Visual AI

CPSC 533R

Lecture 4. Advanced architectures and sparse 2D keypoints

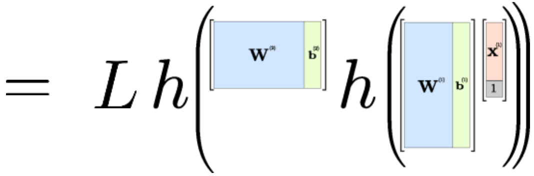
Helge Rhodin

Pick a paper till tonight

	A	B	C	D	E	F	G	H
1	Please insert your name in the subsequent cell and mark your 1st, 2nd and 3rd choices in the Choices column, as examplified below.				Name:	FirstName LastName		
2	There should be no read text left once you submit it to Canvas							
3	13 Presentation sessions							
4	Date*	Topic	Authors	Title & Link	Keywords	Choices		
5	24-Sep	Objective functions II	Bishop	Mixture Density Networks	probabilitic regression			
6			Barron et al	A General and Adaptive Robust Loss Function	optimizing the objective function			
7								
8	1-Oct	Self-supervision I	Vondrick et al.	Tracking Emerges by Colorizing Videos	self-supervision, color and motior			
9			Doersch et al.	Unsupervised visual representation learning by context prediction	self-supervision, classification, tri			
10								
11	8-Oct	Human Pose	Wandt et al	Repnet: Weakly supervised training of an adversarial reprojection network for	3D pose, projection, self-supervisi			
12			Cao et al	Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields	2D pose estimation, skeleton repr			
13								
14	15-Oct	GANs	Chan et al.	Everybody Dance Now	image translation			
15			Park et al.	Semantic Image Synthesis with Spatially-Adaptive Normalization	GAN, batch-norm, pixel-condition			
16			Lu et al.	Layered Neural Rendering for Retiming People in Video	neural rendering, depth estimatio			
17	22-Oct	Implicit functions	Sitzmann et al.	Implicit neural representations with periodic activation functions	implicit function, gradient, optimi			
18			Saito et al.	PIFu: Pixel-Aligned Implicit Function for High-Resolution Clothed Human Digiti:	implicit functions, 3D reconstructi			
19			Peng et al.	Convolutional Occupancy Networks	implicit functions, volumetric grid			
20	29-Oct	Animation and meshes	Xu et al.	RigNet: Neural Rigging for Articulated Characters	animation, rigging, mesh processi			
21			Gao et al.,	Automatic Unpaired Shape Deformation Transfer	autoencorder, meshes, style trans			
22			Hanoca et al	Point2Mesh: A Self-prior for Deformable Meshes	shape transfer, 3D texture, autoer			
23	5-Nov	Object Detection	Carion et al.	End-to-End Object Detection with Transformers	object detection, transformer arch			
24			Wu et al.	Unsupervised Learning of Probably Symmetric Deformable 3D Objects from Ima	audoencoder, depth, symmetry, s			
25								
26	10-Nov	Body models	Zuffi et al.	Three-D Safari: Learning to Estimate Zebra Pose, Shape, and Texture from Imag	body mode, animals, optimization			
27			Ma et al.	CAPE: Dressing SMPL	body model, mesh processing, aut			
28								
29	12-Nov	Objective functions II	Peebles et al.	The Hessian Penalty: A Weak Prior for Unsupervised Disentanglement	GAN, latent space disentanglemer	1		
30			Fort et al.	Deep Ensembles: A Loss Landscape Perspective	neural network theory, ensemple	2		
31								
32	17-Nov	Motion	Chu et al.	Learning Temporal Coherence via Self-Supervision for GAN-based Video Gener	GAN, temporal, self-supervision			
33			Luo et al.	Consistent Video Depth Estimation	autoencoder, 3D geometry, tempo			
34			Holden et al.	Phase-Functioned Neural Networks for Character Control	character animation, temporal mo			
35	19-Nov	Self-supervision II	Bielski et al.	Emergence of Object Segmentation in Perturbed Generative Models	GAN, self-supervision, segmentat	3		
36			Crawford et al.	Spatially invariant unsupervised object detection with convolutional neural ne	probabilistic modeling, self-super			
37			Lorenz et al.	Unsupervised Part-Based Disentangling of Object Shape and Appearance	self-supervision, autoencoder, po			
38	24-Nov	Rendering	Liu et al.	Soft Rasterizer: A Differentiable Renderer for Image-based 3D Reasoning	differentiable rendering, occlusio			
39			Schwartz et al.	The Eyes Have It: An Integrated Eye and Face Model for Photorealistic Facial An	autoencoder, differentiable rende			
40								
41	26-Nov	View synthesis	Mildenhall et al.	NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis	novel view synthesis, ray-tracing,			
42			Hinton et al.	Transforming Auto-encoders	novel view synthesis, stereo, stru			
43	Alternatives (instead of one of the above)							
44			Nguyen-Phuoc at al.	HoloGAN: Unsupervised learning of 3D representations from natural images	self-supervision, view synthesis, 3			
45			Moon et al.	I2L-MeshNet: Image-to-Lixel Prediction Network for Accurate 3D Human Pose a	3D human pose and shape estimat			
46								
47								

Recap: Automatic differentiation and backpropagation

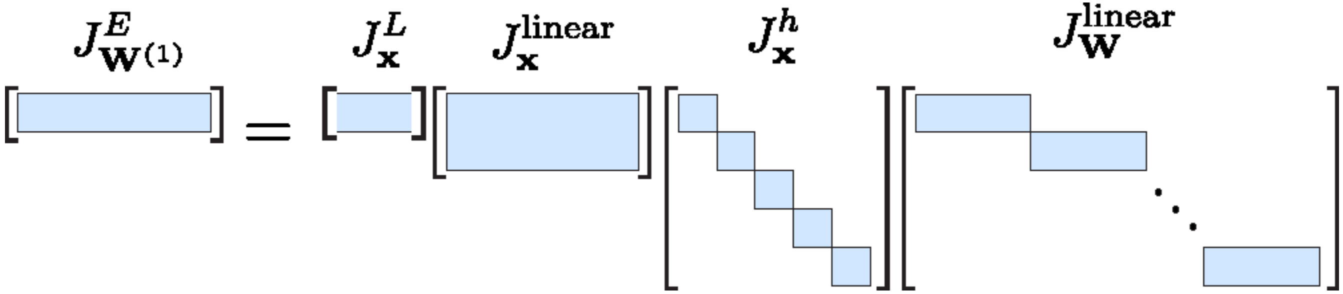
Forward pass $L(h(\text{linear}(h(\text{linear}(x, W^{(1)})), W^{(2)})))$



