# OpenPose – Realtime 2D Human Pose Estimation (Guest Lecture)

LVA Visual Analysis of Human Motion (188.469)
Summer Semester 2023
Andreas Kriegler

# Modalities

- Contacts:
  - andreas.kriegler@tuwien.ac.at
  - margrit.gelautz@tuwien.ac.at

- You can just interrupt me or speak freely to ask a question
- Slides will be available in the TUWEL course

- Machine learning (ML) for Computer Vision builds on decades of mathematics - here it is packed into ~15 mins → this is a high-level and incomplete picture
- DL basics are a modern necessity for Computer Vision, in research *and* industry

If you want to help in keeping the number of errors at a reasonable level send any that you find to andreas.kriegler@tuwien.ac.at

# Literature

- ML & DL use applied statistics, linear algebra & calculus (books):
  - Mathematical foundations and many common algorithms of machine learning – the ML “bible”: [1]
  - The application of deep learning in neural networks: [2], [3]
  - Artificial intelligence in general and multi-agent theory: [4]
  - The necessity of statistics for robotics applications – probabilistic robotics: [5]
  - A number of applied/practical books for developing modern vision systems with DL frameworks (PyTorch, Keras, TF, …)
- Lectures:
  - TU Wien - 194.100 Theoretical Foundations and Research Topics in Machine Learning [6]
  - Stanford - CS229 Machine Learning [7]
  - Stanford - CS231n Convolutional Neural Networks for Visual Recognition [8]
  - MIT - Deep Learning and Artificial Intelligence Lectures [9]
- Videos:
  - Deep Learning Series, 3Blue1Brown [10]
  - Mathematics for Machine Learning, Ulrike von Luxburg [11]

# Machine learning basics

- In classical programming we define rules for input → output relations

- In ML we use data to

  1. Generate our model (**training** / learning)

  2. Apply it to new observations (**testing / inference** / prediction)

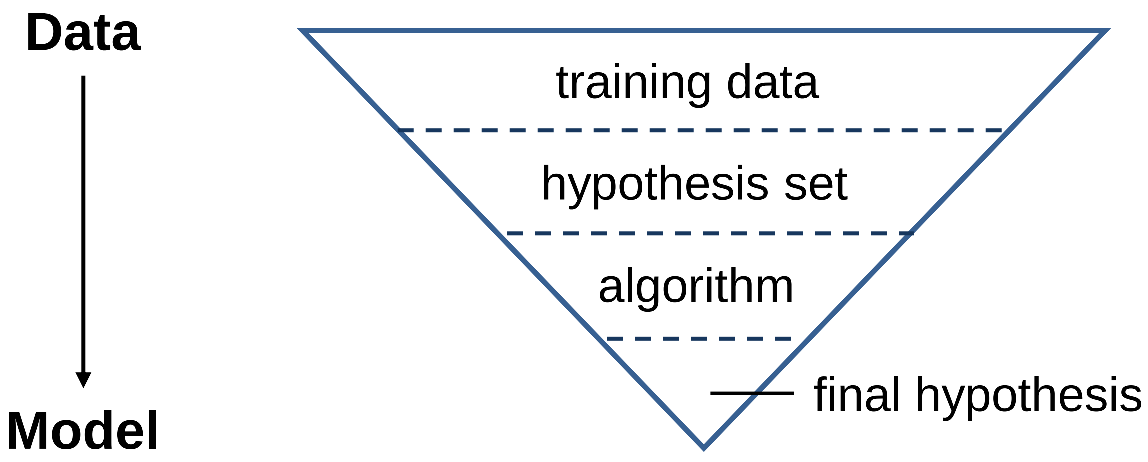



Figure recreated from [12]

# Machine learning basics

- ML often based on calculating the **gradient** of a (convex) **loss-function**
- No „magic“ in ML/DL: find **minima** of objective / cost / error / target / loss function
- Try to find the **weights (parameters)** of the model that minimizes the cost
- **BUT** modern DL systems, esp. CNNs, rely on many other components found to work best in practice

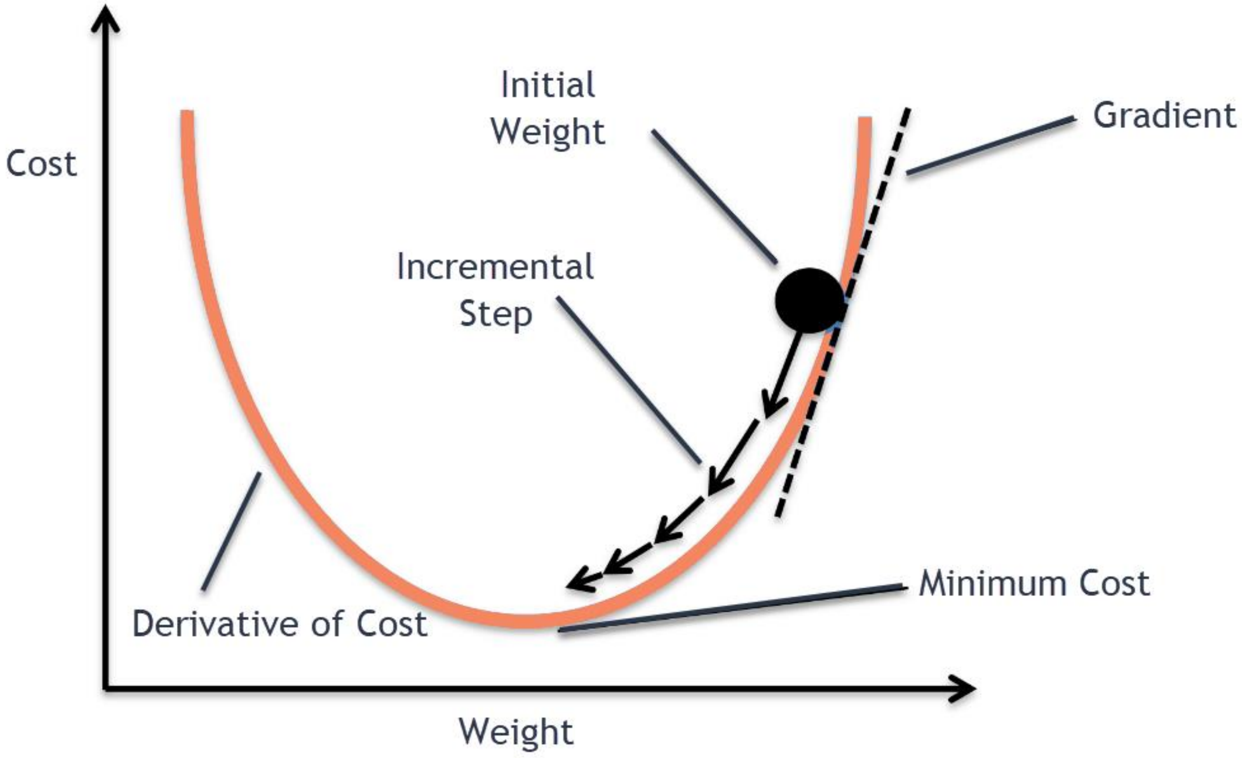



Figure taken from [13]

# ML – supervised classification (transcribed [1])

- We have: a **training set** of $N$ observations of $x$, $\boldsymbol{x} \coloneqq (x_1, \ldots, x_N)^\mathsf{T}$ with corresponding **target** observations $t$, $\boldsymbol{t} \coloneqq (t_1, \ldots, t_N)^\mathsf{T}$
- We want: predict $t$ for new $x$, using parameters $\boldsymbol{\theta}$
- **Regression** if $\boldsymbol{t} \subset \mathbb{R}$ or $n$-way **classification** if $\boldsymbol{t}$ is categorical $t \in \{0, \ldots, n-1\}$
- We can, for example, try to fit a polynomial curve

$$f = y(x, \theta) = \theta_0 + \theta_1 + \theta_2 x^2 + \cdots + \theta_M x^M = \sum_{j=0}^{M} \theta_j x^j \qquad (1)$$

