Bringing Structure to Visual Deep Learning

Cees Snoek, UvA
Arnold W.M. Smeulders, UvA
Efstratios Gavves, UvA
Laurens van de Maaten, Facebook

Recap from Day 2

Convolutional Networks are optimal for images

- ❑ Parameter sharing
- ❑ Much cheaper
- ❑ Much faster
- ❑ Better local invariances

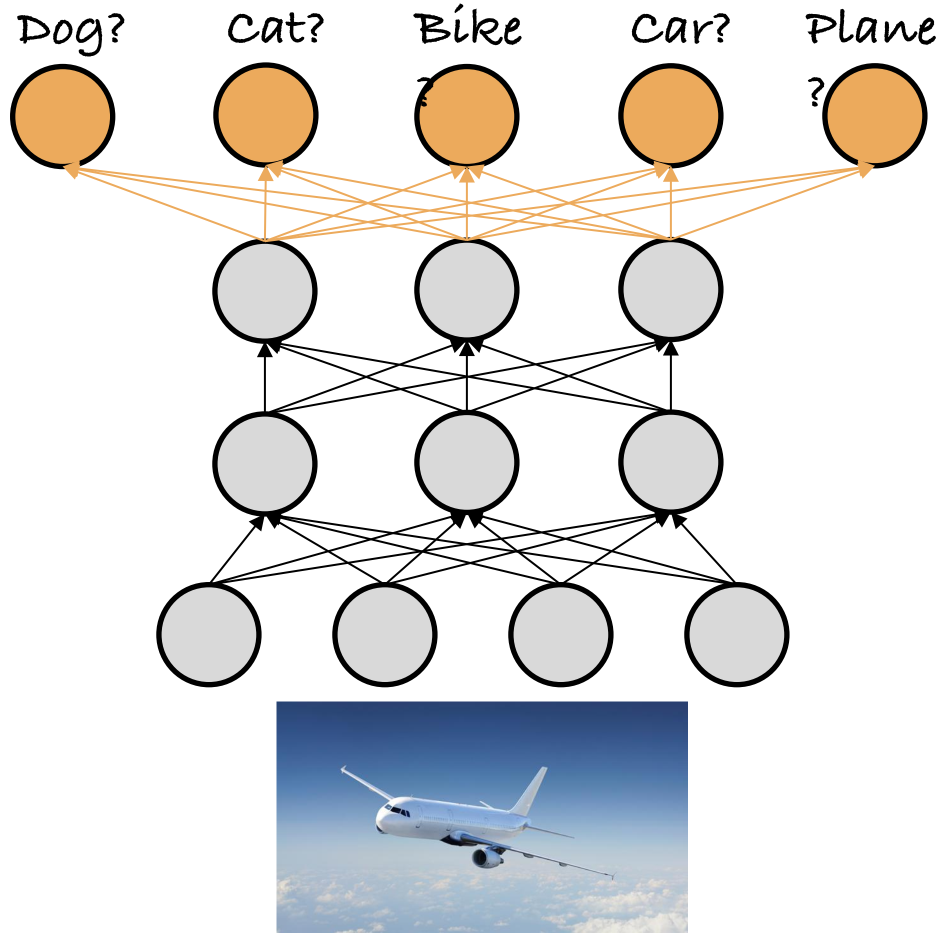
Several possible Convnet architectures possible

- ❑ AlexNet/VGGNet
- ❑ ResNet
- ❑ Google Inception V1-4

Recurrent networks for modelling sequences

Standard inference

N-way classification

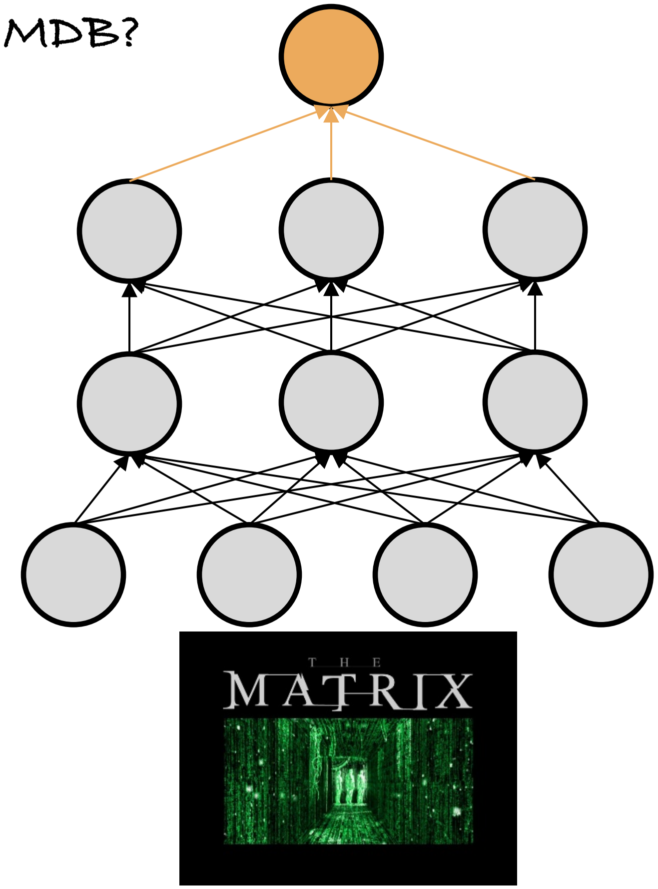


Standard inference

N-way classification

Regression

How popular will this movie be in IMDB?



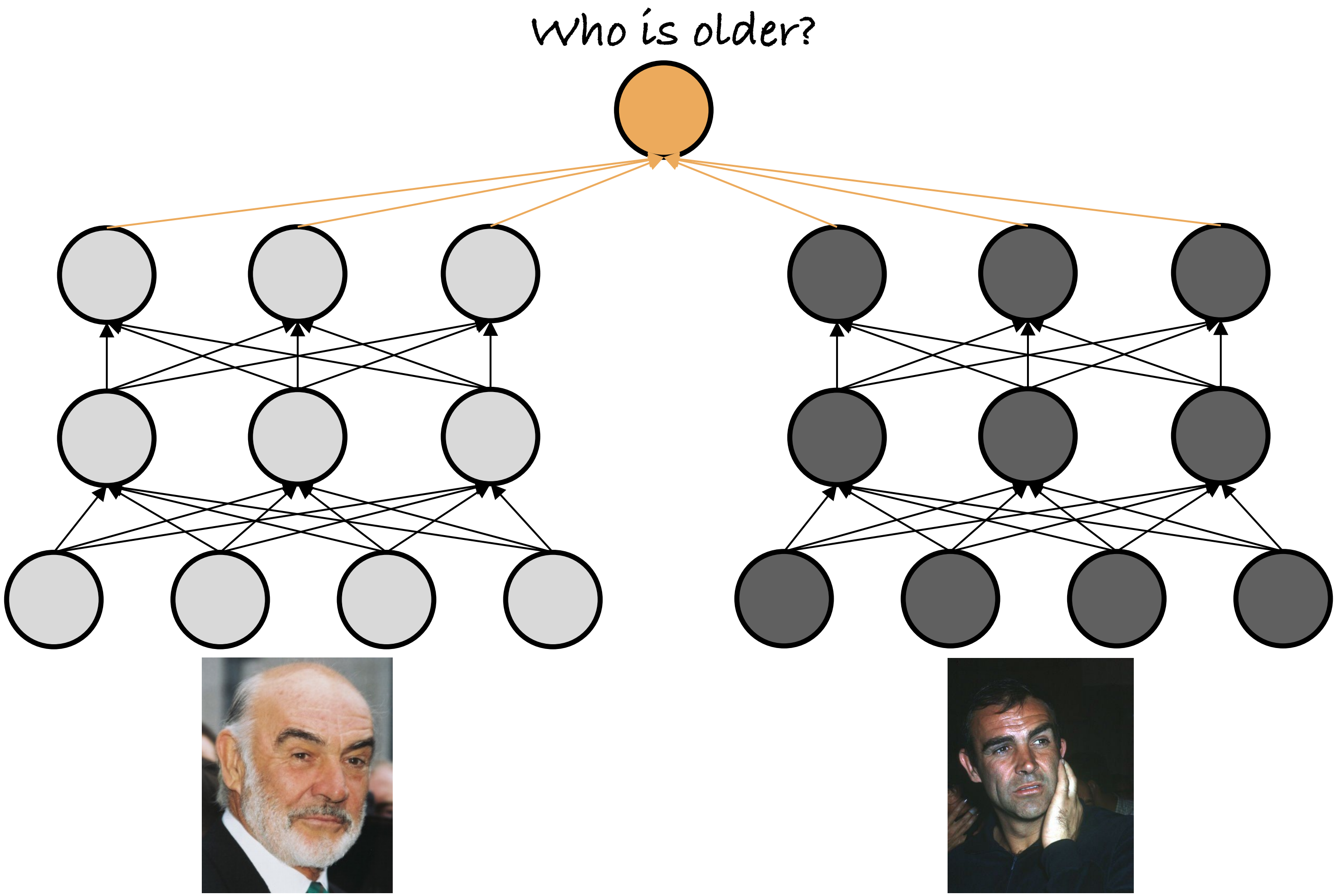
Standard inference

N-way classification

Regression

Ranking

…



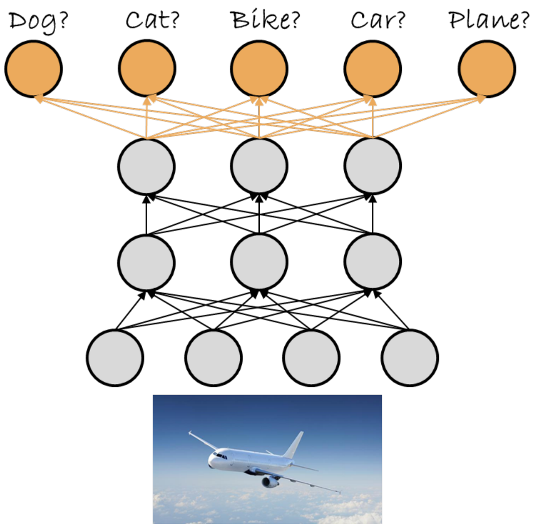
Quiz: What is common?

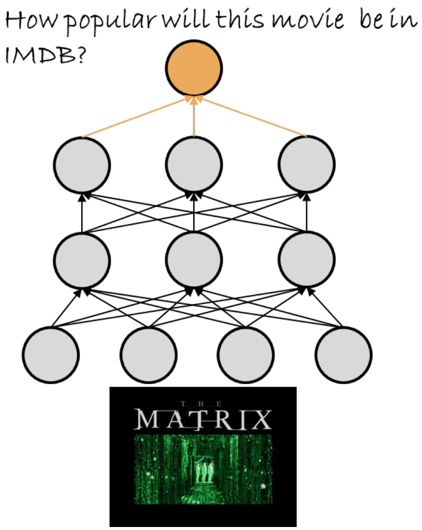
N-way classification

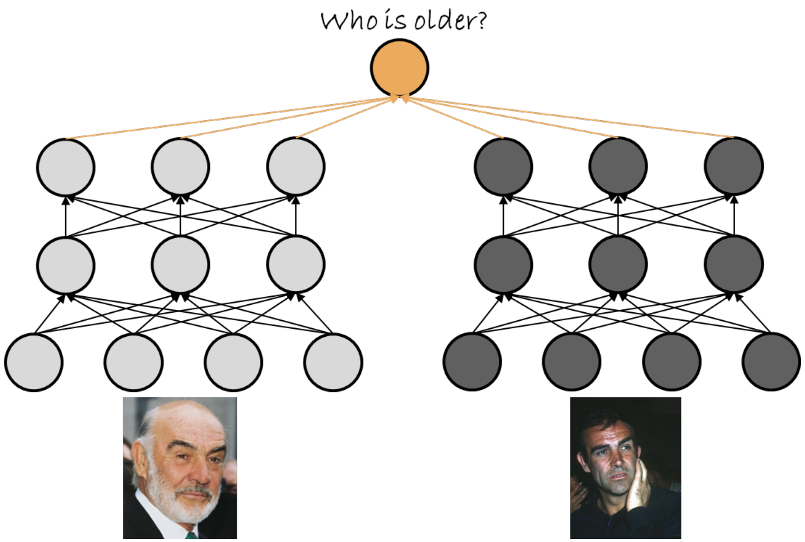
Regression

Ranking

…



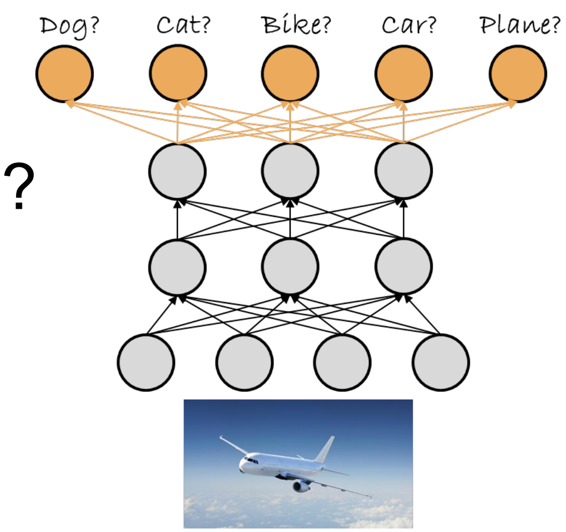


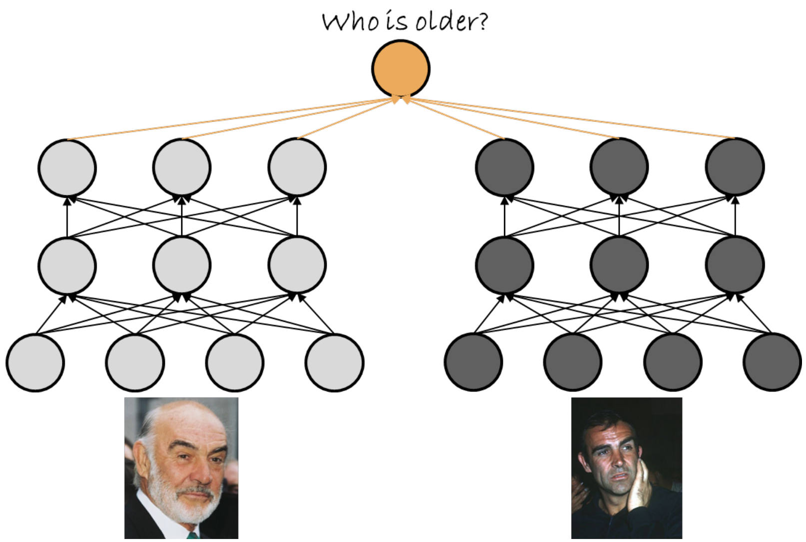


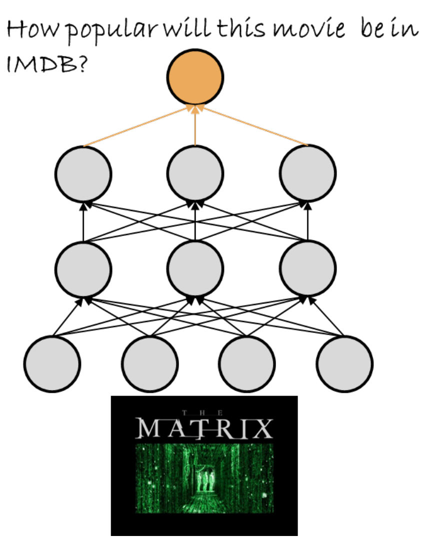
Quiz: What is common?

They all make “single value” predictions

Do all our machine learning tasks
boil down to “single value” predictions?







Beyond “single value” predictions?

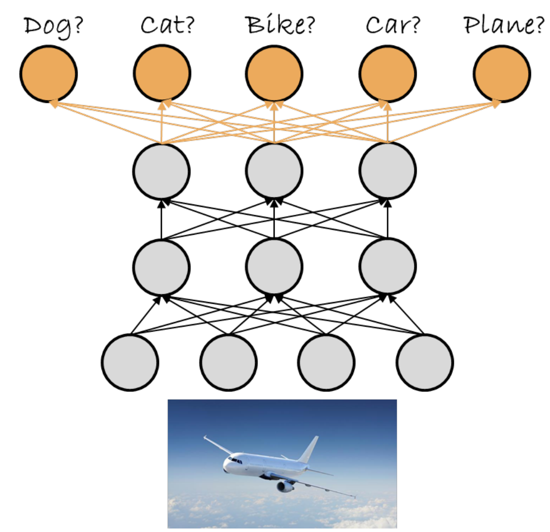
Do all our machine learning tasks boil to “single value” predictions?

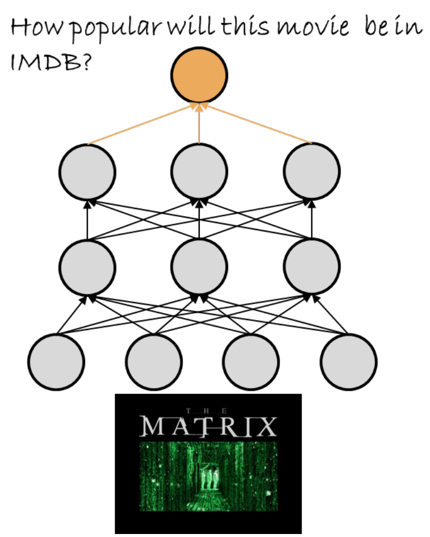
Are there tasks where outputs are somehow correlated?

Is there some structure in this output correlations?

How can we predict such structures?

- Structured prediction





Quiz: Examples?

