THE PRICE TO PAY?

1. LARGE NUMBER OF PARAMETERS IMPLIES LARGE DATASETS TO TRAIN

2. LOOSE EVEN MORE DEGREE OF CONTROL OF WHAT THE ALGORITHM IS DOING SINCE THE FEATURE EXTRACTION PROCESS BECOMES UNSUPERVISED

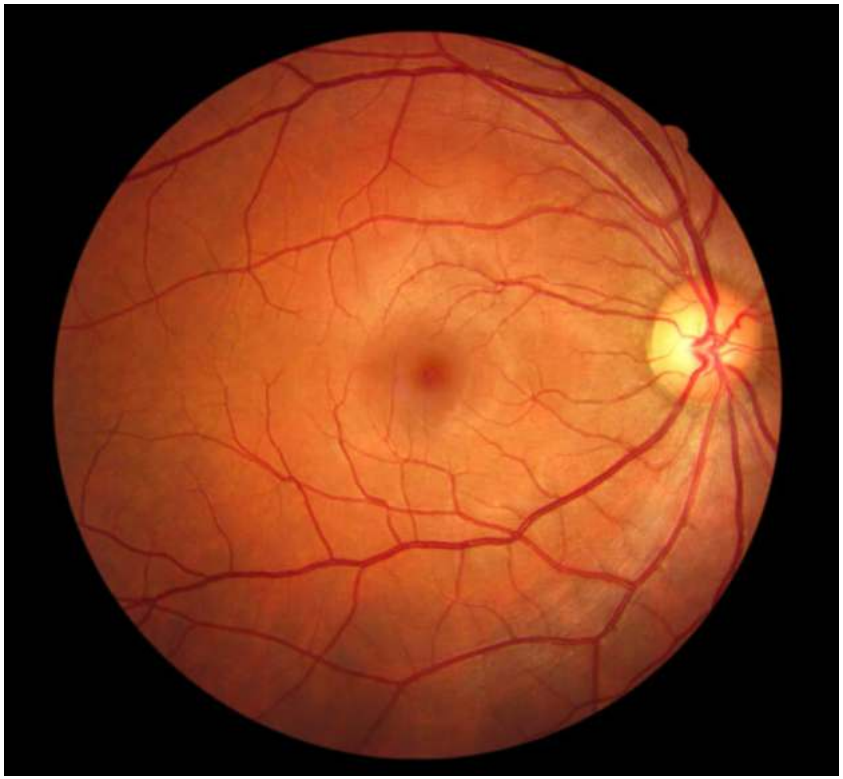


IMAGE OF THE BACK OF THE EYE

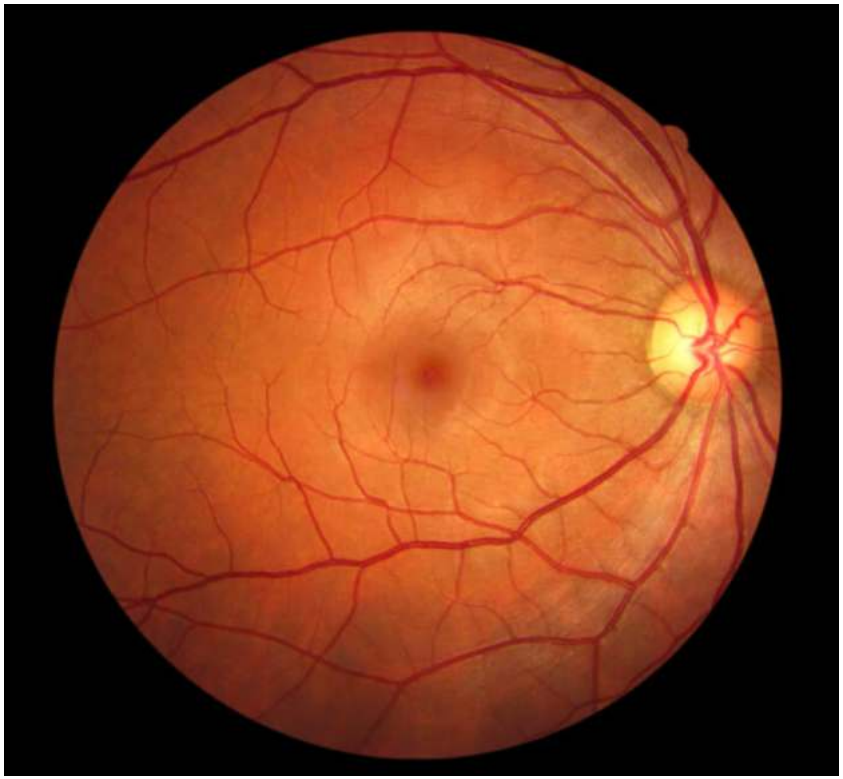


IMAGE OF THE BACK OF THE EYE

DEEP LEARNING CAN IDENTIFY THE PATIENT'S GENDER WITH 95% ACCURACY

VISUALIZING CNNs
[intepreting CNN decisions]

Attribution techniques

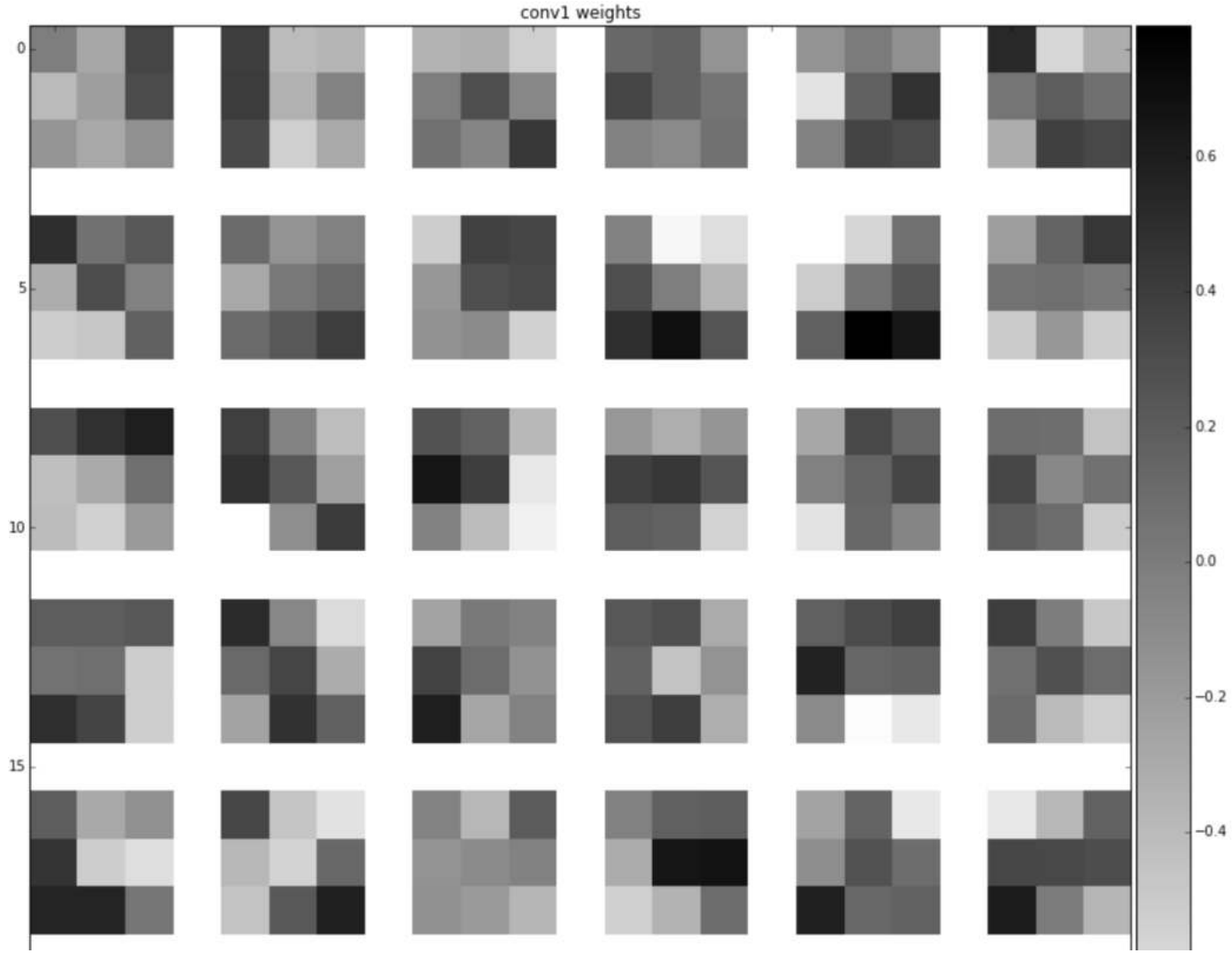
DEEP NETWORKS ARE “BLACK BOXES”?

INTERPRETING THE RESULTS IS
EXTREMELY DIFFICULT

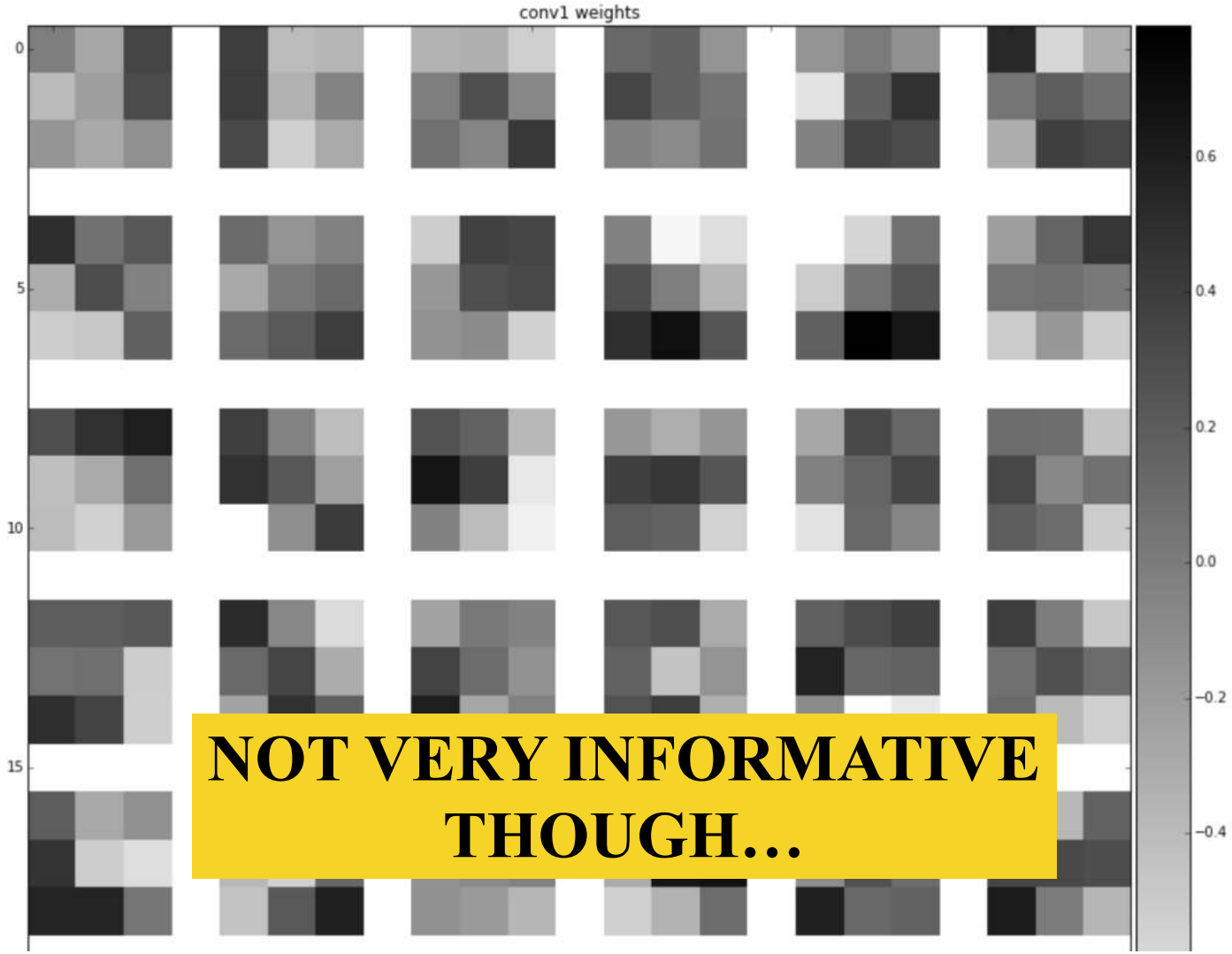
THIS IS TRUE BUT A LOT OF WORK
IS DONE TO UNVEIL THEIR BEHAVIOR

EXPLORING THE FEATURE MAPS

THE SIMPLEST APPROACH IS TO VISUALIZE THE LEARNED WEIGHTS AT INTERMEDIATE LAYERS

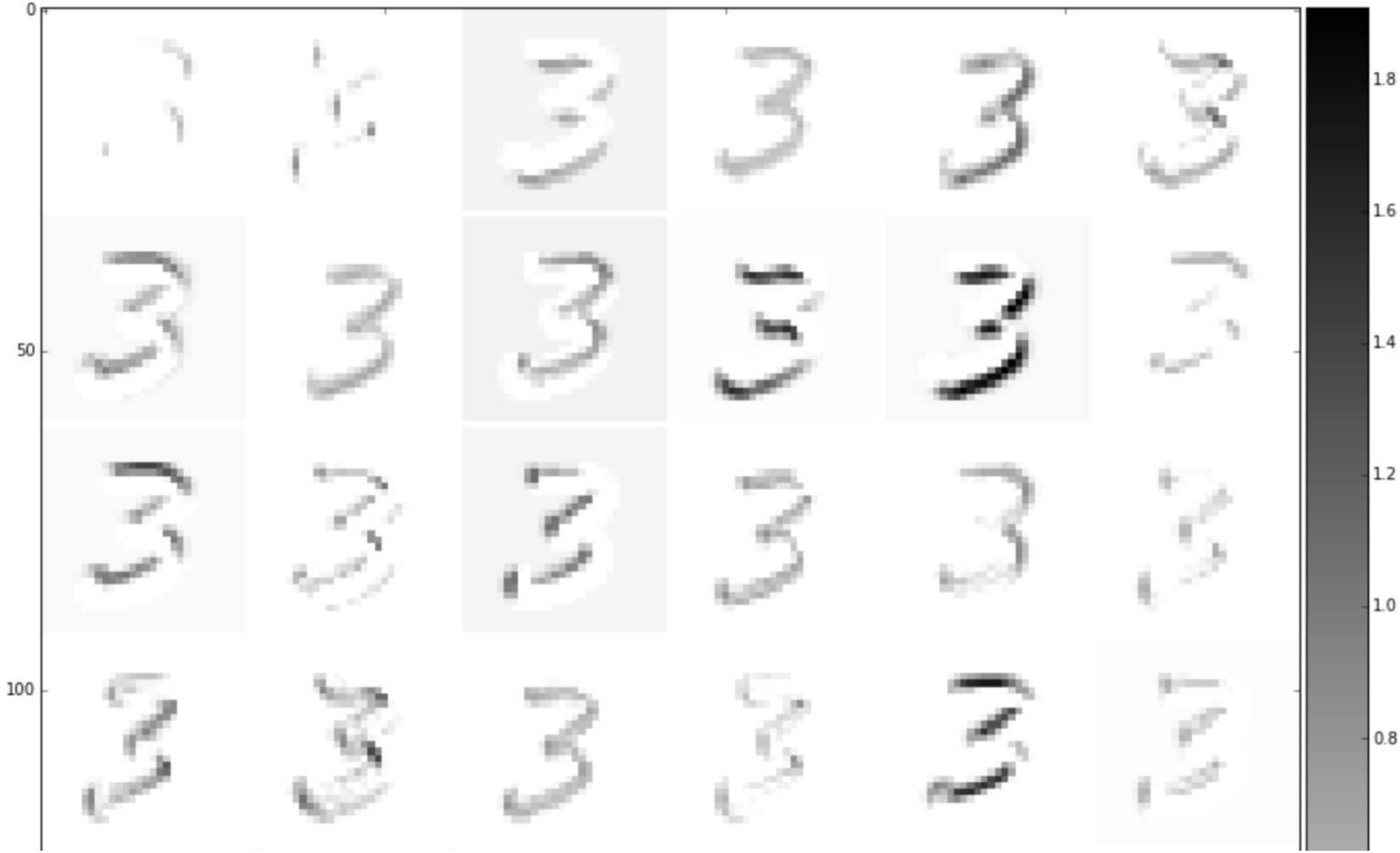


THE SIMPLEST APPROACH IS TO VISUALIZE THE LEARNED WEIGHTS AT INTERMEDIATE LAYERS

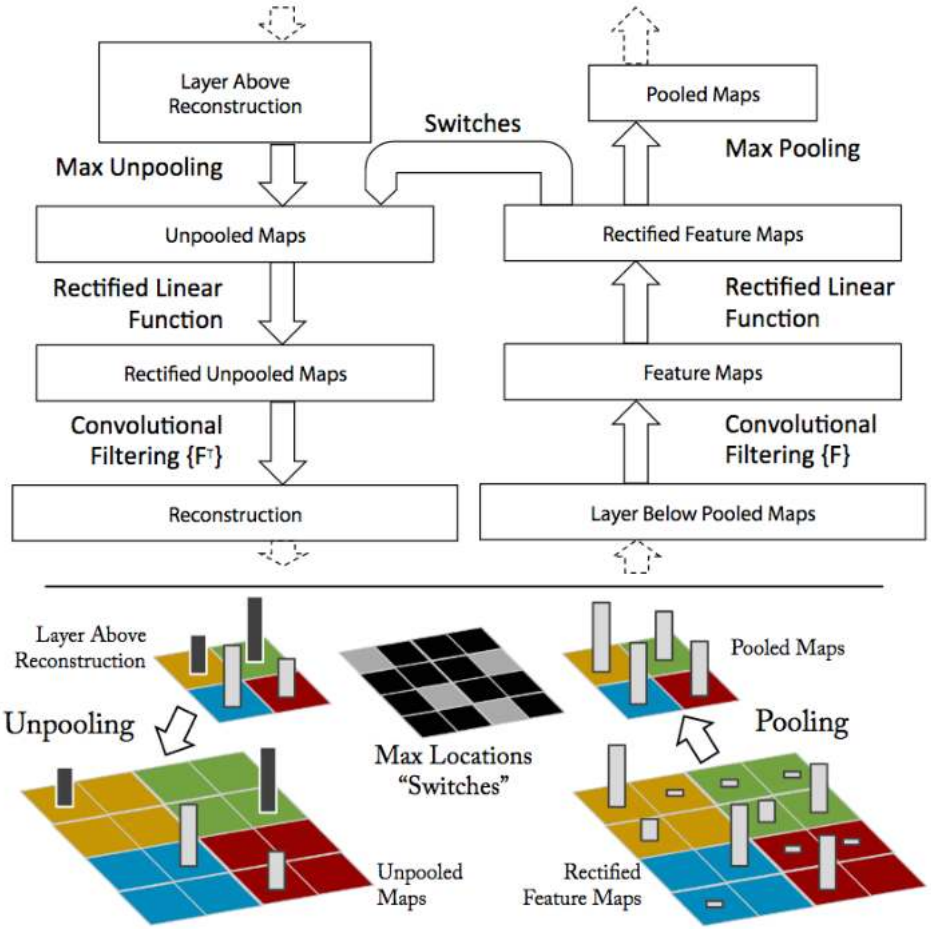


USING THE SAME IDEA, ONE CAN ALSO VISUALIZE THE FEATURE MAPS AT INTERMEDIATE LAYERS

THIS HELPS TRACING THE FEATURES LEARNED BY THE NETWORK

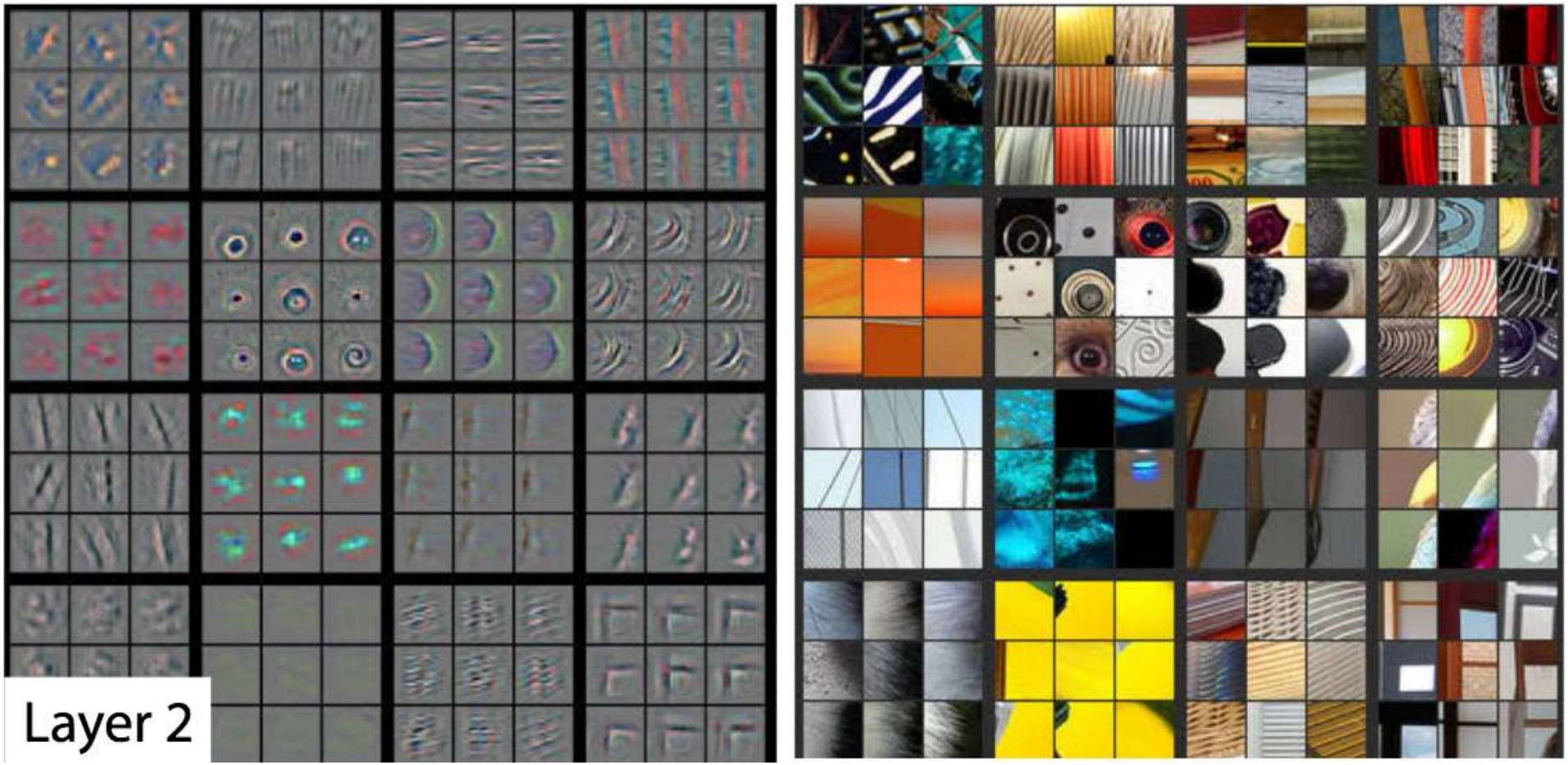

0
50
100
1.8
1.6
1.4
1.2
1.0
0.8

USE “DECONVNETS” TO MAP BACK THE FEATURE MAP INTO THE PIXEL SPACE



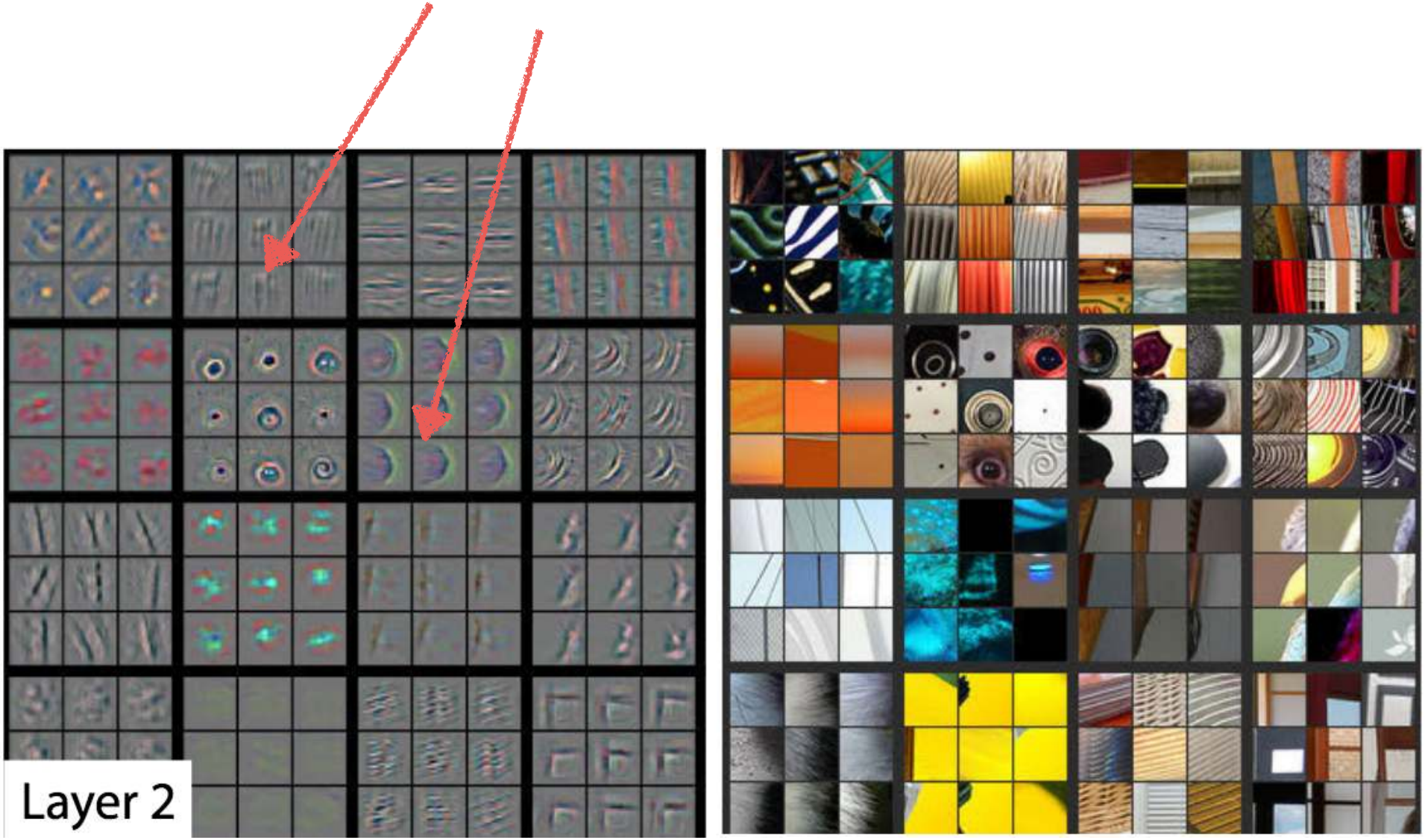
IT ALLOWS TO SEE
WHICH
REGIONS OF THE INPUT
GENERATED
A MAXIMUM RESPONSE
IN A NEURON

