HOW TO I KNOW THAT I HAVE REACHED CONVERGENCE?

FOR HOW MANY EPOCHS TO I HAVE TO TRAIN?

THERE IS NO “MAGIC RULE” TO MY KNOWLEDGE.

NORMALLY A DECISION IS TAKEN BASED
ON THE MONITORING OF THE VALIDATION LOSS

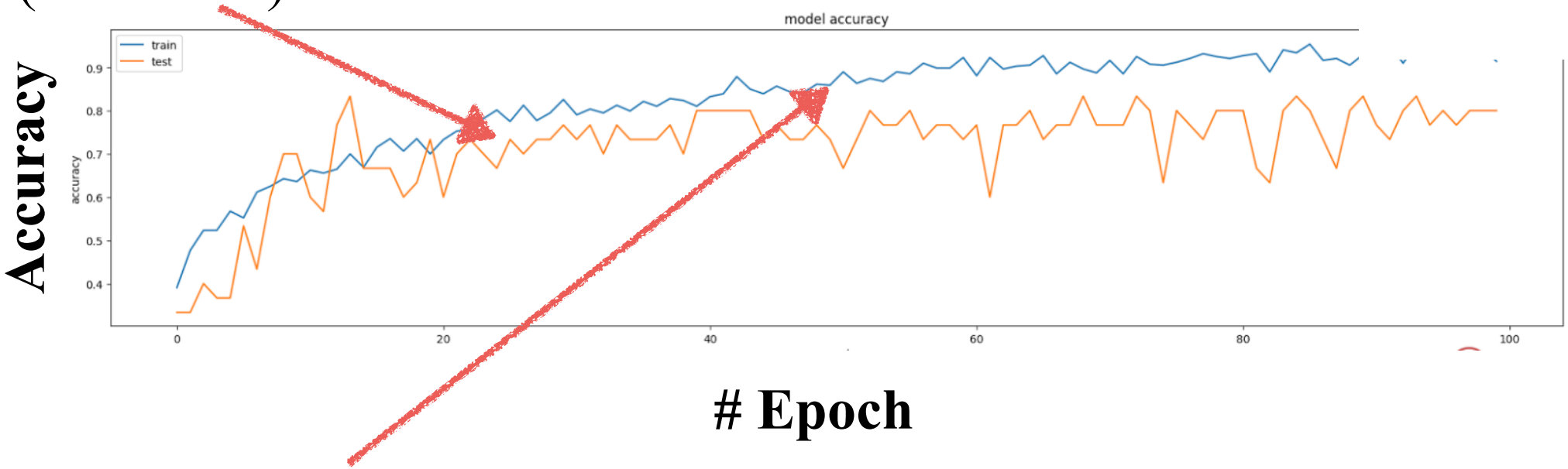
… “IF THE VALIDATION LOSS DOES NOT CHANGE MORE
THAN EPSILON OVER THE LAST N EPOCHS”…

KEEPING TRACK OF PERFORMANCE DURING TRAINING

THE LEARNING HISTORY

[MODEL ACCURACY AS A FUNCTION OF EPOCH]

(validation)



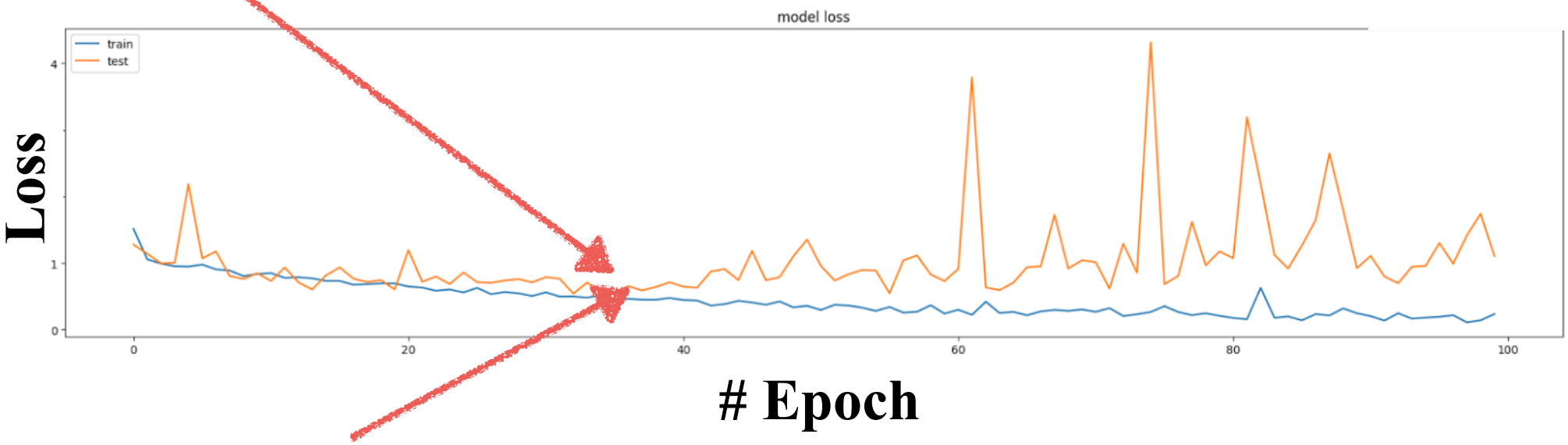
(training)