$$\mathbf{J}_x^f = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

Jocobian matrix

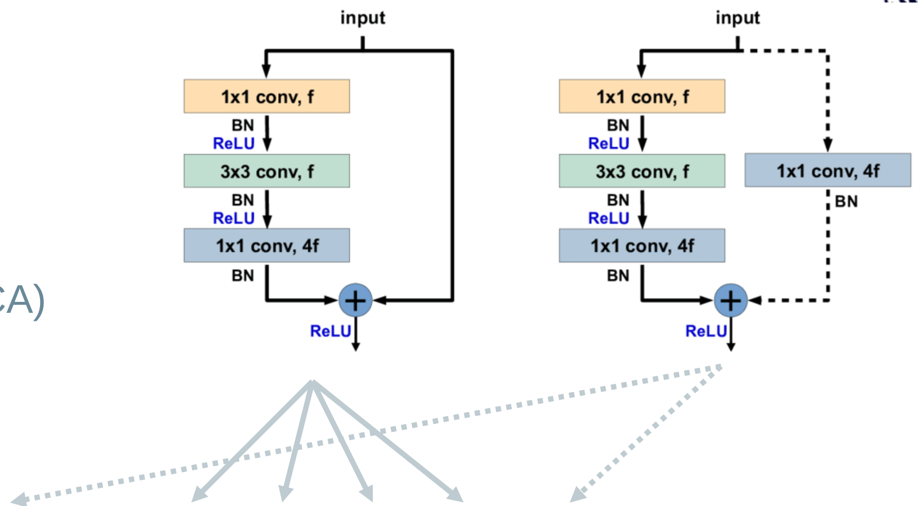
Backwards pass to $W^{(1)}$



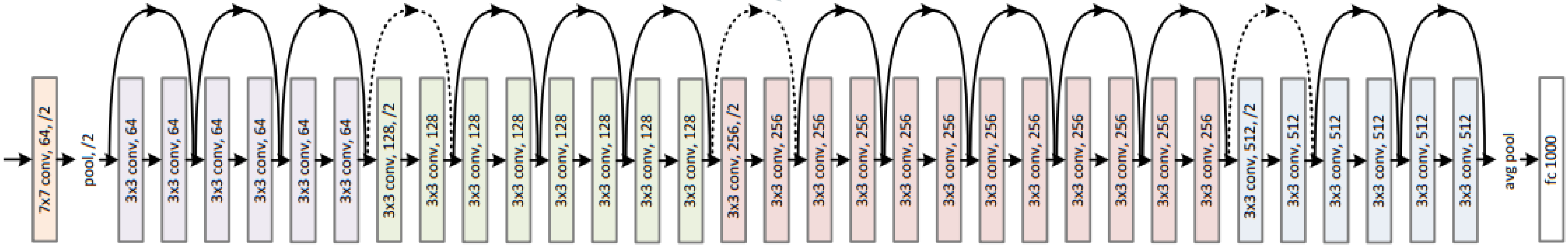
ResNet details

What if number of channels changes?

- apply a linear transformation to match the size
  - a projection on a linear subspace,
    related to principal component analysis (PCA)



ResNet 32 (32 layers)



Batch normalization

[Batch Normalization: Accelerating Deep Network Training ~~by Reducing Internal Covariate Shift~~]

- Normalize after each linear + activation function
  - normalize across minibatch, to have μ=0 and σ=1

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}$$

  - Strict normalization reduces performance, hence, add a learnable offset and scale

$$y_i \leftarrow \gamma \widehat{x}_i + \beta$$

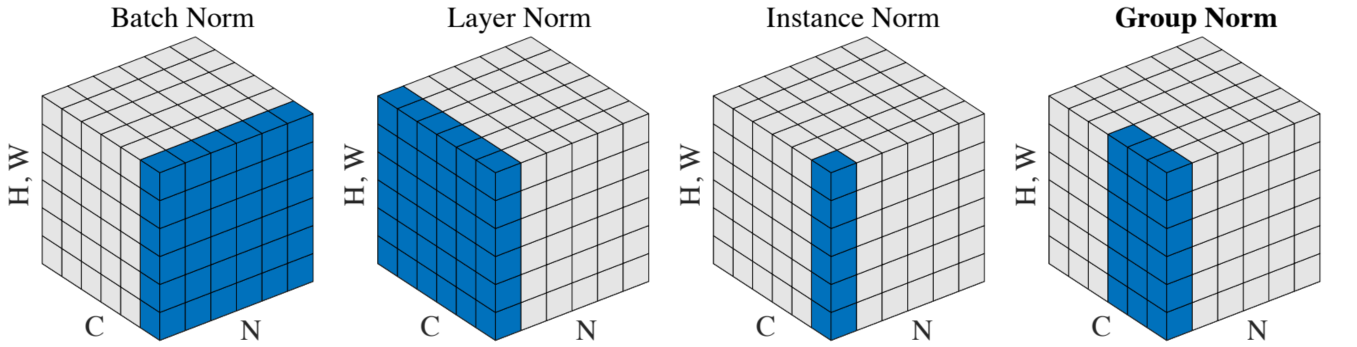
  - What if we only have a single image at inference time?
    - Re-apply mean and variance recorded during training (using exponential moving average)

Batch normalization effect and variants

What is the benefit of first normalizing and then 'denormalizing'?

- noise from other images regularizes
- it separates learning of the variance (scale) and bias (offset) from the values itself
- Empirical: training deeper networks, with sigmoid activation, higher learning rate, and faster convergence

Variants normalize over different slices of the feature tensor:



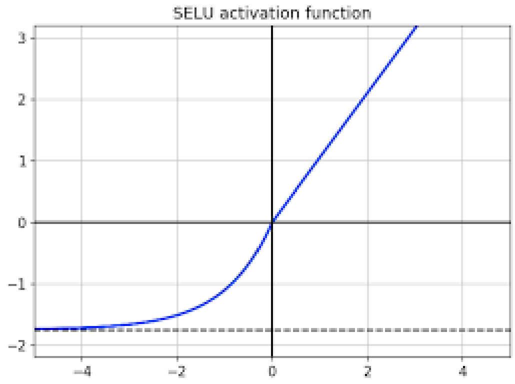
[Wu and He. Group Normalization]

Self-normalizing neural networks

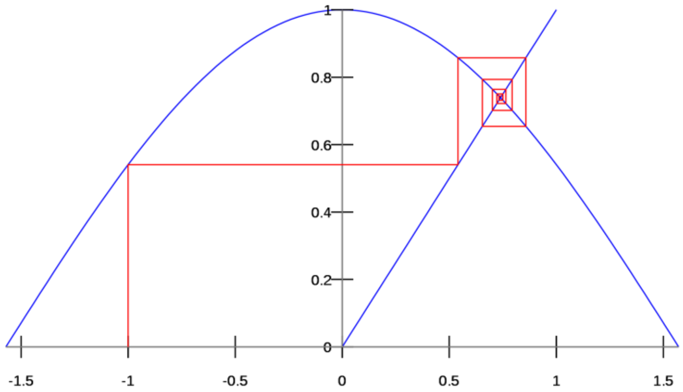
Self-normalizing Neural Networks

[Klambauer et al.]

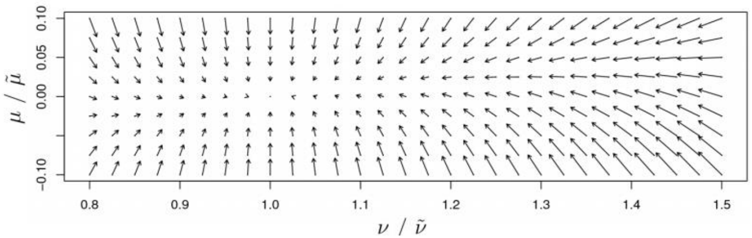
- fixed point enforced by choice of activation function (SELUs)
- stable and attracting fixed point for the function g that maps mean and variance from one layer to the next
- Possibility to train deep fully connected NNs



$$\mathrm{selu}(x) = \lambda \begin{cases} x & \text{if } x > 0 \\ \alpha e^x - \alpha & \text{if } x \leqslant 0 \end{cases}$$



Fixed point iterations for cos(x)



Mapping of the function g towards mu=0 and v=1

Regularization

Dropout

- randomly zero out activations
- re-weight the non-zero ones to maintain the distribution of the unmodified activations
  - induced noise reduces overfitting

Weight decay

$$\tilde{\mathbf{w}} = (1-\tau)\mathbf{w} \text{ with } \tau \text{ small})$$

Prior on neural network weights

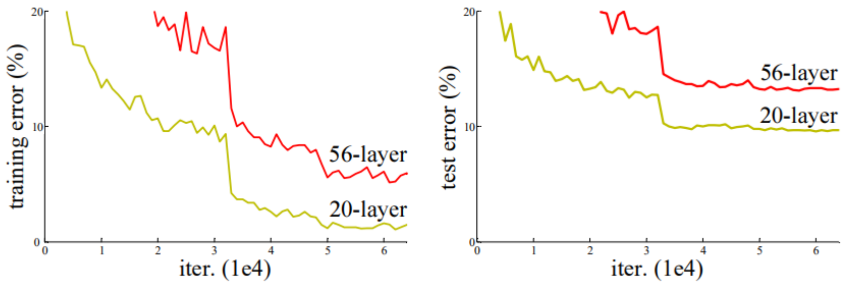
$$\widetilde{E}(\mathbf{w}) = E(\mathbf{w}) + \frac{\lambda}{2}\mathbf{w}^2$$