- We can calculate any loss/error function - often L1 or L2

$$(2) \quad L_1(\theta) = \frac{1}{2} \sum_{n=1}^{N} (|y(x_n, \theta) - t_n|) = ||\theta||_1 \qquad L_2(\theta) = \frac{1}{2} \sum_{n=1}^{N} (|y(x_n, \theta) - t_n|)^2 = ||\theta||_2 \quad (3)$$

- Now we can update – **optimize** – parameters $\boldsymbol{\theta}$: **gradient descent** techniques [14]

# Deep learning – artificial neural networks

- Deep learning with multilayer perceptrons (MLP) since 1965 [15]

- Dealing with vanishing gradients since 1991 [16, 17], deep since 2012 [18]

- Now models with up to $10^{\cancel{12}\ 14}$ parameters trained on cloud-based tensor or graphical processing unit (TPU/GPU) clusters ~~[19]~~ [25]

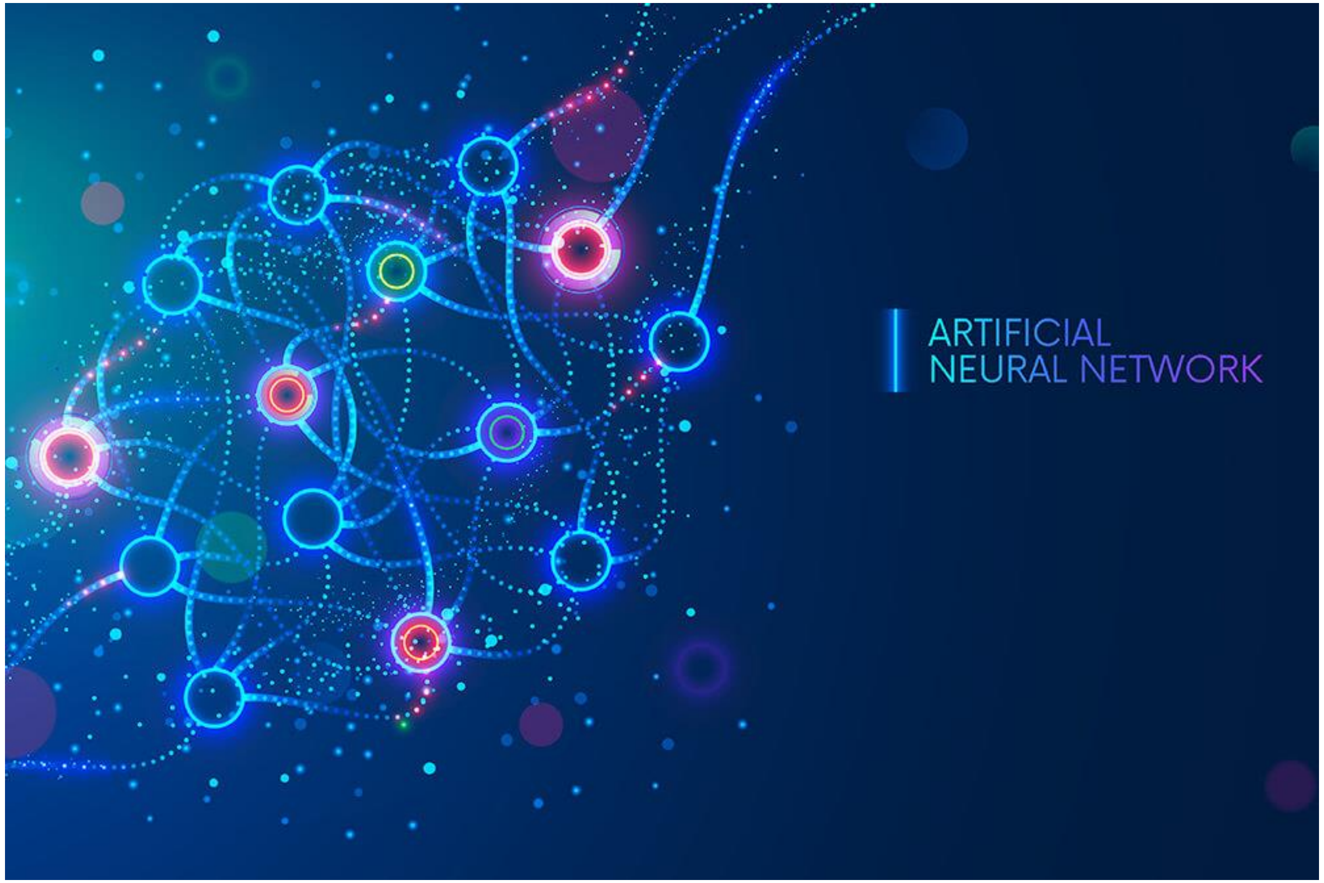
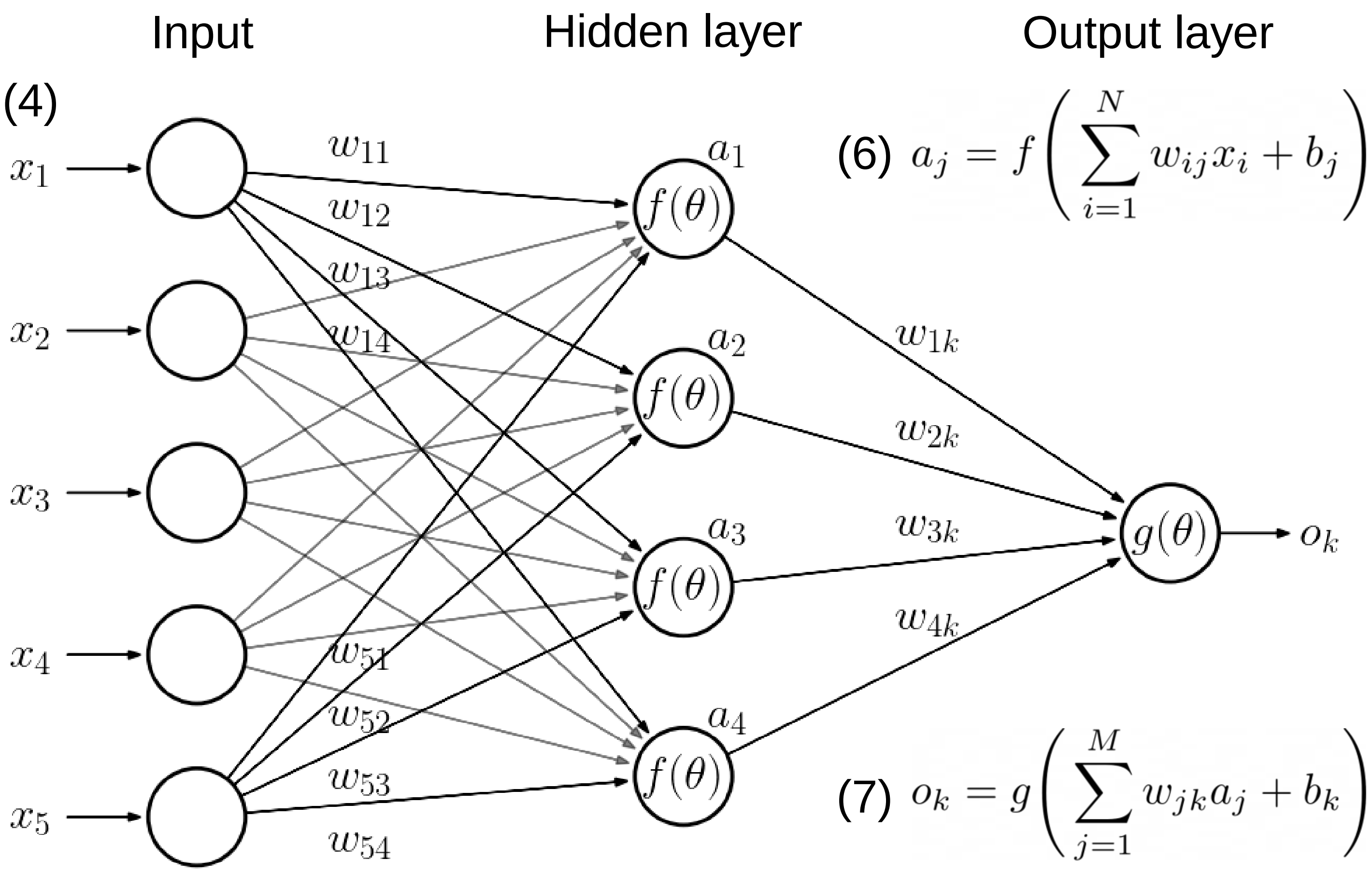