Zeiler+14



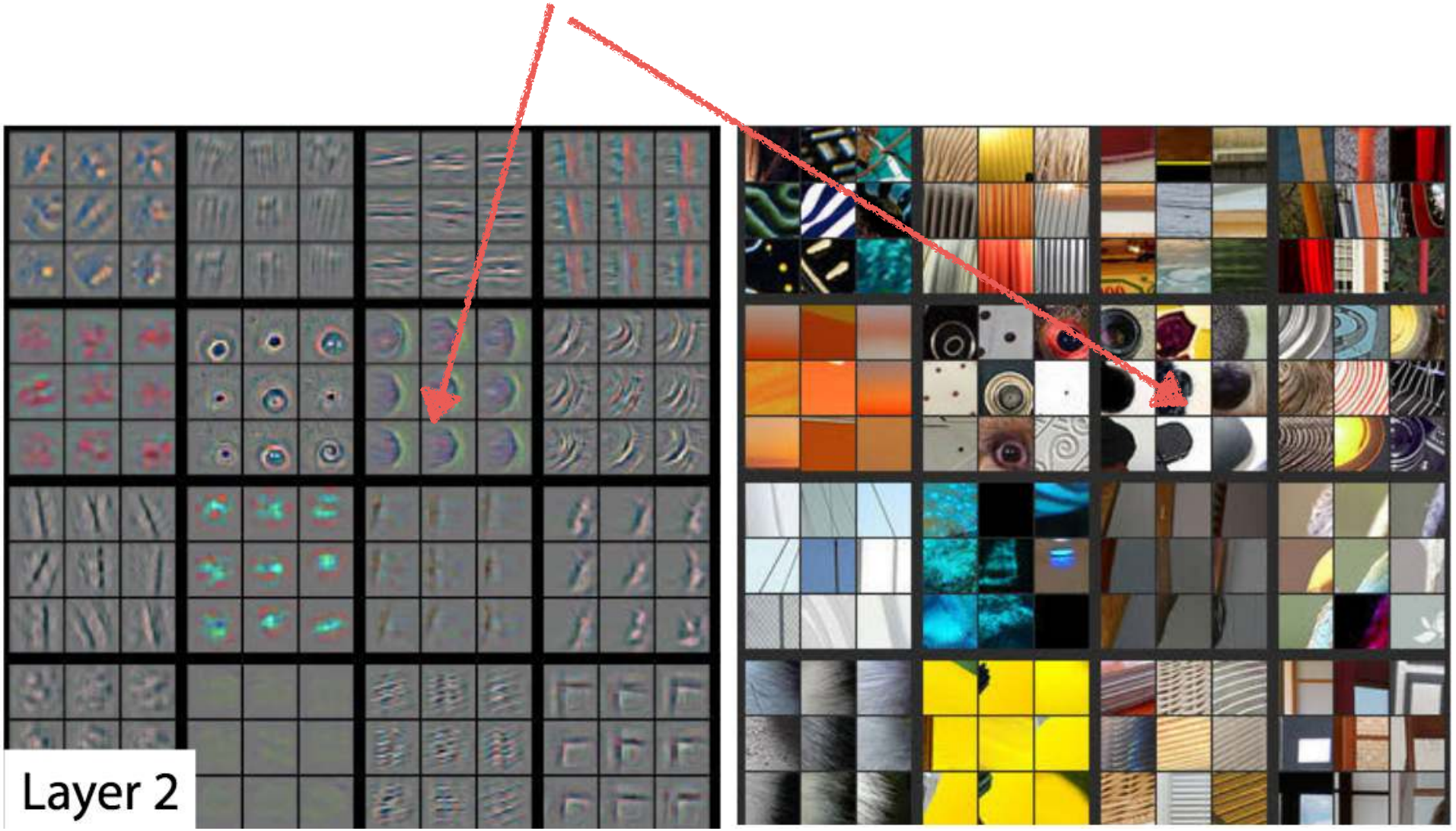
Zeiler+14

EVERY BLOCK OF 9 SHOWS
THE 9 STRONGEST RESPONSES TO A GIVEN FILTER OF LAYER2



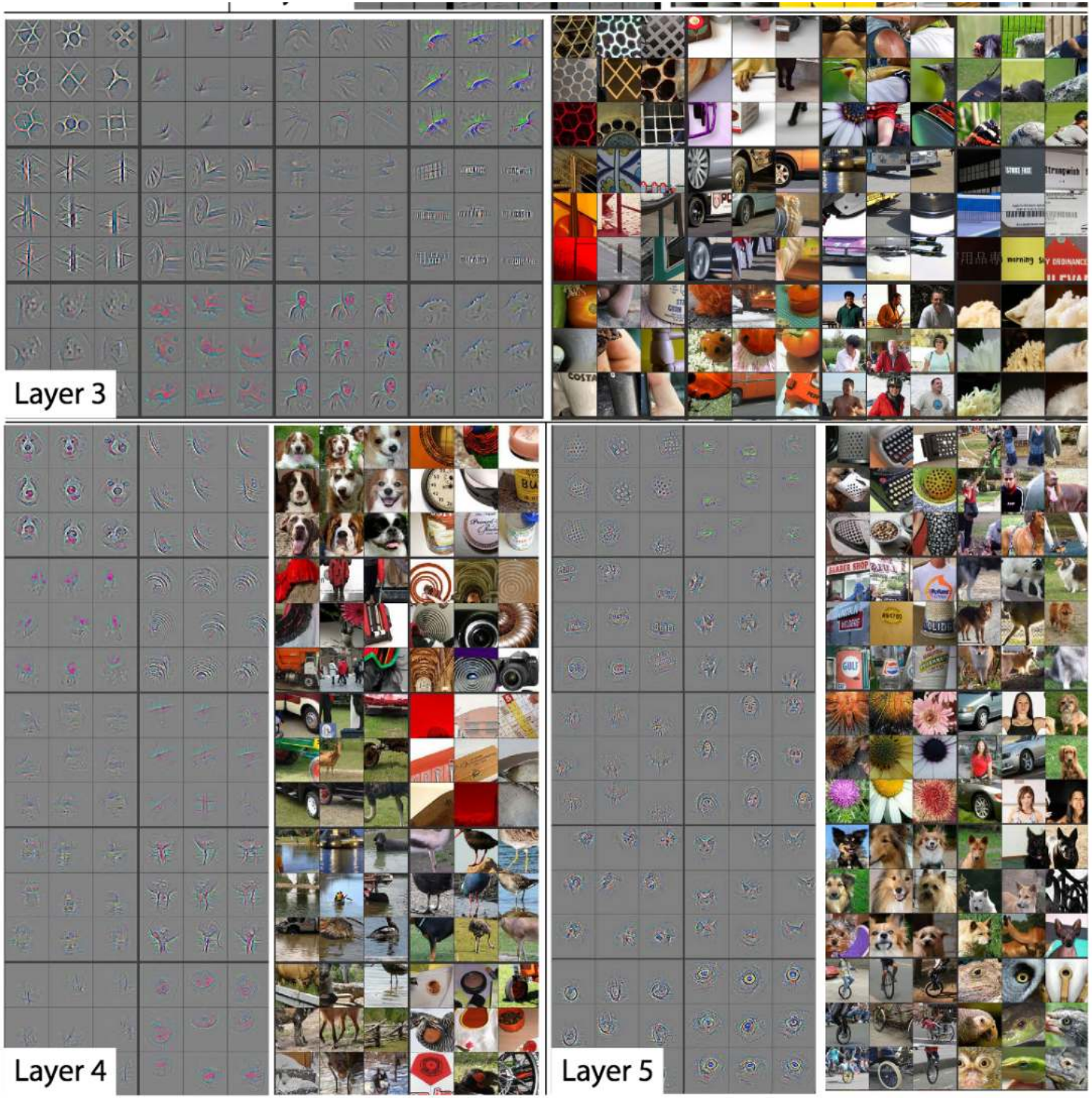
Zeiler+14

THE CORRESPONDING REGIONS OF IMAGES THAT GENERATED THE MAXIMUM RESPONSE



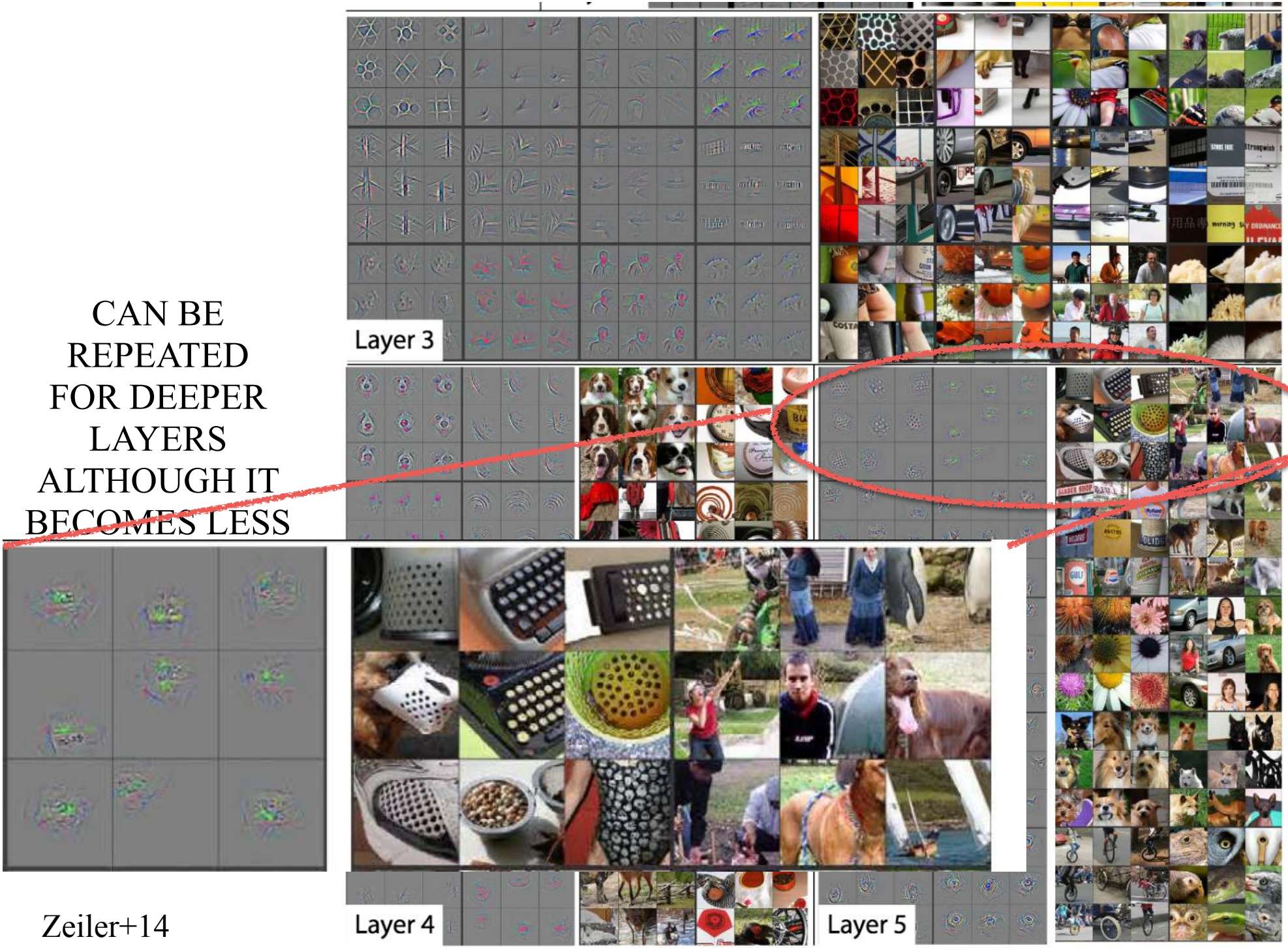
Zeiler+14

CAN BE
REPEATED
FOR DEEPER
LAYERS
ALTHOUGH IT
BECOMES LESS
INTUITIVE



Zeiler+14

CAN BE REPEATED FOR DEEPER LAYERS ALTHOUGH IT BECOMES LESS



Zeiler+14

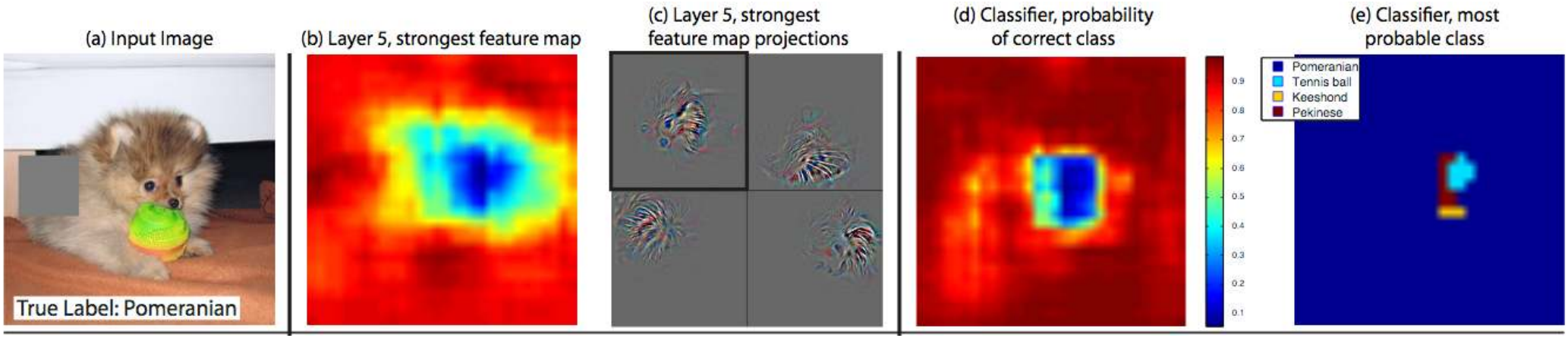
OCCLUSION SENSITIVITY

THE BASIC IDEA IS TO
PERTURB / MODIFY AN INPUT
IMAGE AND SEE THE EFFECT ON
THE PREDICTIONS

OCCLUSION SENSITIVITY TRIES ALSO TO FIND THE REGION OF THE IMAGE THAT TRIGGERED THE NETWORK DECISION BY MASKING DIFFERENT REGIONS OF THE INPUT IMAGE AND ANALYZING THE NETWORK OUPUT

IT ALLOWS TO IF THE NETWORK IS TAKING THE DECISIONS BASED ON THE EXPECTED FEATURES

<u>VERY TIME CONSUMING!</u>



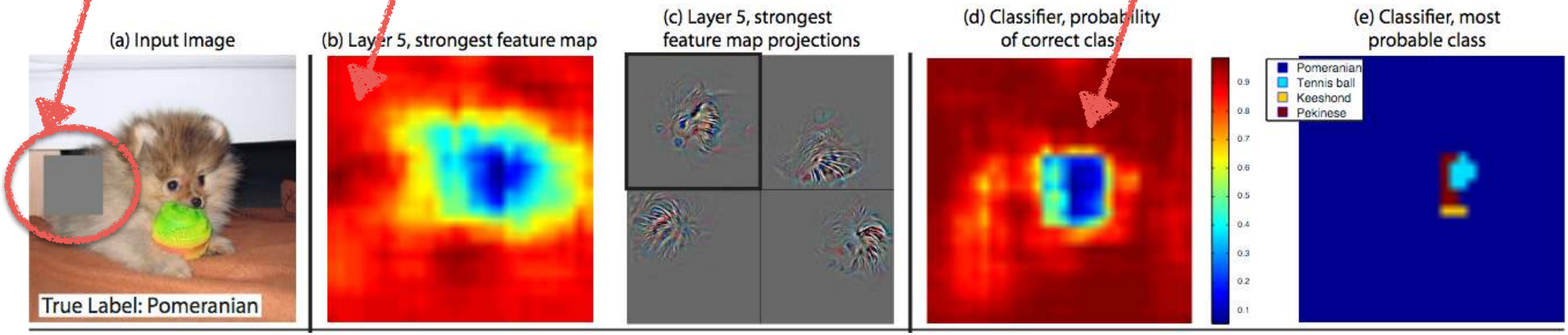
Zeiler+14

OCCLUSION SENSITIVITY TRIES ALSO TO FIND THE REGION OF THE IMAGE THAT TRIGGERED THE NETWORK DECISION BY MASKING DIFFERENT REGIONS OF THE INPUT IMAGE AND ANALYZING THE NETWORK OUPUT

for every position
of the square the maximum response of a given layer
is averaged

occluded region

the output probability as a
function of the occluding square
position



Zeiler+14

“INCEPTIONISM” TECHNIQUES

THE IDEA BEHIND INCEPTIONISM TECHNIQUES
IS TO INVERT THE NETWORK TO GENERATE AN IMAGE
THAT MAXIMIZES THE OUTPUT SCORE

Score of class c for image I

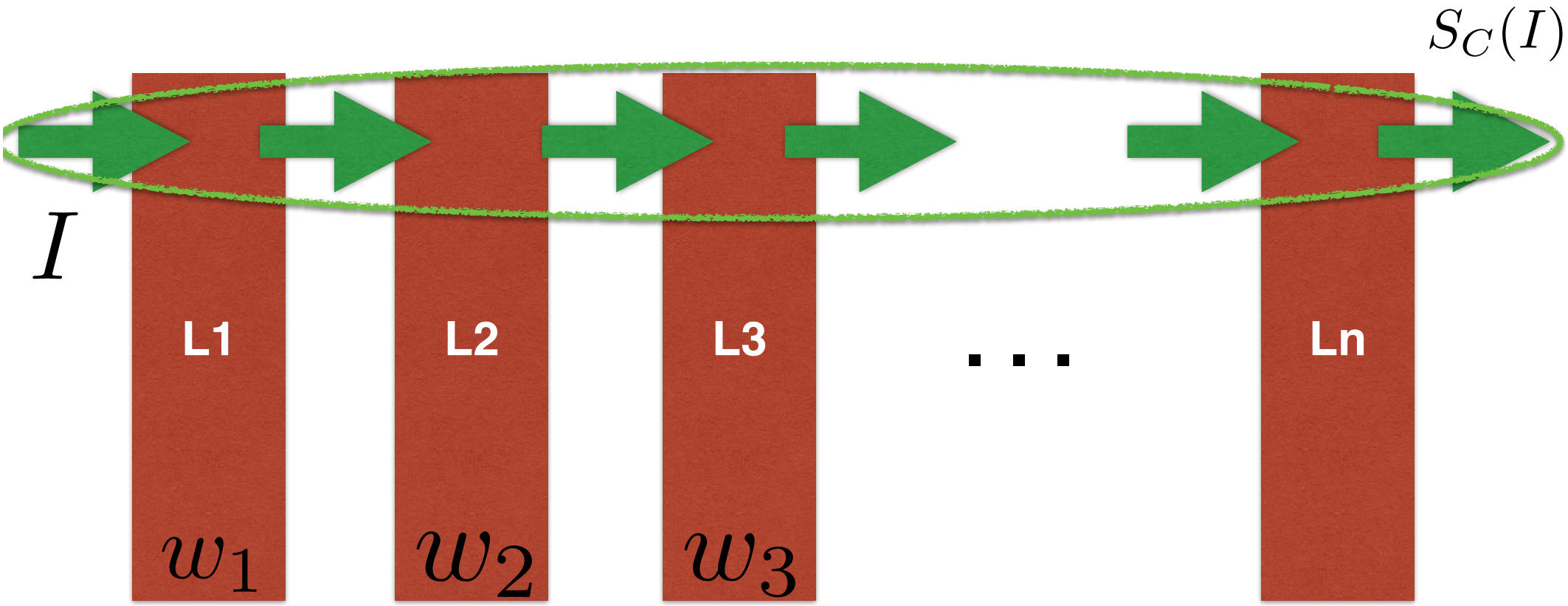
image I

$$arg\,\max_{I} S_c(I) - \lambda||I||_2^2$$

TRY TO FIND AN IMAGE THAT GENERATES A
HIGH SCORE FOR A GIVEN CLASS

Simonyan+14

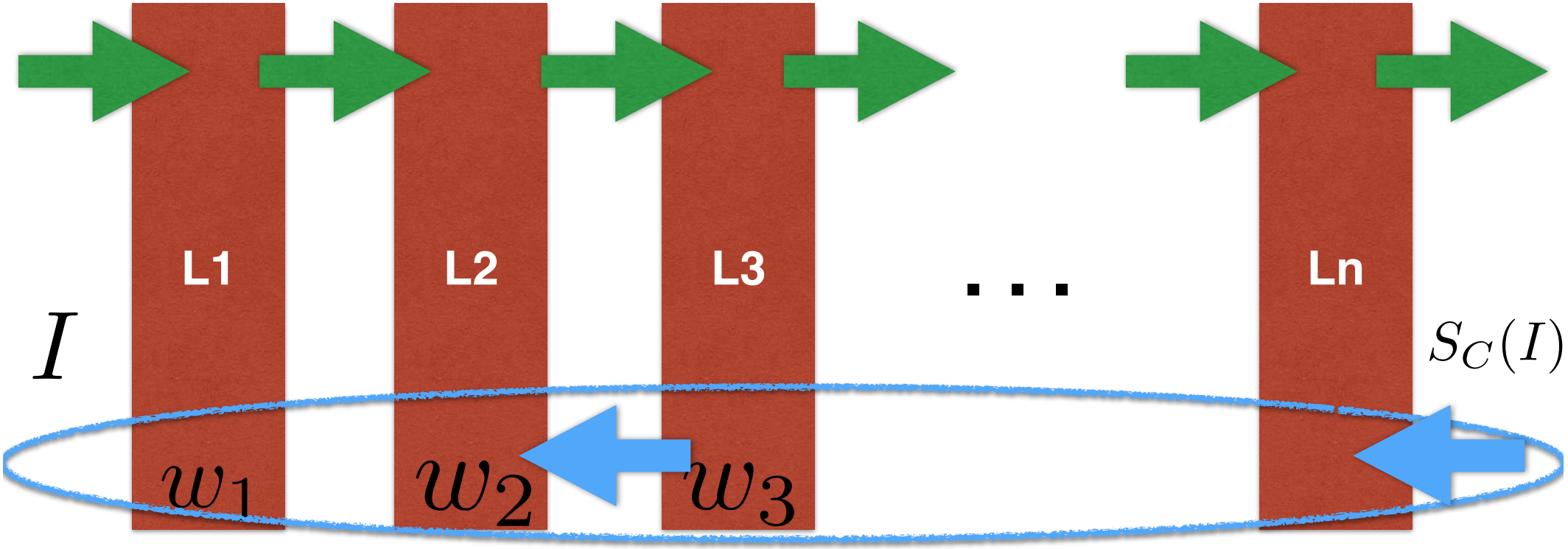
INCEPTIONISM - DEEP DREAM



DURING THE TRAINING PHASE THE WEIGHTS ARE LEARNED TO MAP I INTO Sc

Simonyan+14

INCEPTIONISM - DEEP DREAM



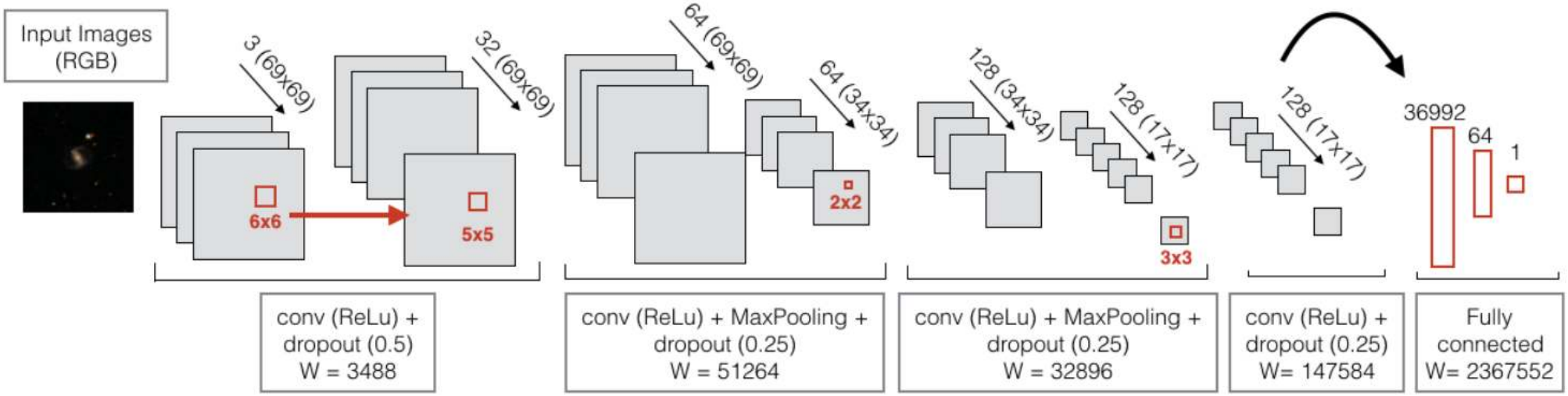
DURING THE RECONSTRUCTION PHASE, I IS LEARNT TRHOUG BACKPROPAGATION KEEPING THE WEIGHTS FIXED

Simonyan+14

SEE SLIDES ON ACTIVATION ATLAS FOR AN EXAMPLE

BEYOND CLASSIFICATION: IMAGE2IMAGE NETWORKS

UP TO NOW CNNs MAP IMAGES (SIGNALS) INTO FLOATS



Dominguez-Sanchez+18

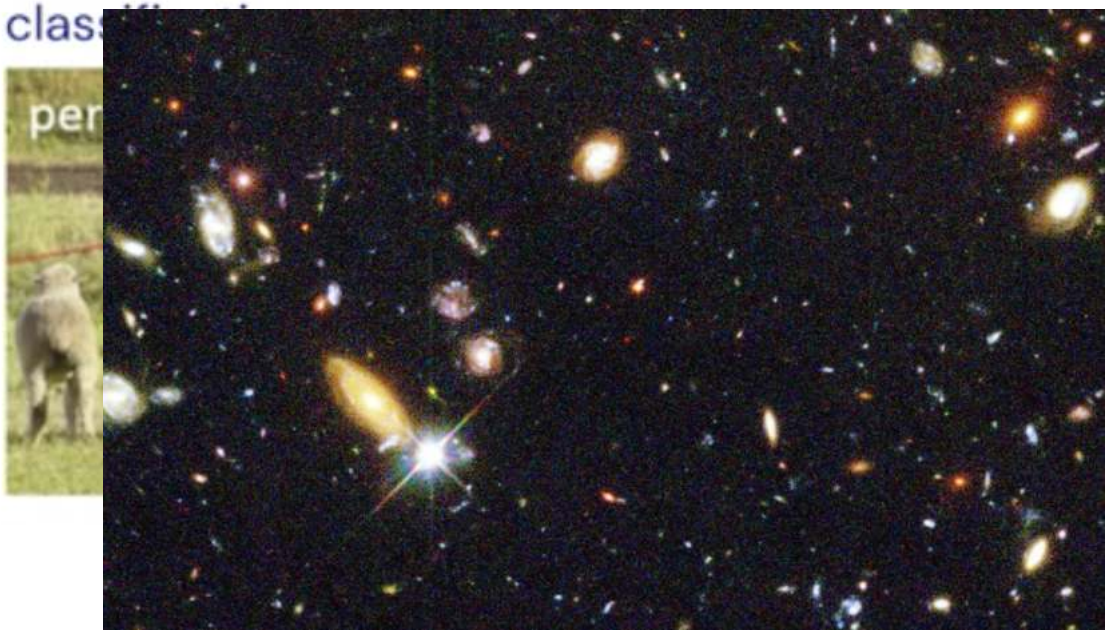
Classification has its limits ….

HOW DO I CLASSIFY THIS IMAGE?

Classification has its limits ….

HOW DO I CLASSIFY THIS IMAGE?

Classification has its limits ….

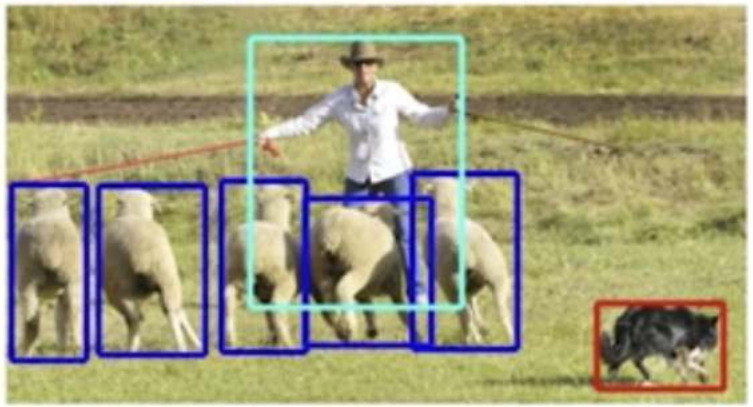


HOW DO I CLASSIFY THIS IMAGE?

Going beyond classification: increasing complexity

classification

object detection



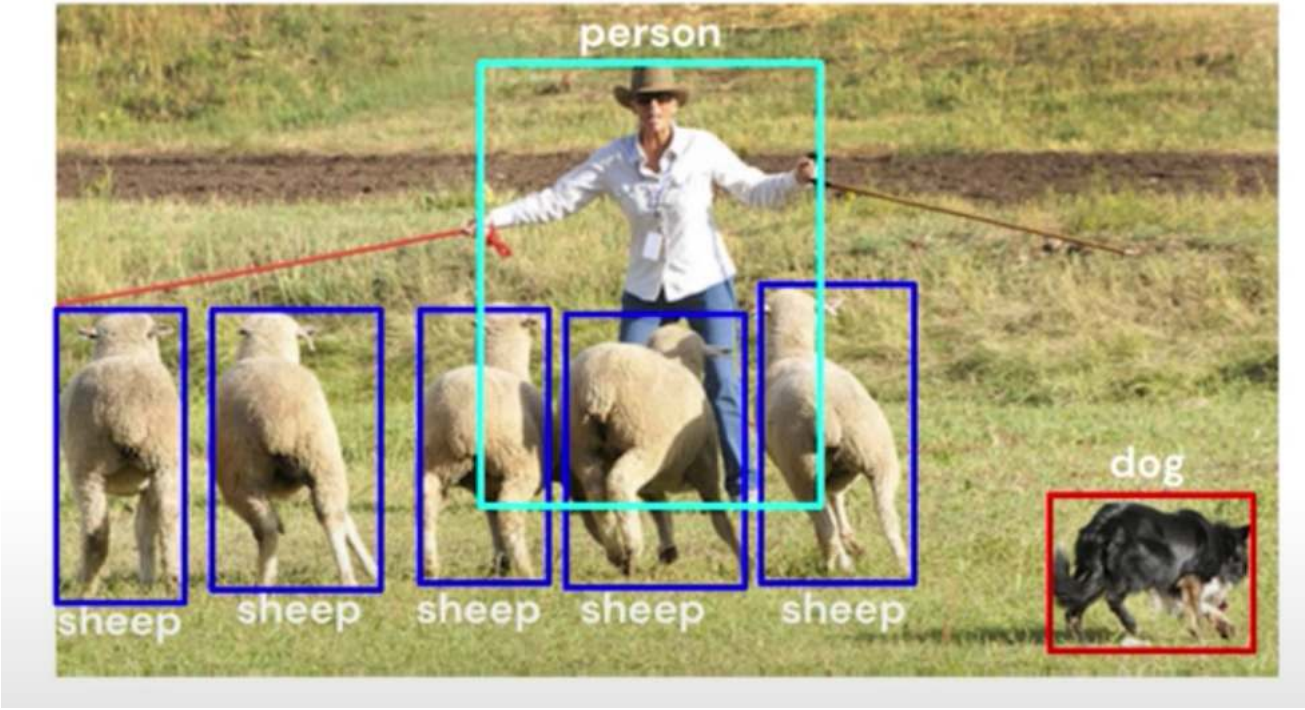
semantic segmentation

instance segmentation

@deepmind (V. Patraucean)

Object detection

First task is to find a bounding box for every object. How we do that?