Object detection

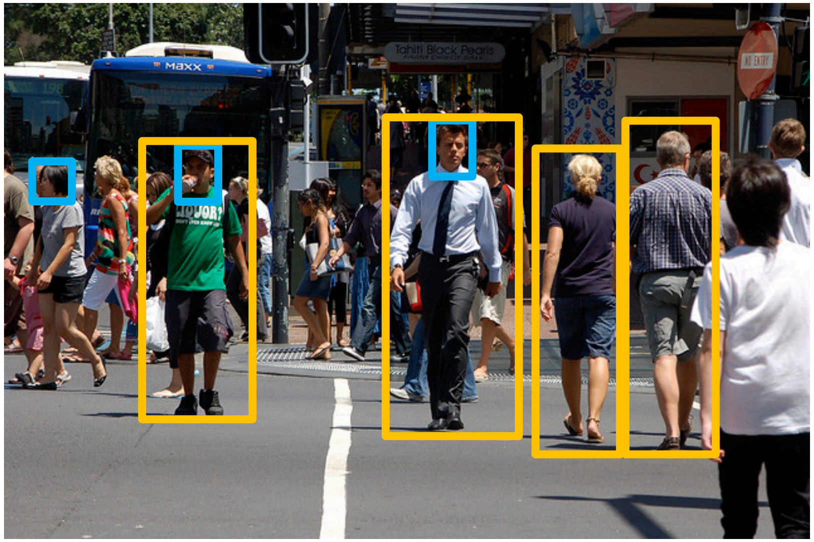
Predict a box around an object

Images

- ❑ Spatial location
- ❑ b(ounding) box

Videos

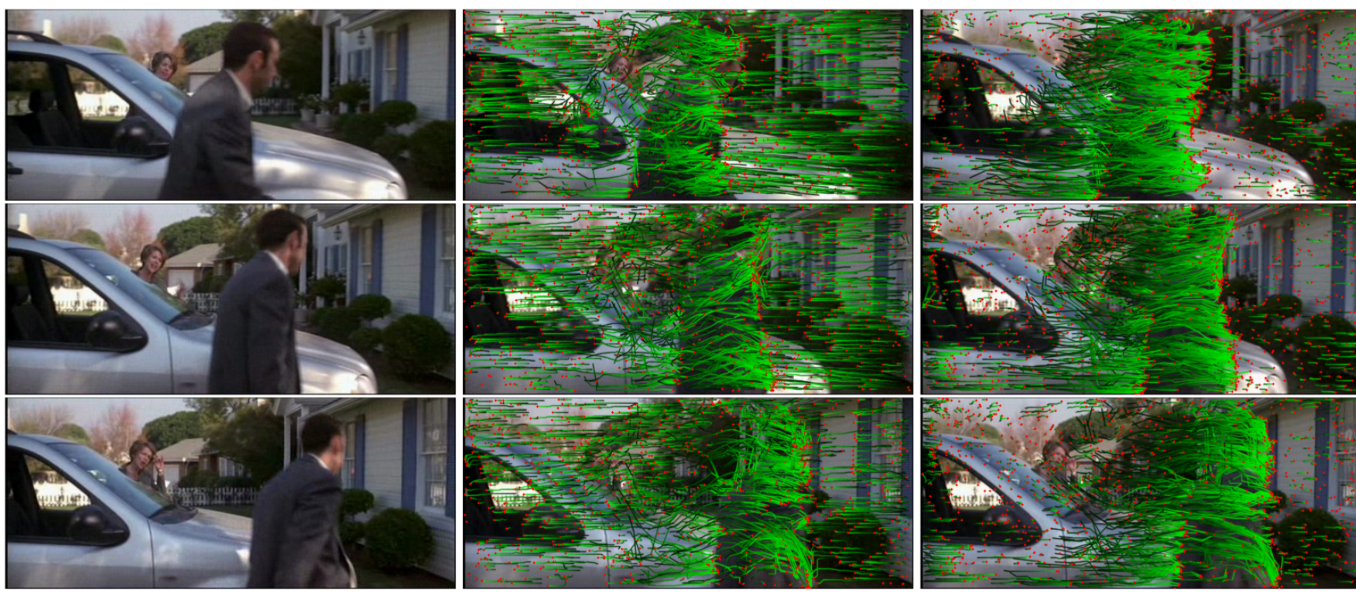
- ❑ Spatio-temporal location
- ❑ bbox@t, bbox@t+1, …



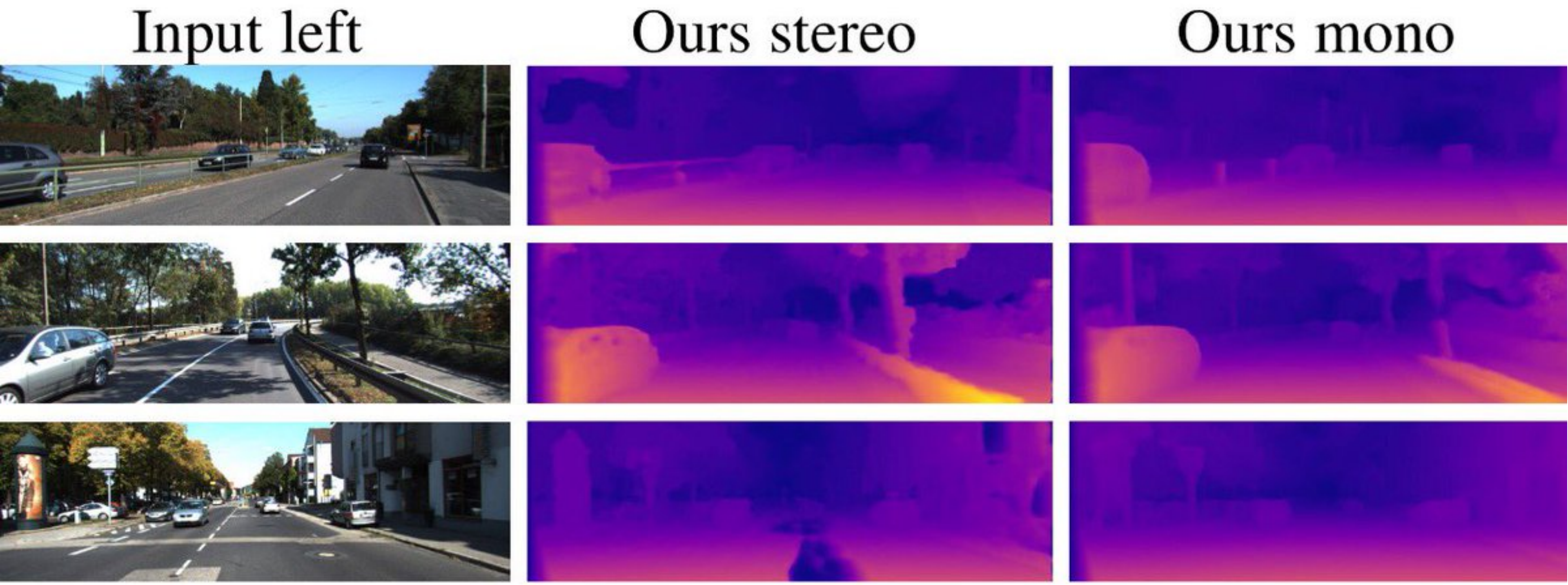
Object segmentation

Image | Class map | Instance map | Part map | Part map (high level

Optical flow & motion estimation

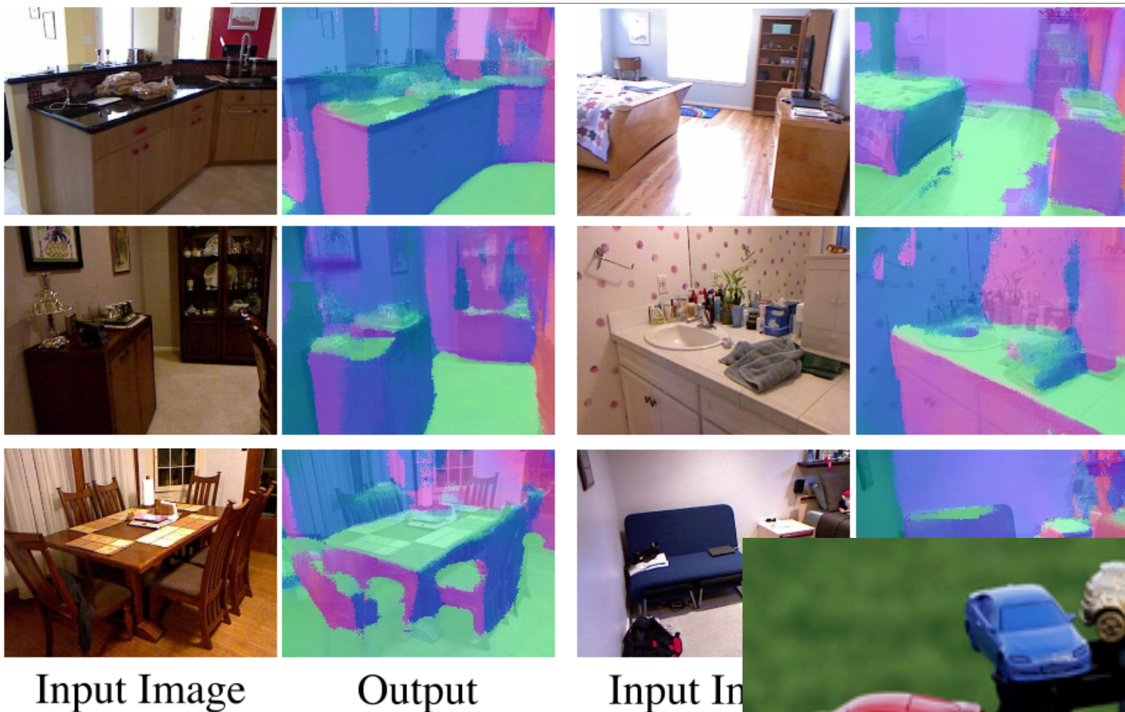


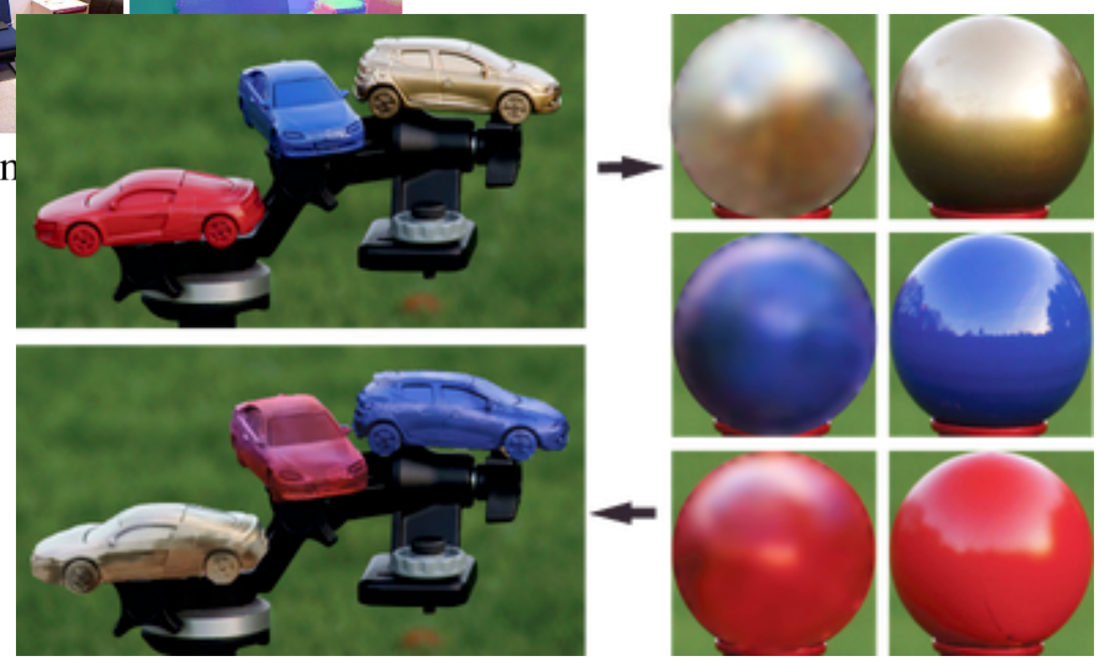
Depth estimation



Godard et al., Unsupervised Monocular Depth Estimation with Left-Right Consistency, 2016

Normals and reflectance estimation





Structured prediction

Prediction goes beyond asking for “single values”

Outputs are complex and output dimensions correlated

Output dimensions have latent structure

Can we make deep networks to return **structured predictions?**

Structured prediction

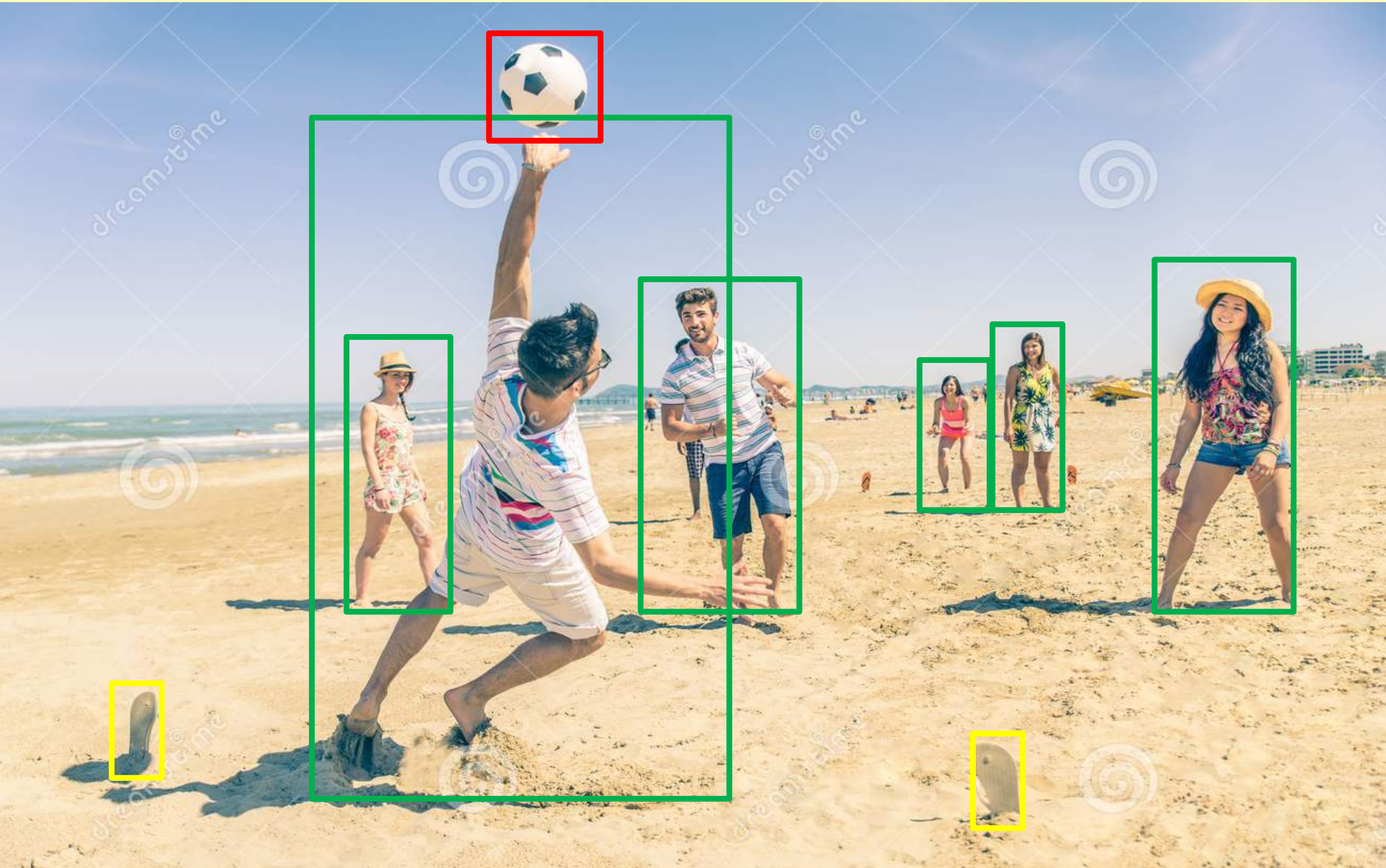
Prediction goes beyond asking for “single values”

