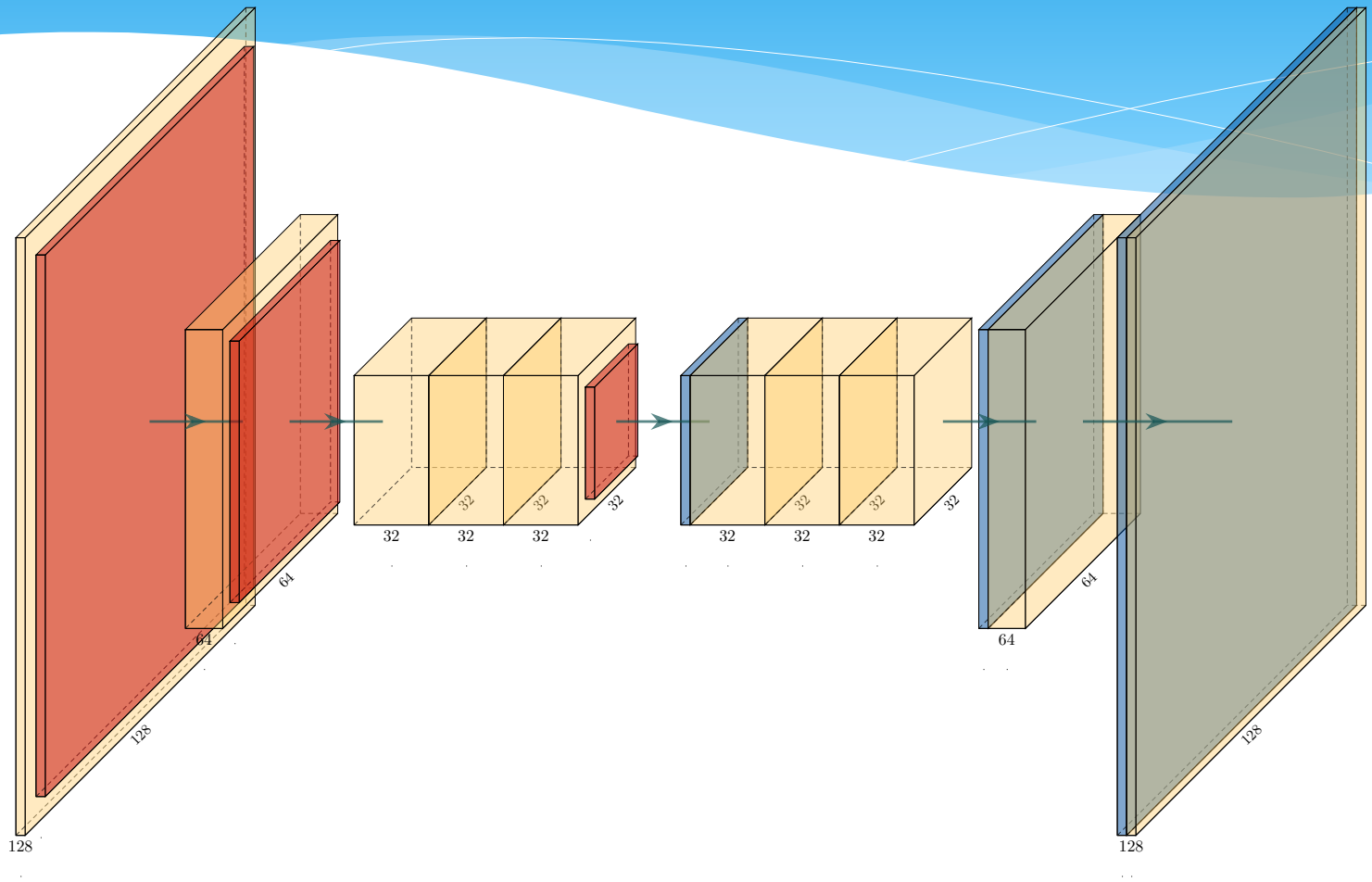


https://neurohive.io/en/popular-networks/alexnet-imagenet-classification-with-deep-convolutional-neural-networks/

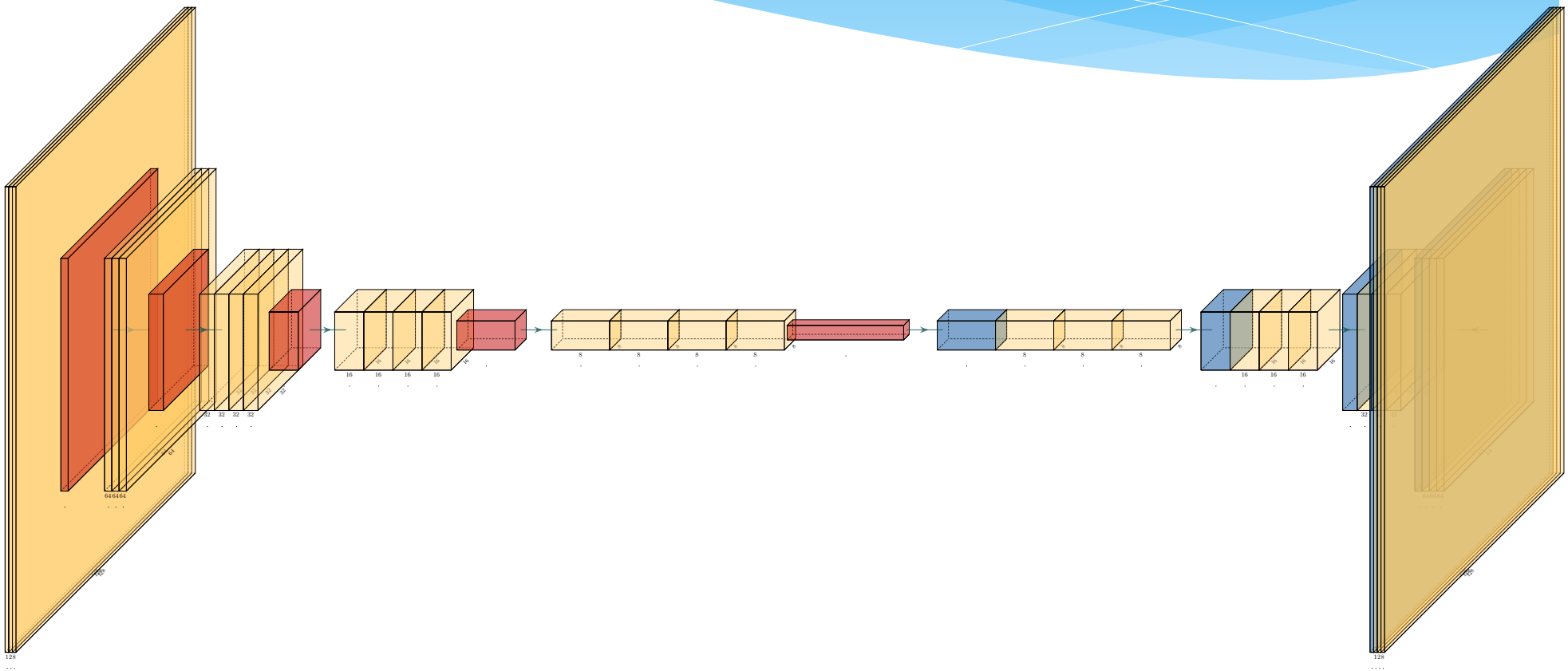
VGG16

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Alexnet-based autoencoder



VGG16-based autoencoder



Training Data

* Use lots of normal fabric images (800×600) to train these two models
  * first 30 seconds are examined normal images(totally 30×60=1800 images) ใช้วิดิโอที่ 60 frame/sec
  * resize 800×600 to 256×256 Care only output it's must be square
  * each 256×256 is overlappedly segmented to 128×128 images
  * totally 20,000 128×128 defect-free fabric images

Training

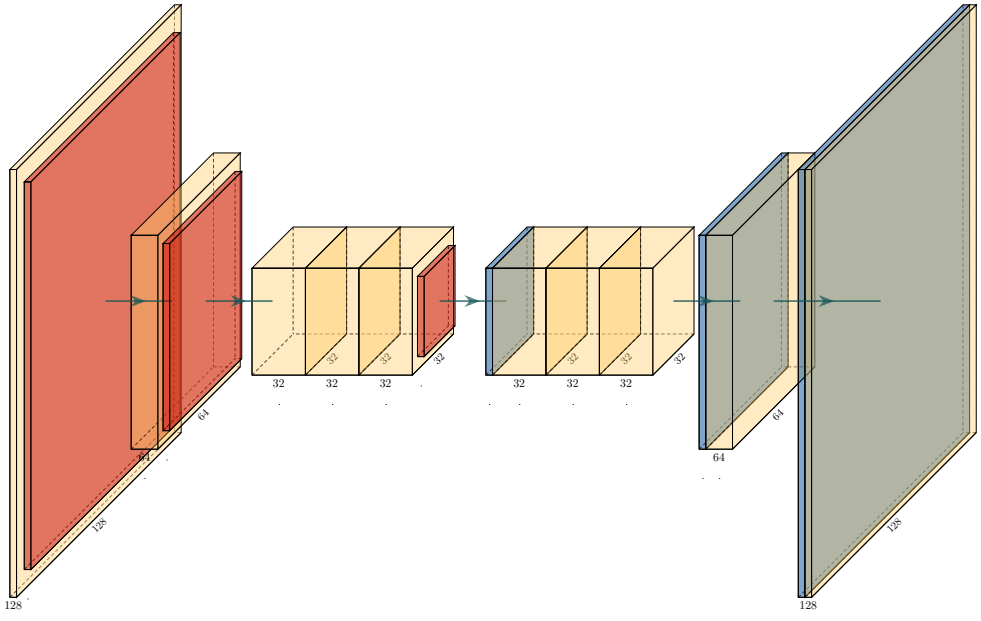
D => Decoder

E => Encoder

$$\hat{\boldsymbol{x}} = \boldsymbol{D}(\boldsymbol{E}(\boldsymbol{x}))$$

$$L_2(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \sum_{i=0}^{c-1} \sum_{j=0}^{h-1} \sum_{k=0}^{w-1} (\boldsymbol{x}(\boldsymbol{i}, \boldsymbol{j}, \boldsymbol{k}) - \hat{\boldsymbol{x}}(i, j, k))^2$$

* Minimize $L_2(\boldsymbol{x}, \hat{\boldsymbol{x}})$ by training

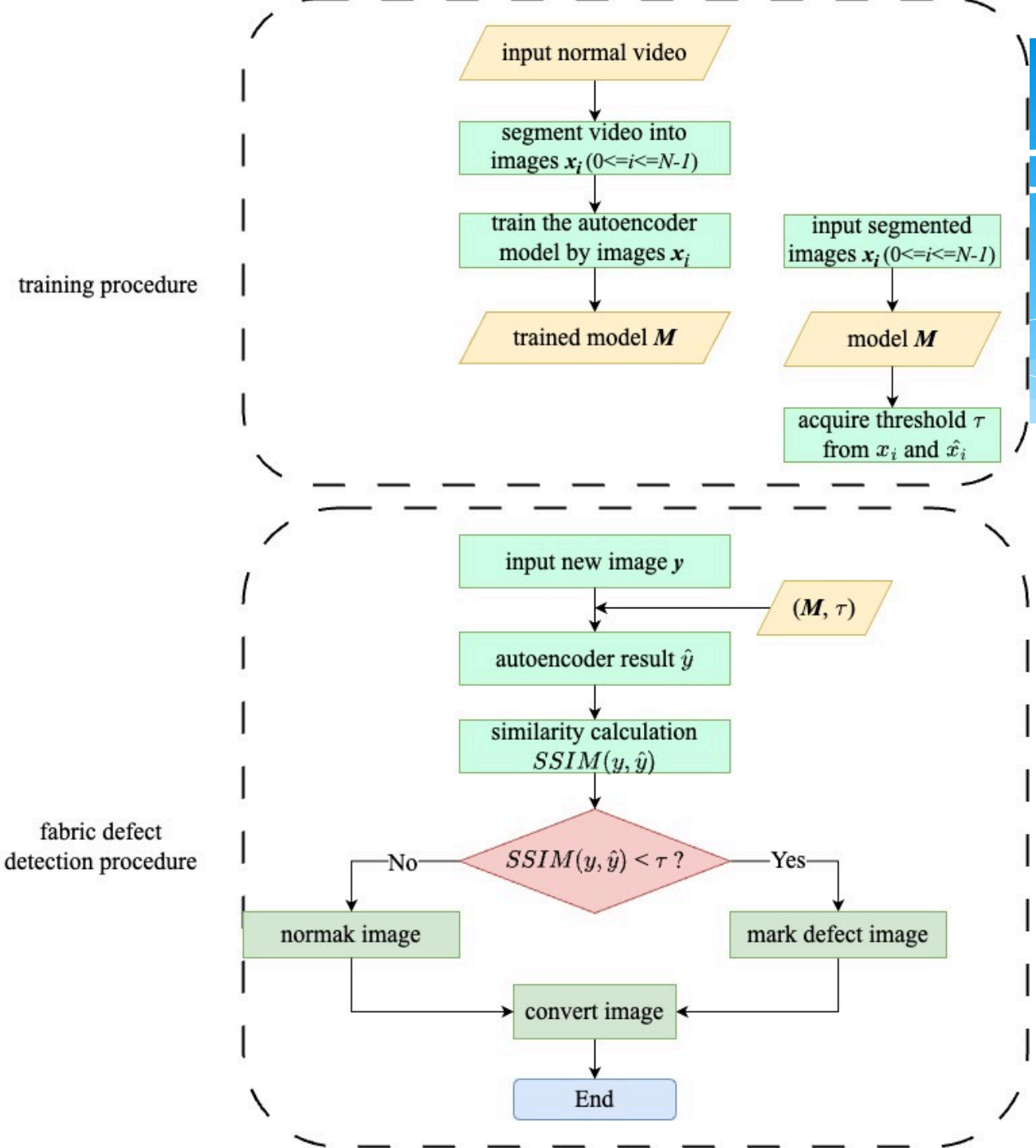


Model training

* Model training parameters
  * 200 epochs with the ADAM optimizer
  * learning rate of 2×10^{-4}
  * batch size of 64
  * activation function of LeakyRelu
  * similarity window size of 11
* The hardware is CPU-intel CORE i7-8700 with GPU-GeForce RTX 2080

fab

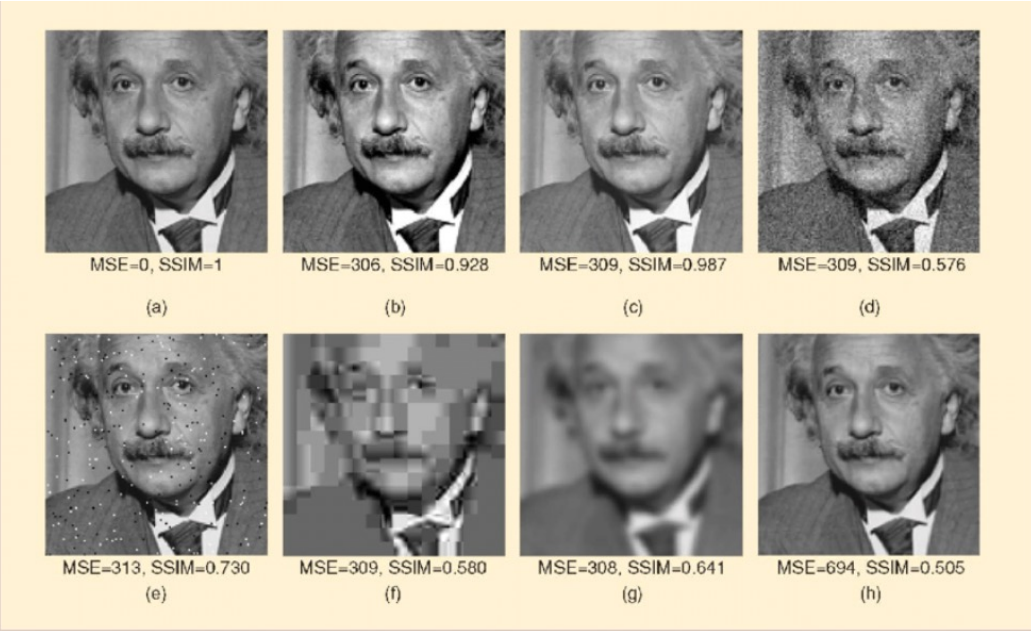
thm


training procedure
input normal video
segment video into images x_i (0<=i<=N-1)
train the autoencoder model by images x_i
trained model M
input segmented images x_i (0<=i<=N-1)
model M
acquire threshold τ from x_i and $\hat{x}_i$
fabric defect detection procedure
input new image y
(M, τ)
autoencoder result $\hat{y}$
similarity calculation $SSIM(y, \hat{y})$
$SSIM(y, \hat{y}) < \tau$?
No
Yes
normak image
mark defect image
convert image
End

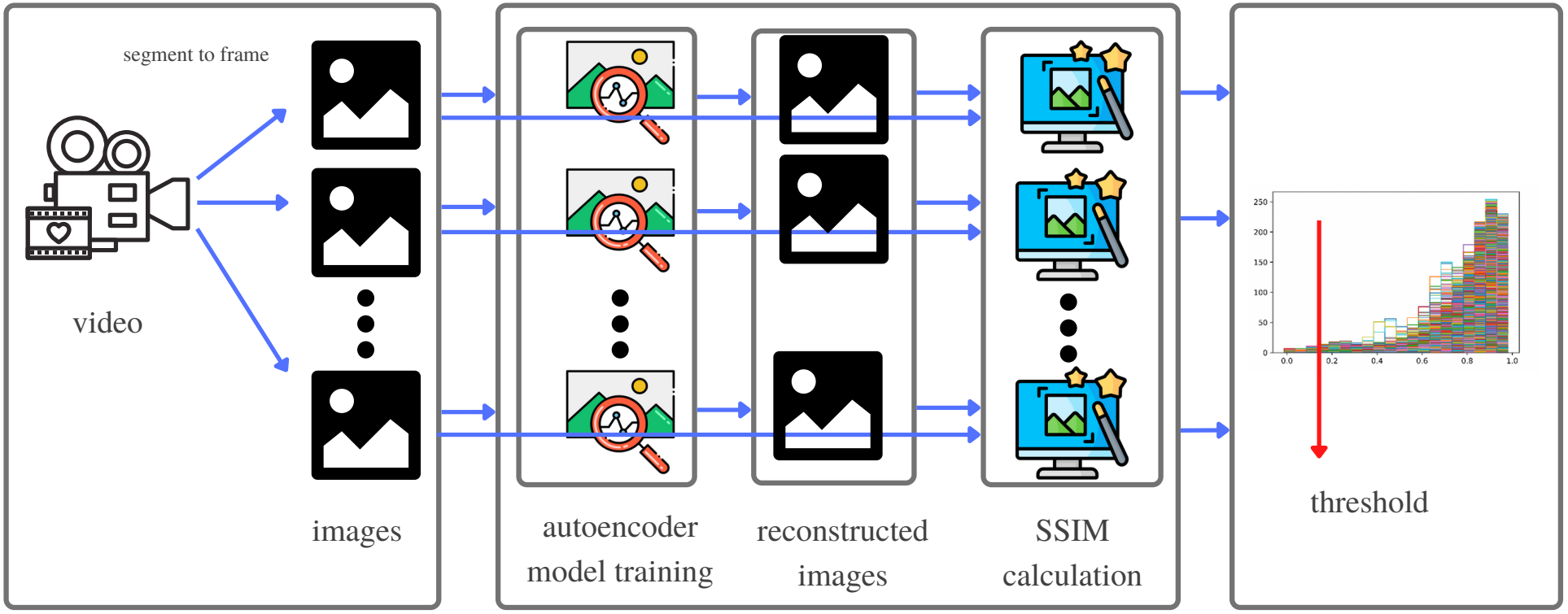
SSIM similarity calculation

* SSIM(Structural Similarity Index Measure)
  * SSIM value within the range of [0, 1]
  * 1 indicates the highest similarity
  * 0 indicates the lowest similarity

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_x\sigma_y + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$



Threshold determination



Bergmann's autoencoder

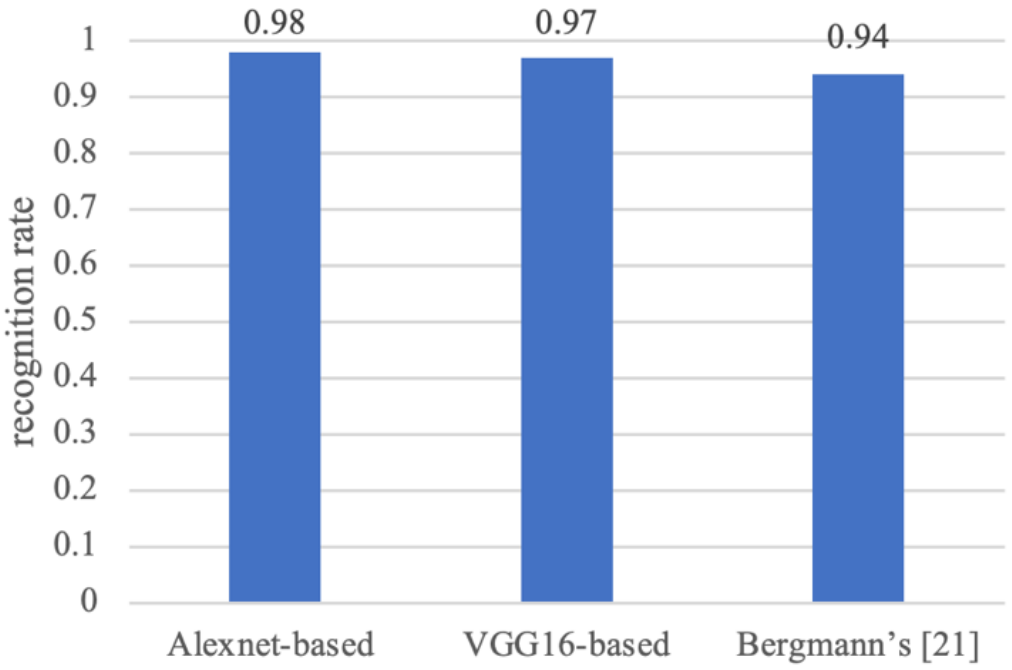
Layer	Output Size	Parameters		
		Kernel	Stride	Padding
Input	128x128x1			
Conv1	64x64x32	4x4	2	1
Conv2	32x32x32	4x4	2	1
Conv3	32x32x32	3x3	1	1
Conv4	16x16x64	4x4	2	1
Conv5	16x16x64	3x3	1	1
Conv6	8x8x128	4x4	2	1
Conv7	8x8x64	3x3	1	1
Conv8	8x8x32	3x3	1	1
Conv9	1x1xd	8x8	1	0

Experimental results

Methods	Training time (sec.)	Detection time (sec.)
Proposed Alexnet-based autoencoder	1,952	0.14
Proposed VGG16-based autoencoder	10,381	0.24
Bergmann's autoencoder	1,973	0.16

Experimental results

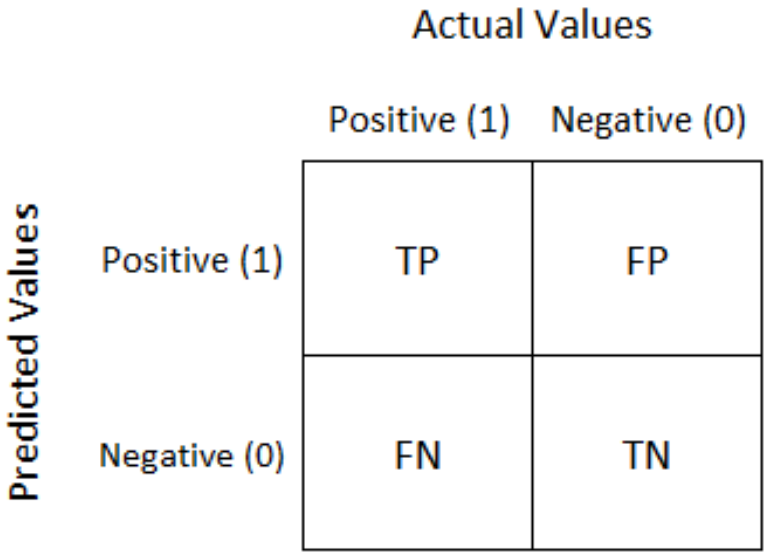
* recognition dataset : 100 fabric images
  * 88 normal + 12 defect images



Confusion matrix

* condition positive (P)
  * the number of real positive cases in the data
* condition negative (N)
  * the number of real negative cases in the data
* true positive (TP)
  * A test result that correctly indicates the presence of a condition or characteristic
* true negative (TN)
  * A test result that correctly indicates the absence of a condition or characteristic
* false positive (FP)
  * Type I error
  * A test result which wrongly indicates that a particular condition or attribute is present
* false negative (FN)
  * Type II error
  * A test result which wrongly indicates that a particular condition or attribute is absent

Confusion Matrix



True Positive:

predicted positive and it's true

You predicted a woman is pregnant and she actually is

True Negative:

predicted negative and it's true

You predicted a man is not pregnant and he actually is not

False Positive: (Type 1 Error)

predicted positive and it's false

You predicted a man is pregnant but he actually is not

False Negative: (Type 2 Error)

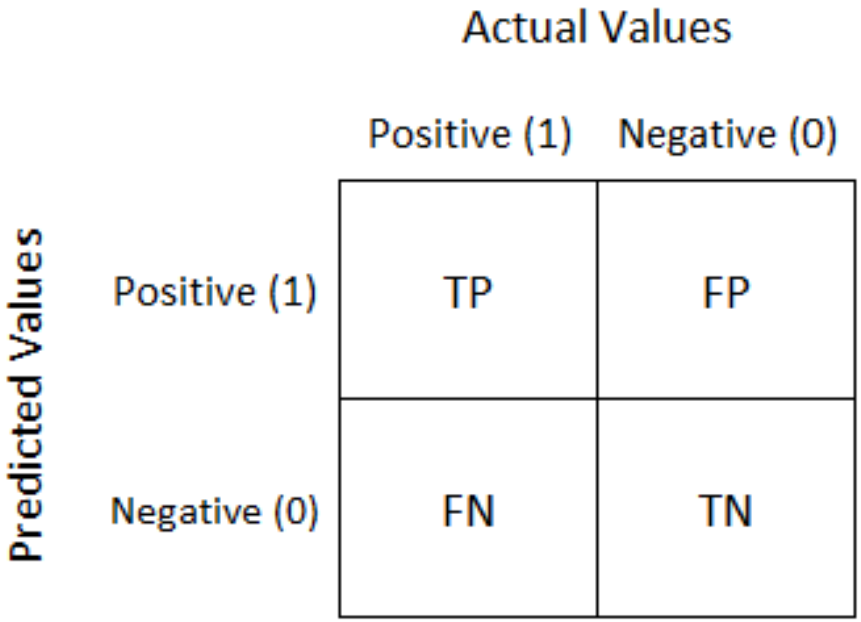
predicted negative and it's false

You predicted a woman is not pregnant but she actually is

Confusion Matrix

Accuracy= $\frac{TP+TN}{TP+TN+FP+FN}$

From all the classes (positive and negative), how many of them we have predicted correctly
Accuracy should be high as possible



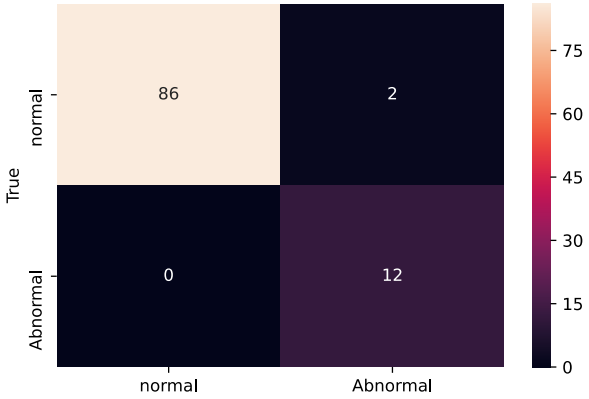
Precision= $\frac{TP}{TP+FP}$

predicted as positive, how many are actually positive
Precision should be high as possible

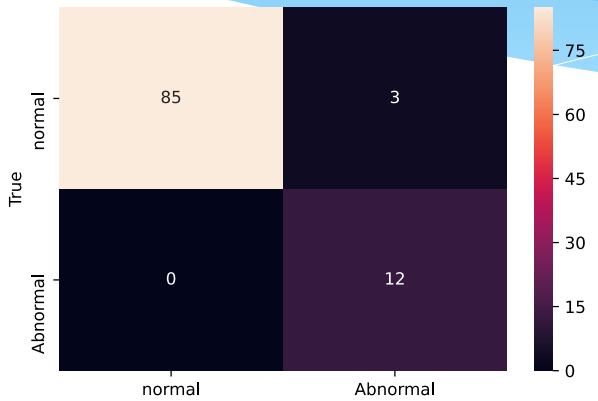
Recall = $\frac{TP}{TP+FN}$

from all the positive classes, how many we predicted correctly
Recall should be high as possible

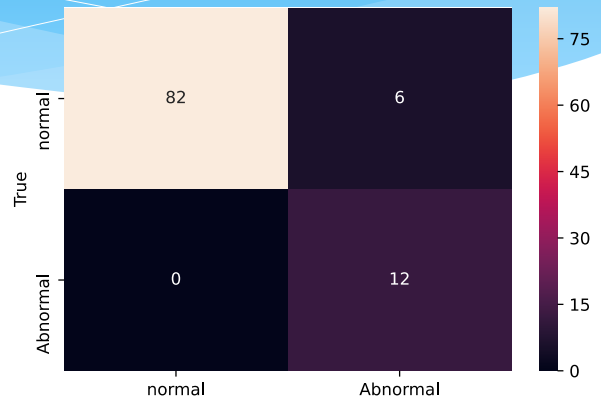
Experimental results



Alexnet-based autoencoder



VGG16-based autoencoder

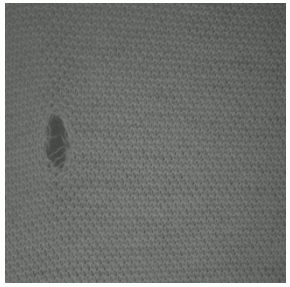


Bergmann's autoencoder

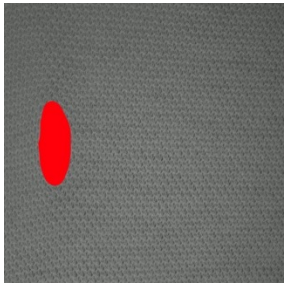
method	Accuracy	Precision	Recall
Proposed Alexnet-based autoencoder	0.98	0.977	1
Proposed VGG16-based autoencoder	0.97	0.966	1
Bergmann's autoencoder [21]	0.94	0.932	1
Liu's method [27]	0.98	0.977	1