@deepmind (V. Patraucean)

Inputs

- RGB image $H \times W \times 3$

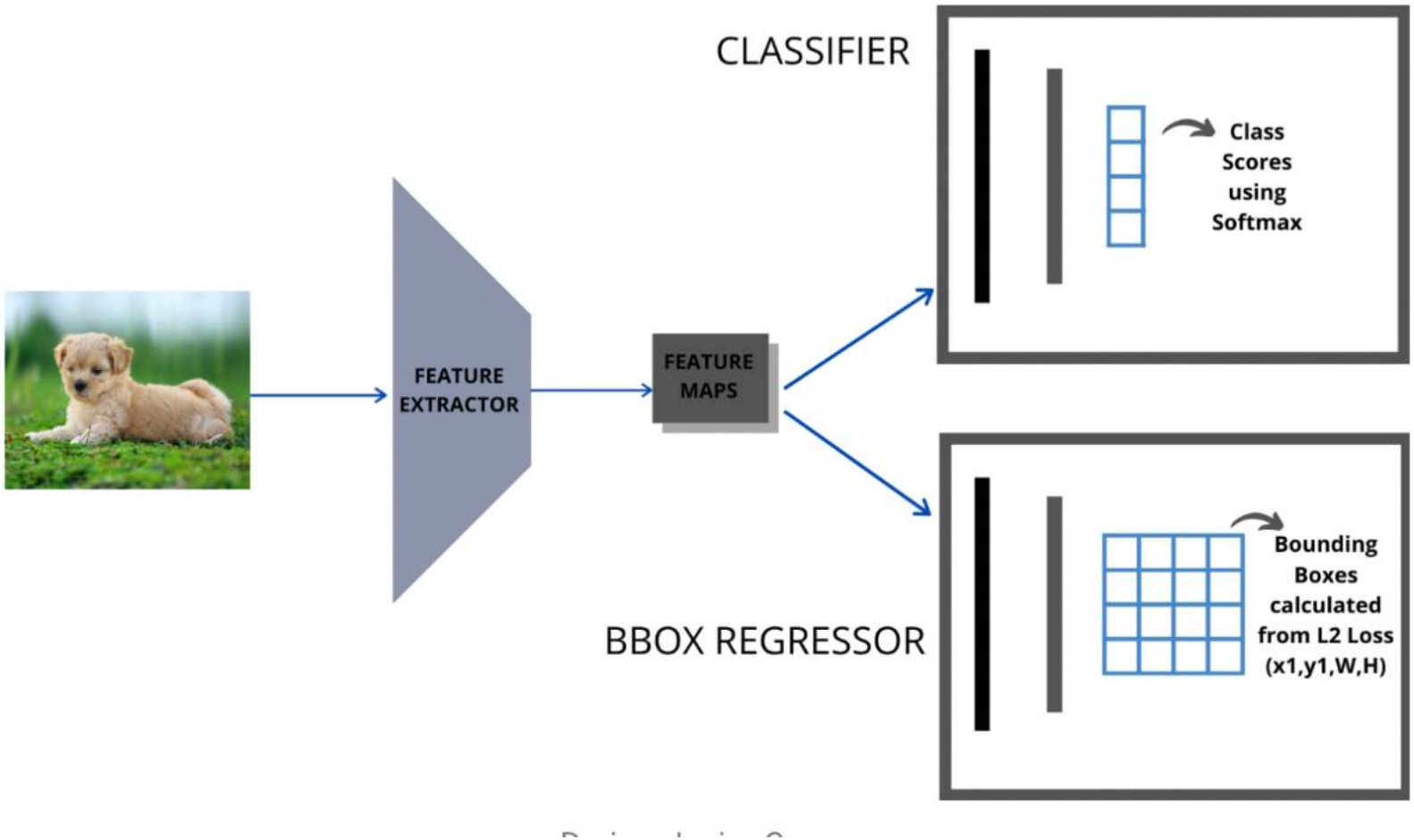
Targets

- Class label *one_hot* 0 0 0 1 0...
- Object bounding box

$$(x_c, y_c, h, w)$$

for all the objects present in the scene

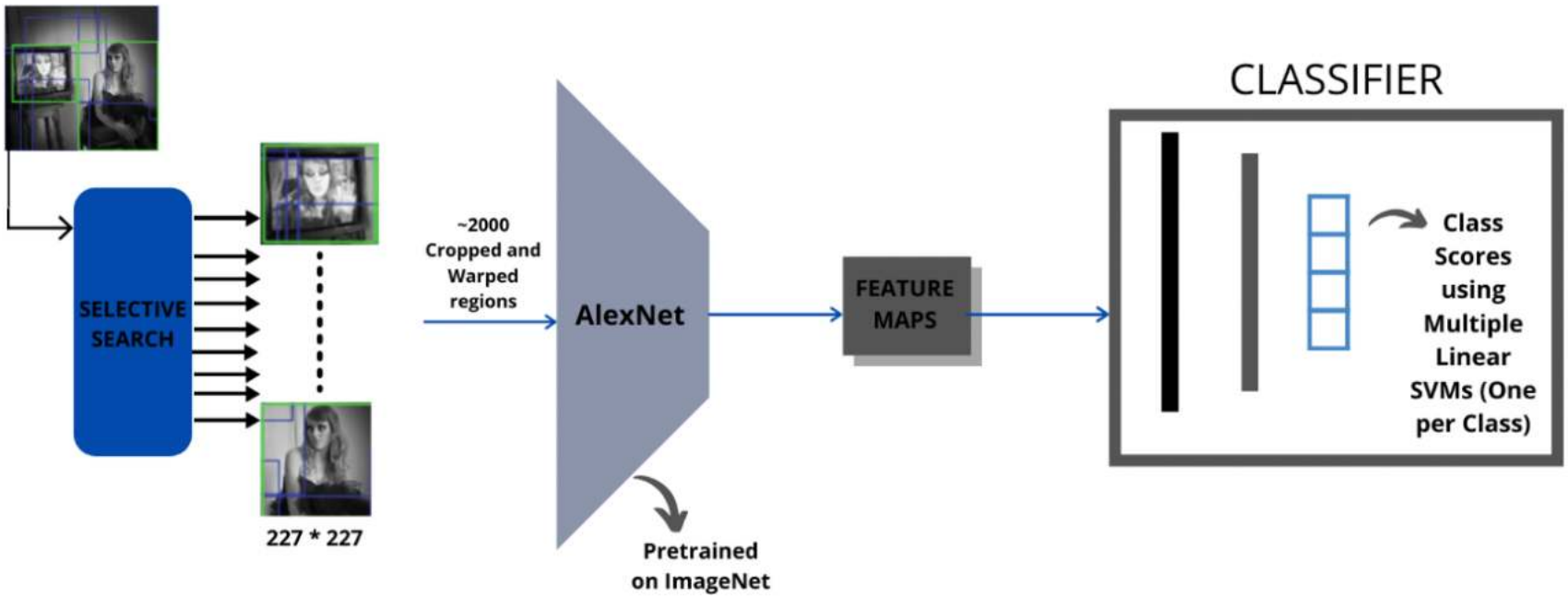
@deepmind (V. Patraucean)


CLASSIFIER
Class Scores using Softmax
FEATURE EXTRACTOR
FEATURE MAPS
BBOX REGRESSOR
Bounding Boxes calculated from L2 Loss (x1,y1,W,H)

The first ideas were based on first using existing methods such as Hierarchical Clustering

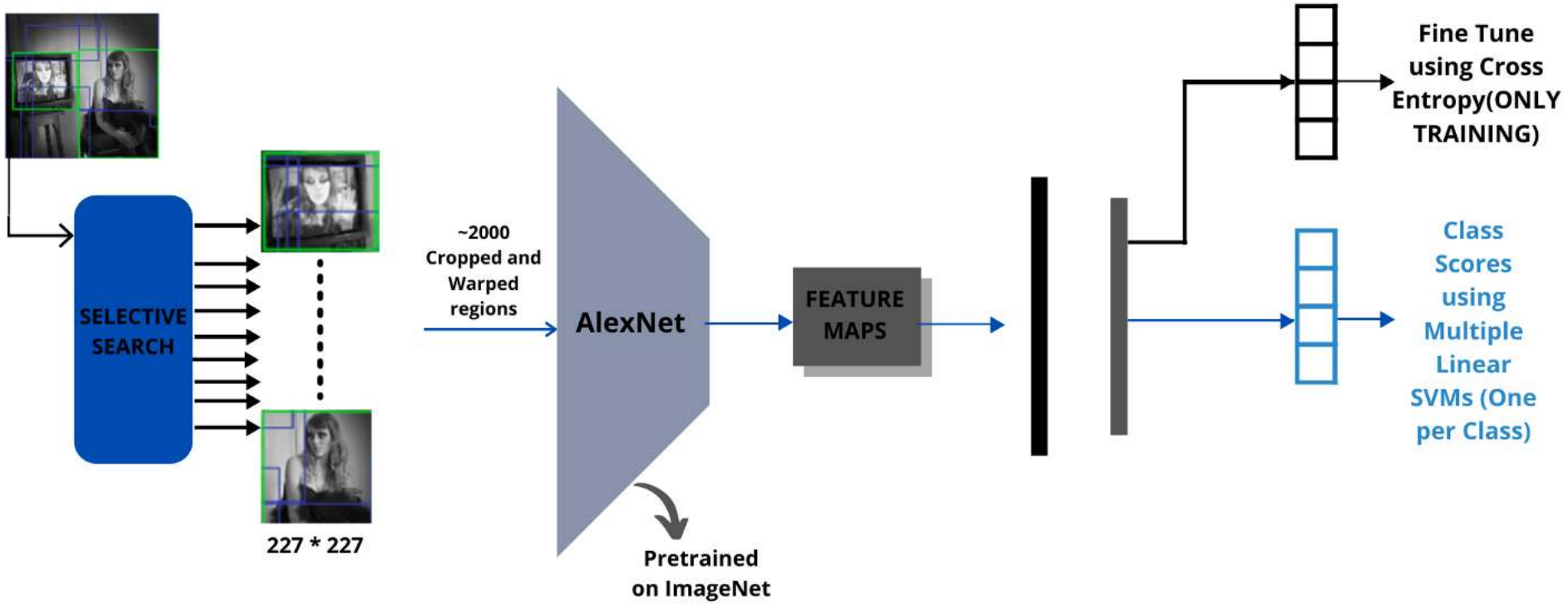
Grouping of pixels based on texture, color, composition …

R-CNN

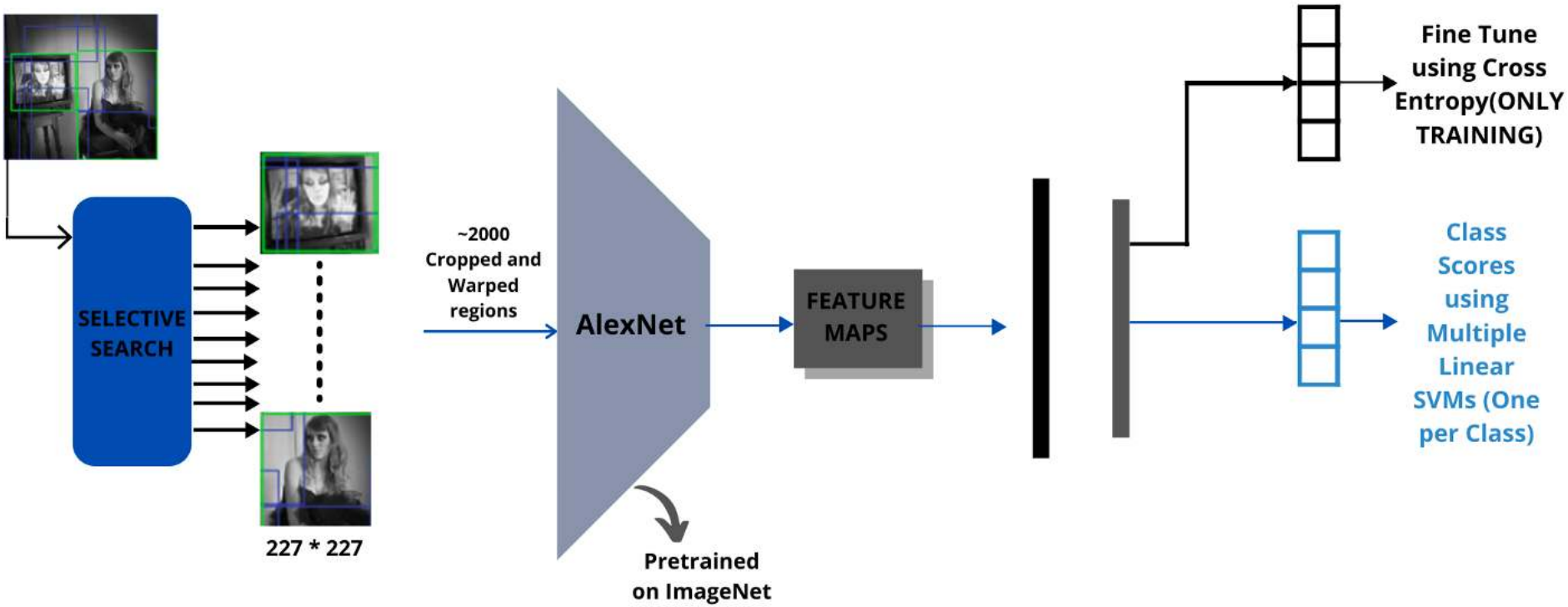


The hierarchical clustering is then combined with a CNN for classification of each region proposal.

The accuracy of this approach is not very high, requires changing the image aspect ratios

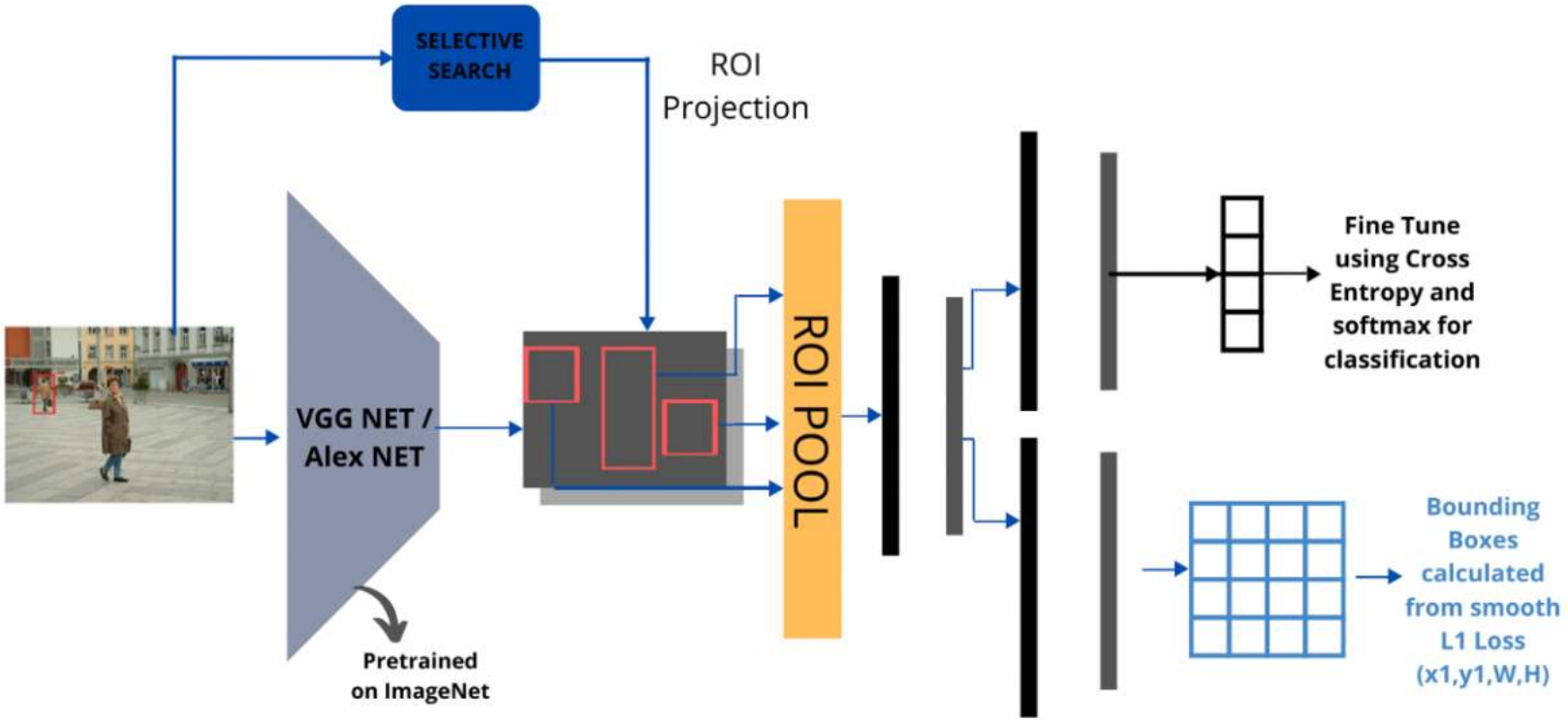


A solution was to add a fine tuning of the weights using an additional cross entropy loss

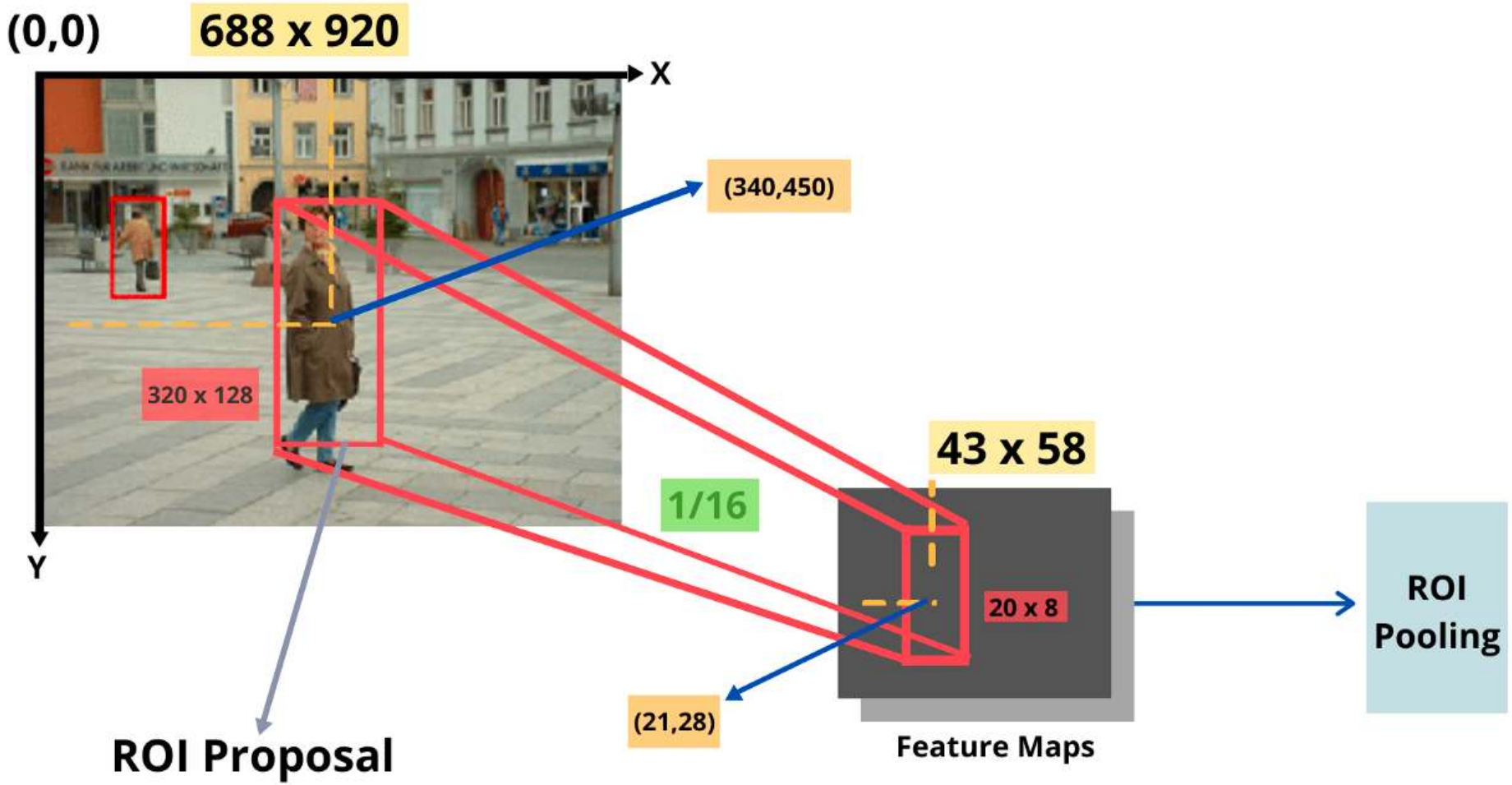


However, this requires a forward pass for every region proposal, which can be high

Fast R CNN

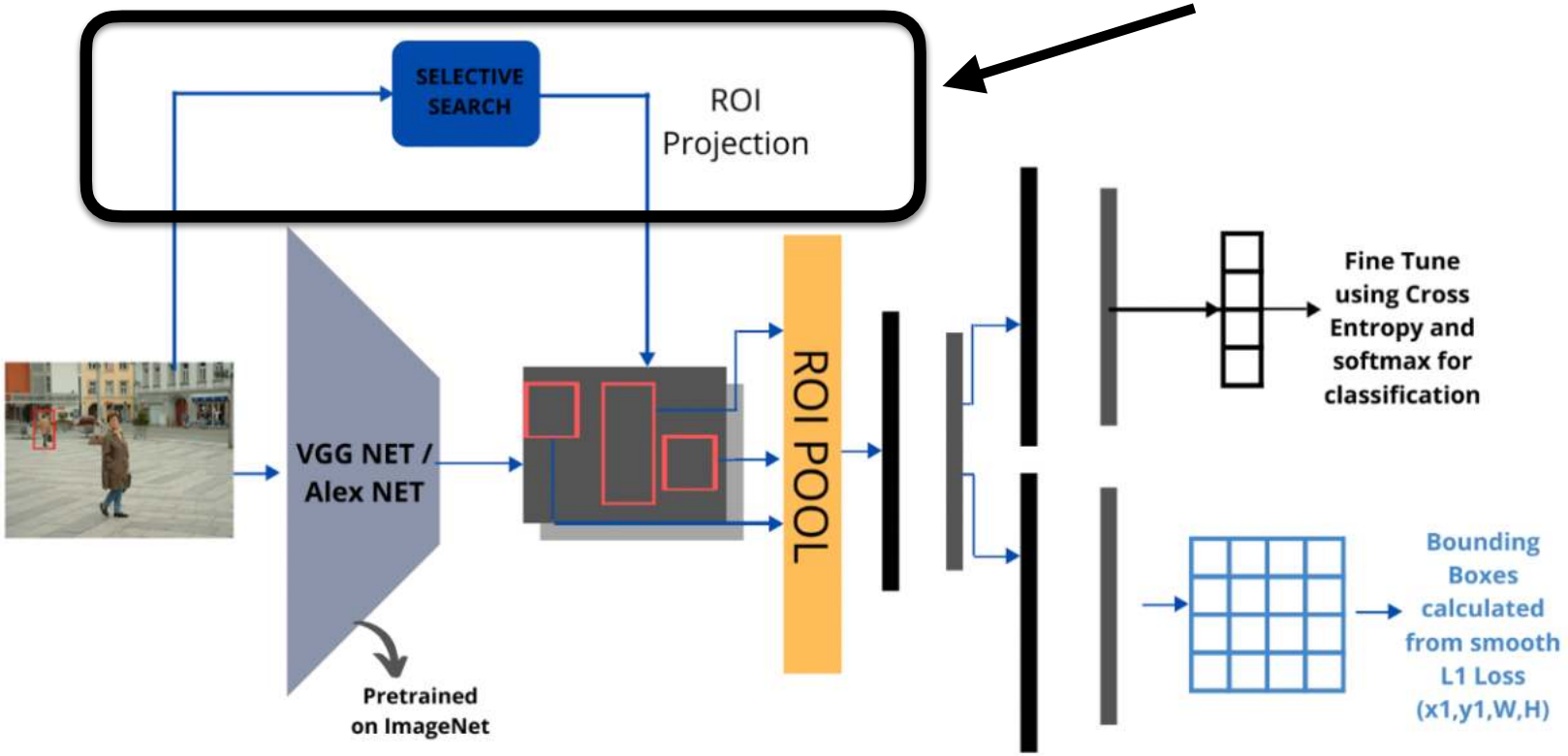


Fast R CNN uses only pass of the entire image by the CNN.


(0,0)
688 x 920
X
Y
(340,450)
320 x 128
1/16
43 x 58
20 x 8
(21,28)
Feature Maps
ROI Pooling
ROI Proposal

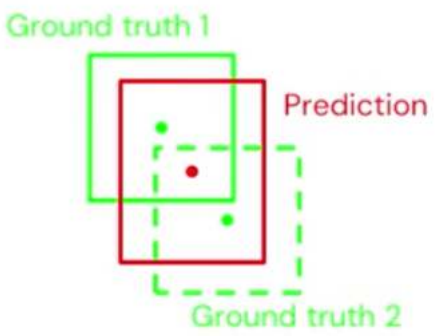
Fast R CNN

Why not
transforming this into a CNN?



WHAT WOULD BE THE LOSS FUNCTION OF SUCH A PROBLEM?

$$\frac{1}{N}\sum_{i=1}^{N}(y_i - p_i)^2$$

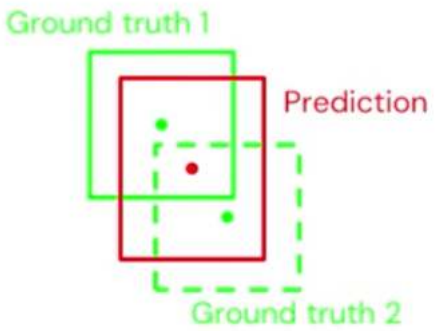


IT IS A REGRESSION WITH A SIMPLE QUADRATIC LOSS. WE TRY TO FIND THE BEST COORDINATES OF THE BOUNDING BOX.

@deepmind (V. Patraucean)

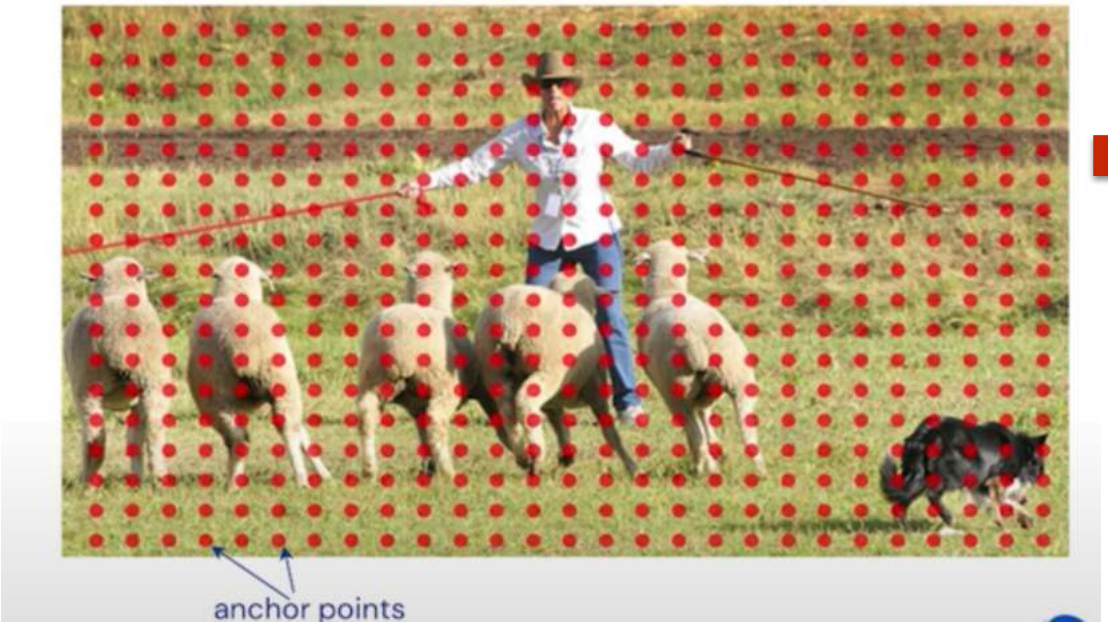
WHAT WOULD BE THE LOSS FUNCTION OF SUCH A PROBLEM?

$$\frac{1}{N}\sum_{i=1}^{N}(y_i - p_i)^2$$



IT IS A REGRESSION WITH A SIMPLE QUADRATIC LOSS. WE TRY TO FIND THE BEST COORDINATES OF THE BOUNDING BOX.