Weight decay and square prior are equivalent under certain conditions (vanilla SGD without momentum)

Check out AdamW in pytorch

Residual networks and skip connections

- Deep networks are hard to train



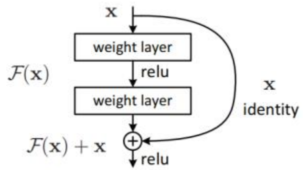


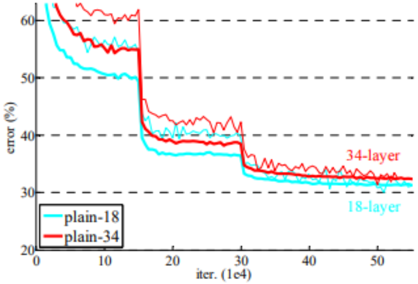
Figure 2. Residual learning: a building block.

- Residual blocks with shortcut/skip connections

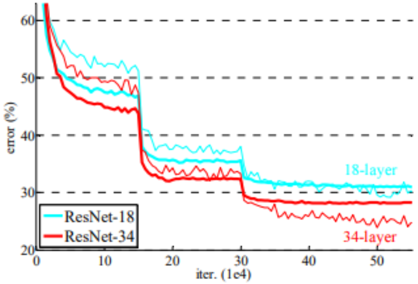
$$\mathbf{y} = F(\mathbf{x}) + \mathbf{x}$$

  - no extra parameters
  - enables training of deep neural networks

Image net training

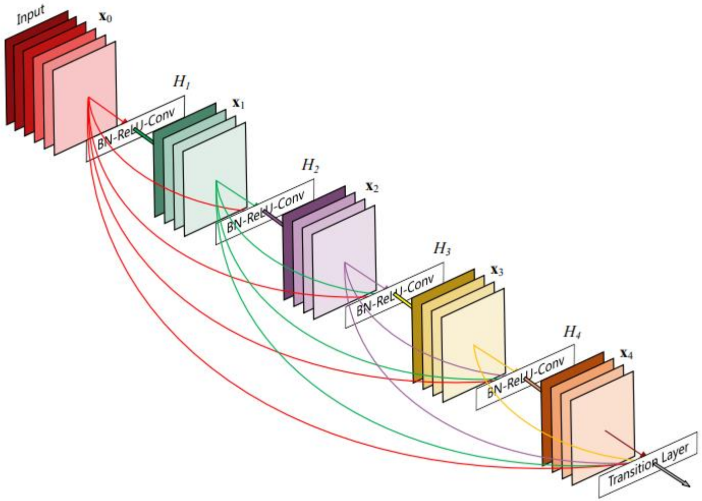


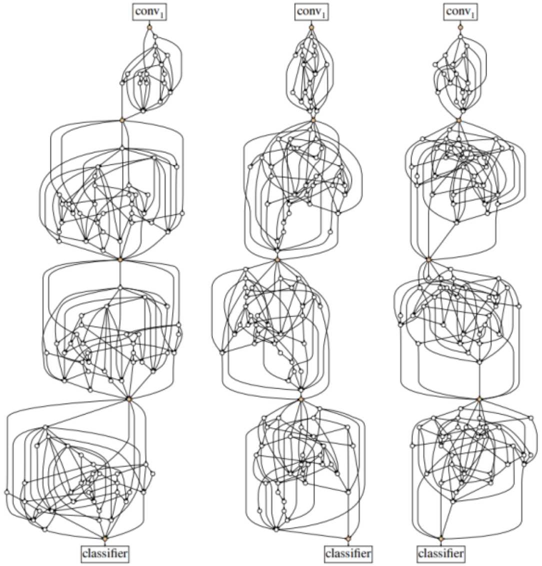
Baseline network



ResNet version

Other network architectures



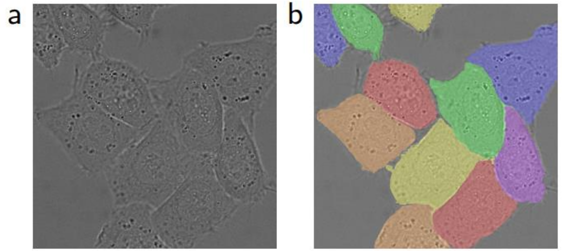


DenseNet
(skip connection to all future layers)

Randomly wired networks
(search for best wiring among candidates)

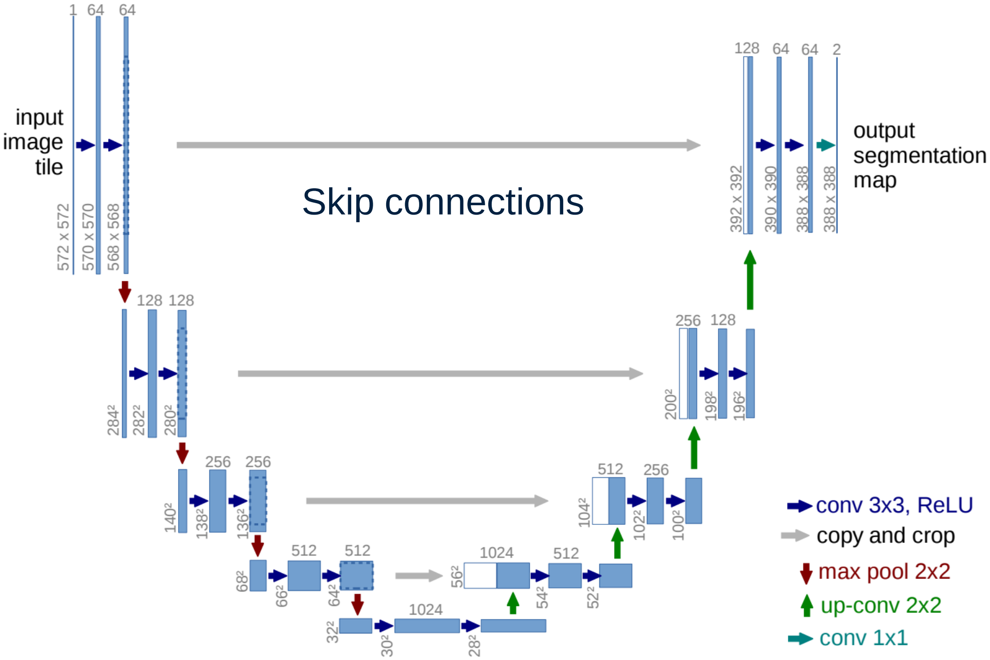
U-Net architecture

- Similar input and output resolution
- A global encoding is learned by down sampling (to 32 x 32 px)
- Progressive increase of channels maintains throughput / capacity
- Skip connections preserve details



Encoder
(spatial down sampling)

Decoder
(spatial up sampling)



[U-Net: Convolutional Networks for Biomedical Image Segmentation]

GoogleLeNet (Inception Net V1)

[Szegedy et al., 2014]

even deeper network with **computational efficiency**

— 22 layers

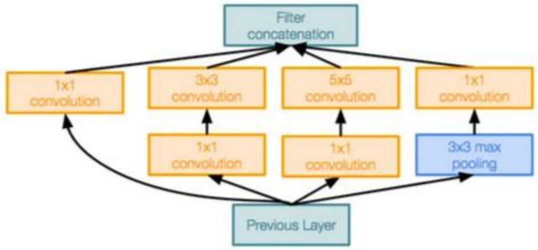
— Efficient “Inception” module

— No FC layers

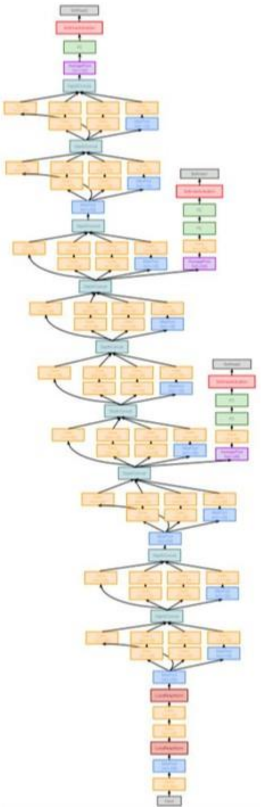
—Only 5 million parameters!