Experimental Results: Fabric Defect Region Detection

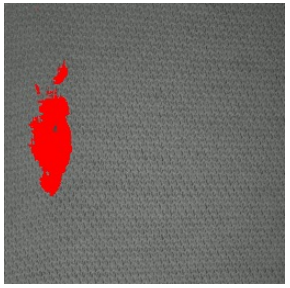
defect image



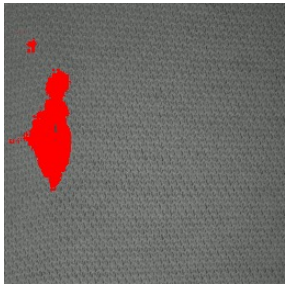
ground-truth



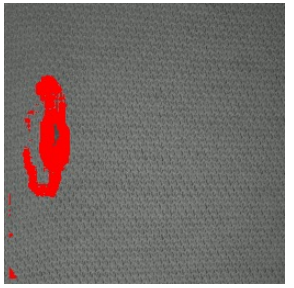
Alexnet-based



VGG16-based

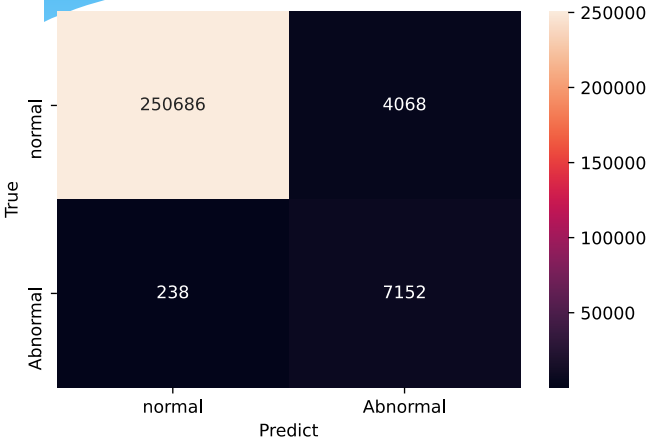


Bergmann's

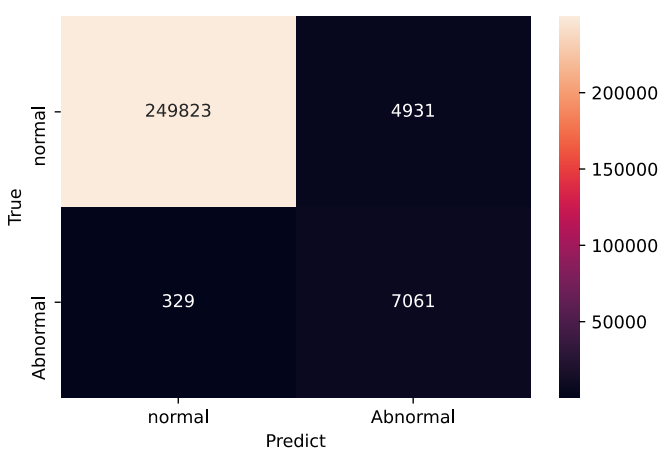


Experimental Results: Fabric Defect Region Detection

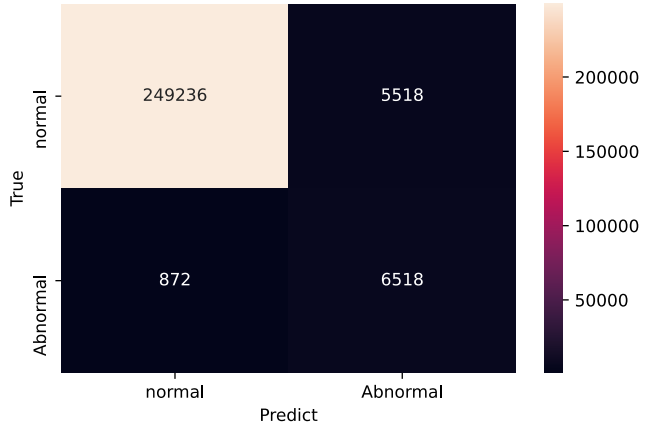
Alexnet-based autoencoder



VGG16-based autoencoder

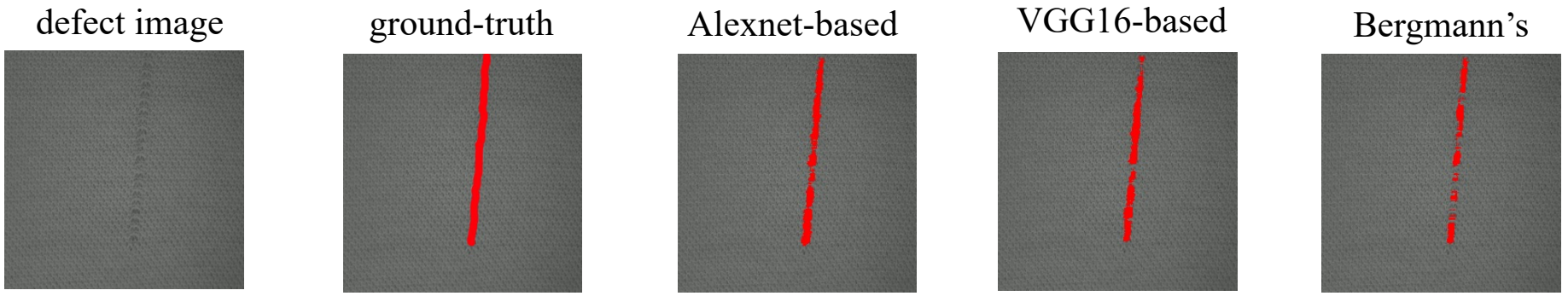


Bergmann's autoencoder



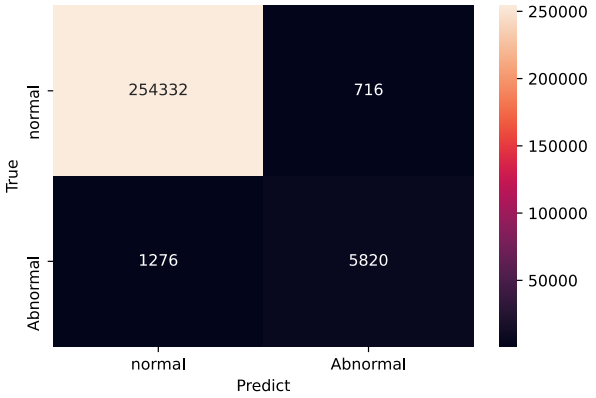
method	Accuracy	Precision	Recall
Proposed Alexnet-based autoencoder	0.98357391	0.98403165	0.99905151
Proposed VGG16-based autoencoder	0.97993469	0.98064407	0.9986848
Bergmann's method [21]	0.97562408	0.97833989	0.99651351

Experimental Results: Fabric Defect Region Detection

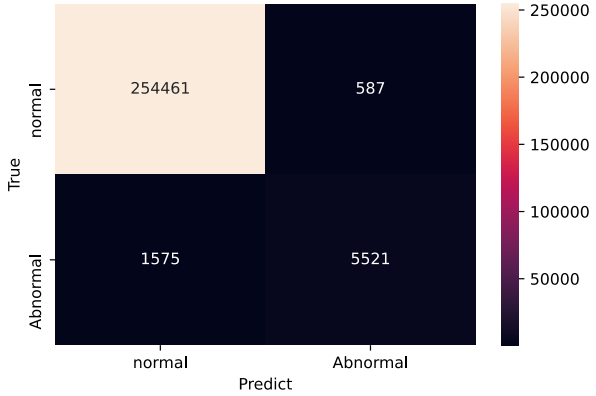


Experimental Results: Fabric Defect Region Detection

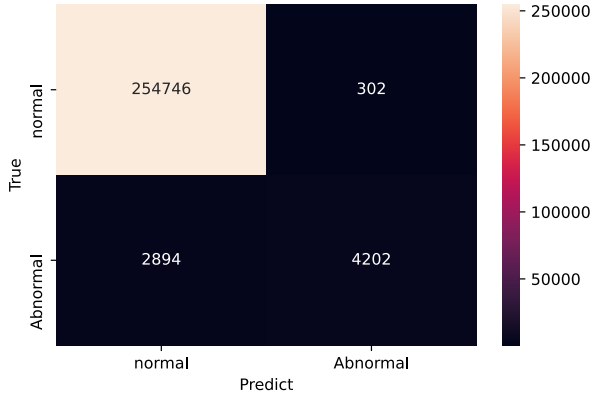
Alexnet-based autoencoder



VGG16-based autoencoder



Bergmann's autoencoder

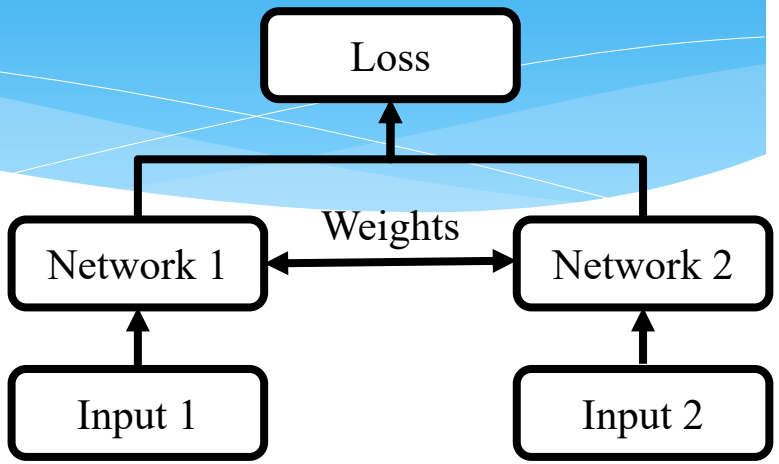


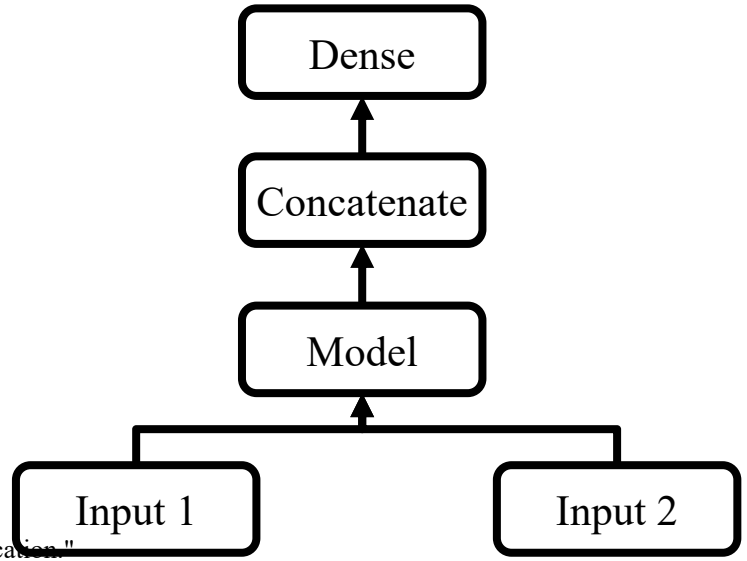
method	Accuracy	Precision	Recall
Proposed Alexnet-based autoencoder	0.99240112	0.99719269	0.99500798
Proposed VGG16-based autoencoder	0.99175262	0.99769847	0.99384852
Bergmann's method [21]	0.98780823	0.99881591	0.98876727

One-shot Siamese Network

Siamese Network

* two inputs: Input 1 & Input 2
* two networks: Network 1 & Network 2



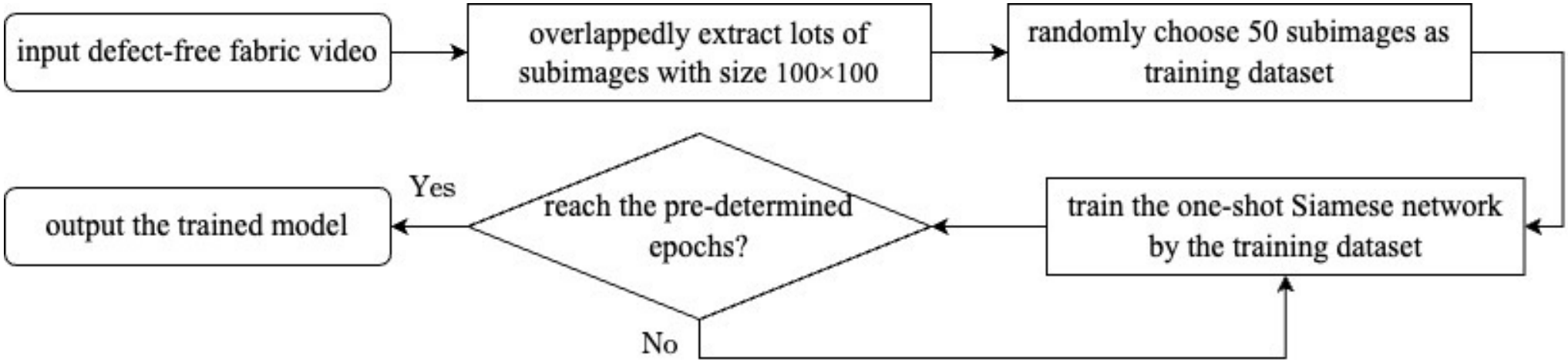


Sumit Chopra, Raia Hadsell, Yann LeCun, "Learning a similarity metric discriminatively, with application to face verification."
IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05). vol.1, pp.539-546, 2005.

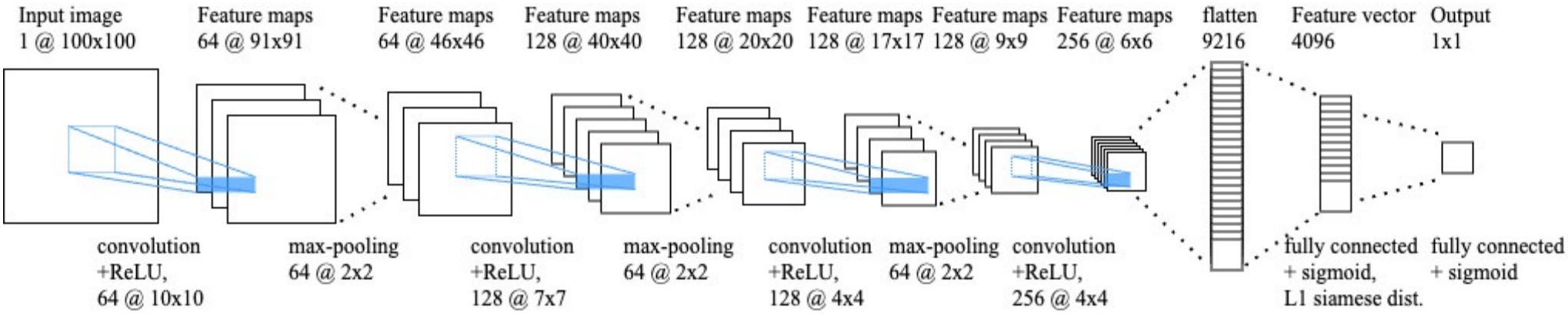
One-shot learning

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, Daan Wierstra, "Matching networks for one shot learning." Advances in Neural Information Processing Systems 29, 2016.

proposed one-shot Siamese network training procedure



proposed one-shot Siamese network system structure



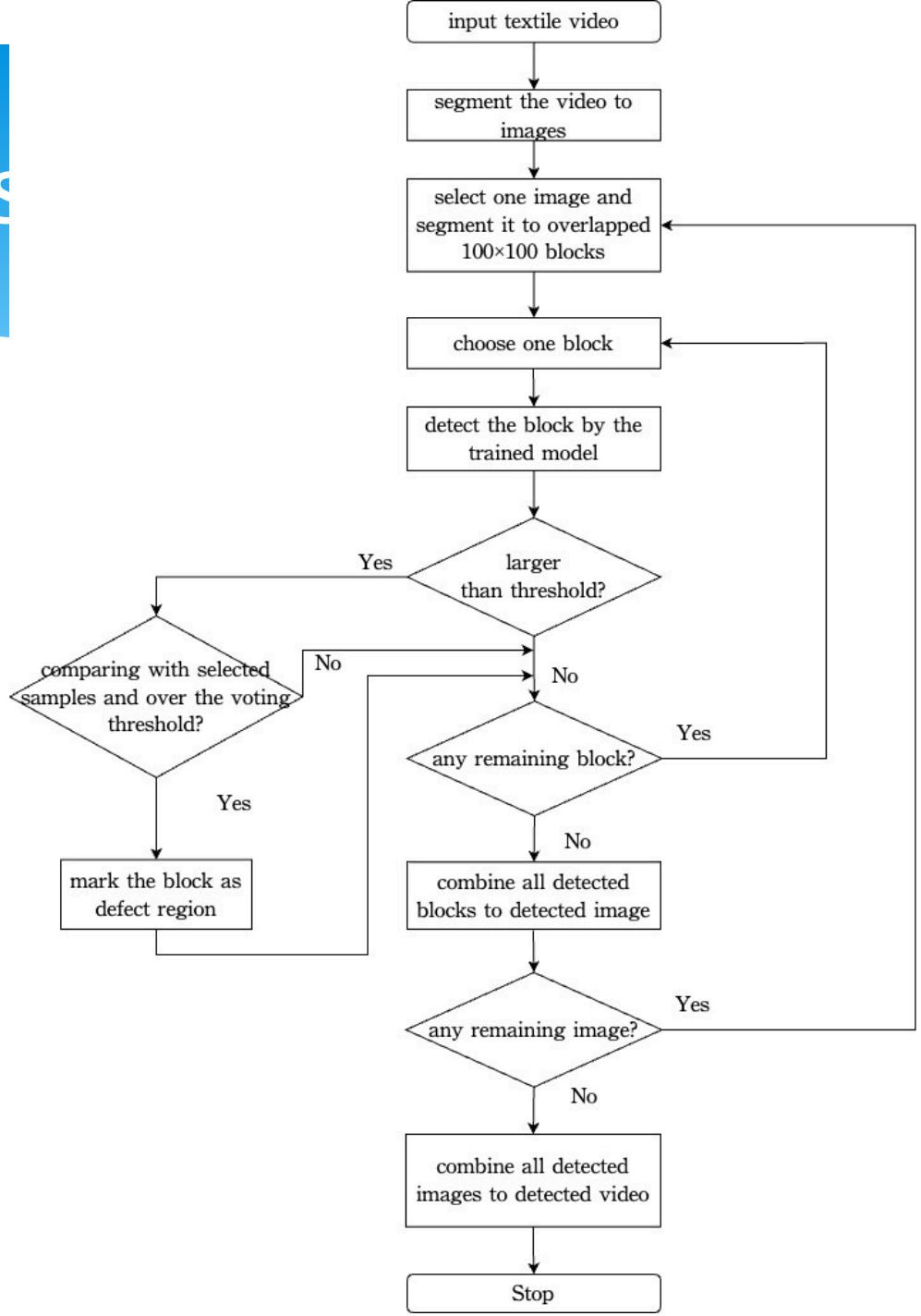
Proposed Siamese Network

layer	size	kernel	Activation function
input	100×100×3		
Convolution layer 1	91×91×64	3×10×10×64	ReLU
Maxpooling layer 1	46×46×64	2×2	
Convolution layer 2	40×40×128	64×7×7×128	ReLU
Maxpooling layer 2	20×20×128	2×2	
Convolution layer 3	17×17×128	128×4×4×128	ReLU
Maxpooling layer 3	9×9×128	2×2	
Convolution layer 4	6×6×256	128×4×4×256	ReLU
Flattern	9216		
Dense	4096		Sigmoid

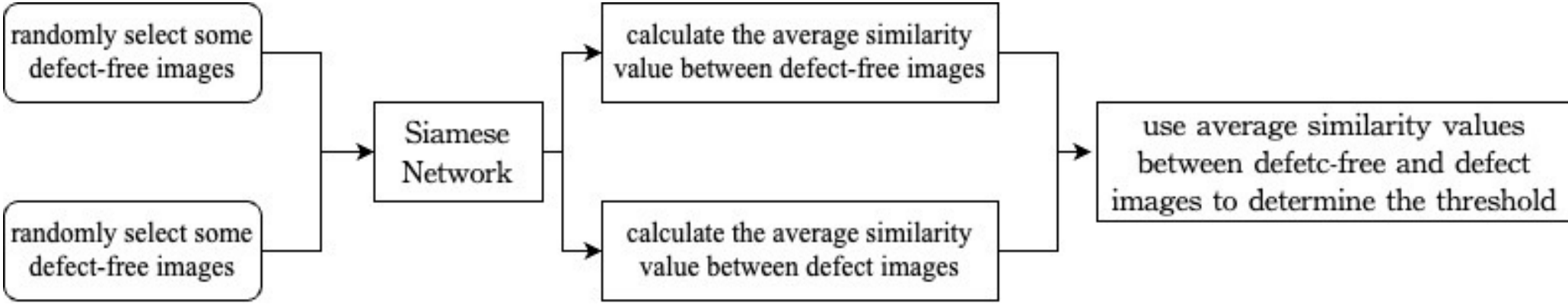
Proposed One-Shot Siamese Network

layer	size	activation function
input	100×100×3	
verification	100×100×3	
Proposed Siamese Network	4096	
L1 distance	4096	
Dense	1	Sigmoid

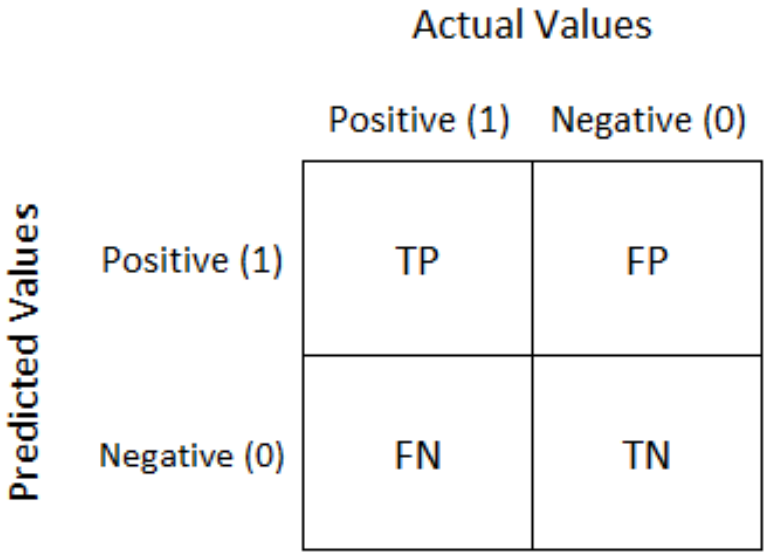
Propos dure


input textile video
segment the video to images
select one image and segment it to overlapped 100×100 blocks
choose one block
detect the block by the trained model
larger than threshold?
Yes
comparing with selected samples and over the voting threshold?
No
Yes
mark the block as defect region
No
any remaining block?
Yes
No
combine all detected blocks to detected image
any remaining image?
Yes
No
combine all detected images to detected video
Stop

Threshold Determination



Confusion Matrix



True Positive:

predicted positive and it's true

You predicted a woman is pregnant and she actually is

True Negative:

predicted negative and it's true

You predicted a man is not pregnant and he actually is not

False Positive: (Type 1 Error)

predicted positive and it's false

You predicted a man is pregnant but he actually is not

False Negative: (Type 2 Error)

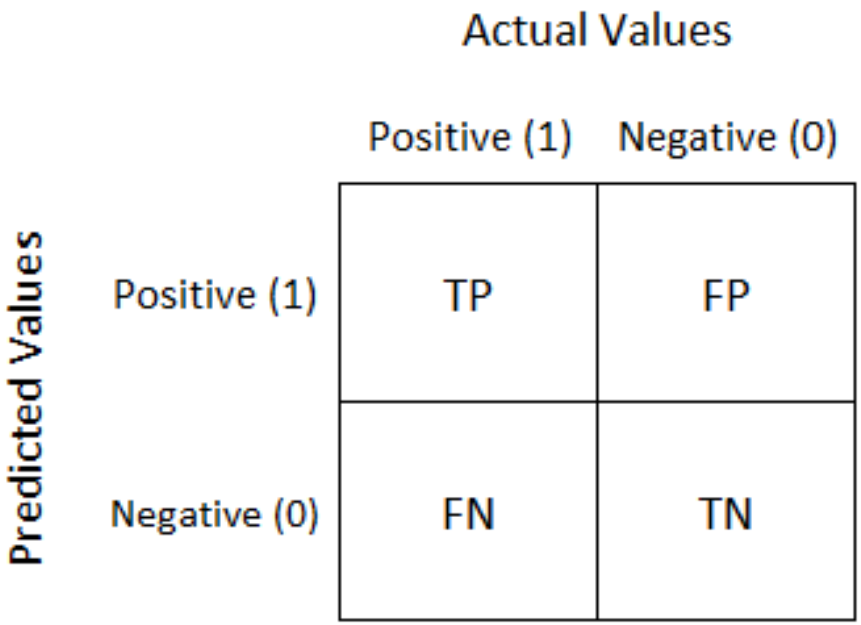
predicted negative and it's false

You predicted a woman is not pregnant but she actually is

Confusion Matrix

Accuracy= $\frac{TP+TN}{TP+TN+FP+FN}$

From all the classes (positive and negative), how many of them we have predicted correctly
Accuracy should be high as possible



Precision= $\frac{TP}{TP+FP}$

predicted as positive, how many are actually positive
Precision should be high as possible

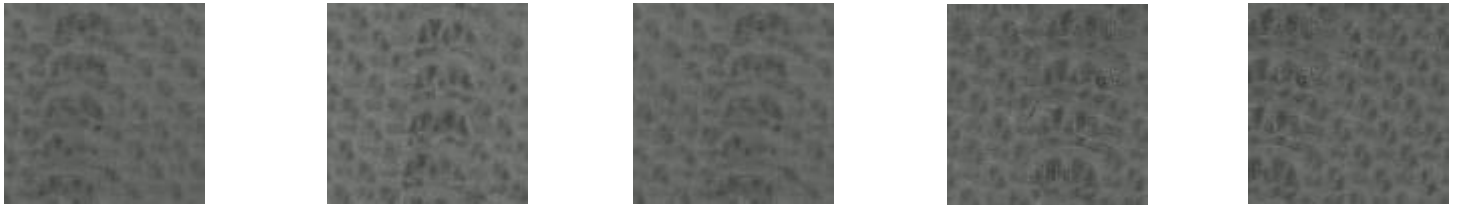
Recall = $\frac{TP}{TP+FN}$

from all the positive classes, how many we predicted correctly
Recall should be high as possible

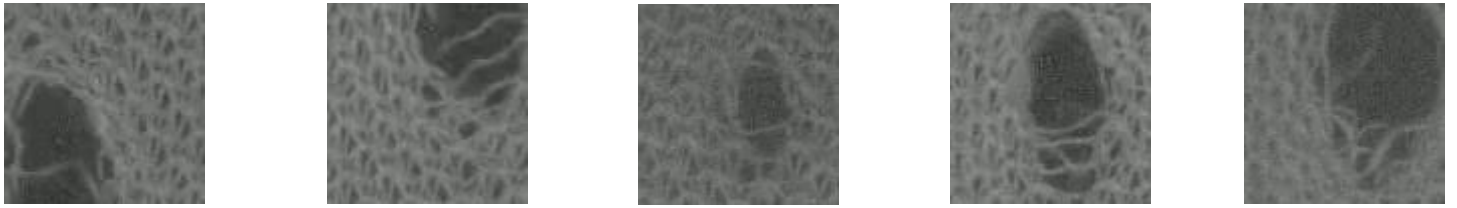
Datasets

* Dataset1: defect-free images

* Datasets 2: line defect images



* Datasets 3: hole defect images

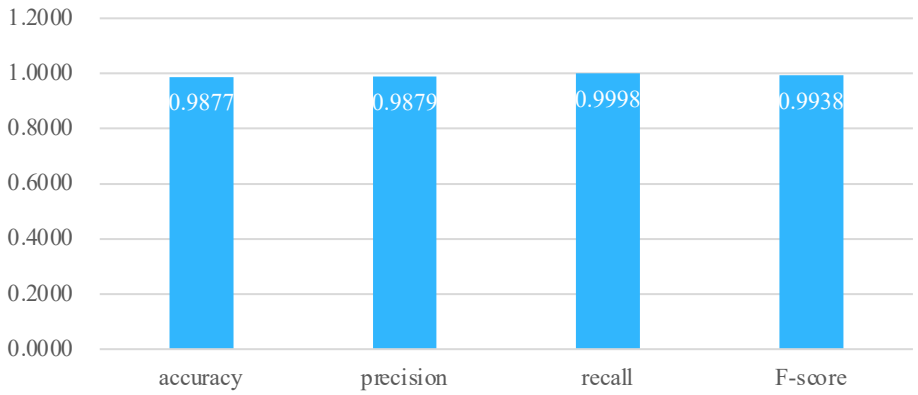


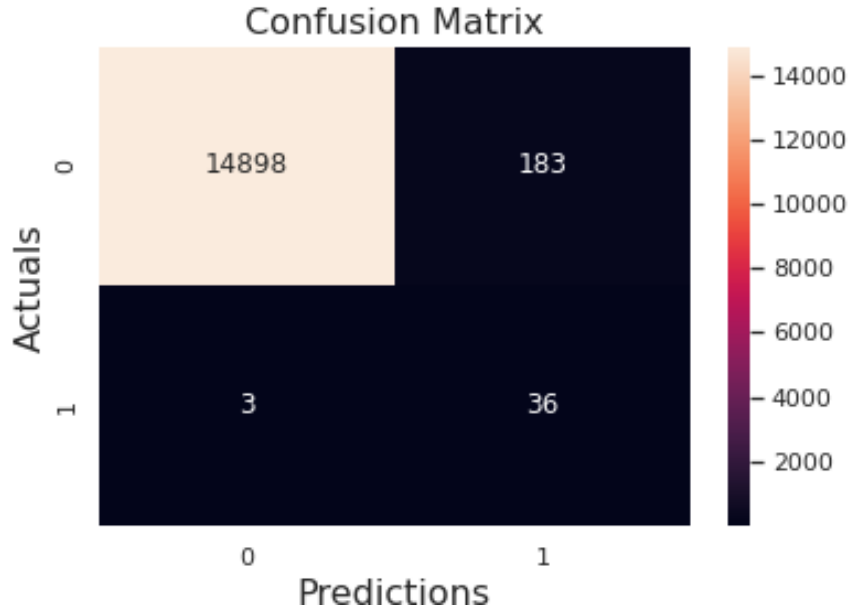
Experimental Results

* randomly choose 500 samples with size 100×100 from datasets
* epoch: 20
* learning Rate: 0.0001
* activation function: ReLU and Sigmoid
* loss function: BinaryCrossEntropy

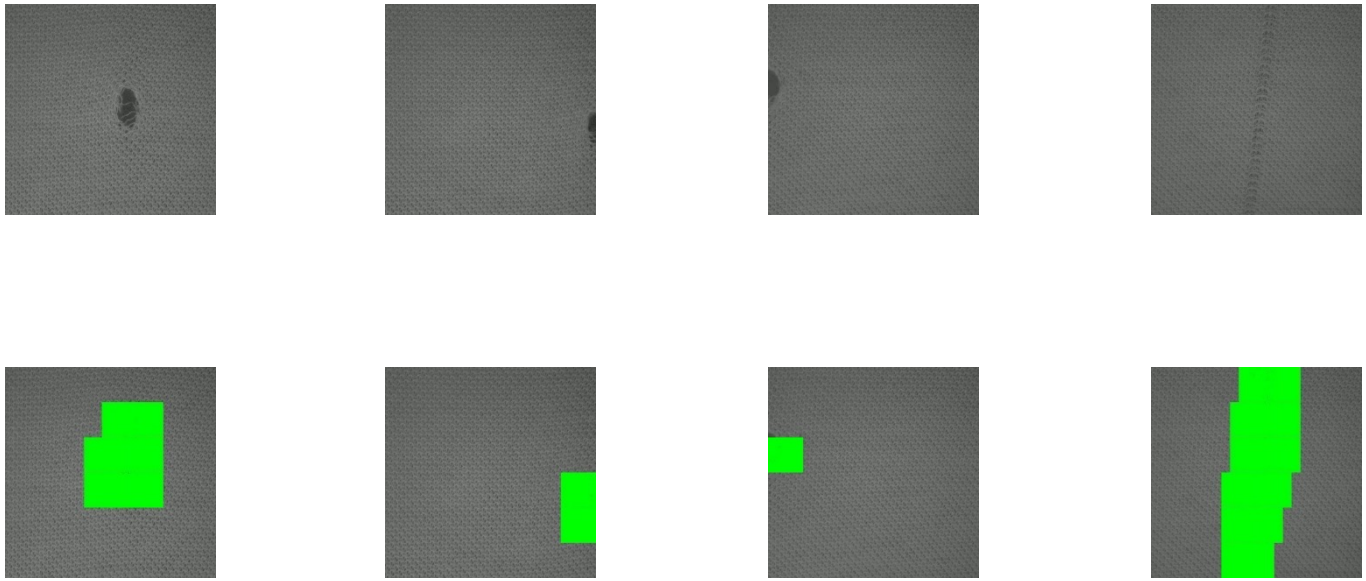
Experimental Results (train by datasets 2,3)

* Totally 15120 test images with size 100×100
  * 15081 defect-free images
  * 39 defect images