Figure taken from [20]

# Deep learning – 1-hidden layer neural network

- **Activations $f$** in hidden neurons
  - Sigmoid (old): $\phi(z) = \frac{1}{1+e^{-z}}$ (4)
  - **ReLU** (rectified linear unit):
    $ReLU(z) = max(0, z)$ (5)
- **Weights** $w$ scale the function and **bias** b shifts it
- $N, M$: number of input neurons – here 5 and 4
- Multiple output neurons form a output vector




(6) $a_j = f\left(\sum_{i=1}^{N} w_{ij}x_i + b_j\right)$

(7) $o_k = g\left(\sum_{j=1}^{M} w_{jk}a_j + b_k\right)$

Figure taken from [21]

# Supervised Convolutional Neural Networks (CNNs)

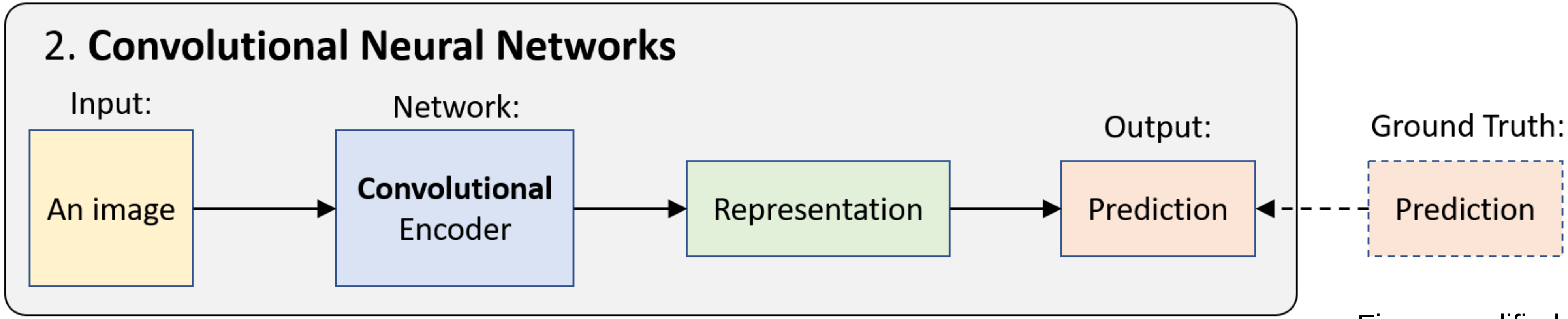
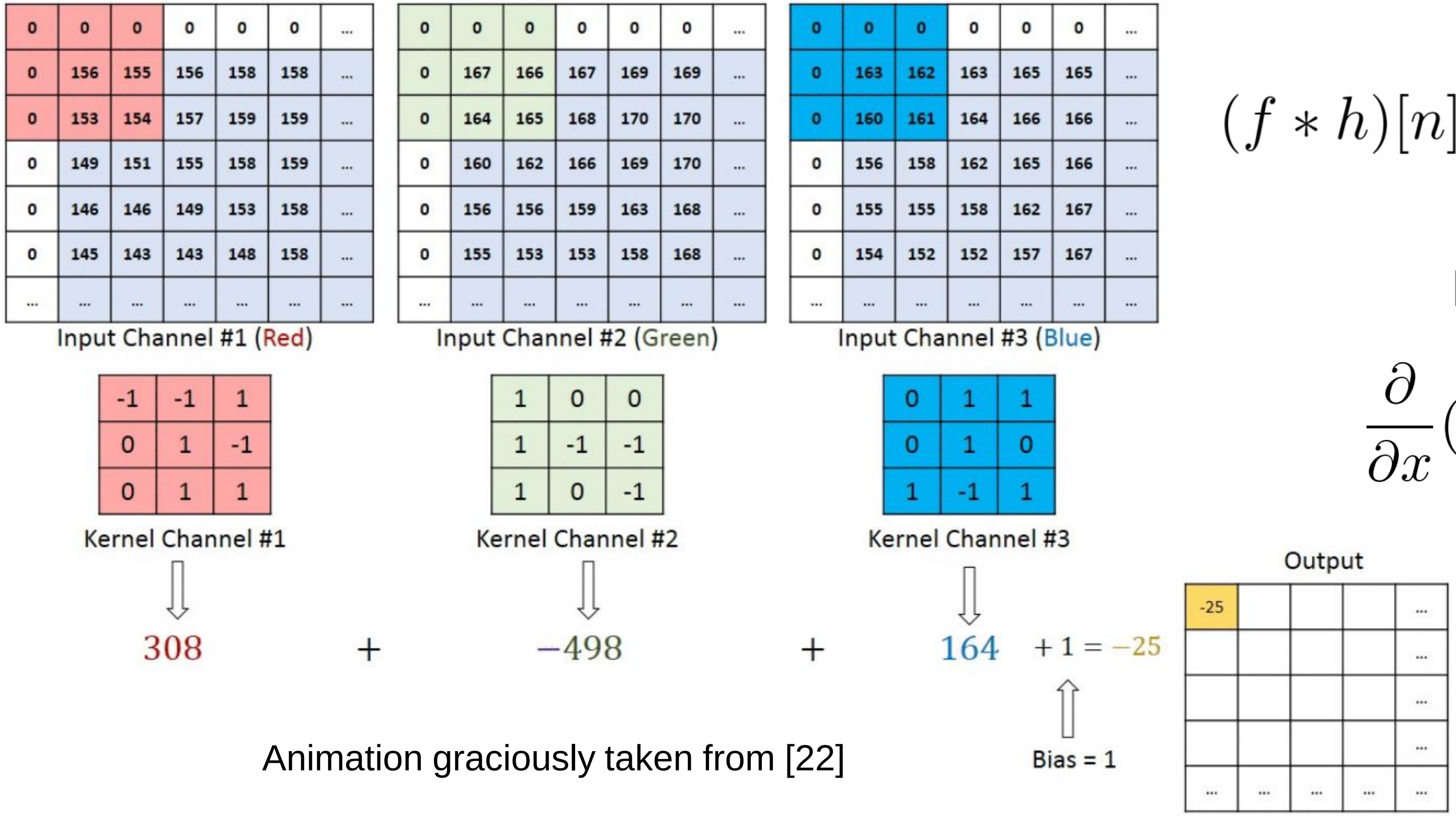



Figure modified and taken from [9]

1. We operate on images, particularly the **RGB** channels
2. The **convolutional layers** act as **encoders** for feature representations
3. From those representations we obtain predictions
4. We compare predictions with the **ground truth (gt)** via a loss function
5. Backpropagate the loss using gradient descent
6. Update our weights, i.e. **kernel entries**

# Convolution operation in CNNs

- Invented in 1979 [21] – see [8] for the Stanford course
- We learn kernel entries using backpropagation




Animation graciously taken from [22]

Discrete convolution (cross-correlation)

$$(f * h)[n] = \sum_{m=-M}^{M} f[n-m]h[m] \quad (8)$$

Easily differentiable

$$\frac{\partial}{\partial x}(h * f) = (\frac{\partial}{\partial x}h) * f \quad (9)$$