Outputs are complex and output dimensions correlated

Output dimensions have latent structure

Can we make deep networks to return **<u>structured predictions?</u>**

Convnets for structured prediction

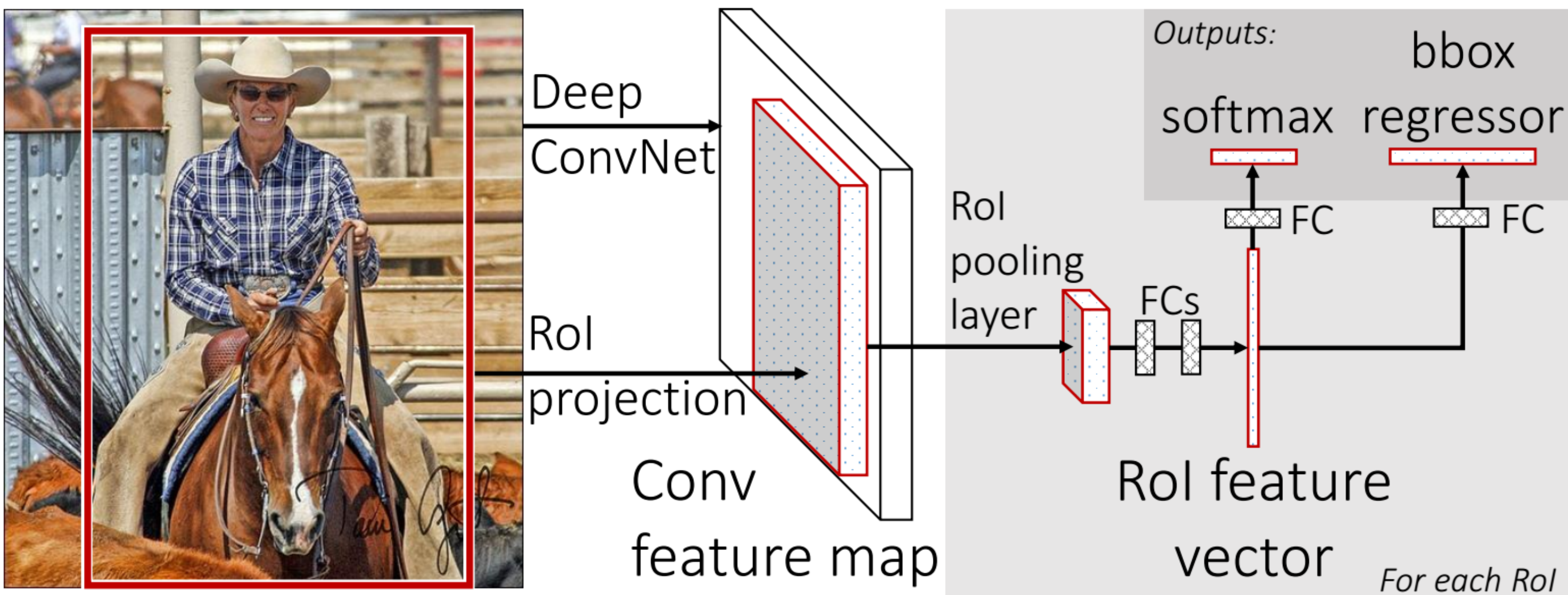


Sliding window on feature maps

Selective Search Object Proposals [Uijlings2013]

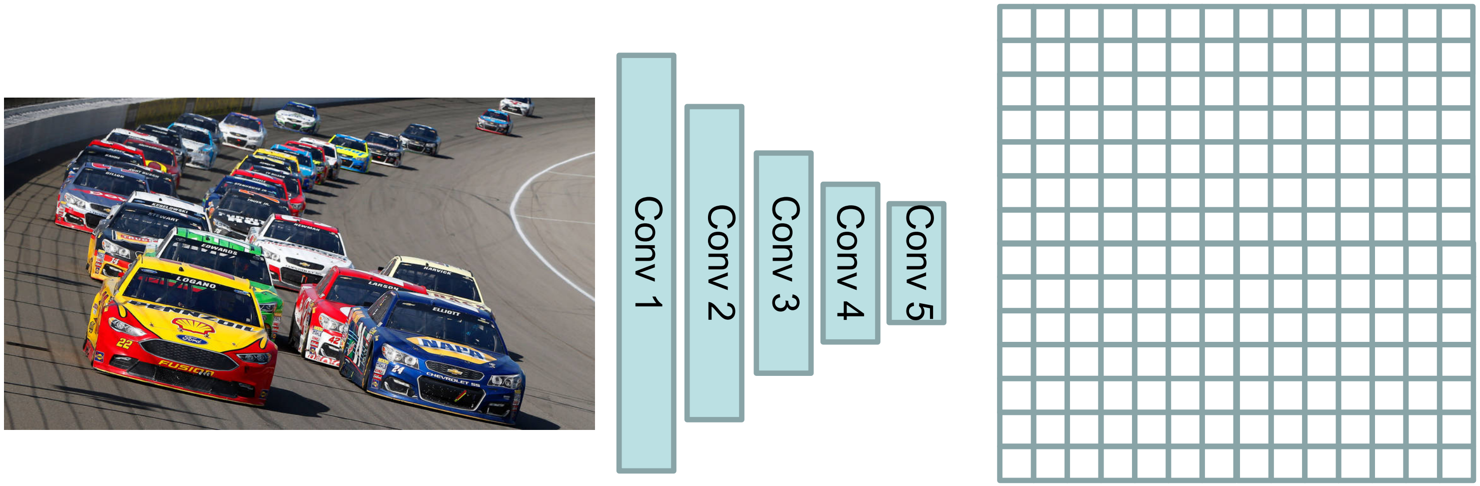
SPPnet [He2014]

Fast R-CNN [Girshick2015]