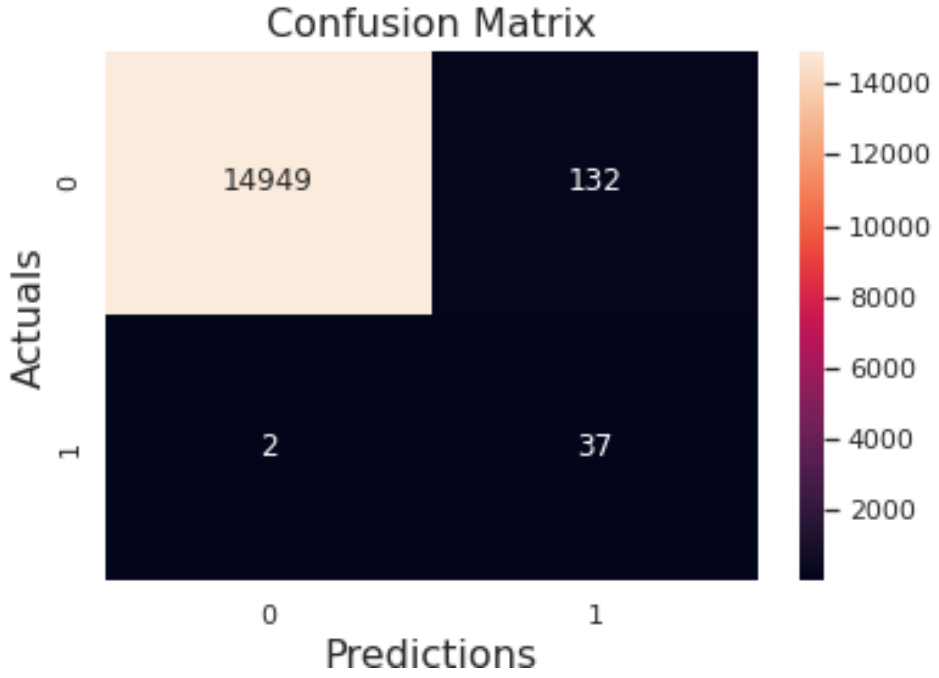




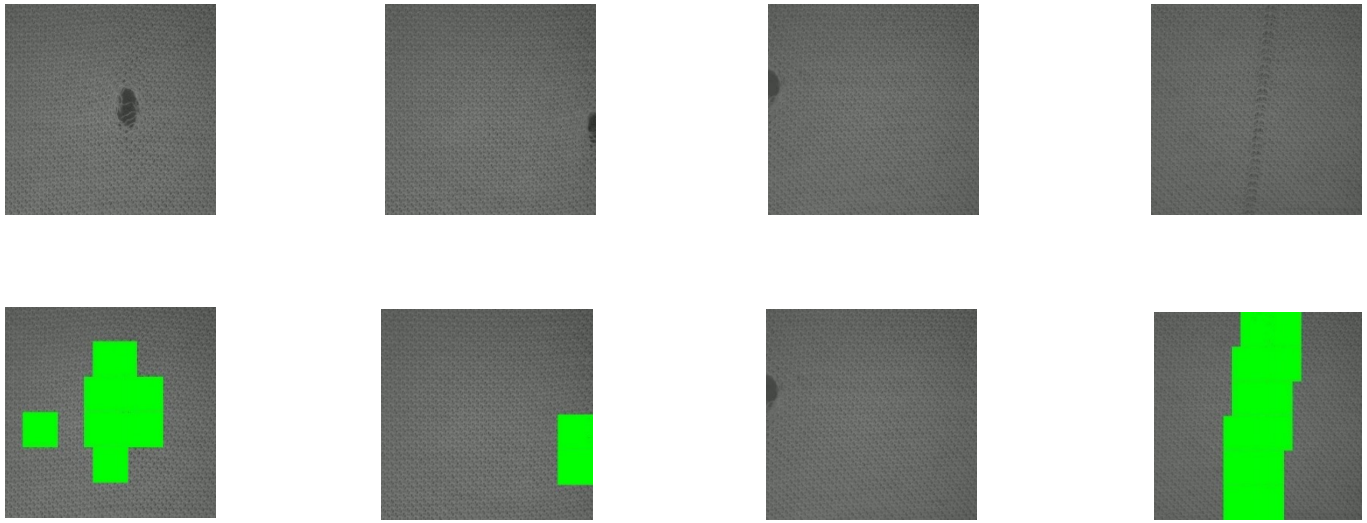
Experimental Results (train by datasets 2,3)



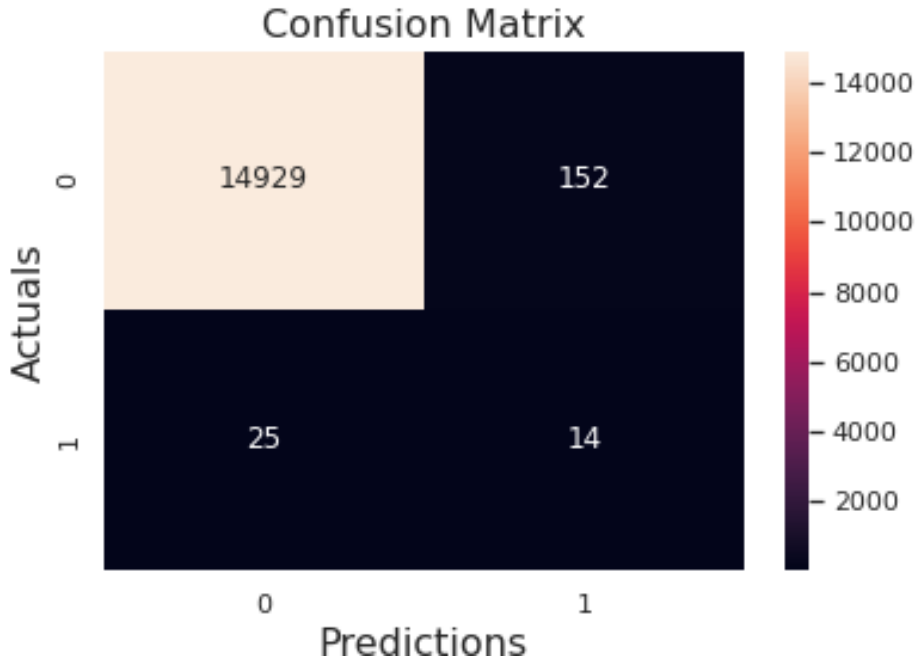
Experimental Results (train by datasets 2(line defect))



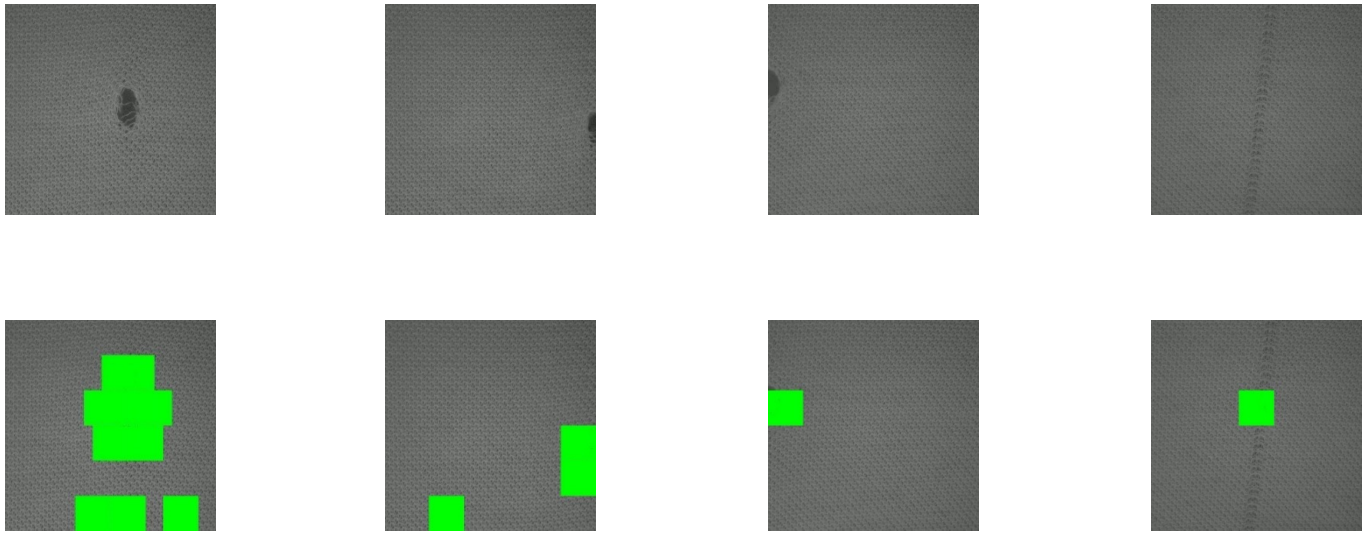
Experimental Results (train by datasets 2(line defect))



Experimental Results (train by datasets 3(hole defect))

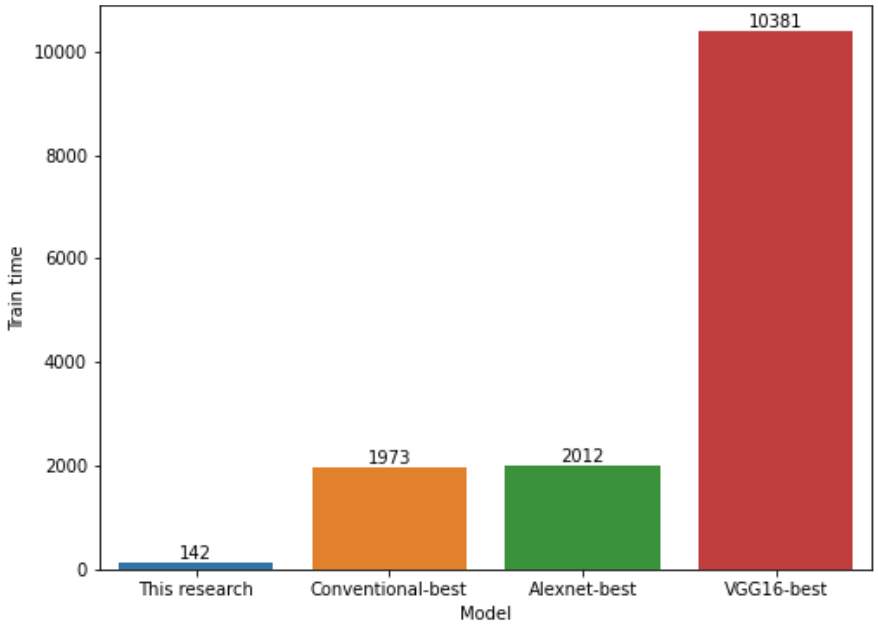


Experimental Results (train by datasets 3(hole defect))

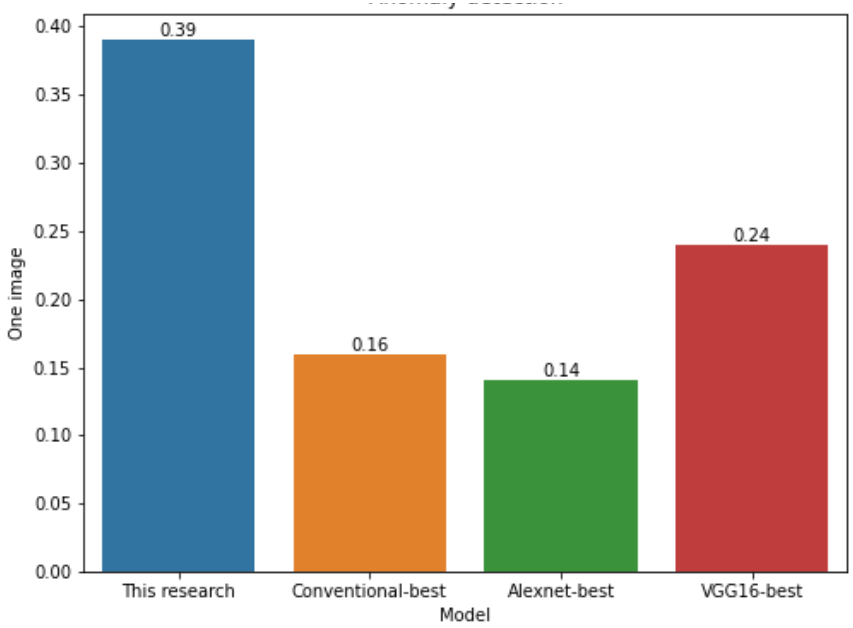


Computation time

Training Time

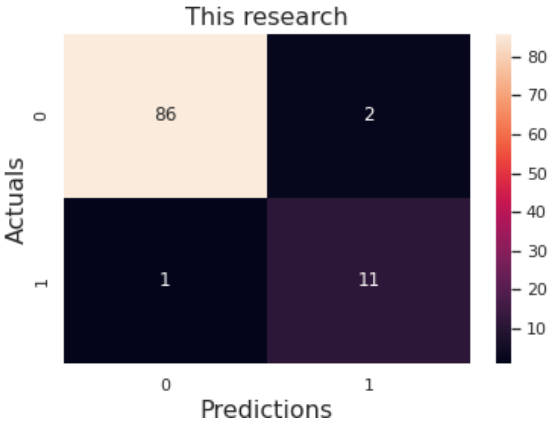


Detection Time

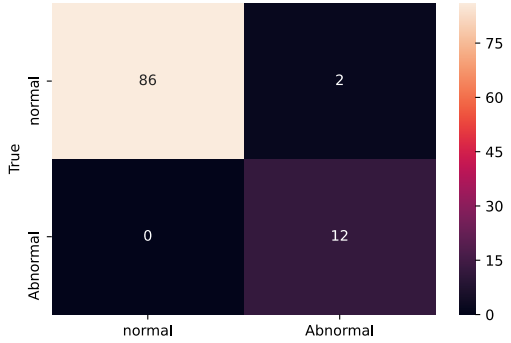


100 test images

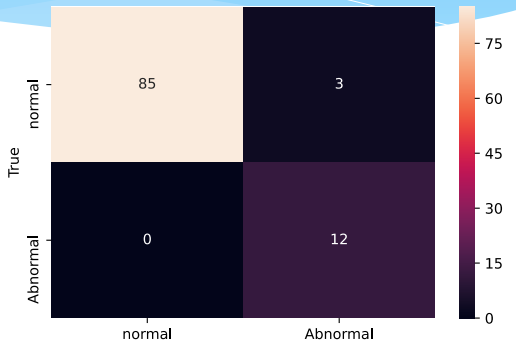
one-shot Siamese



Alexnet-based autoencoder

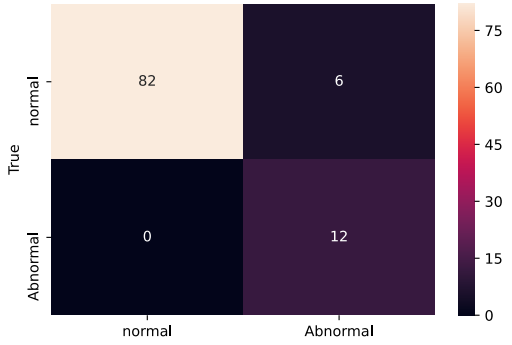


VGG16-based autoencoder

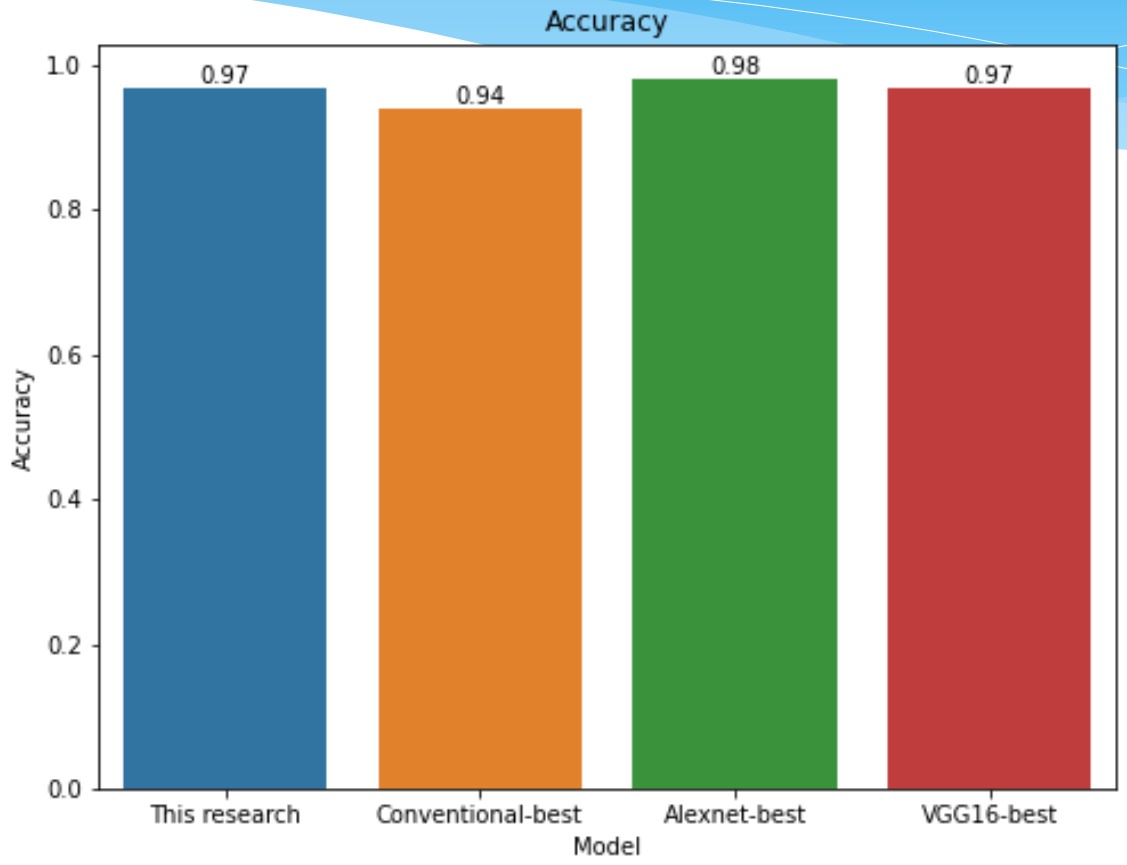


method	Accuracy	Precision	Recall
the proposed one-shot Siamese network	0.97	0.977	0.989
Liu's YOLOv4 based method [13]	0.98	0.977	1
conventional autoencoder [14]	0.94	0.932	1
Alexnet-based autoencoder [15]	0.98	0.977	1
VGG16-based autoencoder [15]	0.97	0.966	1

Bergmann's autoencoder



Experimental Results



Conclusion

* Autoencoder
  * needs lots of data samples (20000)
  * good recognition rate

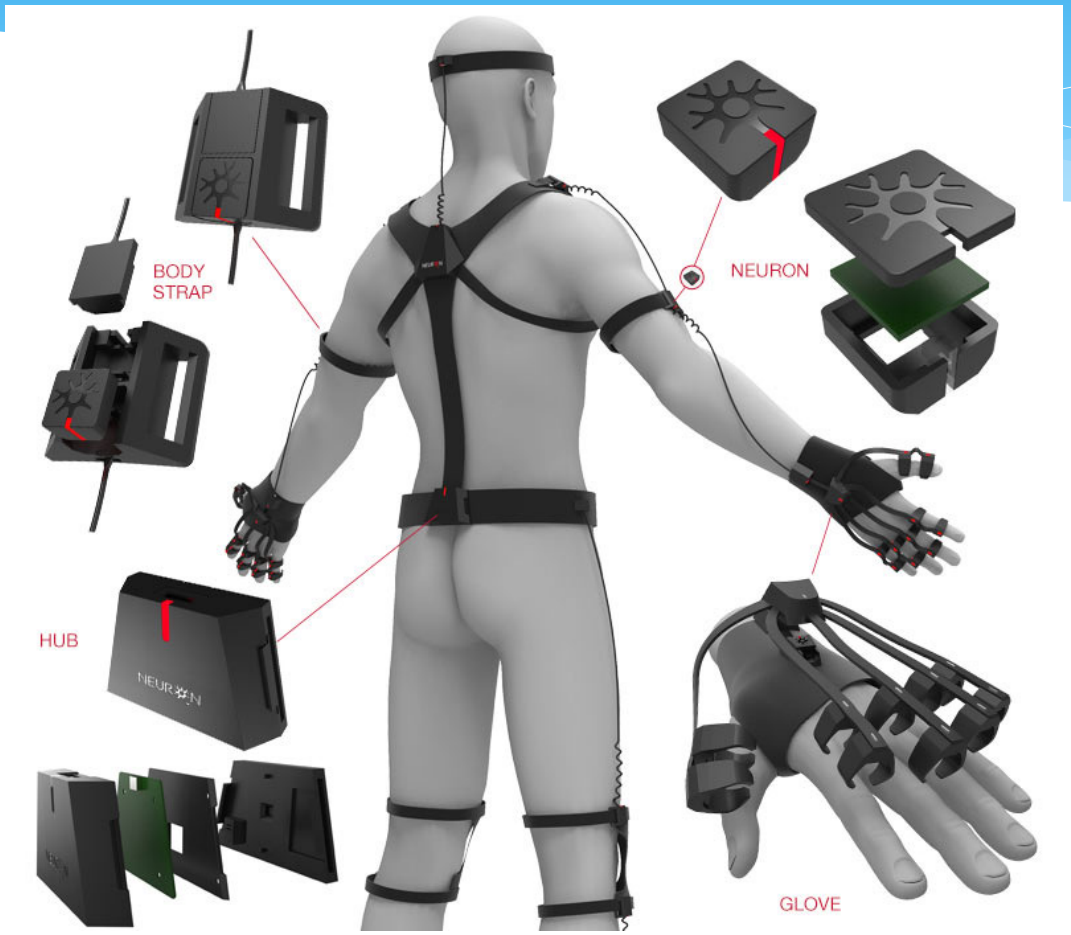
* One-shot Siamese Network
  * only needs some data samples => requires less training time (1500)
  * Needs more detection time because of the voting procedure

Video Based Basketball Shooting Prediction and Pose Suggestion System

Chien-Chang Chen, Chen Chang, Cheng-Shian Lin, Wen-Her Chen, Chien-Hua Chen, I-Cheng Chen, "Video Based Basketball Shooting Prediction and Pose Suggestion System", Multimedia Tools and Applications, 2023.

Basketball Shooting Prediction

In the Past, we need

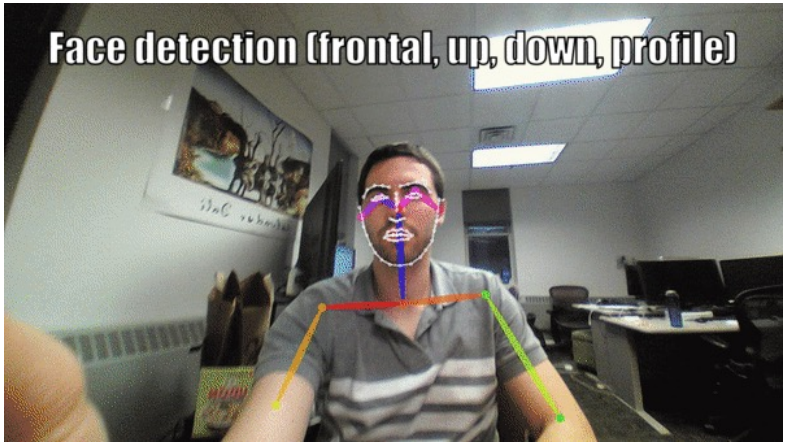


https://www.vgtu.lt/images/3439/171/8/8_1/perception_neuron.jpg

Now, we have

https://www.kindpng.com/picc/m/52-525937_sing-drawing-poses-transparent-png-clipart-free-download.png

OpenPose can do these



https://github.com/CMU-Perceptual-Computing-Lab/openpose

This study

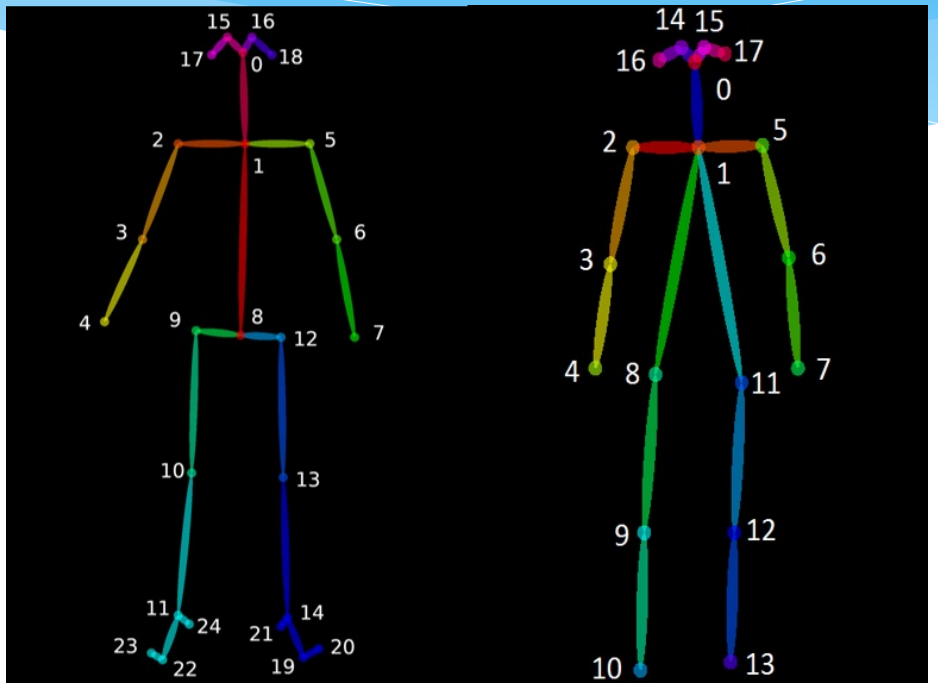
01 find shooting process

02 calculate shooting difference

03 verify the difference

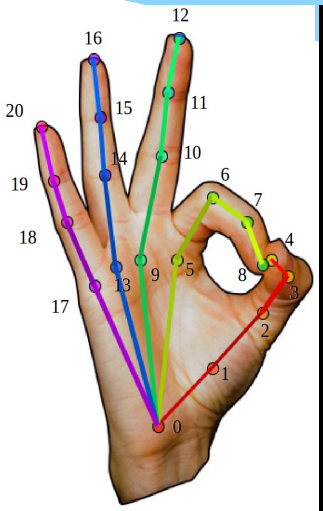
04 give suggestions on shooting

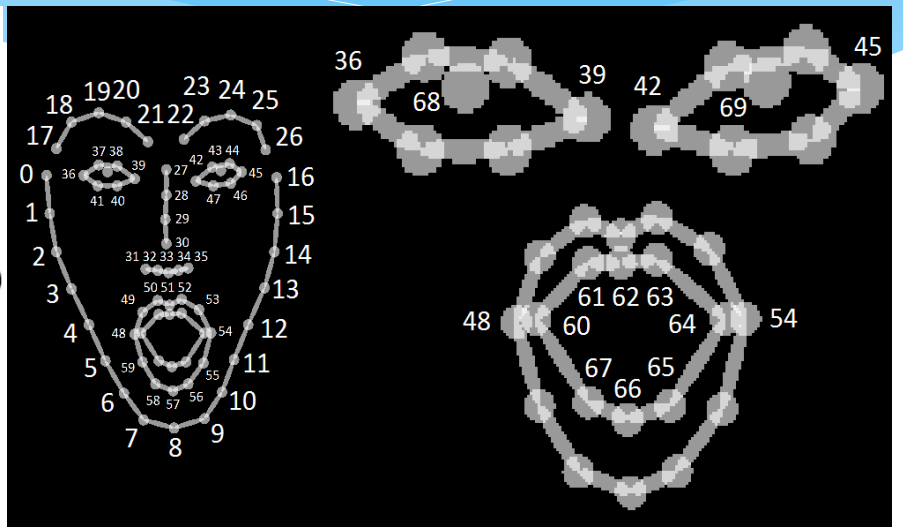
Related works (OpenPose)



https://github.com/CMU-Perceptual-Computing-Lab/openpose/blob/master/doc/output.md

Related works (OpenPose)

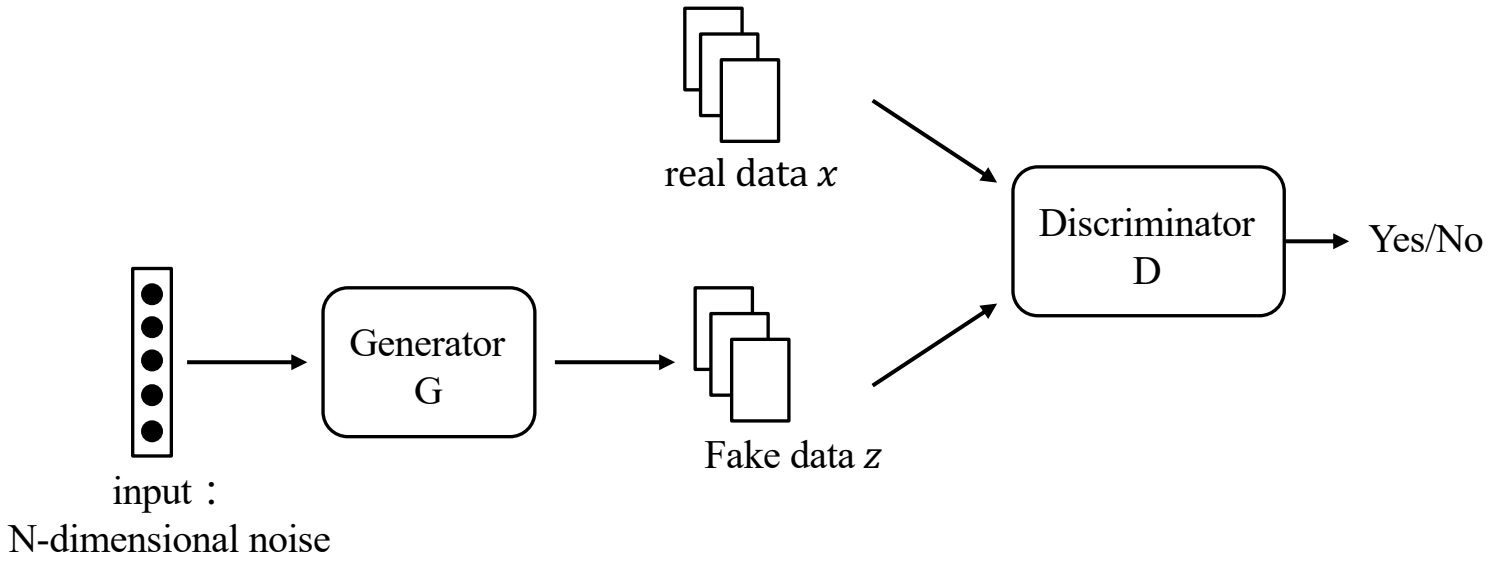




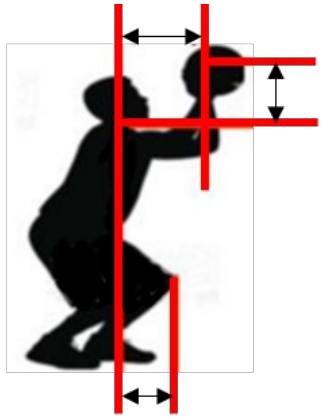
https://github.com/CMU-Perceptual-Computing-Lab/openpose/blob/master/doc/output.md

https://upload.wikimedia.org/wikipedia/zh/thumb/b/bb/Carnegie_Mellon_University_seal.svg/1200px-Carnegie_Mellon_University_seal.svg.png
https://images.exxactcorp.com/CMS/landing-page/resource-center/supported-software/logo/Deep-Learning/caffe.png
https://upload.wikimedia.org/wikipedia/commons/5/53/OpenCV_Logo_with_text.png
https://miro.medium.com/max/490/0*EqIN97kgnwyUEU1i.png

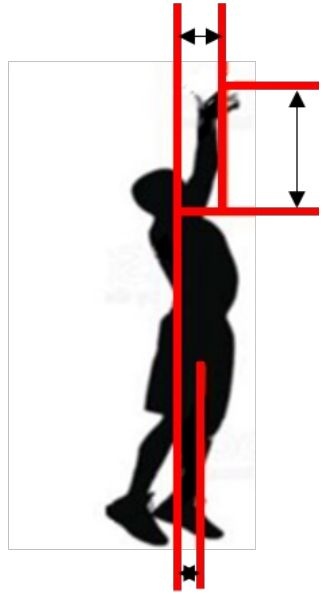
GAN(Generative Adversarial Network)