(12x less than AlexNet!)

— Better performance (@6.7 top 5 error)

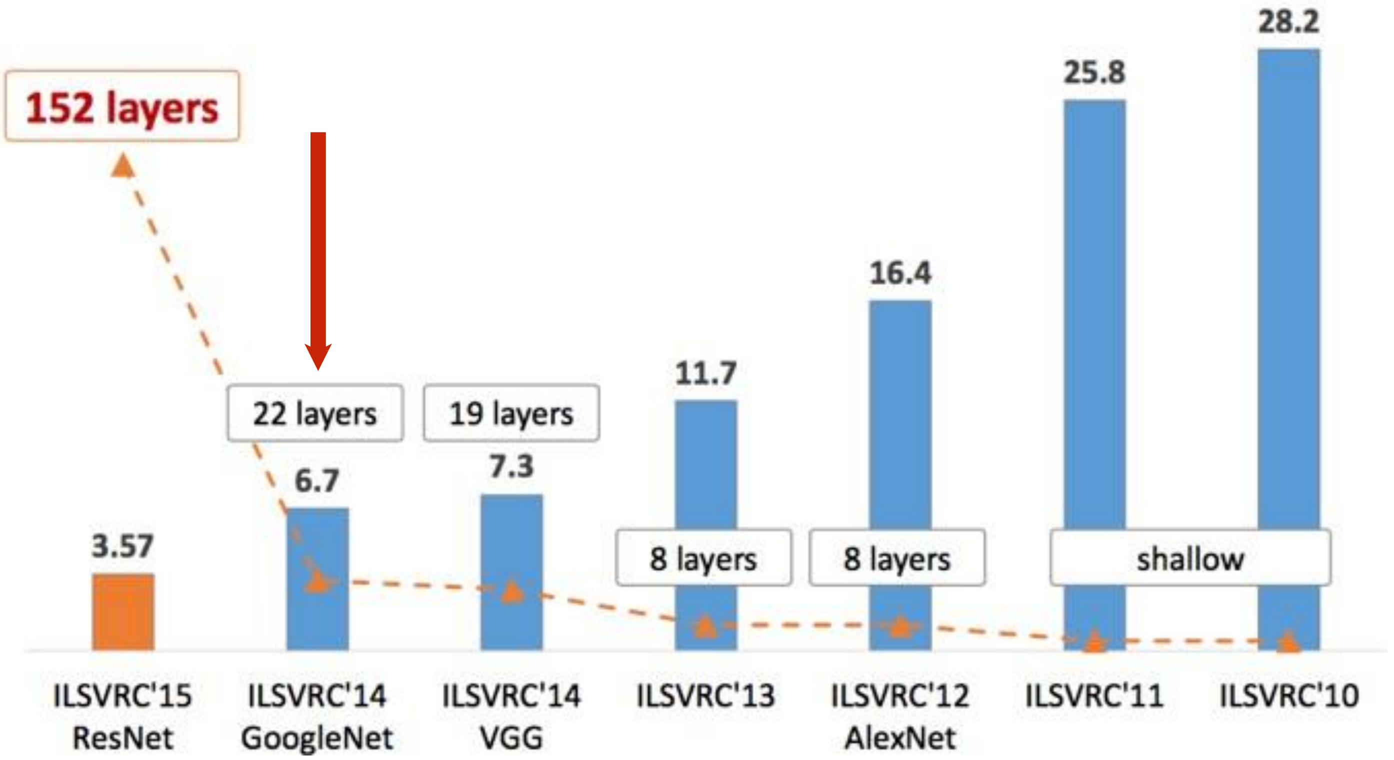


Inception module



* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

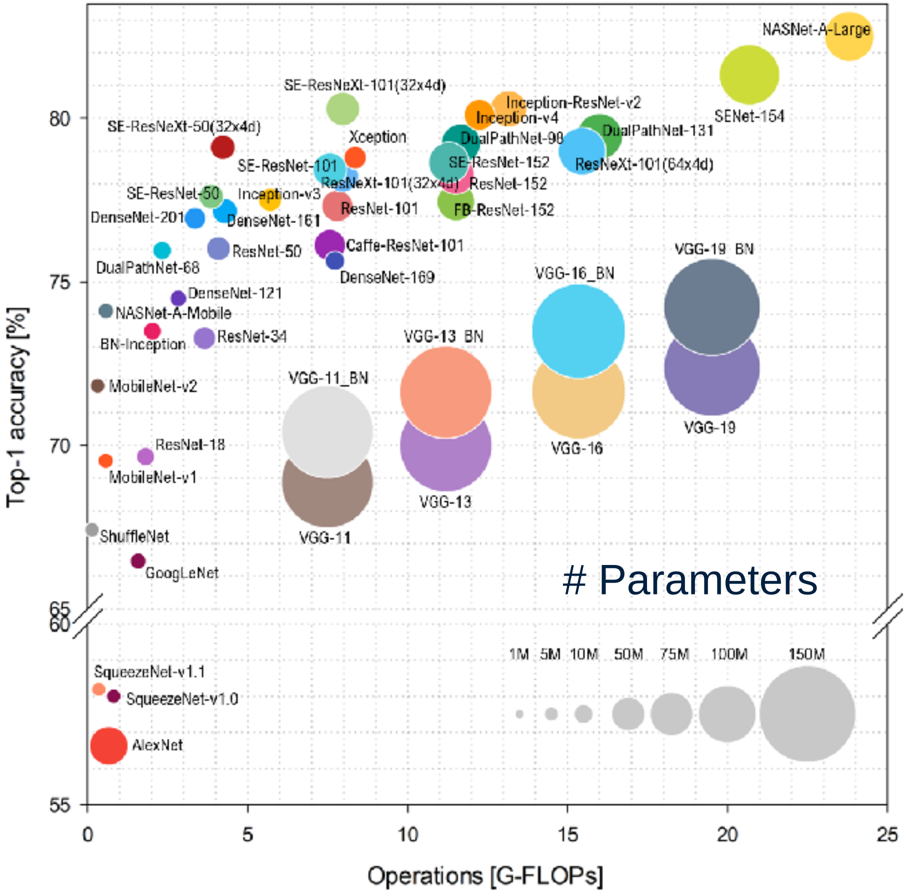
ILSVRC winner 2012



* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Network comparison

The goal is to balance
- accuracy (maximize)
- number of parameters (minimize)
- number of operations (minimize)
- efficiency of operations (maximize)
    - cache efficiency
    - parallel execution
    - precision, e.g. float, float16, binary,…



History of Inception Networks

Inception V1 GoogleNet

- Network in network approach

Inception V2

- Use of batch normalization

  [Batch normalization: Accelerating deep network training by …]

Inception V3

- Factorizations

  [Rethinking the inception architecture for computer vision]