Fast R-CNN: Steps

Process the whole image up to conv5

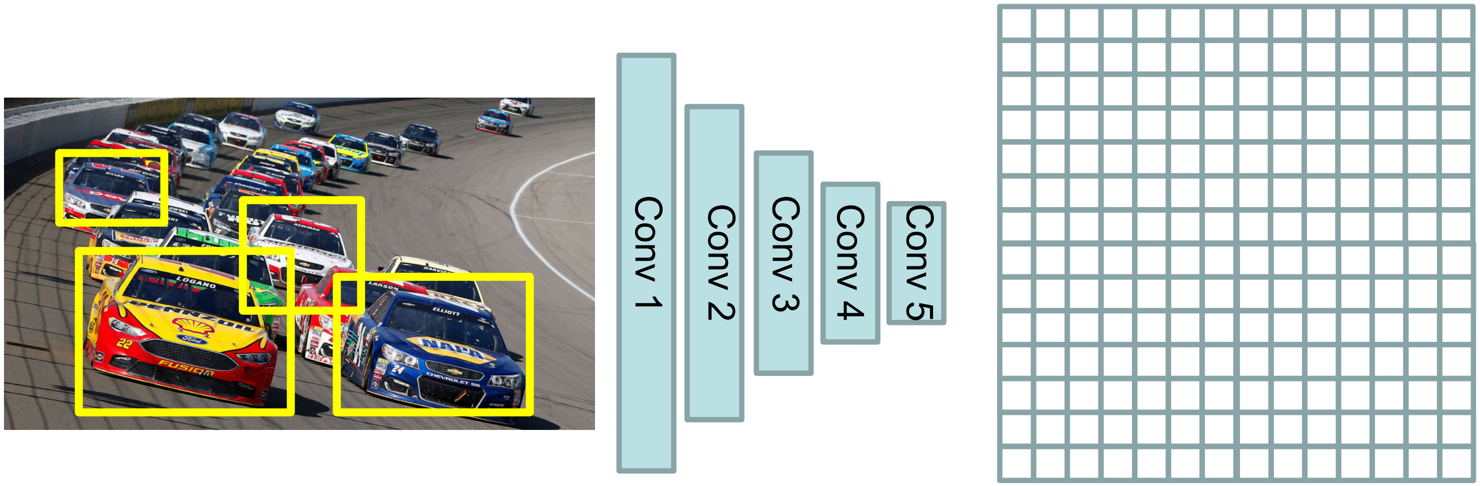


Conv 5 feature map

Fast R-CNN: Steps

Process the whole image up to conv5

Compute possible locations for objects



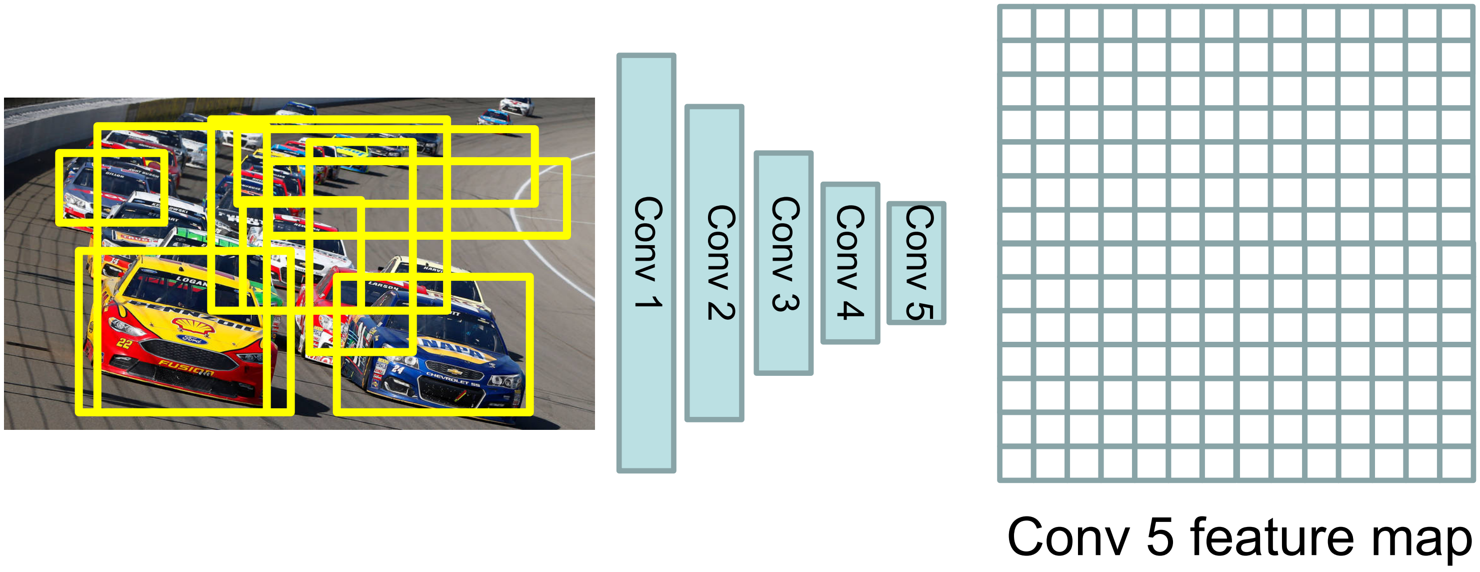
Conv 5 feature map

Fast R-CNN: Steps

Process the whole image up to conv5

Compute possible locations for objects

- some correct, most wrong



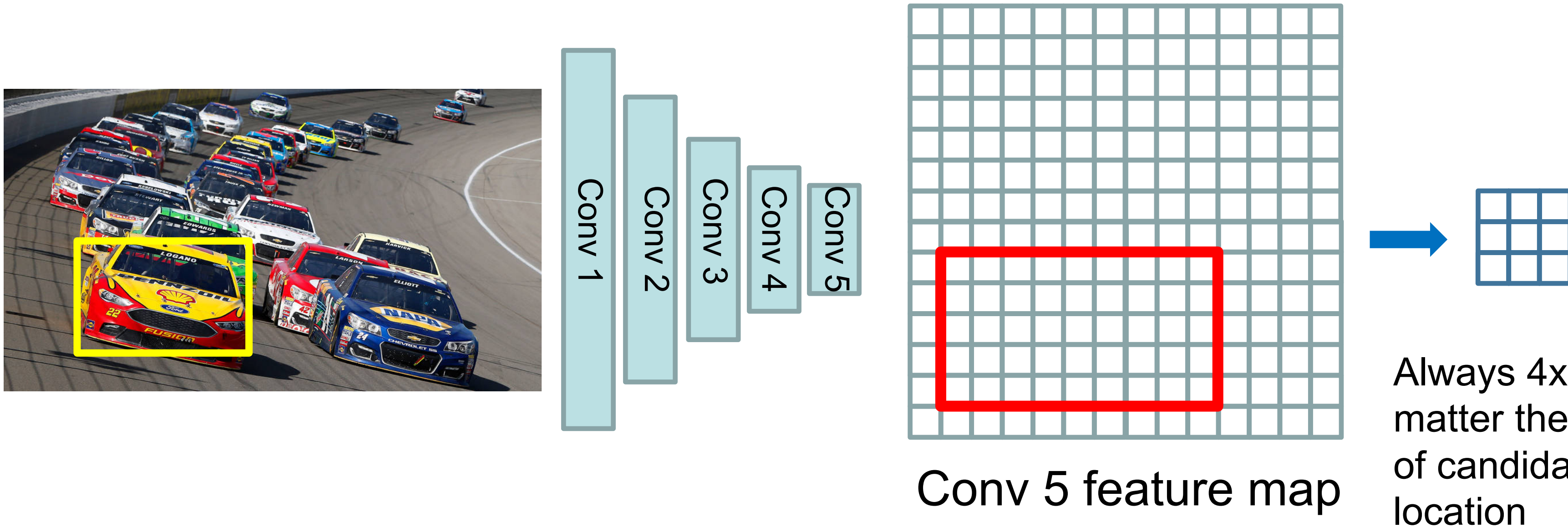
Fast R-CNN: Steps

Process the whole image up to conv5

Compute possible locations for objects

- ❑ some correct, most wrong

Given single location → ROI pooling module extracts fixed length feature



Always 4x4 no matter the size of candidate location

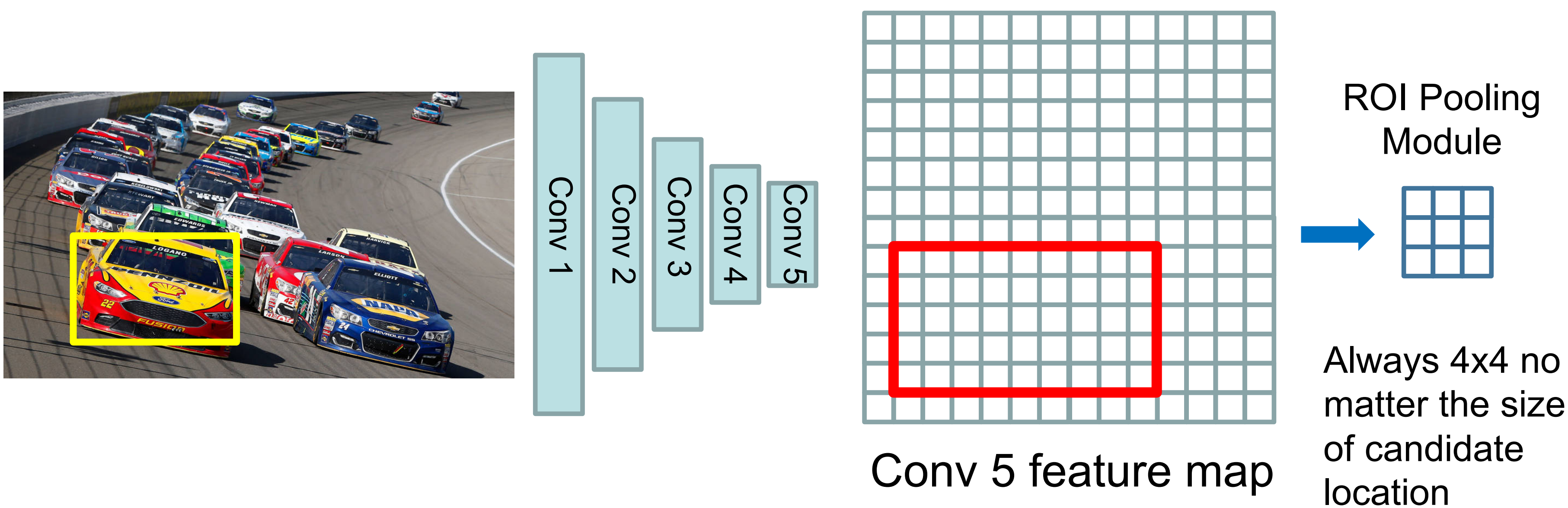
Fast R-CNN: Steps

Process the whole image up to conv5

Compute possible locations for objects

- some correct, most wrong

Given single location → ROI pooling module extracts fixed length feature



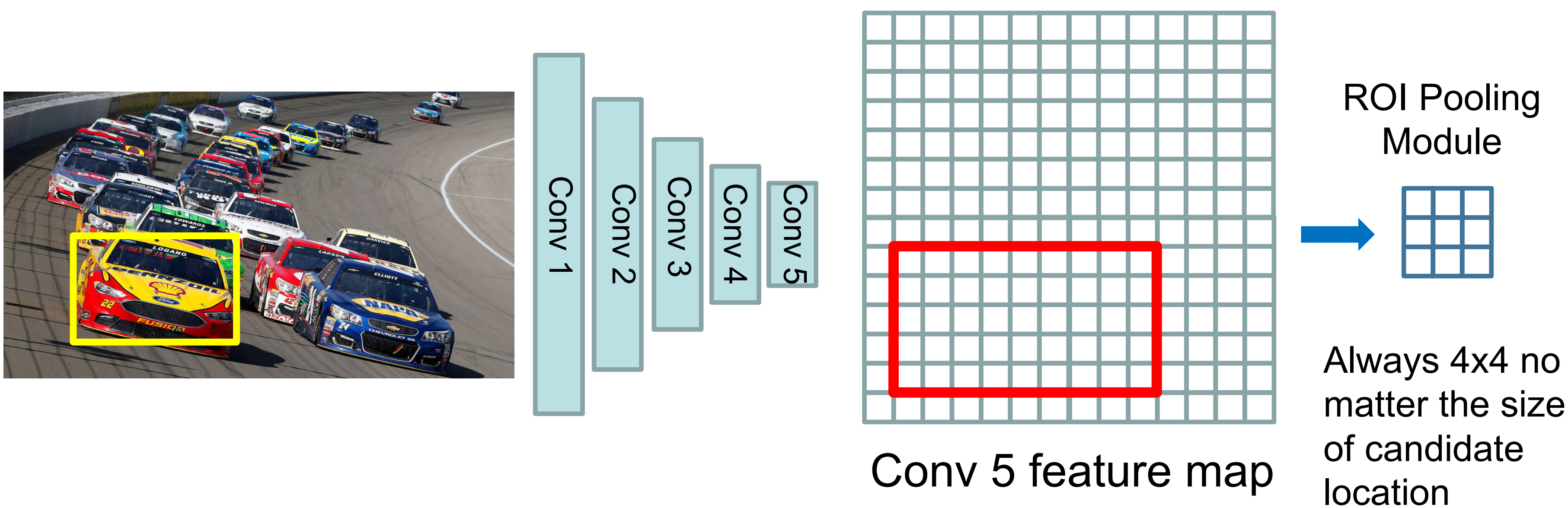
Fast R-CNN: Steps

Process the whole image up to conv5

Compute possible locations for objects

- some correct, most wrong

Given single location → ROI pooling module extracts fixed length feature



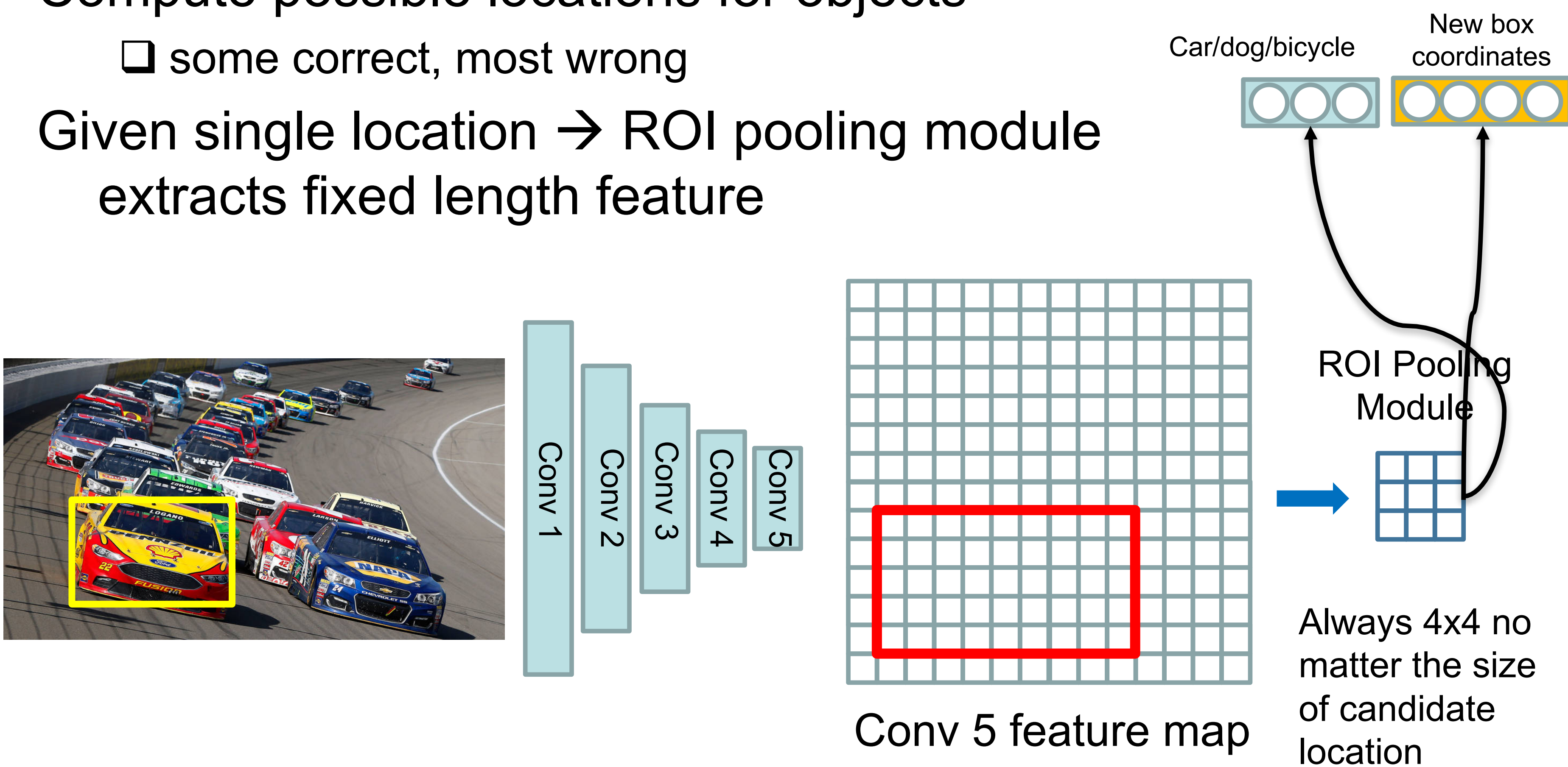
Fast R-CNN: Steps

Process the whole image up to conv5

Compute possible locations for objects

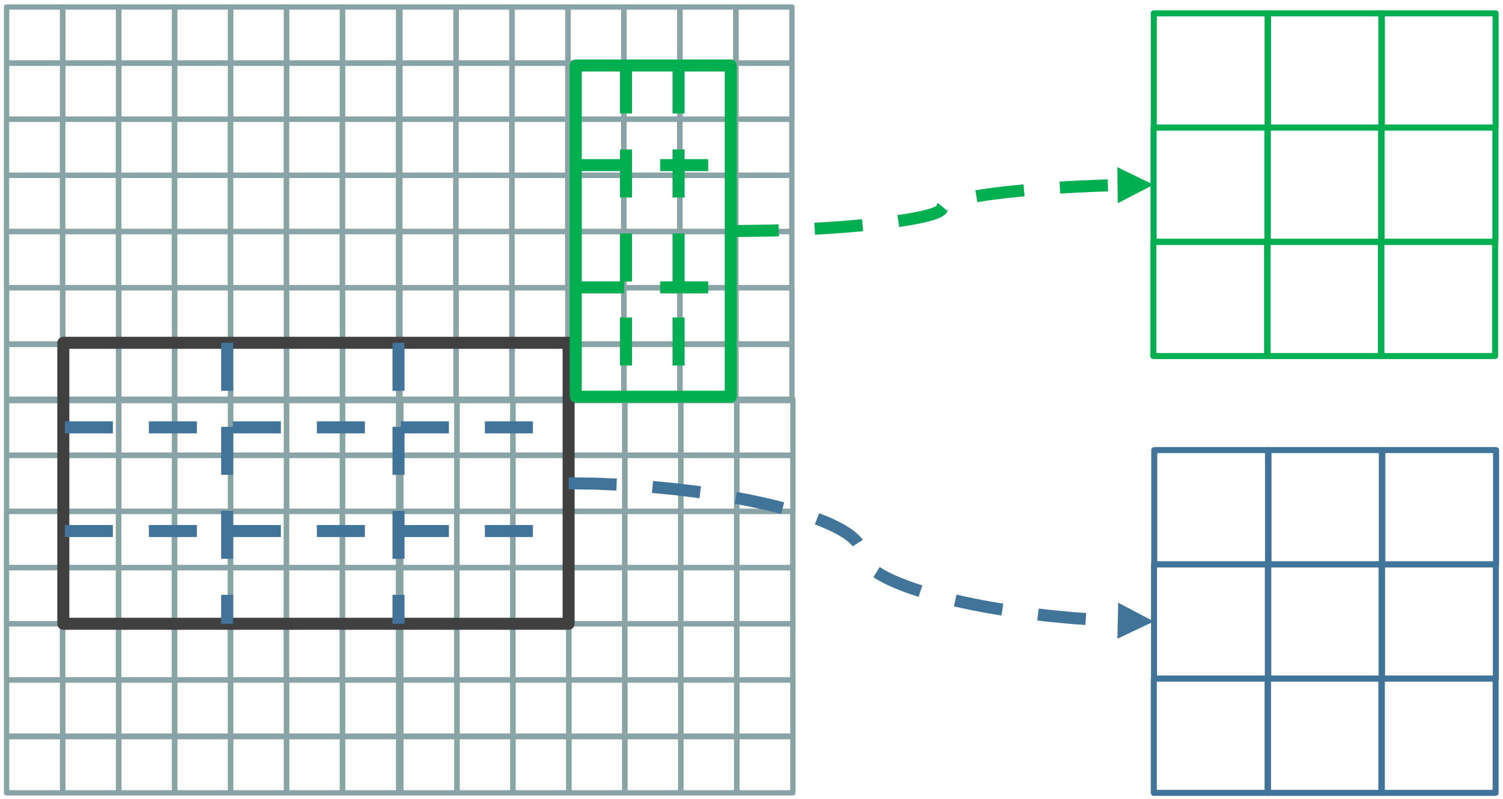
- some correct, most wrong

Given single location → ROI pooling module extracts fixed length feature



Divide feature map in TxT cells

- ❑ Cell size changes depending on the size of the candidate location



Always 3x3 no matter the size of candidate location

Some results

Fast R-CNN

Reuse convolutions for different candidate boxes

- ❑ Compute feature maps only once

Region-of-Interest pooling

- ❑ Define stride relatively → box width divided by predefined number of “poolings” T
- ❑ Fixed length vector

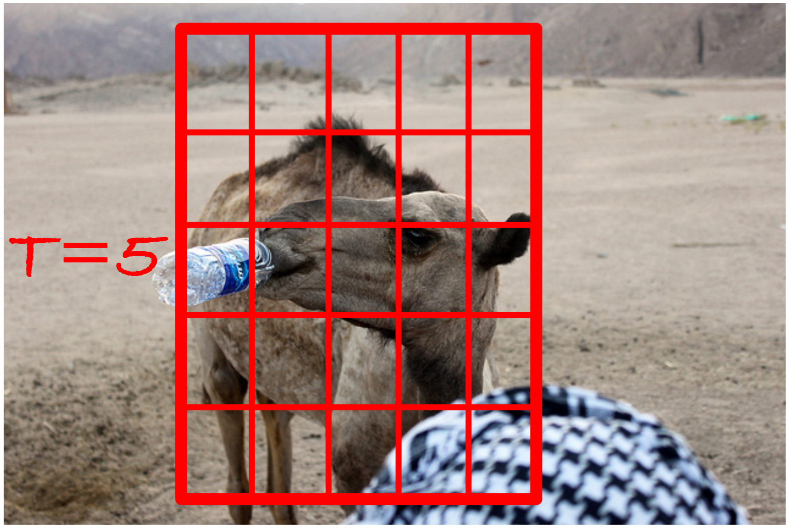
End-to-end training!

(Very) Accurate object detection

(Very) Faster

- ❑ Less than a second per image

External box proposals needed



Faster R-CNN [Girshick2016]

Fast R-CNN

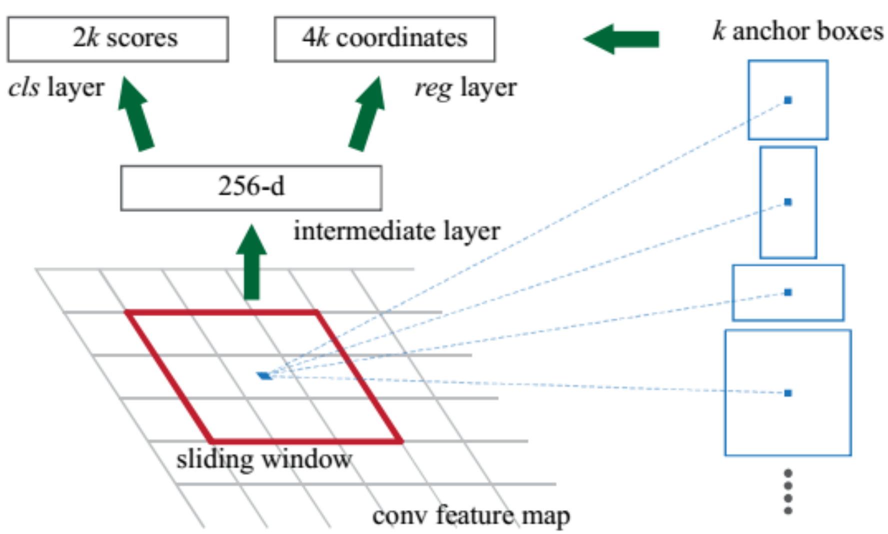
- ❑ external candidate locations

Faster R-CNN

- ❑ deep network proposes candidate

Slide the feature map

- ❑ k anchor boxes per slide



Region Proposal Network

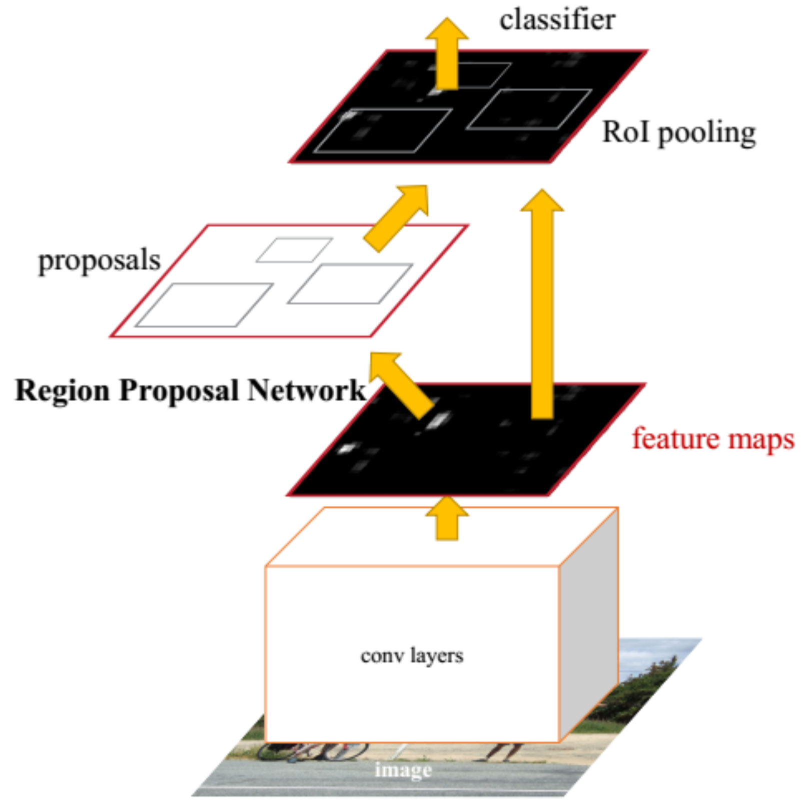


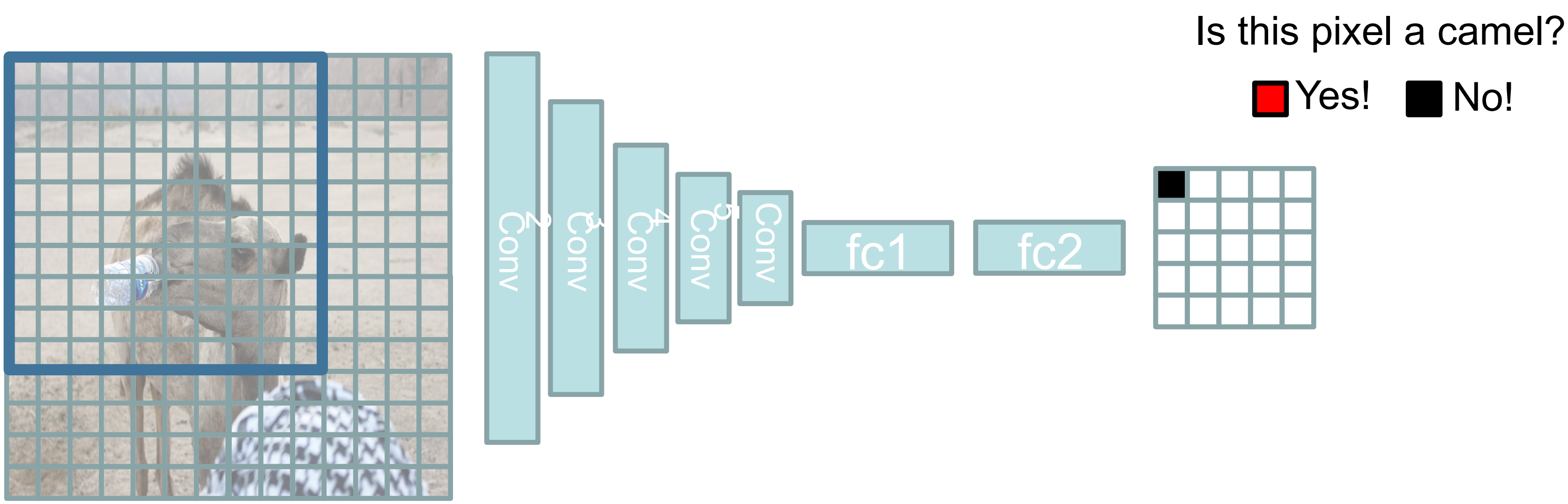
Figure 2: Faster R-CNN is a single, unified network for object detection. The RPN module serves as the 'attention' of this unified network.