$*$ is typically used for the convolution (sum-of-products) operation

# Convolution operation in CNNs

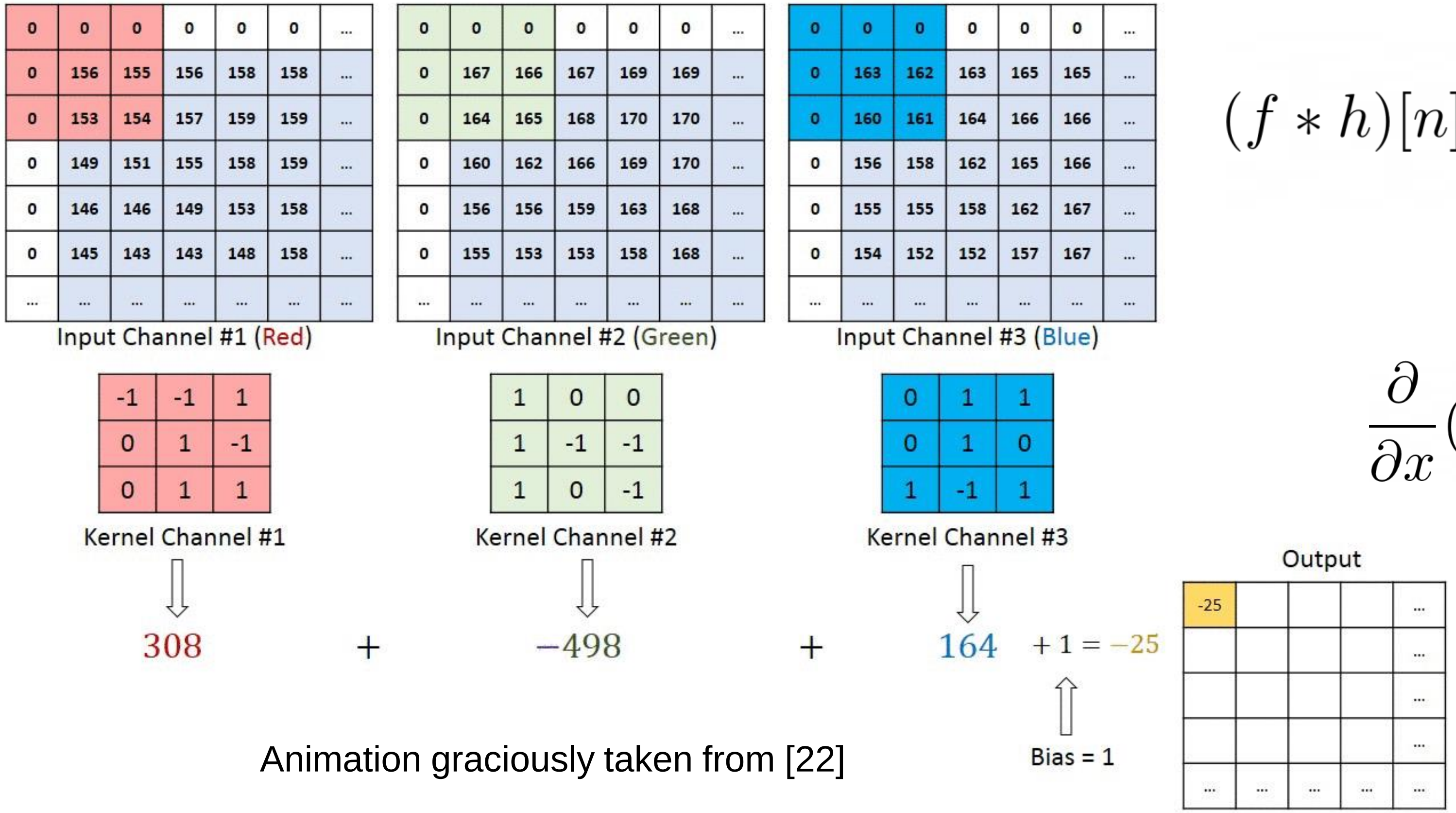
- Invented in 1979 [21] – see [8] for the Stanford course
- We learn kernel entries using backpropagation

Discrete convolution (cross-correlation)

$$(f * h)[n] = \sum_{m=-M}^{M} f[n-m]h[m] \quad (8)$$

Easily differentiable

$$\frac{\partial}{\partial x}(h * f) = (\frac{\partial}{\partial x}h) * f \quad (9)$$




Animation graciously taken from [22]

$*$ is typically used for the convolution (sum-of-products) operation

# CNNs as powerful feature extractors

- Augmented RGB images
- The features become increasingly complex
- Visualizations on the right [23] are „projected activations of selected feature maps“
- No heatmaps but highlights of contributing features

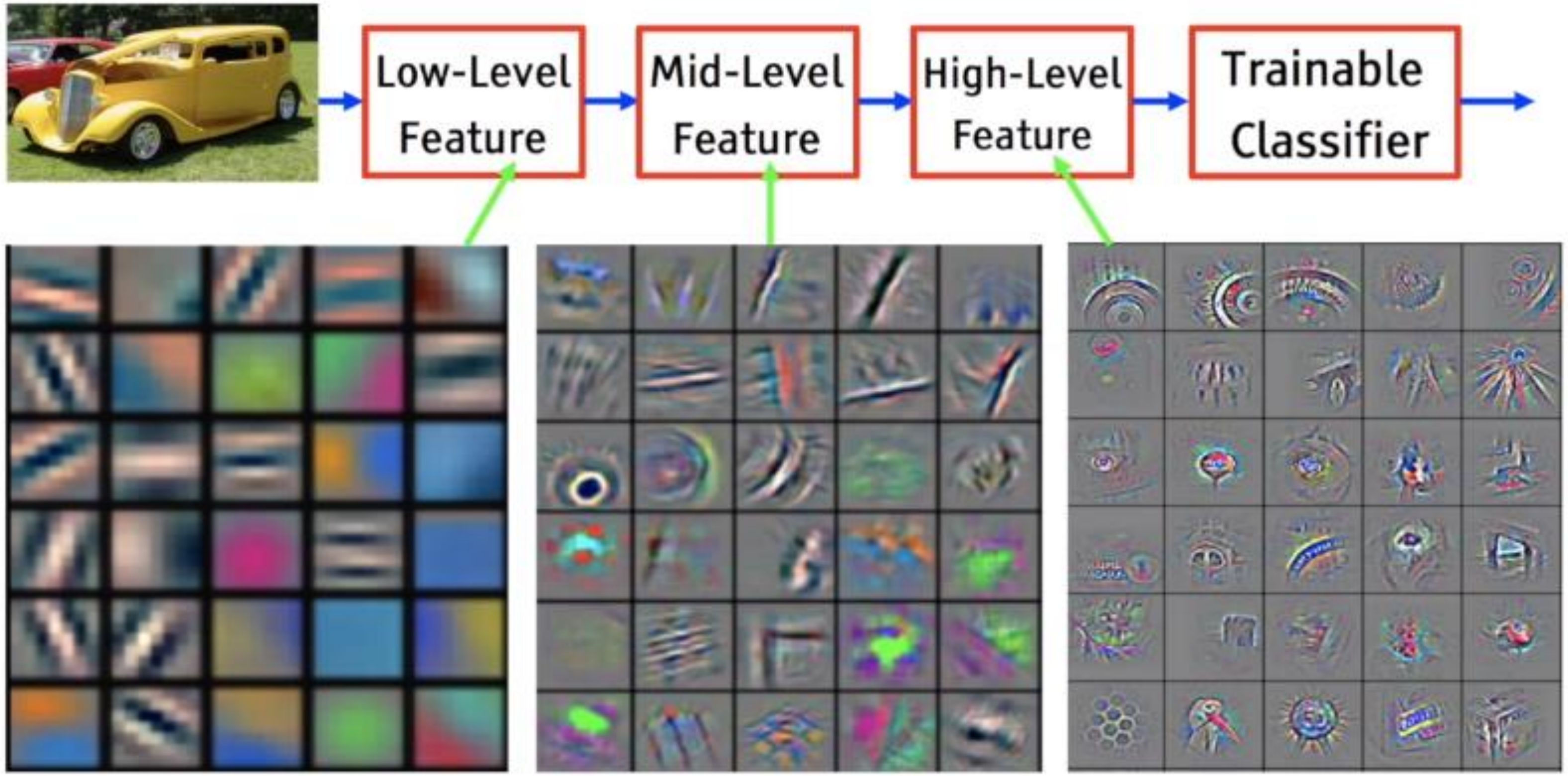



Figure taken from [23]