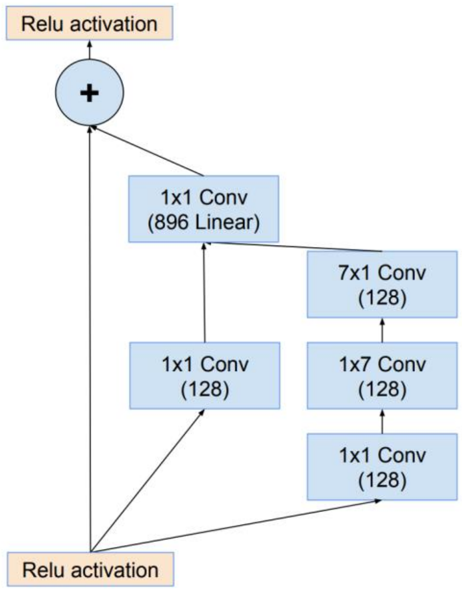
Inception V4

- Tuning of filters

  [Inception-v4, Inception-ResNet and the Impact of …]

Inception-ResNet

- Skip connections instead of concatenations



Inception ResNet block example

Separable Convolutions

Idea:

Separate a single convolution operation into a sequence of simpler operations

- e.g., 7x7 convolution into
  - 1x7 and 7x1
- reduction of parameters
  - e.g., 14 vs. 49 for 7x7 conv.

$$\underbrace{\begin{bmatrix} 6 & 3 & 3 & 6 \\ 4 & 2 & 1 & 8 \\ 2 & 5 & 6 & 4 \\ 9 & 3 & 2 & 1 \end{bmatrix}}_{f} \otimes \underbrace{\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}}_{k} \Rightarrow \underbrace{\begin{bmatrix} 8 & -9 & 12 & -7 \\ 12 & -5 & 14 & -11 \\ 10 & -2 & 2 & -15 \\ 11 & -10 & -5 & -10 \end{bmatrix}}_{f \otimes k}$$

Drawback:

It models simpler functions

- no ‘diagonal’ entries possible
- successive layers can’t be run in parallel

$$\underbrace{\begin{bmatrix} 6 & 3 & 3 & 6 \\ 4 & 2 & 1 & 8 \\ 2 & 5 & 6 & 4 \\ 9 & 3 & 2 & 1 \end{bmatrix}}_{f} \otimes \underbrace{\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}}_{c} \otimes \underbrace{\begin{bmatrix} -1 & 0 & 1 \end{bmatrix}}_{r} \Rightarrow \underbrace{\begin{bmatrix} 8 & -9 & 12 & -7 \\ 12 & -5 & 14 & -11 \\ 10 & -2 & 2 & -15 \\ 11 & -10 & -5 & -10 \end{bmatrix}}_{f \otimes c \otimes r}$$

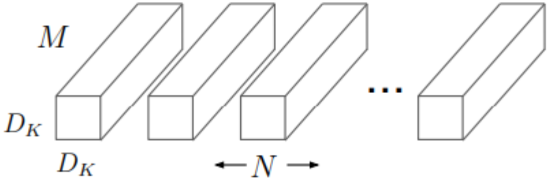
Mobile Net V1

Depthwise separable convolution

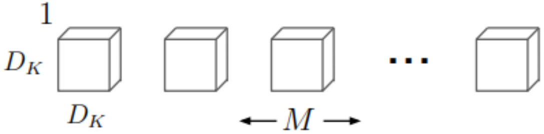
- separate a 3x3 convolution with M input and N output channels
  - M 3x3 convolutions, each applied on a single channel
  - N 1x1 convolutions, combining the M intermediate features
  - add ReLU and Batch Normalization after each layer

Advantages

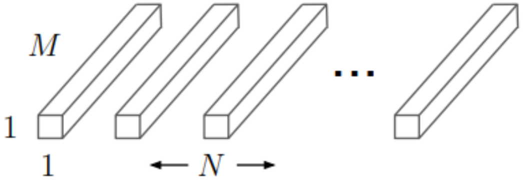
- fewer add-mul operations
  (8-9 times less than conventional convolution)
- highly efficient operations
  - 95% of time spend on 1x1 convolutions
  - 1x1 convolutions are highly optimized
    - an instance of general matrix multiplication (GMM)



(a) Standard Convolution Filters



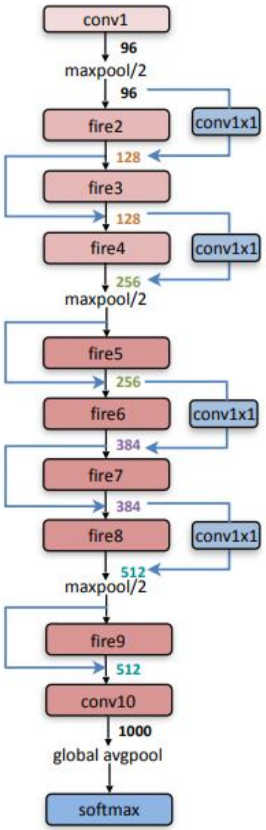
(b) Depthwise Convolutional Filters



SqueezeNet

Goal: Very small model size and efficient execution

- Strategy 1. Replace 3x3 filters with 1x1 filters
  - a 1x1 filter has 9X fewer parameters than a 3x3 filter
- Strategy 2. Decrease the number of input channels to 3x3 filters
  - (number of input channels) * (number of output channels) * (3*3)
- Strategy 3. Spatially downsample late
  - known to yield higher accuracy
- No fully connected layers
  - use global average pooling instead



SqueezeNet architecture

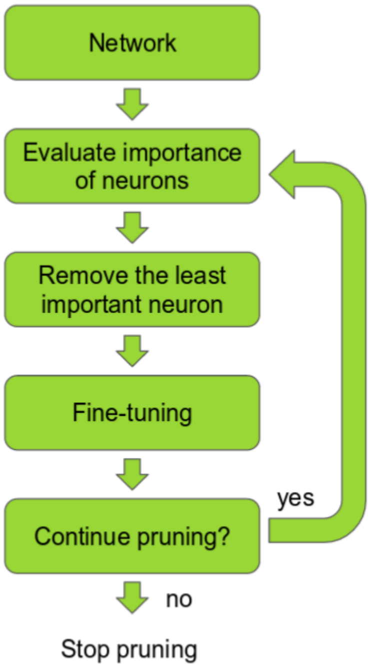
Reducing the model footprint

Pruning

- rank the neurons in the network according to how much they contribute, e.g.:
    - L1/L2 mean of neuron weights
    - their mean activations
    - the number of times a neuron wasn't zero on some validation set
- remove the low-ranking neurons

Deep Compression [Han et al., 2015]

- quantize CNN parameters (e.g. 8-bits of precision)
- uses a codebook



[Pruning Convolutional Neural Networks for Resource Efficient Inference]

ResNeXt: Aggregated Residual Transformations

Idea: “vertical residual blocks”