Going Fully Convolutional

[LongCVPR2014]

Image larger than network input

- ❑ slide the network

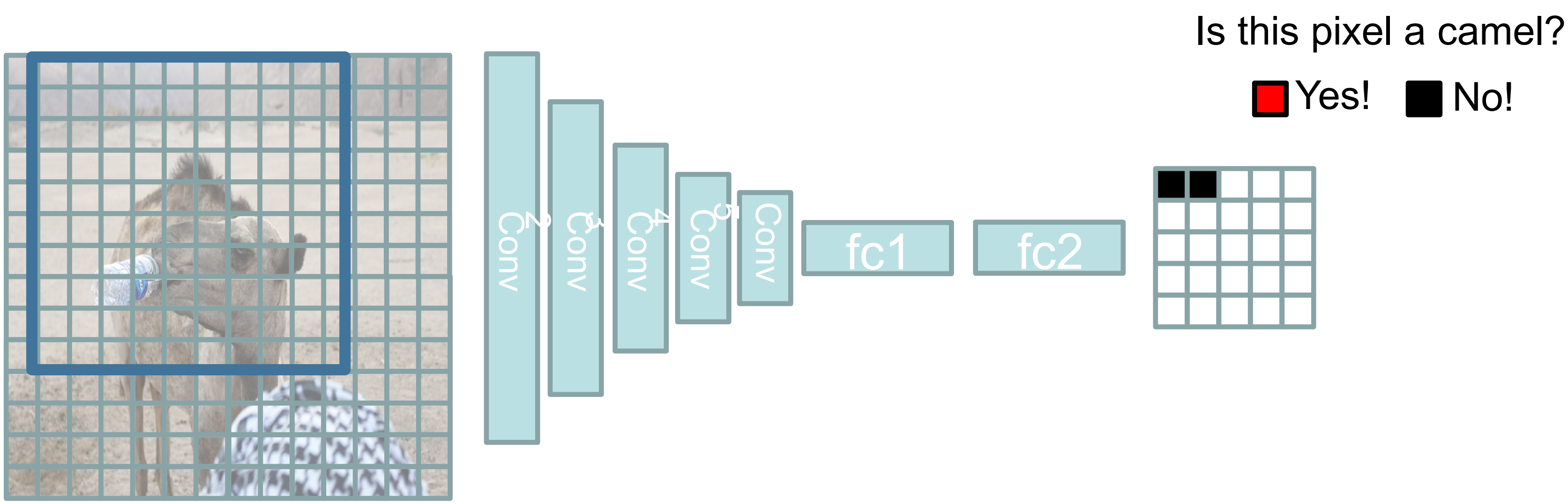


Going Fully Convolutional

[LongCVPR2014]

Image larger than network input

- slide the network

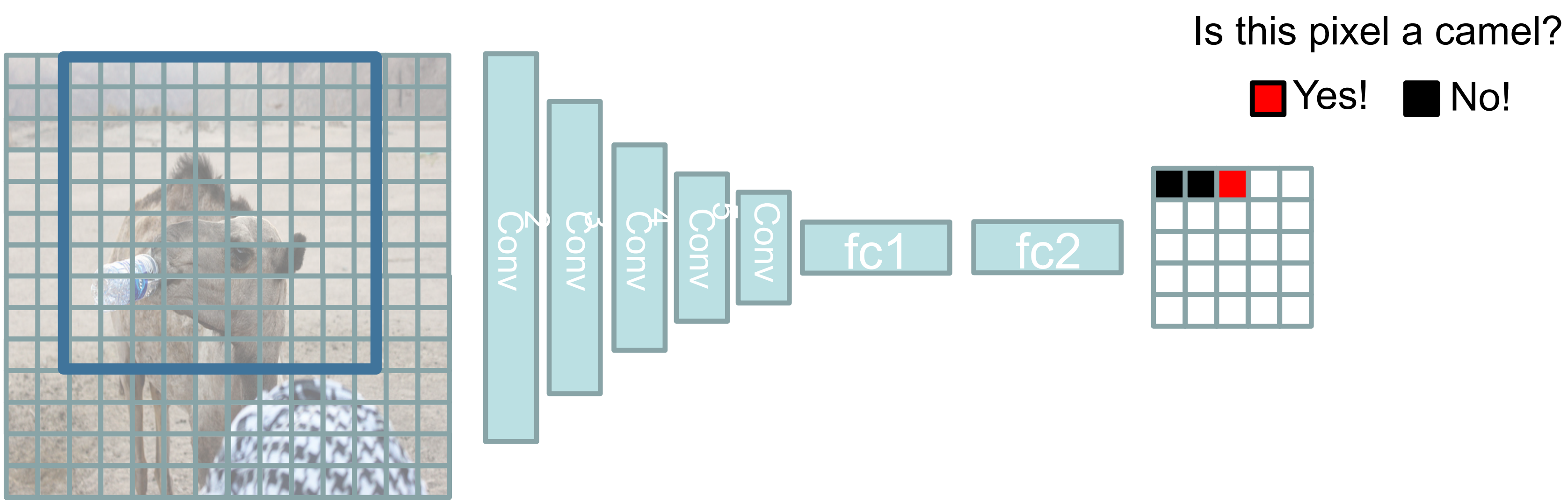


Going Fully Convolutional

[LongCVPR2014]

Image larger than network input

- ❑ slide the network

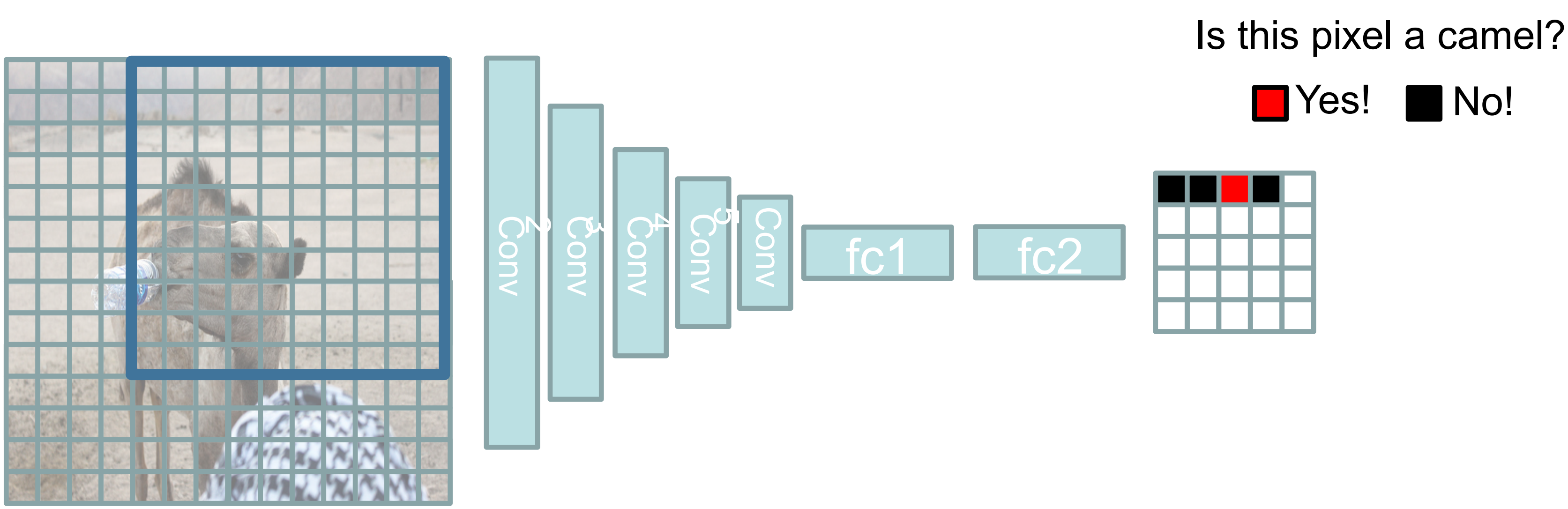


Going Fully Convolutional

[LongCVPR2014]

Image larger than network input

- ❑ slide the network

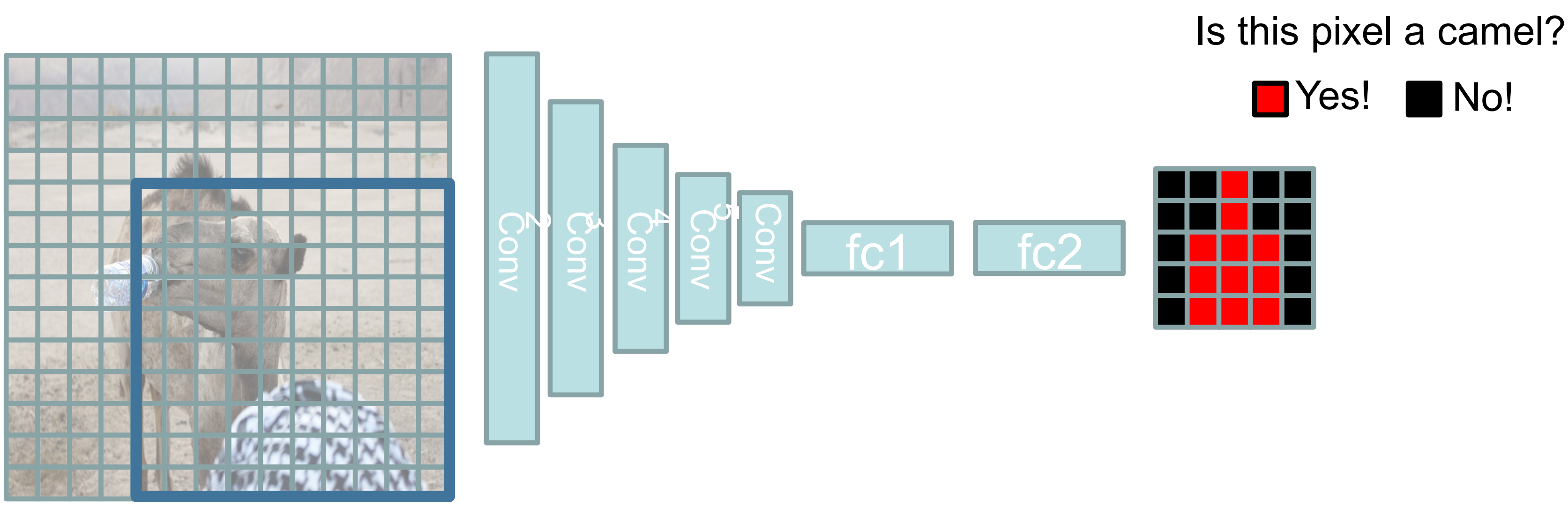


Going Fully Convolutional

[LongCVPR2014]

Image larger than network input

- ❑ slide the network



Fully Convolutional Networks

[LongCVPR2014]

Connect intermediate layers to output

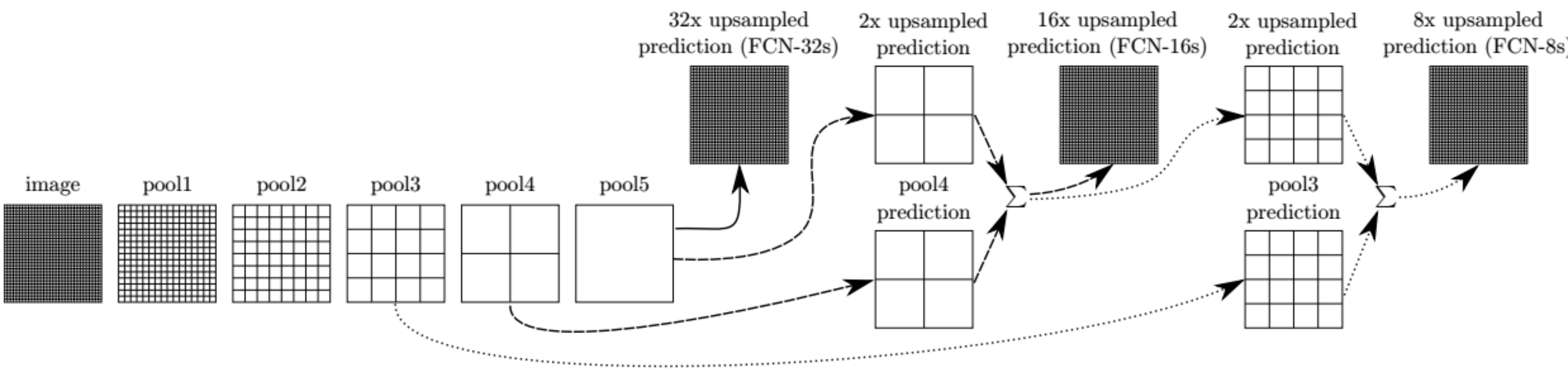


Figure 3. Our DAG nets learn to combine coarse, high layer information with fine, low layer information. Layers are shown as grids that reveal relative spatial coarseness. Only pooling and prediction layers are shown; intermediate convolution layers (including our converted fully connected layers) are omitted. Solid line (FCN-32s): Our single-stream net, described in Section 4.1, upsamples stride 32 predictions back to pixels in a single step. Dashed line (FCN-16s): Combining predictions from both the final layer and the `pool4` layer, at stride 16, lets our net predict finer details, while retaining high-level semantic information. Dotted line (FCN-8s): Additional predictions from `pool3`, at stride 8, provide further precision.

Fully Convolutional Networks

Output is too coarse

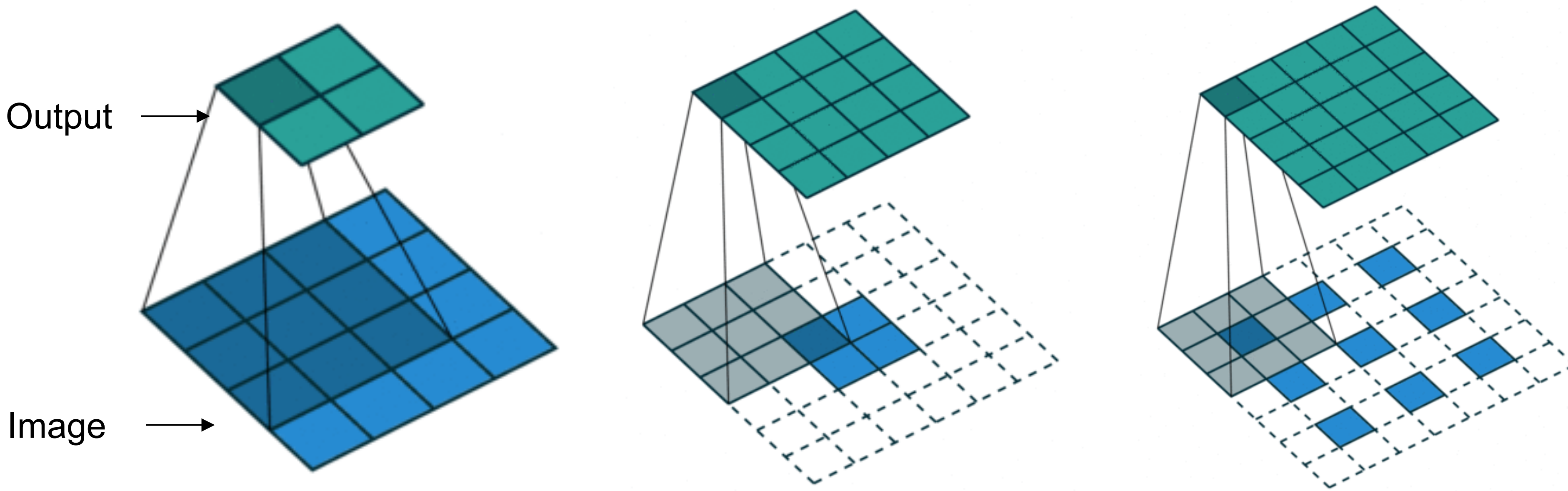
- Image Size 500x500, Alexnet Input Size: 227x227 → Output: 10x10

How to obtain dense predictions?

Upconvolution

- Other names: deconvolution, transposed convolution, fractionally-strided convolutions

Deconvolutional modules



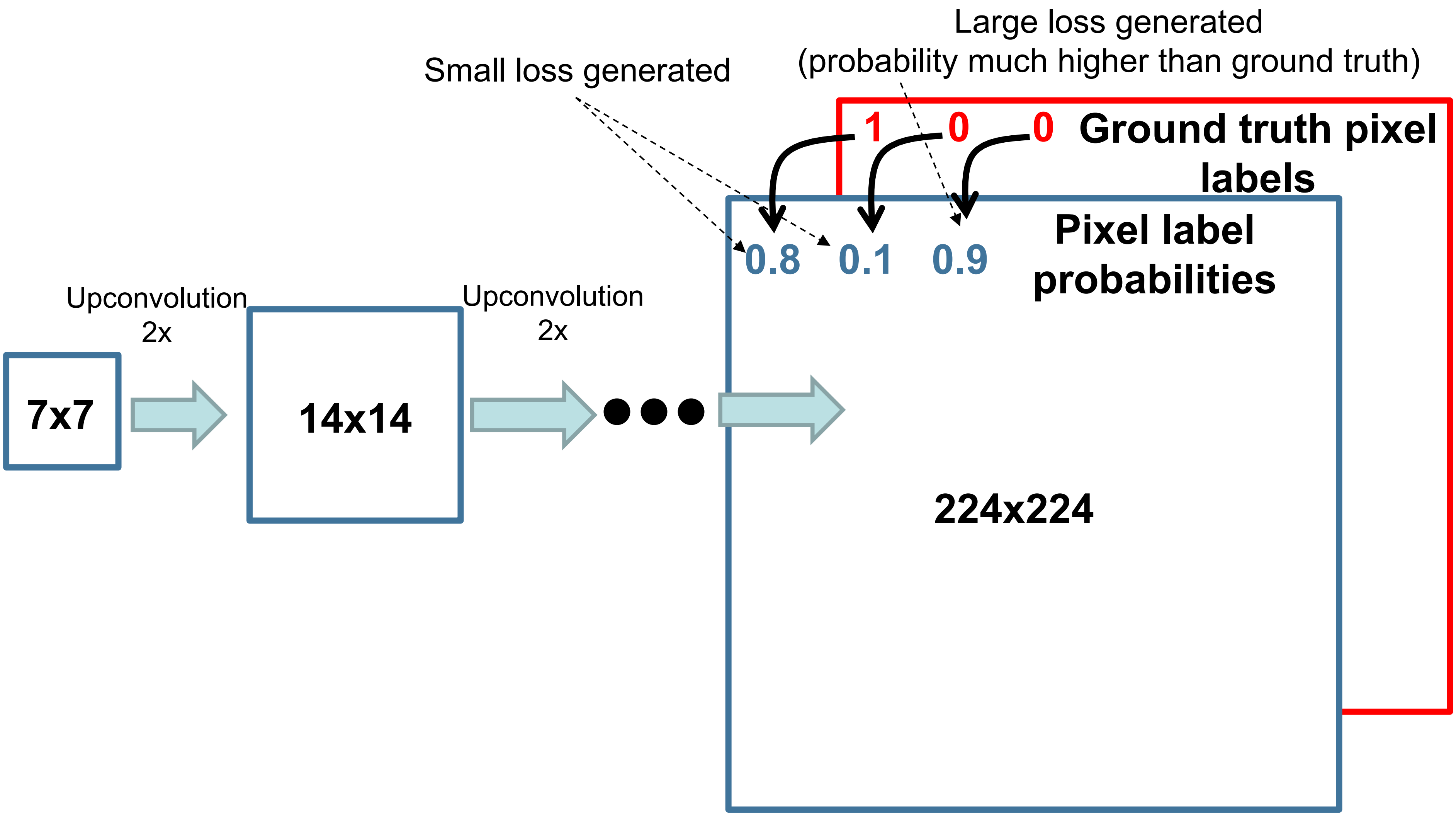
<u>Convolution</u>
No padding, no strides

