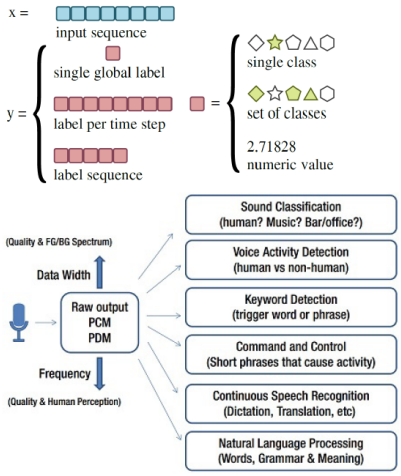
Audio and Video

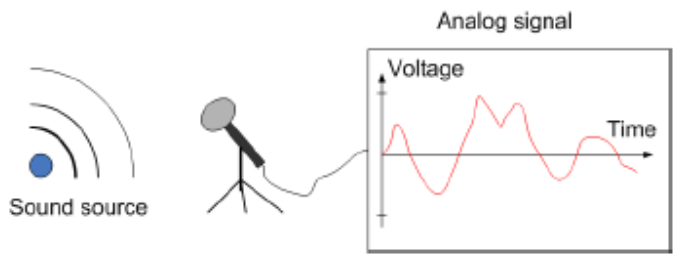
- Audio and video data ->multi-model
- Audio
  - Analyzing
    - Speech
    - Music
    - Environmental sound
  - Synthesis and transformation of audio
    - Source separation
    - Speech enhancement
    - Audio generation

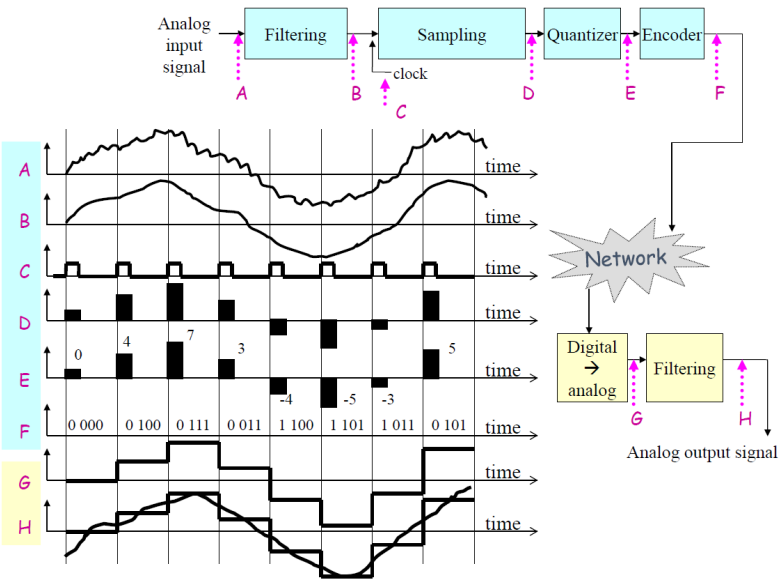


  - Extract high-level descriptions from raw audio signals (sounds) using either signal analysis to extract features and representations, or machine learning (supervised or unsupervised) to discover patterns.

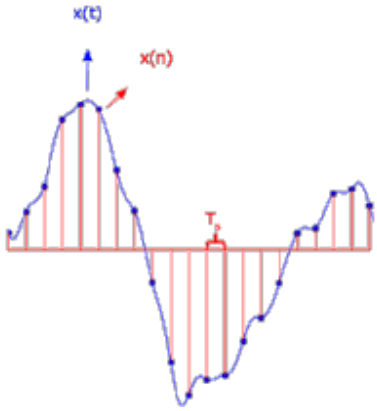
Applications	Audio	Speech	Music
Automatic Speech Recognition (ASR)			
Virtual Assistants			
Music Search			
Surveillance			
Environmental Monitoring			
Recommendation			

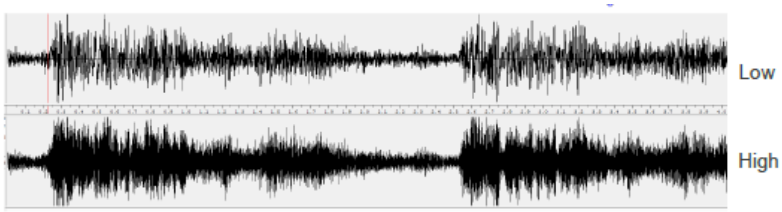
  - Audio signal





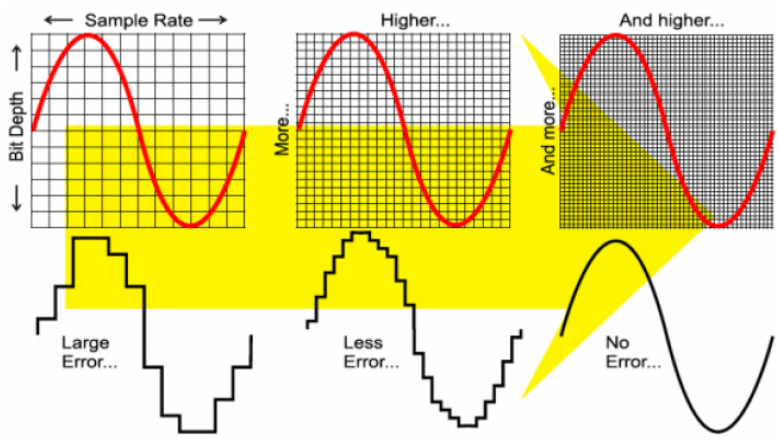
- Sound (physics): A travelling vibration (wave) through a medium (e.g. air) transfers energy (particle to particle) until "perceived" by our ears
- Amplitude: Loudness
- Frequency: Vibrations per second
- sample rate
  - Convert the time-varying continuous signal $x(t)$ to a discrete sequence of real numbers $x(n)$. The interval between two successive discrete samples is the sampling period (Ts). We use the sampling frequency ($fs = 1/Ts$) as the attribute that describes the sampling process. For example: CD recordings 44.1 kHz, telephone 8 kHz, PC and smartphone 16 kHz.





  - According to the Nyquist Shannon theorem, the sampling rate must be at least 2f max , or twice the highest analog frequency component.
  - If the sampling rate is less than 2f max , some of the highest frequency components in the analog input signal will not be correctly represented in the digitized output

- Sampling resolution(quantization)
  - Represent each real number, $xx(nn)$, of the sequence of samples with an approximation from a finite set of discrete values. We usually call this bit resolution property of the quantization procedure “sample resolution” and it is measured in bits per sample.

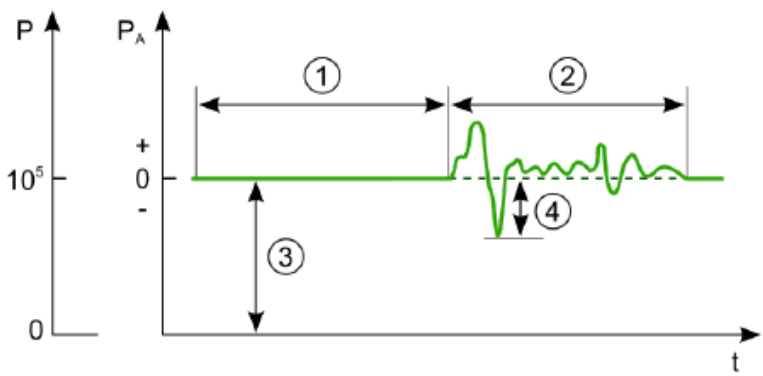


Example: Two methods to represent $(0V, 6V)$ using 2 bits and 3 bits, respectively.

Binary	Voltage (v)
00	0
01	2
10	4
11	6

Binary	Voltage (v)
000	0
001	0.86
010	1.72
011	2.58
100	3.44
101	4.3
110	5.16
111	6

- Audio sound wave



- 1. silence

- 2. audible sound
- 3. atmospheric pressure
- 4. Sound pressure

- Audio features: Time-domain features

Extract a segment (called *frame*) from the audio sequence $\{x(n)\}$, where $n = 1,2,\cdots,W$, that is covered by the window length W

- Energy: Energy of the signal $E = \frac{1}{W}\sum_{n=1}^{W}|x(n)|^2$
- Zero Crossing Rate: Rate of sign changes during the frame

$$Z = \frac{1}{2W}\sum_{n=1}^{W}\left|sign(x(n)) - sign(x(n-1))\right|$$

$$sign(x(n)) = \begin{cases} 1 & x(n) \geq 0 \\ -1 & x(n) < 0 \end{cases}$$

It measures smoothness of a signal is to calculate the number of zero-crossing within a segment of that signal. A voice signal oscillates slowly. A 100 Hz signal will cross zero 100 per second, whereas an unvoiced fricative can have 3000 zero crossing per second.

- Fourier analysis

Signal (Fourier domain) | Apply all signal values | Signal (time domain) | Basis (sinusoid functions)

$$\text{Forward: } F(u) = \sum_{x=0}^{N-1} f(x)e^{\frac{-i2\pi ux}{N}}, \text{where } u = 0,1,\cdots,N-1$$

$$\text{Inverse: } f(x) = \frac{1}{N}\sum_{u=0}^{N-1} F(u)e^{\frac{i2\pi ux}{N}}, \text{where } x = 0,1,\cdots,N-1$$

Note: $e^{ix} = \cos x + i\sin x$; $e^{i\pi} = \cos\pi + i\sin\pi = -1$, Reference: https://en.wikipedia.org/wiki/Euler%27s_identity

Example
- Signal: $f(x) = [2,3,4,4]$
- Fourier coefficients: $F(u) = [13,(-2+i),-1,(-2-i)]$, where i is the imaginary unit

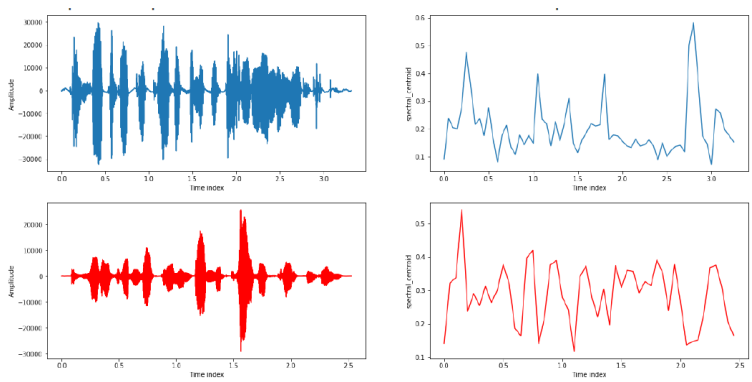
$$F(0) = \sum_{x=0}^{3} f(x)e^{\frac{-i2\pi 0x}{4}} = 2+3+4+4 = 13$$

$$F(1) = \sum_{x=0}^{3} f(x)e^{\frac{-i2\pi x}{4}} = 2e^0 + 3e^{-i\pi/2} + 4e^{-i\pi} + 4e^{-i3\pi/2} = -2+i$$

$$F(2) = \sum_{x=0}^{3} f(x)e^{\frac{-i4\pi x}{4}} = 2e^0 + 3e^{-i\pi} + 4e^{-i2\pi} + 4e^{-i3\pi} = -1$$

$$F(3) = \sum_{x=0}^{3} f(x)e^{\frac{-i6\pi x}{4}} = 2e^0 + 3e^{-i3\pi/2} + 4e^{-i3\pi} + 4e^{-i9\pi/2} = -2-i$$

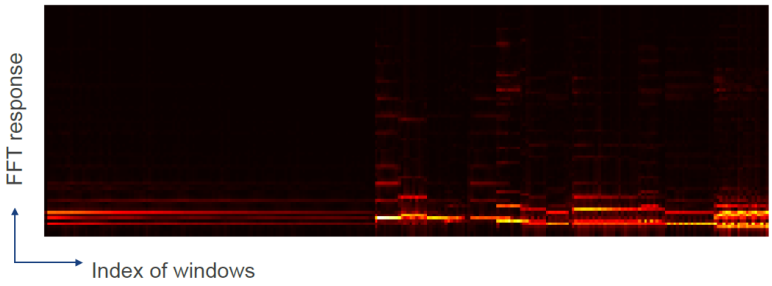
- Audio features: Frequency-domain features