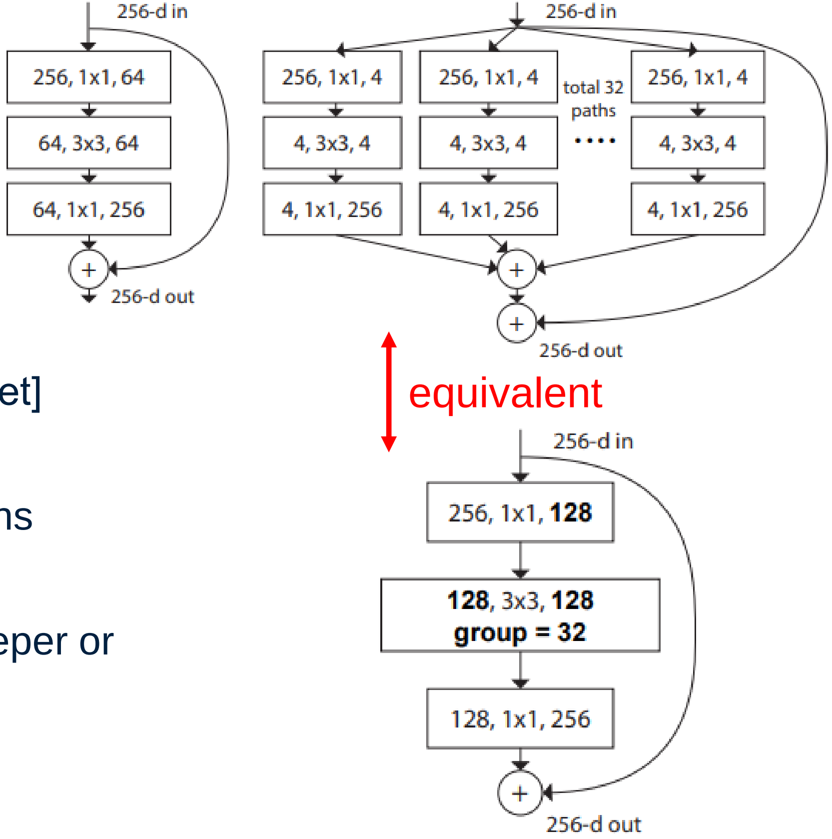
- create blocks with identical topology
    - and independent weights
    - replicate them c times (“cardinality”)
- add these blocks together
- add a skip connection as in ResNet

Related: Krizhevsky et al.’s grouped convolutions [AlexNet]

Advantage:

- larger number of channels, same number of operations
- improved performance
- increasing cardinality is more effective than going deeper or wider when we increase the capacity

A new (NeXt) dimension: depth, width, and **cardinality**



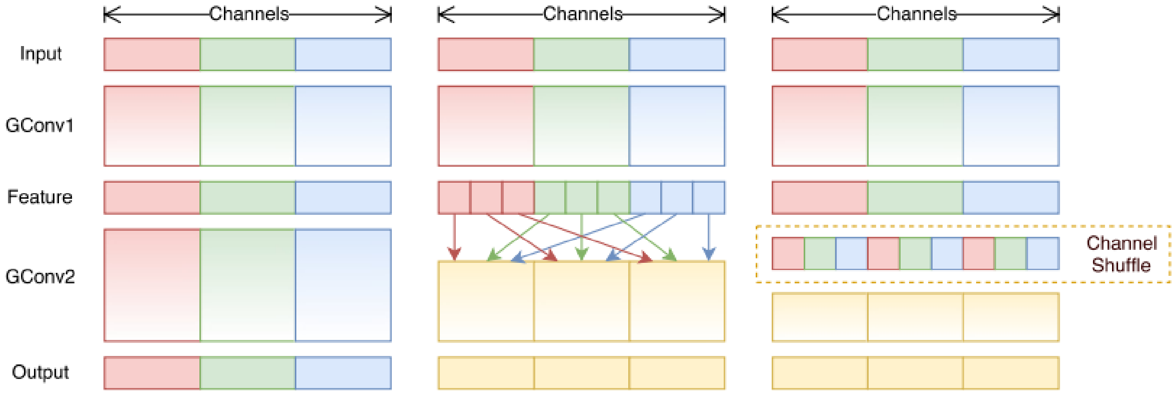
ShuffleNet

Idea:

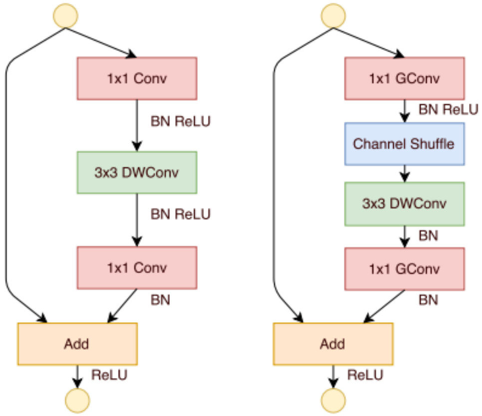
- Group-wise convolution
- shuffle features for cross-talk

Advantage:

- less parameters for 1x1 conv
  - 1/ #g parameters, where #g is the cardinality
  - recall that MobileNet spent 95% in 1x1 convolution



MobileNet block ShuffleNet block



Discussion

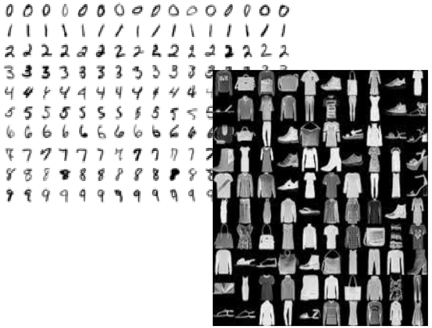
Assignment I & II

Assignment I

- is due ~~today~~ tomorrow!
- submit on Canvas

Assignment II

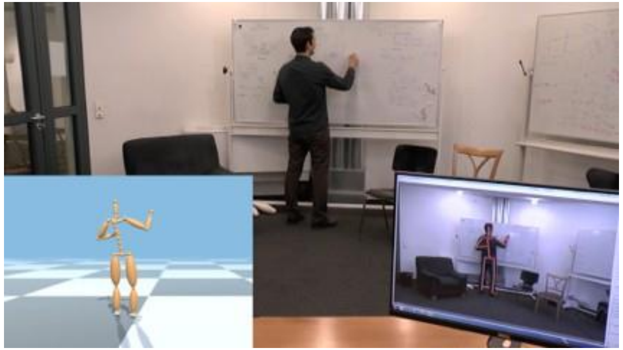
- released today
- later parts may require content from Thursday
    - start with preliminaries and Task I