<u>Upconvolution</u>
No padding, no strides

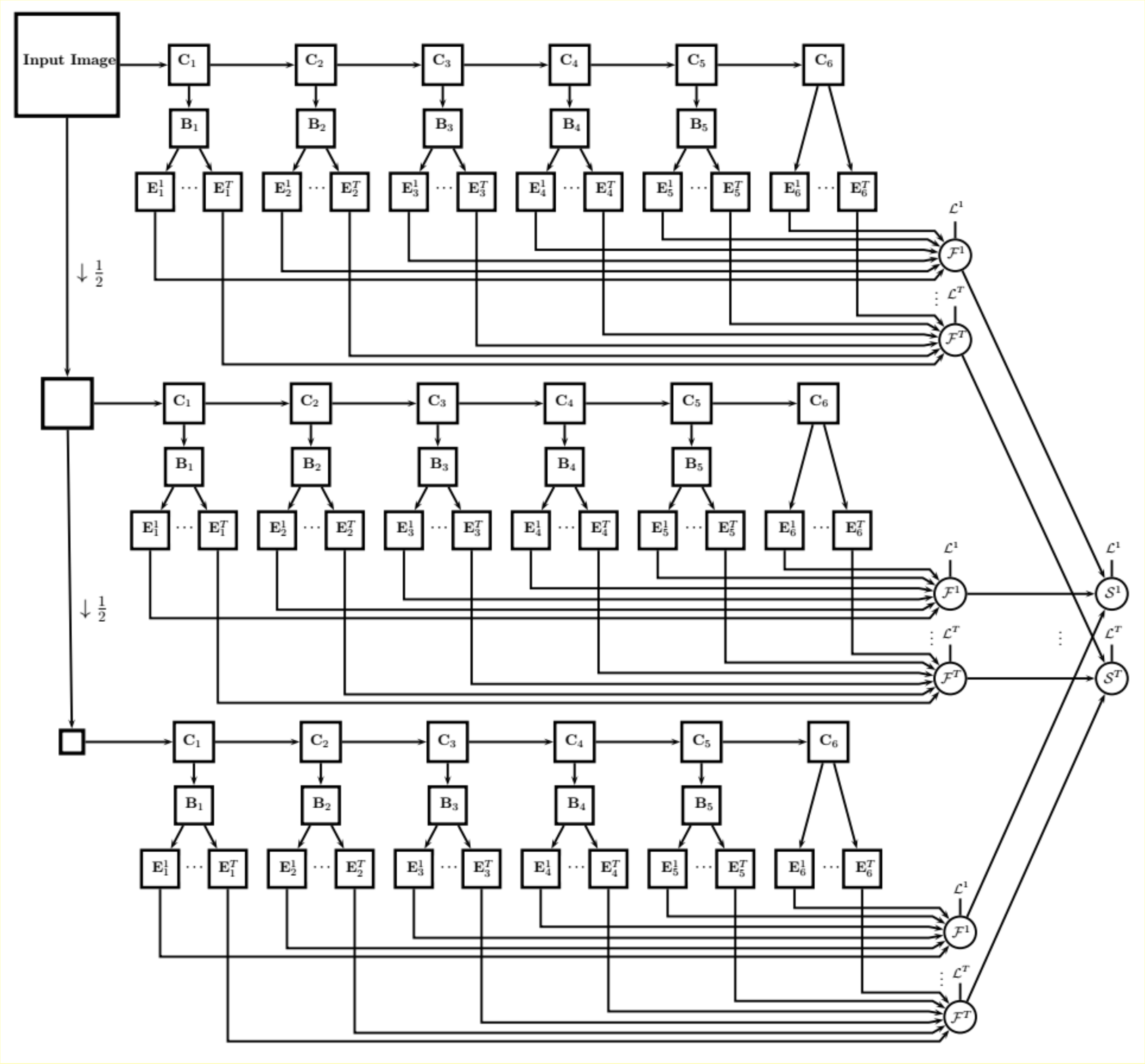
<u>Upconvolution</u>
Padding, strides

https://github.com/vdumoulin/conv_arithmetic

Coarse → Fine Output



Structured losses



Deep ConvNets with CRF loss

[Chen, Papandreou 2016]

Segmentation map is good but not pixel-precise

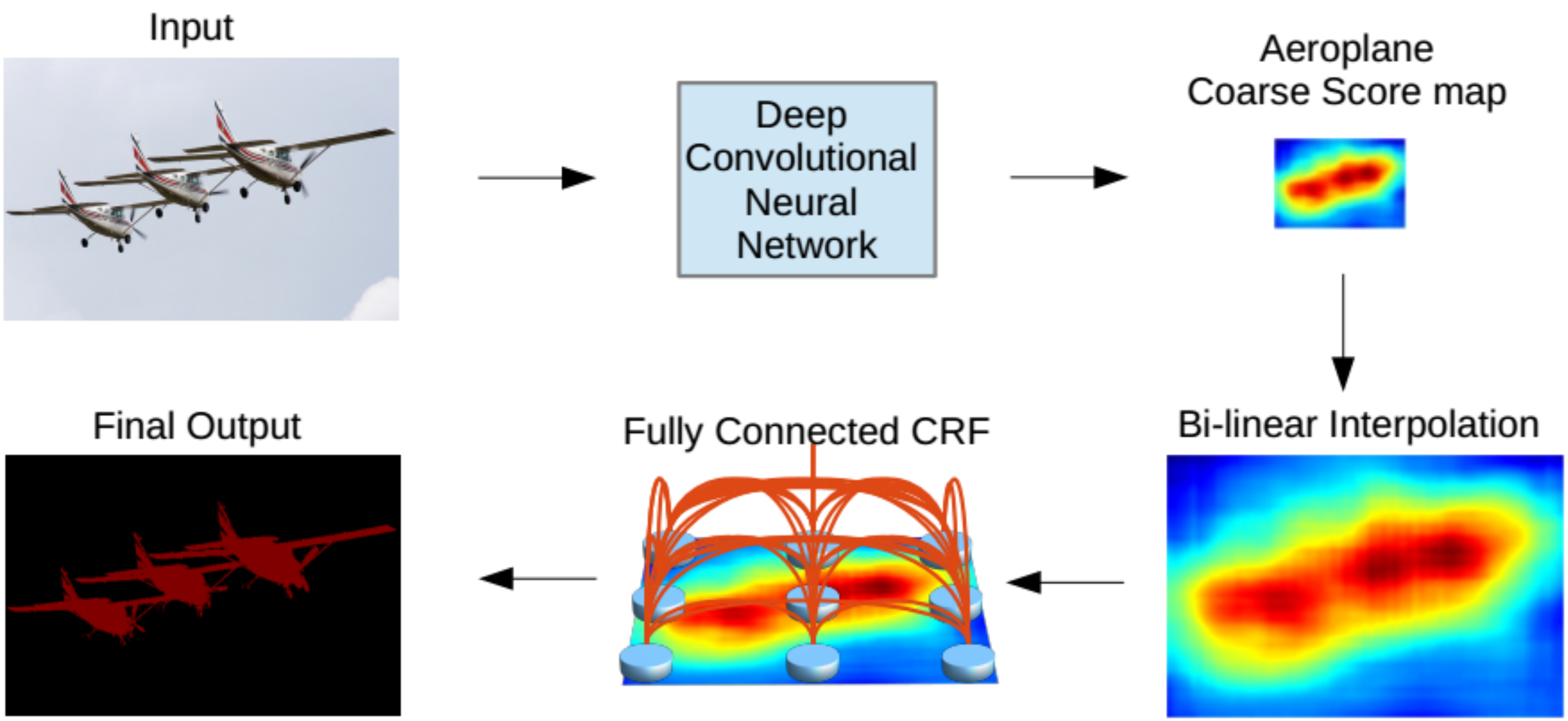
- Details around boundaries are lost

Cast fully convolutional outputs as unary potentials

Consider pairwise potentials between output dimensions

Deep ConvNets with CRF loss

[Chen, Papandreou 2016]



Deep ConvNets with CRF loss

[Chen, Papandreou 2016]

Segmentation map is good but not pixel-precise

- Details around boundaries are lost

Cast fully convolutional outputs as unary potentials

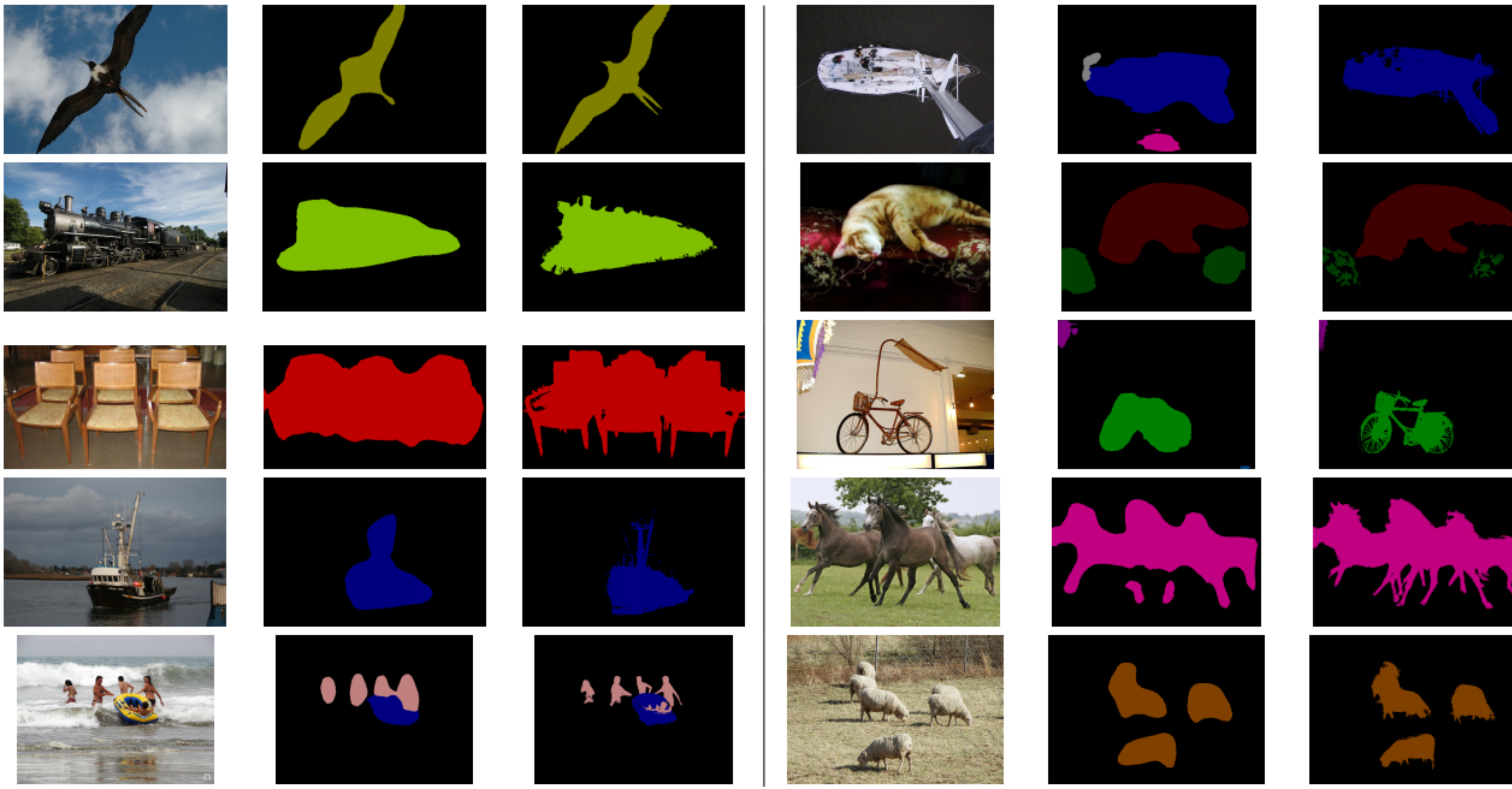
Consider pairwise potentials between output dimensions

Include Fully Connected CRF loss to refine segmentation

$$\underbrace{E(x)}_{\text{Total loss}} = \sum \underbrace{\theta_i(x_i)}_{\text{Unary loss}} + \sum \underbrace{\theta_{ij}(x_i, x_j)}_{\text{Pairwise loss}}$$

$$\theta_{ij}(x_i, x_j) \sim w_1 \exp\left(-\alpha\left|p_i - p_j\right|^2 - \beta\left|I_i - I_j\right|^2\right) + w_2 \exp(-\gamma\left|p_i - p_j\right|^2)$$

Examples



Multi-task learning

The total loss is the summation of the per task losses

The per task loss relies on the common weights (VGGnet) and the weights specialized for the task

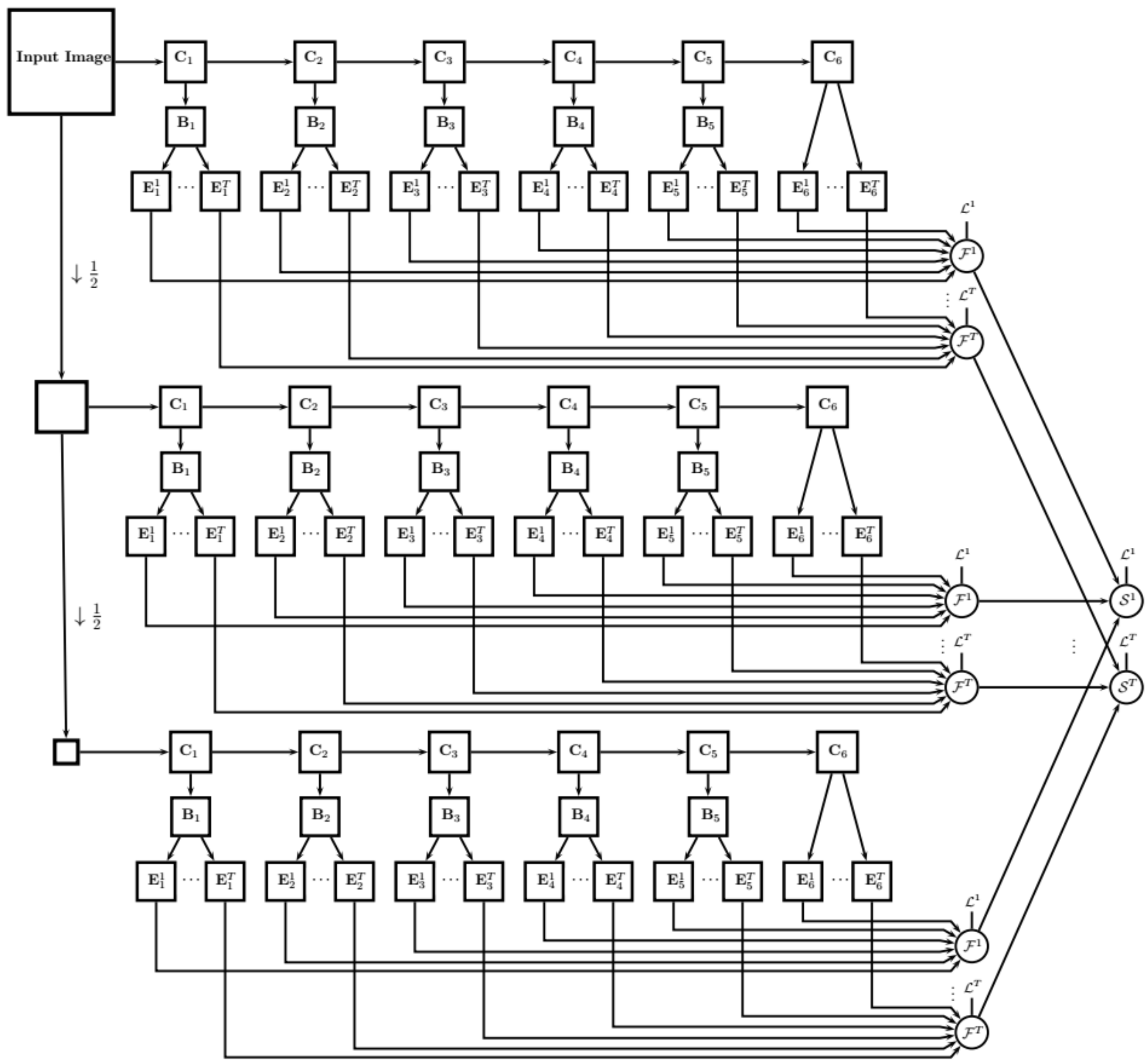
$$\mathcal{L}_{total} = \sum_{task} \mathcal{L}_{task}(\theta_{common}, \theta_{task}) + \mathcal{R}(\theta_{task})$$

One training image might contain specific only annotations

- Only a particular task is "run" for that image

Gradients per image are computed for tasks available for the image only

Ubernet [Kokkinos2016]



One image → Several tasks

Per image we can predict, boundaries, segmentation, detection, …

- Why separately?