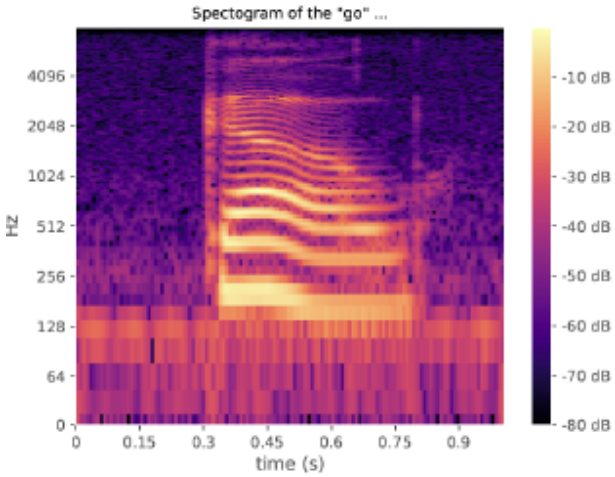


- Spectral centroid: Center of gravity of the spectrum. It is calculated as the weighted mean of the frequencies present in the signal

$$Centroid = \frac{\sum_{u=0}^{N-1} f(u)x(u)}{\sum_{u=0}^{N-1} x(u)}$$

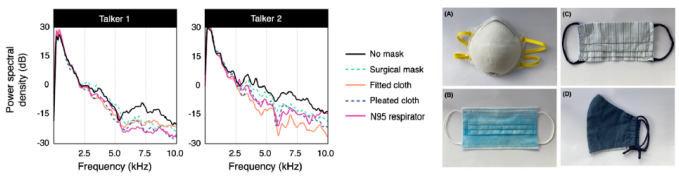
- where $x(u)$ represents the weighting factor (frequency amplitude) of bin number uu, and $f(u)$ represents the center frequency of that bin.
- Frequency representation: Spectrogram



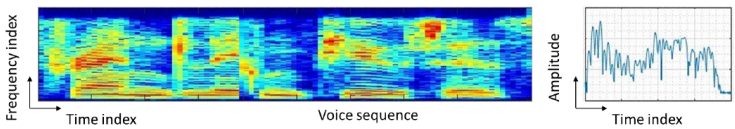


- Spectrogram: Time-Frequency 2D representation
- Step 1, Windowing: Signal broken into (non)overlapping short-term windows
- Step 2, FFT: Fast Implementation of the Discrete Fourier Transform (DFT) Online

Face masks may cause a reduction in speech intelligibility. According to a recent study, there are small differences compared to the no-mask condition at lower frequencies and significant decreases at higher frequencies.

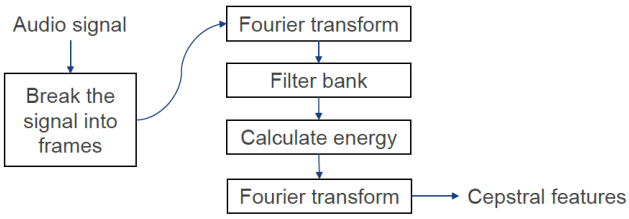


Question: Given a spectrogram shown below, how the spectrogram will change if the talker wears a mask?



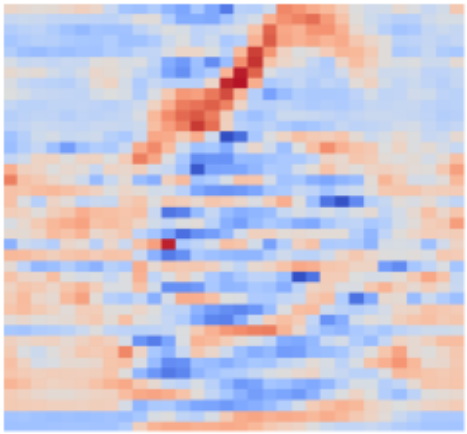
- Audio features: Cepstral domain

Usually select the first 13 MFCCs (considered to carry sufficient discriminative information especially for speech classification tasks)



- Mel-Frequency Cepstral Coefficients(MFCC)

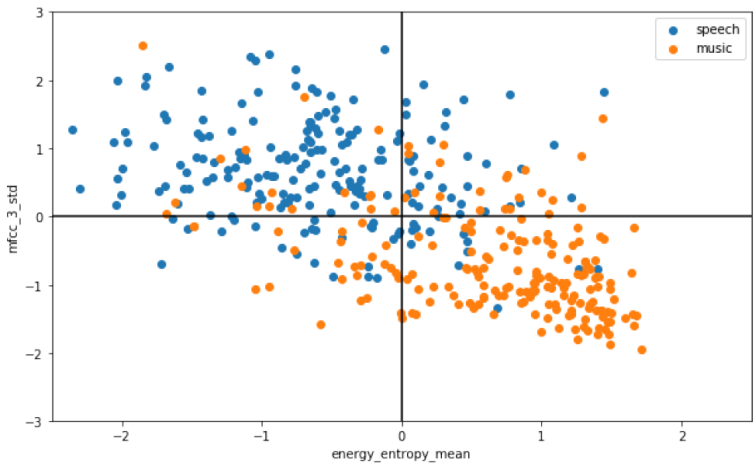
Mel-frequency cepstral coefficients

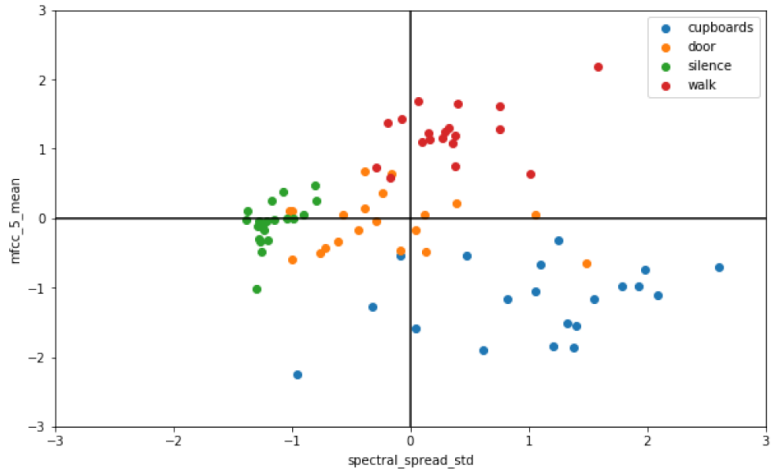


- Compute Fourier transform
- Apply Mel-scale filter bank

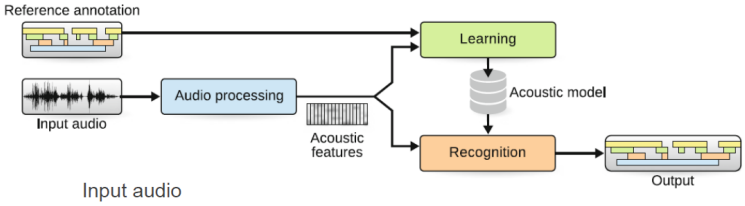
Thresholds(change in frequency)				
Freq	2 Hz	5 Hz	10 Hz	20 Hz
250 Hz	0.7	1	1	1
500 Hz	0.4	0.8	0.9	1
1000 Hz	0.6	0.8	1	0.9
2000 Hz	0.5	0.4	0.9	1
3000 Hz	0.5	0.5	0.6	1

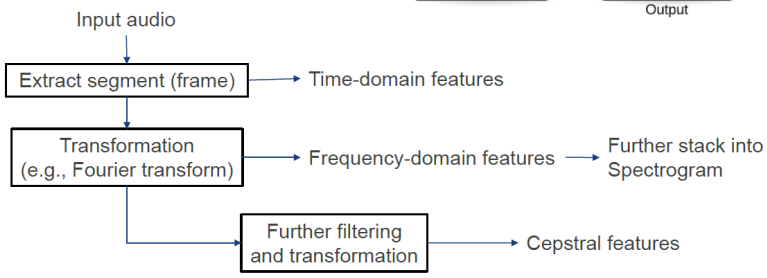
  - Mel scale relates perceived frequency, or pitch, of a pure tone to its actual measured frequency. Humans are much better at discerning small changes in pitch at low frequencies than they are at high frequencies. Incorporating this scale makes our features match more closely what humans hear.
- Compute the power of the output of each filter
- Compute MFCCs as the Fourier transform coefficients of the mel-scaled log-power spectrum
- example