IT BECOMES MESSY QUITE RAPIDLY…



Discretize Bounding Box Space

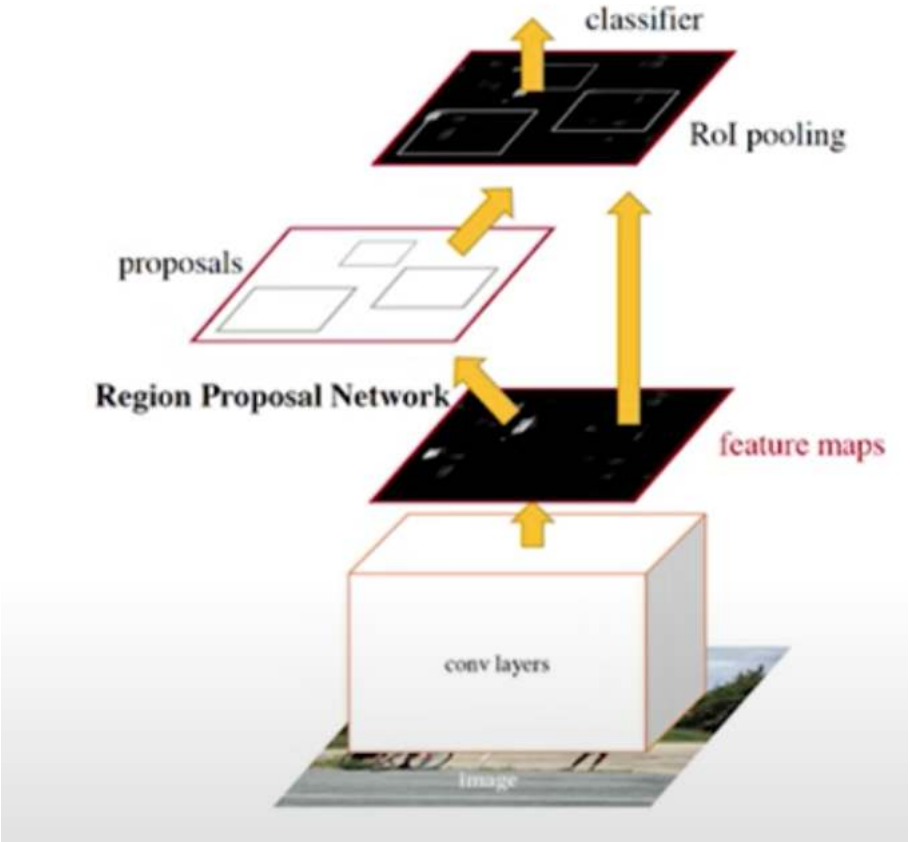
Choose n candidates per position

Predict objectness score (classification)

Sort and keep top K

@deepmind (V. Patraucean)

FASTER R-CNN



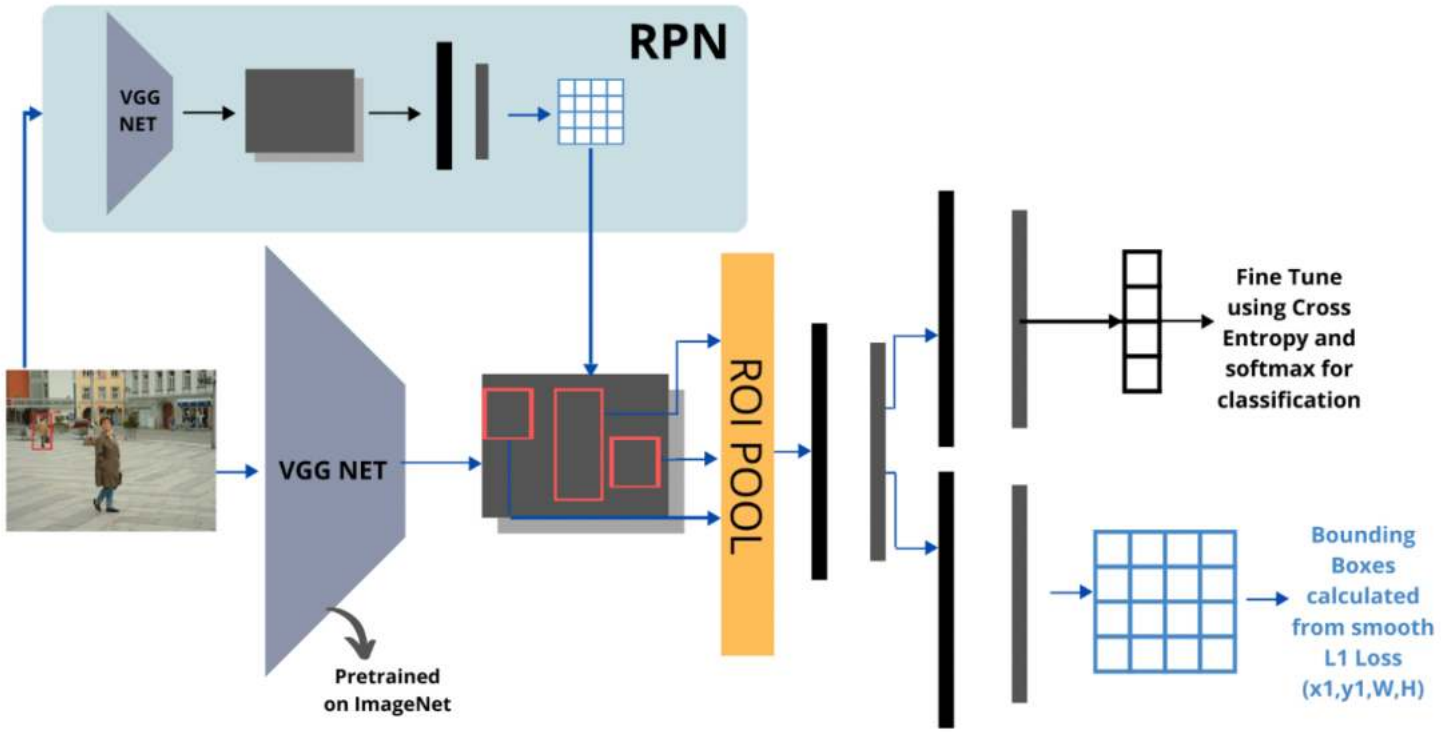
We divide the task in 2 steps:

1. Identify bounding box candidates (CLASSIFICATION)

2. Classify and Refine (REGRESSION)

Ren+16

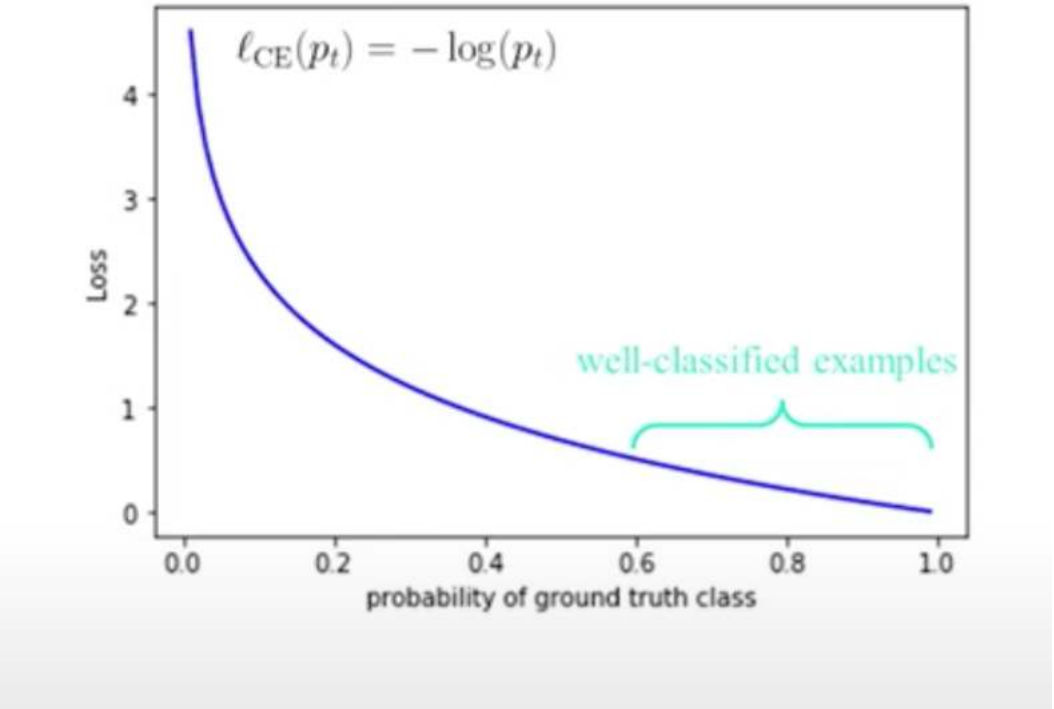
FASTER R-CNN


RPN
VGG NET
VGG NET
Pretrained on ImageNet
ROI POOL
Fine Tune using Cross Entropy and softmax for classification
Bounding Boxes calculated from smooth L1 Loss (x1,y1,W,H)

WHY NOT DOING IT ONE STAGE?

MOST OF THE CANDIDATES
ARE BACKGROUND,
EASY TO IDENTIFY

THE LOSS OF THE MANY
EASY EXAMPLES
DOMINATES OVER THE
RARE USEFUL ONES



@deepmind (V. Patraucean)

YOLO: You Only Look Once

Transform everything in a big regression

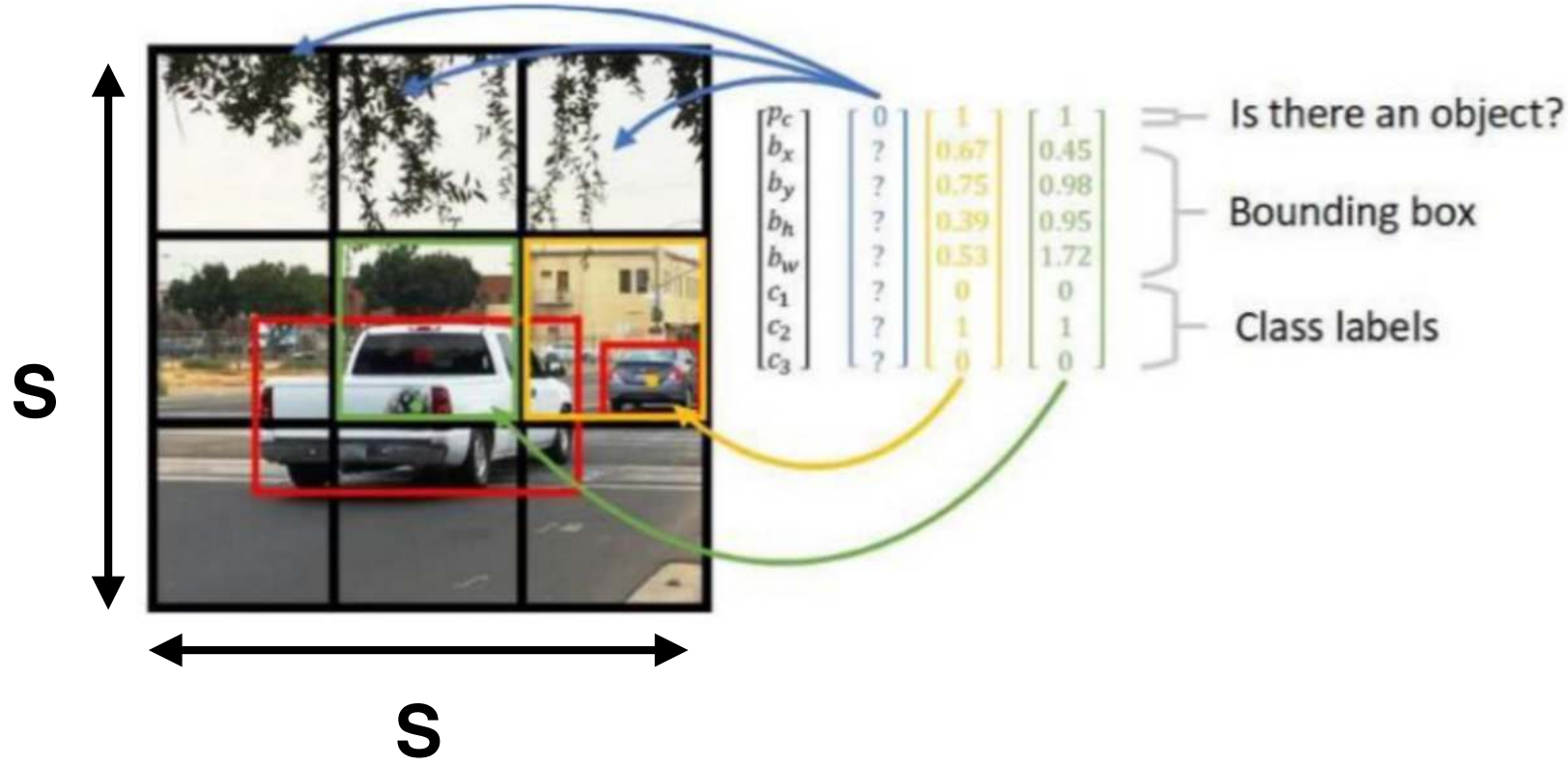
S × S grid on input

1. Divide the image in cells

2. Each cell is responsible for detecting B bounding boxes as well as a confidence class for each box

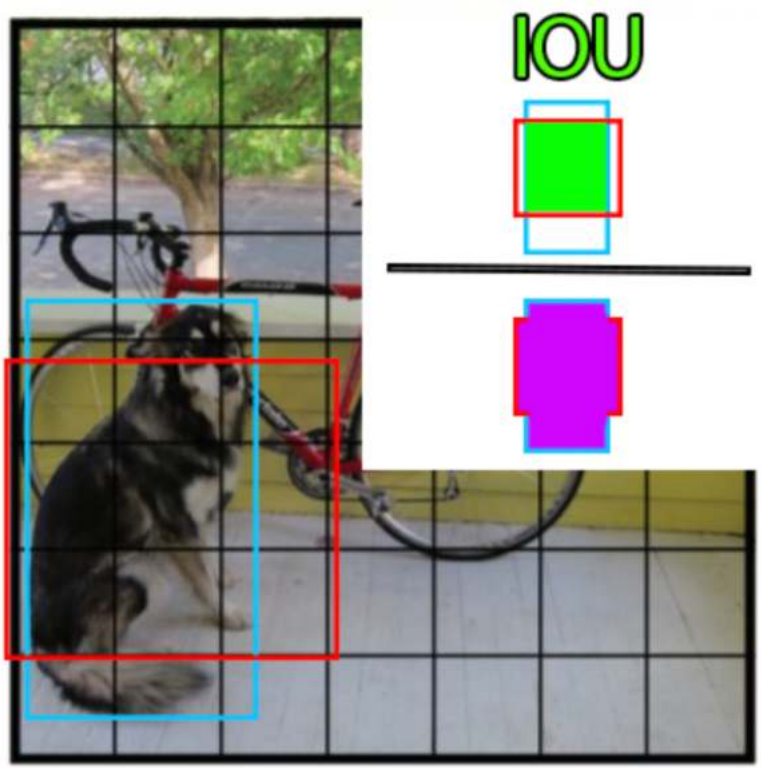
Each bounding box is parametrised with 5 numbers:
- coordinates: x,y
- heights width: w, h
- confidence: c

Then we attach to each bounding box, a one-hot encoder vector contains the class labels

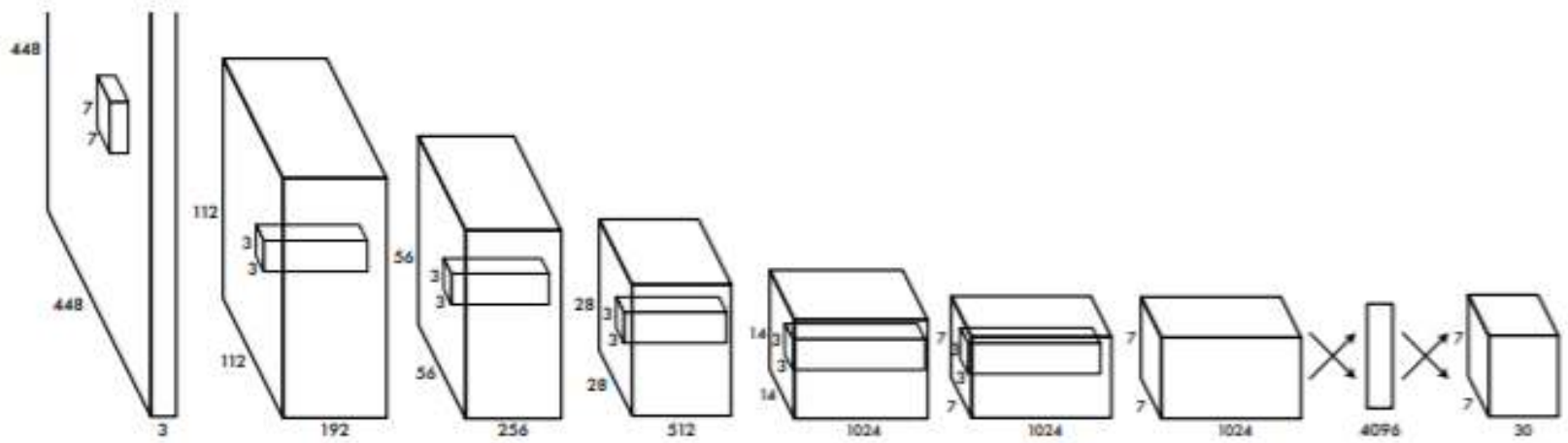


Measuring the “confidence” of a box

WE TYPICALLY USE THE INTERSECTION OVER UNION LOSS



YOLO REGRESSION NETWORK



$$S \times S \times (C + B * 5)$$

$$\lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

$$+ \lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2$$

$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} (x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2$$

loss related to the predicted bounding box position (x,y)

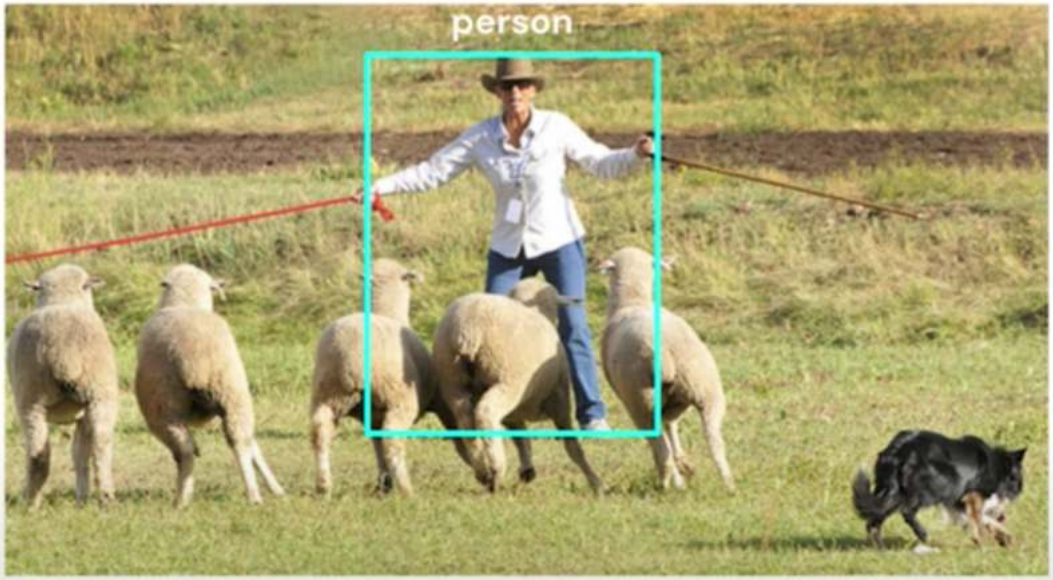
$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} (\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2$$

loss related to the predicted bounding box height and width

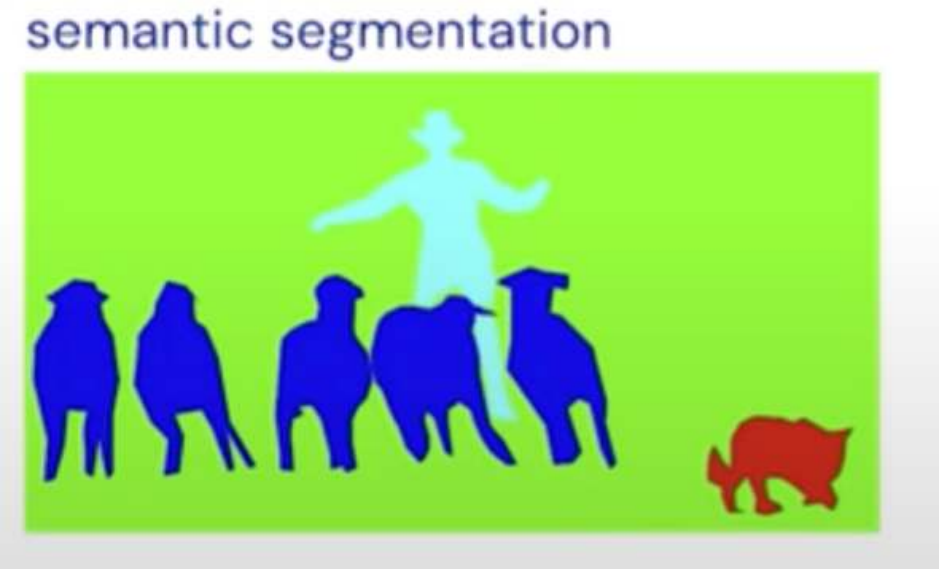
$$\sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} (C_i - \hat{C}_i)^2 + \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{noobj} (C_i - \hat{C}_i)^2$$

loss related to the correctness of the bounding box

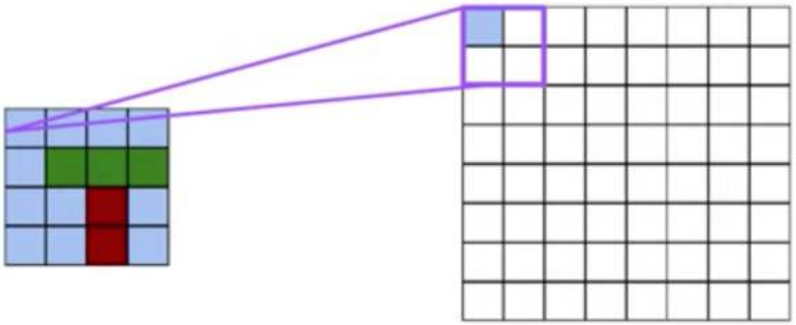
LET'S GO A STEP FURTHER INTO SEMANTIC SEGMENTATION



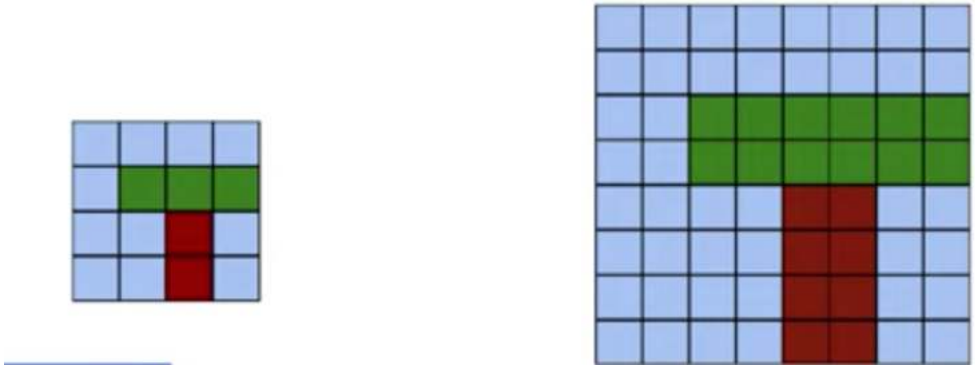
BOUNDING BOXES ARE NOT ALWAYS GOOD REPRESENTATIONS



UNPOOLING OPERATION (INVERSE OF POOLING)



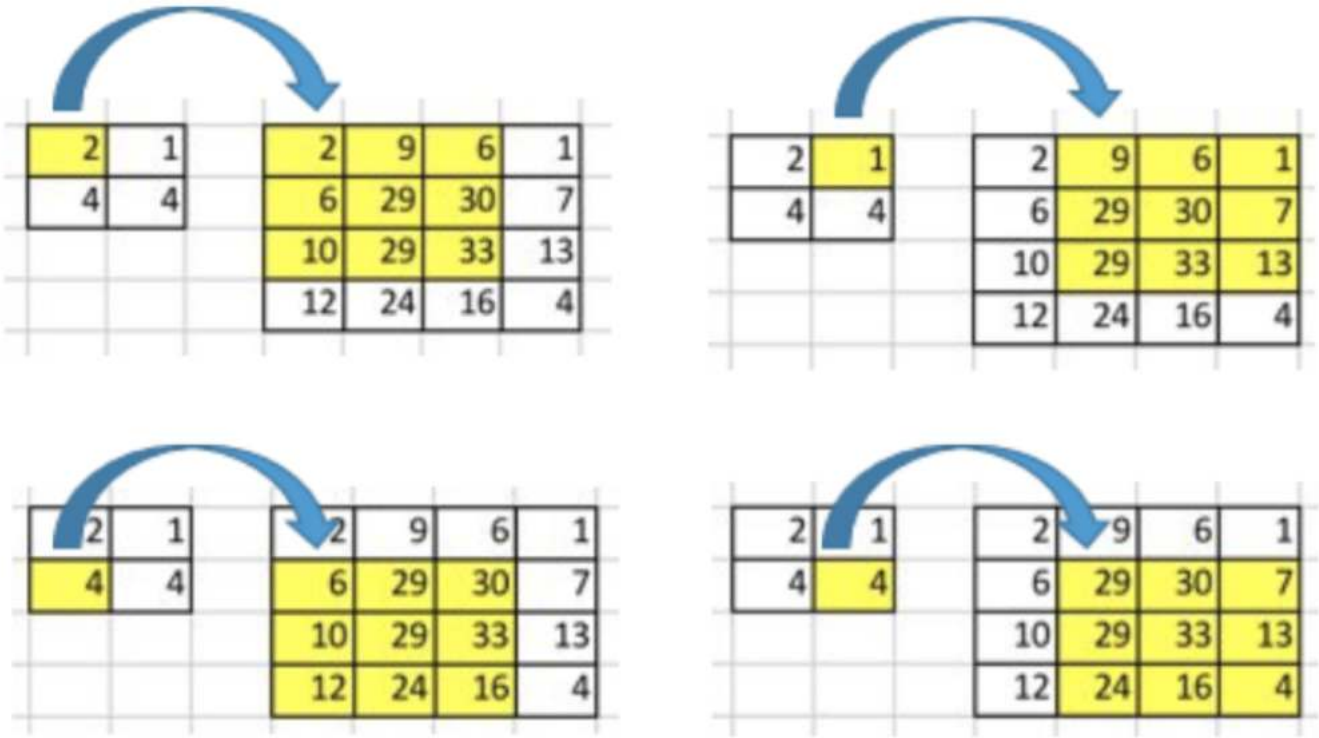
COPY PIXELS IN A GIVEN WINDOW



GENERATES LARGER IMAGES FROM SMALLER ONES

TRANSPOSED CONVOLUTION

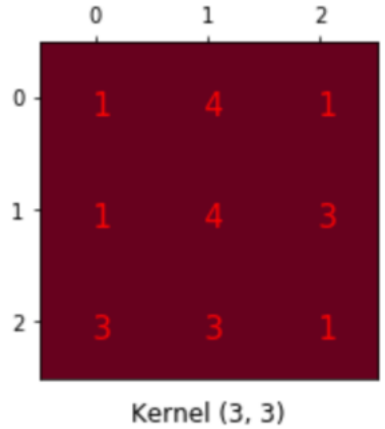
ALLOWS TO INCREASE THE SIZE



Going Backward of Convolution

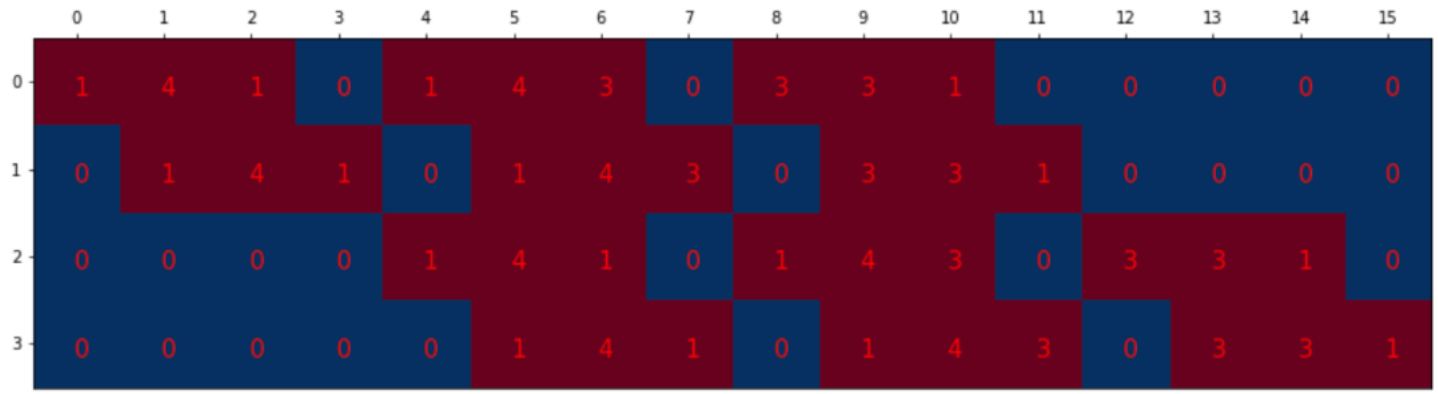
EXAMPLE TAKEN FROM HERE

CONVOLUTION MATRIX



Kernel (3, 3)