Solve multiple tasks simultaneously

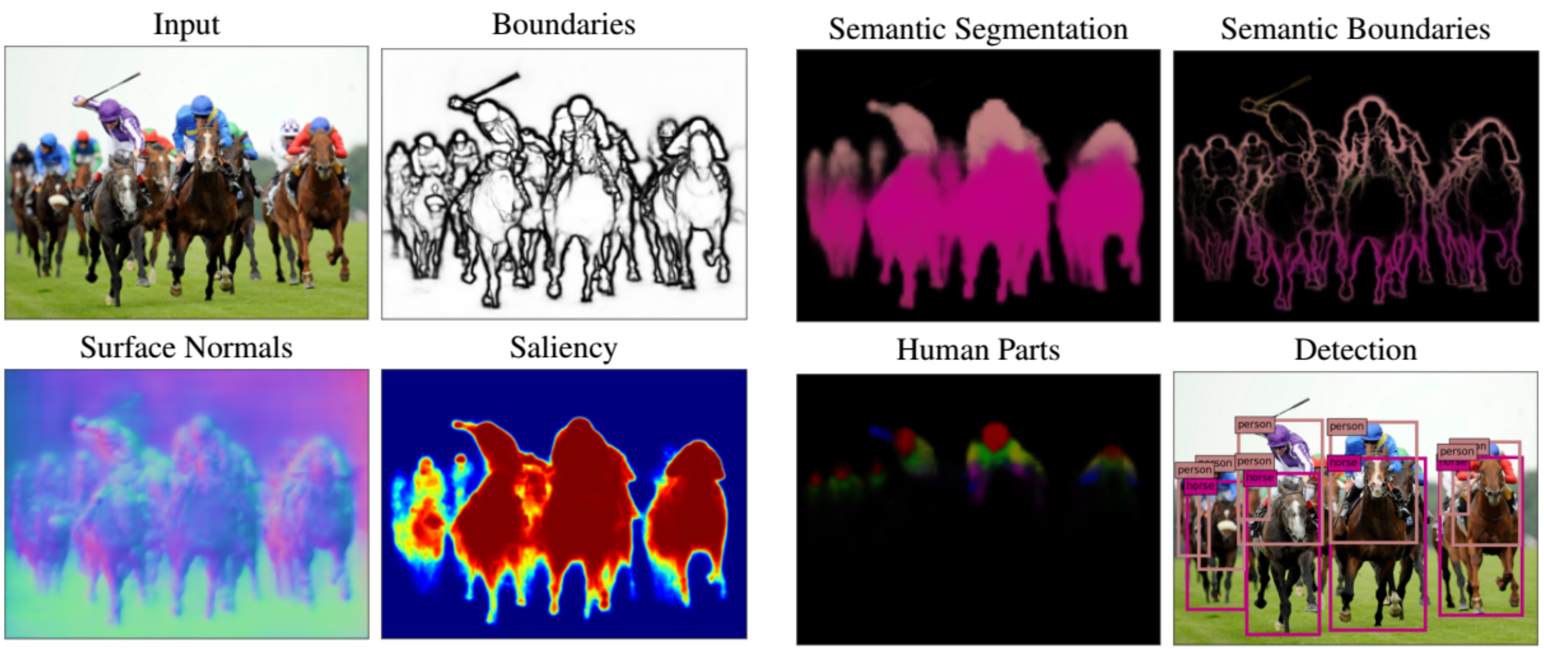
One task might help learn another better

One task might have more annotations

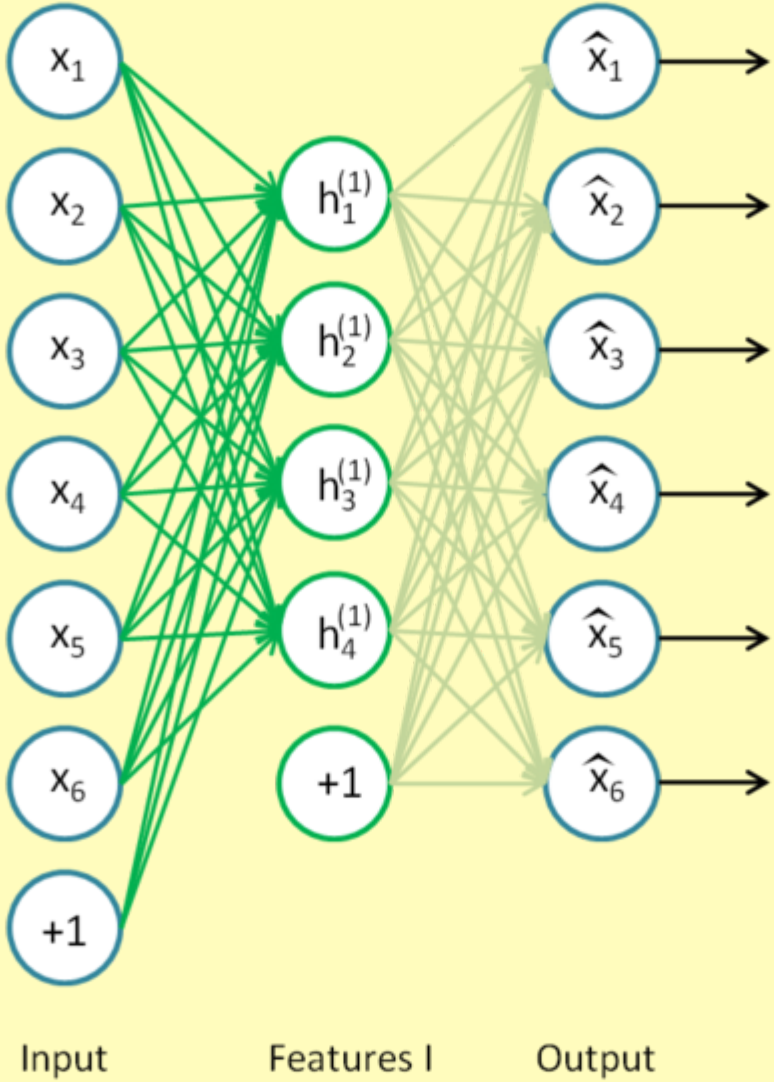
In real applications we don't want 7 VGGnets

- 1 for boundaries, 1 for normals, 1 for saliency, …

One image → Several tasks



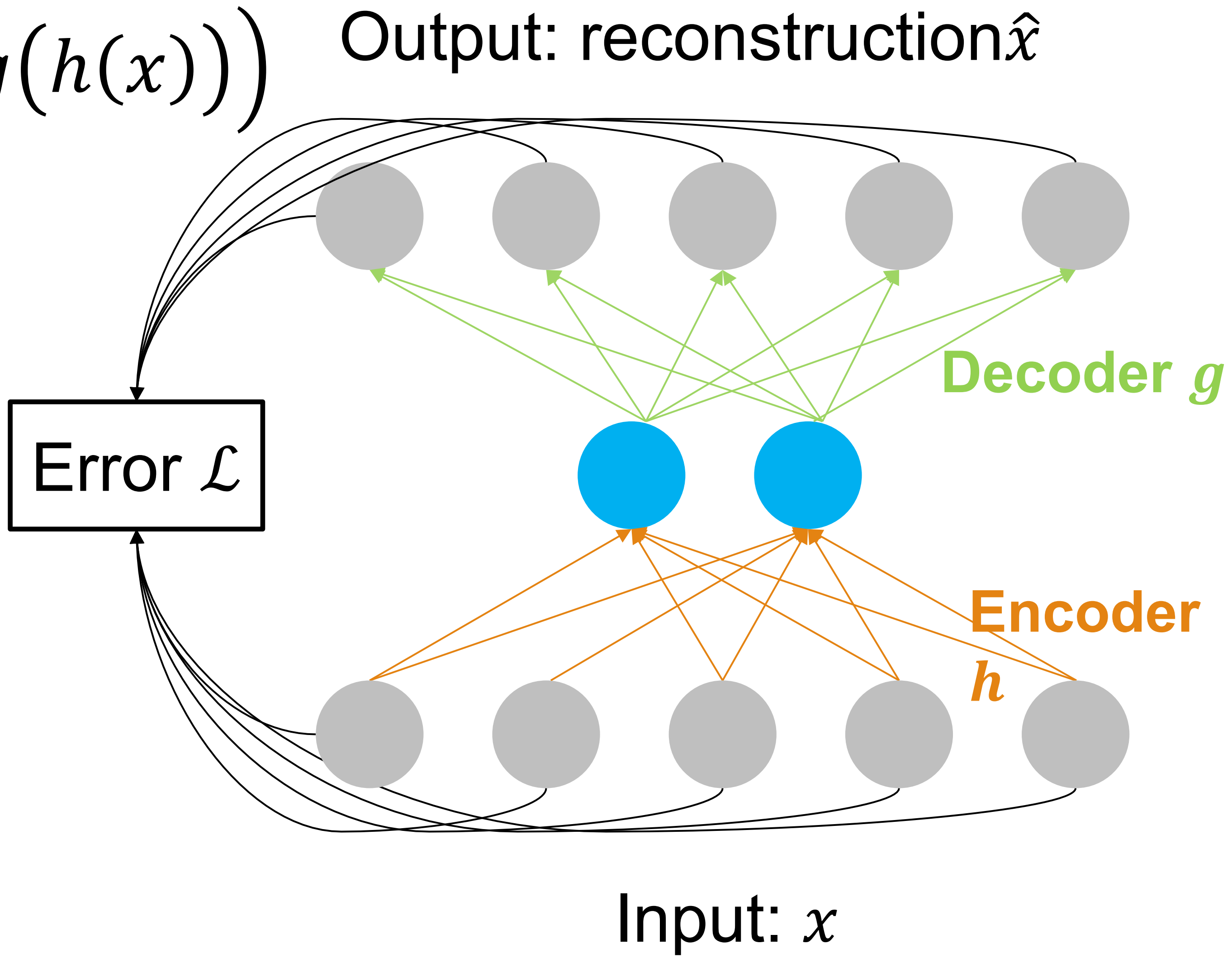
Discovering structure



Standard Autoencoder

$$\mathcal{L} = \sum_d \ell\Big(x, g\big(h(x)\big)\Big)$$

$$\ell = |x - \hat{x}|^2$$



Standard Autoencoder

The latent space should have fewer dimensions than input

- **Undercomplete** representation
- Bottleneck architecture

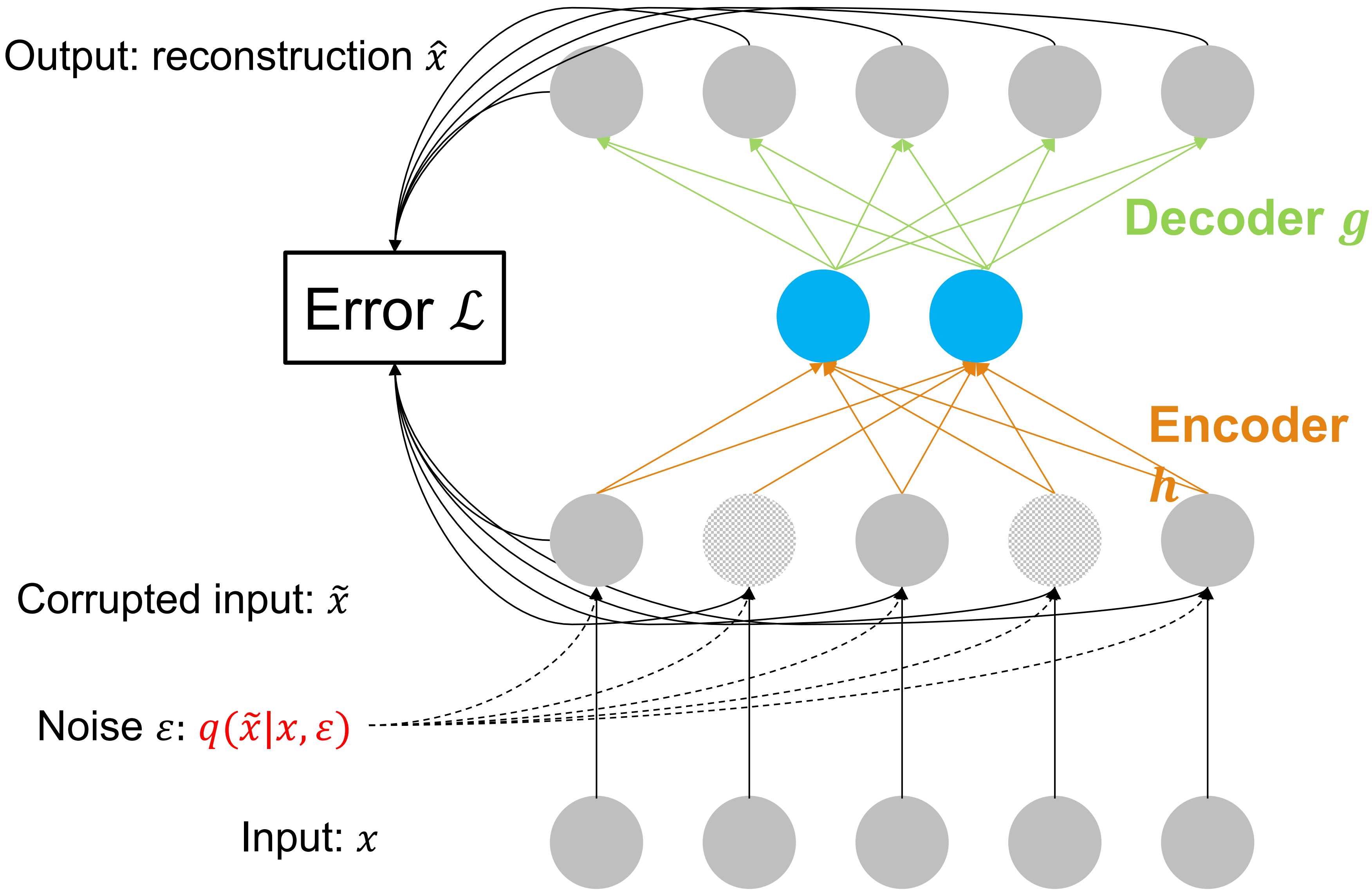
Otherwise (overcomplete) autoencoder might learn the identity function

$$W \propto I \implies \tilde{x} = x \implies \mathcal{L} = 0$$

- Assuming no regularization
- Often in practice still works though

Also, if $z = Wx + b$ (linear) autoencoder learns same subspace as PCA

Denoising Autoencoder



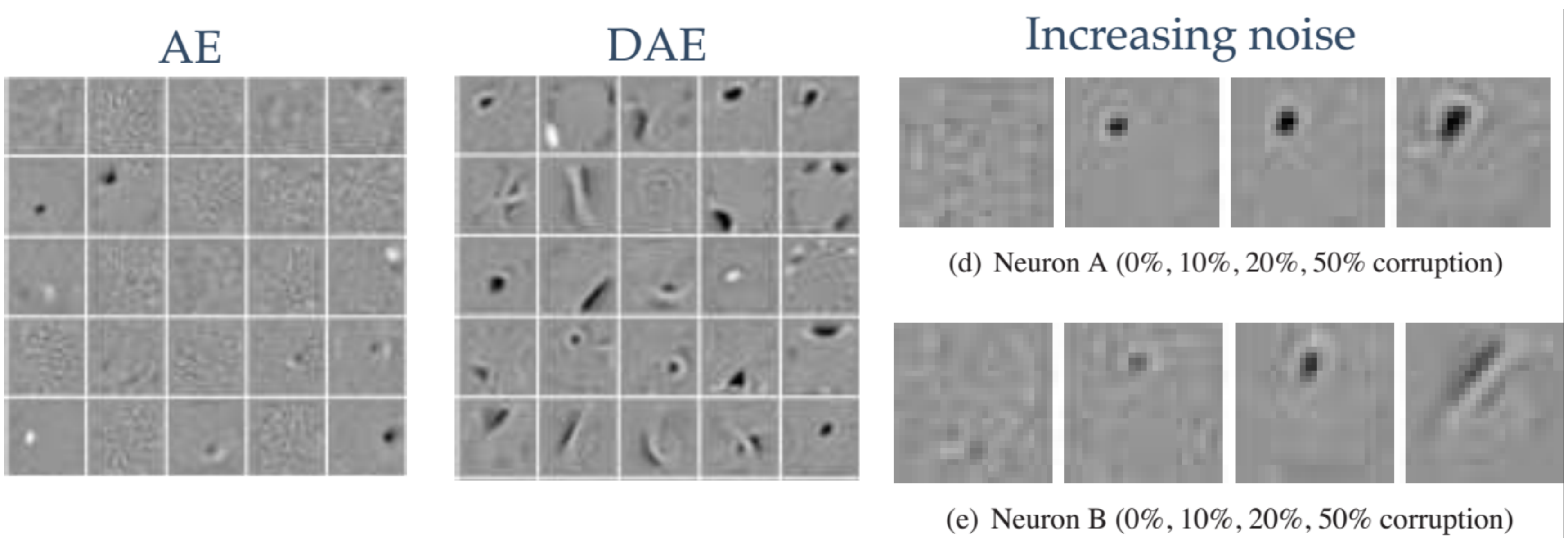
Denoising Autoencoder

The network does not overlearn the data

- ❑ Can even use overcomplete latent spaces

Model forced to learn more intelligent, robust representations

- ❑ Learn to ignore noise or trivial solutions(identity)
- ❑ Focus on “underlying” data generation process



(d) Neuron A (0%, 10%, 20%, 50% corruption)

(e) Neuron B (0%, 10%, 20%, 50% corruption)