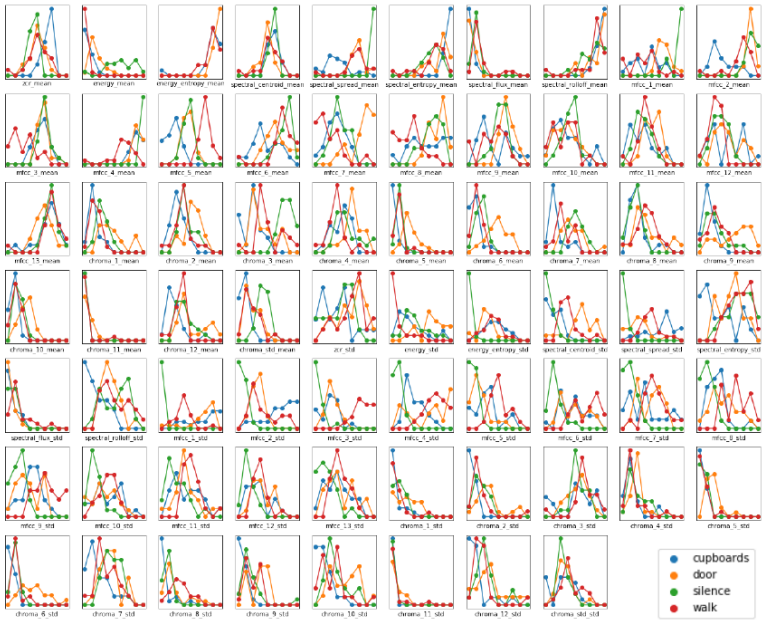


- audio analytics system

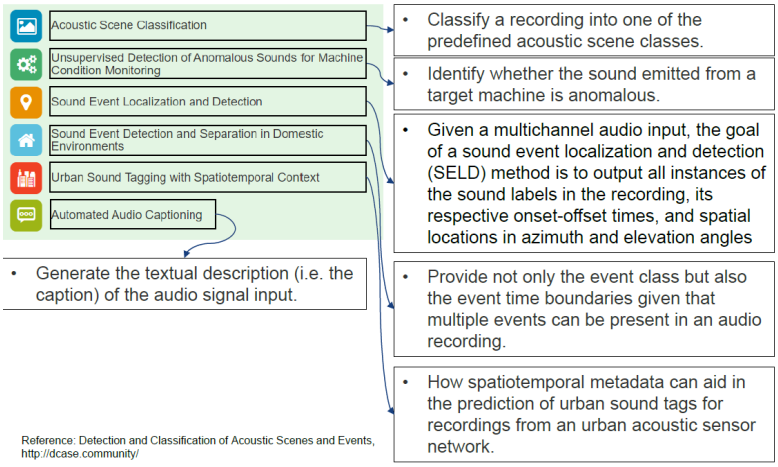




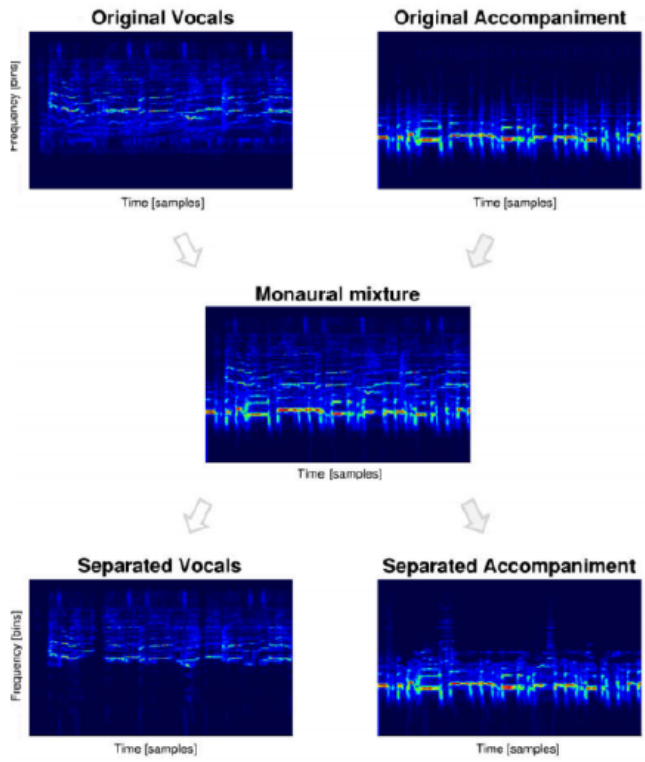
- example



- Sense making from audio

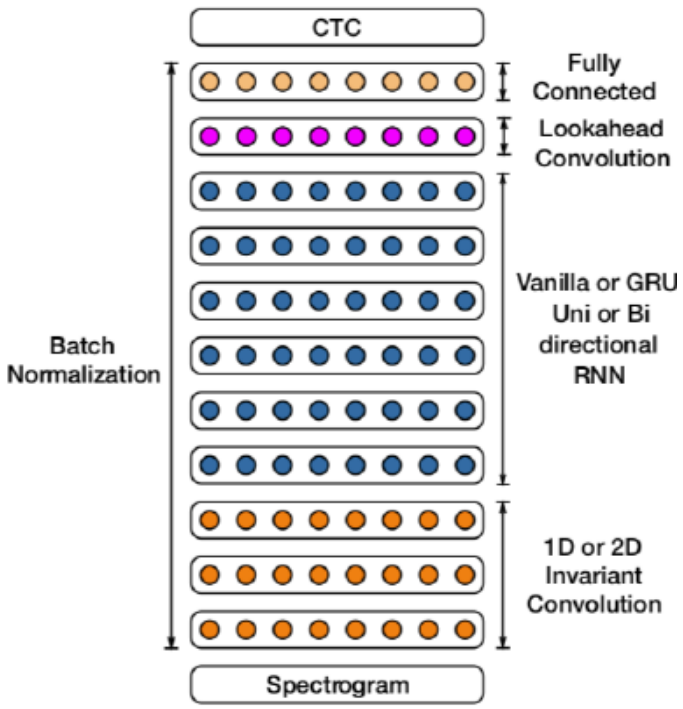


- Audio classification
  - A fundamental problem in audio processing, and the most explored topic, plenty papers published
  - Useful applications based on audio classification:Genre classification\Instrument recognition\Artist identification\Voice recognition
  - Common approach to solve the task:
    - 1.Pre-process audio inputs
    - 2.Extract useful features
    - 3.Perform classification
- Audio Segmentation
  - Segment audio stream into homogeneous regions
  - Purpose: separate regions of different nature, e.g. music/noise, speech/non-speech, male/female
  - Often a preprocessing step for further classification of the segments: Speaker identification/tracking Automatic transcription Segmentation in news broadcast Automatic speech recognition
  - Two steps invovled: feature extraction, statistical analysis (to find segment boundaries)
- Audio Source Separation



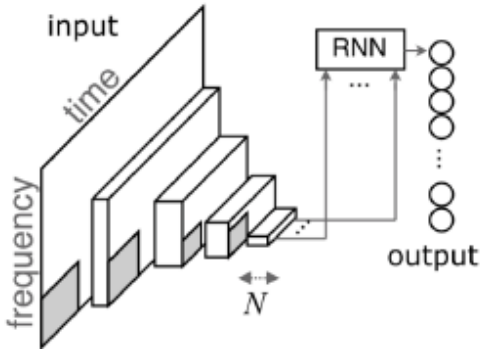
- Cocktail party effect: The ability to focus on a specific human voice while filtering out other voices or background noise (source separation)
- Convert a song (consists of vocals and accompaniment) into a spectogram. Feed a 20-sample spectogram (along time) into a deep neural network.
- The output of the net is a mask (the same size as the input) that will be used to retain values in the input that belonged to vocal
- Get the next 20-sample spectogram (stride of 1, not 20) and repeat the steps until the end
- The masked spectogram, together with the phase components of the original spectogram are used to get the separated audio

- Speech recognition
  - Acoustic model: Maps segments of audio (usually 10 milliseconds) to phonemes
  - Pronunciation model: Connect phonemes together to form words
  - Language model: Express the likelihood of given phrases
  - Since 2014, researchers started to focus on training a single neural network that could map an input audio waveform to an output sentence
    - end to end model



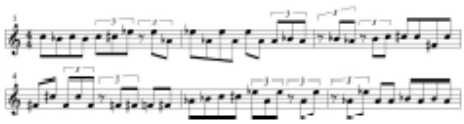
- Example: Baidu has come out an End to-end model that works for English and Chinese
- Inputs: a sequence of log spectograms of power normalized audio segments, calculated on 20ms windows
- Outputs: At each output time-step t, in English, the output is probability prediction on alphabets plus space, apostrophe and blank
- For Chinese, the output is probability prediction on 6000 characters + Roman alphabet
- Note: Connectionist Temporal Classification (CTC) is a type of network output and associated scoring function

- Audio fingerprinting
  - 3 steps are involved to generate audio fingerprint
  - Step 1. Convert an audio file into spectogram. Y-axis represents frequency, X-axis represents time, intensity of shading represents amplitude
  - Step 2. For each section of an audio file, choose the strongest peaks and spectogram is reduced to scatter plot (constellation map)
  - Step 3. Generate combinatorial hashes based on every pair of anchor-point to point in the map
- Automatic music tagging



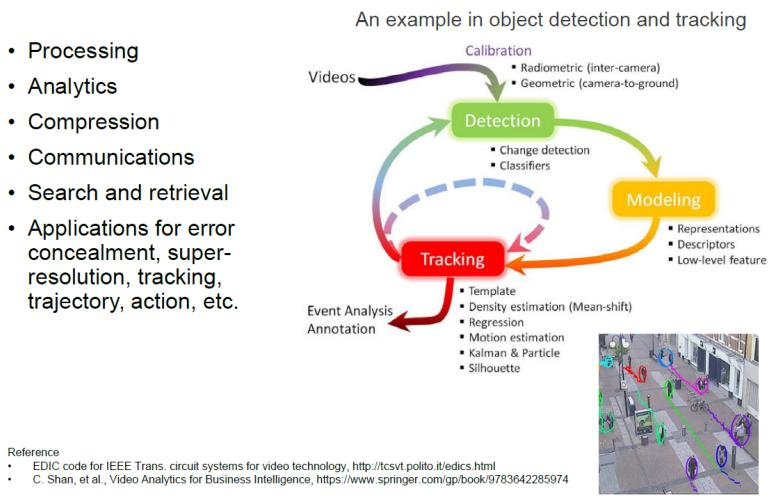
- Tags: Text-based labels that encode semantic information related to sound (Panagakis &
- Kotropoulos, 2011)
- Music tagging: Add tags to song
- Example of tags: rock, pop, jazz, 60s, sad and etc ...
- Music tagging is a multi-label classification problem (should not use softmax in the last layer, use sigmoid instead)
- Input: Mel-spectograms of a song

- Automatic music transcription



- Music transcription: Turn a piece of music into musical score (and make it playable by electronic instrument)
- Convert song into spectogram,split the spectogram into segments with a length equivalent to 1/8 second
- Model: Convolutional neural network with sigmoid used for last layers
- Output: Notes to be played with specified velocities and time delay

- Video



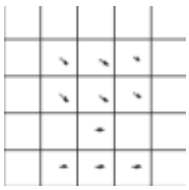
- Brief Info

- A video is a sequence of frames captured over time. Data is a function of space (x, y) and time (t).
- Frame resolution: Dimension of each frame.
- Frame rate: The number of frames or images that are captured or displayed per second, usually measured in frames per second (fps).
- why do we need video(compare with static image)
  - object motion
  - see scene change over the time
  - track object trajectory
  - recognize event

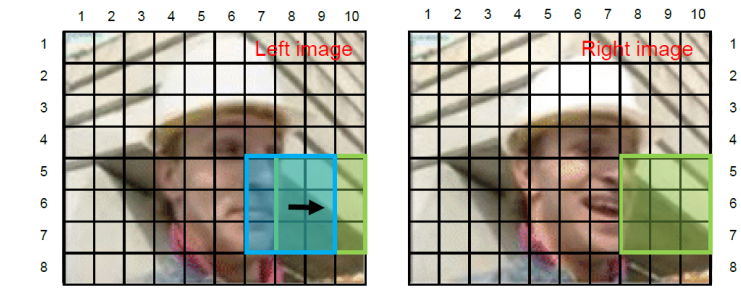
- Motion representation
  - motion vector
    - The motion vector describes the2 D displacement at the pixel location between the reference image (right) and the other target image (left).
    - Frame-based

      - Entire motion field is represented by a few global parameters, such as translation.
    - Pixel-based

      - One motion vector at each pixel.
    - Block-based

      - Entire frame is divided into blocks, each block is represented by a constant motion vector.
      - Assume all pixels in a block undergo a coherent motion, and search for the motion parameters for each block independently



Current block
Best matched block
Motion vector

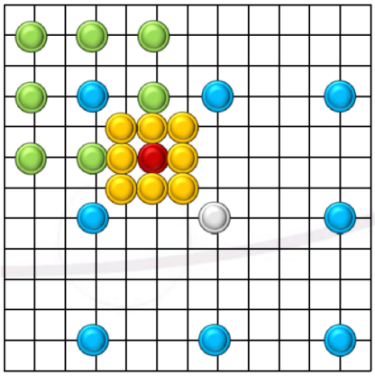
Each image has a size of 10×8 pixels, each block has a size of 3×3 pixels	(row coordinate, column coordinate)	Motion vector
Block center in current frame	$(6, 9)$	Either method is okay. • Current – previous: $(0,1)$ • Previous – current: $(0, -1)$
Best matched block center in previous frame	$(6, 8)$	

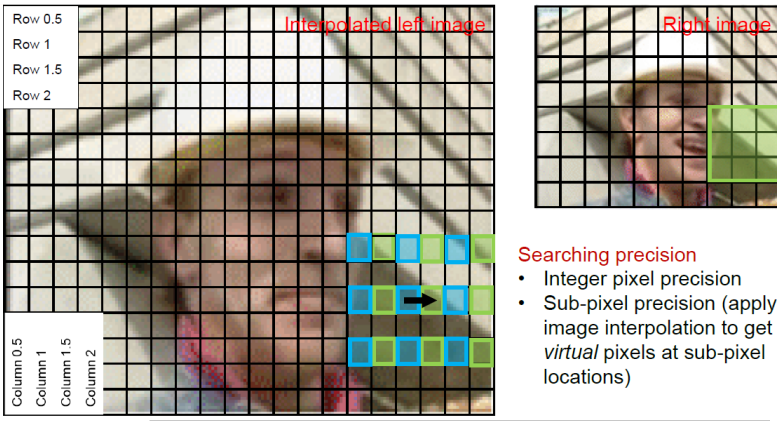
Searching criterion: For the reference image (block) X and the target image (block) Y, each has a size of $M \times N$, calculate their difference (e.g., MSE)

$$MSE = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}(X(i,j) - Y(i,j))^2$$

Searching strategy
- Full (slow) search
- Sub-optimal (fast) search, e.g., three-step search shown in right figure

Search centre
1st stage
2nd stage
3rd stage
Best candidate





Searching precision
- Integer pixel precision
- Sub-pixel precision (apply image interpolation to get *virtual* pixels at sub-pixel locations)

Current block
Best matched block
Motion vector

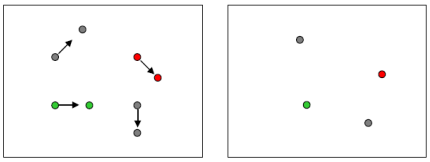
Interpolated left image: 20×16, Right image: 10×8, Each block has a size of 3×3 pixels	(row coordinate, column coordinate)	Motion vector
Block center in current frame	$(6, 9)$	Either method is okay. • Current – previous: $(0, 0.5)$ • previous – current: $(0, -0.5)$
Best matched block center in previous frame	$(6, 8.5)$	

1. Start with search location at center
2. Set step size S = 4 and search parameter p = 7
3. Search 8 locations +/- S pixels around location (0,0) and the location (0,0)
4. Pick among the 9 locations searched, the one with minimum cost function
5. Set the new search origin to the above picked location
6. Set the new step size as S = S/2
7. Repeat the search procedure until S = 1

The resulting location for S=1 is the one with minimum cost function and the macro block at this location is the best match.