Shooting procedure



Start



End

M. Nakai, Y. Tsunoda, H. Hayashi, H. Murakoshi, “Prediction of Basketball Free Throw Shooting by OpenPose,” In *Japanese Society for Artificial Intelligence International Symposia on Artificial Intelligence*: *New Frontiers in Artificial Intelligence*, pp. 435-446, 2018.

video frames for one shoot

(a)

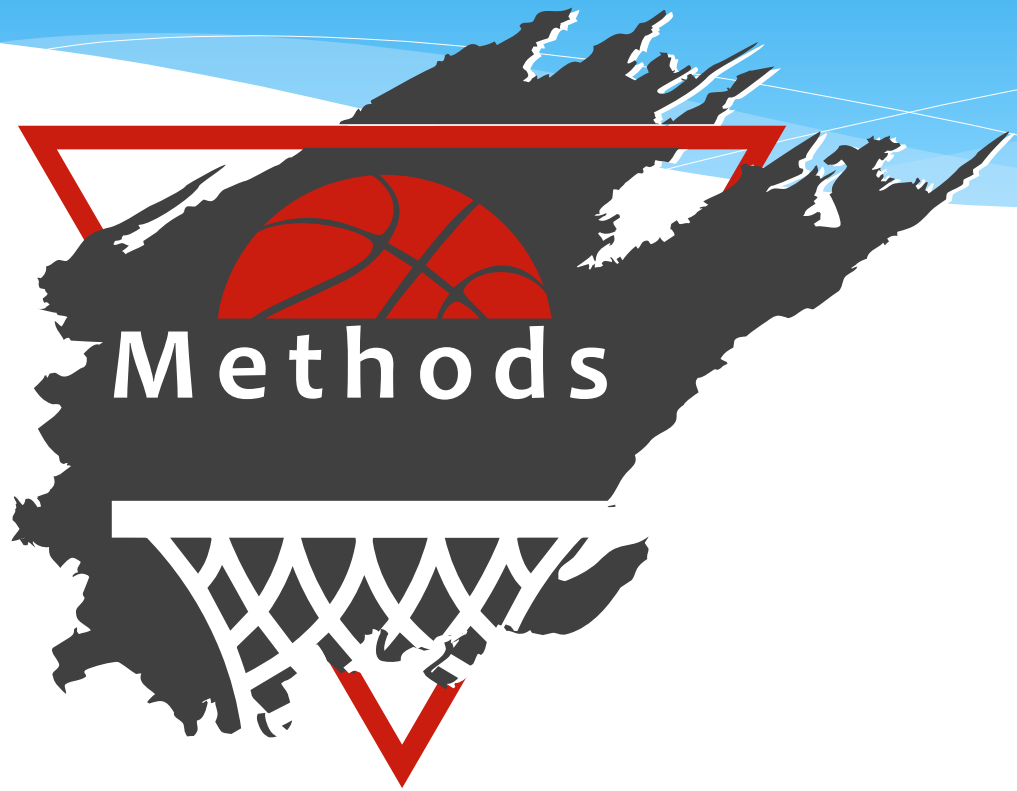
(a) Start of shooting basketball

(b)

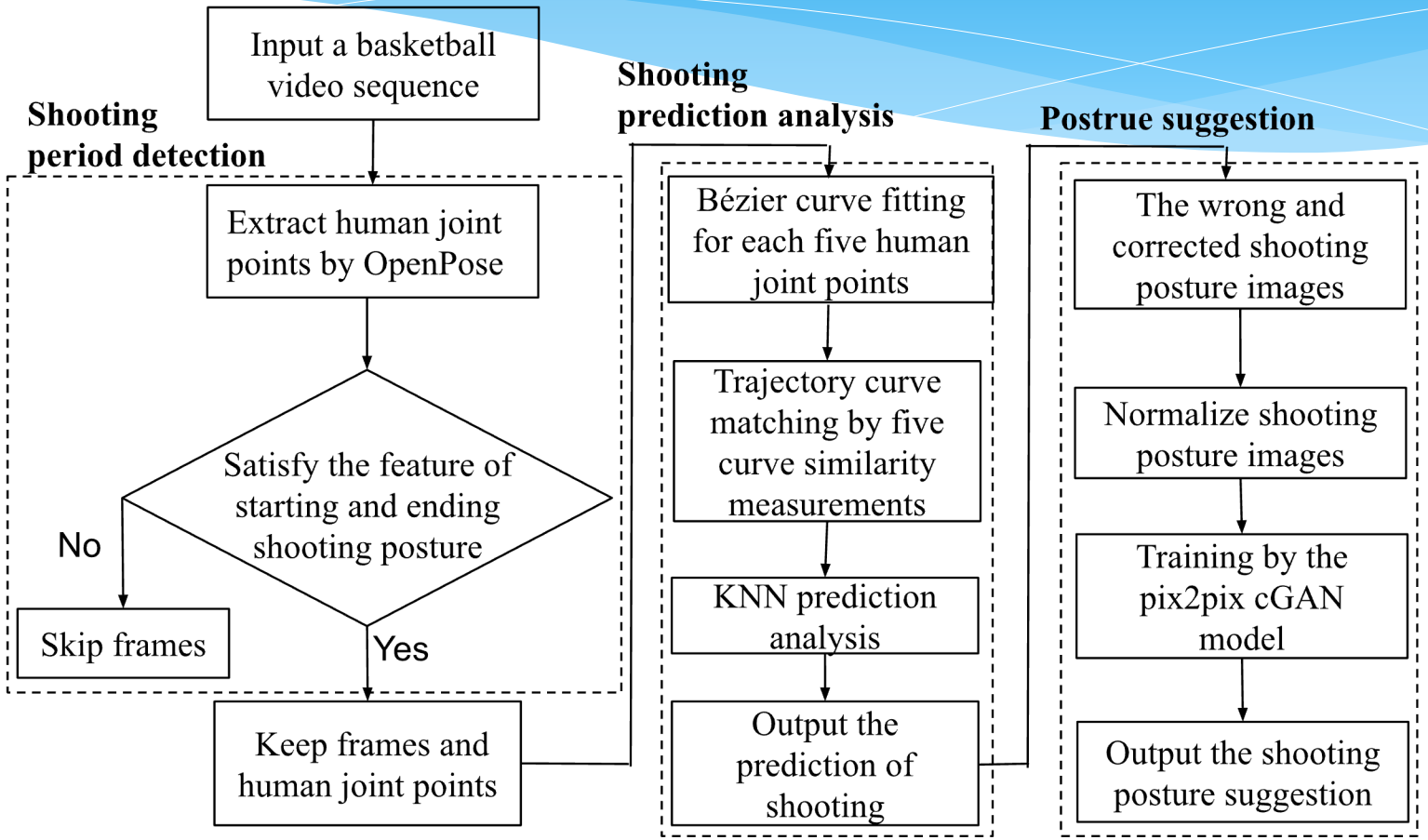
(b) End of shooting basketball

System Extracted Frames

15 Frame from start shooting to end shooting

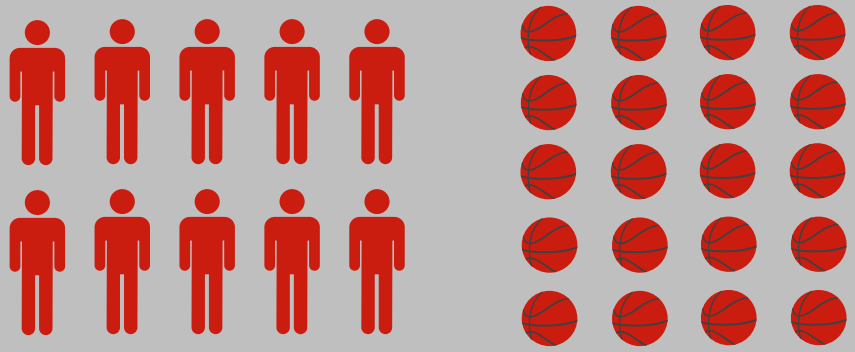

Methods

Shooting prediction and pose suggestions



Record shooting videos

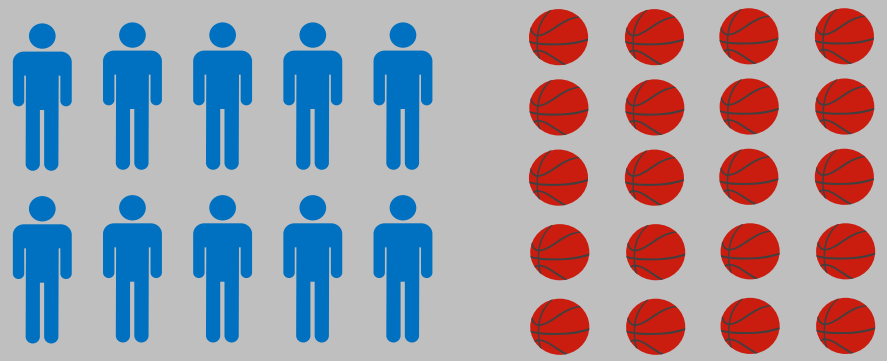
University Team Players



10 people

20 shoots

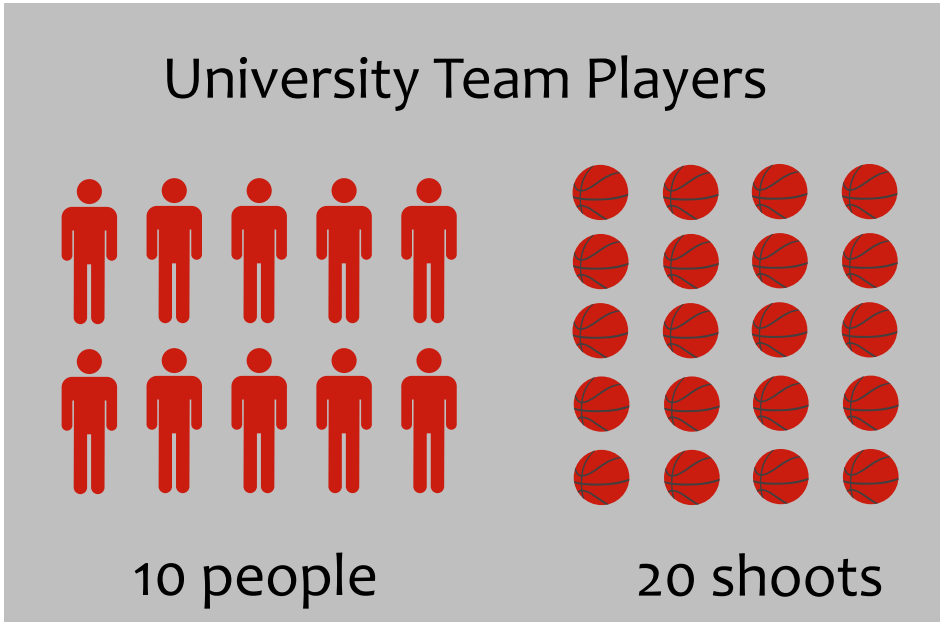
Basketball Class Students

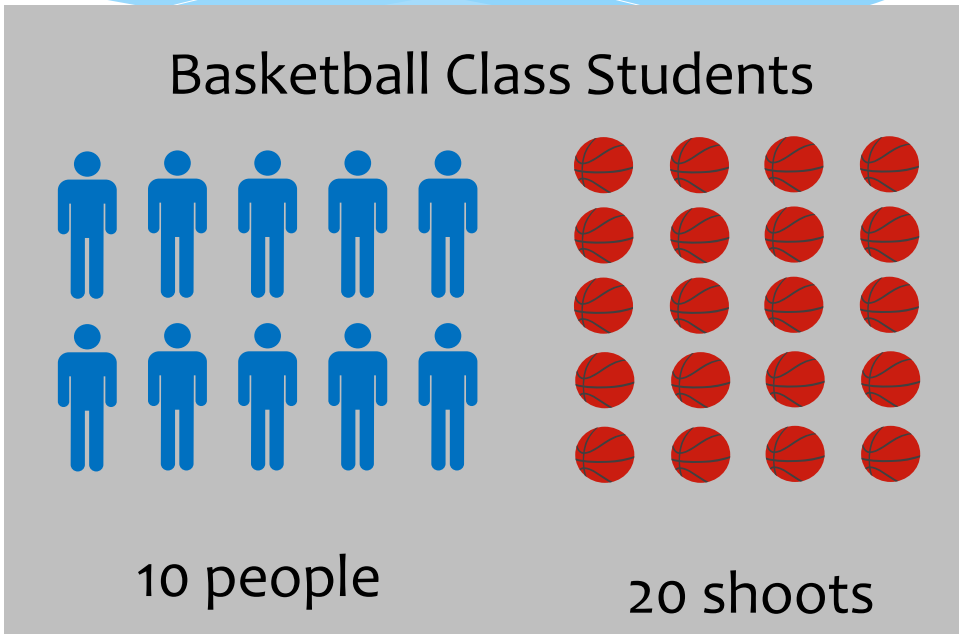


10 people

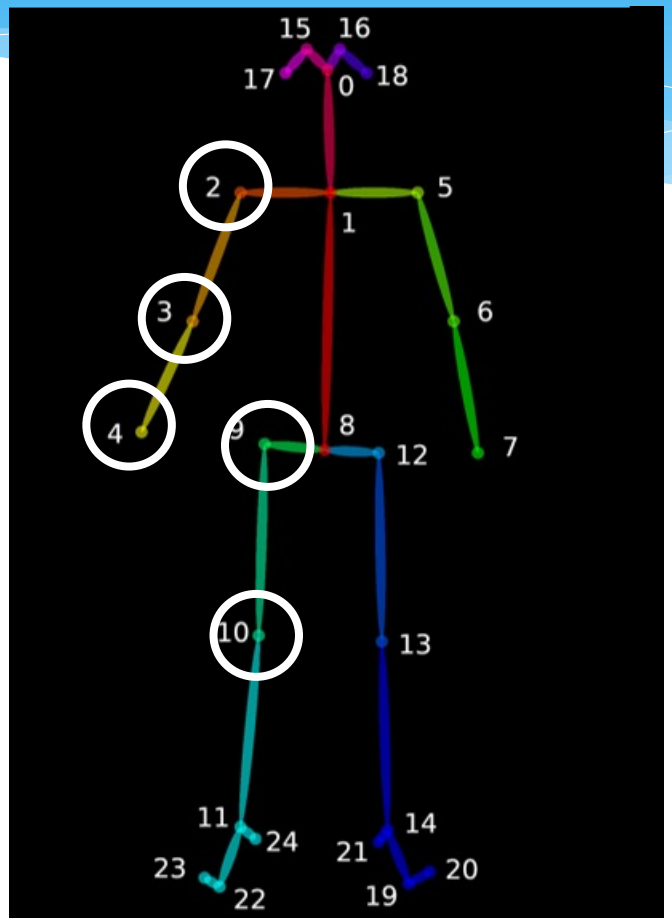
20 shoots

Get joint points by OpenPose



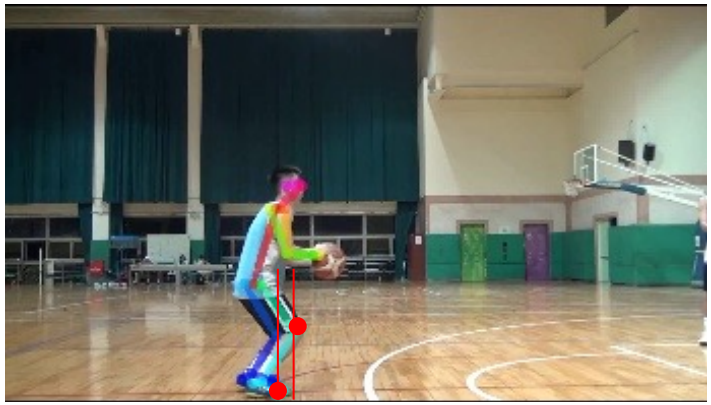


Only choose 5 joint point data



One shoot process

elbow reaches the highest point



...

knee exceeds the tip of the toe

101 frame

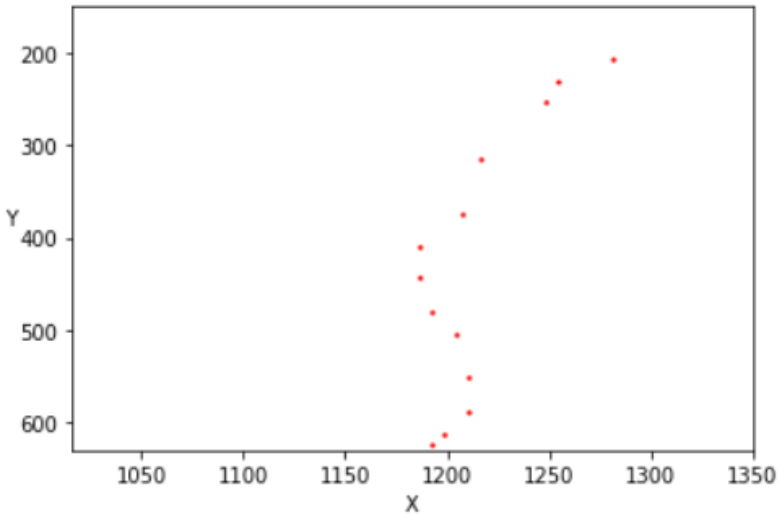
115 frame

start shooting

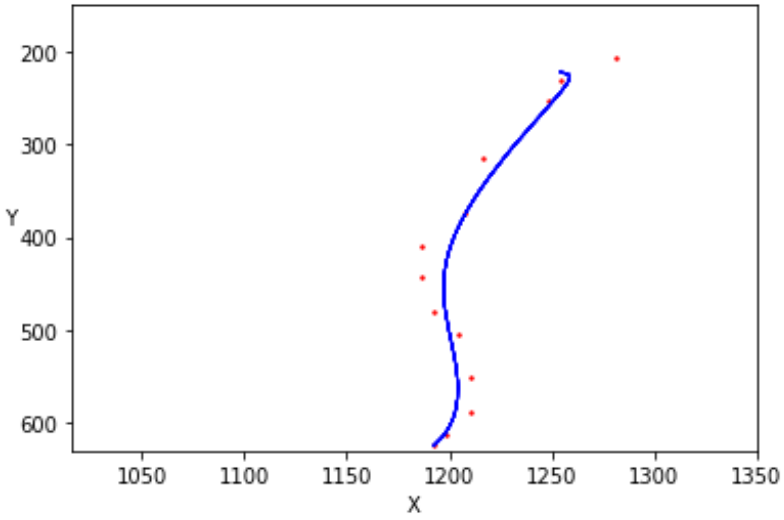
end shooting

Use Bezier Curve to draw

15 frame 15 curve

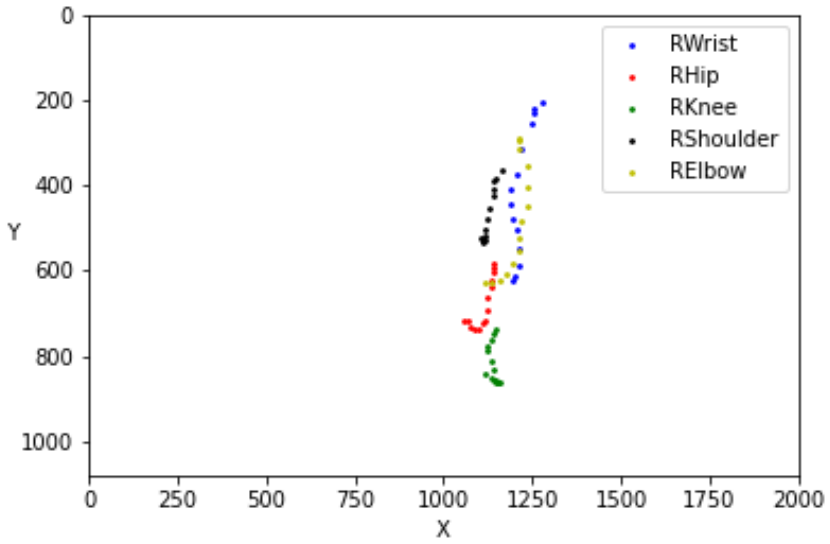


Joint points coordinates

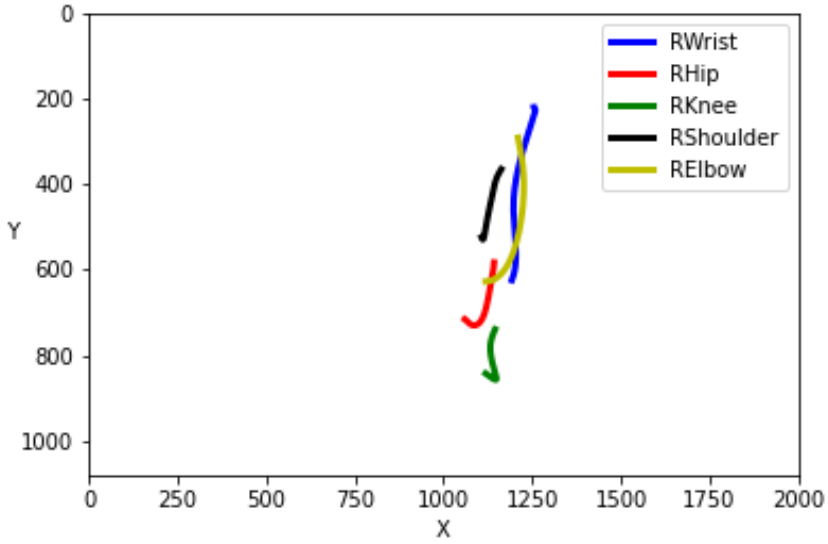


Bezier Curve

joint points during a shooting period



The created Bezier Curve

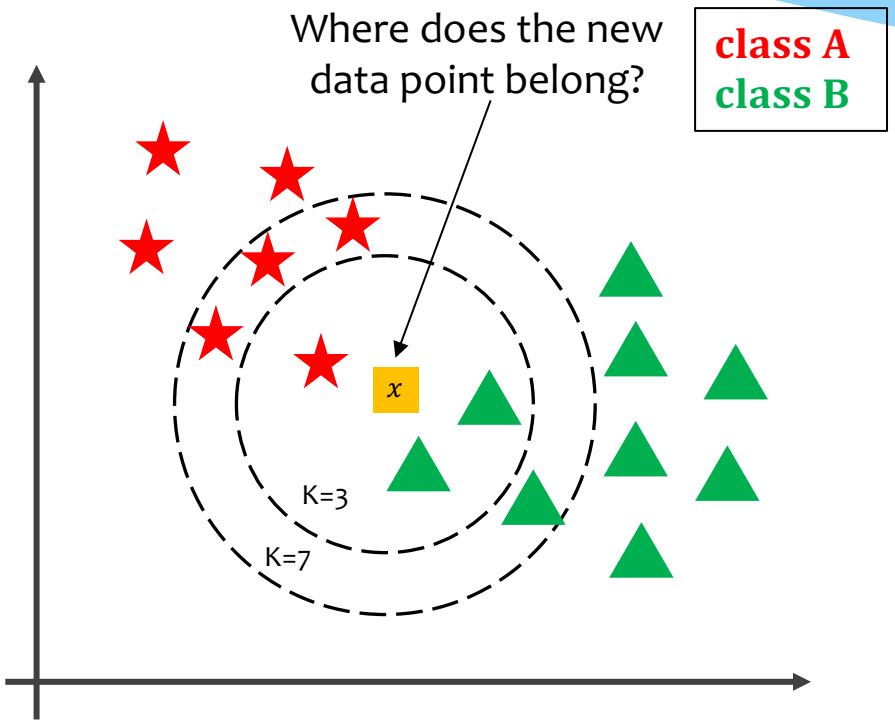


Dataset

10 people 20 shots 5 joint points

5 different distance measurements

K-NN(Nearest Neighbor)



$$h(x) = \arg\max_{y \in \{1 \ldots t\}} \sum_{i=1}^{k} \delta\left(y, f(x_i)\right), k \equiv 1 \pmod 2$$

$$\delta(a,b) = \begin{cases} 1, & if\ a = b \\ 0, & if\ a \neq b \end{cases}$$

$y \in \{$**class A**、**class B**$\}$

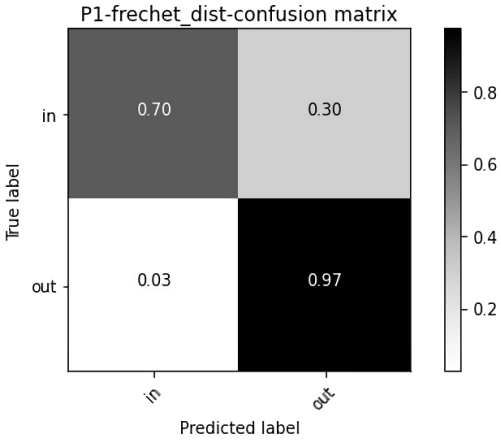
Datasets

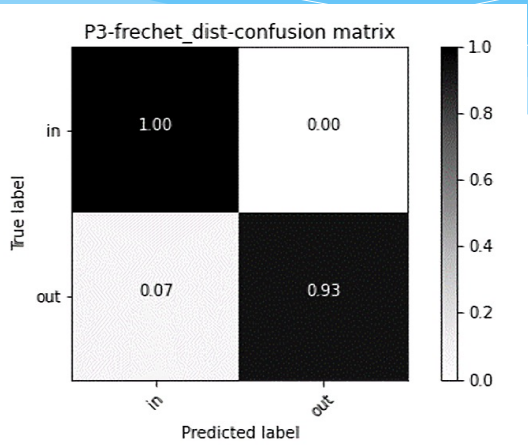
	University Team Players	Basketball Class Players
The number of people	10	10
The number of shooting	20	20
The number of valid shooting	191	181
# of prediction of individual difference experiment	955	905
# of prediction of total difference experiment	191	181
The k value in KNN algorithm	5	3

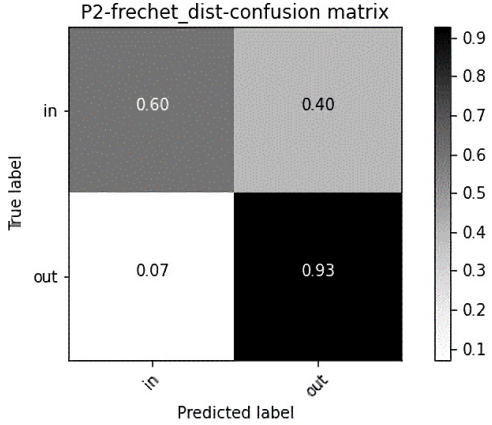
Similarity Algorithms

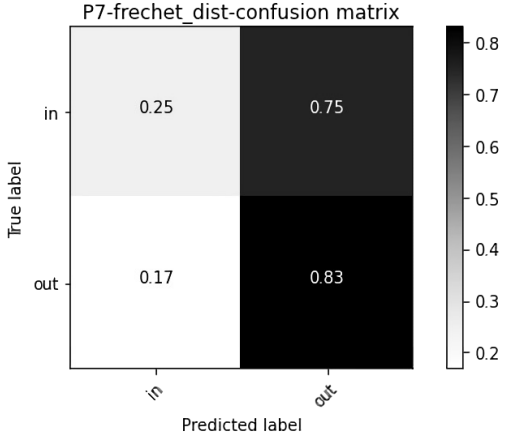
Algorithms	Notation
Partial Curve Mapping	C1
Area method	C2
Discrete Frechet distance	C3
Curve Length	C4
Dynamic Time Warping, DTW	C5