# OpenPose [24] - Preliminaries

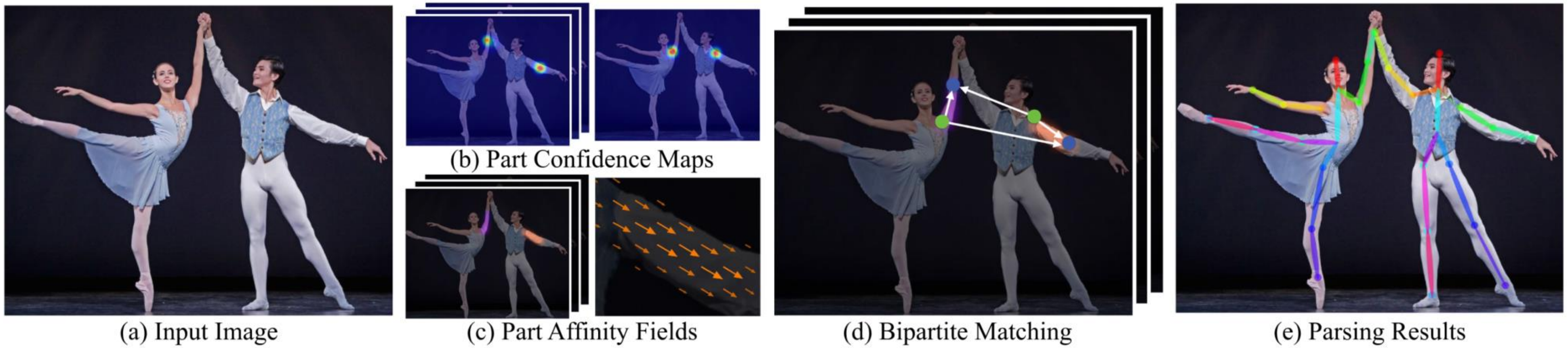
- 2D human pose estimation, i.e. skeletal structure in images
  - Core component for understanding people in images/videos
  - Very challenging in multi-person scenario:
    - Unknown number of people
    - Interactions between people induce complex spatial interference
    - Runtime complexity typically depends on number of people

- Top-down: Person detector -> pose estimation for every detection
  - Potentially very slow runtime

- Bottom-up: Find anatomic parts -> build skeleton & pose
  - Cannot use global context yet has to match parts to person correctly

- OpenPose is a bottom-up approach

# OpenPose pipeline




- Colour input image to 2D locations of anatomical keypoints

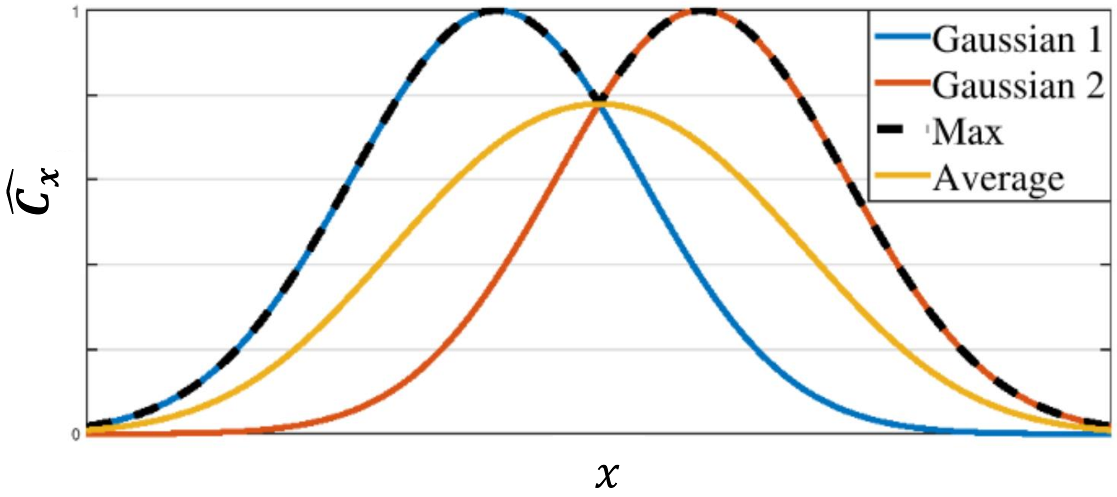
1. Confidence maps (body part locations)
2. PAFs (part affinitiy fields – association between parts -> limbs)
3. Staging
4. Bipartite matching for multi-person parsing

# Confidence maps – beliefs for parts (joints)

- Set $\mathbf{C} = (\mathbf{C_1}, \mathbf{C_2}, \dots, \mathbf{C_n})$ of n confidence maps, one per part
- How to get ground truth confidence map $\hat{\mathbf{C}}$?
- Use annotated 2D keypoints $\widehat{l_{n,k}}$ for belief for body part $n$ of person $k$ at pixel $x$, σ controls peak spread

(10) $$\widehat{\mathbf{C}}_{n,k}(\mathbf{x}) = \exp\left(-\frac{||\mathbf{x} - \widehat{\mathbf{l}}_{n,k}||_2^2}{\sigma^2}\right)$$

(11) $$\widehat{\mathbf{C}}_n(\mathbf{x}) = \max_k \widehat{\mathbf{C}}_{n,k}(\mathbf{x})$$




- Testing: detect body part candidates via non-maximum supression

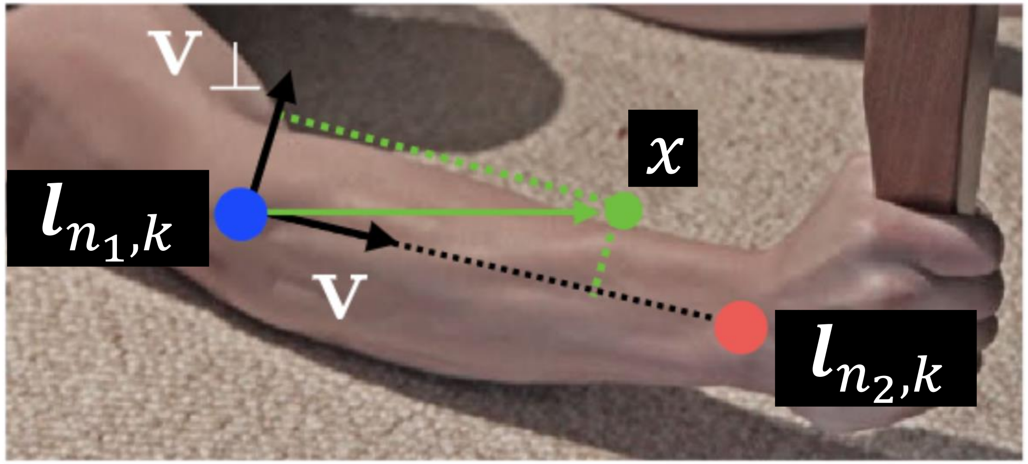
# Part Affinity Field PAF – joining parts to limbs

- Set $\mathbf{P} = (\boldsymbol{P}_1, \boldsymbol{P}_2, \dots, \boldsymbol{P}_\mathbf{m})$ of $m$ PAFs, one per limb
- 2D direction vectors in pixels close to line segment joining parts
- How to get ground truth PAF $\widehat{\boldsymbol{P}}$? - $z_m(\boldsymbol{x})$ number of non-zero vectors