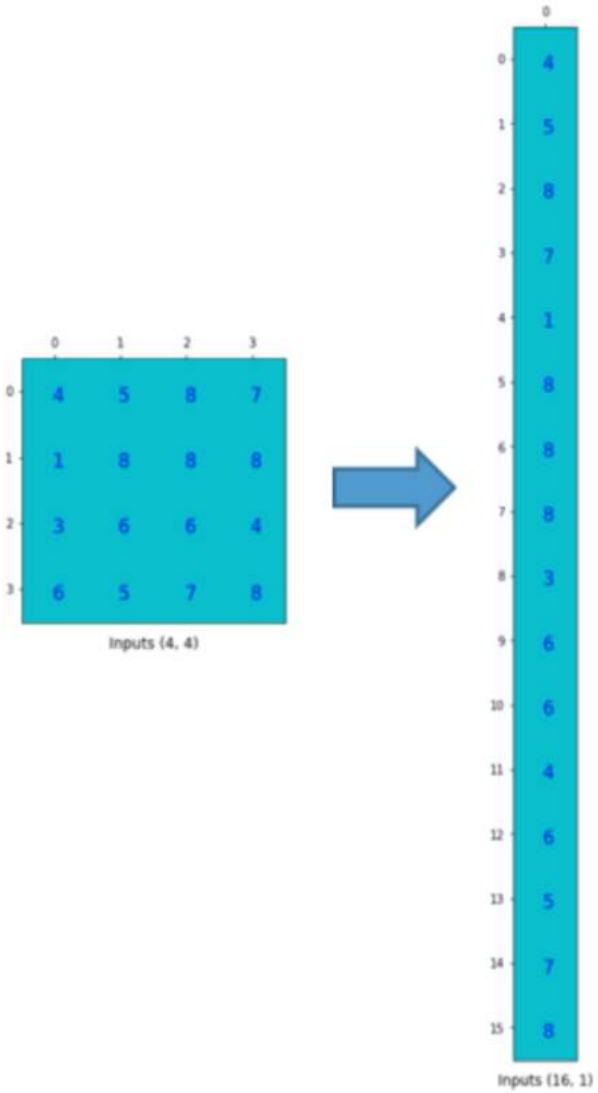
THE KERNEL CAN BE ARRANGED IN FORM OF A MATRIX:



Convolution Matrix (4, 16)

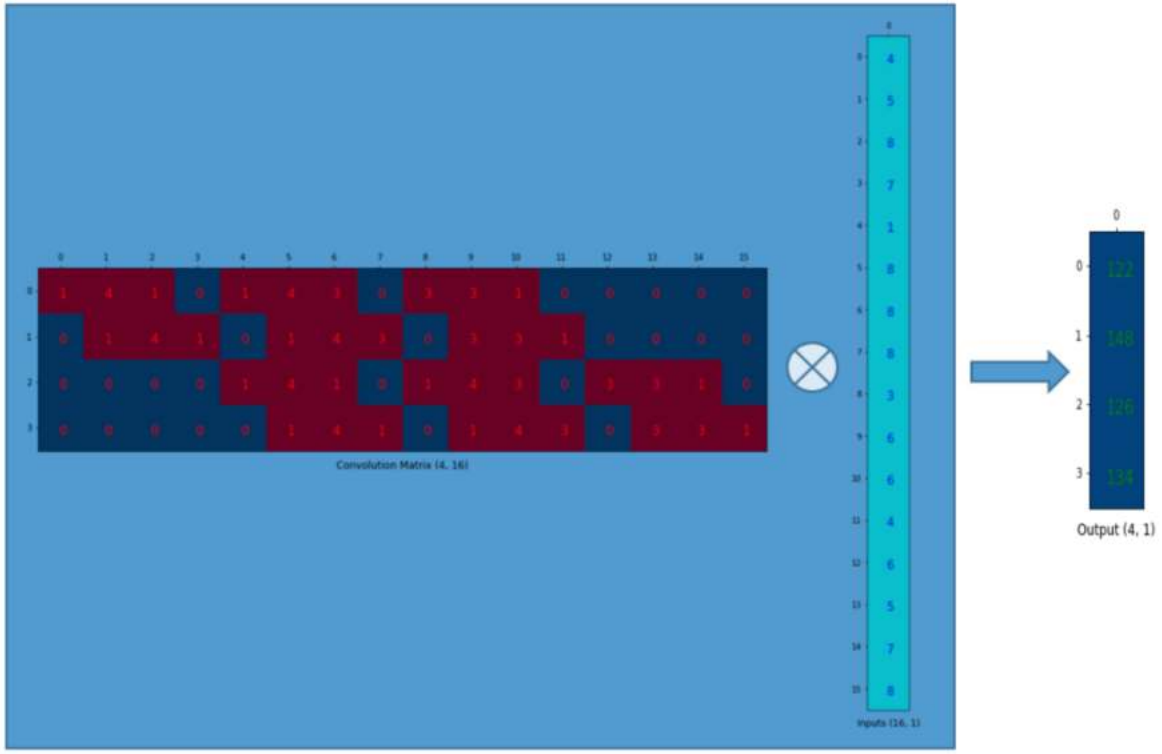
EXAMPLE TAKEN FROM HERE

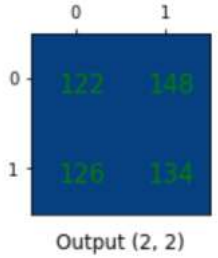
THE INPUT IS FLATTENED INTO A COLUMN VECTOR



EXAMPLE TAKEN FROM HERE

THE CONVOLUTION IS TRANSFORMED INTO A PRODUCT OF MATRICES

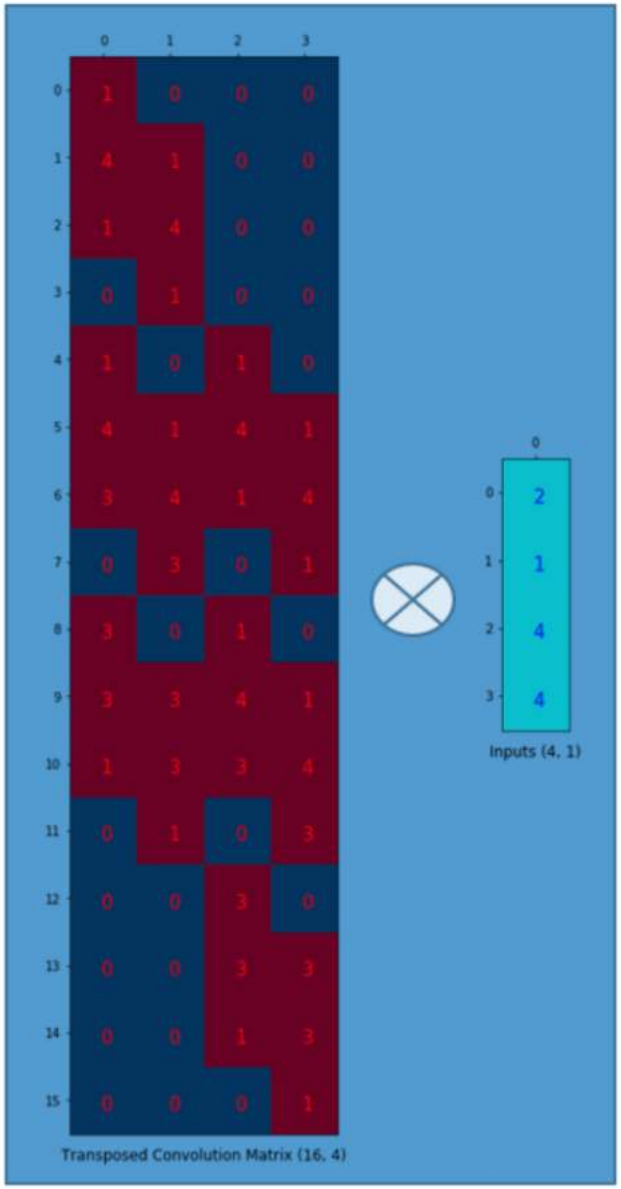


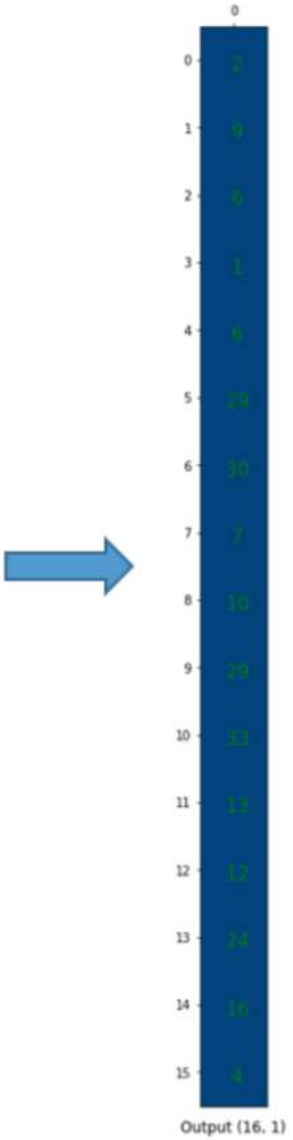


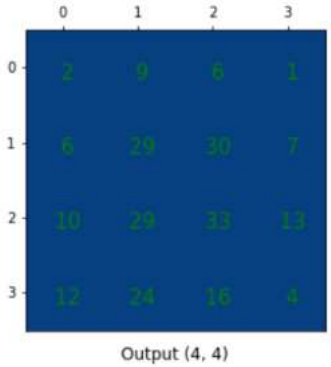
RESHAPED OUTPUT

EXAMPLE TAKEN FROM HERE

THE TRANSPOSED CONVOLUTION IS THE INVERSE OPERATION



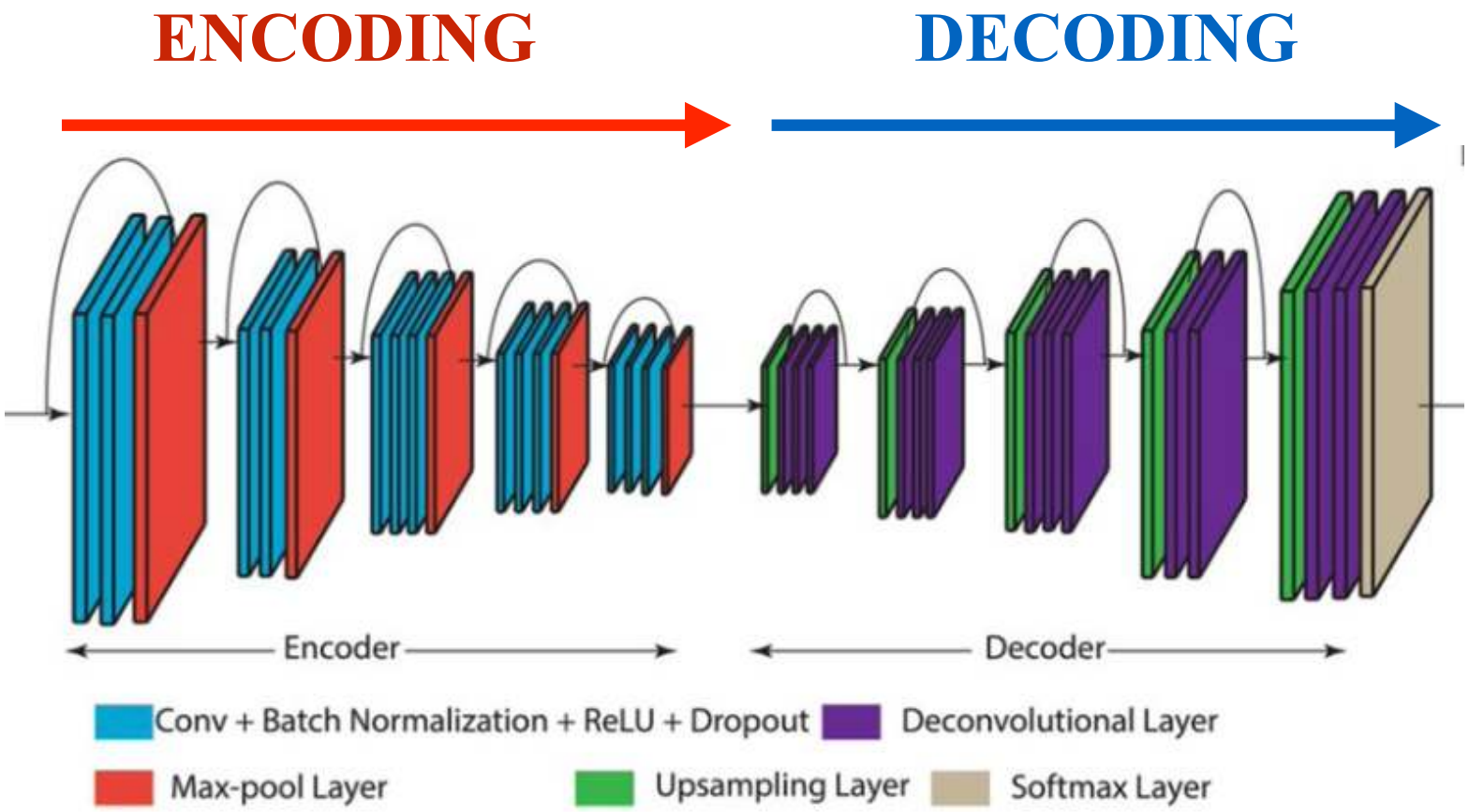




RESHAPED OUTPUT

EXAMPLE TAKEN FROM HERE

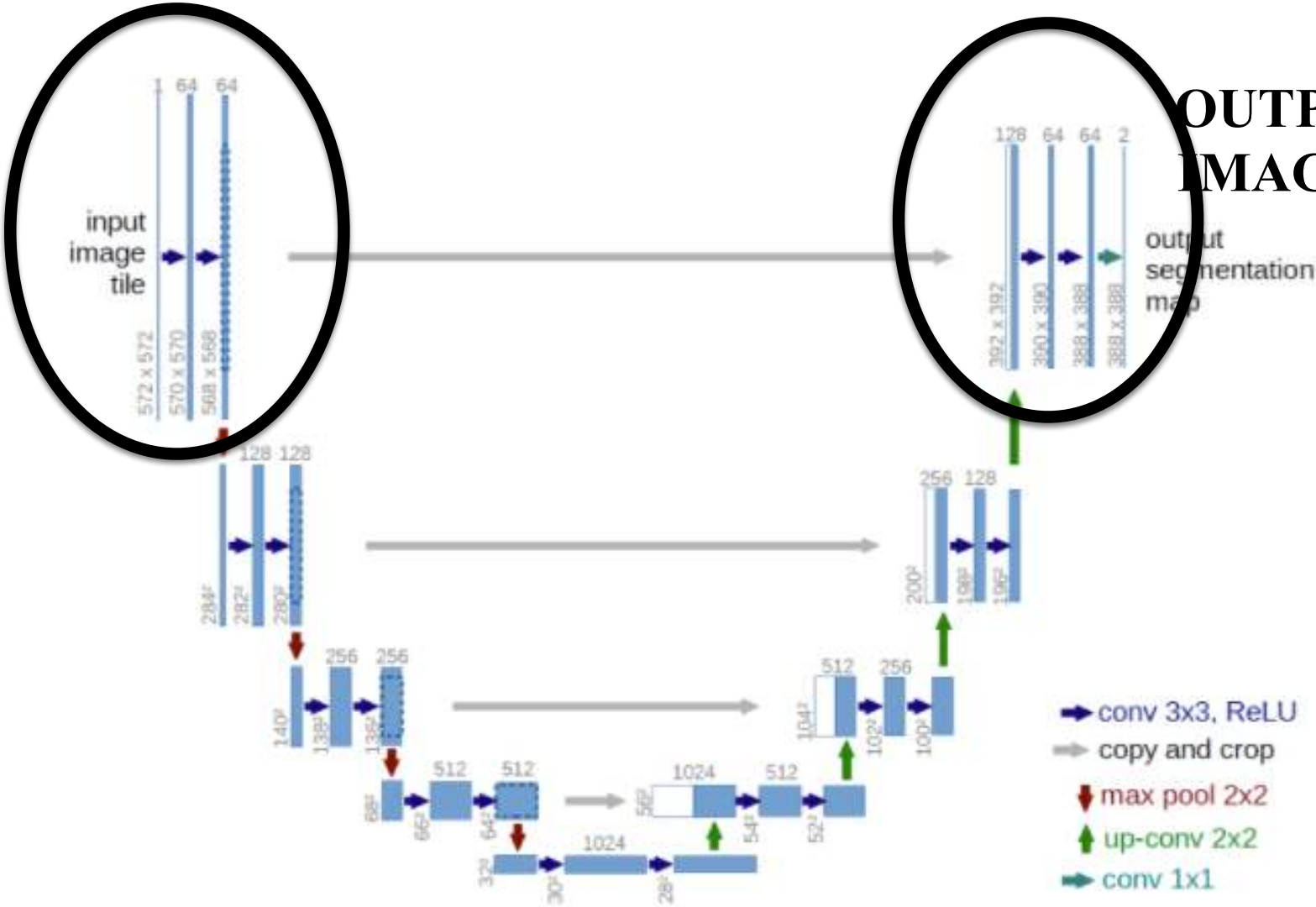
ENCODER-DECODERS GO FROM IMAGE 2 IMAGE



WE CALL THIS FULLY CONVOLUTIONAL NEURAL NETWORKS

ENCODING-DECODING TO EXTRACT IMAGE FEATURES: U-NET

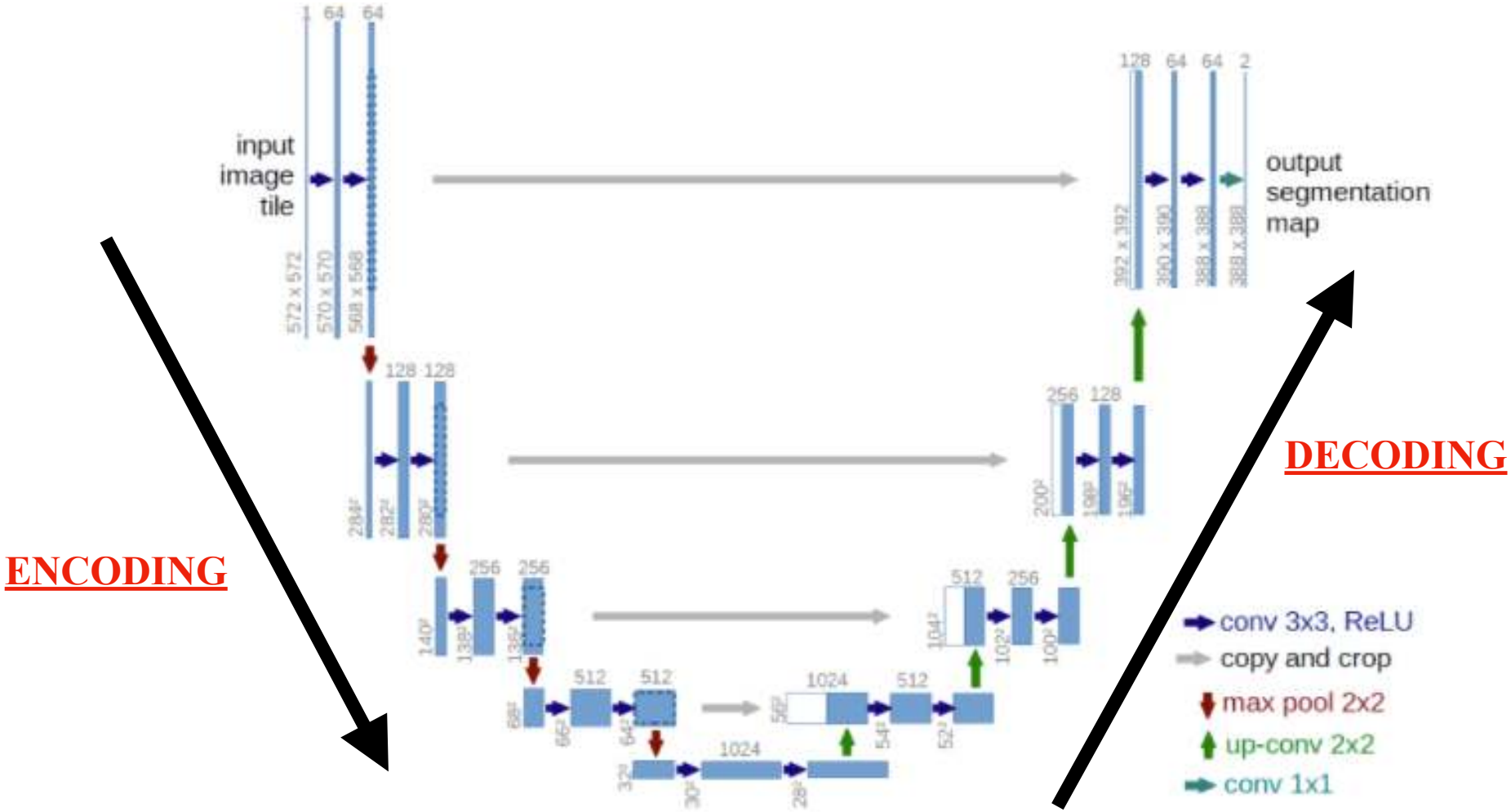
INPUT IMAGES



OUTPUT IMAGES

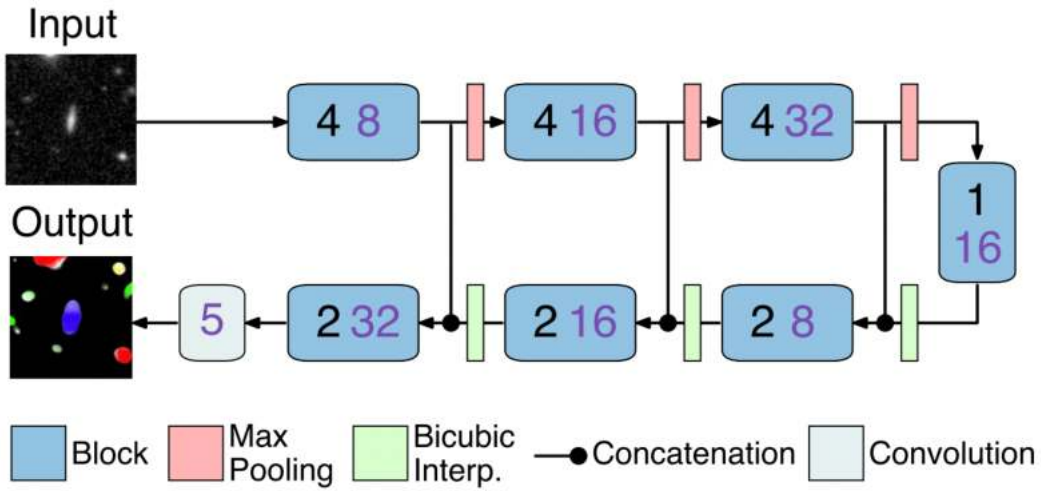
Ronnerberger+15

ENCODING-DECODING TO EXTRACT IMAGE FEATURES: THE U-NET

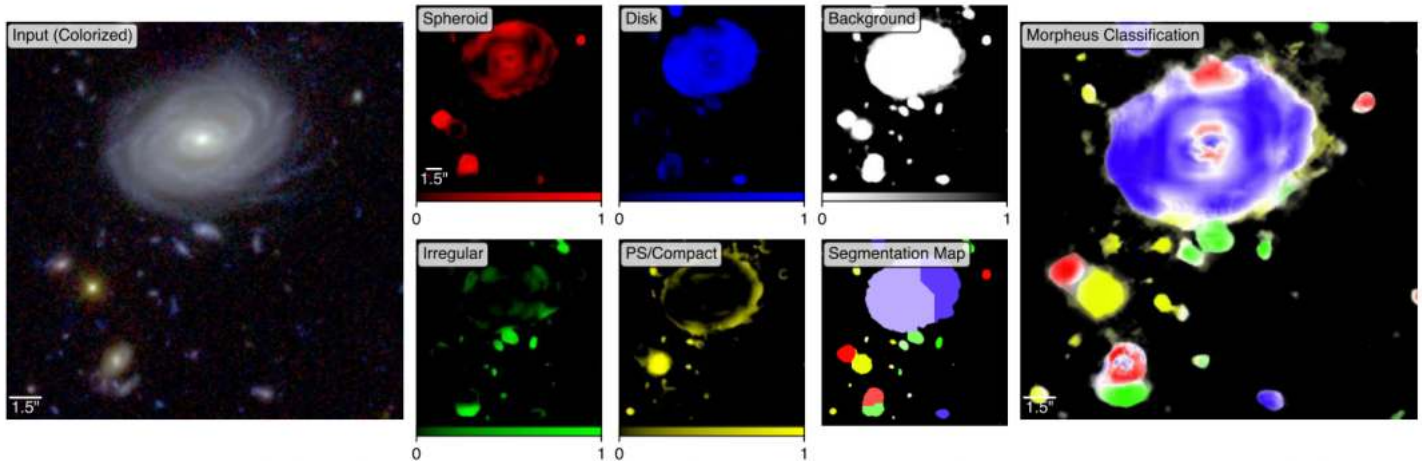


Ronnerberger+15

2. Segmentation



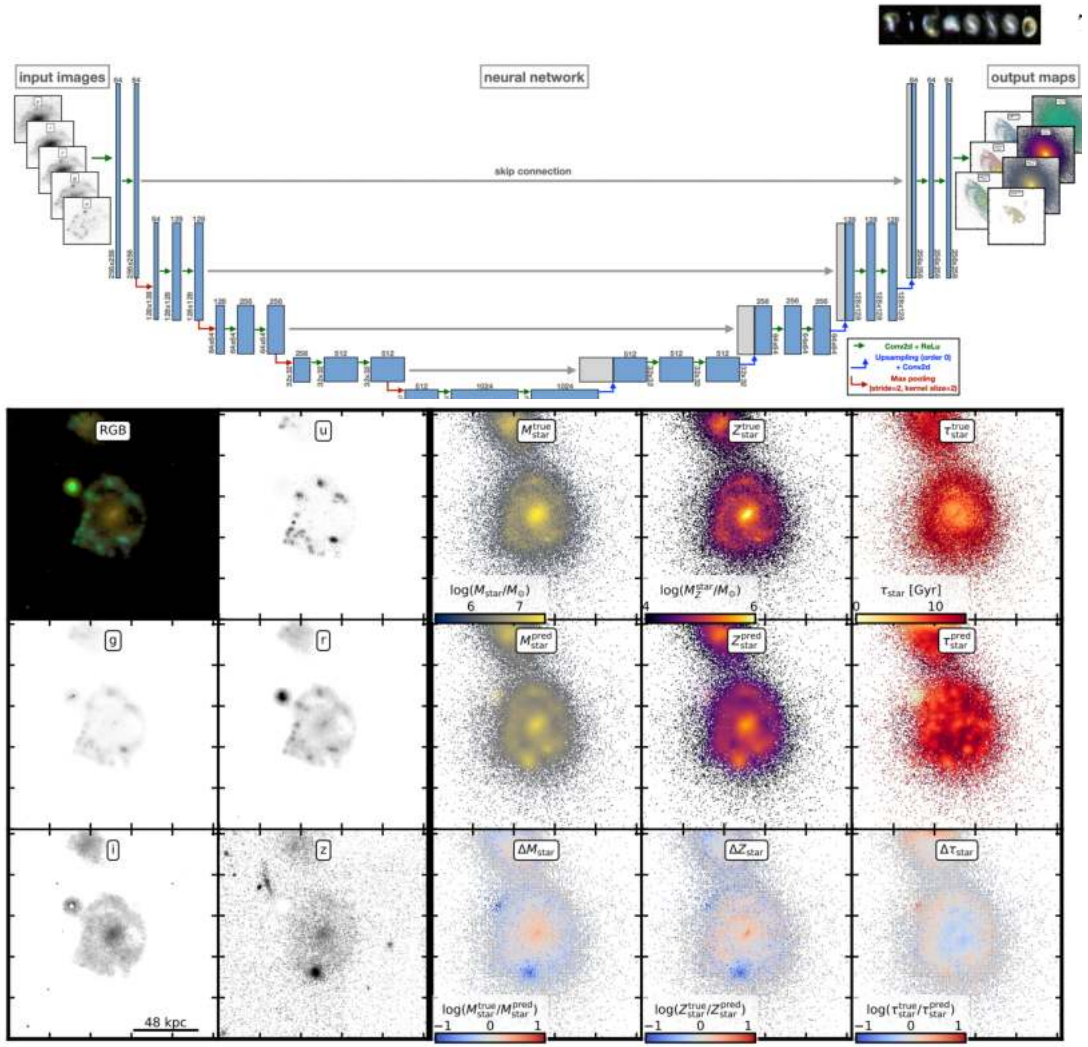
Classification at the pixel level



Hausen+20

Dieleman+15, Huertas-Company+15, Aniyan+17, Charnock+17, Gieseke+17, Jacobs+17, Petrillo+17, Schawinski+17, Alhassan+18, Dominguez-Sanchez+18, George+18, Hinners+18, Lukic+18, Moss+18, Razzano+18, Schaefer+18, Allen+19, Burke+19, Carrasco-Davis+19, Chatterjee+19, Davies+19, Dominguez-Sanchez+19, Fusse+19, Glaser+19, Ishida+19, Jacobs+19, Katebi+19, Lanusse+19Liu+19, Lukic+19, Metcalf+19, Muthukrishna+19, Petrillo+19, Reiman+19, Boucaud+20, Chiani+20, Ghosh+20, Gomez+20, Hausen+20, Hlozek+20, Hosseinzadeh+20. Huang+20, Li+20, Moller+20, Paillassa+20, Tadaki+20, Vargas dos Santos+20, Walmsley+20, Wei+20, Allam+21, Arcelin+21, Becker+21, Bretonniere+21, Burhanudin+21, Davison+21, Donoso-Oliva+21, Jia+21, Lauritsen+21, Ono+21, Ruan+21, Sadegho+21, Tang+21, Tanoglidis+21, Vojtekova+21, Dhar+22Hausen+22, Orwat+22, Pimentel+22, Rezaei+22, Samudre+22, Shen+22, Walmsley+22