Basketball Class Player Confusion Matrix

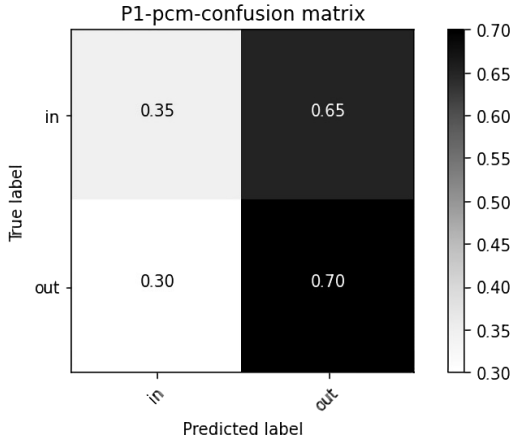


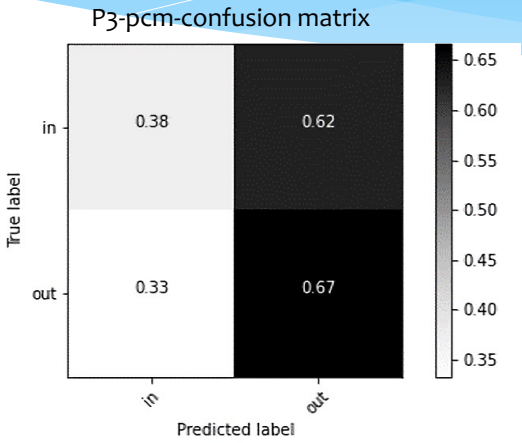


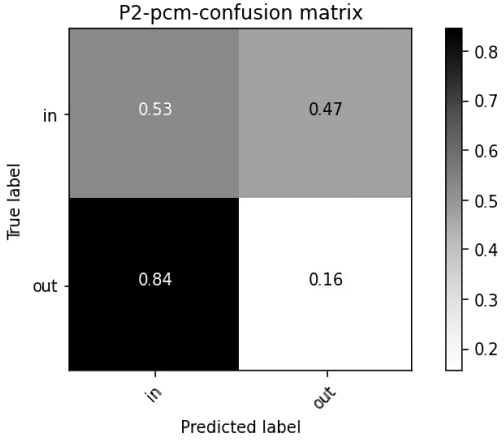


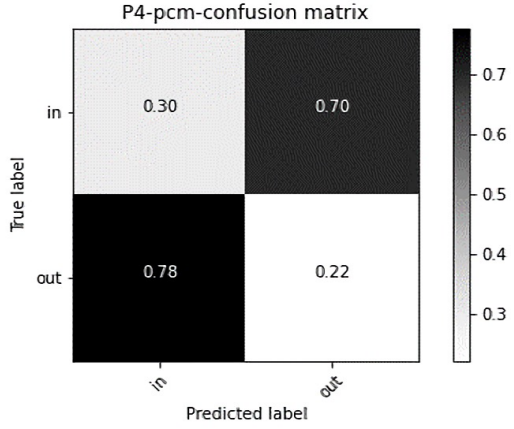


University Team Players Confusion Matrix



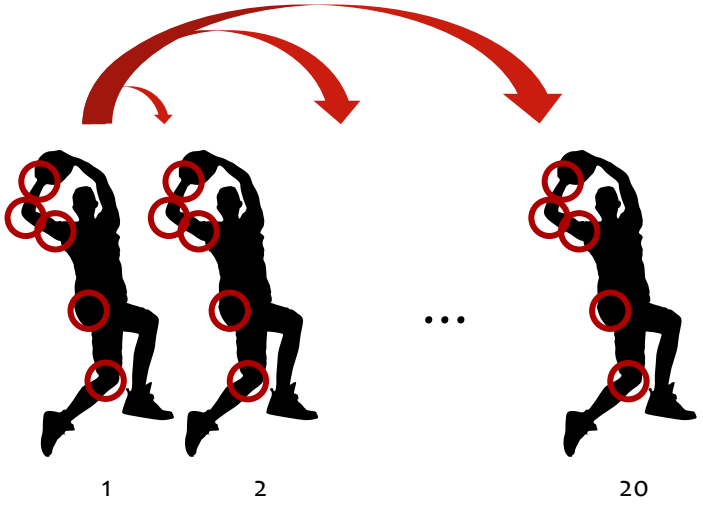




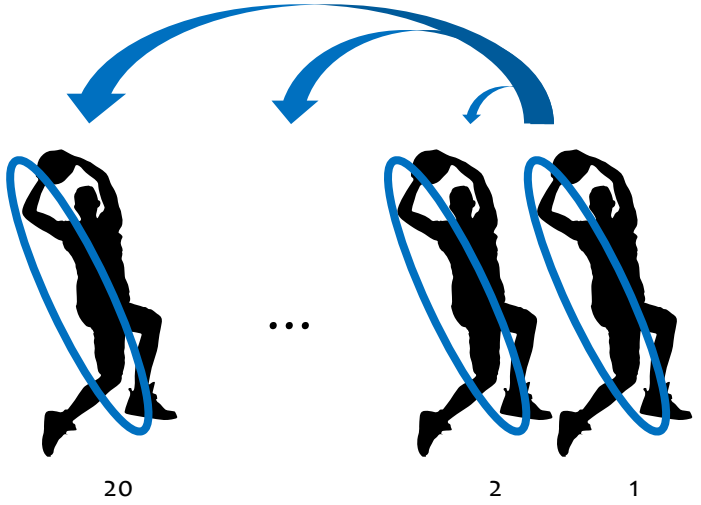


Two Kinds of Experiments

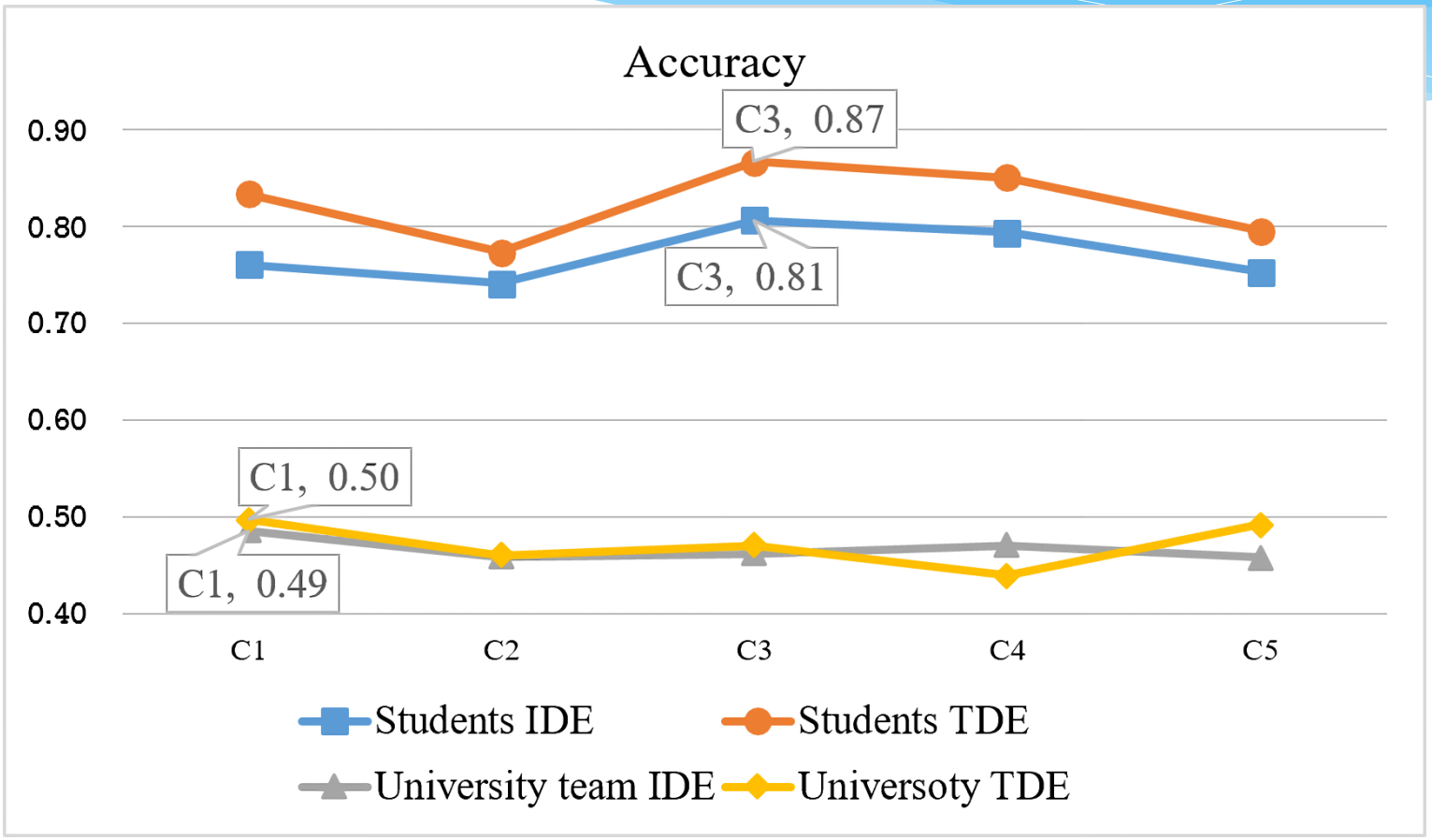
Individual difference experiment



Total difference experiment

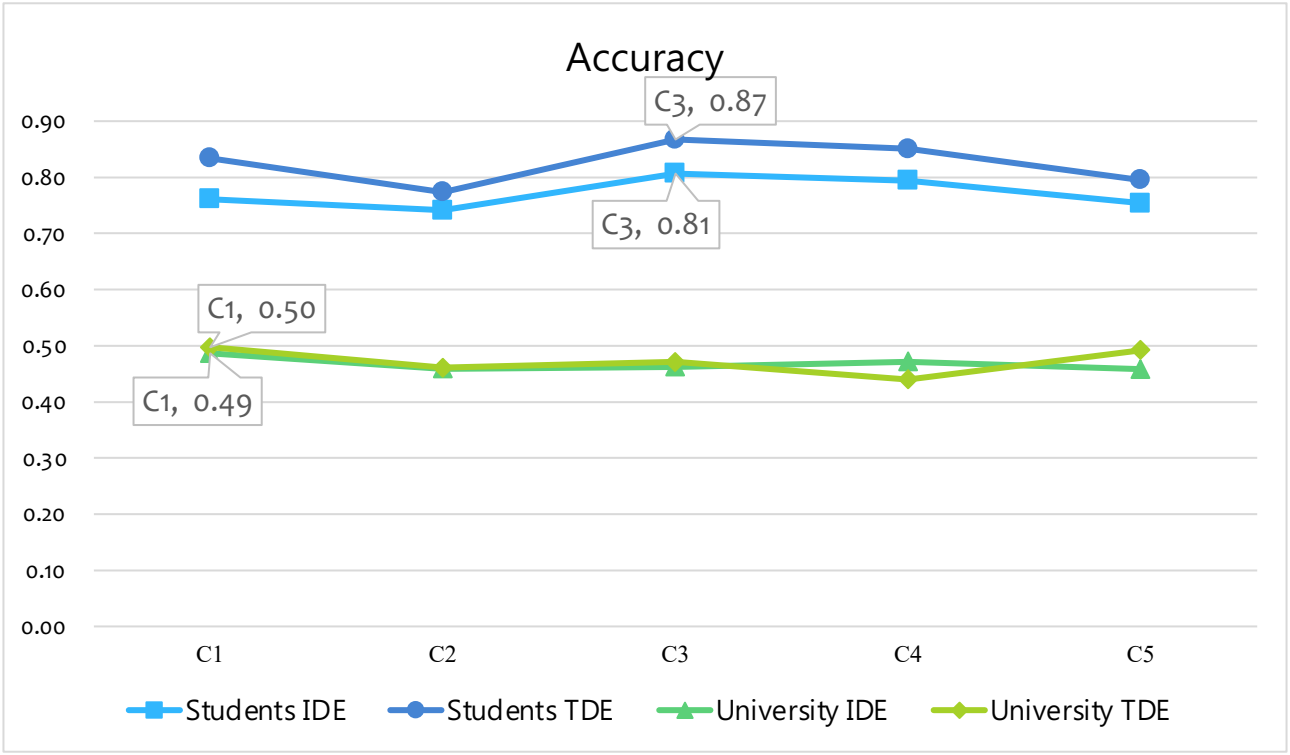


Experimental results of 4 experiments on 5 curve similarity algorithms



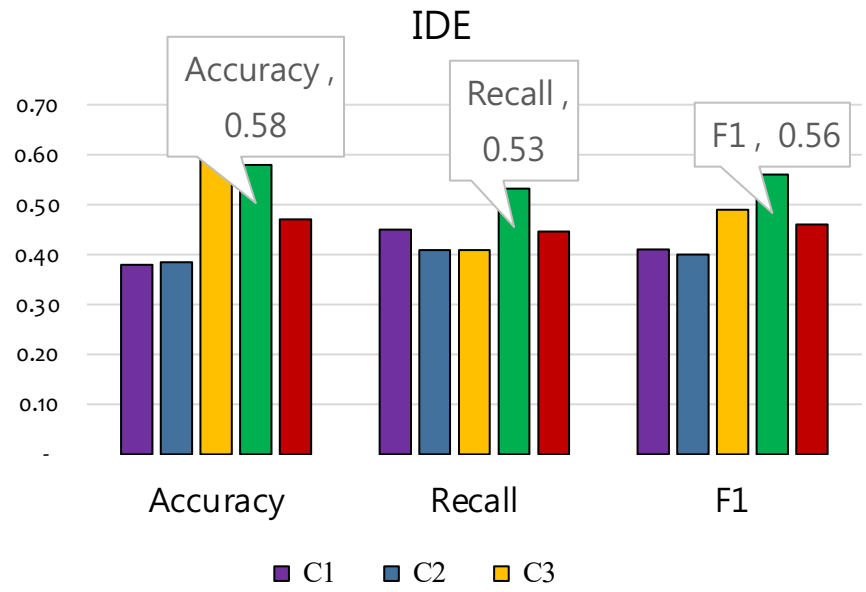
Accuracy

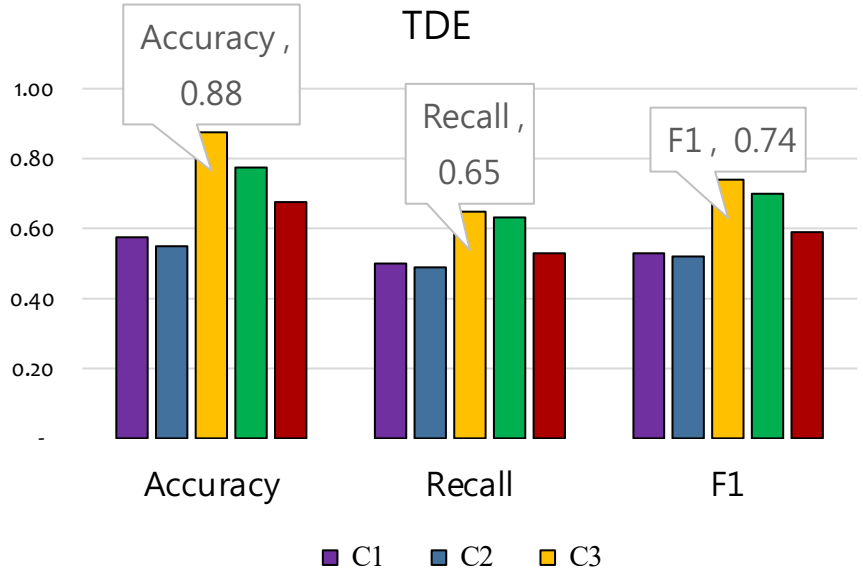
C1	C2	C3	C4	C5
Partial Curve Mapping	Area method	Discrete Frechet distance	Curve Length	Dynamic Time Warping, DTW



Basketball Class Players

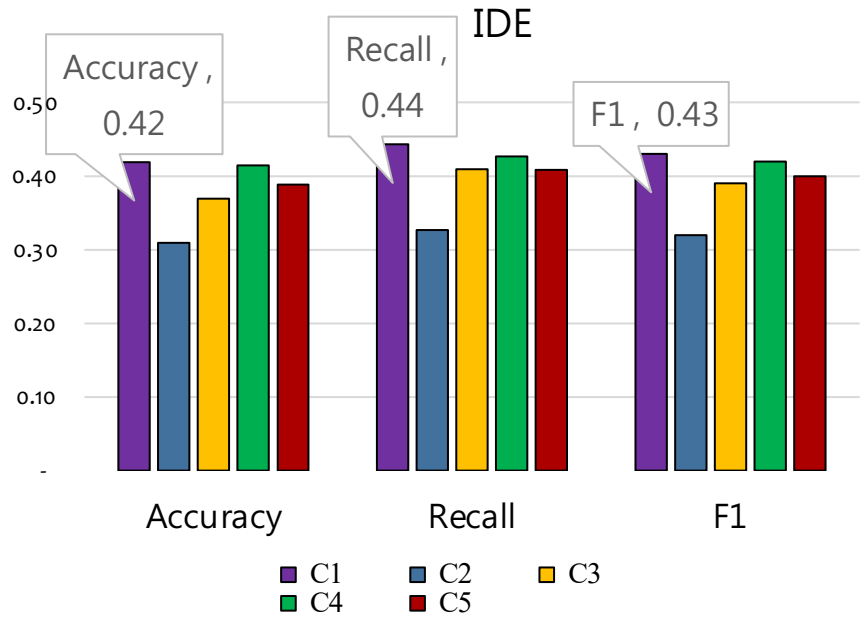
C1	C2	C3	C4	C5
Partial Curve Mapping	Area method	Discrete Frechet distance	Curve Length	Dynamic Time Warping, DTW

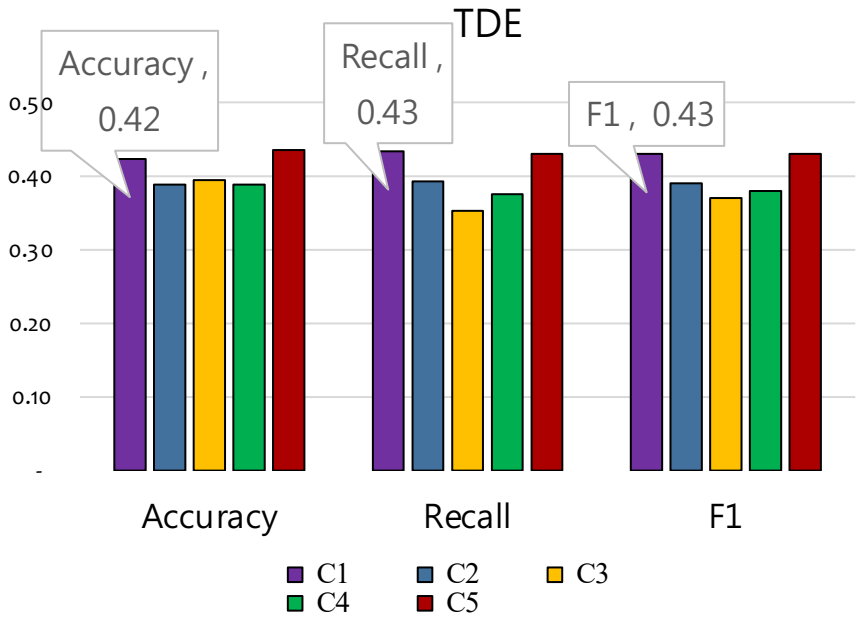




University Team Players

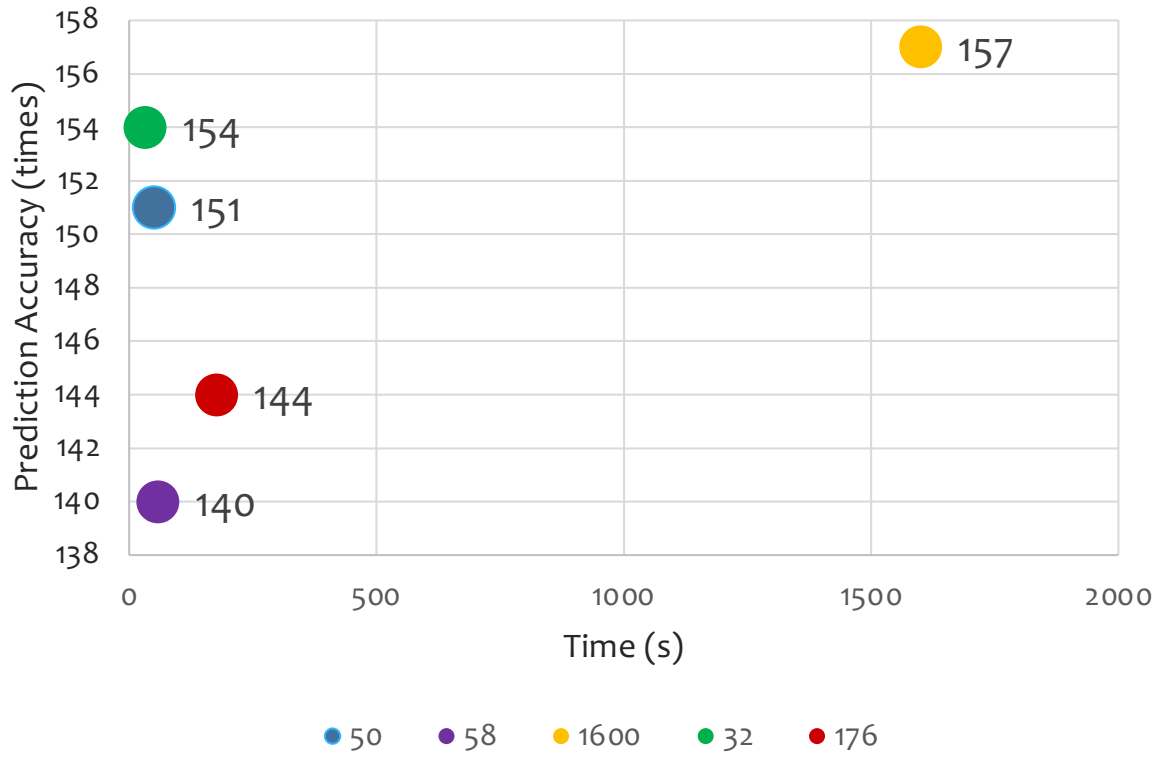
C1	C2	C3	C4	C5
Partial Curve Mapping	Area method	Discrete Frechet distance	Curve Length	Dynamic Time Warping, DTW



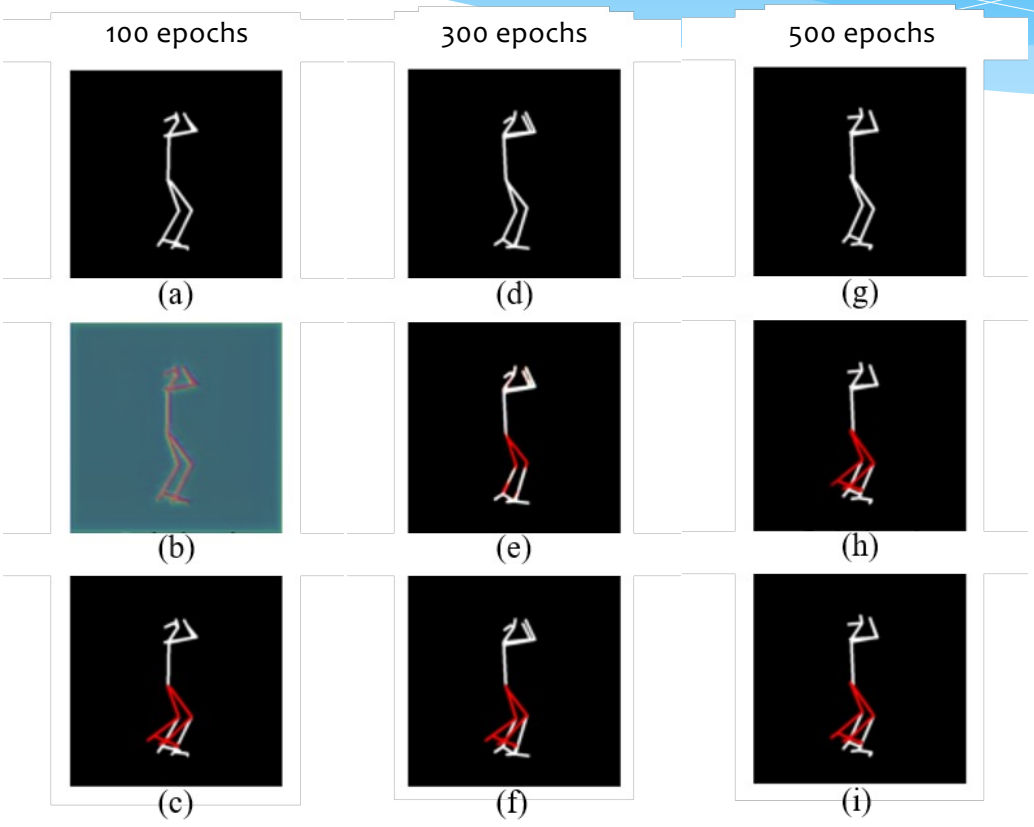


Prediction Accuracy v.s. Computation Time

C1	C2	C3	C4	C5
Partial Curve Mapping	Area method	Discrete Frechet distance	Curve Length	Dynamic Time Warping, DTW



Comparison of three trainings by cGAN

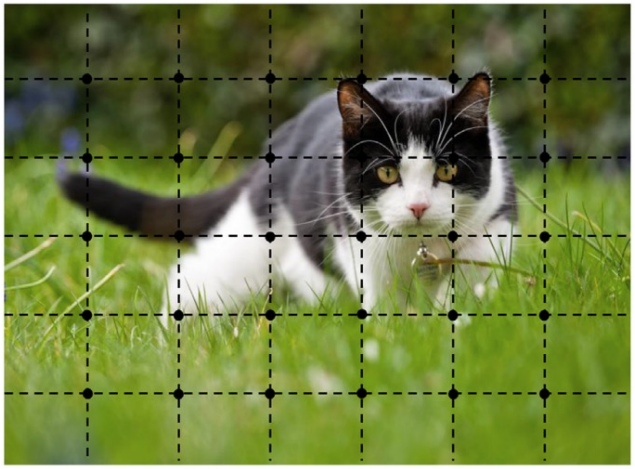


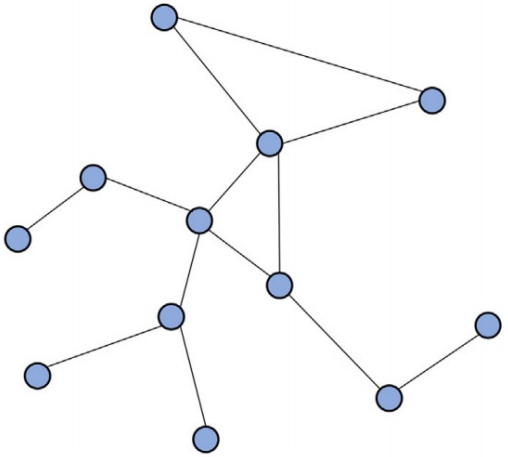
Conclusion

* estimate shooting results through the OpenPose system
* convert the posture of the shooting process into a trajectory curves
* Using similarities of curves and KNN model to effectively predict the shooting results
* Use pix2pix cGAN model to develop a shooting posture suggestion system
* Basketball Class Players have higher prediction rate than University Team Players

Athletes‘ Movements Analysis by Graph Neural Network

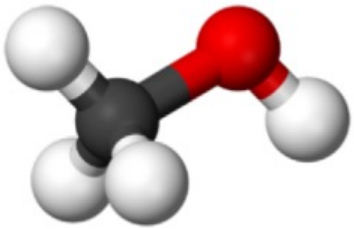
Graph Neural Network

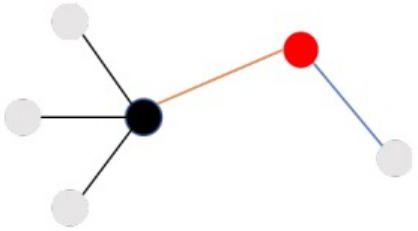


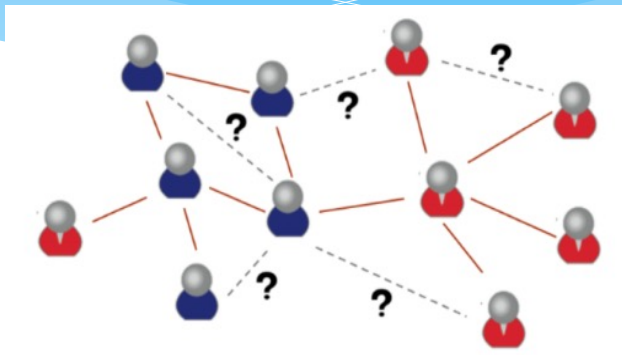


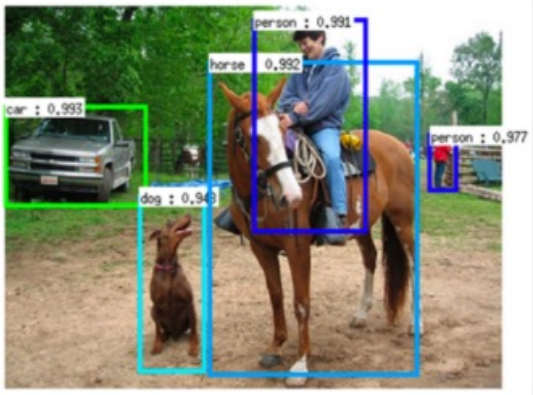
https://arxiv.org/ftp/arxiv/papers/1812/1812.08434.pdf

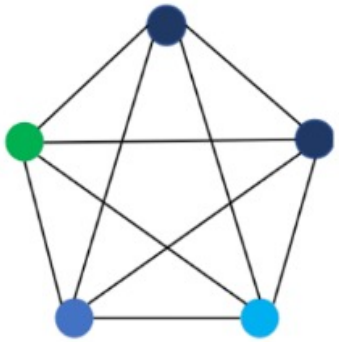
Graph Neural Network





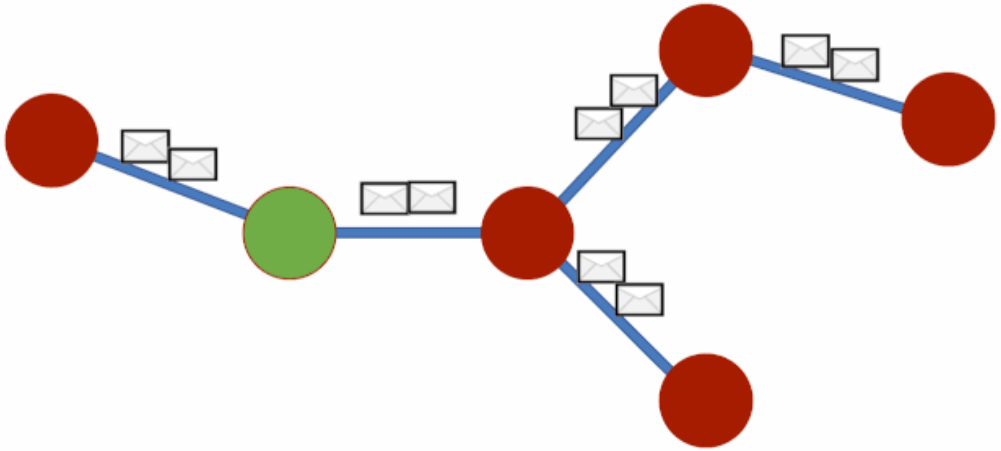






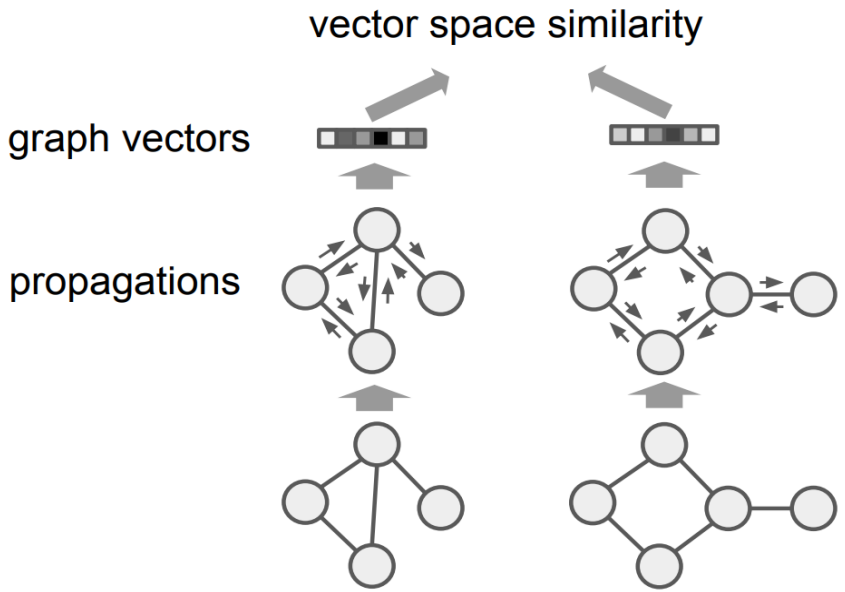
https://arxiv.org/ftp/arxiv/papers/1812/1812.08434.pdf

Graph Neural Network



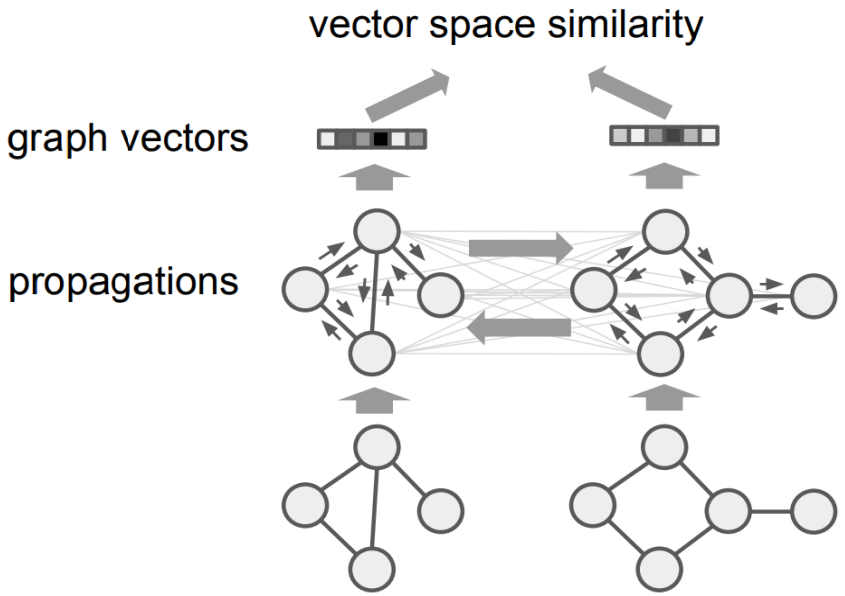
https://www.outsystems.com/blog/posts/graph-neural-networks/