There is a reduction in computation by a factor of 9 in this algorithm. For p=7, while ES evaluates cost for 225 macro-blocks, TSS evaluates only for 25 macro blocks.

- Optical Flow



$I(x, y, t)$ $\quad$ $I(x, y, t+1)$

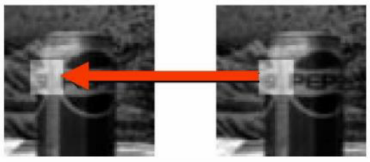
Notation	
(x, y)	Pixel coordinate
I	Image intensitz
t	Frame index
$[u, v]$	Motion vector

- Definition:Optical flow is the apparent motion of brightness patterns in the image.

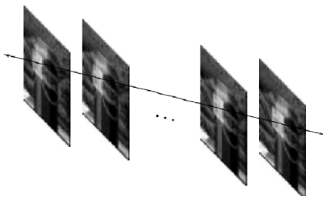
- Objective:Estimate image motion at each pixel location to obtain optical flow

Brightness consistency: Image brightness in a small region remain the same although their location may change. That is,

$$I(x+u, y+v, t+1) = I(x, y, t)$$



Small motion: The image motion of a patch changes gradually over time. (u, v are less than 1 pixel, and smoothly varying over the time)



- Calculation
  - Brightness constancy: $I(x, y, t) = I(x+u, y+v, t+1)$
  - Small motion: We take Taylor series expansion of the image intensity I

$$I(x+u, y+v, t+1) = I(x, y, t) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \frac{\partial I}{\partial t} + \cancel{\text{higher order terms}}$$

$$\cancel{I(x+u, y+v, t+1)} = \cancel{I(x, y, t)} + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \frac{\partial I}{\partial t}$$

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

$$0 = I_x u + I_y v + I_t$$

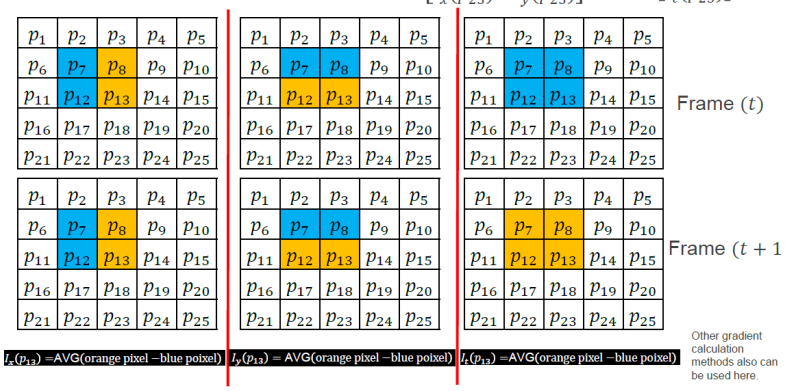
Note: $I_x = \frac{\partial I}{\partial x}, I_y = \frac{\partial I}{\partial y}, I_t = \frac{\partial I}{\partial t}$

Spatial coherence constraint: Assume a small block of neighboring pixels have similar or even the same (u, v). If we use a 5×5 window centered at the pixel location (x, y), that gives us 25 equations at 25 pixel locations $p_1, p_2, \cdots, p_{25}$.

$$\begin{bmatrix} I_x(p_1) & I_y(p_1) \\ I_x(p_2) & I_y(p_2) \\ \vdots & \vdots \\ I_x(p_{25}) & I_y(p_{25}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(p_1) \\ I_t(p_2) \\ \vdots \\ I_t(p_{25}) \end{bmatrix}$$

Spatial coherence constraint: Assume the pixel's neighbors have the same (u, v). If we use a 5×5 window centered at the pixel location (x, y), that gives us 25 equations.

$$\begin{bmatrix} I_x(p_1) & I_y(p_1) \\ I_x(p_2) & I_y(p_2) \\ \vdots & \vdots \\ I_x(p_{25}) & I_y(p_{25}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(p_1) \\ I_t(p_2) \\ \vdots \\ I_t(p_{25}) \end{bmatrix}$$



Question: How can we use non-consecutive frames (e.g., frame skipping)?

- Recall our small motion assumption

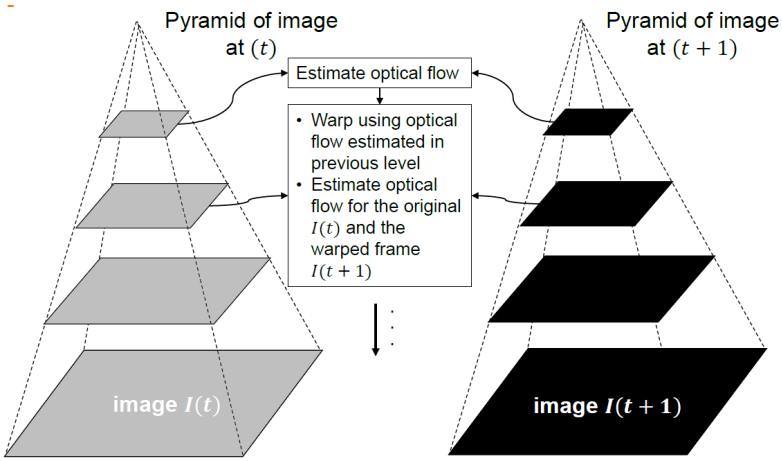
$$I(x+u, y+v) = I(x, y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \cancel{\text{higher order terms}}$$

$$\approx I(x, y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v$$

- To do better, we need to add higher order terms back

Iterative Lucas-Kanade estimation (see next slide)

1. Estimate optical flow at each pixel by solving Lucas-Kanade equations
2. Warp $I(t)$ towards $I(t+1)$ using the estimated flow field
3. Repeat until convergence



- when/where the optical flow fail

> - When: Can we differentiate the optical flow due to camera motion, lighting change, object motion? [No]
> - Where: Which part of the flow vector cannot be determined from images? [flat regions of the image.]

Recall the optical flow solution for a 5×5 image patch

$$\begin{bmatrix} I_x(p_1) & I_y(p_1) \\ I_x(p_2) & I_y(p_2) \\ \vdots & \vdots \\ I_x(p_{25}) & I_y(p_{25}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(p_1) \\ I_t(p_2) \\ \vdots \\ I_t(p_{25}) \end{bmatrix} \qquad \underset{25 \times 2}{\mathbf{A}} \quad \underset{2 \times 1}{\mathbf{d}} = \underset{25 \times 1}{\mathbf{b}}$$

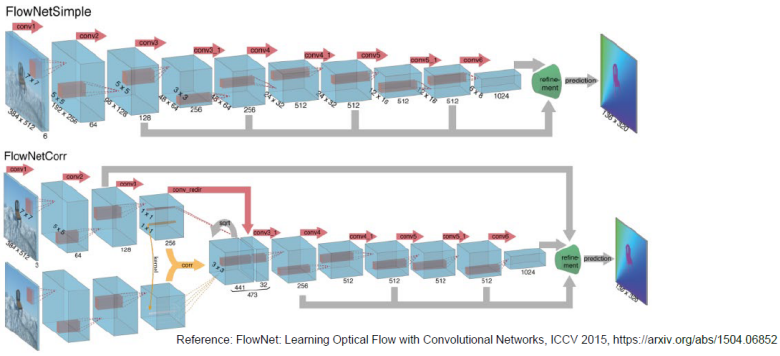
Least squares solution for $\mathbf{d}$ given by $(\mathbf{A}^T\mathbf{A})\mathbf{d} = \mathbf{A}^T\mathbf{b}$

$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

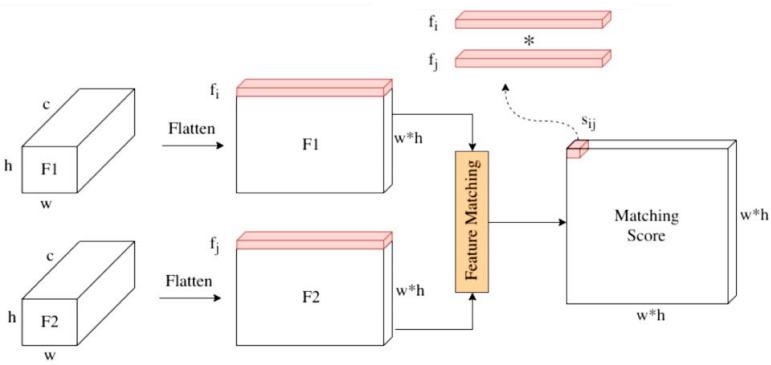
> - Question: Where is $\mathbf{d}$ solvable?
> $\mathbf{A}^T\mathbf{A}$ should be invertible, $\mathbf{A}^T\mathbf{A}$ should not be too small due to noise, eigenvalues λ_1 and λ_2 of $\mathbf{A}^T\mathbf{A}$ should not be too small.
> - A texture image region: Yes
> - A flat image region (e.g., plain wall): No

The summations in above equation are over all 25 pixels.

- Optical flow using deep learning



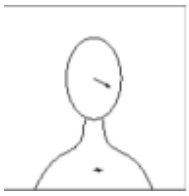
Reference: FlowNet: Learning Optical Flow with Convolutional Networks, ICCV 2015, https://arxiv.org/abs/1504.06852



- FlowNet Simple:Stack two sequentially adjacent input images together and feed them into the network.

- FlowNet Corr:First produce representations of two images separately,and then combine them together in the ‘correlationlayer’(Dot product between pair-wise feature vectors) ,and learn the higher representation together.

- Region-based



  - entire frame is divided into regions, each region is represented by an object with consistent motion

- Object tracking

Detection	Tracking
Applied on static image, or independently on each frame of the sequence	Applied on a sequence of images
Relies on detector trained from (known) object data off line.	Can handle (unknown) object on line.
Can get the position of objects	Preserve identity, can get the trajectory of the objects
Can not handle occluded objects	Can handle occluded objects (based on tracking assumption)

  - challenges

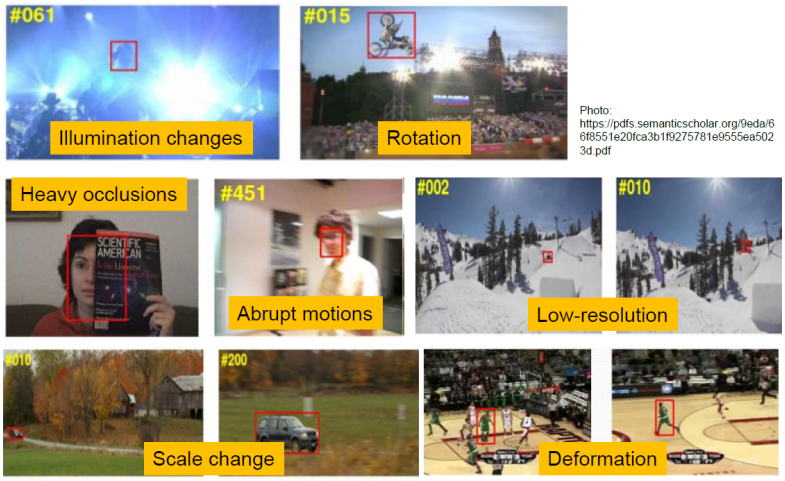
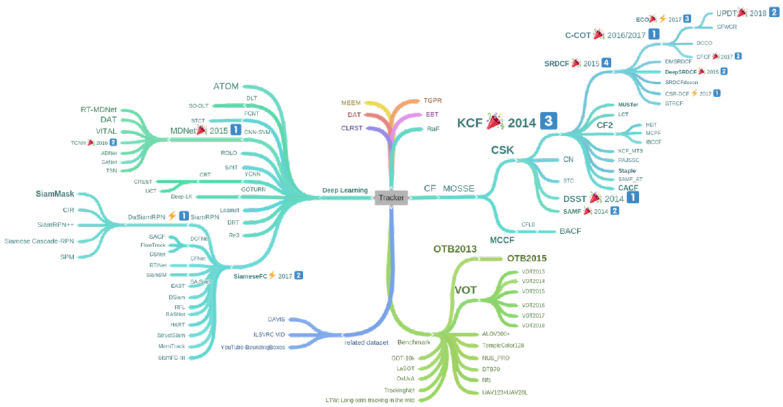


Photo: https://pdfs.semanticscholar.org/9eda/66f8551e20fca3b1f9275781e9555ea5023d.pdf

  - Key components
    - Tracker initiation: We need to define the initial state of the target by either (manually) drawing a bounding box or (automatically) detecting an object of interest (say, human, car, etc).
    - A module holding the internal representation of the object: We need to learn the visual appearance of the object.
    - A module observing the features of the search image (candidate regions).
    - A module evaluating the object and the search image for estimating the motion: Evaluate the candidates in a searching region to identify the exact location of the target.