Variational Autoencoder

We want to model the data distribution

$$p(x) = \int p_\theta(z) p_\theta(x|z) dz$$

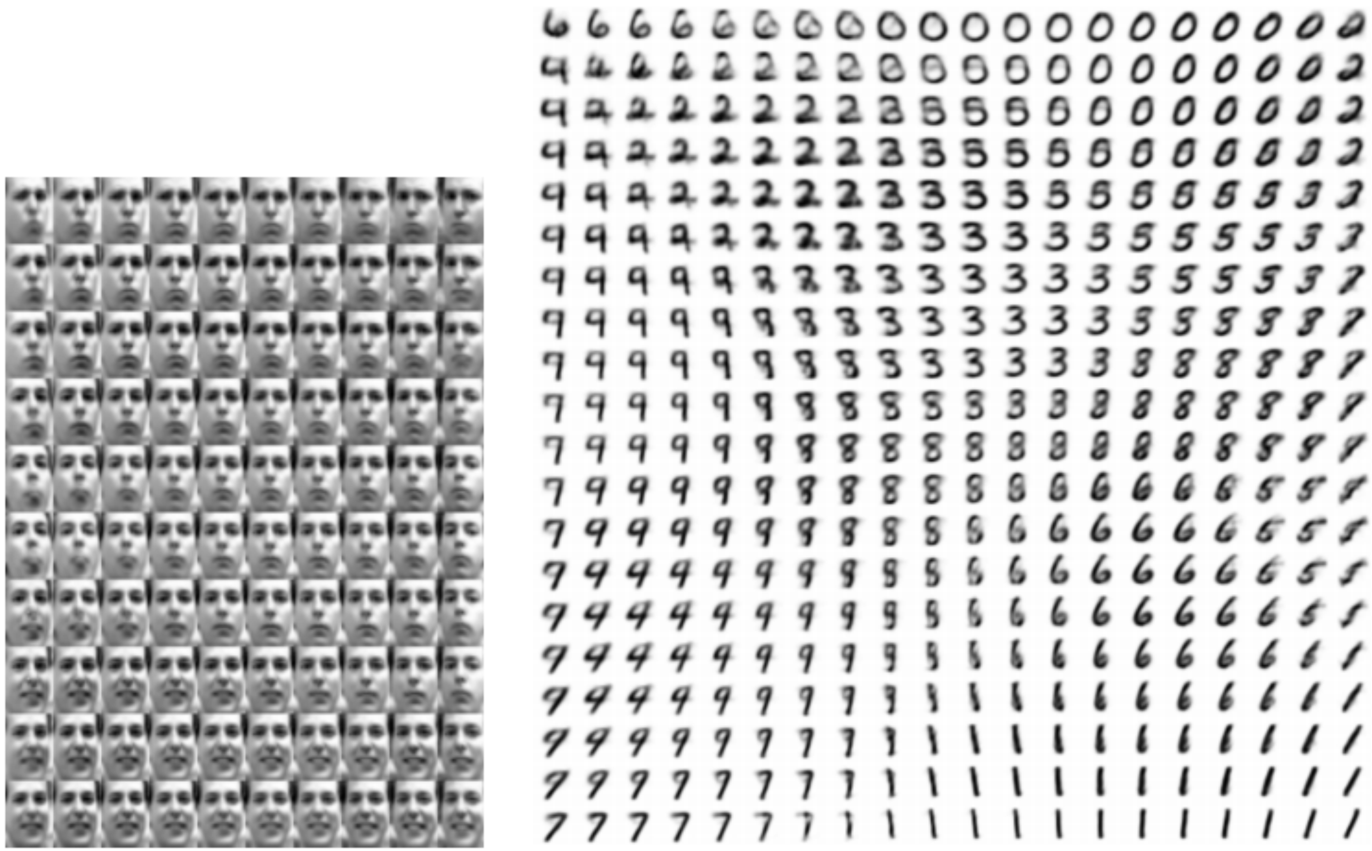
Posterior $p_\theta(z|x)$ is intractable for complicated likelihood functions $p_\theta(x|z)$, e.g. a neural network → $p(x)$ is also intractable

Introduce an inference machine $q_\varphi(z|x)$ (e.g. another neural network) that **learns to approximate** the posterior $p_\theta(z|x)$

- Since we cannot know $p_\theta(z|x)$ define a variational lower bound to optimize instead

$$\mathcal{L}(\theta, \varphi, x) = \underbrace{-D_{KL}\big(q_\varphi(z|x) \| p_\theta(z)\big)}_{\text{Regularization term}} + \underbrace{E_{q_\varphi(z|x)}\big(\log p_\theta(x|z)\big)}_{\text{Reconstruction term}}$$

Examples



(a) Learned Frey Face manifold (b) Learned MNIST manifold

Figure 4: Visualisations of learned data manifold for generative models with two-dimensional latent space, learned with AEVB. Since the prior of the latent space is Gaussian, linearly spaced coordinates on the unit square were transformed through the inverse CDF of the Gaussian to produce values of the latent variables $\mathbf{z}$. For each of these values $\mathbf{z}$, we plotted the corresponding generative $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ with the learned parameters $\boldsymbol{\theta}$.

Generative Adversarial Networks

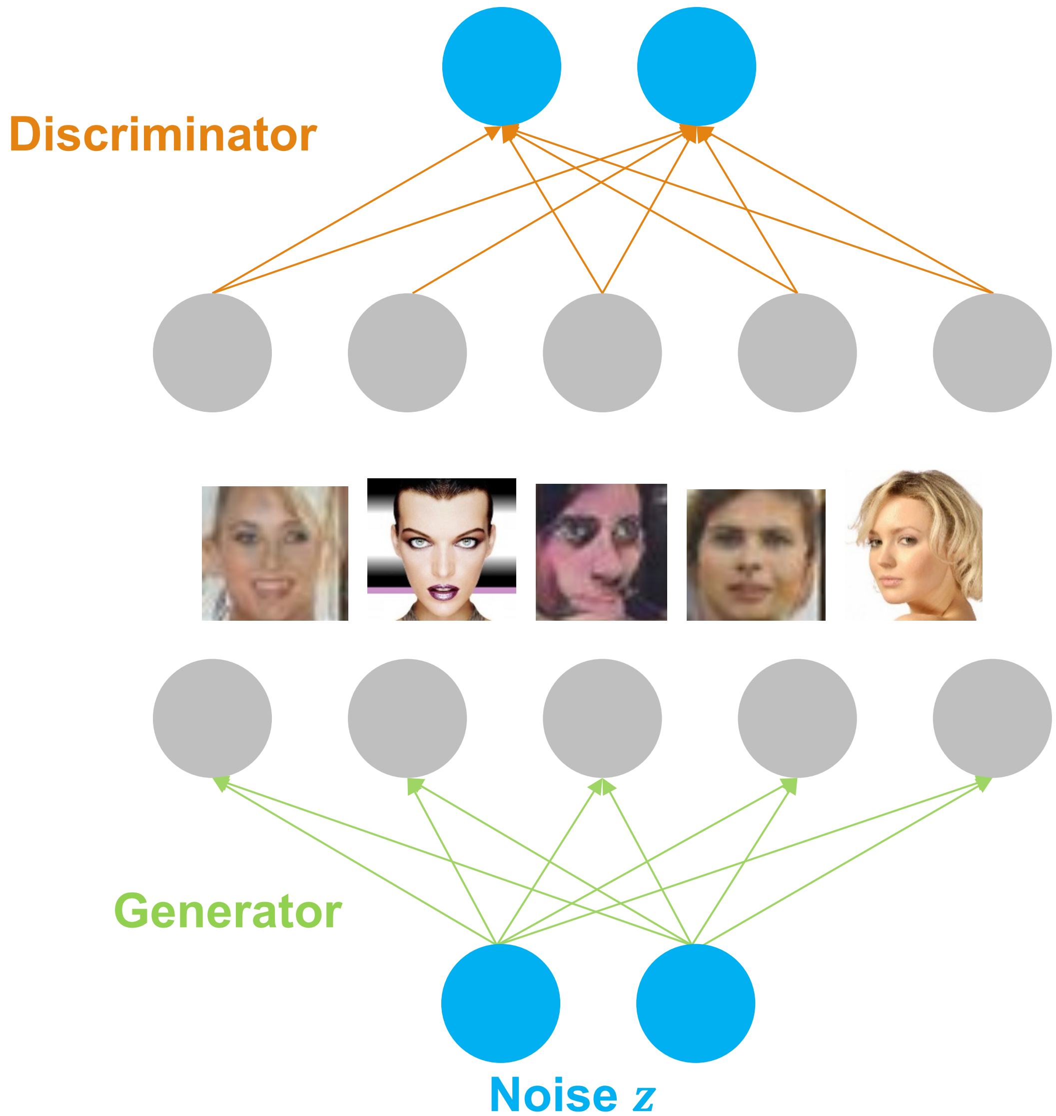
Composed of two successive networks

- Generator network (like upper half of autoencoders)
- Discriminator network (like a convent)

Learning

- Sample “noise” vectors z
- Per z the generator produces a sample x
- Make a batch where half samples are real, half are the generated ones
- The discriminator needs to predict what is real and what is fake

Generative Adversarial Networks



“Police vs Thief”

Generator and discriminator networks optimized together

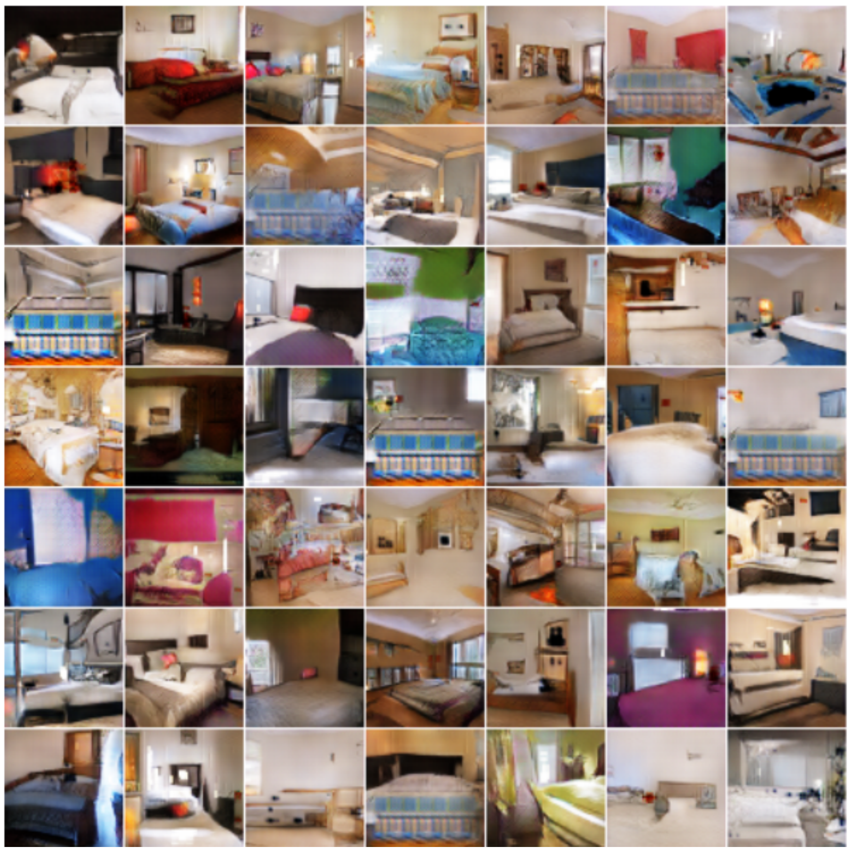
- ❑ The generator (thief) tries to fool the discriminator
- ❑ The discriminator (police) tries to not get fooled by the generator

Mathematically

$$\min_G \max_D V(G,D) = E_{x\sim p_{data}(x)} \log D(x) + E_{z\sim p_z(z)} \log(1 - D\big(G(z)\big)$$

Examples

Bedrooms



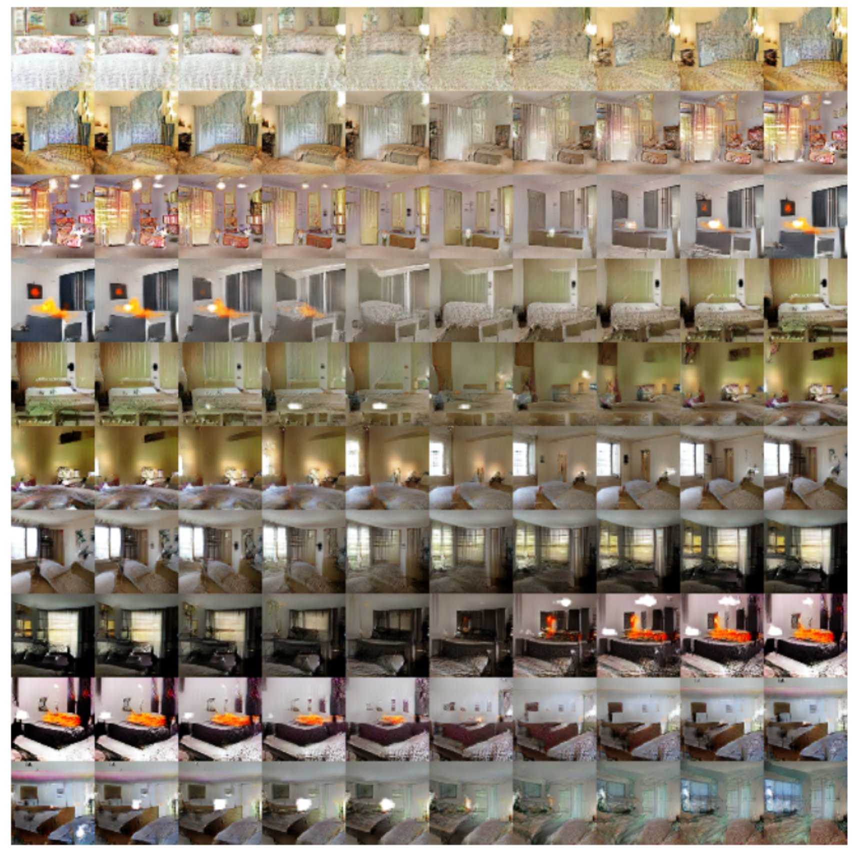
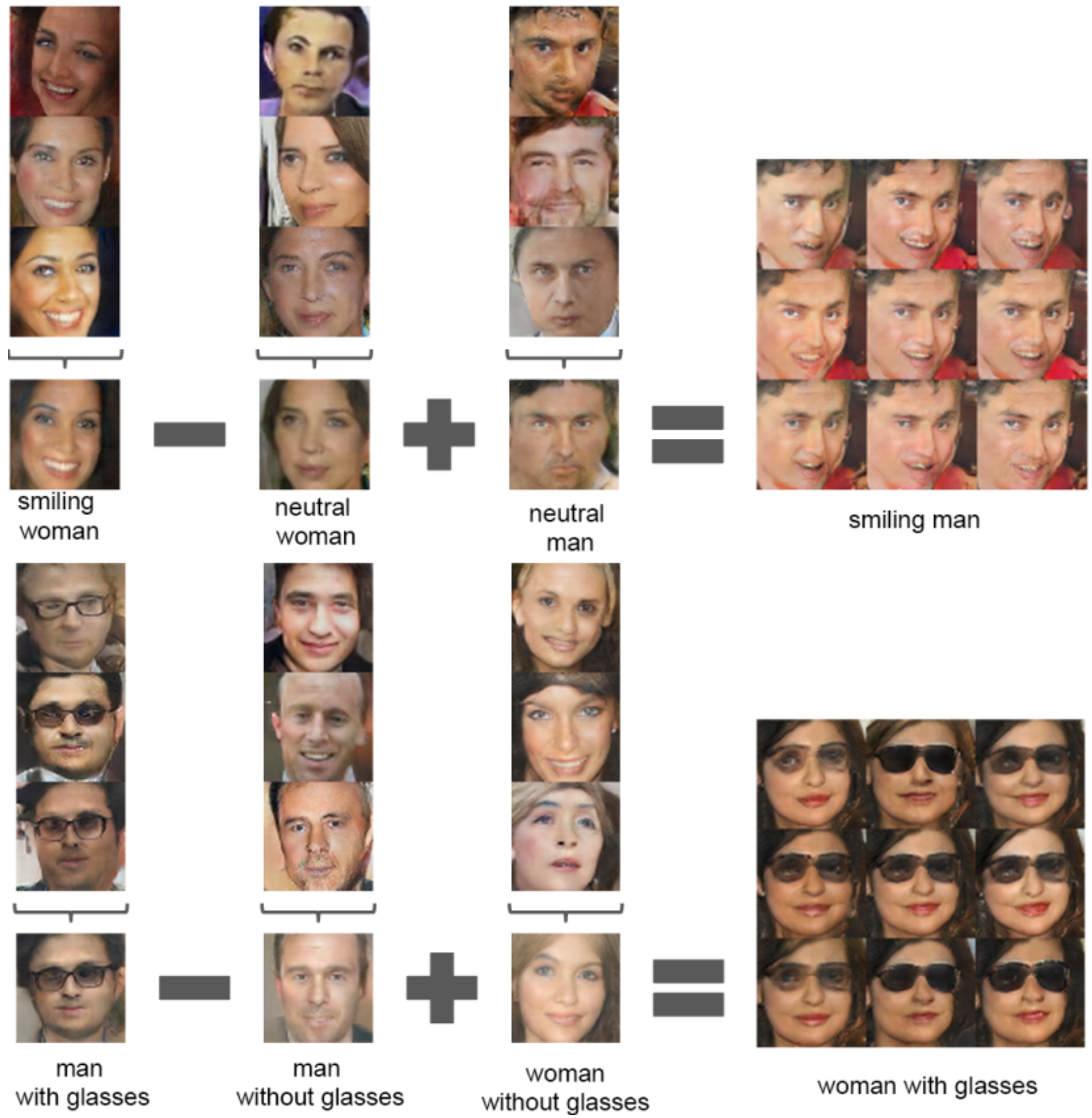


Image “arithmetics”



Take away message

Deep Learning is good not only for classifying things

Structured prediction is also possible

Multi-task structure prediction allows for unified networks

Discovering structure in data is also possible

Thank you!