Graph Embedding Model



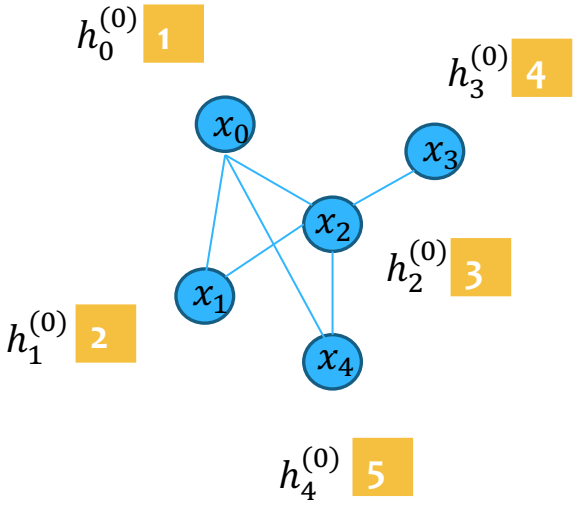
https://arxiv.org/pdf/1904.12787.pdf

Graph Matching Network

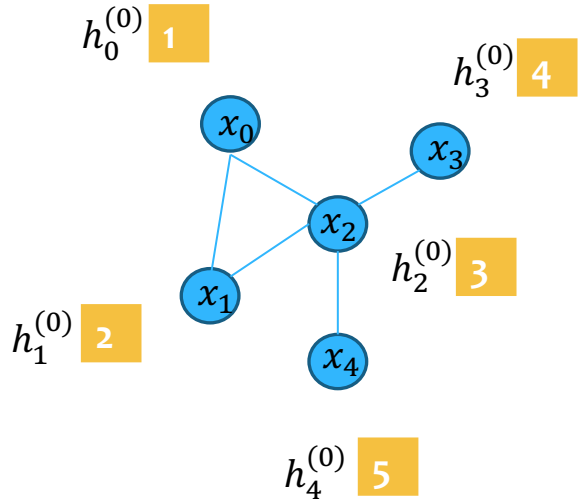


https://arxiv.org/pdf/1904.12787.pdf

Two Graphs Examples



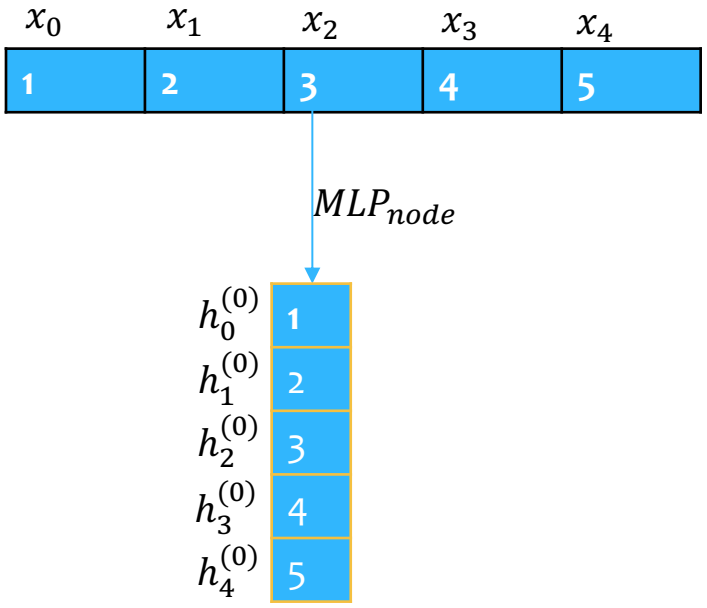
G1

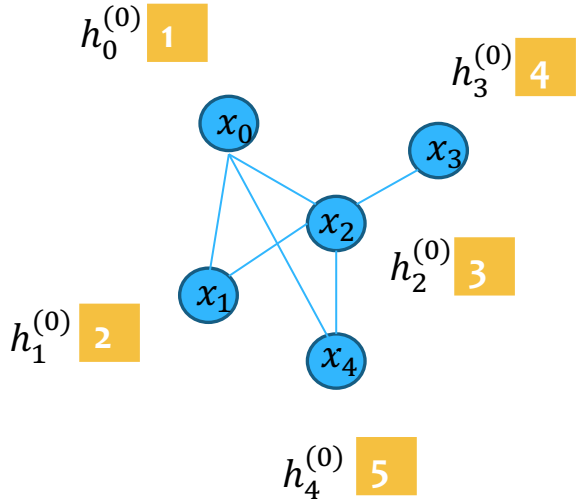


G2

G1 encoder layer

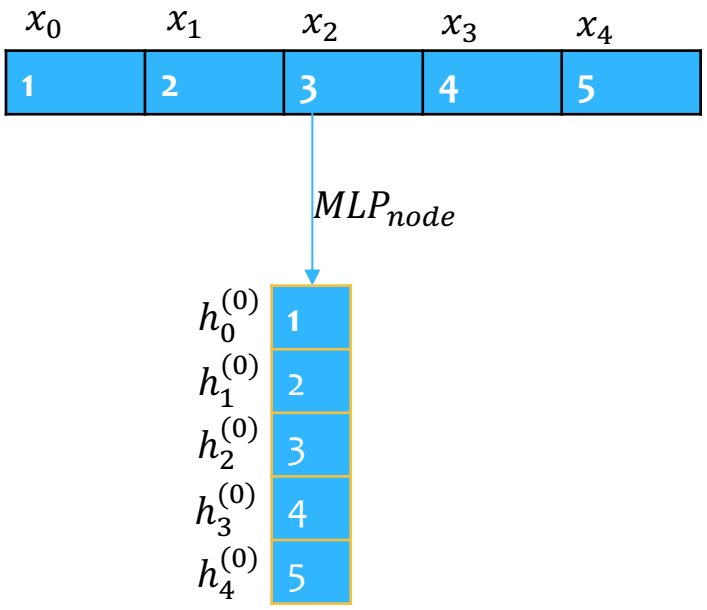
$$h_i^{(0)} = MLP_{node}(x_i), \quad \forall i \in V$$
$$e_{ij} = MLP_{edge}(x_{ij}), \quad \forall (i,j) \in E$$

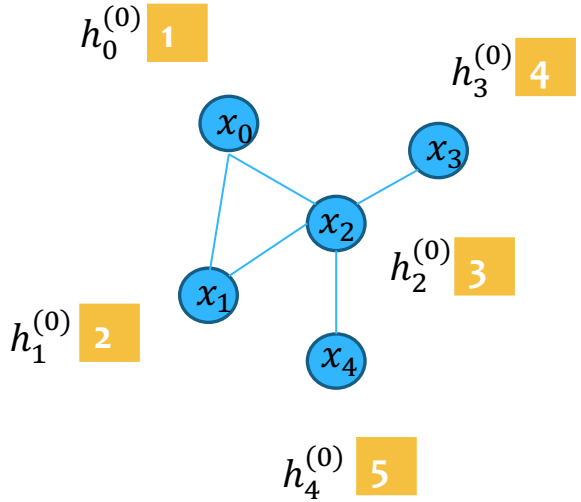




G2 encoder layer

$$h_i^{(0)} = MLP_{node}(x_i), \quad \forall i \in V$$
$$e_{ij} = MLP_{edge}(x_{ij}), \quad \forall (i,j) \in E$$





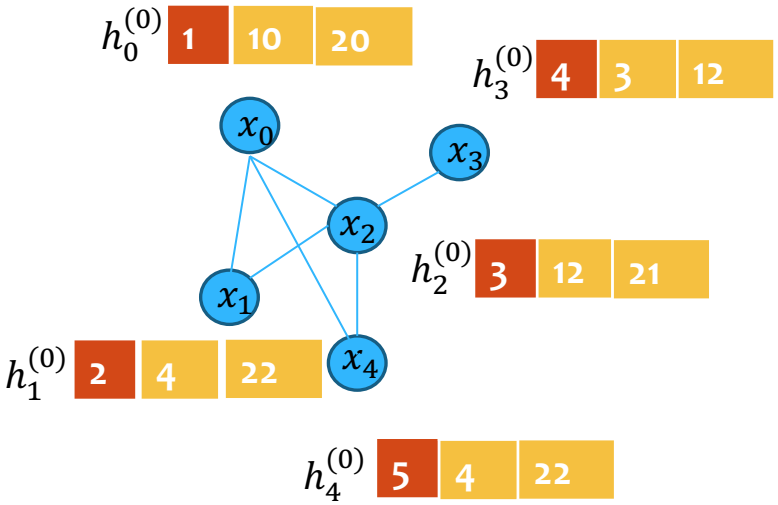
G1 propagation layer

$$m_{j\to i} = f_{message}\left(h_i^{(t)}, h_j^{(t)}, e_{ij}\right)$$

$$h_i^{(t+1)} = f_{node}\left(h_i^{(t)}, \sum_{j:(j,i)\in E} m_{j\to i}\right)$$

$h_0^{(0)}$	1	10	20
$h_1^{(0)}$	2	4	22
$h_2^{(0)}$	3	12	21
$h_3^{(0)}$	4	3	12
$h_4^{(0)}$	5	4	22

10 is starting weight

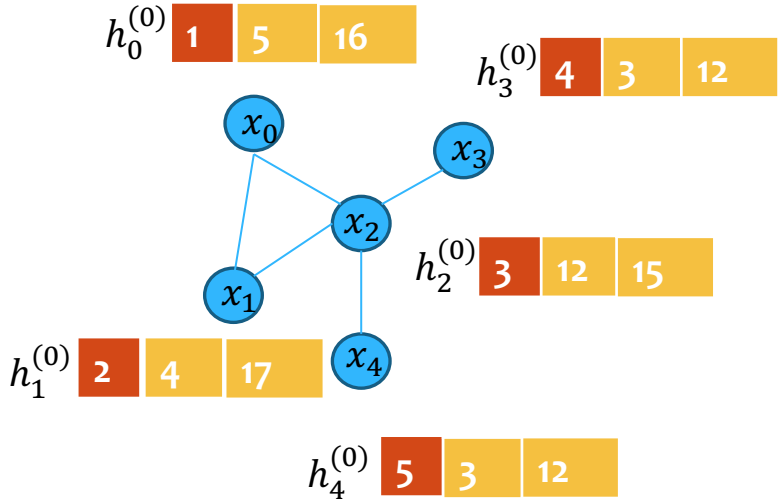


G2 propagation layer

$$m_{j\to i} = f_{message}\left(h_i^{(t)}, h_j^{(t)}, e_{ij}\right)$$

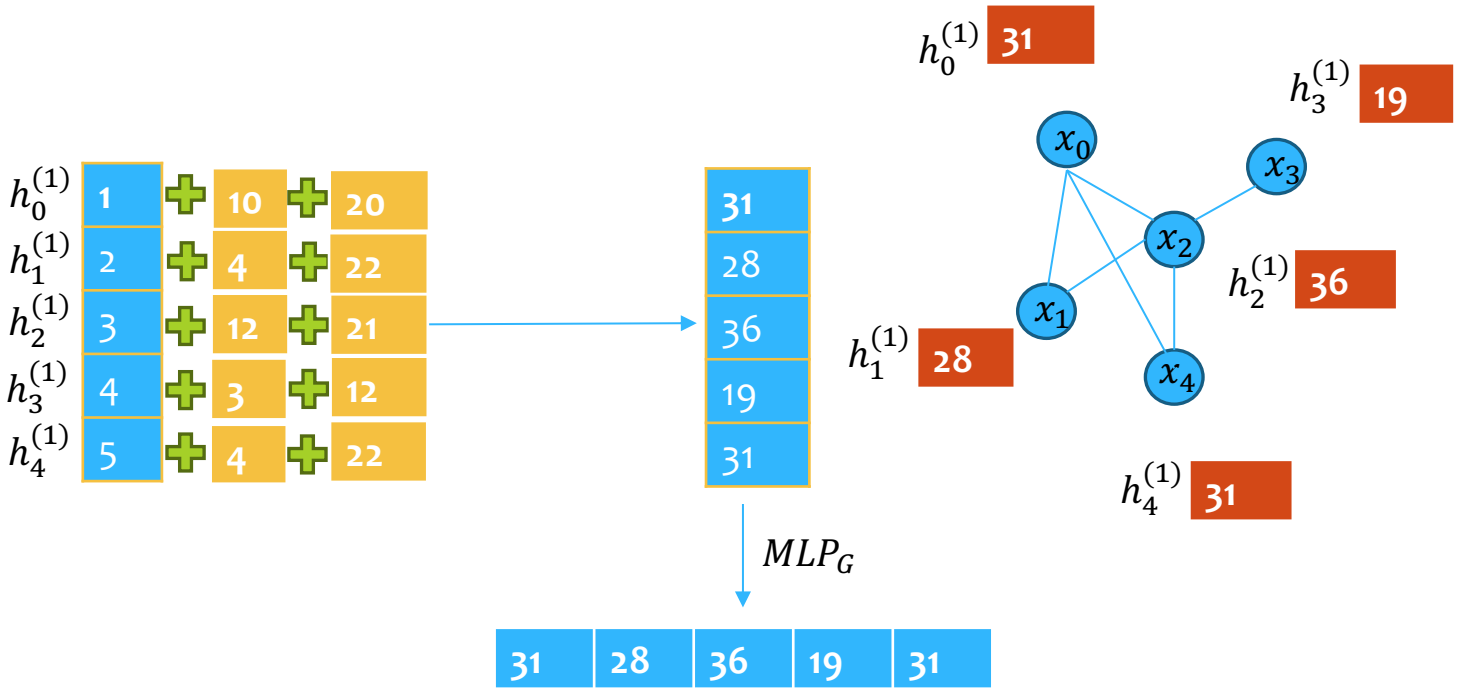
$$h_i^{(t+1)} = f_{node}\left(h_i^{(t)}, \sum_{j:(j,i)\in E} m_{j\to i}\right)$$

$h_0^{(0)}$	1	5	16
$h_1^{(0)}$	2	4	17
$h_2^{(0)}$	3	12	15
$h_3^{(0)}$	4	3	12
$h_4^{(0)}$	5	3	12



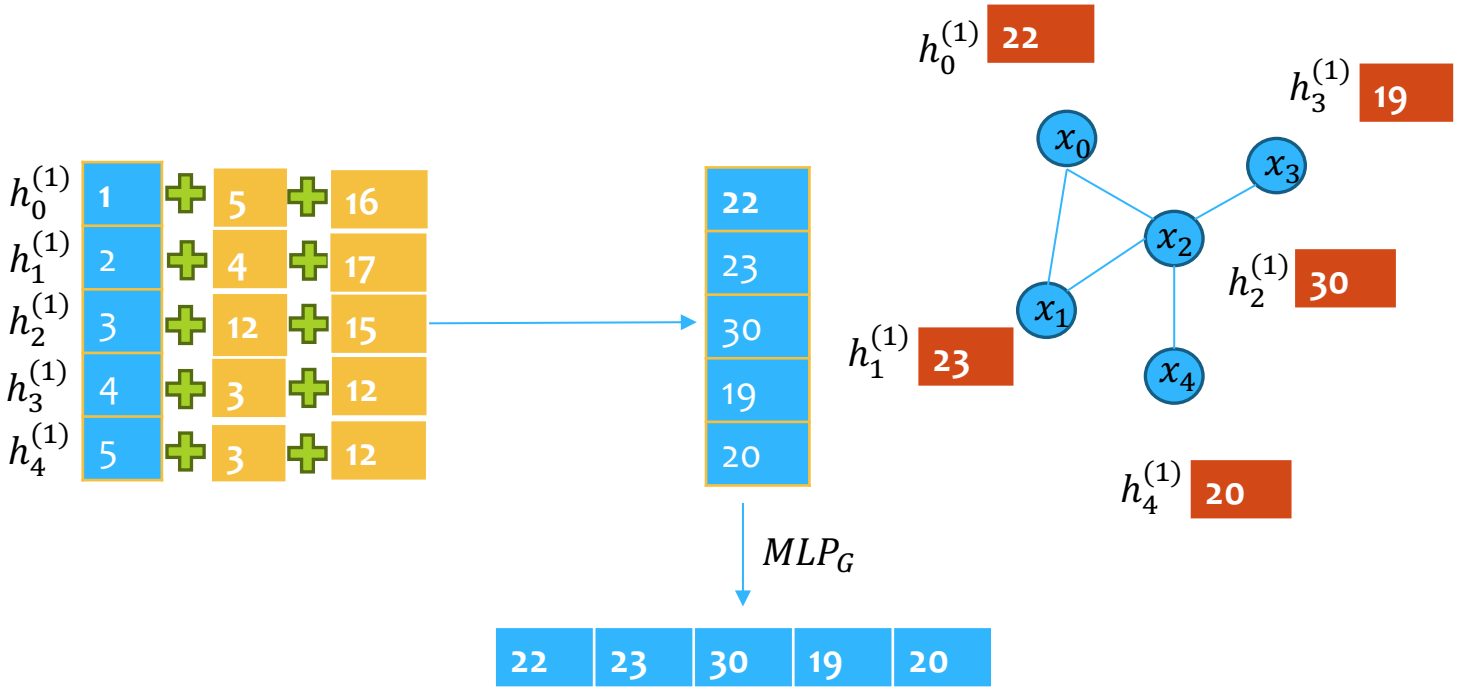
G1 Aggregator layer

$$h_G = MLP_G(\sum_{i \in V} h_i^{(T)})$$



G2 Aggregator layer

$$h_G = MLP_G(\sum_{i \in V} h_i^{(T)})$$



Similarity

G1	31	28	36	19	31

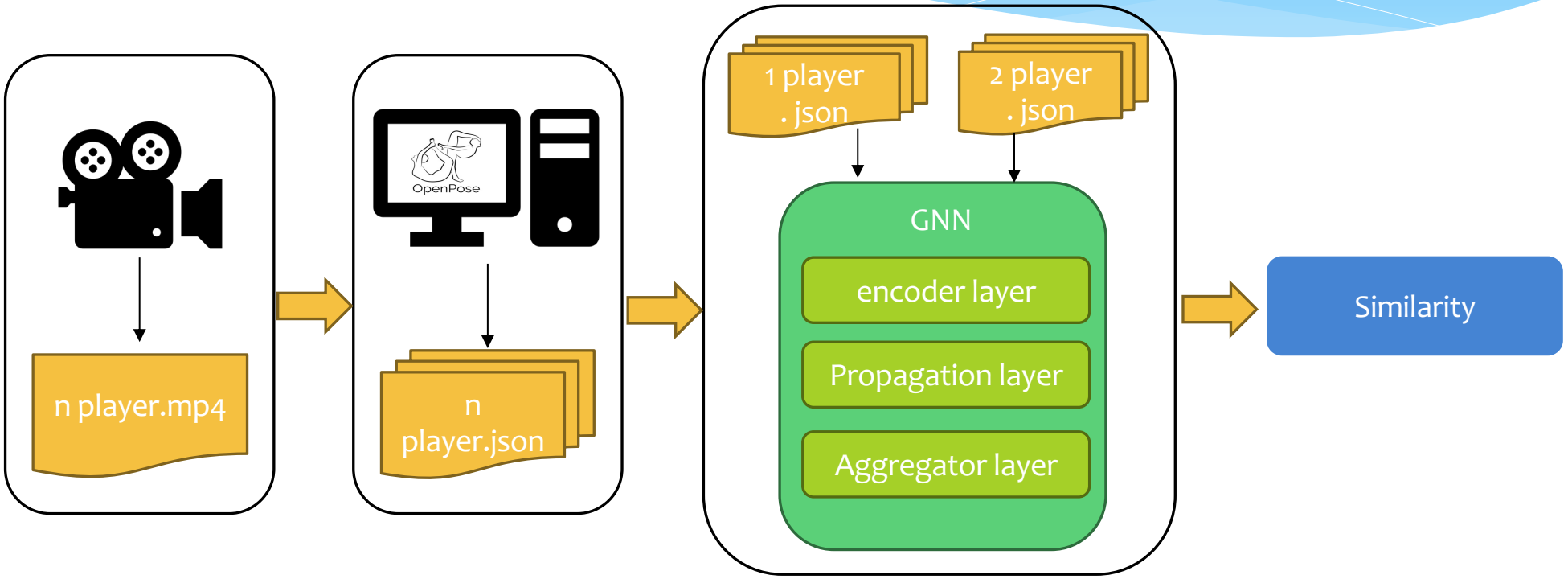
G2	22	23	30	19	20

$(31-22)^2 \quad +(28-23)^2 \quad +(36-30)^2 \quad +(19-19)^2 \quad +(31-20)^2$

181+25+36+0+121=363(distance)

Similarity between G1 and G2 = -363

Processing Procedure



Methodologies

1D Graph Embedding Model

1D Graph Matching Model

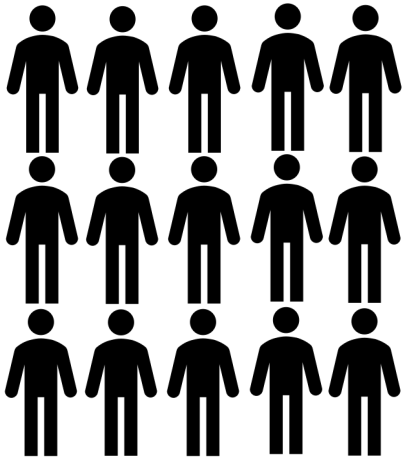
1D
average of x-axis similarity and y-axis similarity

2D Graph Embedding Model

2D Graph Matching Model

2D
similarity of x-axis and y-axis

Dataset

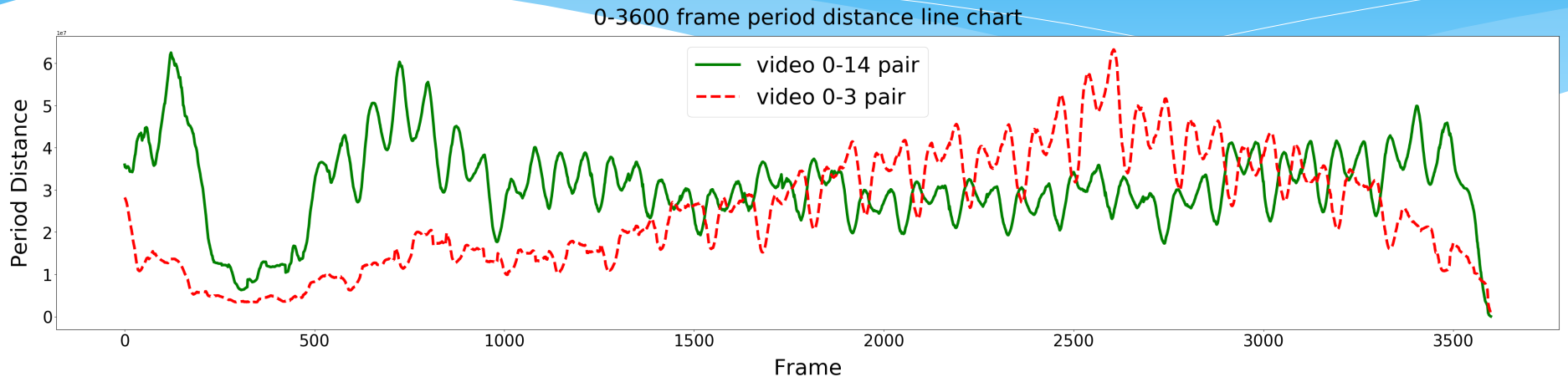


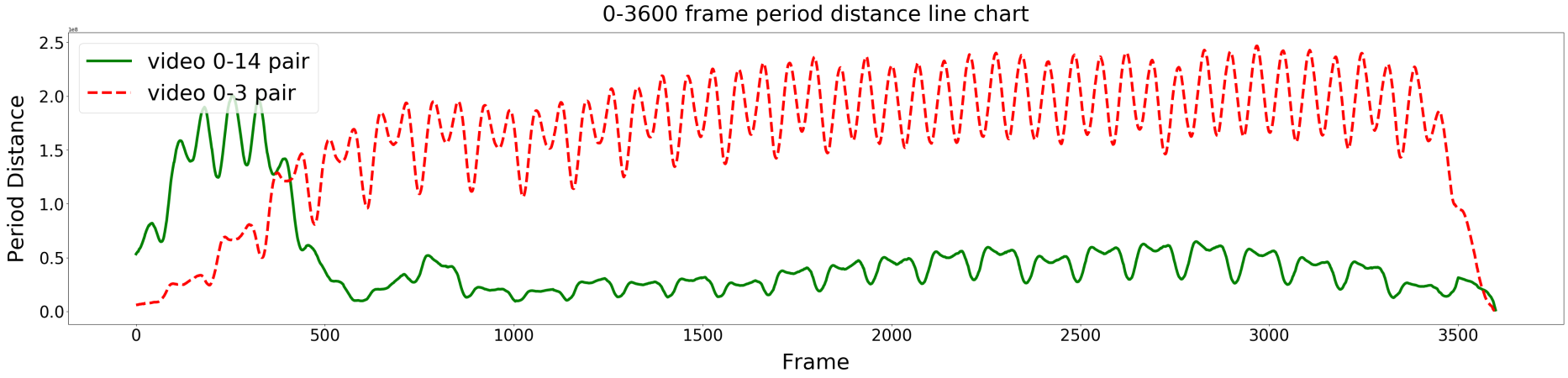
15 players
Each one rowing one minute

one rowing machine

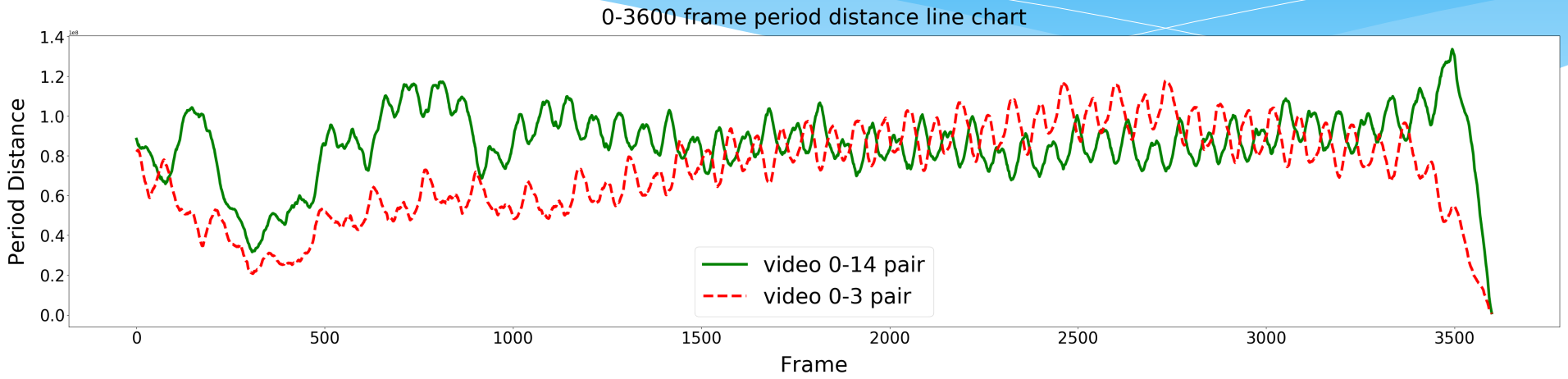
Dataset

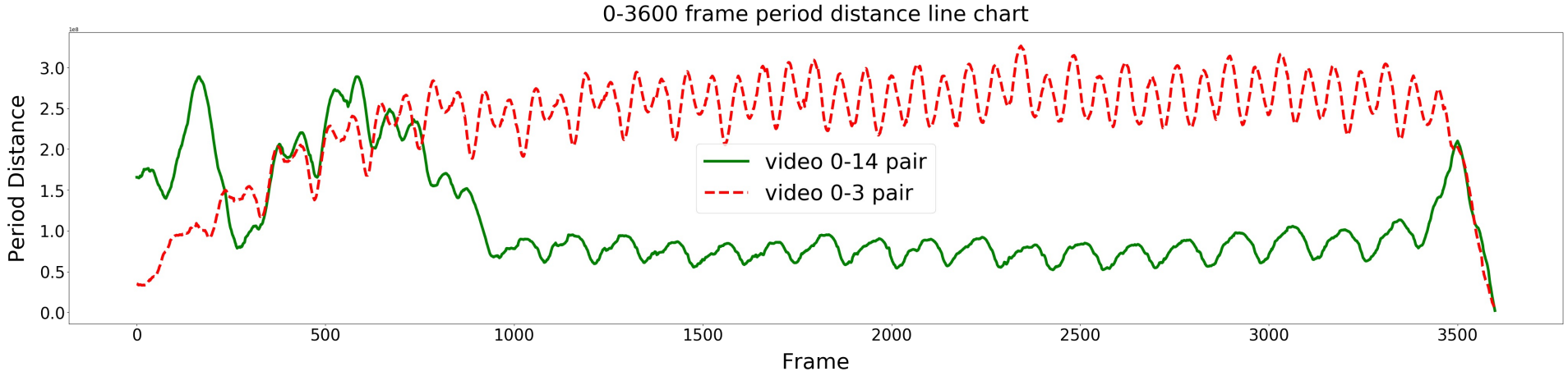
Results of Graph Embedding Models





Results of Graph Matching Models

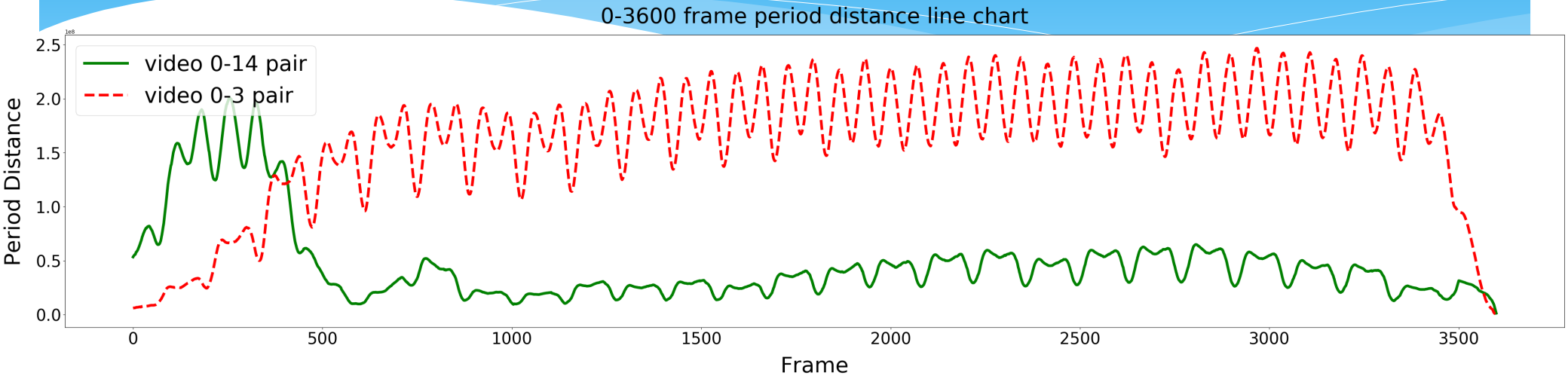


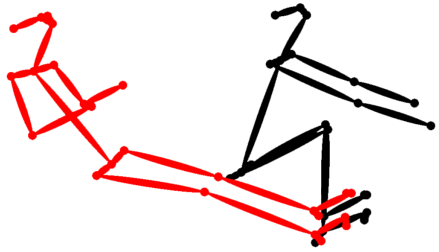


Model Selection

* 2D Embedding Model performs better

Verification players of 0,3 and 0,14





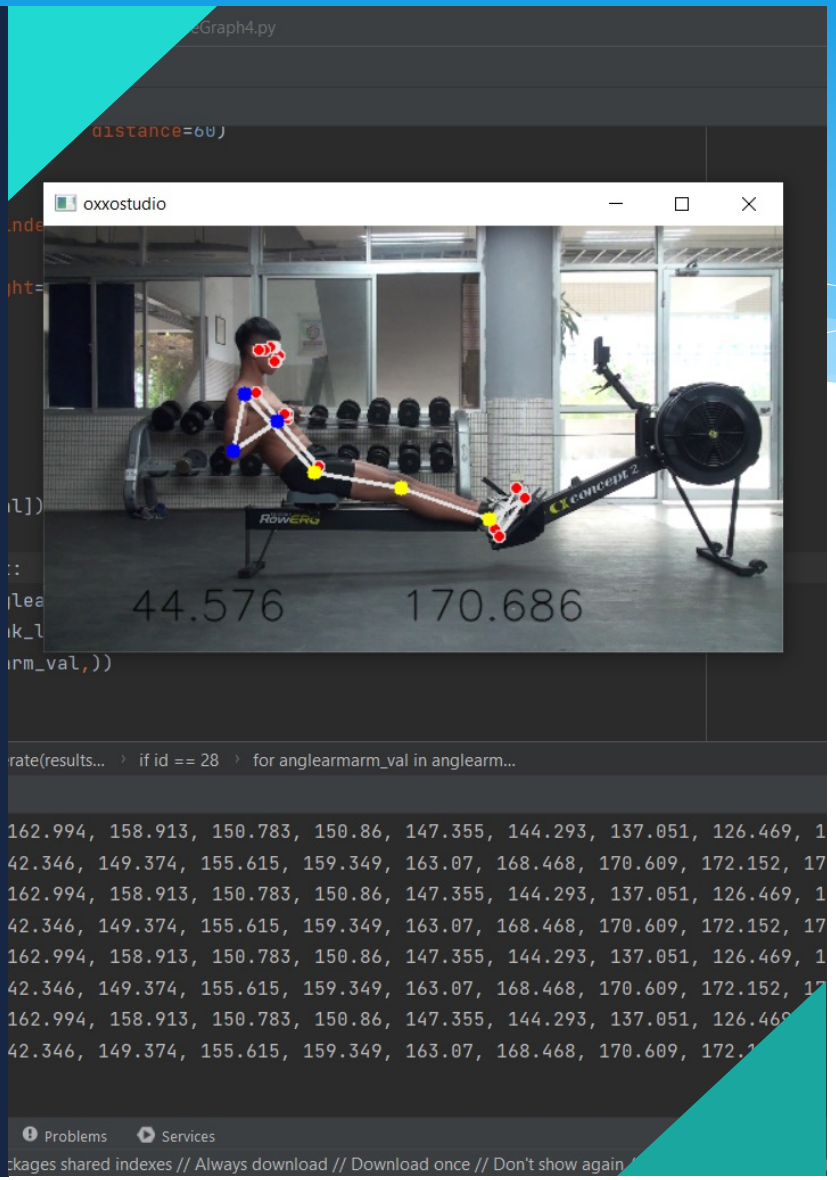
Conclusion

* 2D v.s. 1D
* 2D Graph Embedding Model v.s. 2D Graph Matching Model
* Skeleton change shows good performance of the proposed method

Automatic rowing cycle detection

Thamonwan Sangsamrit
Department of Physics, Faculty of Science,
Kasetsart University, Thailand

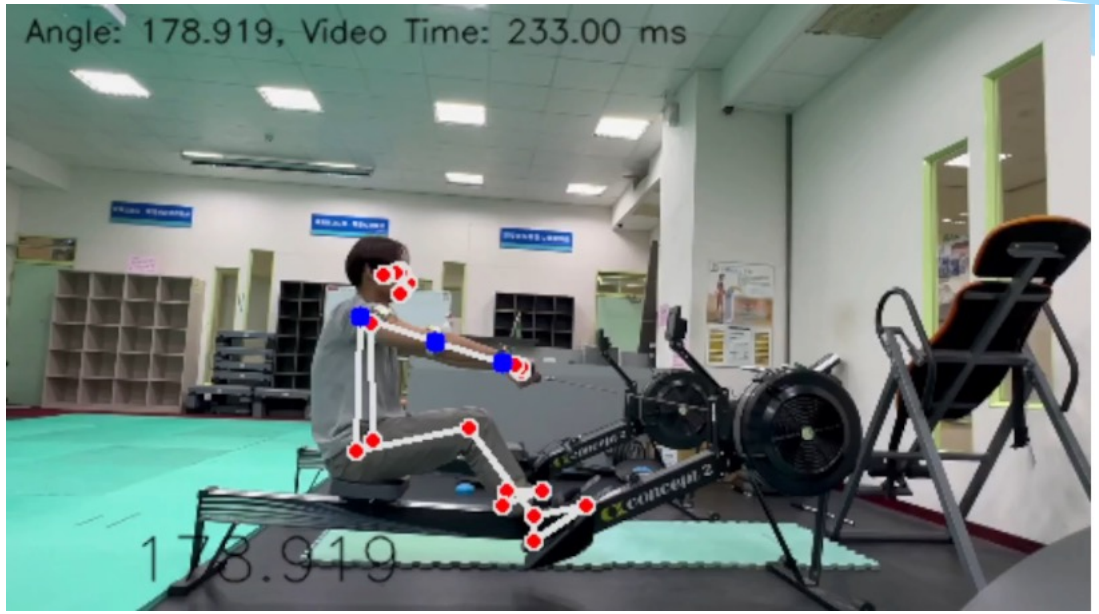
About the Machine Learning



Apply computer vision techniques to estimate the angles between the upper arm and lower arm, and the upper leg and lower leg extracted by the MediaPipe Pose model and OpenCV library.