- A module updating the representation of the object: The object might change its appearance during tracking.
- Generative methods: Represent objects with appearance models, and track targets by searching for the image region most similar to the models.
- Discriminative methods: Treat objects tracking as a binary classification problem to distinguish the target from non-target background.

- BenchMark

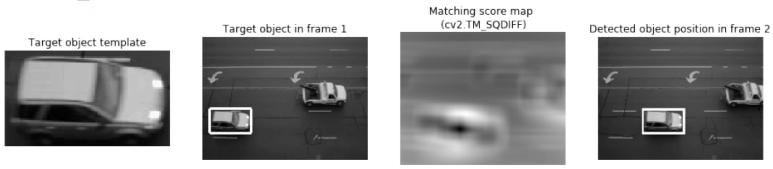


- Generative: Template matching
  - Simple color similarity based template matching.

    The function slides through the image I, compares the overlapped patches of size $w \times h$, against the object template T, using the specified method and stores the comparison result.

    At the position (x, y), the score $R(x, y) = \sum_{m,n}(T(m, n) - I(x + m, y + n))^2$

    cv2.TM_SQDIFF: Absolute difference

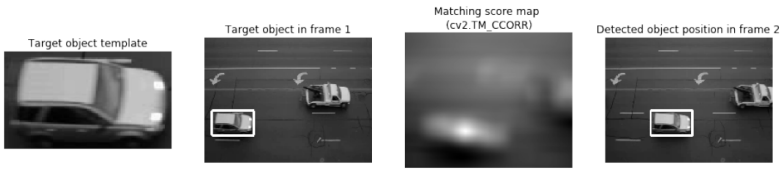
    

    - Good for matching the same type of objects with similar appearance
    - Not work well for the variations in pose, lighting, natural variations or non-rigid transformations.
  - Simple cross correlation based template matching.

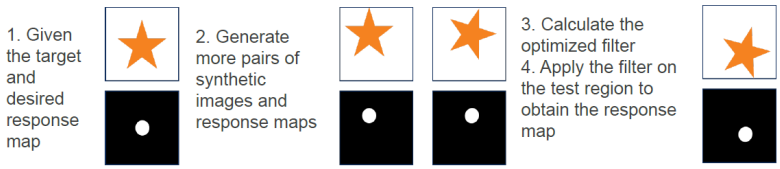
    The function slides through the image I, compares the overlapped patches of size $w \times h$, against the object template T, using the specified method and stores the comparison result.

    At the position (x, y), the score $R(x, y) = \sum_{m,n}(T(m, n) \times I(x + m, y + n))$

    cv2.TM_CCORR: Cross correlation

  - Challenge



- Time consuming: Repeat this correlation calculation on the whole image.
  - Perform checking in the search region only (not the whole image).
- Not robust to illumination change: Correlation function relies on intensity.
  - Learn an optimized filter based on a set of templates not a single template (see next slide).
- Search region (in the current frame) where a searching is performed
- Filter
  - Learn an “optimized correlation filter”, when correlated with a target template, produces a strong peak at the location of the object and low (or even zero) values at all the other locations. (More robust than a single template-based tracking.)
  - Minimum output sum of squared error filter (MOSSE)
    - filter minimizes the sum of squared error between the actual output of the correlation and the desired output of the correlation.Given a target template and a synthetic response image wwcontains a Gaussian peak centred on the object. Apply affine transformation (e.g., transition, rotation) to generate several pairs of $xxii$ (synthetic input image) and $yyii$ (synthetic response image) to calculate ww (unkown optimized filter).

      $$w^* = \text{argmin}(\textstyle\sum_i \|x_i \circledast w - y_i\|^2).$$

  - Kernelized Correlation Filters (KCF)
    - Further introduce regularization, kernel matrix, into the cost function

      $$w^* = \text{argmin}(\textstyle\sum_i \|x_i \circledast w - y_i\|^2 + \text{Regularization } (w))$$

- Response map that indicates the location of best matching

- Discriminative tracking

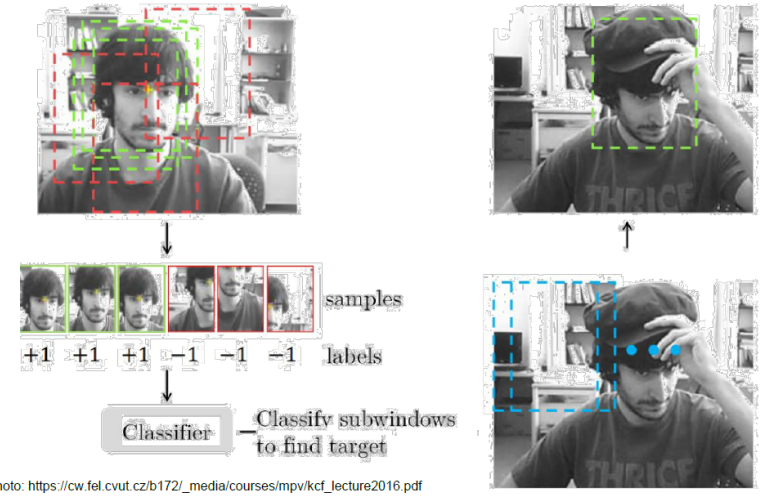
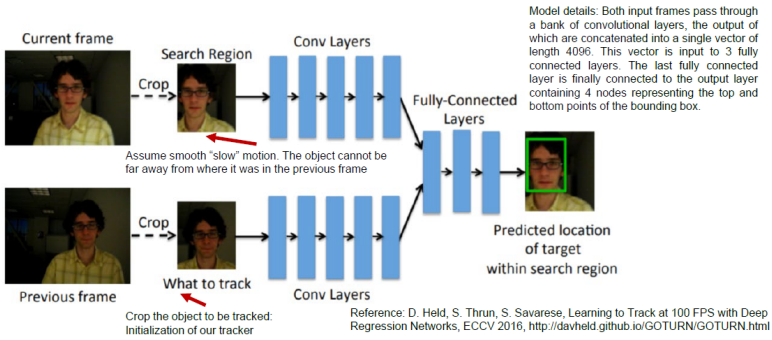
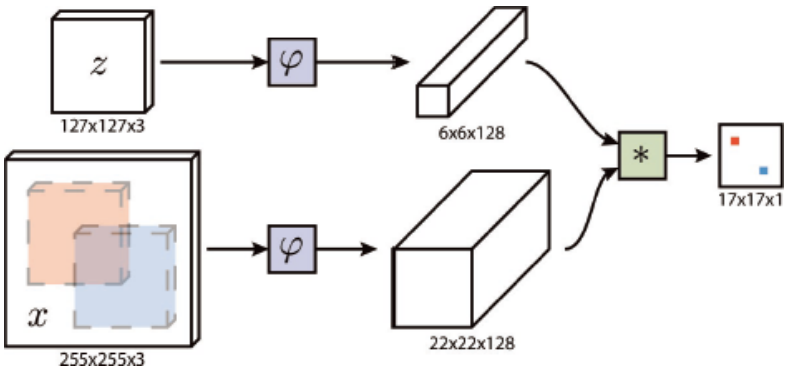


Photo: https://cw.fel.cvut.cz/b172/_media/courses/mpv/kcf_lecture2016.pdf

  - Train an online discriminative classifier based on positive and negative samples.
  - No offline annotated data is needed.
- Tracking using deep learning:
  - GOTURN(Generic Object Tracking Using Regression Networks)



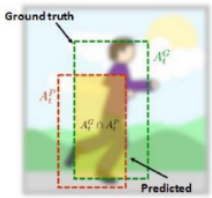
    - Input: A search region in the current frame and a target from the previous frame
    - Output: The coordinates (regression) of the object in the current frame, relative to the search region. The network's output consists of the coordinates of the top left and bottom right corners of the bounding box.
    - It requires offline annotated data.
  - SiamFC

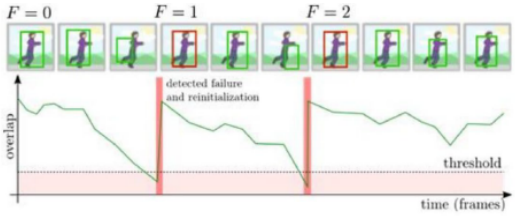


    - Input: A target (in the previous frame, z) and a search region (candidates, xx) centered at the previous position of the target. The dimensions (e.g., $127 \times 127 \times 3$) used in this example is just for illustration.

- Output: A score map. The position of the maximum score relative to the center of the score map, multiplied by the stride of the network, gives the displacement of the target from frame to the frame.
- It requires offline annotated data.
- The ground truth score maps are automatically generated: values 1 at pixel locations where they are within certain radius of the target center (manually labelled); values 0 elsewhere.
- Model details: Convolutional network φ with respect to the search image x. The output is a scalar-valued score map whose dimension depends on the size of the search image. The red and blue pixels in the score map contain the similarities for the corresponding sub-windows.

- Single object tracking evaluation metric





  - Accuracy: Average overlap during successful tracking
  - Robustness: Number of times a tracker drifts off the target
  - One Pass Evaluation: Run tracker throughout a test sequence initialized by ground truth bounding box in the first frame and return the average precision.
  - Spatial Robustness Evaluation: Run tracker throughout a test sequence with initialization from a few different bounding boxes by shifting or scaling ground truth in the first frame and return the average precision.

- Multiple object tracking
  - For each frame, first localize all objects using an object detector
  - Associate detected objects between frames
  - Make multiple object tracking to be a association problem, instead of a tracking problem.
  - matching

We have N objects in previous frame and M objects in current frame. We can build a table of match scores $m(i,j)$ for $i = 1,2,\cdots,N$ and $j = 1,2,\cdots,M$.

The pairwise match score between objects in two consecutive frames

	A	B	C	D	E
A	0.95	0.76	0.62	0.41	0.06
B	0.23	0.46	0.79	0.94	0.35
C	0.61	0.02	0.92	0.92	0.81
D	0.49	0.82	0.74	0.41	0.01
E	0.89	0.44	0.18	0.89	0.14

Q1: How to obtain match score?

- Association based on location, motion, appearance and so on. (see following slides)

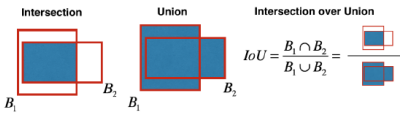
Q2: How to choose 1-1 correspondence that maximizes sum of match scores?

- We have many methods to address this issue, such as Hungarian algorithm, see more methods in the "Vision-based tracking" course notes http://www.cse.psu.edu/~rtc12/CSE598C/datassocPart1.pdf

The overlapping between the detected in current frame with that of the previous frame

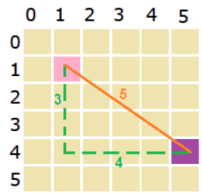
If there is no overlapping, we can calculate various distances based on center of object position.

The object positions are indicated by their bounding box.



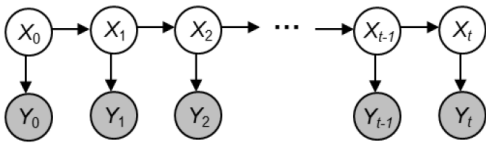
Euclidean distance: $\sqrt{(5-1)^2 + (4-1)^2} = 5$

Manhattan distance: $|5-1| + |4-1| = 7$



- motion model -> model the movement of object

We can use Kalman filter or particle filter.