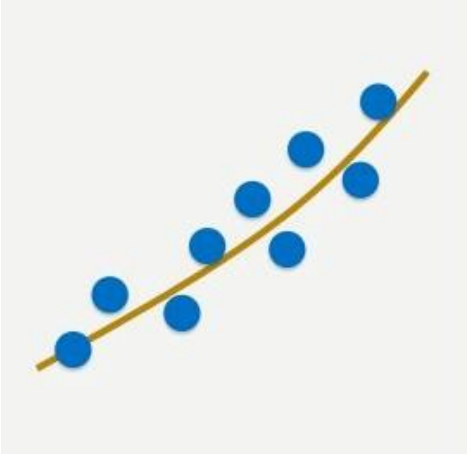


Classification vs. regression

Classification

Regression





Classification and regression

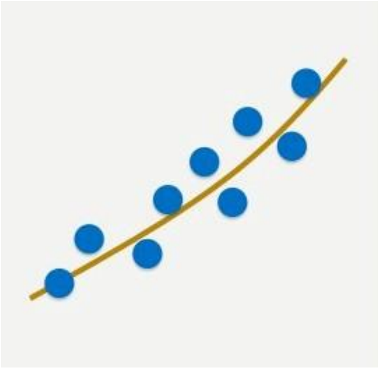
Regression

- for continuous values

$$\mathrm{nn}(\mathbf{x}) \rightarrow y \in \mathbb{R}$$

- squared loss is most common

$$l_2(y, l) = (y - l)^2$$



Classification

- discrete classes

$$\mathrm{nn}(\mathbf{x}) \rightarrow \mathbf{y} \in [0, 1]$$

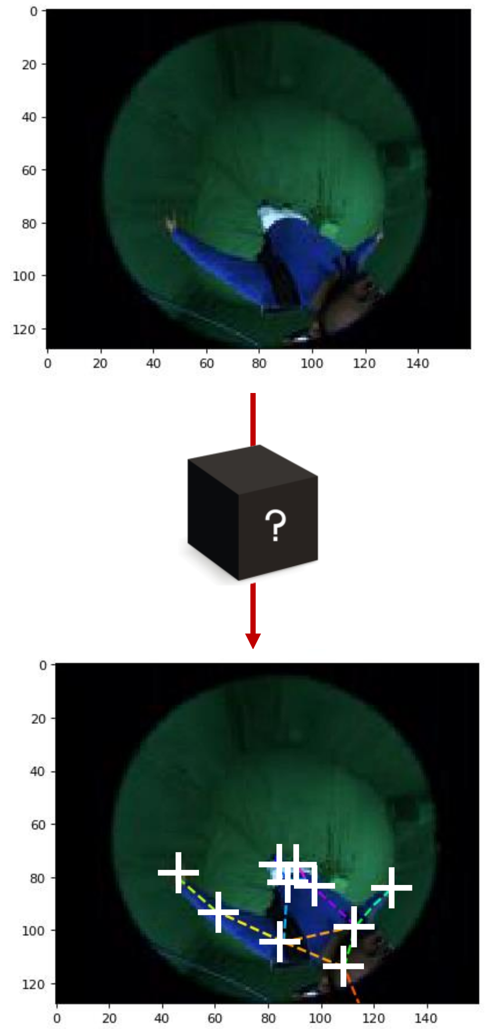
- naïve least-squares loss

$$l_2(\mathbf{y}, \mathbf{l}) = \|\mathbf{y} - \mathbf{l}\|^2$$

Regression-based 2D pose estimation

A classical regression task

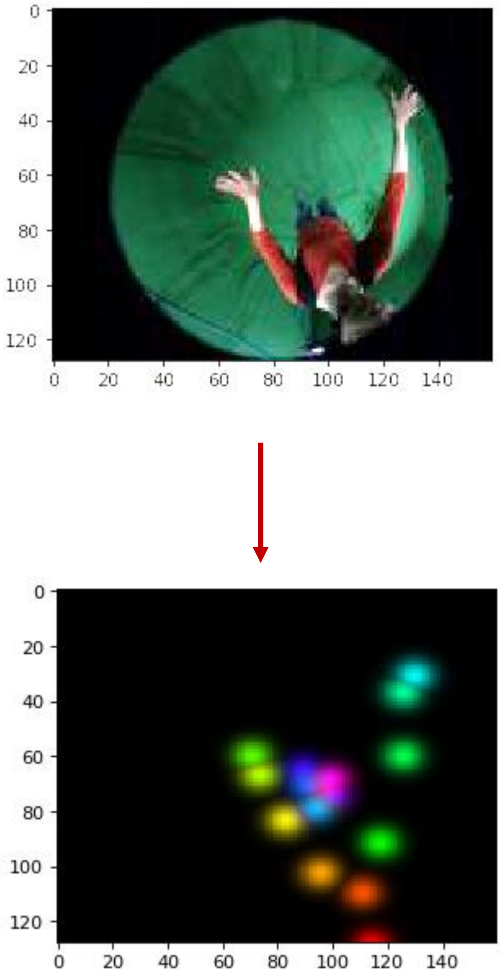
- Input:
  - grid of color values, an image (3 x W x H)
- Output:
  - pairs of continuous values, the position in the image
  - one pair for each of the K keypoints (2 x K)
- Neural network architecture:
  - Some convolutional layers to infer an internal representation of the human pose (C x W' x H')
  - One or more fully-connected layers to aggregate spatial information into the output values (C * W' * H') → (2 x K)



Heatmap-based 2D pose estimation

Phrase the regression task as classification

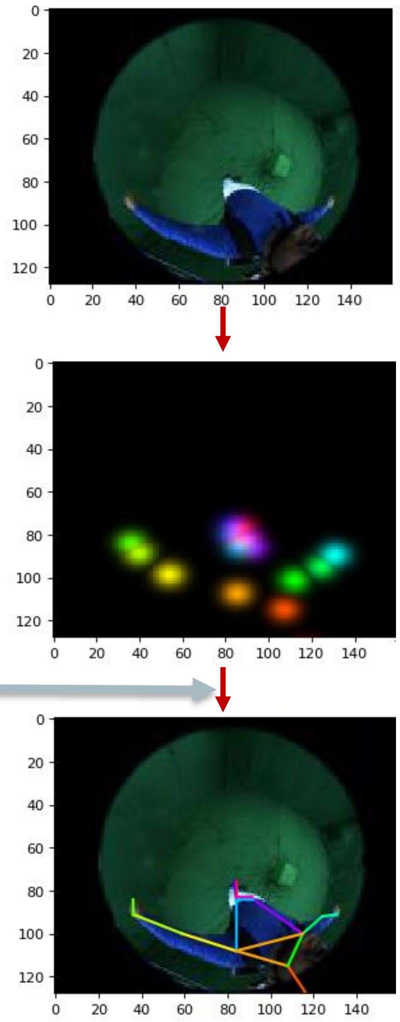
- separate heatmap H_j for each joint j
- Each pixel of H_j encodes the '*probability*' of containing joint j
  - not a true probability as pixels don't sum to one
- Advantages:
  - Inferred with fully convolutional networks
    - less parameters than fully connected ones (MLPs)
    - applies to arbitrary image resolution and aspect ratio (can be different from training)
    - translation invariance
    - locality
  - Generalizes to multiple and arbitrary number of persons



[Tompson et al., Efficient object localization using convolutional networks.]

Disadvantages of heatmaps

- Disadvantage:
  - Large image scale variations
    - Two-stage pipelines are alleviating this
      1. Detect person bounding box at coarse resolution
      2. Infer skeleton pose within box at high resolution
  - Not end-to-end differentiable (pose extraction requires arg-max function)
  - No sub-pixel accuracy
    - multi-scale approaches can overcome this at the cost of execution time (average over runs on re-scaled input)



Integral Regression-based 2D pose estimation I

A combination of classification and regression

1. Detection network to produce heatmaps
   - same CNN as for heatmap prediction
2. Soft-max layer to turn heatmap H into probability map P
   - normalizing all pixels in each heatmap H

$$P[u,v] = \text{soft-max}(H,(u,v)) = \frac{e^{H[u,v]}}{\sum_{x=1}^{\text{width}} \sum_{y=1}^{\text{height}} e^{H[x,y]}}$$

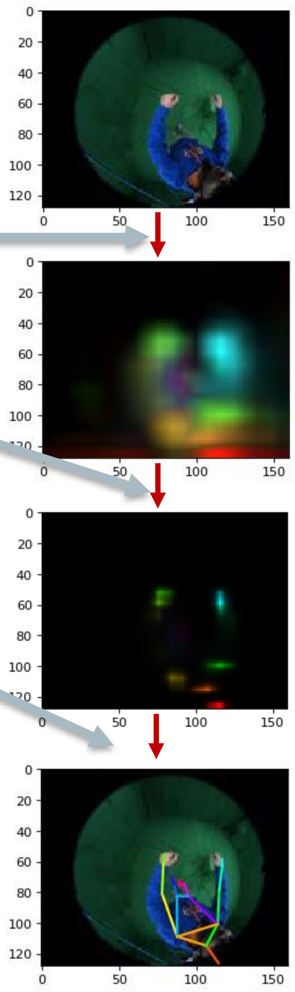
3. Integration layer to regress joint position (expected position)
   - can be interpreted as voting/weighted average

     each pixel votes for its own position, weighted by its probability

$$\text{pose}_x = \sum_{x=1}^{\text{width}} \sum_{y=1}^{\text{height}} xP[x,y]$$

$$\text{pose}_y = \sum_{x=1}^{\text{width}} \sum_{y=1}^{\text{height}} yP[x,y]$$

[Sun et al., Integral Human Pose Regression.]