Workflow

Computer vision



This involves writing code

Calculates the vectors

Calculates the angle and can be used to create a graph to count the number of rowing rounds.

Calculation of angles

Computer vision

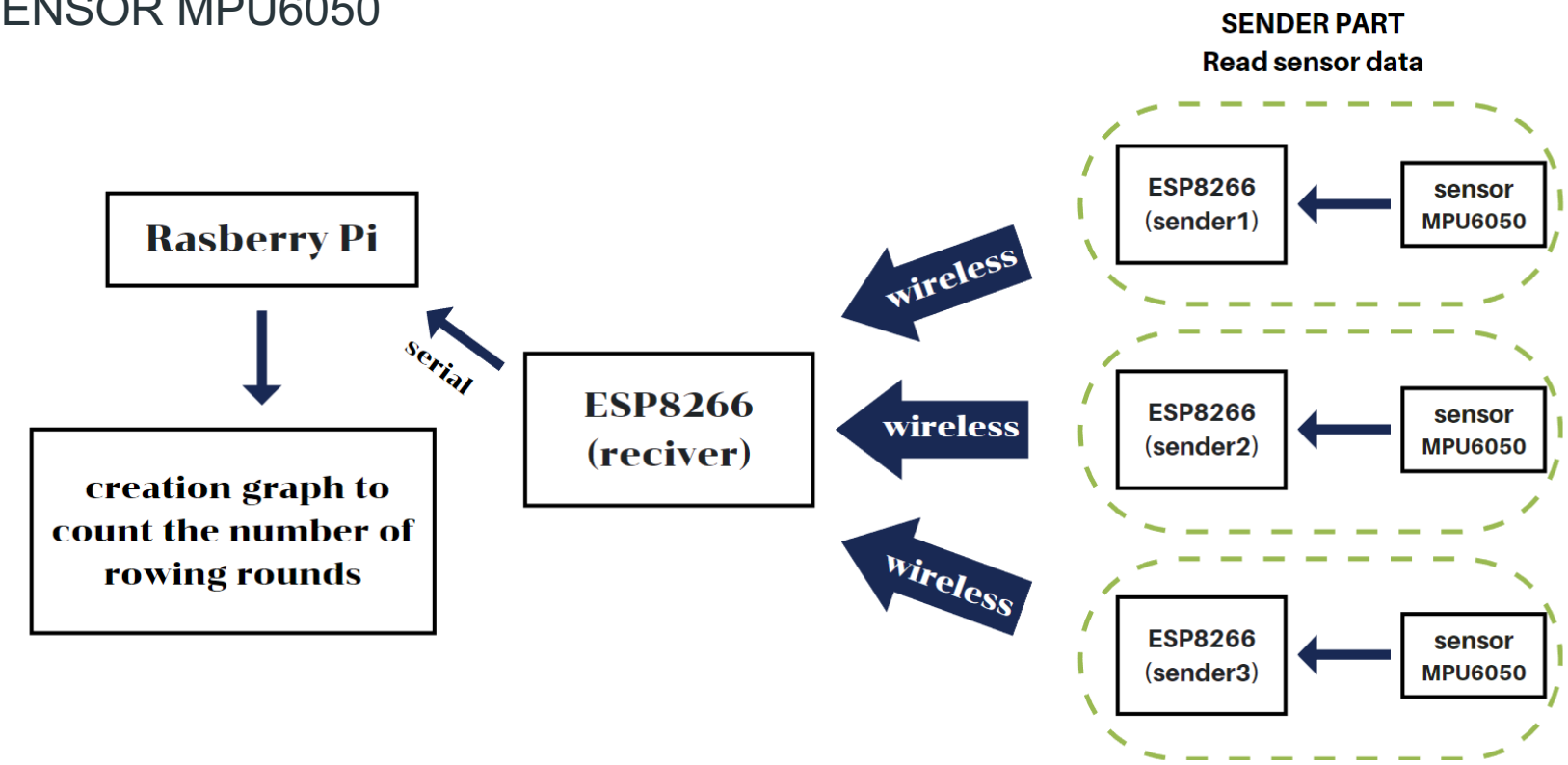
angle arm =
np.degrees(np.arccos(np.dot(vec1, vec2)/
(np.linalg.norm(vec1)*np.linalg.norm(vec2))))

About the Internet of Things

Sensor MPU6050 is a commonly used sensor module that combines a 3-axis gyroscope and a 3-axis accelerometer into a single package. It is often used in electronic projects and robotics to measure motion, orientation, speed, and acceleration.

Paragraph Workflow

SENSOR MPU6050



Paragraph Workflow

SENSOR MPU6050

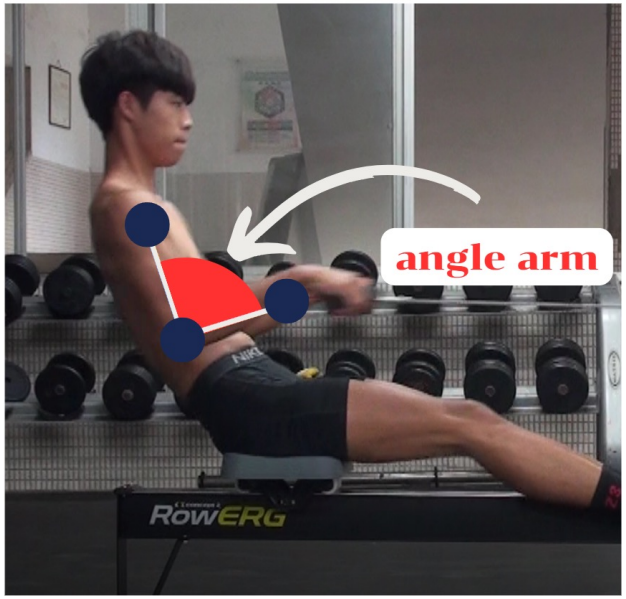
Sensor mpu 6050
read sensor data

Measure the acceleration and velocity values as numerical data in each axis

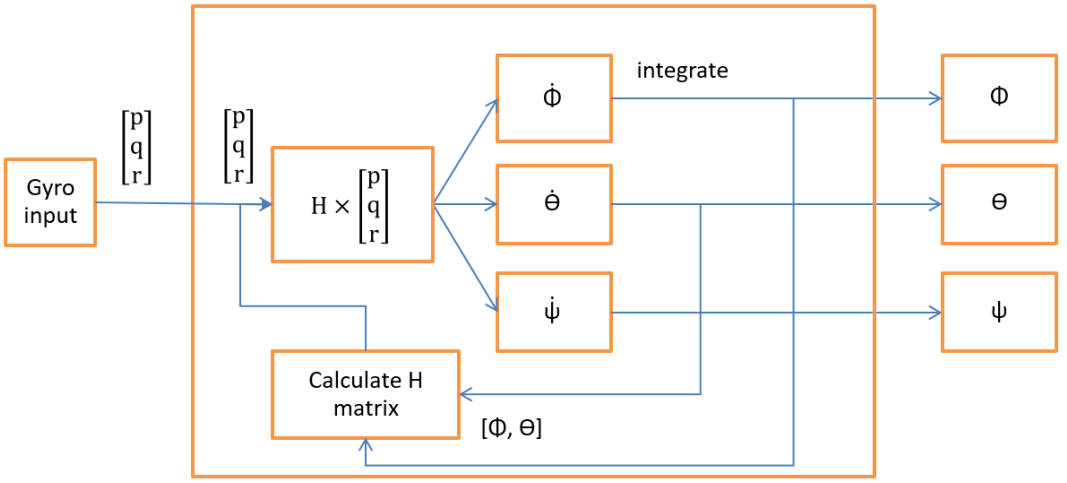
calculate the angles

Calculation of angles

SENSOR MPU6050



● = sensor



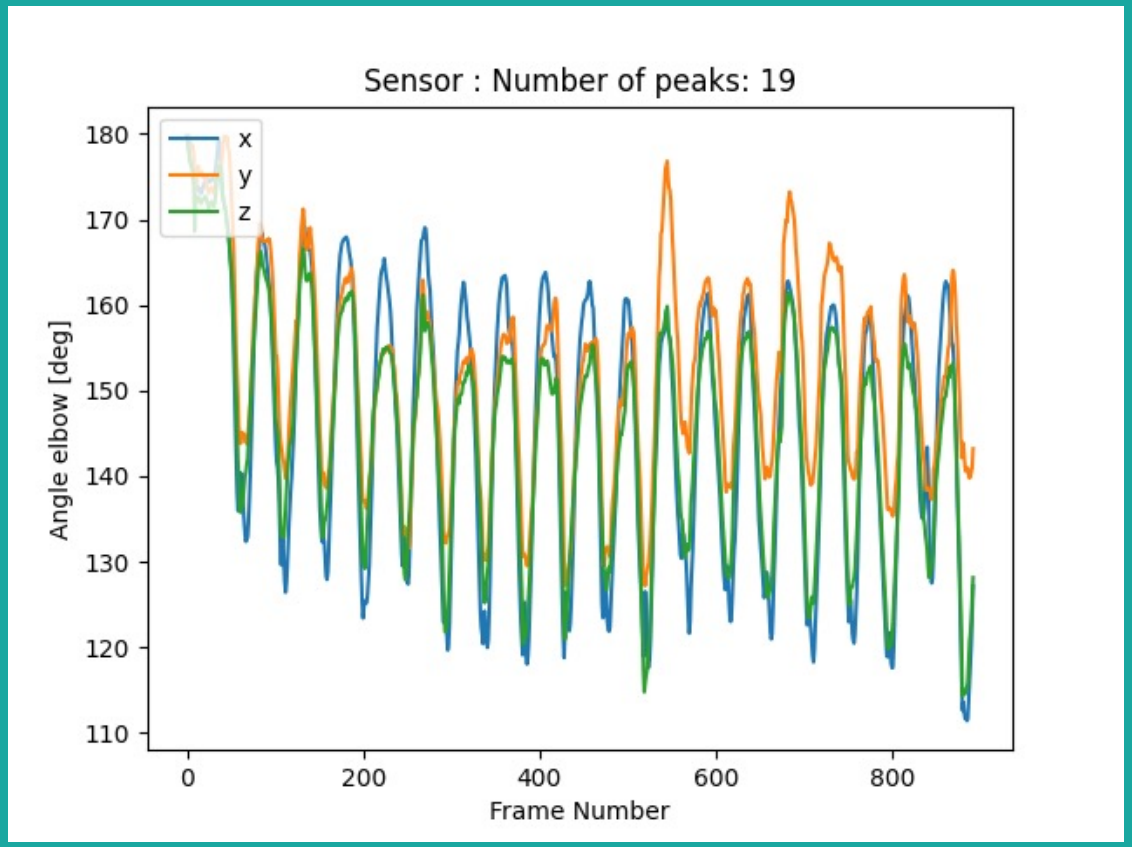
Calculate Euler Angles from Gyro

angle arm =

np.degrees(np.arccos(np.dot(vec1, vec2)/
(np.linalg.norm(vec1)*np.linalg.norm(vec2))))

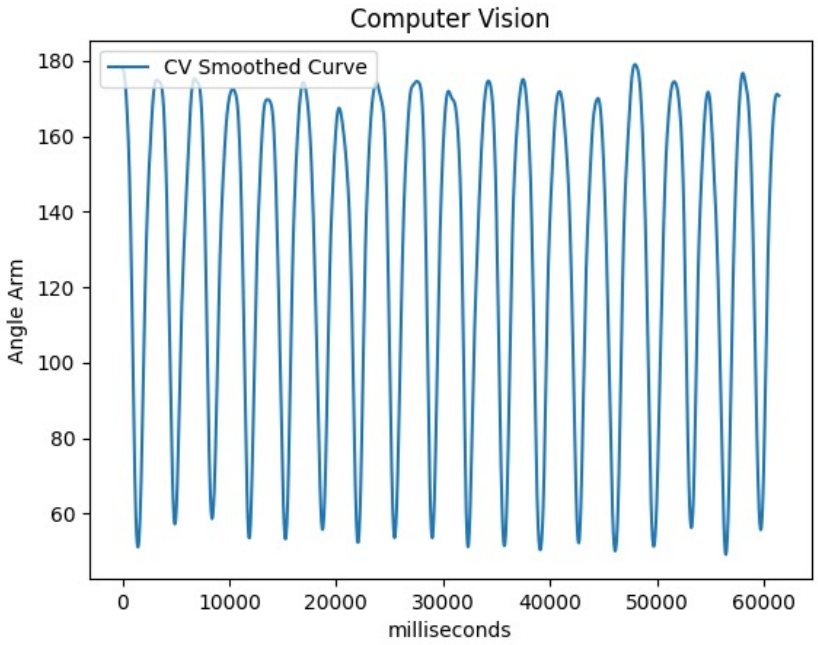
Problem

The MPU6050 sensor to calculate the vector use the acceleration and velocity data in the x and y axis to calculate the angles. The resulting angles would depend on the speed and acceleration of the motion.

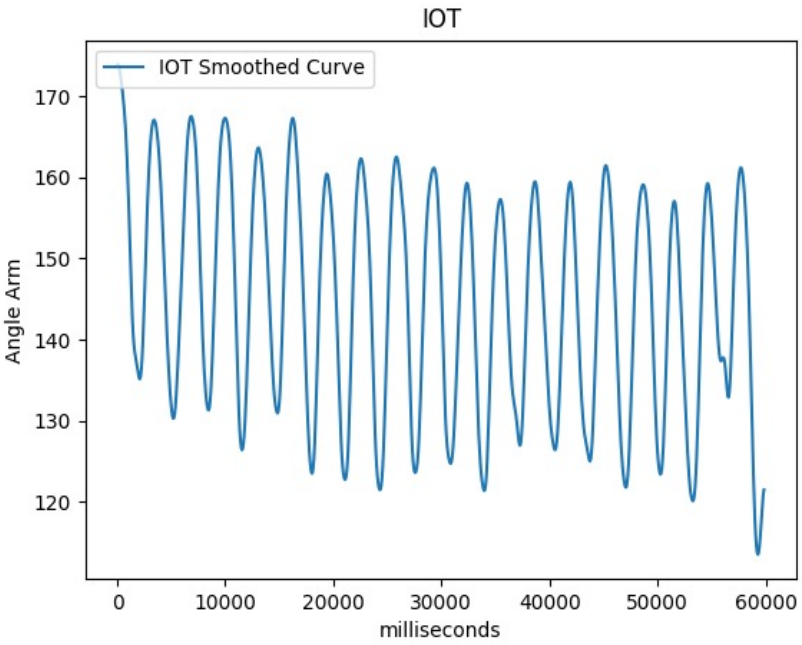


Machine Learning AND Internet of Things

computer vision between sensor mpu6050

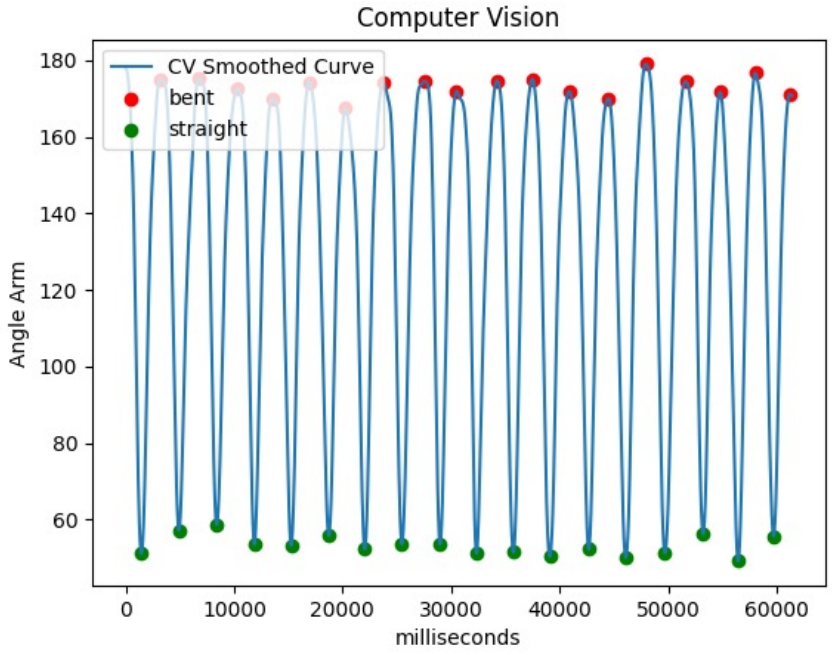


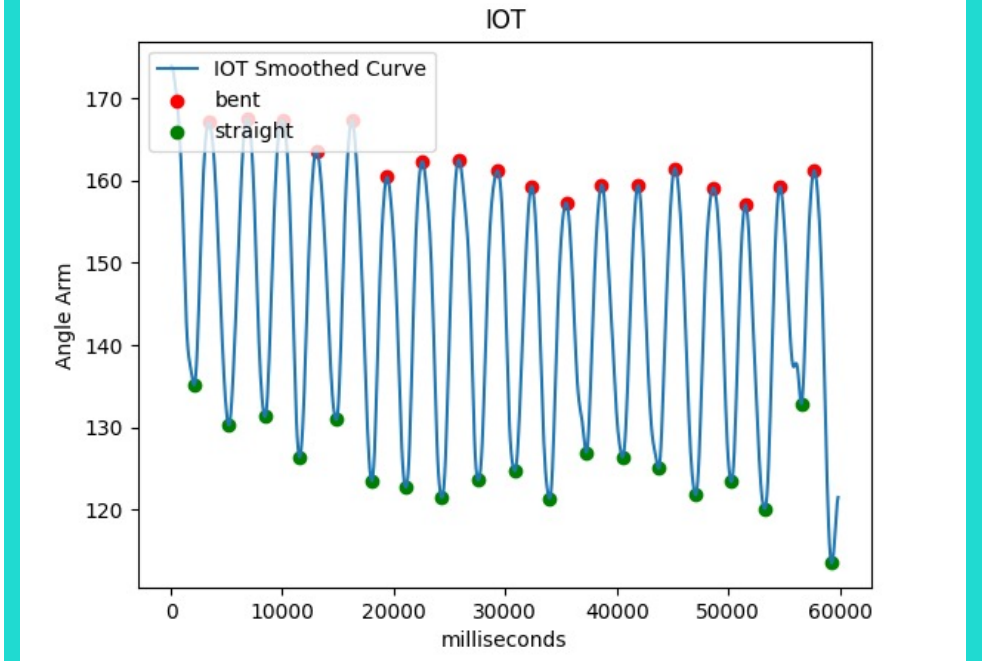
computer vision



sensor mpu6050

“ Straight ” or “ bent ”

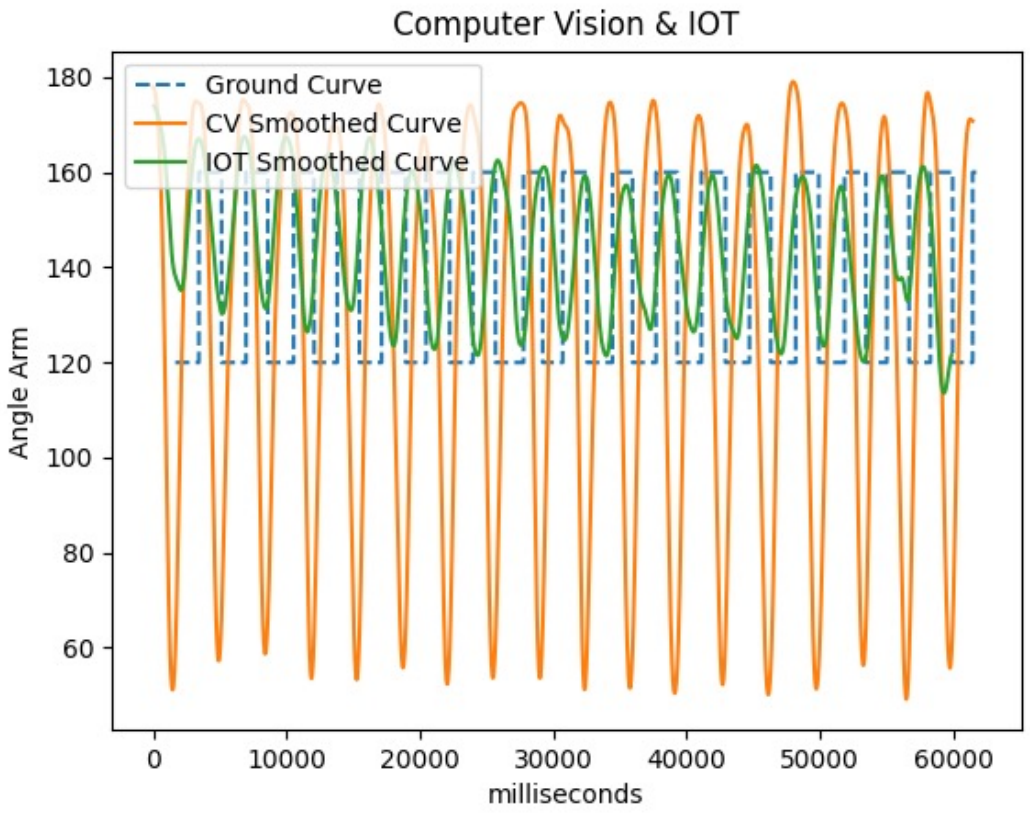




Graph between CV and IOT

Computer vision

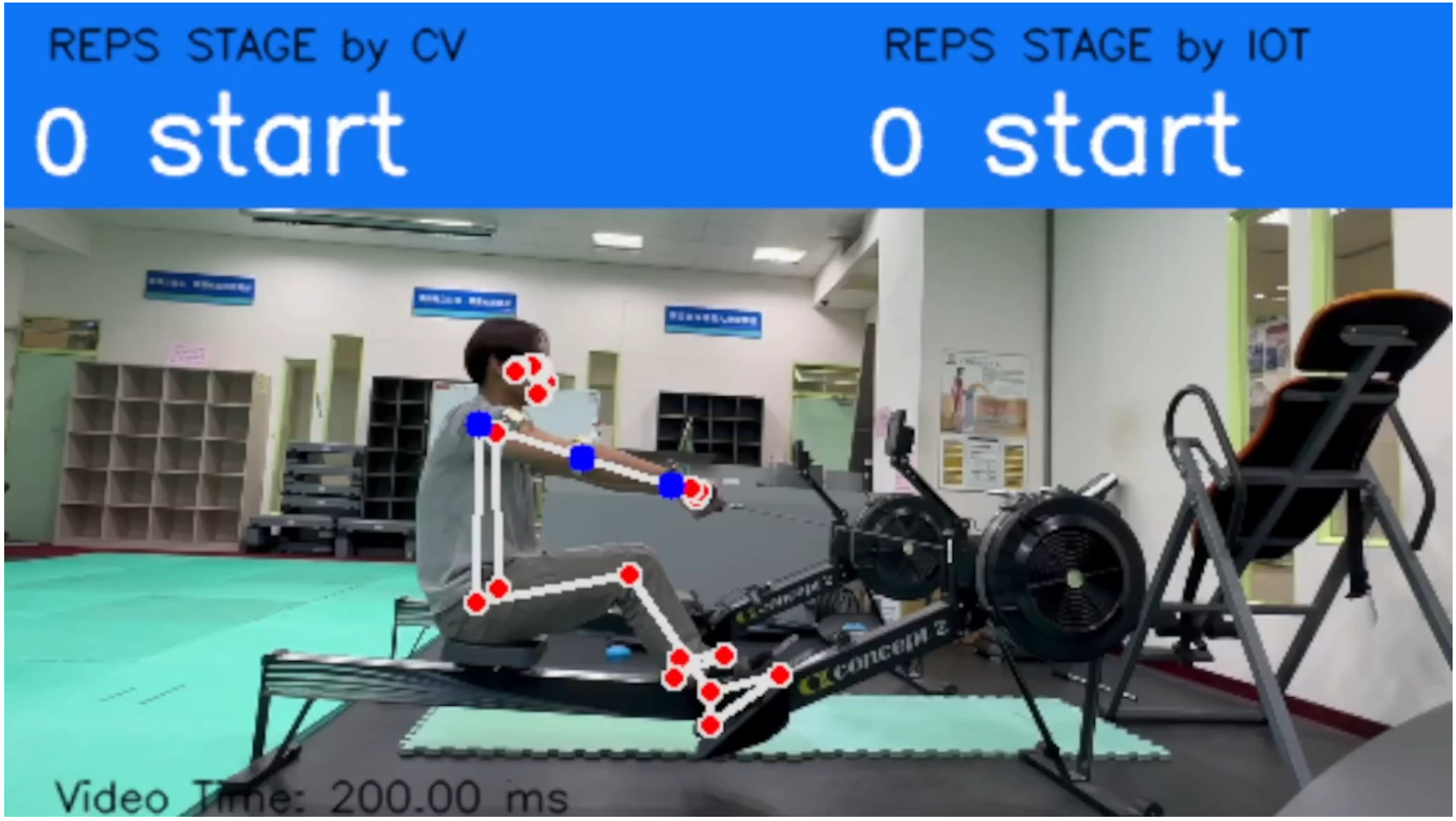
1 frame number
/
33 milliseconds



Internet of Things

1 frame number /
70 milliseconds

Graph showing the relationship from IOT data between arm angle (degree) and time (milliseconds)


REPS STAGE by CV
0 start
REPS STAGE by IOT
0 start
Video Time: 200.00 ms

Our future plan

* Train automatically
* movement cycle + self-learning model

Chinese Calligraphy Air Writing

Mr. Nopparat Chuprayoon

Mr. Peerapat Srisarungkarn

Prof. Chien-Chang Chen

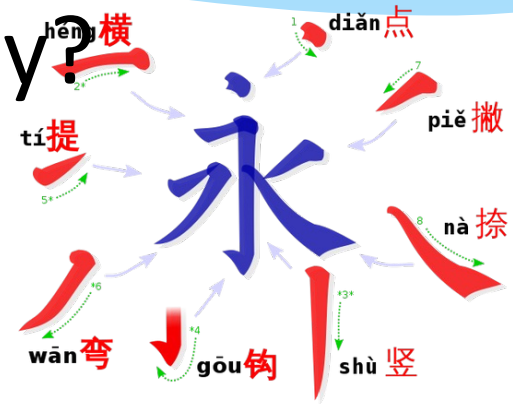
Faculty of Science at Sriracha Information Technology

Kasetsart University Sriracha Campus fourth-year student

Department of Computer Science and Information Engineering Tamkang University

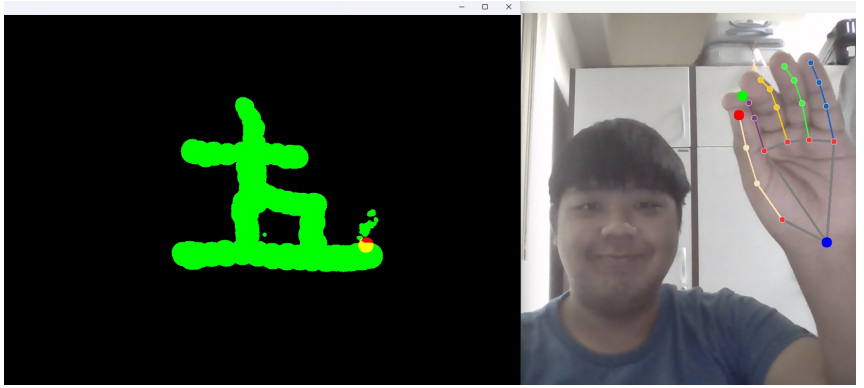
What is Chinese Calligraphy?

Chinese calligraphy is an art form that involves creating characters using a brush or other writing instrument.



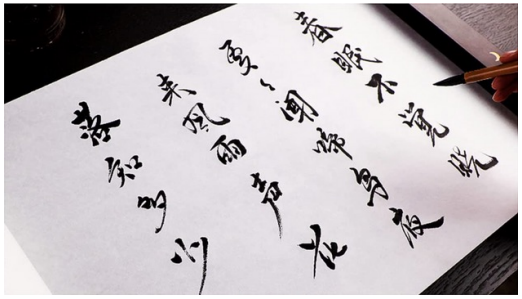
What is Air Writing?

Air writing is a technology that enables users to write or draw in the air using their hand or a device, which is then translated into digital data.

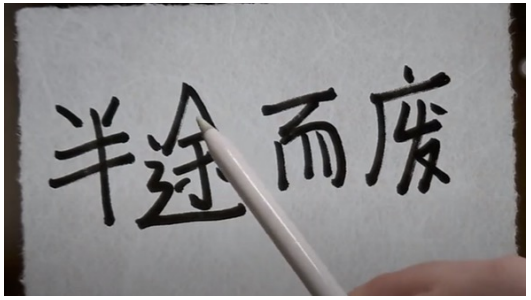


Problems

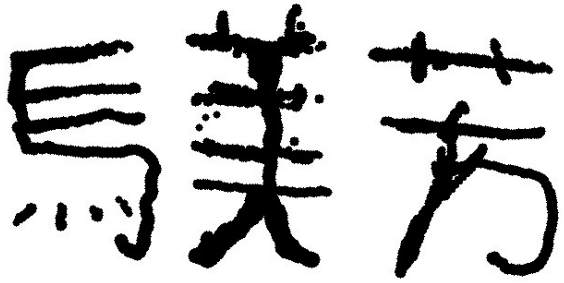
- With normal Air Writing, it is impossible to replicate the Chinese brush patterns.
- Drawing without a controller or digital pen can easily lose accuracy and precision.