  - State X: The actual state of the moving object that we want to estimate but cannot observe
  - E.g. center position of the object, aspect ratio a, width w, height h and their respective $vvvvvvvvviittiivvv$ of the bounding box.
  - Observations Y: Our actual measurement or observation of state X, which can be very noisy
  - At each time tt, the state changes to Xt to have a new observation Yt
  - Our goal is to recover the most likely state Xt given:
    - All observations so far, i.e. $Y1, Y2, \cdots, Yt$
    - Knowledge about dynamics of state transitions
- Appearance model -> Model the appearance of the object to evaluate the similarities.

  - What to track?
    - Bounding box, ellipse, contour, interest point, and silhouette
  - What visual representations are appropriate and robust for visual object tracking?
    - Hand-crafted features like histograms and color, CNN features

- Which similarity evaluation schemes are suitable for visual object tracking?
  - Techniques in single object tracking like cross correlation and SiameseFC can be used here
- Integrated -> DeepSort

  - Track initialization (e.g., using a deep learning-based detector)
  - Matching detections at two consecutive time stamps
  - Repeat for every pair of frames
- Evaluation metric

Evaluation metric: *Multiple Object Tracking Accuracy* (MOTA), it combines three error sources for every frame t, false negative fn_t, false positives fp_t, and identity switches id_sw_t.

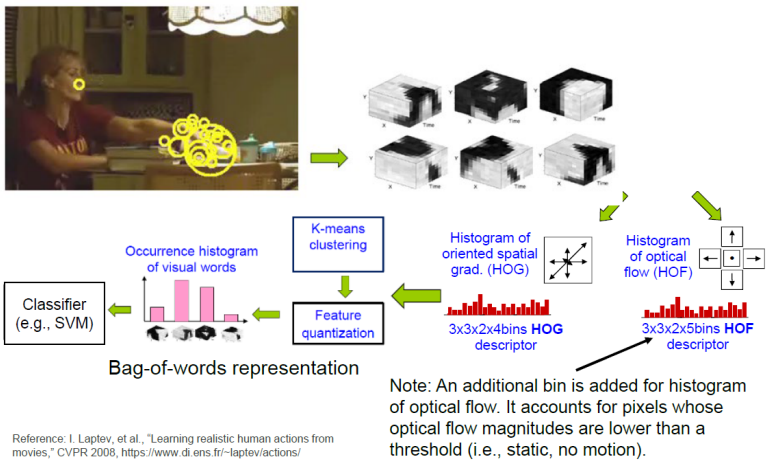
$$MOTA = 1 - \frac{\sum_t (fn_t + fp_t + id_sw_t)}{\sum_t g_t}$$

- How to initialize the object in the first frame to activate the tracker?
  - Manual initialization (different users have different targets)
  - Object detector (a list of pre-defined objects)
- Where to find labelled data for training a object tracker?
  - Unfortunately, there is no ImageNet dataset in object tracking research domain.
  - Fortunately, we don't need to re-train the tracker, as the tracker should be general to handle various types of objects.
- Speed is extremely important requirement for real-time tracker.

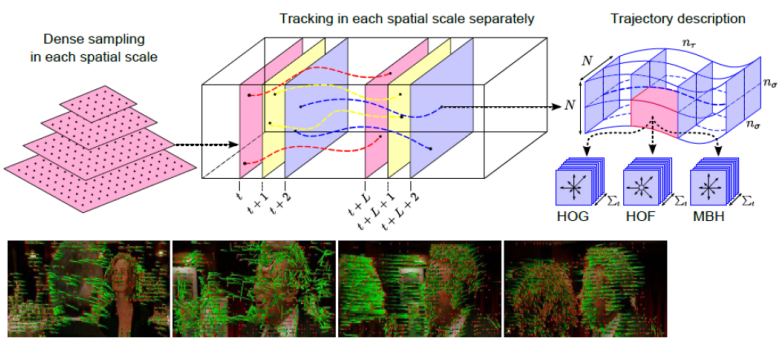
- Action recognition

Non-deeply learned representation	Sparse feature	HoF	I. Laptev, et al., “Learning realistic human actions from movies,” CVPR 2008, https://www.di.ens.fr/~laptev/actions/
	Dense feature	DT	H. Wang, et al., “Action recognition by dense trajectories,” CVPR 2011, https://hal.inria.fr/hal-00725627v2/document, H. Wang, et al., “Action recognition with improved trajectories,” ICCV 2013, https://hal.inria.fr/hal-00873267v2/document.
Deeply learned representation	Single-stream	CNN	A. Karpathy, et al., “Large-scale video classification with convolutional neural networks,” CVPR 2014, https://cs.stanford.edu/people/karpathy/deepvideo/
		LRCN	J. Donahue, et al., "Long-term recurrent convolutional networks for visual recognition and description," CVPR 2015, https://arxiv.org/abs/1411.4389
	Two-stream	Two-stream	K. Simonyan, et al., “Two-stream convolutional networks for action recognition in videos,” NIPS 2014, https://arxiv.org/abs/1406.2199
		Two-stream fusion	C. Feichtenhofer, et al., “Convolutional two-stream network fusion for video action recognition,” CVPR 2016, https://arxiv.org/abs/1604.06573
		TDD	L. Wang, et al., “Action recognition with trajectory-pooled deep-convolutional descriptors,” CVPR 2015, https://arxiv.org/abs/1505.04868
	3D CNN	C3D	D. Tran, et al., “Learning spatiotemporal features with 3D convolutional networks“, ICCV 2015, https://arxiv.org/abs/1412.0767
		i3D	Carreira, et al., “Action recognition? A new model and the Kinetics dataset”, CVPR 2017, https://arxiv.org/abs/1705.07750
		P3D	Qiu et al., Learning spatio-temporal representation with Pseudo-3D residual networks, ICCV 2017, https://arxiv.org/abs/1711.10305
		SlowFast	C. Feichtenhofer, et al., SlowFast networks for video recognition, ICCV 2019, https://arxiv.org/abs/1812.03982

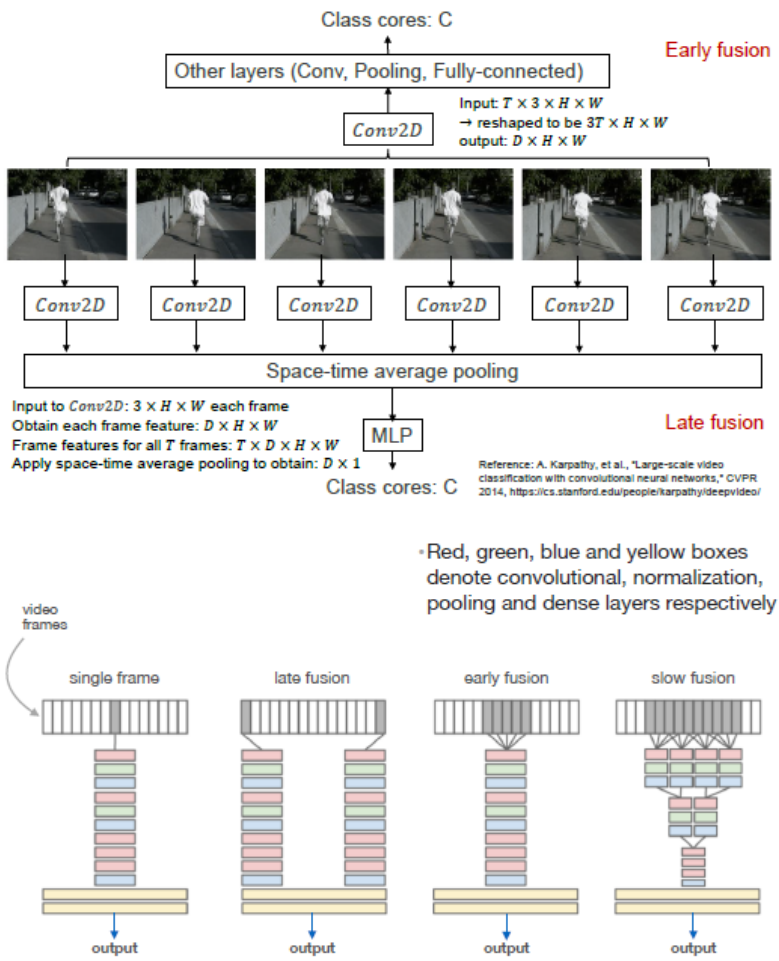
- traditional pipeline
  - 1.Extract local visual features
  - 2.Combine features into video level description
  - 3.Train a classifier (e.g. SVM) on the resulting "bag of words" representation
- HOF

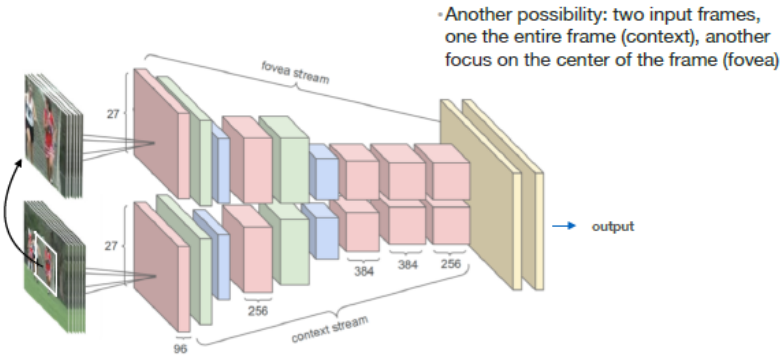


  - Detect local structures in space-time where the image values have significant local variations in both space and time.
  - At every keypoint, extract its space-time parch in its neighborhood.
  - Each space-time patch is subdivided into a $(nnxx, nnyy, nntt)$ grid ($3 \times 3 \times 2$ in the paper) of cuboids; for each cuboid we compute histograms of oriented gradient (HoG) and optic flow (HoF).
- DT



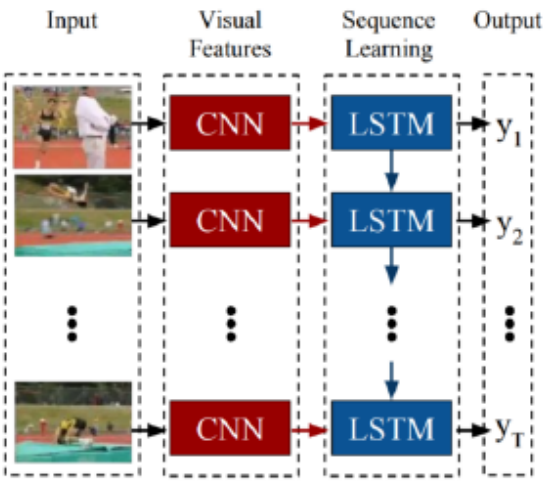
- Densely sample feature points on a grid spaced by a few pixels. The volume is subdivided into a spatio-temporal grid of $n\sigma \times n\sigma \times nt$ (2 × 2 × 3 in the paper) cuboids. We compute descriptor in each cuboid of the spatial-temporal grid.
- In both HOG and HOF, orientations are quantized into 8 bins, magnitudes are used for weighting. An additional zero bin is added for HOF (i.e., in total 9 bins). It accounts for pixels whose optical flow magnitudes are lower than a threshold. The final descriptor size is 96 for HOG (i.e., 2 × 2 × 3 × 8) and 108 for HOF (i.e., 2 × 2 × 3 × 9).
- Motion boundary histograms (MBH) descriptor
  - Compute optical flow for two consecutive frames. Separate the optical flow for the horizontal and vertical components of the optical flow.
  - Compute spatial derivatives for each of them and orientation information is quantized into histograms. The magnitude is used for weighting. We obtain a 8-bin histogram for each component.
  - The MBH descriptor encodes the relative motion between pixels (in the sense that the camera motion, which is same for all pixels, is somehow excluded).
- CNN





- Single Frame: Apply single-image-based CNN on a randomly chosen single frame.
- Early Fusion: Combines information across an entire sequence. The filters on the first convolutional layer is modified (e.g., $11 \times 11 \times 3 \times TT$) if 11×11 $ccccnncc2DD$ filter is applied for TT RGB frames.
- Late Fusion: Places separate single-frame network with shared parameters for consecutive or sampled frames, and then merges them.
- Slow Fusion: Slowly fuses temporal information throughout the network such that higher layers get access to progressively more global information in both spatial and temporal dimensions.
- simplest way:grab frame by frame,feed single frame to deep learning model, then get output
  - Simple and effective, but with a problem: unstable output
  - Why is this happening?
    - Because we analyze video frame by frame
    - Performance of deep learning model is not perfect, easily swayed by unseen objects or events in image
    - possible solution: voting average

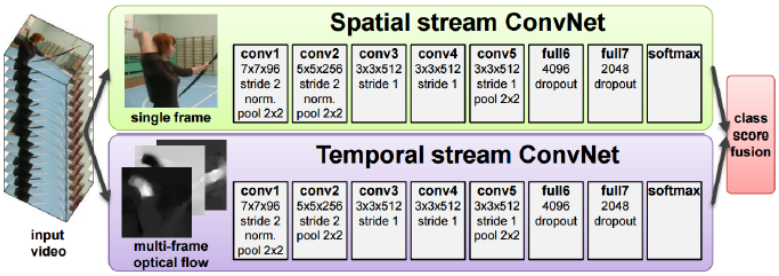
- LRCN



- LRCN processes the (possibly) variable-length visual input (left) with a CNN (middle-left), whose outputs are fed into a stack of recurrent sequence models

(LSTMs, middle-right), which finally produce a variable-length prediction (right). Both the CNN and LSTM weights are shared across time, resulting in a representation that scales to arbitrarily long sequences.