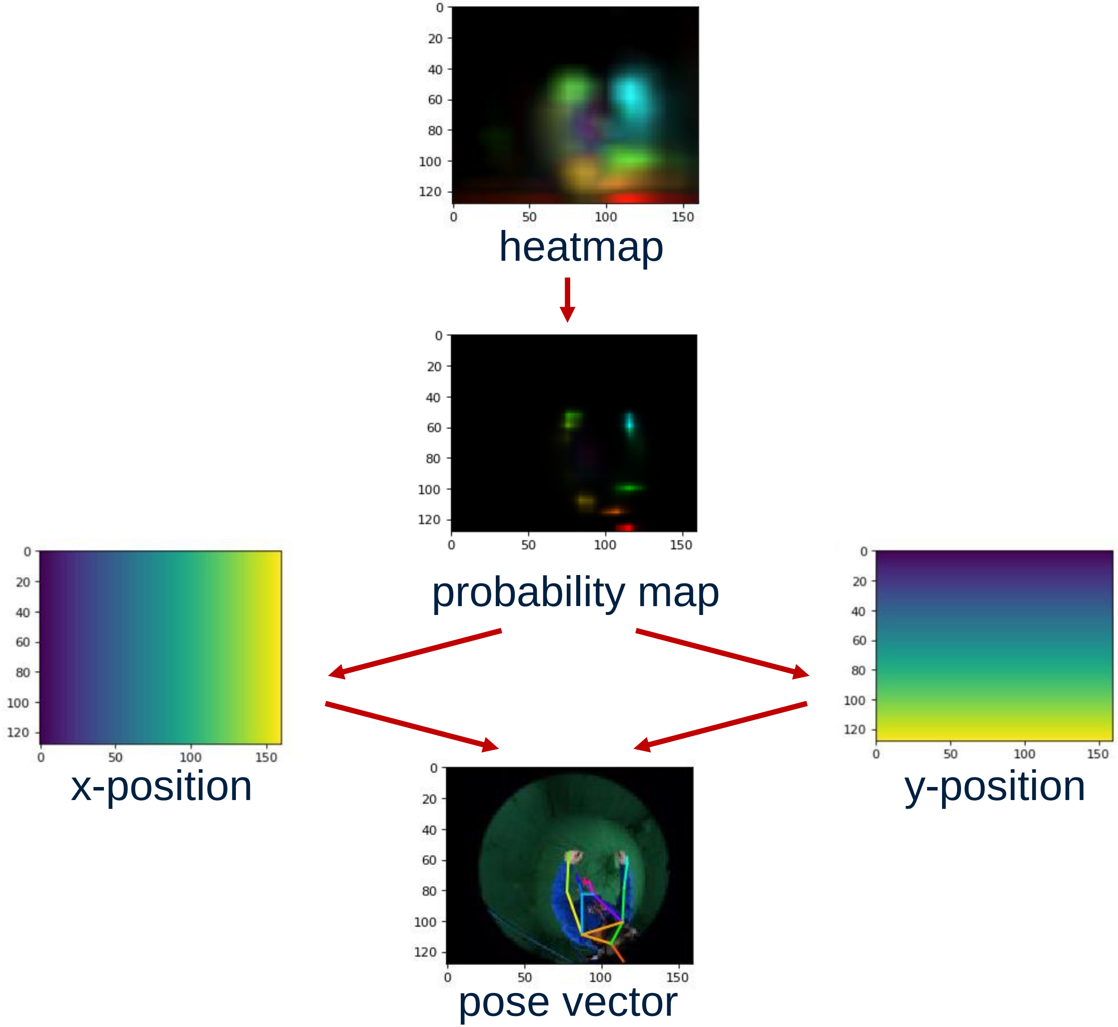
input

heatmap

prob. map

pose vector

Details



Integral Regression-based 2D pose estimation II

Advantages

1. Fully-convolutional CNN (as for heatmap classification)
2. Differentiable 2D pose regression
   - soft-max is differentiable, stable, and efficient to compute

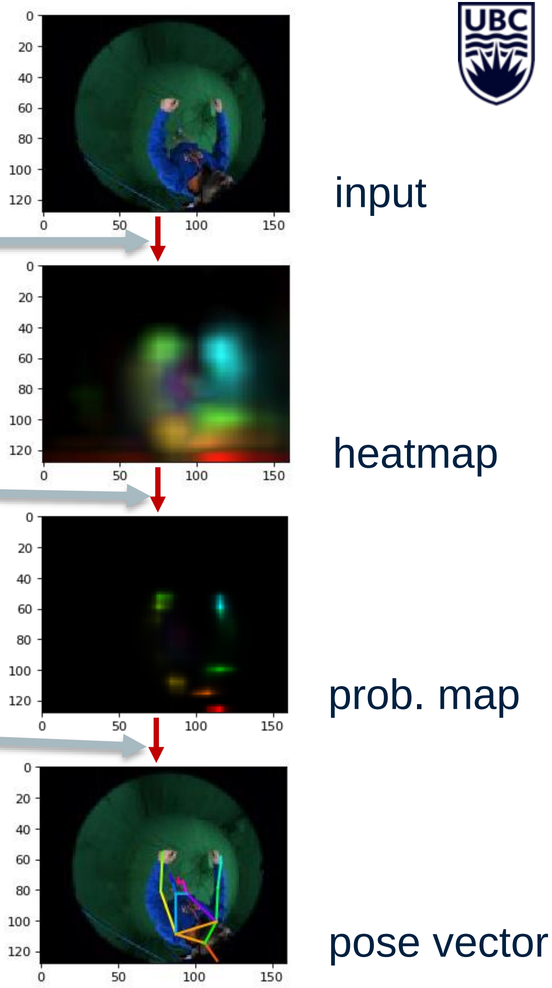
$$P[u,v] = \text{soft-max}(H,(u,v)) = \frac{e^{H[u,v]}}{\sum_{x=1}^{\text{width}} \sum_{y=1}^{\text{height}} e^{H[x,y]}}$$

   - sum over probability map is differentiable

$$\text{pose}_x = \sum_{x=1}^{\text{width}} \sum_{y=1}^{\text{height}} x P[x,y]$$

$$\text{pose}_y = \sum_{x=1}^{\text{width}} \sum_{y=1}^{\text{height}} y P[x,y]$$

3. End-to-end training
   - no difference between training and inference
   - sub-pixel accuracy possible through joint influence of pixels
     - low-resolution heatmaps possible



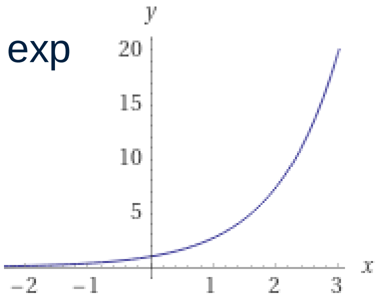
Attention: numerical stability

Exp normalize trick within cross-entropy

$$\text{soft-max}(z,i) = \frac{e^{z[i]\boxed{-\bar{z}}}\; e^{\bar{z}}}{\sum_{j=1}^{K} e^{z[j]\boxed{-\bar{z}}}\; e^{\bar{z}}}$$

$$= \frac{e^{z[i]-\bar{z}}}{\sum_{j=1}^{K} e^{z[j]-\bar{z}}}$$

shift invariance is used to increase numerical stability!



The PyTorch implementation includes this step

Integral Regression-based 2D pose estimation III

Disadvantages / open questions = possible course projects!

1. Sensitive to outliers
   - if there are two maxima in the heatmap, the predicted position will be in the middle of the two
2. How to support multiple people, at different scales?
   - Some form of hierarchical model?
3. Part affinity fields have been successful, can we develop a differentiable model?
   - An elongated ellipse that has position and orientation?
4. What about occluded joints?
5. What about temporal information?
6. Is it possible to infer neck-centered human pose (not knowing the absolute position, only relative distance of keypoints to the neck)?

Issues?

Your laptop / desktop

- No GPU? -> google colab or university (see lecture 2)
- Note, parallel dataloaders might not work well on Windows:
  Error: “Can't pickle <function <lambda> …”
  - fix: disable threading by setting num_workers=0
- Other issues encountered?