Ink Brush on Xuan paper



Apple Pencil on Zen Brush 3



Air Writing on laptop webcam

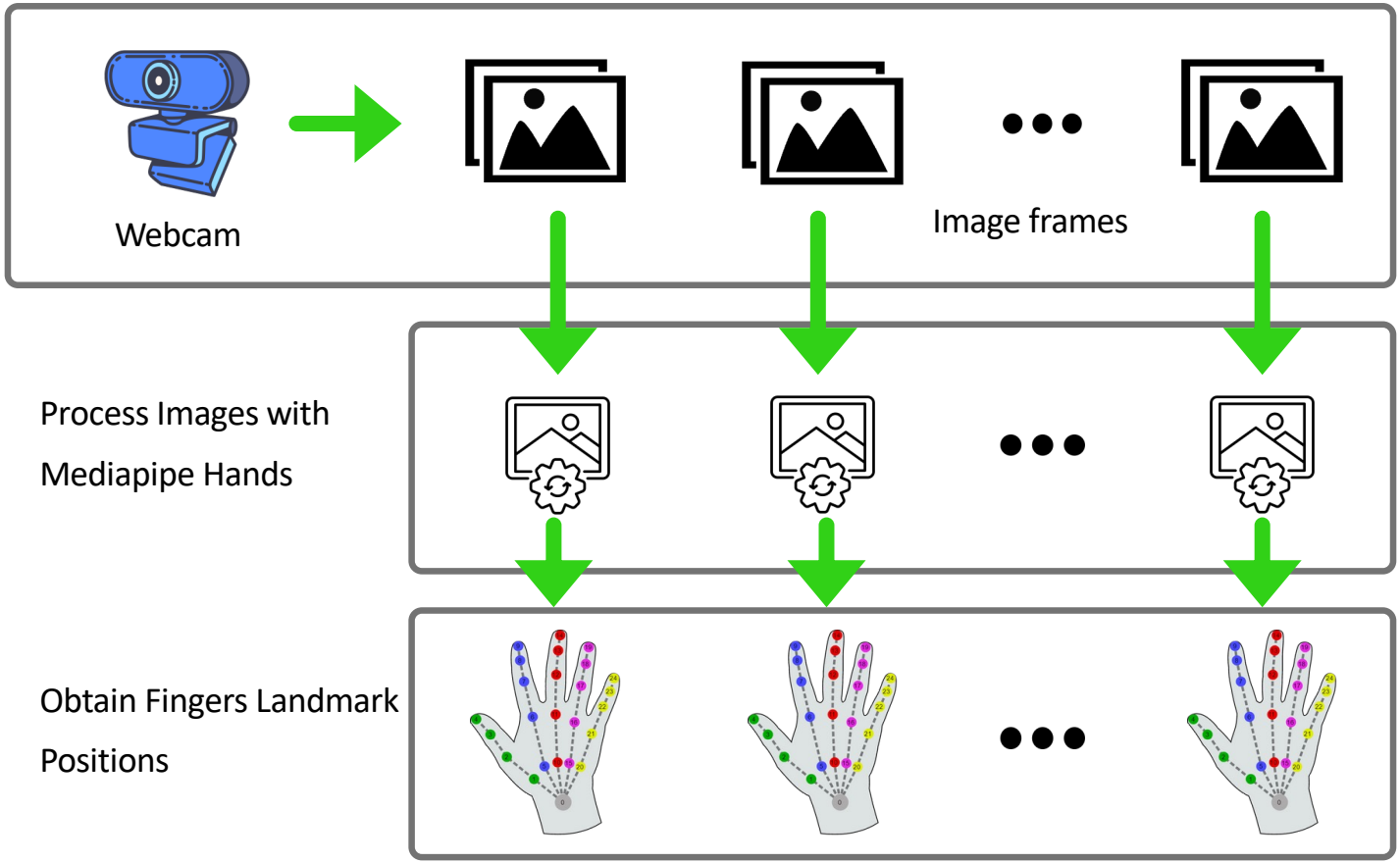
Expected Outcomes

- The user can draw dots and lines by moving their hand to the camera.
- The user can determine the brush size using the distance between their index and thumb fingers.
- The user can stop the drawing by putting their index and thumb fingers close together.
- The generated result from this program will imitate the Chinese calligraphy ink brush pattern.

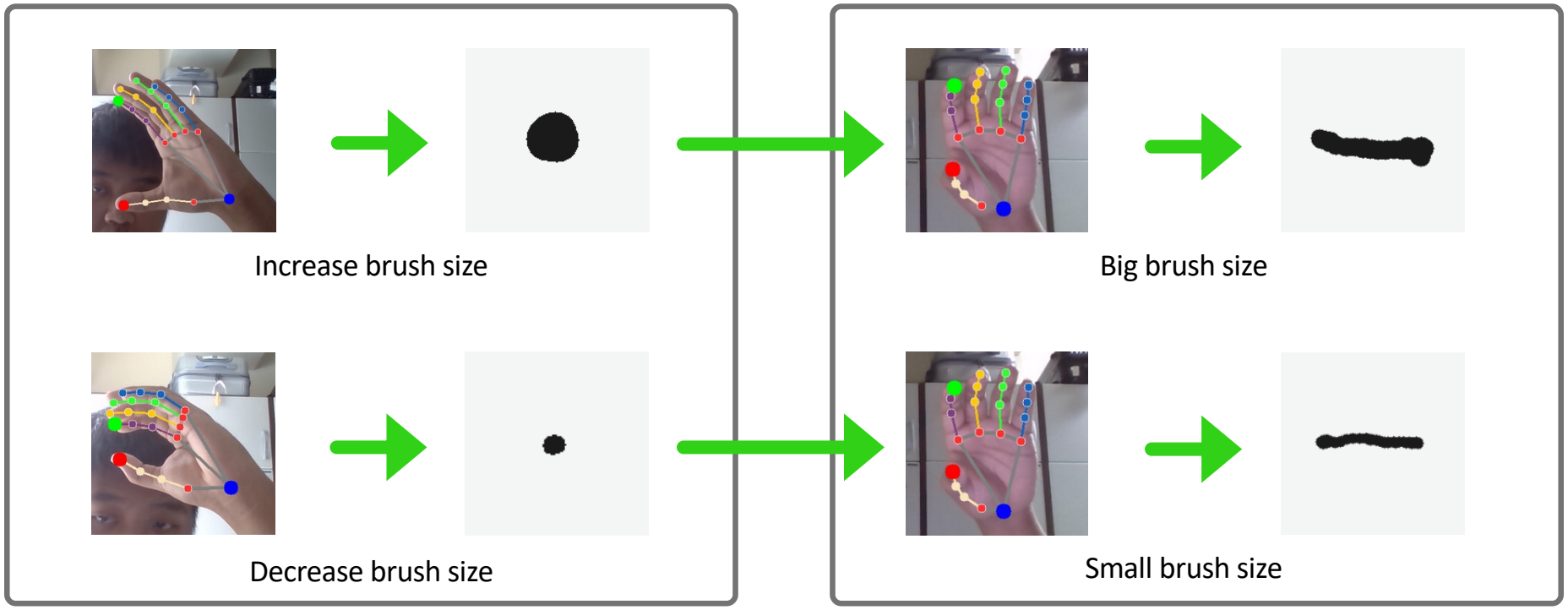
Library

Library	Purpose
OpenCV	• handling video input from the camera • image manipulation • drawing functions
MediaPipe	• hand tracking and obtaining hand landmarks from the video stream
Numpy	• creating and manipulating image arrays
Torch	• used for general PyTorch functionalities
Torchvision	• working with computer vision-related tasks

Air Writing Work Flow



Air Writing Work Flow

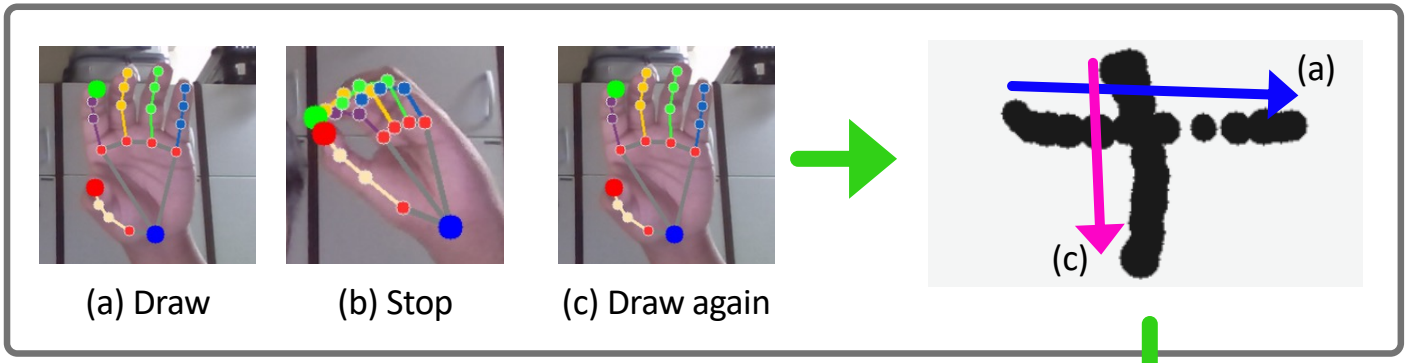


Determine the brush size by the distance between index finger and thumb finger

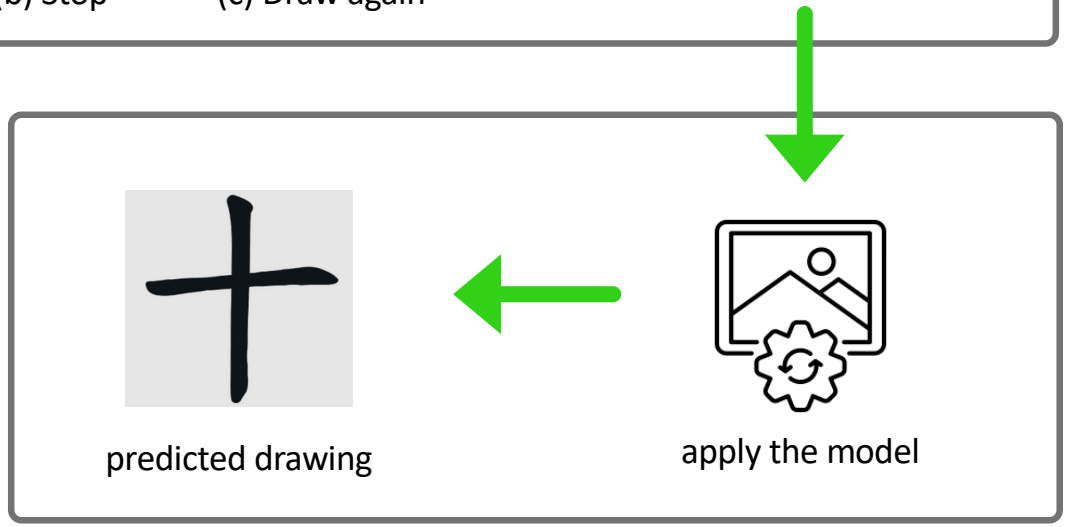
Draw points and lines with current size by moving hand to the camera

Air Writing Work Flow

Stop the drawing when index finger and thumb finger is close together

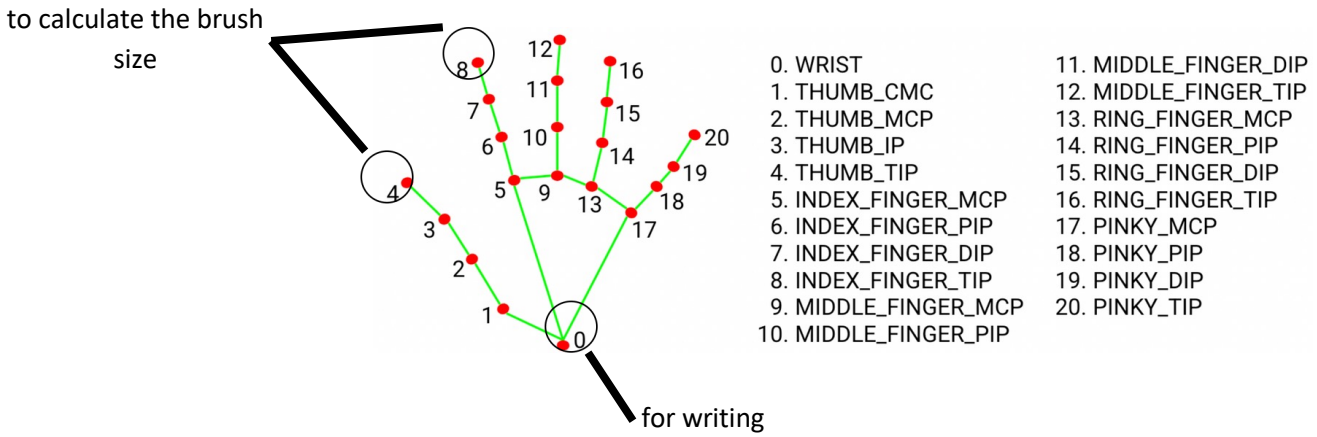


Apply model to make the drawing look like Chinese Caligraphy



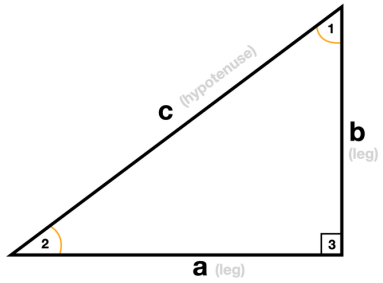
Fingers landmarks

- The landmark number 0 will be the drawing brush cursor
- The landmarks number 4 and 8 will be use in decreasing and increasing the brush size

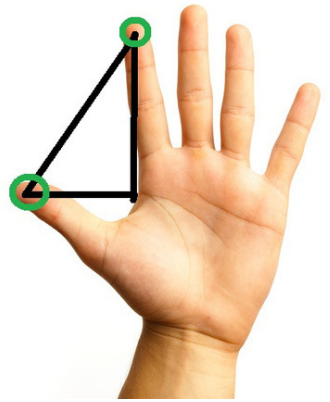


Calculate the distance

This is the formula for calculating the distance from the index finger to the thumb finger which will be determined the size of the drawing brush



$$distance = \sqrt{(ind_y - thu_y)^2 + (ind_x - thu_x)^2}$$



- indy = y-axis of the index finger
- indx = x-axis of the index finger
- thux = x-axis of the thumb finger
- thuy = y-axis of the thumb finger

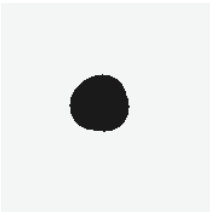
Calculate the strokes size from distance

This is the formula for determine the brush size using the MediaPipe drawing thickness function

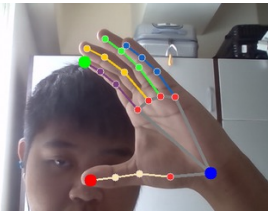
$$\text{strokes size} = \max(1, \text{int}(\text{distance}/5))$$

- Using the distance and divide it by 5 because the distance value is too high for the drawing canvas
- The int() function is used to round the number down to an integer
- The max() function and the value 1 ensure that the size value is at least 1, even if the calculated value is less than 1.

Big brush size

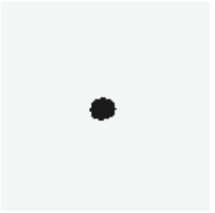


(a)

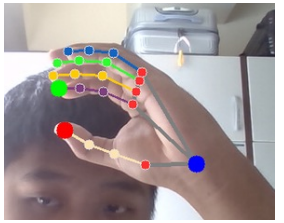


(b)

Small brush size



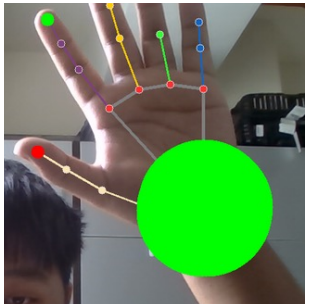
(c)



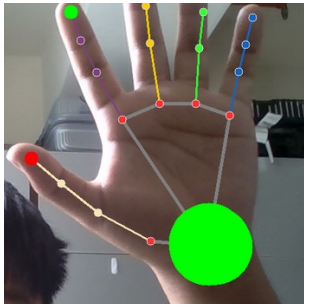
(d)

Distance value to strokes size

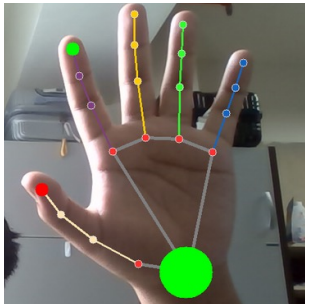
The reason we need to divide distance value by 5 is because the original value is too big



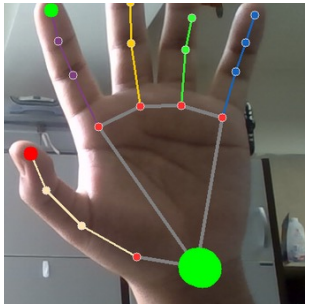
Original



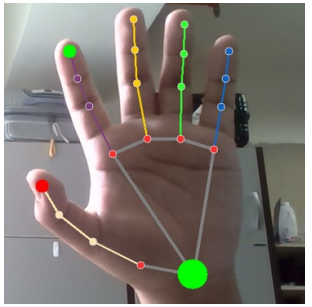
Divide by 2



Divide by 3



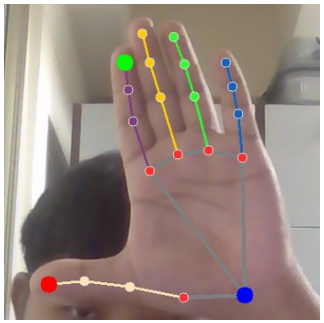
Divide by 4



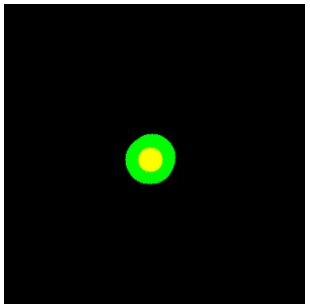
Divide by 5

Maximum & Minimum Strokes size

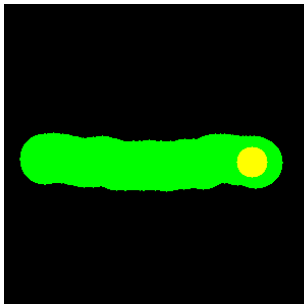
Maximum size Gesture



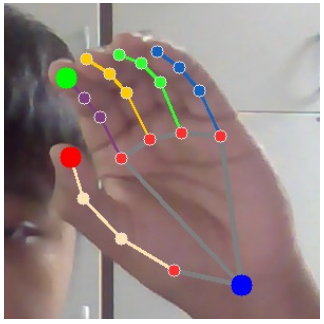
Maximum size Dot



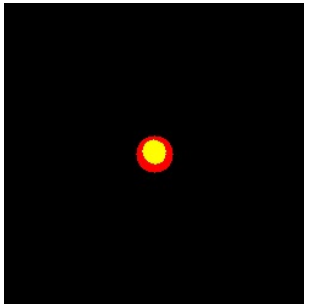
Maximum size Dot



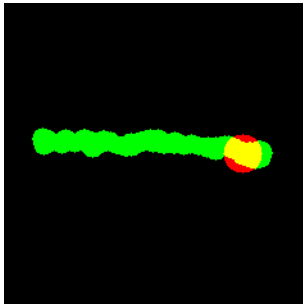
Minimum size Gesture



Minimum size Dot

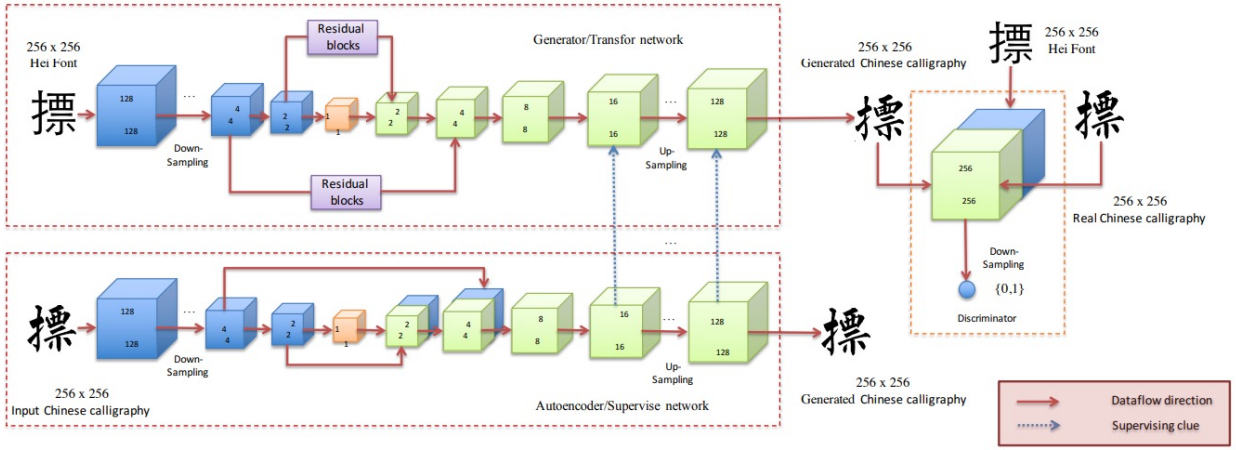


Minimum size Line



Data Model

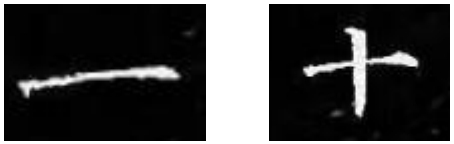
Auto-Encoder Guided GAN for Chinese Calligraphy



What we can do now

(a) (c)

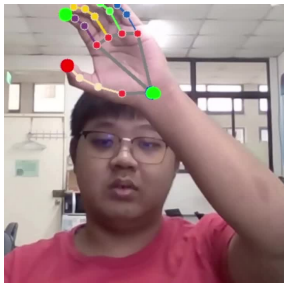
What we want to do



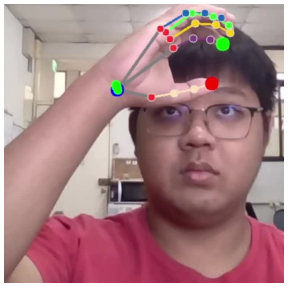
(b) (d)

Writing Gesture Comparison

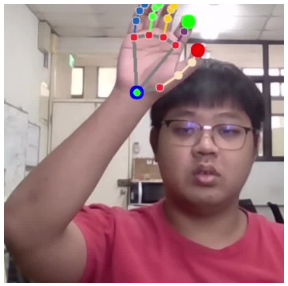
Right hand - side



Right hand - palm



Left hand - side



Left hand - palm

(a) Air writing input 1

(c) Air writing input 2

(e) Air writing input 3

(g) Air writing input 4

(b) Generated Output 1

(b) Generated Output 2

(f) Generated Output 3

(h) Generated Output 4

Results Comparison

Writer: Nopparat

Character: wǔ

Meaning: five

五

Condition:
Writing Hand: Right hand
Environment: Dormitory Room
Camera: HP Pavilion Laptop Camera
Spec: AMD Ryzen 5 7530U 2.00 GHz
with Radeon Graphics and 16 Gbs RAM

No.	Air Writing	Generated Result
1		
2		
3		
4		
5		

Results Comparison

Writer: Nopparat

Character: yǒng

Meaning: forever

永

Condition:
Writing Hand: Right hand
Environment: Dormitory Room
Camera: HP Pavilion Laptop Camera
Spec: AMD Ryzen 5 7530U 2.00 GHz
with Radeon Graphics and 16 Gbs RAM

No.	Air Writing	Generated Result
1	永	永
2	永	永
3	永	永
4	永	永
5	永	永

Results Comparison

Writer: Nopparat

Character: liáng

Meaning: good

良

Condition:
Writing Hand: Right hand
Environment: Dormitory Room
Camera: HP Pavilion Laptop Camera
Spec: AMD Ryzen 5 7530U 2.00 GHz
with Radeon Graphics and 16 Gbs RAM

No.	Air Writing	Generated Result
1	良	良
2	良	良
3	良	良
4	良	良
5	良	良

Results Comparison

Writer: Nopparat

Character: yǒu

Meaning: friend

友

Condition:
Writing Hand: Right hand
Environment: Dormitory Room
Camera: HP Pavilion Laptop Camera
Spec: AMD Ryzen 5 7530U 2.00 GHz
with Radeon Graphics and 16 Gbs RAM

No.	Air Writing	Generated Result
1	友	友
2	友	友
3	友	友
4	友	友
5	友	友

Results Comparison

Writer: Nopparat

Character: rén

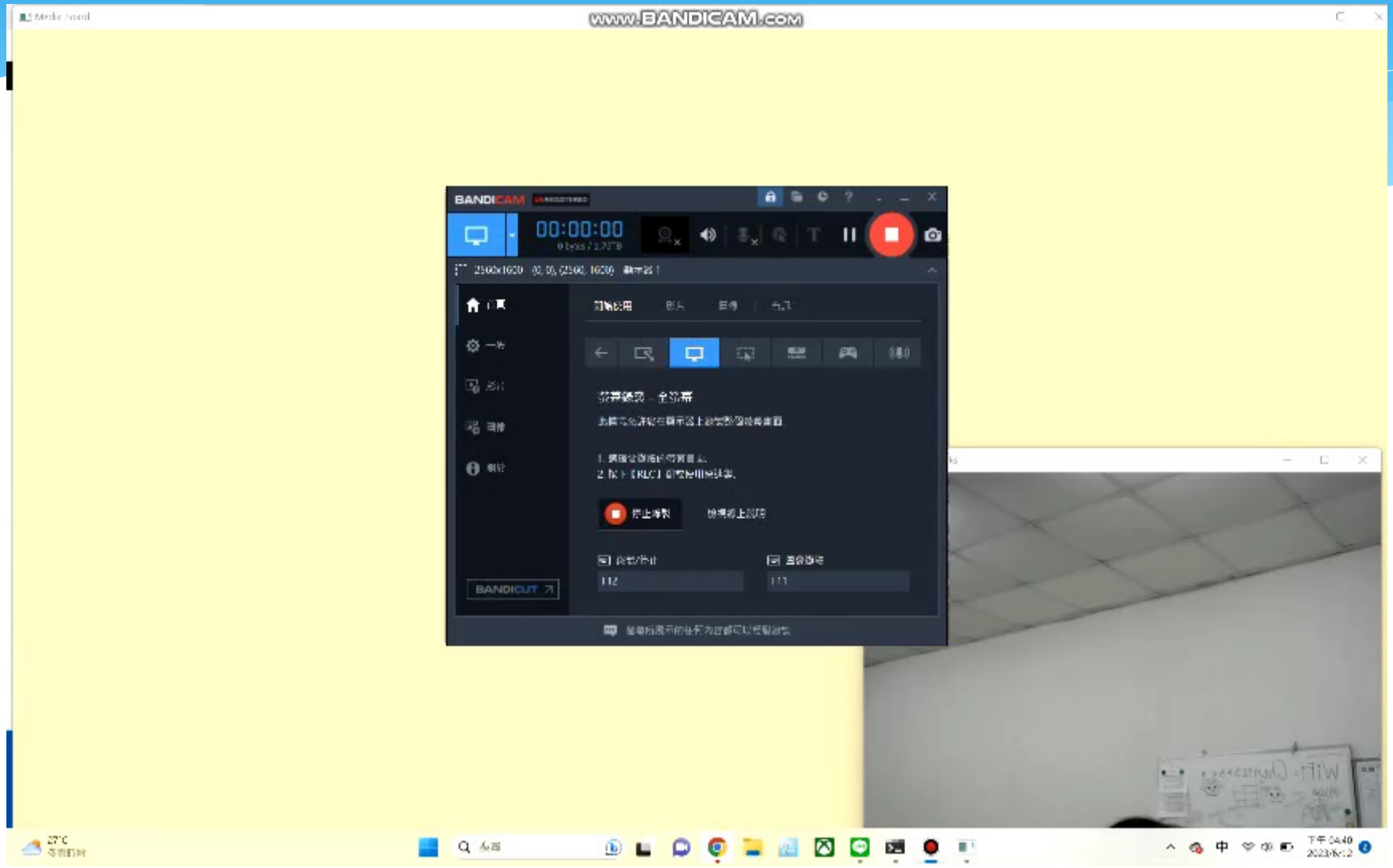
Meaning: human

Condition:
Writing Hand: Right hand
Environment: Dormitory Room
Camera: HP Pavilion Laptop Camera
Spec: AMD Ryzen 5 7530U 2.00 GHz
with Radeon Graphics and 16 Gbs RAM

No.	Air Writing	Generated Result
1		
2		
3		
4		
5		

Another Air Writing version

Ver 1 (gesture control)

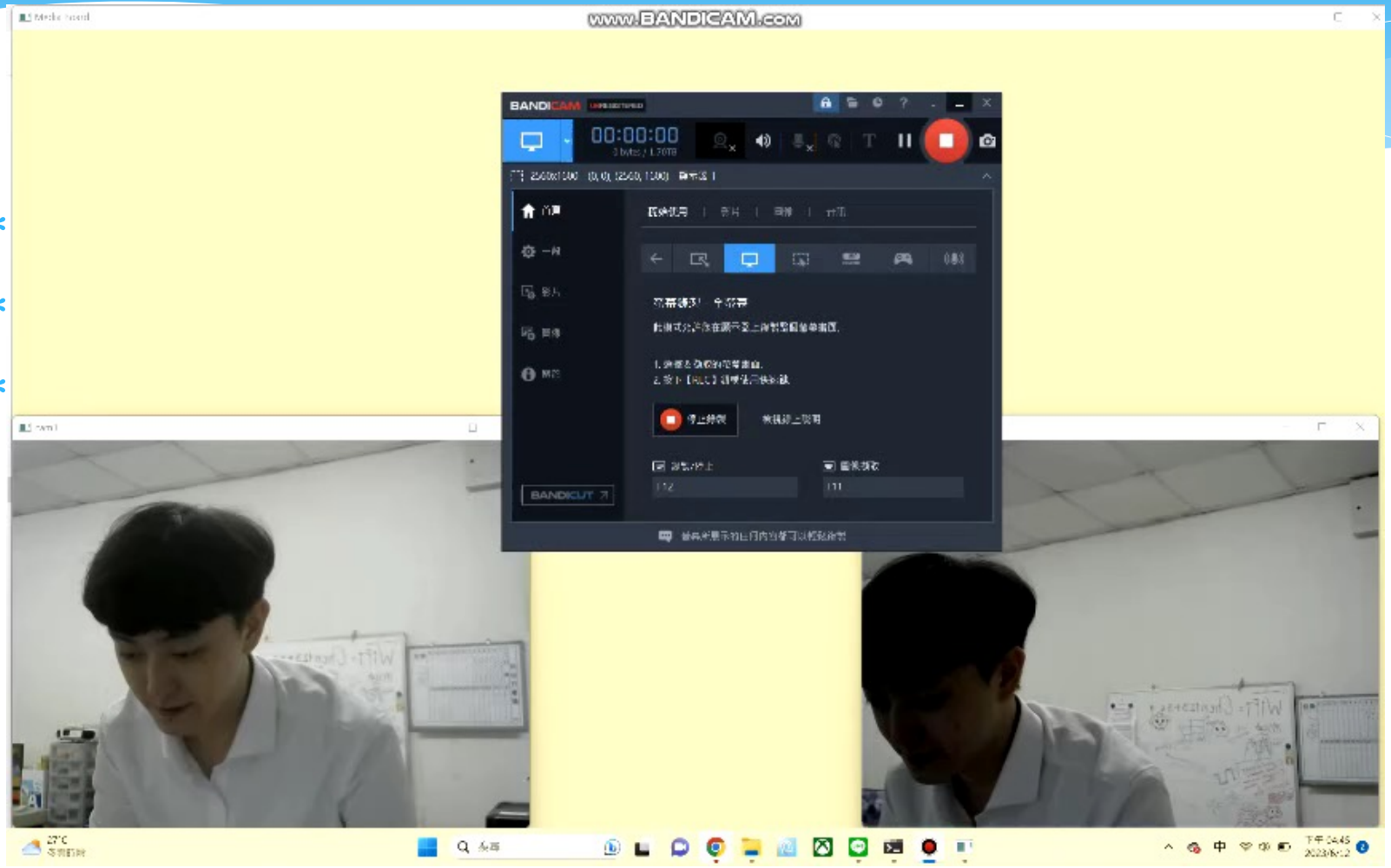


Ver 1 (Grand Hotel)

Ver 2

* Environment
  * Logitech C270 HD Webcam
  * 720p, 30 frames/ second
  * camera length 20 cm

Dual camera



Our future plan

* Depth camera (or two camera)
* (distance + gesture) to control stokes size

Thanks for your listening!