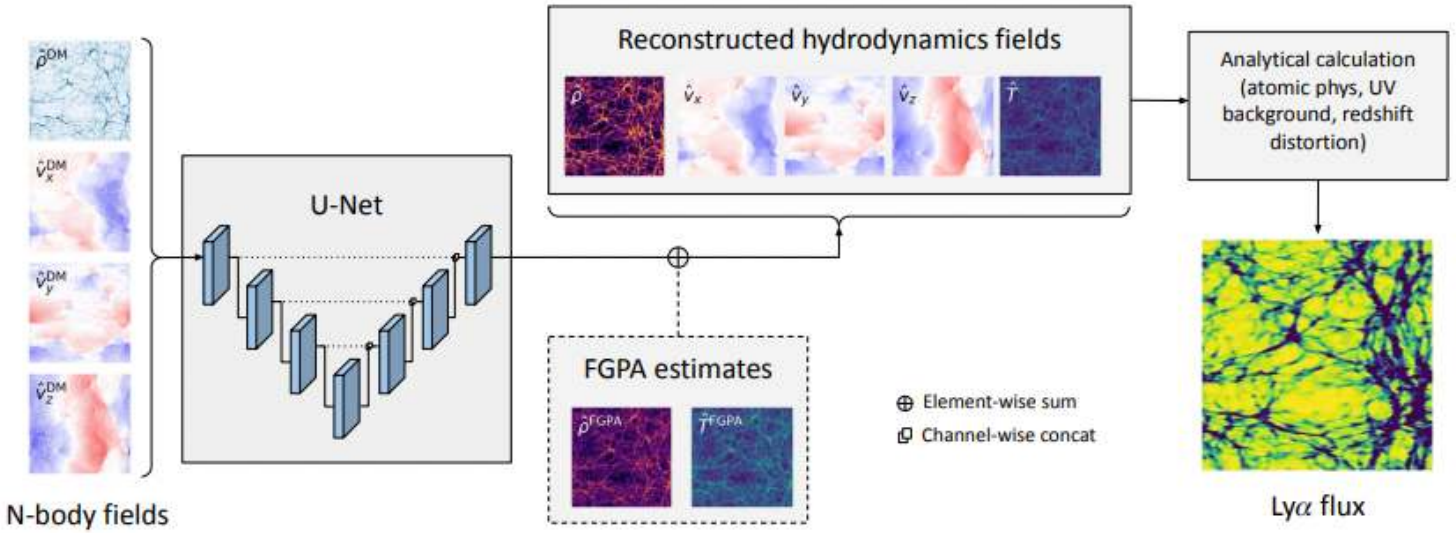
Stellar Populations



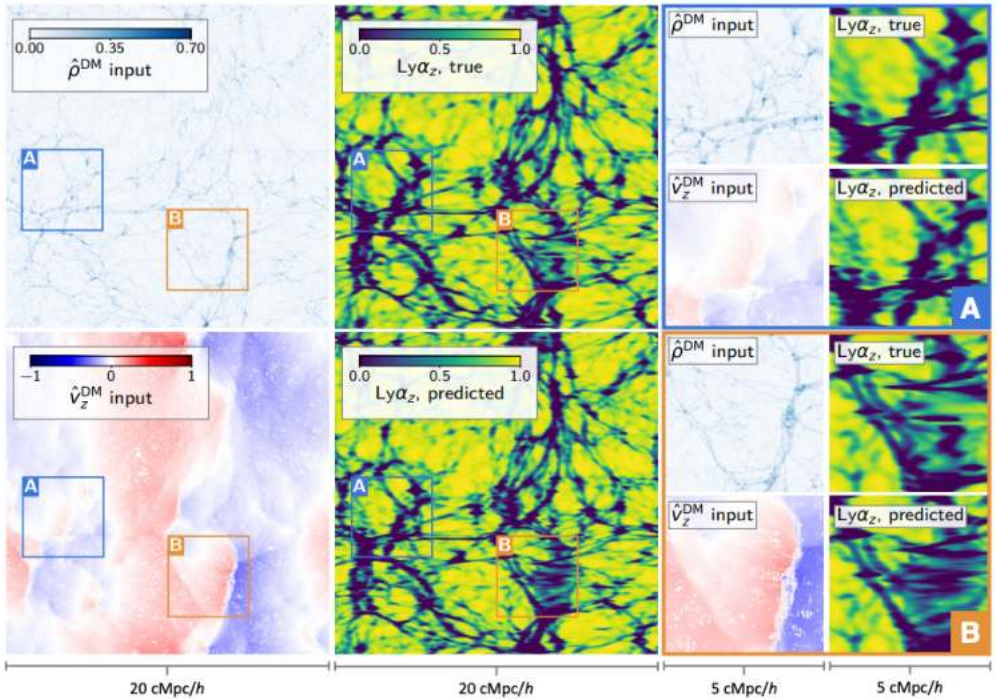
Buck+21

Hazevah+17, Perreault-Levasseur+17, Morningstar+18, Stark+18, Tuccillo+18, Bom+19, Lovell+19, Madireddy+19, Pearson+19, Simet+19, Pasquet+19, Wu+19, Menou+19, Aragon-Calvo+20, Chianese+20, Stahl+20, Surana+20, Shuntov+20, Cabayol-Garcia+20, Campagne+20, Buck+21, Grover+21, Hayat+21, Li+21, Maresca+21, Qiu+21, Rhea+21, Schuldt+21, Tohill+21, Yao-Yu+21, Dey+21, Lee+21, Zhou+21, Henghes+21, Ansari+21

Painting Baryons

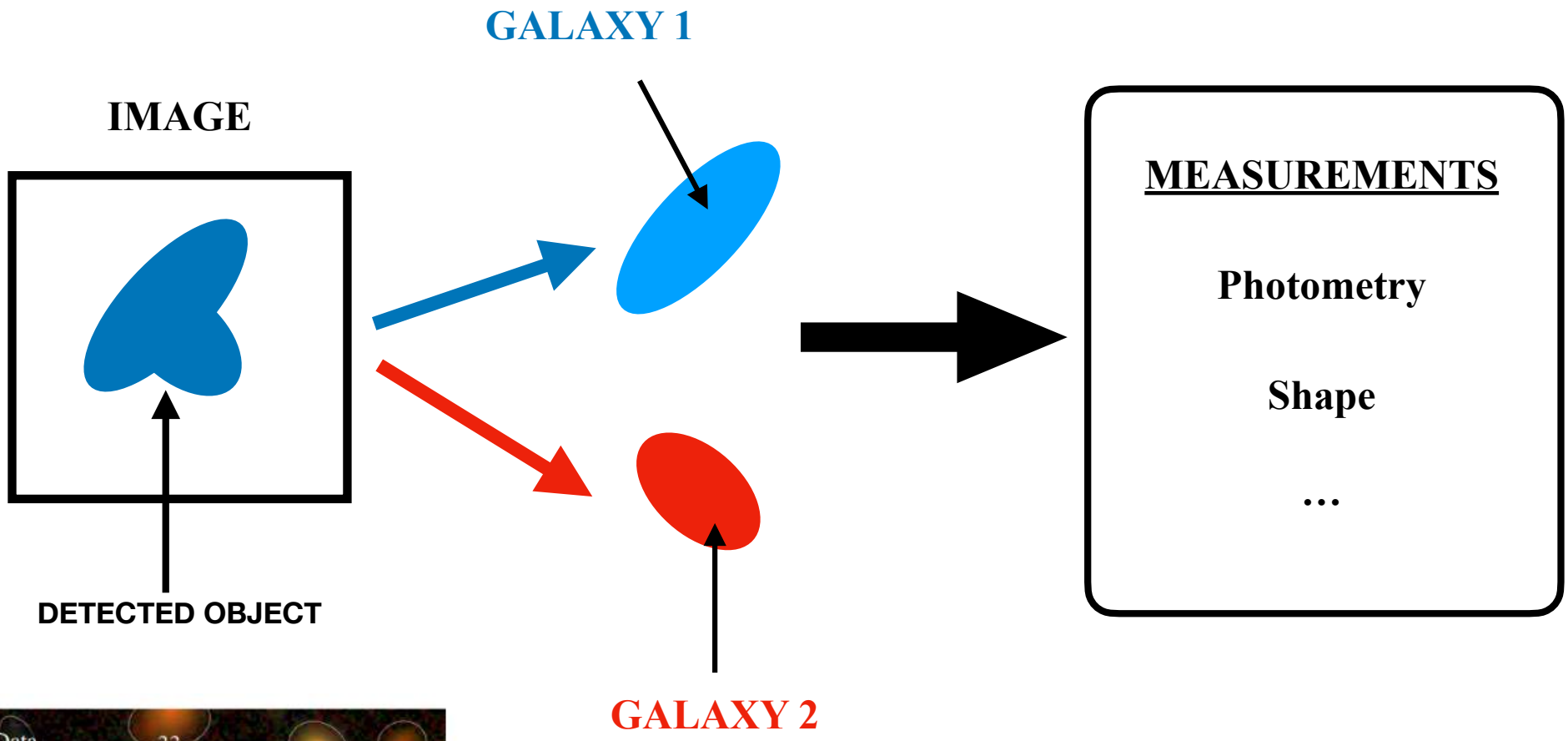


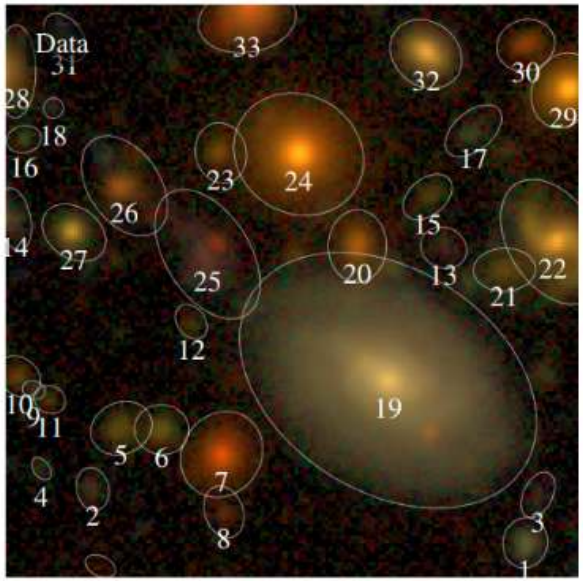
Harrington+21



Neural Networks are used to learn the non-linear mapping between cheap dark matter only simulations to expensive baryonic physics

Rodriguez+19, Modi+18, Berger+18, He+18, Zhang+19, Troster+19, Zamudio-Fernandez+19, Perraudin+19, Charnock+19, List+19, Giusarma+19, Bernardini+19, Chardin+19, Mustafa+19, Ramanah+20, Tamosiunas+20, Feder+20, Moster+20, Thiele+20, Wadekar+20, Dai+20, Li+20, Lucie-Smith+20, Kasmanoff+20, Ni+21, Rouhiainen+21, Harrington+21, Horowitz+21, Horowitz+21, Bernardini+21, Schaurecker+21, Etezad-Razavi+21, Curtis+21