- Two-stream



  - video: Appearance + Motion
  - Single RGB frame: Static appearance
  - Multiple motion frames: Optical flow, pixel displacement as motion information
  - Spatial stream: Operates on (randomly) individual video frames, effectively performing action recognition from still images.
  - Temporal stream: Input to the model is formed by stacking optical flow displacement fields between several consecutive frames.
  - Two-stream fusion

Input	Method	Output
Two feature maps, each has a dimension of $H \times W \times D$	Sum fusion	$H \times W \times D$
	Max fusion	$H \times W \times D$
	Concatenation fusion	$H \times W \times 2D$
	Bilinear fusion	• Outer product at each pixel location: $H \times W \times D \times D$ • Summation over pixel locations: $D \times D$

    - Objective:Fuse the two networks(at a particular convolutional layer)such that channel responses at the same pixel position are put in correspondence.
    - Sum fusion:Computes the sum of two feature maps at the same spatial locations$iij$and feature channel

      d: $y_{i,j,d}^{\text{sum}} = x_{i,j,d}^{a} + x_{i,j,d}^{b}$

    - Max fusion:Takes maximum of the two feature map

      $y_{i,j,d}^{\text{max}} = \max\{x_{i,j,d}^{a}, x_{i,j,d}^{b}\}$

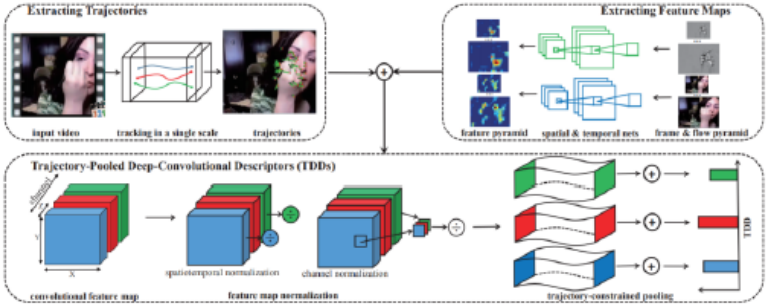
    - Concatenation fusion:Stacks the two feature maps at the same spatial locationsi,j across the feature channels

      d: $y_{i,j,2d}^{\text{cat}} = x_{i,j,d}^{a}, \quad y_{i,j,2d-1}^{\text{cat}} = x_{i,j,d}^{b}$

    - Bilinear fusion:Computes a matrix outer product of two features at each pixel location,followed by a summation over the locations

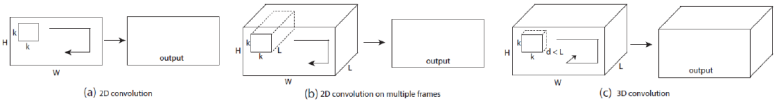
      $y_{i,j,d}^{\text{bil}} = \sum_i \sum_j \mathbf{x}_{i,j}^{a\,T} \mathbf{x}_{i,j}^{b}$

  - TDD



- Treat the learned two-stream ConvNets as generic feature extractors, and use them to obtain multi-scale convolutional feature maps for each video.
- Detect a set of point trajectories using dense trajectories in original video.
- Pool the local ConvNet responses over the spatiotemporal tubes centered at the trajectories, based on convolutional feature maps and trajectories.

- C3D



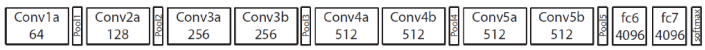
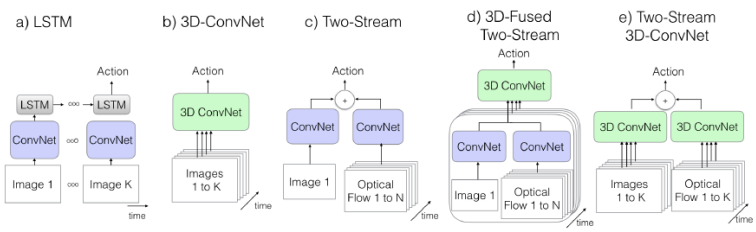


Figure 3. **C3D architecture**. C3D net has 8 convolution, 5 max-pooling, and 2 fully connected layers, followed by a softmax output layer. All 3D convolution kernels are $3 \times 3 \times 3$ with stride 1 in both spatial and temporal dimensions. Number of filters are denoted in each box. The 3D pooling layers are denoted from `pool1` to `pool5`. All pooling kernels are $2 \times 2 \times 2$, except for `pool1` is $1 \times 2 \times 2$. Each fully connected layer has 4096 output units.

Method	Model architecture example	Size ($C \times T \times H \times W$)
Early Fusion (2D CNN)	Input	$3 \times 20 \times 64 \times 64$
	Conv2D (3×3, 12 kernels)	$12 \times 64 \times 64$
	Pool2D (4×4)	$12 \times 16 \times 16$
	Conv2D (3×3, 24 kernels)	$24 \times 16 \times 16$
	GlobalAvgPool	$24 \times 1 \times 1$
Late Fusion (2D CNN)	Input	$3 \times 20 \times 64 \times 64$
	Conv2D (3×3, 12 kernels)	$12 \times 20 \times 64 \times 64$
	Pool2D (4×4)	$12 \times 20 \times 16 \times 16$
	Conv2D (3×3, 24 kernels)	$24 \times 20 \times 16 \times 16$
	GlobalAvgPool	$24 \times 1 \times 1 \times 1$
3D CNN (used in C3D)	Input	$3 \times 20 \times 64 \times 64$
	Conv3D ($3 \times 3 \times 3$, 12 kernels)	$12 \times 20 \times 64 \times 64$
	Pool3D ($4 \times 4 \times 4$)	$12 \times 5 \times 16 \times 16$
	Conv3D ($3 \times 3 \times 3$, 24 kernels)	$24 \times 5 \times 16 \times 16$
	GlobalAvgPool	$24 \times 1 \times 1 \times 1$

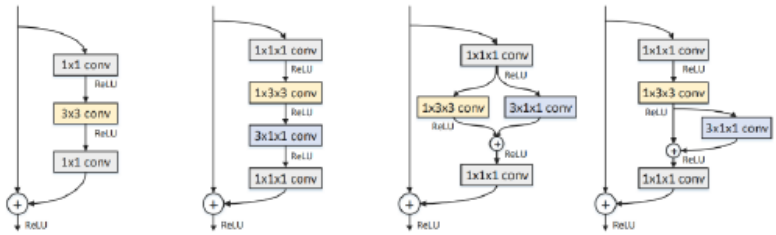
- 3D ConvNets are more suitable for spatiotemporal feature learning compared to 2D ConvNets.
- The learned features, namely C3D (Convolutional 3D), can be further integrated with other classifiers.

- i3D



- Question: Can we leverage the pre-trained 2D ConvNet for video?
- Solution: Inflated 3D ConvNet (I3D) is based on 2D ConvNet inflation. Filters and pooling kernels of image classification 2D ConvNets are expanded into 3D, making it possible to learn seamless spatio-temporal feature extractors from video while leveraging successful 2D ConvNet architecture designs and even their parameters. Replace 2D CNN with 3D CNN version. Duplicate pre-trained 2D CNN parameters to be 3D CNN parameters.

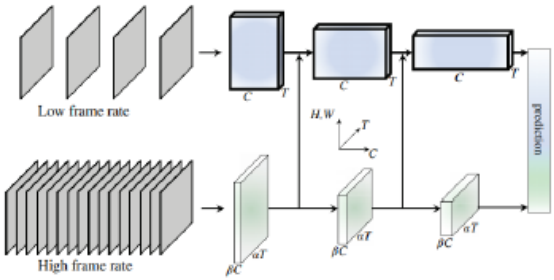
- P3D



Conventional ResNet Proposed P3D ResNet with three variants

- Question:CanweaddressthecomputationalcomplexityofConv3D?
- Solution:Pseudo-3D Residual Net (P3D ResNet).Convert 3×3×3 convolutions with 1×3×3 convolutional filters on spatial domain(equivalent to 2D CNN)plus 3×1×1convolutions to construct temporal connections on adjacent feature maps in time.

- SlowFast

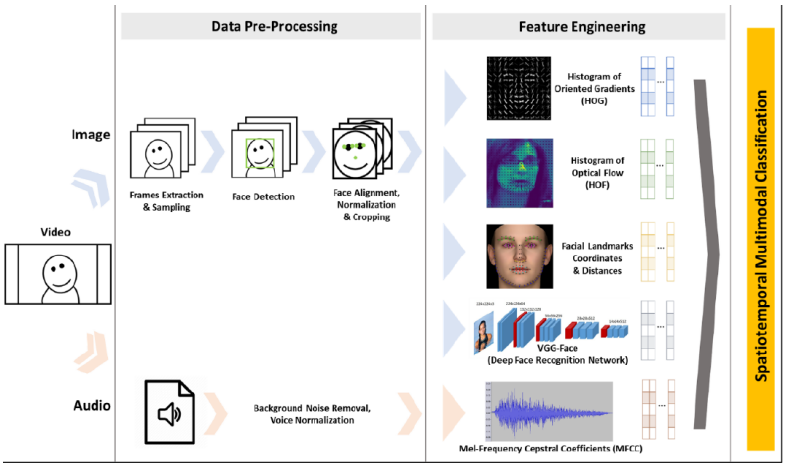


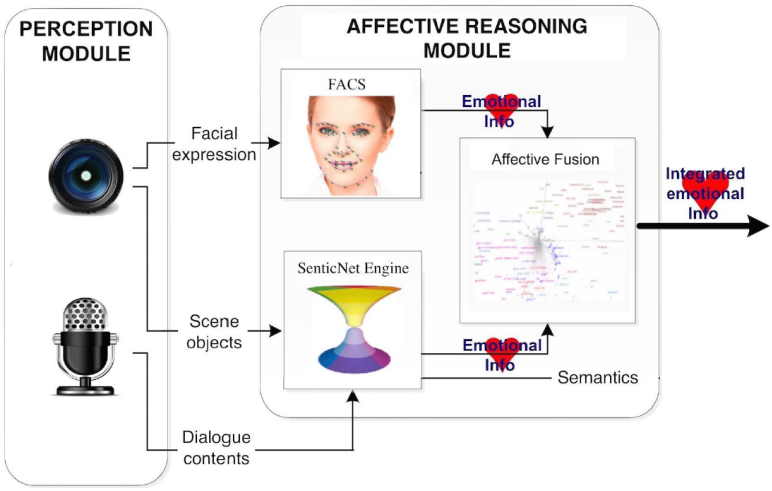
α: Adjust frame rate
β: Adjust convolutional channels (for consideration of computational complexity)

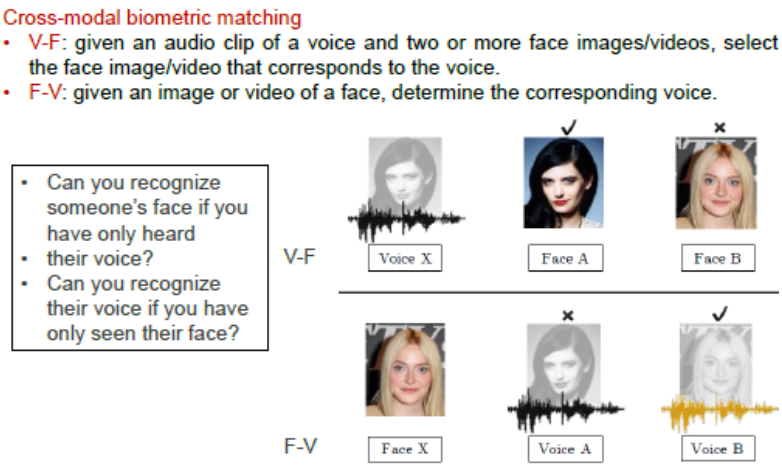
- Two streams with different frame rates
- A low frame rate,low temporal resolution Slow pathway
- A high frame rate,higher temporal resolution Fast pathway.The Fast pathway is light weight by using a fraction of convolutional channels.