(12) $$\widehat{\mathbf{P}}_{m,k}(\mathbf{x}) = \begin{cases} \mathbf{v} & \text{if } \mathbf{x} \text{ on limb } m, k \\ \mathbf{0}, & \text{otherwise.} \end{cases}$$

(13) $$\mathbf{v} = \frac{(\mathbf{l}_{n_2,k} - \mathbf{l}_{n_1,k})}{||\mathbf{l}_{n_2,k} - \mathbf{l}_{n_1,k}||_2}$$

(14) $$\widehat{\mathbf{P}}_m(\mathbf{x}) = \frac{1}{z_m(\mathbf{x})} \sum_k \widehat{\mathbf{P}}_{m,k}(\mathbf{x})$$




- „limb point“ conditions

(15) $$0 \leq \mathbf{v} \cdot (\mathbf{x} - \mathbf{l}_{n_1,k}) \leq ||\mathbf{l}_{n_2,k} - \mathbf{l}_{n_1,k}||_2$$

(16) $$|\mathbf{v}_\perp \cdot (\mathbf{x} - \mathbf{l}_{n_1,k})| \leq r_l$$ $r_l$ limb width

# PAFs – joining parts to limbs

- Set $\mathbf{P} = (\boldsymbol{P}_1, \boldsymbol{P}_2, \ldots, \boldsymbol{P}_\mathbf{m})$ of PAFs, one per limb
- 2D direction vectors in pixels close to line segment joining parts
- Testing: measure alignment of predicted PAF with candidate limb formed by connecting detected body parts
- Two candidate locations $\boldsymbol{d}_{n_1}$and $\boldsymbol{d}_{n_2}$, sample PAF along line segment

$$E = \int_{u=0}^{u=1} \mathbf{P}_c(\mathbf{x}(u)) \cdot \frac{\mathbf{d}_{n_2} - \mathbf{d}_{n_1}}{||\mathbf{d}_{n_2} - \mathbf{d}_{n_1}||_2} du \quad (17)$$

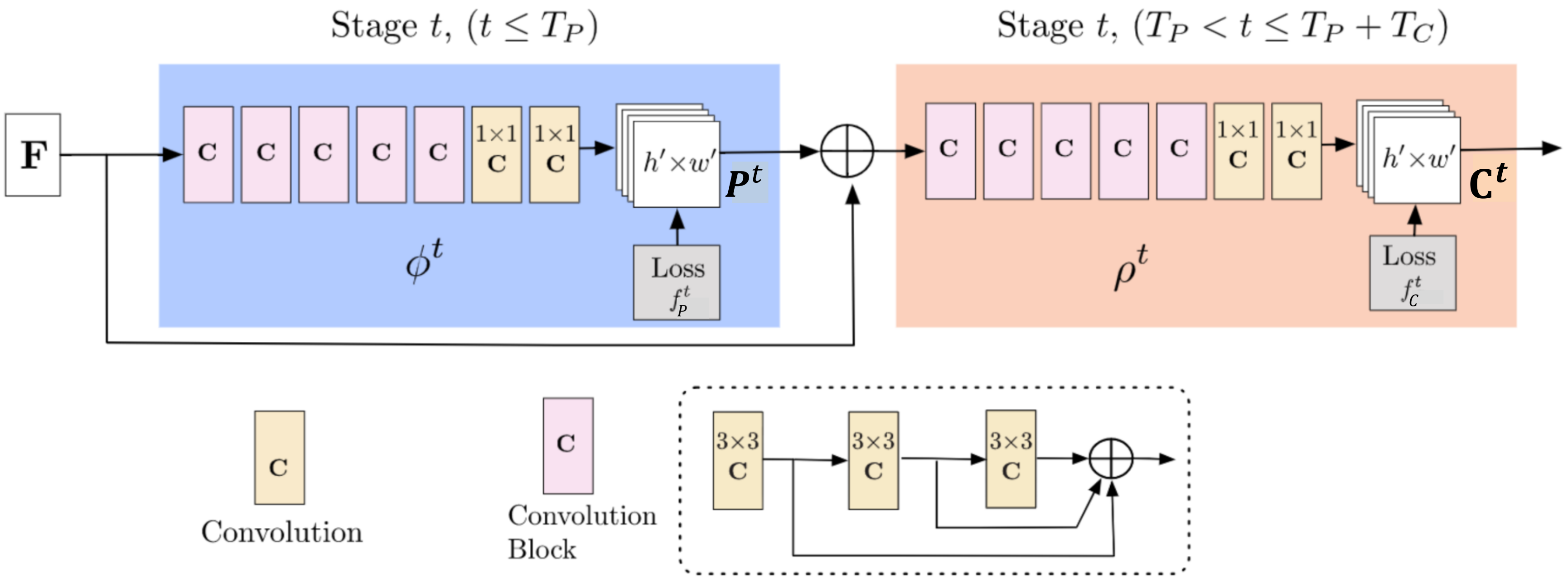
$$\mathbf{x}(u) = (1-u)\mathbf{d}_{j_1} + u\mathbf{d}_{j_2} \quad (18)$$

- Approximate integral by sampling and summing uniformly-spaced $u$

# Obtaining $\mathbf{P}$ and $\mathbf{C}$

- Input: Set of feature maps $\boldsymbol{F}$ obtained from VGG-19 CNN
- Start with $\boldsymbol{P}^1 = \phi^1(\boldsymbol{F})$ where $\phi^1$ is inference CNN at stage 1
- Then, for $T_P$ PAF stages: $\mathbf{P}^t = \phi^t(\mathbf{F}, \mathbf{P}^{t-1}), \forall 2 \leq t \leq T_P$ (19)
- Afterwards, for $T_C$ with $\rho^t$, do staging for $\mathbf{C}$, using most recent PAF $\boldsymbol{P}^{T_P}$

(20) $\mathbf{C}^{T_P} = \rho^t(\mathbf{F}, \mathbf{P}^{T_P}), \forall t = T_P$ $\mathbf{C}^t = \rho^t(\mathbf{F}, \mathbf{P}^{T_P}, C^{t-1}), \forall T_P < t \leq T_P + T_C$ (21)



# Learning $\mathbf{P}$ and $\mathbf{C}$

- Iterative learning: apply $L_2$ loss at end of every stage
- Weight loss with binary mask $\boldsymbol{W}$ to solve minor practical issue

$$f_{\mathbf{P}}^{t_i} = \sum_{m=1}^{M} \sum_{X} \mathbf{W}(\mathbf{x}) \cdot ||\mathbf{P}_m^{t_i}(\mathbf{x}) - \widehat{\mathbf{P}}_m(\mathbf{x})||_2^2 \quad (22)$$

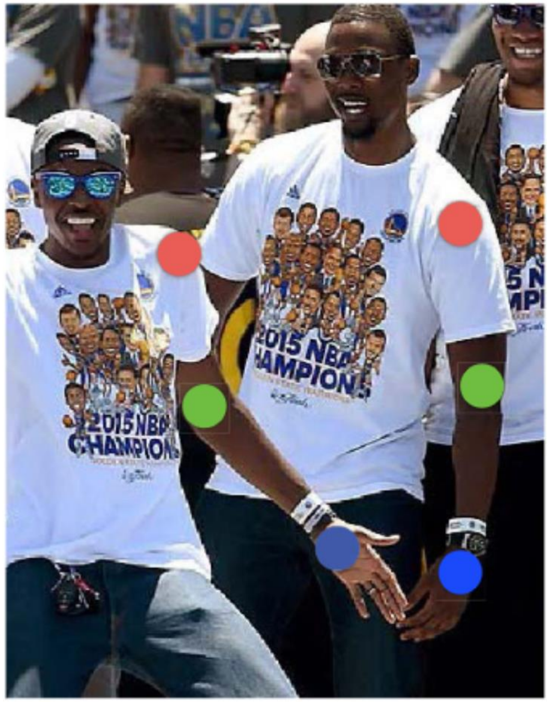
$$f_{\mathbf{C}}^{t_k} = \sum_{n=1}^{N} \sum_{X} \mathbf{W}(\mathbf{x}) \cdot ||\mathbf{C}_m^{t_k}(\mathbf{x}) - \widehat{\mathbf{C}}_m(\mathbf{x})||_2^2 \quad (23)$$