Bosch+17

>50% of objects will be affected by blending in future deep surveys such as LSST

Boucaud+19

GALAXY 1

IMAGE

DETECTED OBJECT

GALAXY 2

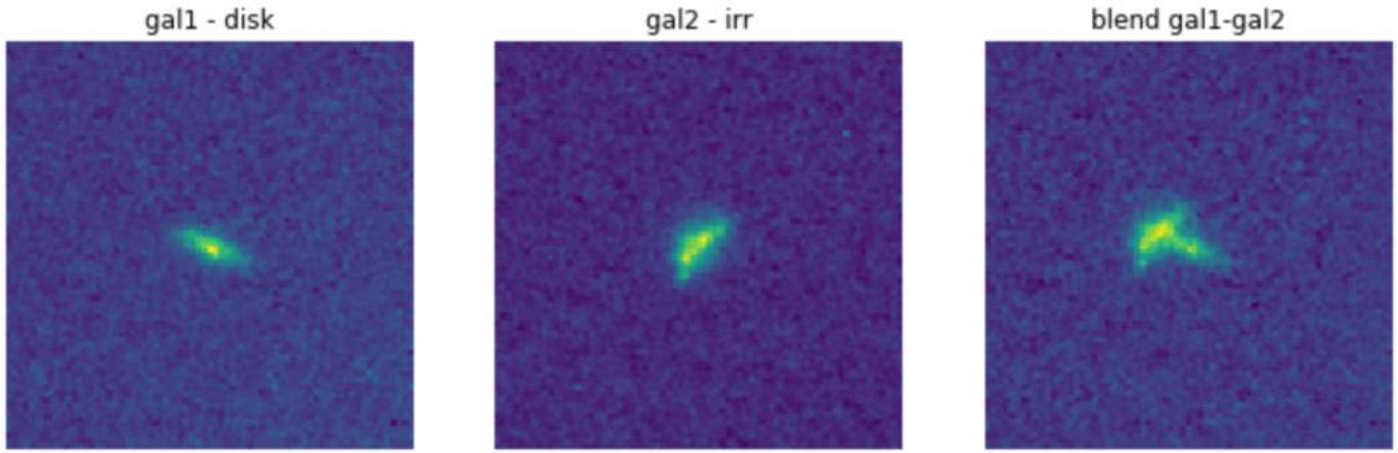
MEASUREMENTS

Photometry

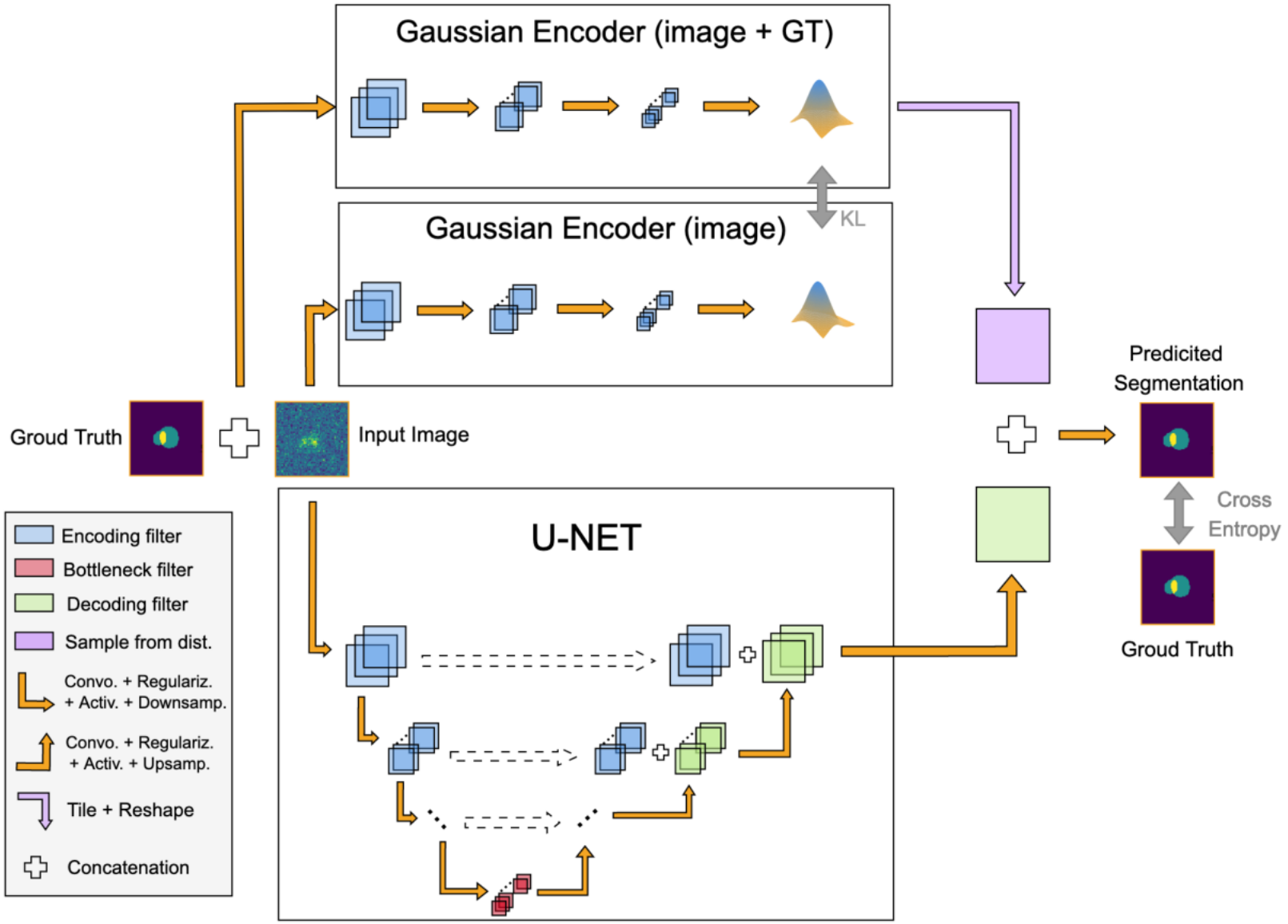
Shape

…

ISOLATED GALAXIES ARTIFICALLY BLENDED



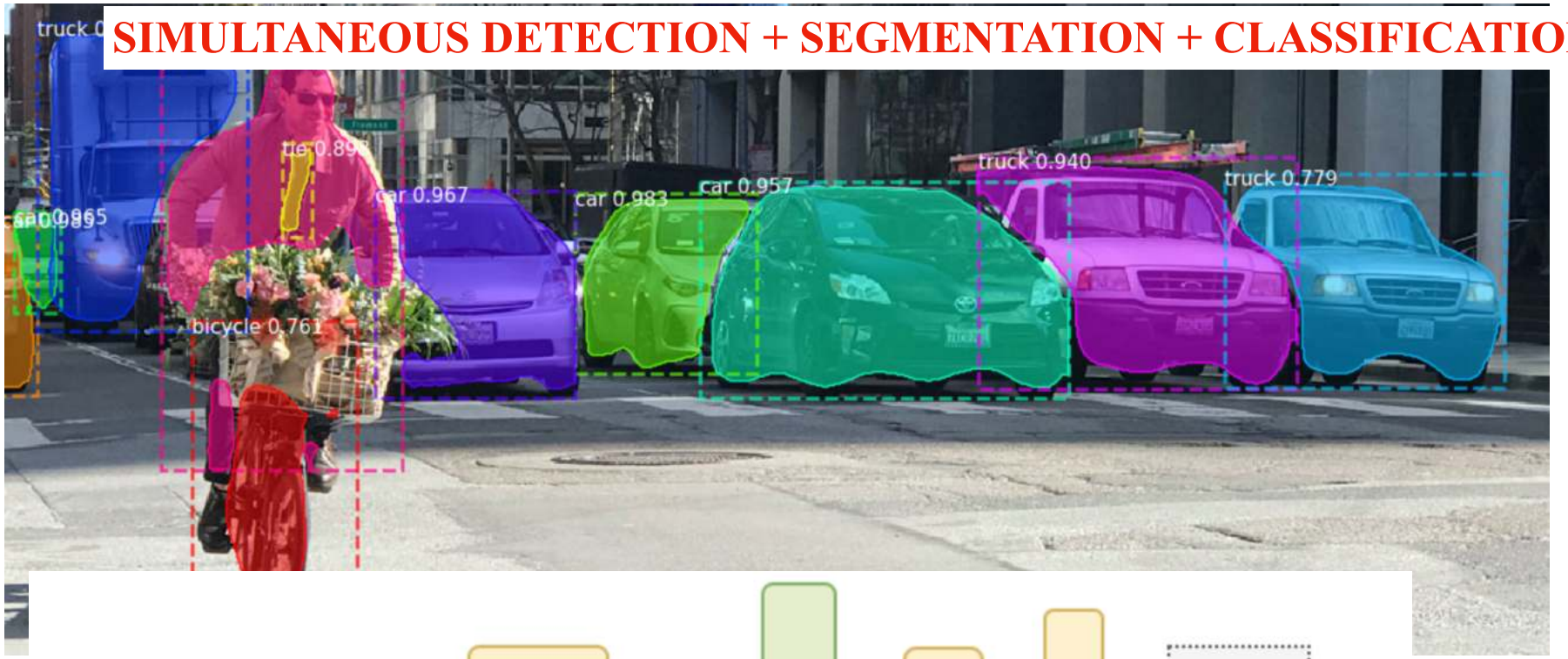
PROBABILISTIC U-NET

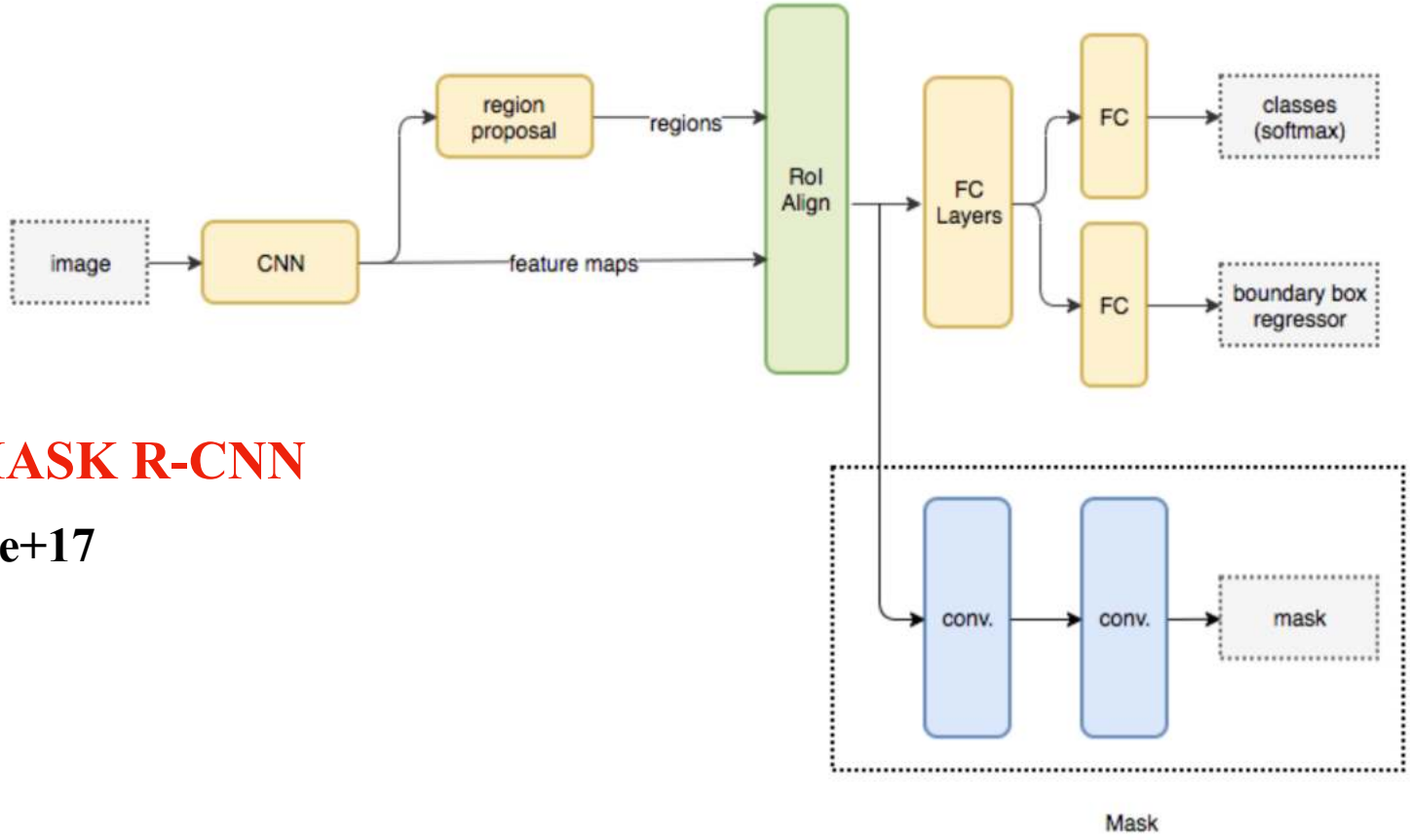


Bretonniere+20

instance segmentation

SIMULTANEOUS DETECTION + SEGMENTATION + CLASSIFICATION

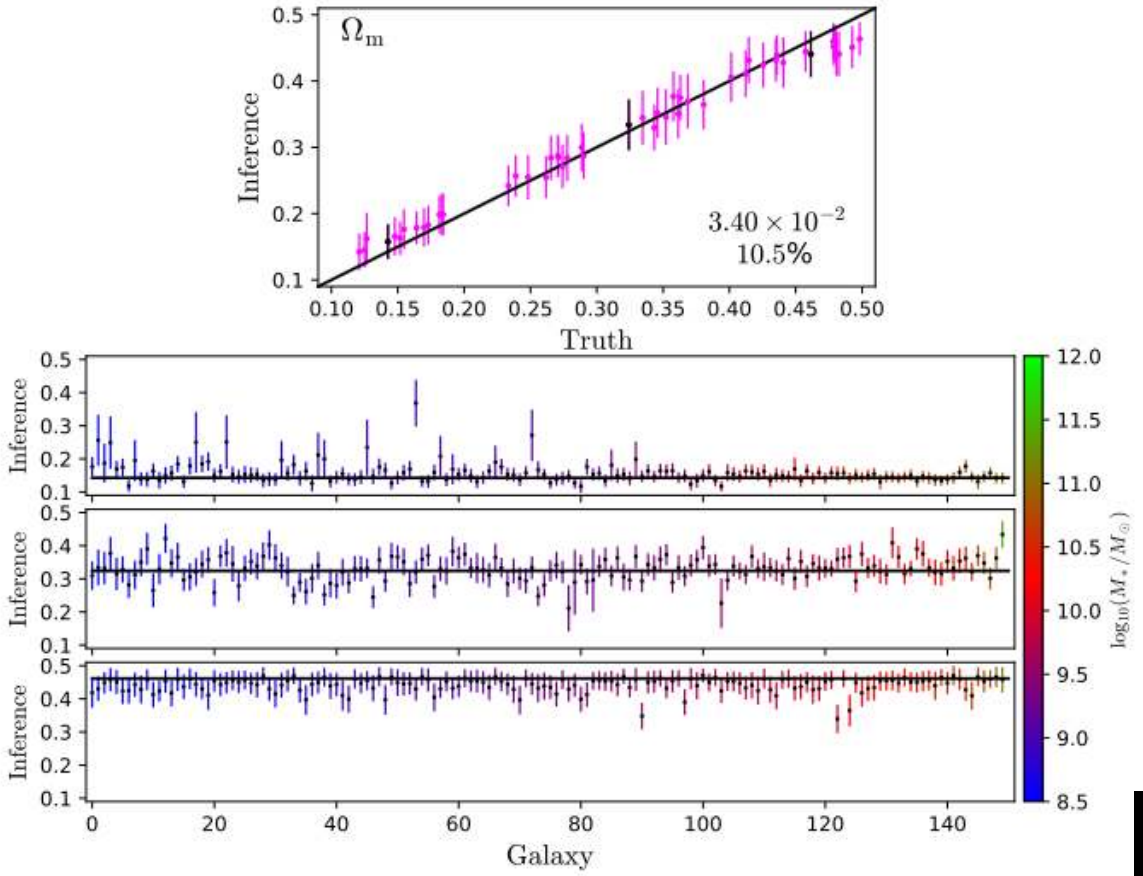


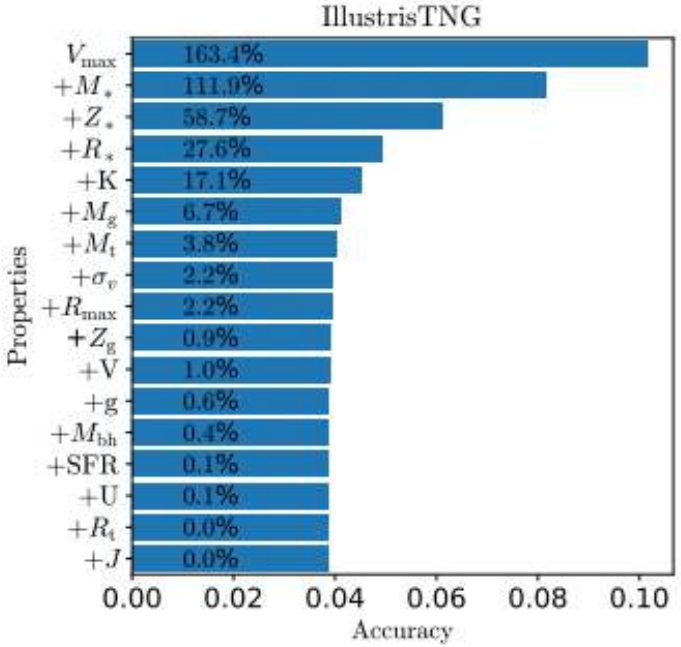


MASK R-CNN

He+17

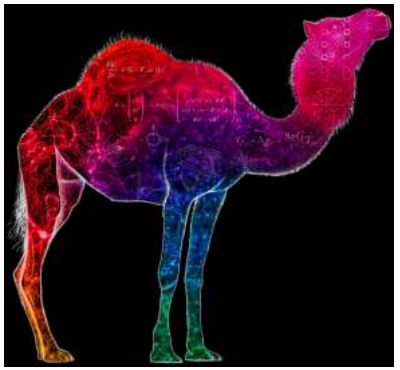
Cosmological Inference





Villaescusa-Navarro+22

Neural Networks to learn new relations between models and observables

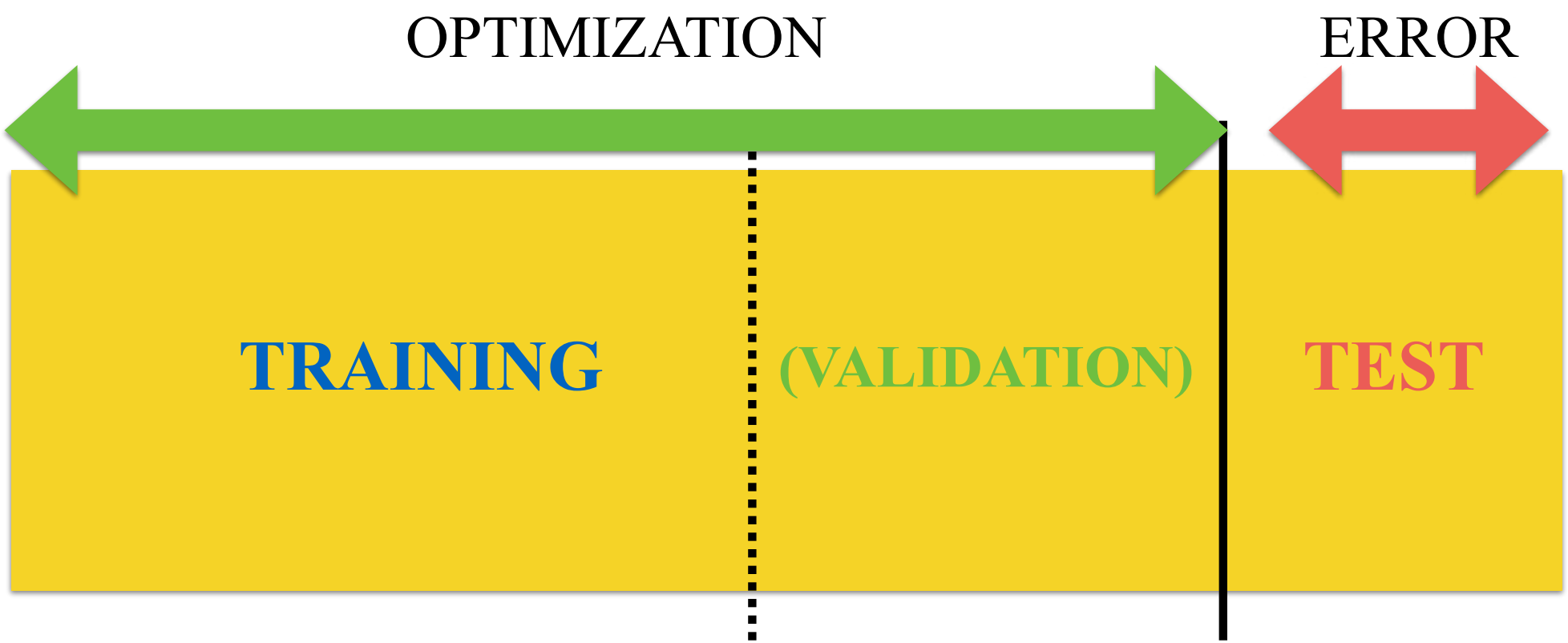


CAMELS

Ravanbakhsh+17, Brehmer+19, Ribli+19, Pan+19, Ntampaka+19, Alexander+20, Arjona+20, Coogan+20, Escamilla-Rivera+20, Hortua+20, Vama+20, Vernardos+20, Wang+20, Mao+20, Arico+20, Villaescusa_navarro+20, Singh+20, Park+21, Modi+21, Villaescusa-Navarro+21ab, Moriwaki+21, DeRose+21, Makinen+21, Villaescusa-Navaroo+22

Training deeper networks

IN PRACTICE



training set: use to train the classifier

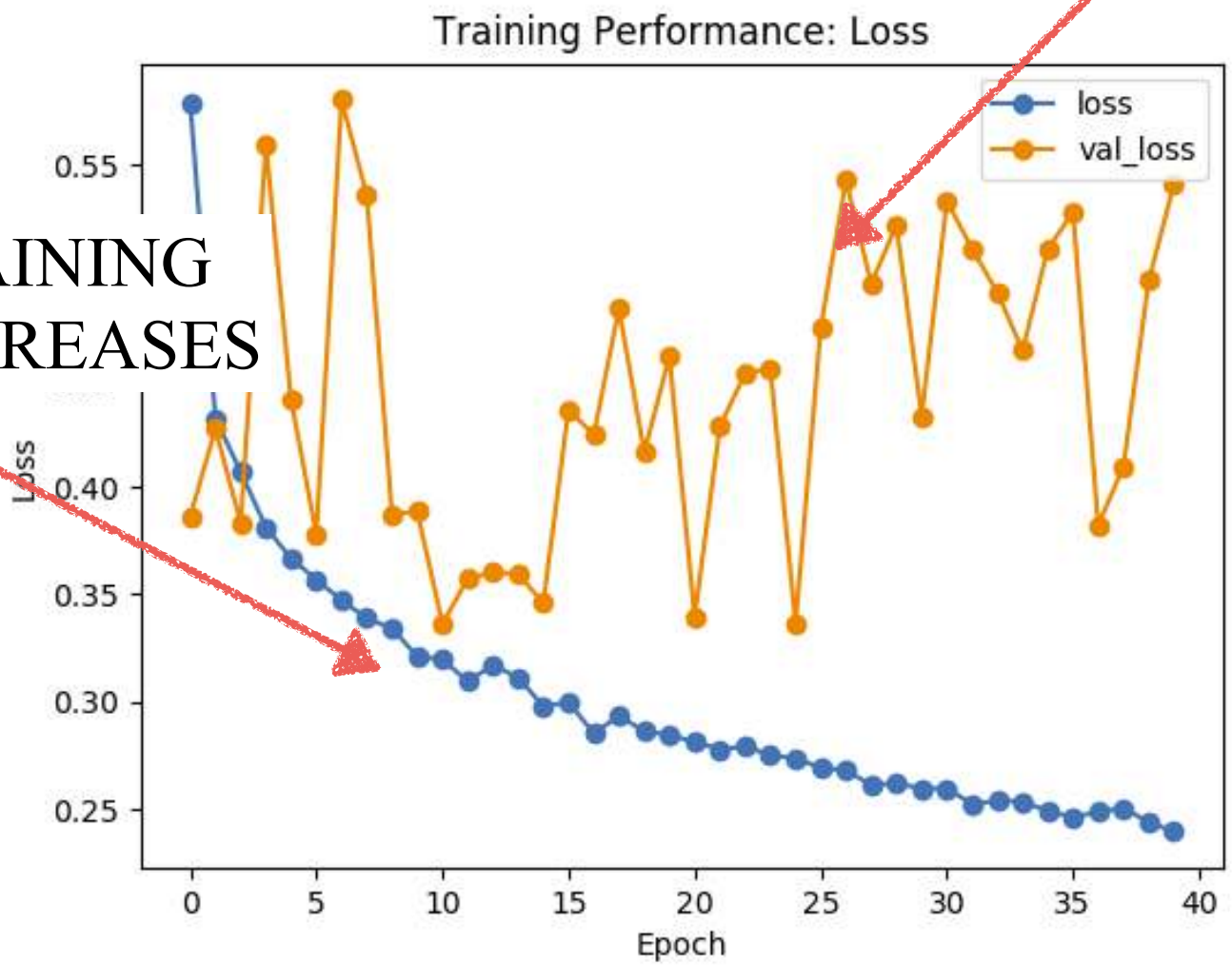
validation set: use to monitor performance in real time - check for overfitting

test set: use to train the classifier

OVER-FITTING

THE TEST STAYS CONSTANT
OR INCREASES

THE TRAINING
LOSS DECREASES

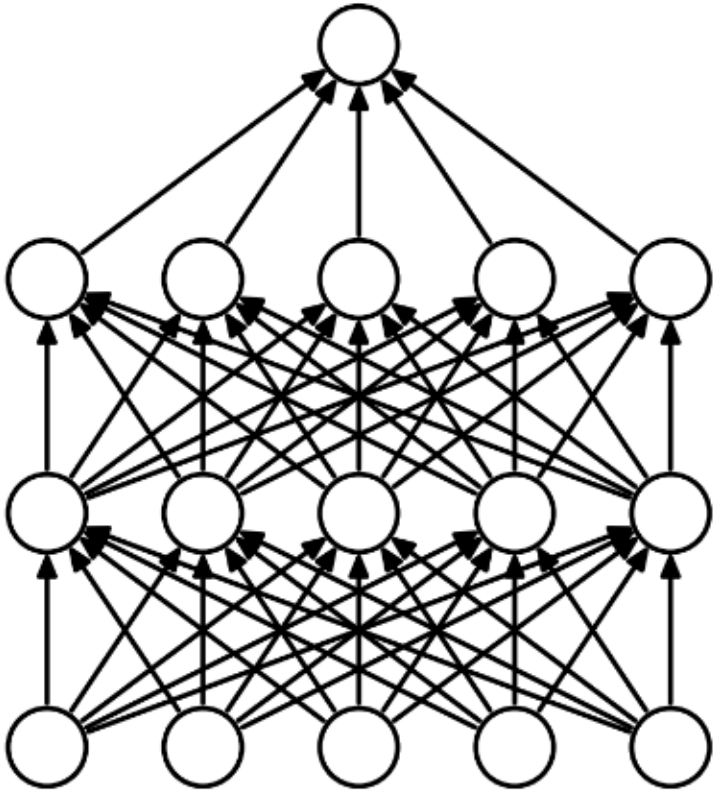


DROPOUT

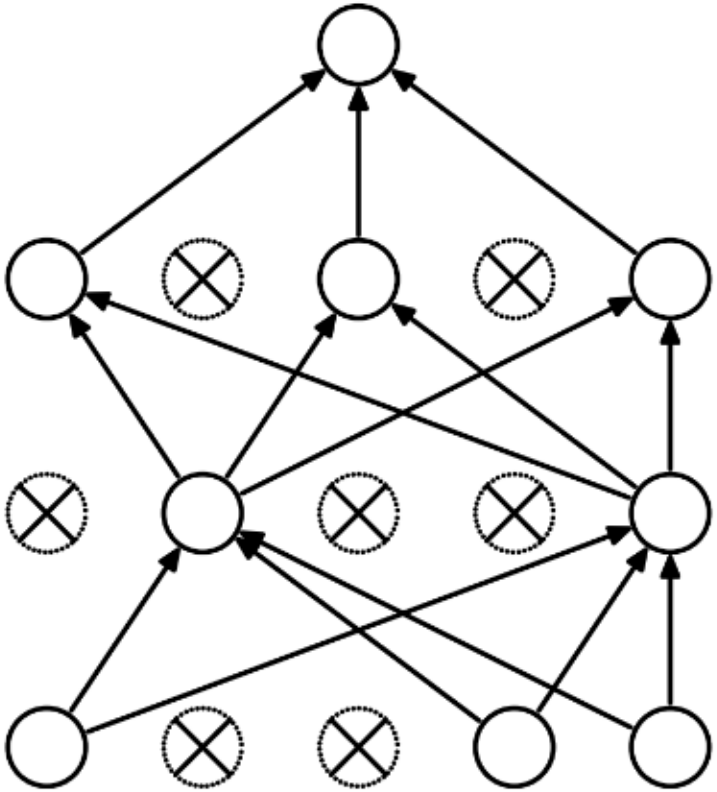
[Hinton+12]

- THE IDEA IS TO REMOVE NEURONS RANDOMLY DURING THE TRAINING

- ALL NEURONS ARE PUT BACK DURING THE TEST PHASE



(a) Standard Neural Net



(b) After applying dropout.

DROPOUT

WHY DOES IT WORK?

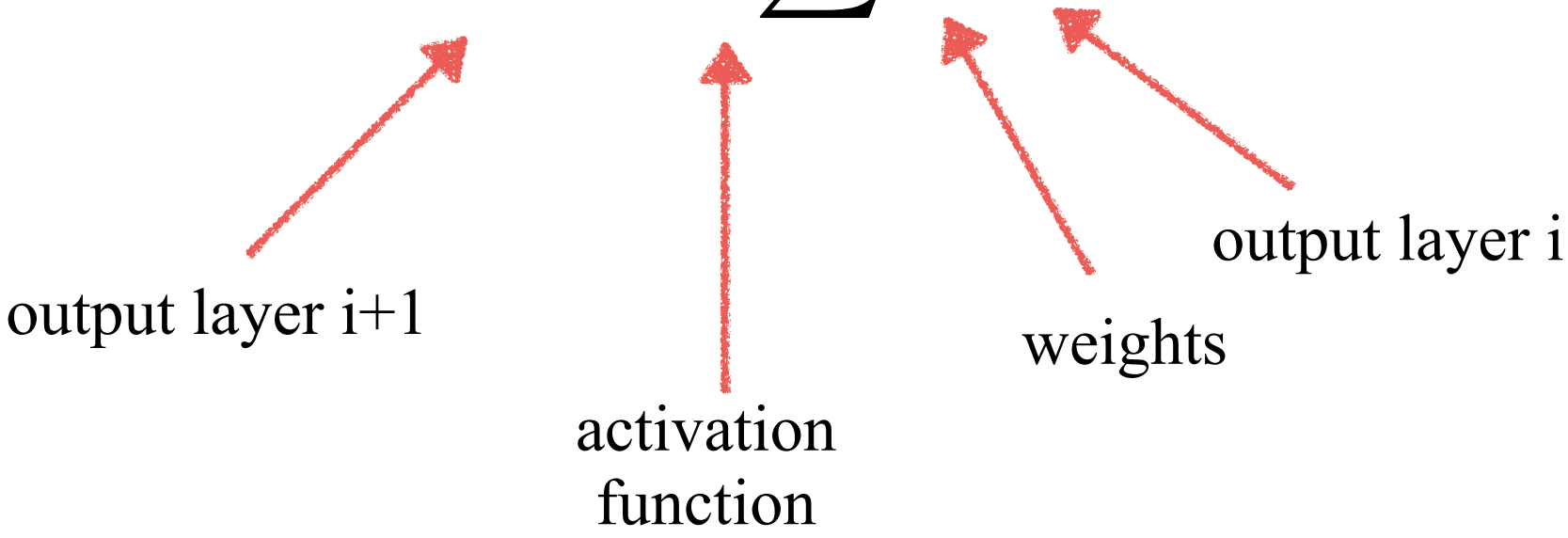
1. SINCE NEURONS ARE REMOVED RANDOMLY, IT AVOIDS CO-ADAPTATION AMONG THEMSELVES

2. DIFFERENT SETS OF NEURONS WHICH ARE SWITCHED OFF, REPRESENT A DIFFERENT ARCHITECTURE AND ALL THESE DIFFERENT ARCHITECTURES ARE TRAINED IN PARALLEL. FOR N NEURONS ATTACHED TO DROPOUT, THE NUMBER OF SUBSET ARCHITECTURES FORMED IS 2^N. SO IT AMOUNTS TO PREDICTION BEING AVERAGED OVER THESE ENSEMBLES OF MODELS.

VANISHING / EXPLODING GRADIENT PROBLEM

REMEMBER THAT:

$$y_{i+1} = \sigma[\sum w_i y_i]$$

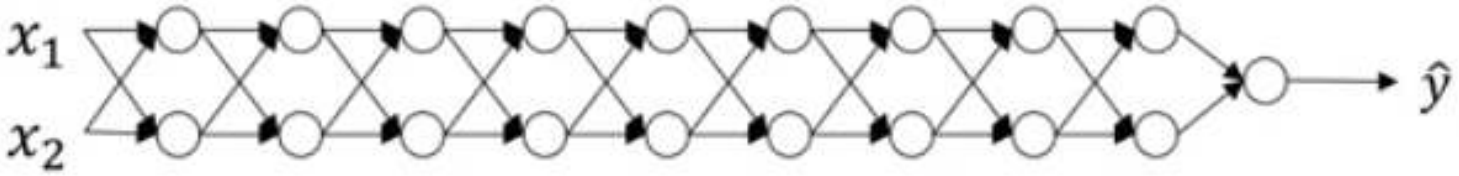


VANISHING / EXPLODING GRADIENT PROBLEM

WITH MANY LAYERS:

$$y_n = \sigma\left(...\sigma\left(...\sigma\left(\sum w_0 x\right)\right)\right)$$

VANISHING/EXPLODING GRADIENT PROBLEM



$$w_i = \begin{pmatrix} w_i^0 & 0 \\ 0 & w_i^1 \end{pmatrix} \qquad \hat{y} = x \prod_n w_i$$

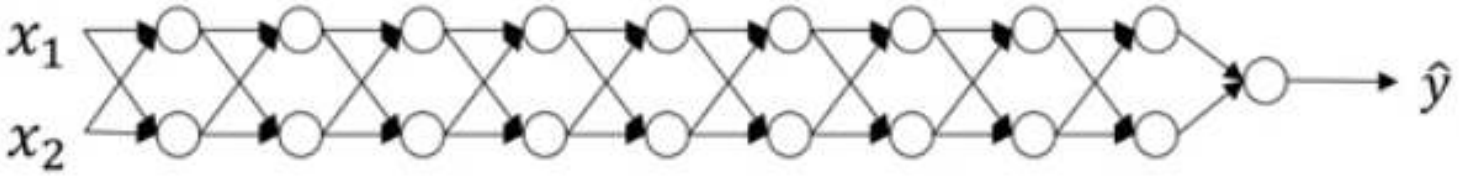
$$x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

IF WEIGHTS ARE ALL INITIALIZED TO VALUES <<1:

$$\hat{y} \to 0$$

VANISHING GRADIENT

VANISHING/EXPLODING GRADIENT PROBLEM



$$w_i = \begin{pmatrix} w_i^0 & 0 \\ 0 & w_i^1 \end{pmatrix} \qquad \hat{y} = x \prod_n w_i$$

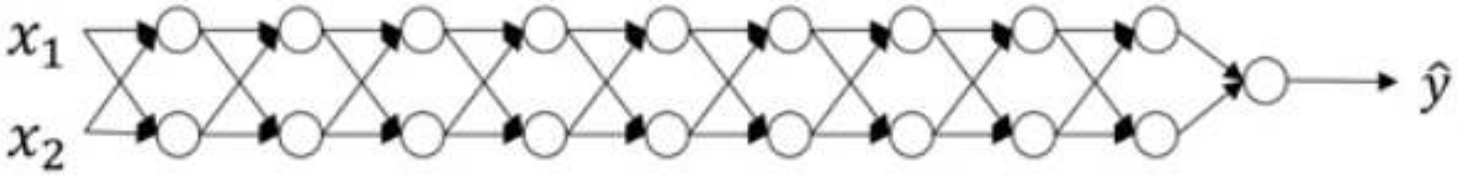
$$x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

IF WEIGHTS ARE ALL INITIALIZED TO VALUES >1:

$$\hat{y} \to \infty$$

EXPLODING GRADIENT

VANISHING/EXPLODING GRADIENT PROBLEM



$$w_i = \begin{pmatrix} w_i^0 & 0 \\ 0 & w_i^1 \end{pmatrix} \qquad \hat{y} = x \prod_n w_i$$

$$x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

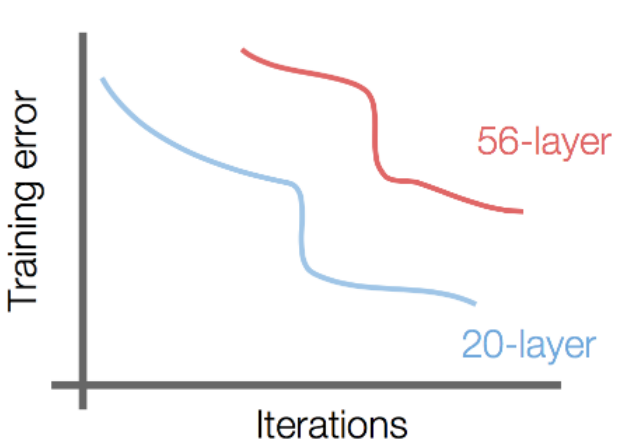
IF WEIGHTS ARE ALL INITIALIZED TO VALUES >1:

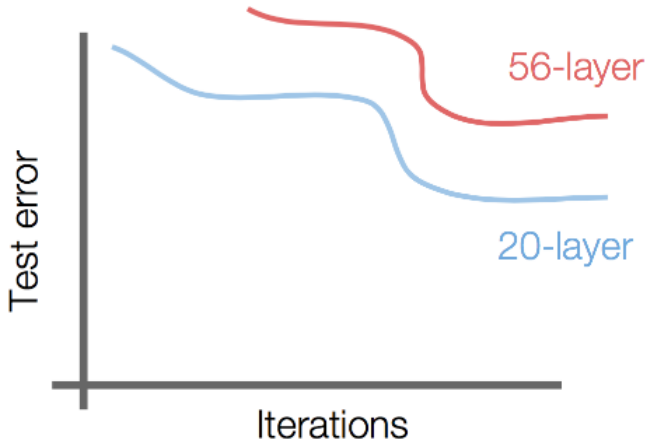
$$w_i^L \to \infty$$

EXPLODING GRADIENT

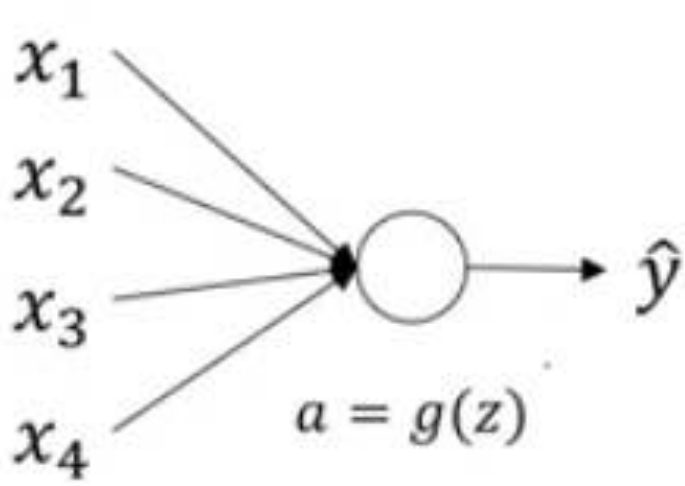
VANISHING/EXPLODING GRADIENT PROBLEM

TRAINING BECOMES UNSTABLE
VERY SLOW OR NO CONVERGENCE



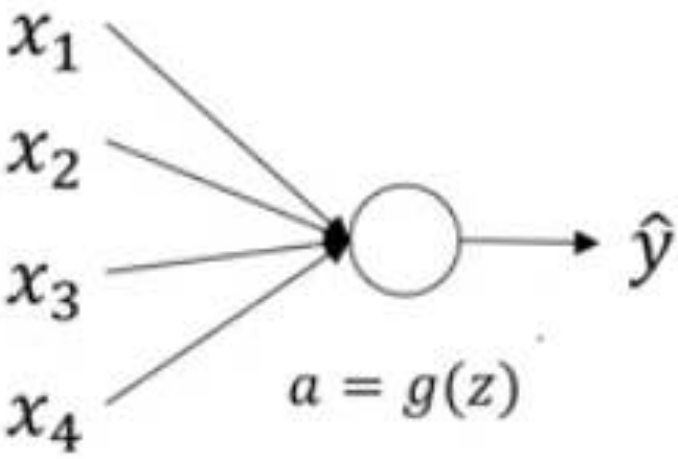


WEIGHT INITIALIZATION IS A KEY POINT…



$$z = x_1 w_1 + x_2 w_2 + \ldots + x_n w_n$$

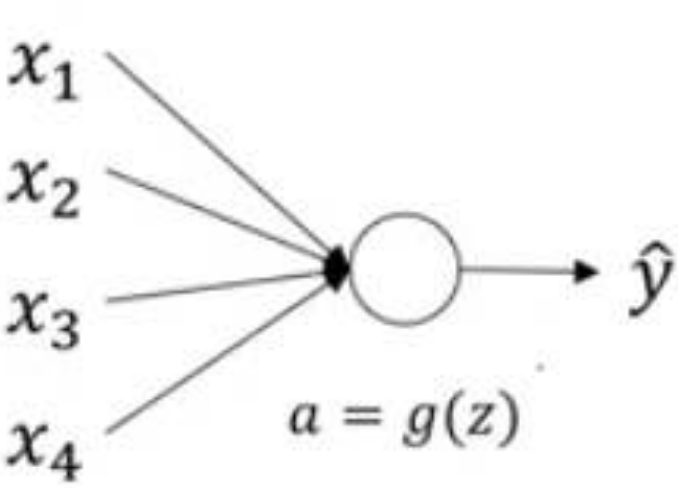
WEIGHT INITIALIZATION IS A KEY POINT…



$$z = x_1 w_1 + x_2 w_2 + \ldots + x_n w_n$$

THE LARGER n, THE SMALLER WEIGHTS SHOULD BE…

WEIGHT INITIALIZATION IS A KEY POINT…



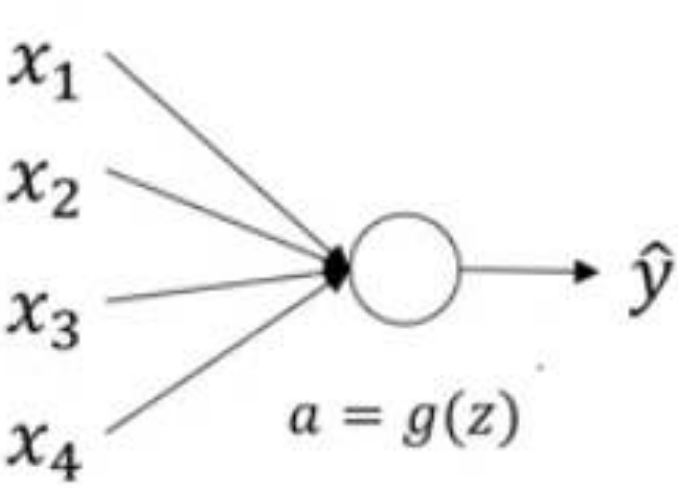
$$z = x_1 w_1 + x_2 w_2 + \ldots + x_n w_n$$