-

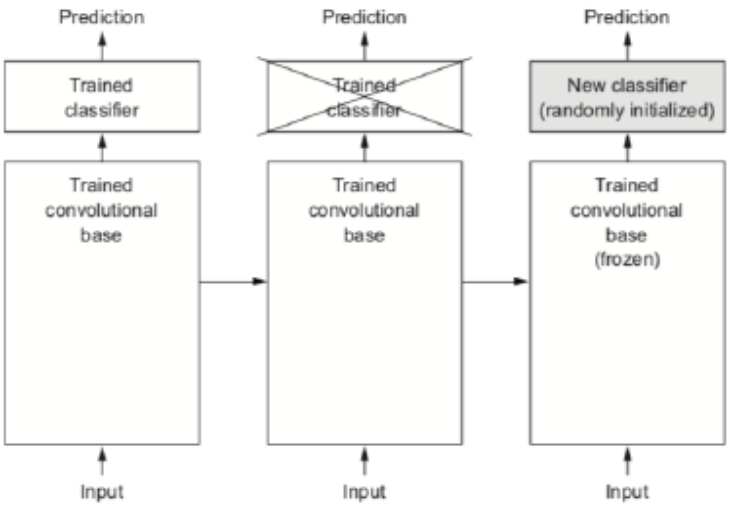
- Combination
  - example





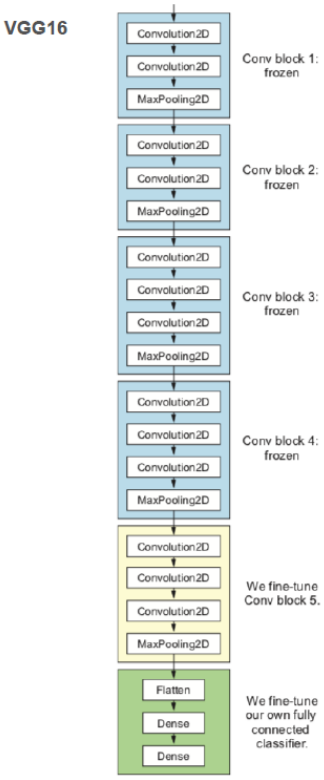


- Transfer learning
  - A common and highly effective approach on small image datasets
  - Use a pre-trained network that was previously trained on a large-scale dataset
  - If the original dataset is large and general enough, the pre-trained model may have actually learned how to extract the generic features(e.g. edges, lines, shapes, texturesand etc.) that truly matters to vision
  - Thus, the way the net extracts features is useful to many different vision problem
  - feature extraction and fine-tuning
    - feature extraction



- Representations learned by convnet base likely more generic, thus reusable
- Representations learned by classifier more specific to the set of classes the model was arranged to be trained on, not so reusable

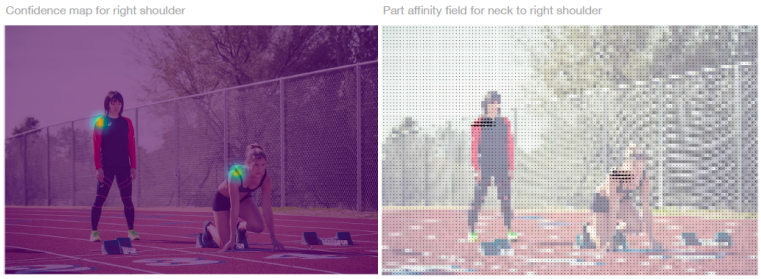
- fine tuning



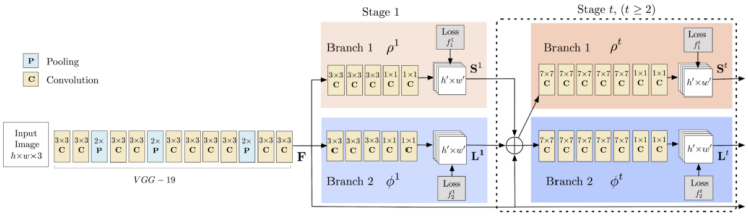
- After we have trained the added classifier with the convnet base frozen, we can unfreeze a few layers of the frozen base model that are near to the classifier
- train the newly unfreezed layers together with the classifier
- Must the classifier be trained before we perform a fine-tuning?
  - Yes, else the errors back propagated will be too large and distort the weights of the just unfreezed layers
  - Note: we are doing fine-tuning on both the unfreezed layers and classifier, not re-train the both
- Can unfreeze and fine-tune more layers?

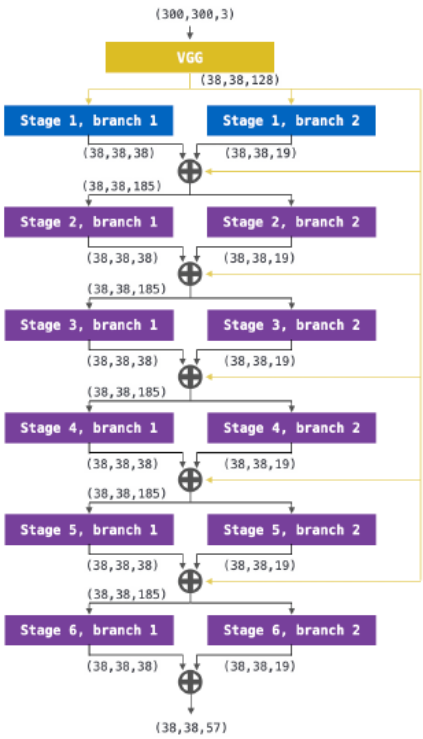
- Things to consider: Earlier layers in the base net (layers that are near to input) encode more generic and more reusable features; this is something you want to keep
- Later layers are more specific, you want to fine-tune them to make them fit for your problem
- Pre-trained networks usually have much more parameters and are much more powerful (that's why you want to use them); re-train more layers will easily lead to overfitting on small dataset

- Pose Estimation
  - OpenPose
    - Step 1, generate Confidence Maps and Part Affinity Fields from deep learning net

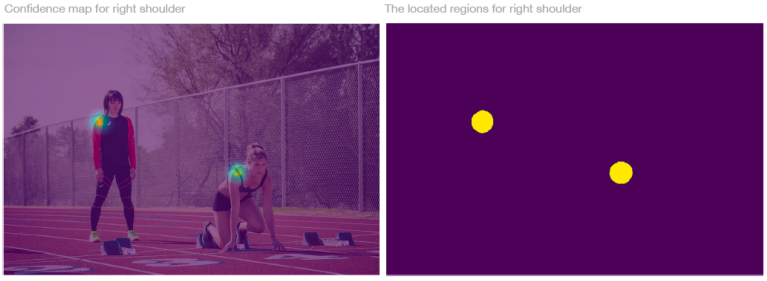


- Confidence maps show us where the anatomical points are likely to be
- Part affinity fields show us how anatomical points should connect with each other points
- Net

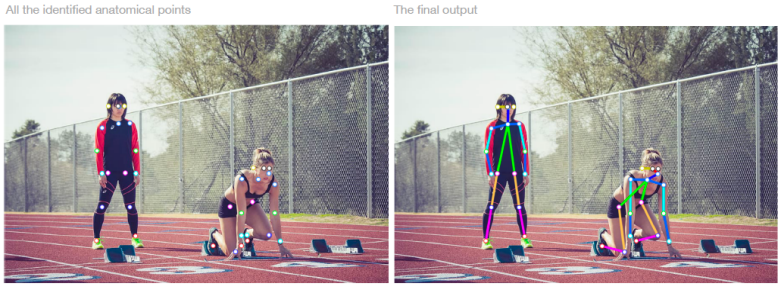




- Step 2, identify the anatomical points from the Confidence Maps



  - Confidence map gives a region of possible locations for each anatomical point, there is a need to reduce these regions into specific points
  - To remove those scattered values,image smoothing is required and probability values below a certain threshold to be discarded
  - Loop through each region to find the point that has the maximum value
  - The point with the maximum value will be taken as an anatomical point
- Step 3, pair the anatomical points and form human subjects



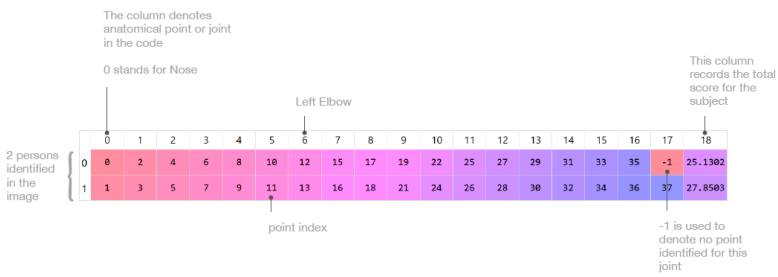
  - A need to establish how points are linked to each other
  - Solution: For each anatomical link (say, neck to right shoulder), for each identified neck point, loop through every neck-right shoulder links, and

evaluate each link against Part Affinity Fields

- Human-link Method

  - 5.Calculate the average PAF score (from the 10 vectors)
  - 6.Determine which vector has a PAF score higher than pafThres (a threshold)
  - 7.If there are more than 7 vectors, each with a PAF score higher than pafThres, this is likely a true link (a candidate)
  - 8.If the average PAF score is the highest among the candidates, this is a true link

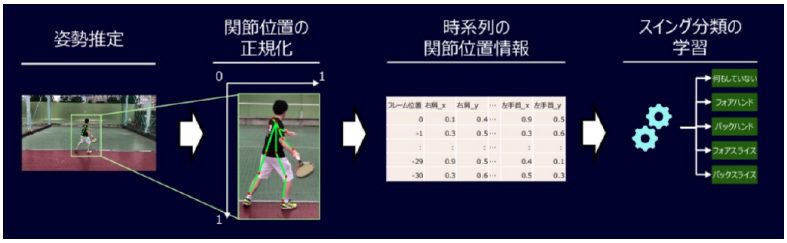
- Get the human subjects



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
0	0	2	4	6	8	10	12	15	17	19	22	25	27	29	31	33	35	-1	25.1302
1	1	3	5	7	9	11	13	16	18	21	24	26	28	30	32	34	36	37	27.8503

  - With all links identified, it is then possible to group links to form human subjects
  - The output persons is a n x 19 numpy array. n is the number of human subjects identified in the image.

- Video openpose



  - Step 1: Perform pose estimation in each frame
  - Step 2: Normalize the positions of key anatomical points with respect to the player in each frame
  - Step 3: Send the normalized positions from 15 frames to lightGBM model to classify 5 classes: idle, backhand, forehand, fore slice, back slice

- Real-time
  - Perceptual Sense
    - Real-time in the perceptual sense is used mainly to describe the interaction between a human and a computer device for a near instantaneous response of the device to an input by a human user.
    - the result of processing appears effectively ‘instantaneously’ (usually in a perceptual sense) once the input becomes available.
    - digital processing of an image which occurs seemingly immediately; without a user-perceivable calculation delay.
  - Signal Processing Sense
    - completing the processing within the allowable or available time between samples