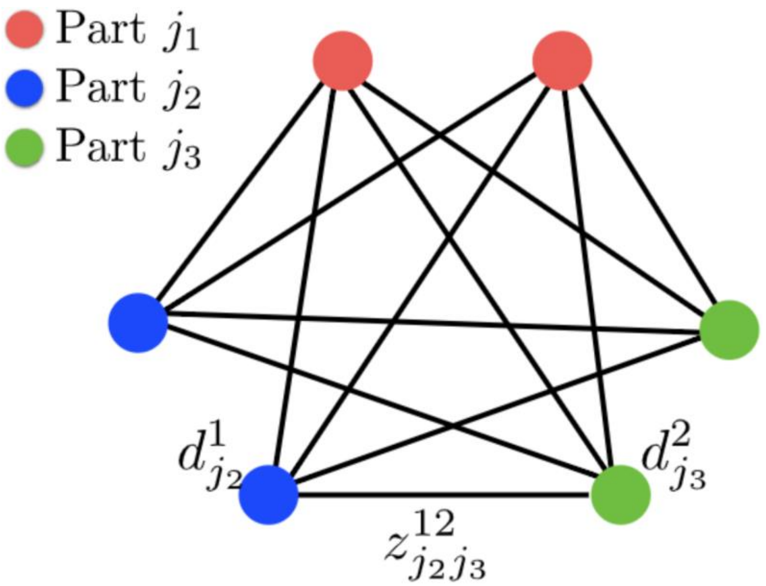
- Overall objective becomes: $$f = \sum_{t=1}^{T_P} f_{\mathbf{P}}^t + \sum_{t=T_P+1}^{T_P+T_C} f_{\mathbf{C}}^t \quad (24)$$

- But how to put it all together for multi-person 2D pose estimation?

# Multi-person 2D pose parsing via graphs

- Consider part detection as nodes (vertices)
- Possible connections along limbs as edges
- PAF score as weight for those edges

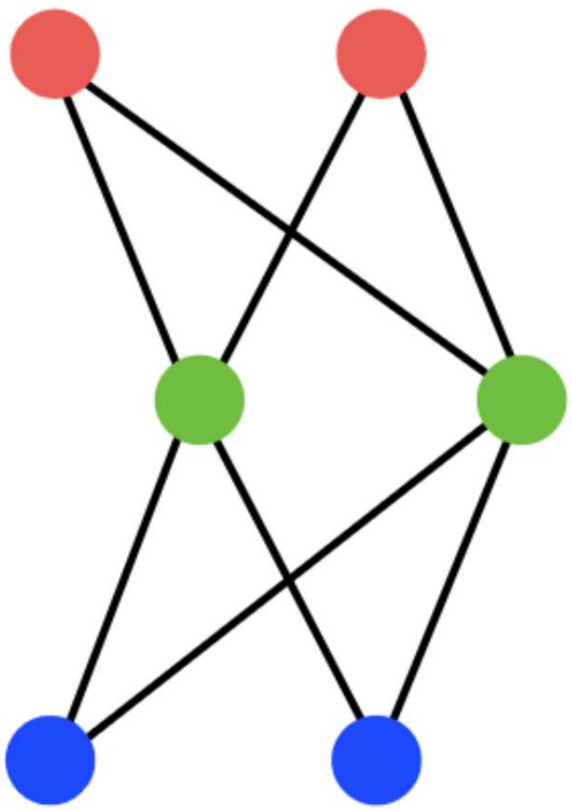




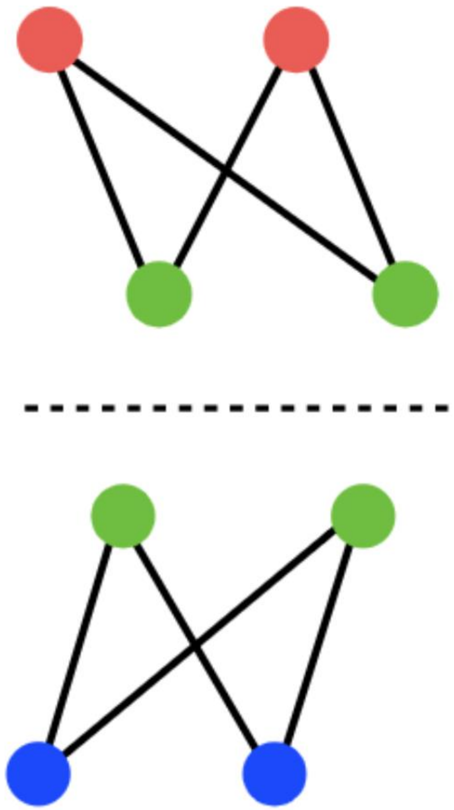

All edges (solved with Integer Linear Programming ILP)

# Multi-person 2D pose parsing via graphs

- K-dimensional matching problem
- Finding optimal parse is NP-Hard
- Relaxation 1: Minimal number of edges for spanning tree skeleton
- Relaxation 2: Decomposition into set of bipartite graphs



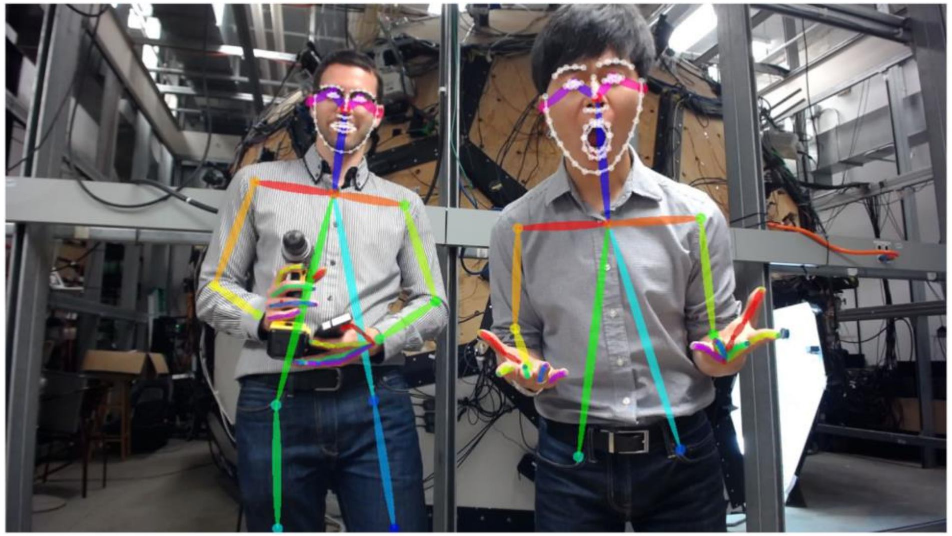
Minimal tree edges (approximated with ILP)