THE LARGER n, THE SMALLER WEIGHTS SHOULD BE…

variance

<u>ONE SIMPLE SOLUTION:</u> $\sigma^2(w_i) = \frac{1}{n}$ ← number of inputs

WEIGHT INITIALIZATION IS A KEY POINT…



$$z = x_1 w_1 + x_2 w_2 + \ldots + x_n w_n$$

THE LARGER n, THE SMALLER WEIGHTS SHOULD BE…

variance

<u>ONE SIMPLE SOLUTION:</u> $\sigma^2(w_i) = \frac{1}{n}$ ← number of inputs

WEIGHT INITIALIZATION IS A KEY POINT…

IMPLEMENTATION IN KERAS:

```
initialization = 'he_normal'
act = 'relu'

model = Sequential()
model.add(Convolution2D(depth, conv_size, conv_size, activation=act, border_mode='same',
name = "conv%i"%(layer_n), init=initialization, W_constraint=constraint))
```

WEIGHT INITIALIZATION IS A KEY POINT…

IMPLEMENTATION IN KERAS:

```
initialization = 'he_normal'
act = 'relu'

model = Sequential()
model.add(Convolution2D(depth, conv_size, conv_size, activation=act, border_mode='same',
name = "conv%i"%(layer_n), init=initialization, W_constraint=constraint))
```

MANY OTHER INITIALIZATIONS AVAILABLE:

`keras.initializers` → https://keras.io/initializers/

BATCH NORMALIZATION

[SZEGEDY+15]

A SOLUTION TO KEEP REASONABLE VALUES OF THE ACTIVATIONS IN DEEP NETWORKS

BATCH NORMALIZATION PREVENTS LOW OR LARGE VALUES BY RE-NORMALIZING THE VALUES BEFORE ACTIVATION FOR EVERY BATCH

INPUT

$$\hat{y}_i = \gamma \frac{y_i - E(y_i)}{\sigma(y_i)} + \beta$$

NORMALIZED INPUT

SCATTER

BATCH NORMALIZATION

[SZEGEDY+15]

BATCH NORMALIZATION SPEEDS UP AND STABILIZES TRAINING

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

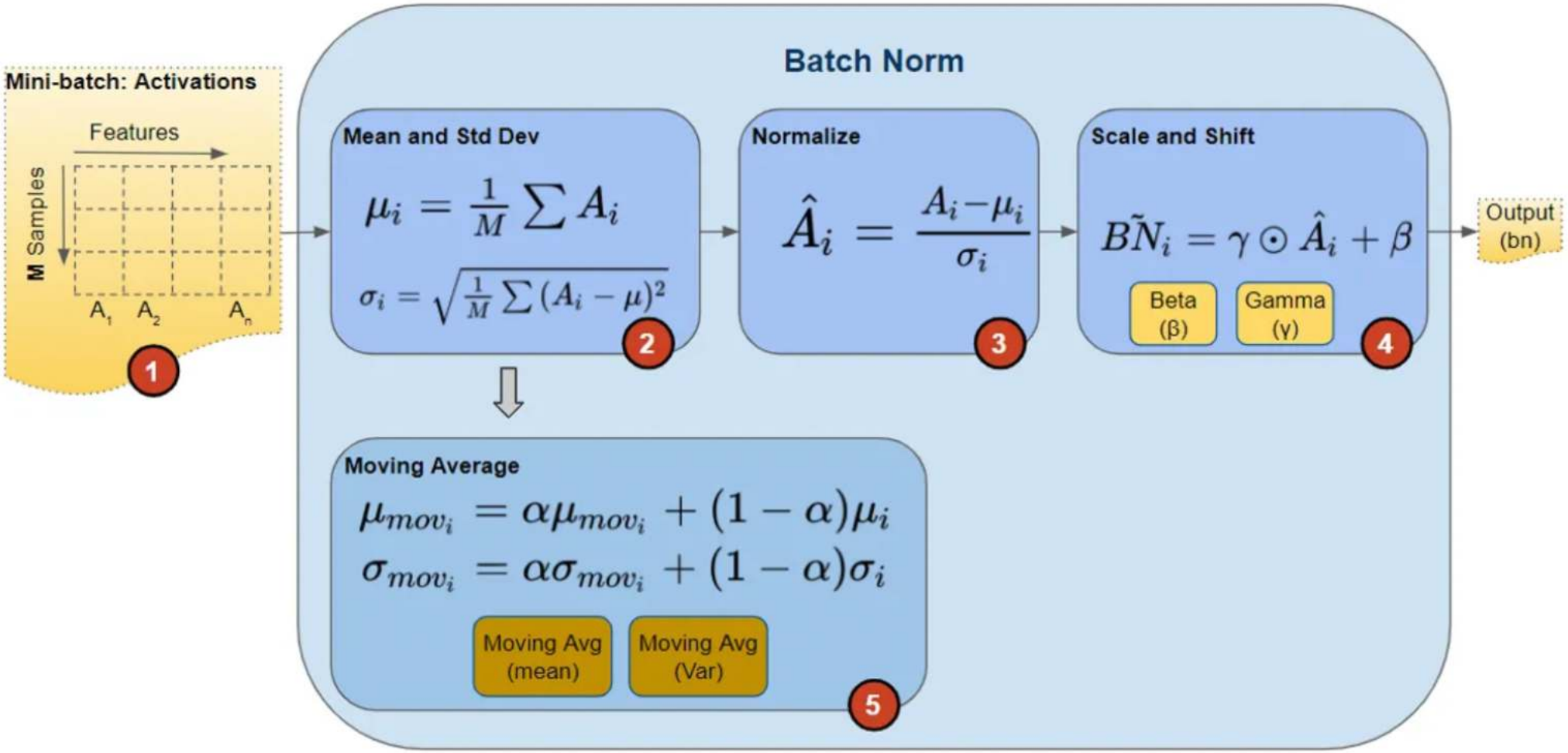
$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

BATCH NORMALIZATION

[SZEGEDY+15]

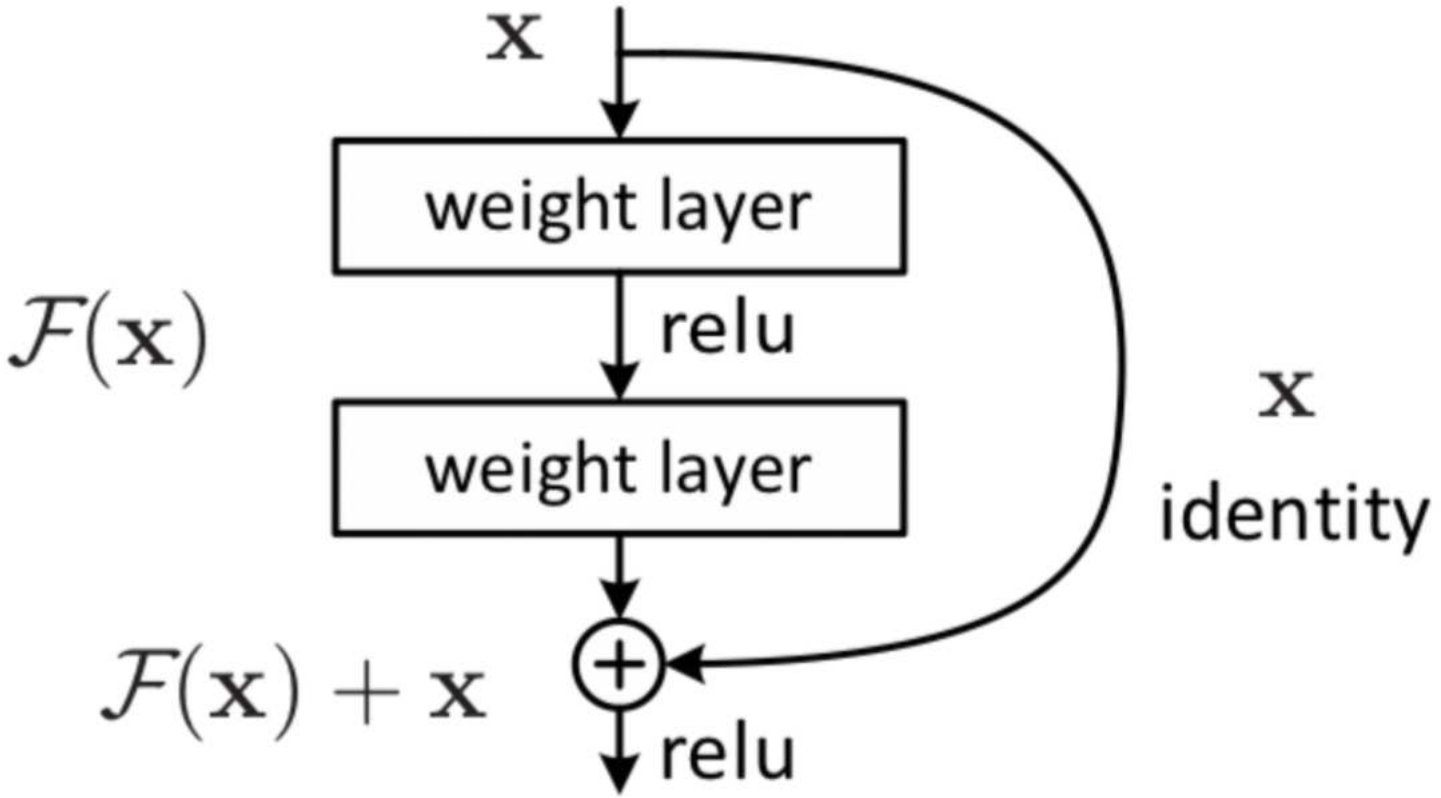


BATCH NORMALIZATION

[SZEGEDY+15]

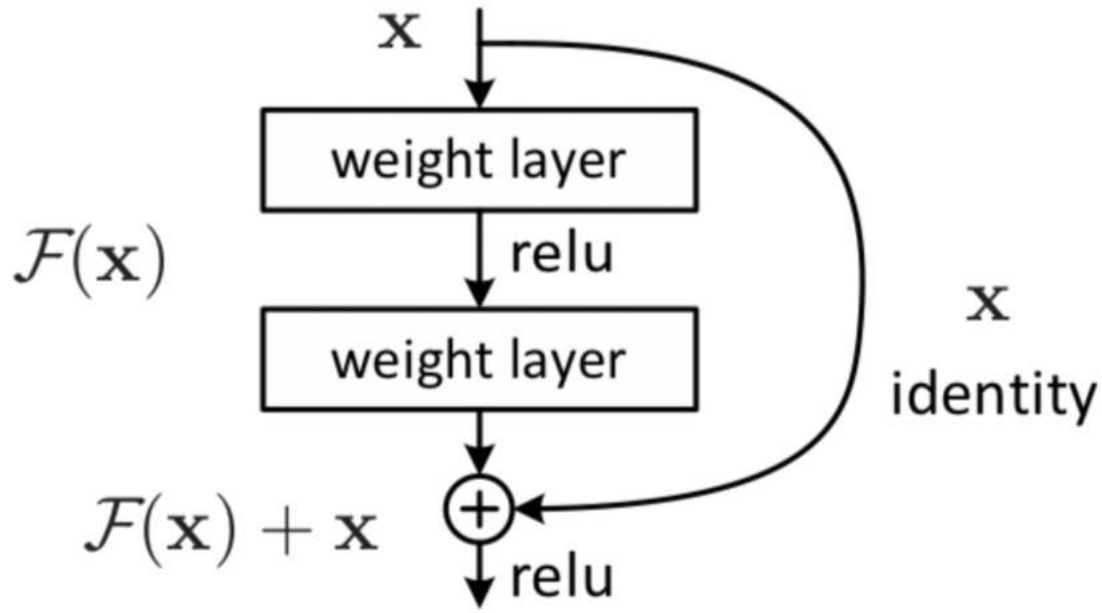
- Speeds up training
- Reduces internal covariate shift of the network
- Regularizes the network, prevents over fitting

RESIDUAL NETWORKS



Kaiming et al. 2015

RESIDUAL NETWORKS



- Adding additional / new layers would not hurt the model's performance as regularisation will skip over them if those layers were not useful.

- If the additional / new layers were useful, even with the presence of regularisation, the weights or kernels of the layers will be non-zero and model performance could increase slightly.

Kaiming et al. 2015

Dealing with a lack of labelled data

Transfer learning

THE CONVOLUTIONAL PART OF A CNN IS
A FEATURE EXTRACTOR ….

Transfer learning

THE CONVOLUTIONAL PART OF A CNN IS
A FEATURE EXTRACTOR

IN THAT RESPECT, THEY ARE VERY FLEXIBLE …

Transfer Learning)

EVEN IF OUR TRAINING SET IS NOT SO LARGE …

WE CAN USE A CNN PRE-TRAINED ON A LARGER SAMPLE

DEPENDING ON HOW SIMILAR BOTH DATASETS ARE, WE CAN:

- RECYCLE THE SAME FEATURES
- FINE-TUNING THE WEIGHTS

DATA FROM NEW SURVEY

How robust to different datasets? Do we always need a big training set?

DEEP-LEARNING BASED MACHINE

"Improved" Galaxy ZOO like classifications for for the entire sample

Transfer knowledge?

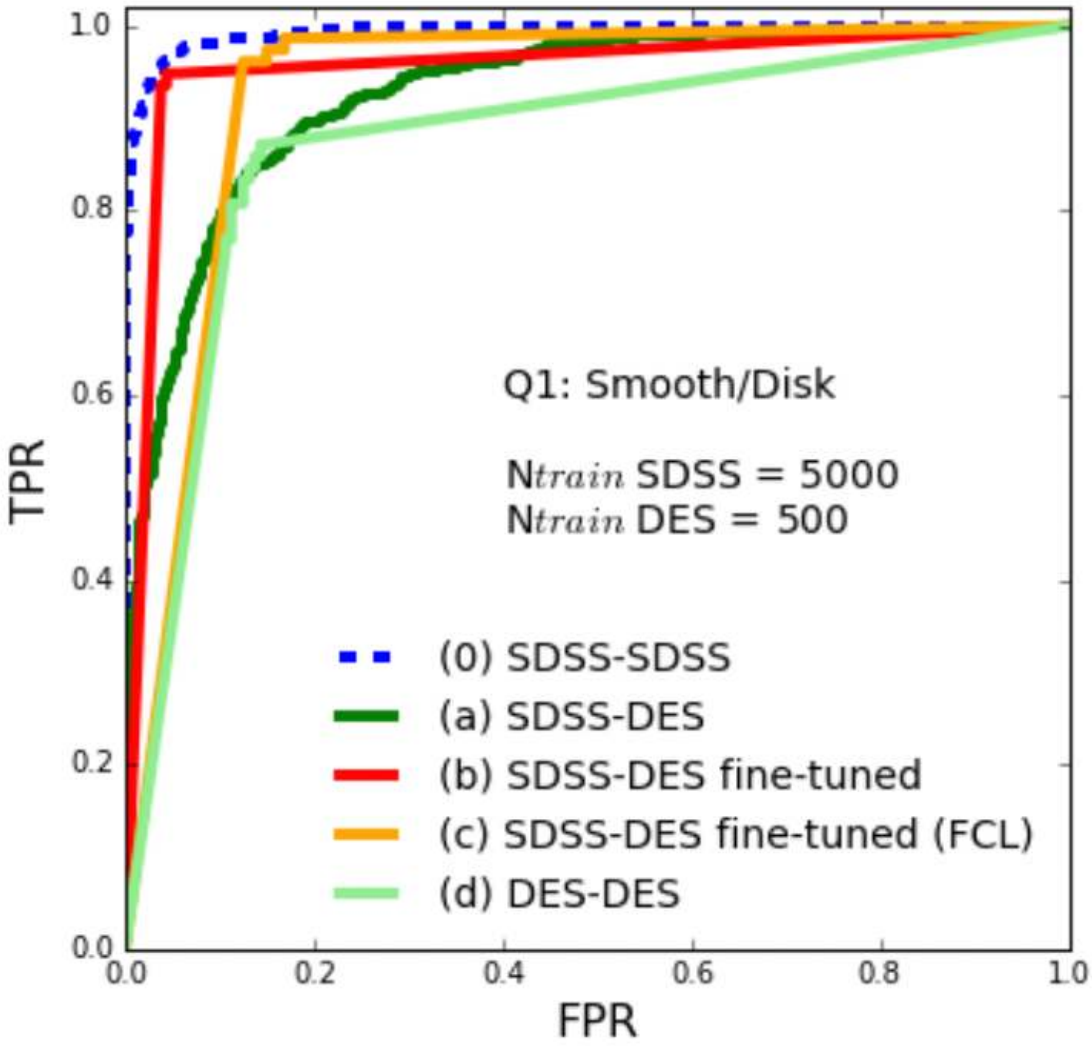
Human classifications from existing survey

Dominguez-Sanchez+18b

SDSS

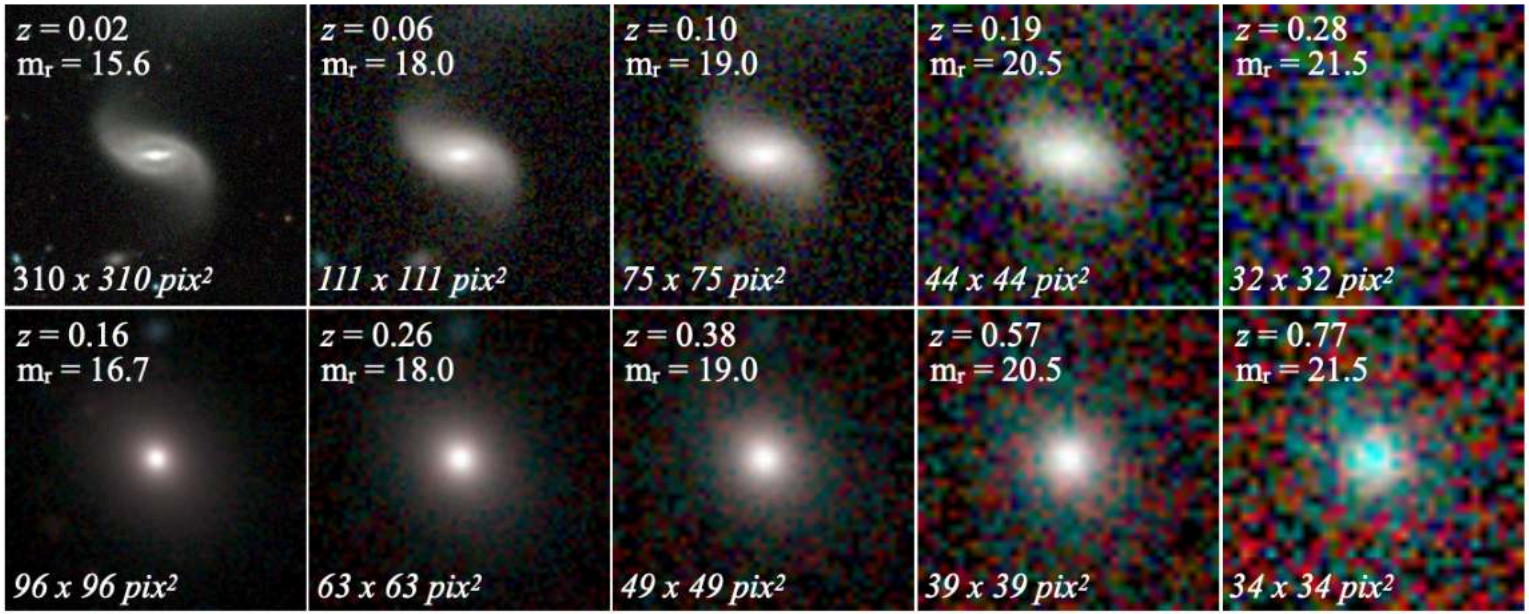
DES

Dominguez-Sanchez+18b

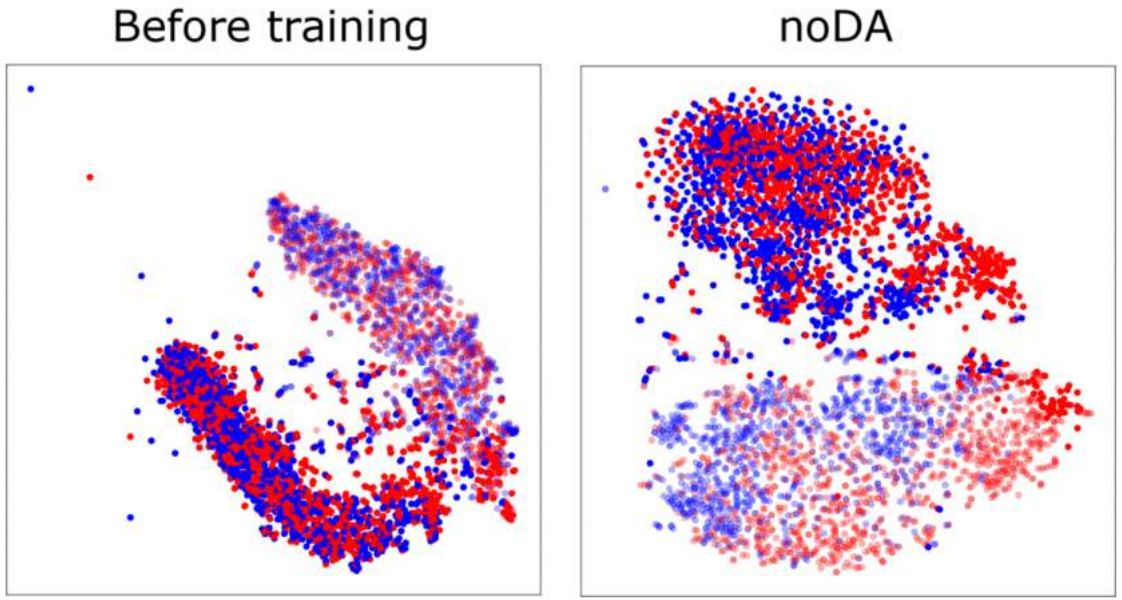


Dominguez-Sanchez+18b

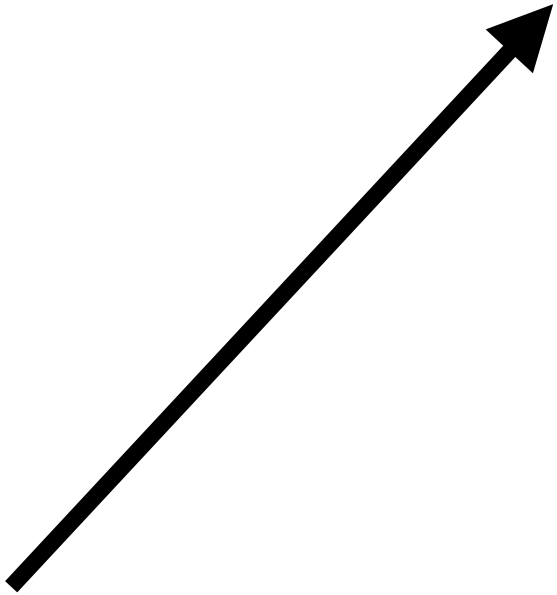
OR YOU CAN ALSO TRAIN ON SIMULATIONS



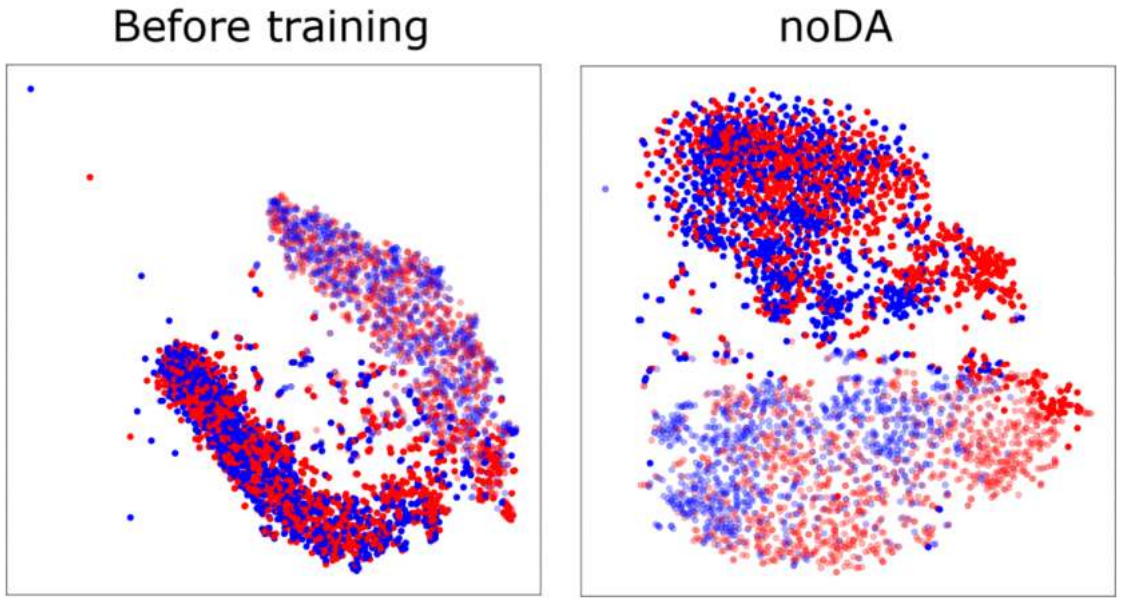
THIS IS WHERE DOMAIN KNOWLEDGE COMES INTO PLAY



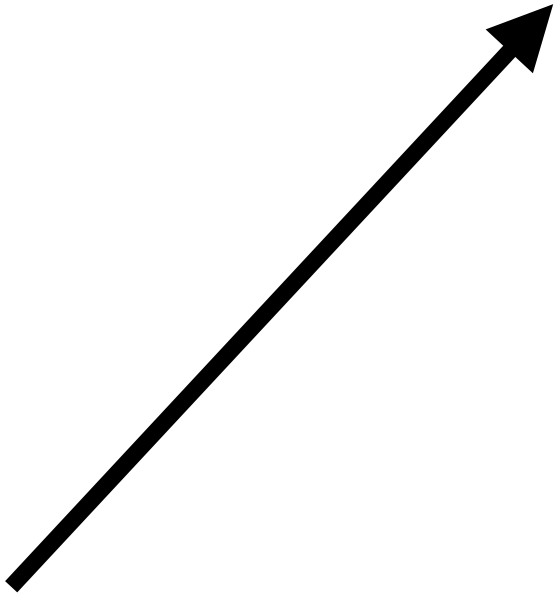
Layers	Properties	Stride	Padding	Output Shape	Parameters
Input	$3 \times 75 \times 75$[a]	-	-	(3, 75, 75)	0
Convolution (2D)	Filters: 8	1	2	(8, 75, 75)	608
	Kernel: 5×5	-	-	-	-
	Activation: ReLU	-	-	-	-
Batch Normalization	-	-	-	(8, 75, 75)	16
MaxPooling	Kernel: 2×2	2	0	(8, 37, 37)	0
Convolution (2D)	Filters: 16	1	1	(16, 37, 37)	1168
	Kernel: 3×3	-	-	-	-
	Activation: ReLU	-	-	-	-
Batch Normalization	-	-	-	(16, 37, 37)	32
MaxPooling	Kernel: 2×2	2	0	(16, 18, 18)	0
Convolution (2D)	Filters: 32	1	1	(32, 18, 18)	4640
	Kernel: 3×3	-	-	-	-
	Activation: ReLU	-	-	-	-
Batch Normalization	-	-	-	(32, 18, 18)	64
MaxPooling	Kernel: 2×2	2	0	(32, 9, 9)	0
Flatten	-	-	-	(2592)	-
Fully connected	Activation: ReLU	-	-	(64)	165952
Fully connected	Activation: ReLU	-	-	(32)	2080
Fully connected	Activation: Softmax	-	-	(2)	66



Ciprijanovic+21



Layers	Properties	Stride	Padding	Output Shape	Parameters
Input	$3 \times 75 \times 75$[a]	-	-	(3, 75, 75)	0
Convolution (2D)	Filters: 8	1	2	(8, 75, 75)	608
	Kernel: 5×5	-	-	-	-
	Activation: ReLU	-	-	-	-
Batch Normalization	-	-	-	(8, 75, 75)	16
MaxPooling	Kernel: 2×2	2	0	(8, 37, 37)	0
Convolution (2D)	Filters: 16	1	1	(16, 37, 37)	1168
	Kernel: 3×3	-	-	-	-
	Activation: ReLU	-	-	-	-
Batch Normalization	-	-	-	(16, 37, 37)	32
MaxPooling	Kernel: 2×2	2	0	(16, 18, 18)	0
Convolution (2D)	Filters: 32	1	1	(32, 18, 18)	4640
	Kernel: 3×3	-	-	-	-
	Activation: ReLU	-	-	-	-
Batch Normalization	-	-	-	(32, 18, 18)	64
MaxPooling	Kernel: 2×2	2	0	(32, 9, 9)	0
Flatten	-	-	-	(2592)	-
Fully connected	Activation: ReLU	-	-	(64)	165952
Fully connected	Activation: ReLU	-	-	(32)	2080
Fully connected	Activation: Softmax	-	-	(2)	66



Ciprijanovic+21