Proposed greedy parsing

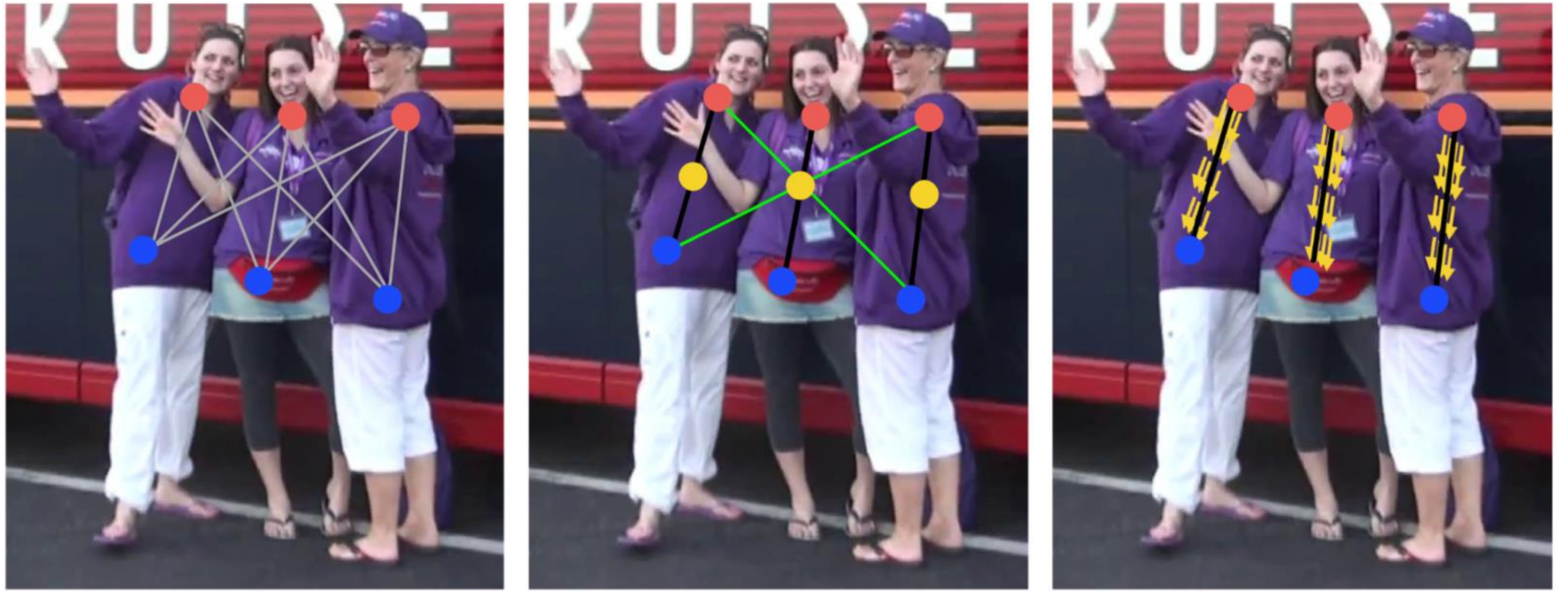
# Additional components

- Redundant PAF connections (e.g. ear-shoulder)
- PAF -> confidence map better than confidence map -> PAF
- Receptive field increase by replacing 1 7x7 conv with 3 3x3 convs
- Similar results between different skeleton structures (parsing)



# Summary

- Learn and predict confidence parts for detecting joints / parts
- Learn and predict PAFs for limb connections between parts
- Interpret as graph, solve via greedy parsing
- OpenPose library very frequently used, have fun experimenting!



Questions?

Suggestions?

Complaints?

Thank you for your attention

# References

References marked with * have not been read by the lecturer (A. Kriegler) but are instead included for the sake of providing a historically accurate representation of the scientific progress in the field of deep learning. The correctness of this representation is largely based on works by J. Schmidhuber [2, 66].


[1] C.M. Bishop, *Pattern Recognition and Machine Learning,* Singapore: Springer, 2006. (PDF)

[2] J. Schmidhuber, "Deep learning in neural networks: An overview", *Neural networks,* 2015, 61, pp.85-117, 2015. (PDF)

[3] I. Goodfellow, Y. Bengio and A. Courville, *Deep learning*, Cambridge: MIT press, 2016. (e-Book)

[4] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach,* Fourth edition, New Jersey: Pearson, 2021. (accompanying website)

[5] M. Bronstein, J. Bruna, T. Cohen and P. Velickovic, "Geometric Deep Learning", GDL course part of AMMI 2021, 2021. [Online]. Available at: https://geometricdeeplearning.com/lectures/ (Accessed: 22.10.2021).

[6] T. Gärtner, M. Thiessen and A. Sepliarskaia, "194.100 Theoretical Foundations and Research Topics in Machine Learning". [Online]. Available at: https://preview.tinyurl.com/yf73xbu7 (Accessed: 11.05.2021).

[7] M. Charikar and C. Ré, "CS229: Machine Learning". [Online]. Available at: http://cs229.stanford.edu (Accessed: 11.05.2021).

[8] L. FeiFei, K. Ranjay, X. Danfei, „CS231n: Convolutional Neural Networks for Visual Recognition". [Online]. Available at: http://cs231n.stanford.edu (Accessed: 11.05.2021).

[9] L. Fridman, „MIT Deep Learning and Artificial Intelligence Lectures". [Online]. Available at: https://deeplearning.mit.edu (Accessed: 11.05.2021).

[10] G. Sanderson, „Deep Learning Series". [Online]. Available at: https://preview.tinyurl.com/m92b8r9u (1st part) (Accessed: 11.05.2021).

[11] U. Von Luxburg, „Mathematics for Machine Learning". [Online]. Available at: https://preview.tinyurl.com/jk9p3m9x (Accessed: 11.05.2021).

[12] L. Von Rueden, S. Mayer, R. Sifa, C. Bauckhage and J. Garcke, „Combining Machine Learning and Simulation to a Hybrid Modelling Approach: Current and Future Directions", *International Symposium on Intelligent Data Analysis (IDA)*, 2020, pp.548-560, 2020. (PDF)

[13] H. Dawar, *Stochastic Gradient Descent.* [Online]. Available at: https://preview.tinyurl.com/erz8k26h (Accessed: 11.05.2021).

[14] J. Duchi, „Introduction to Convex Optimization for Machine Learning", *Practical Machine Learning, Fall 2009*, UC Berkeley, 2009. (PDF)

[15] A. Ivakhnenko and V.G. Lapa, *Cybernetic predicting devices,* New York: CCM Information Corp., 1965. (accompanying website)

[16] J. Schmidhuber, "Learning Complex, Extended Sequences using the Principle of History Compression", *Neural Computation*, 1992, 4(2), pp.234-242, 1992. (PDF)

# References


[17] S. Hochreiter, *Untersuchungen zu Dynamischen Neuronalen Netzen*. München: Technische Universität München (master's thesis, german), 1991. (PDF)

[18] A. Krizhevsky, I. Sutskever and G.E. Hinton, "Imagenet Classification with Deep Convolutional Neural Networks", *Advances in Neural Information Processing Systems (NIPS)*, 2015, 25, pp.1097-1105, 2015. (PDF)

[19] W. Fedus, B. Zoph and N. Shazeer, "Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity". [Online]. Available at: https://arxiv.org/abs/2101.03961 (Accessed: 11.05.2021).

[20] Original source unknown. Taken from: L. Langer, „Algorithm Journey from PC to IoT – A Signal Processing Pipeline with TensorFlow 2.X on a Smartwatch". [Online]. Available at: https://preview.tinyurl.com/5xfkchpb (Accessed: 11.05.2021).

*[21] K. Fukushima and S. Miyake, "Neocognitron: A Self-Organizing Neural Network Model for a Mechanism of Visual Pattern Recognition", *Biological Cybernetics,* 1980, 36, pp.193-202, 1980. (PDF)

[22] S. Saha, "A Comprehensive Guide to Convolutional Neural Networks – the ELI5 way". [Online]. Available at: https://preview.tinyurl.com/6d8njrv6 (Accessed: 11.05.2021).

[23] M.D. Zeiler and R. Fergus, „Visualizing and Understanding Convolutional Networks", *European Conference on Computer Vision*, 2014, pp.818-833, 2014. (arXiv preprint)

[24] Z. Cao, G. Hidalgo, T. Simon, S.E. Wei and Y. Sheikh, „OpenPose: Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields", *Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 43(1), pp. 172-186, 2021. (arXiv)

[25] Z. Ma et al., „BAGUALU: Targeting Brain Scale Pretrained Models with over 37 Million Cores", *SIGPLAN Symposium on Principles and Practice of Parallel Programming*, 2022, pp. 192-204, 2022. (PDF)