KEEPING TRACK OF PERFORMANCE DURING TRAINING

THE LEARNING HISTORY

[MODEL LOSS AS A FUNCTION OF EPOCH]

(validation)



(training)

IMPLEMENTATION IN KERAS

```
history = model.fit_generator(
            datagen.flow(X_train, Y_train, batch_size=batch_size),
            samples_per_epoch=X_train.shape[0],
            nb_epoch=nb_epoch,
            validation_data=(X_val, Y_val),
            callbacks=[earlystopping, modelcheckpoint]
        )
```

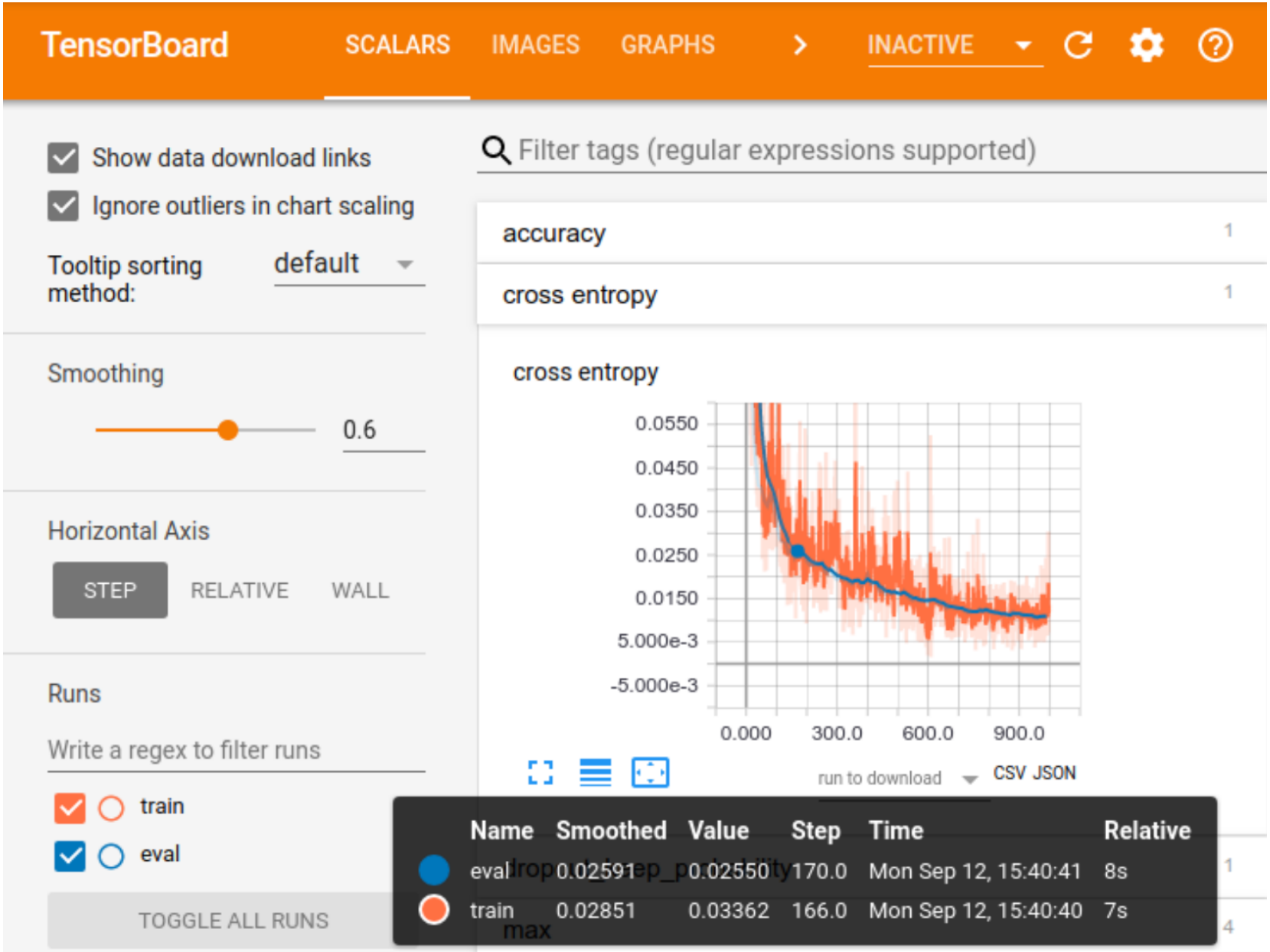
THE TRAINING RETURNS A **HISTORY** OBJECT

IMPLEMENTATION IN KERAS

THE TRAINING RETURNS A **HISTORY** OBJECT

```
from keras.models import Sequential

print(history.history.keys())
['acc', 'loss', 'val_acc', 'val_loss']

plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
```

“LIVE” LEARNING HISTORY



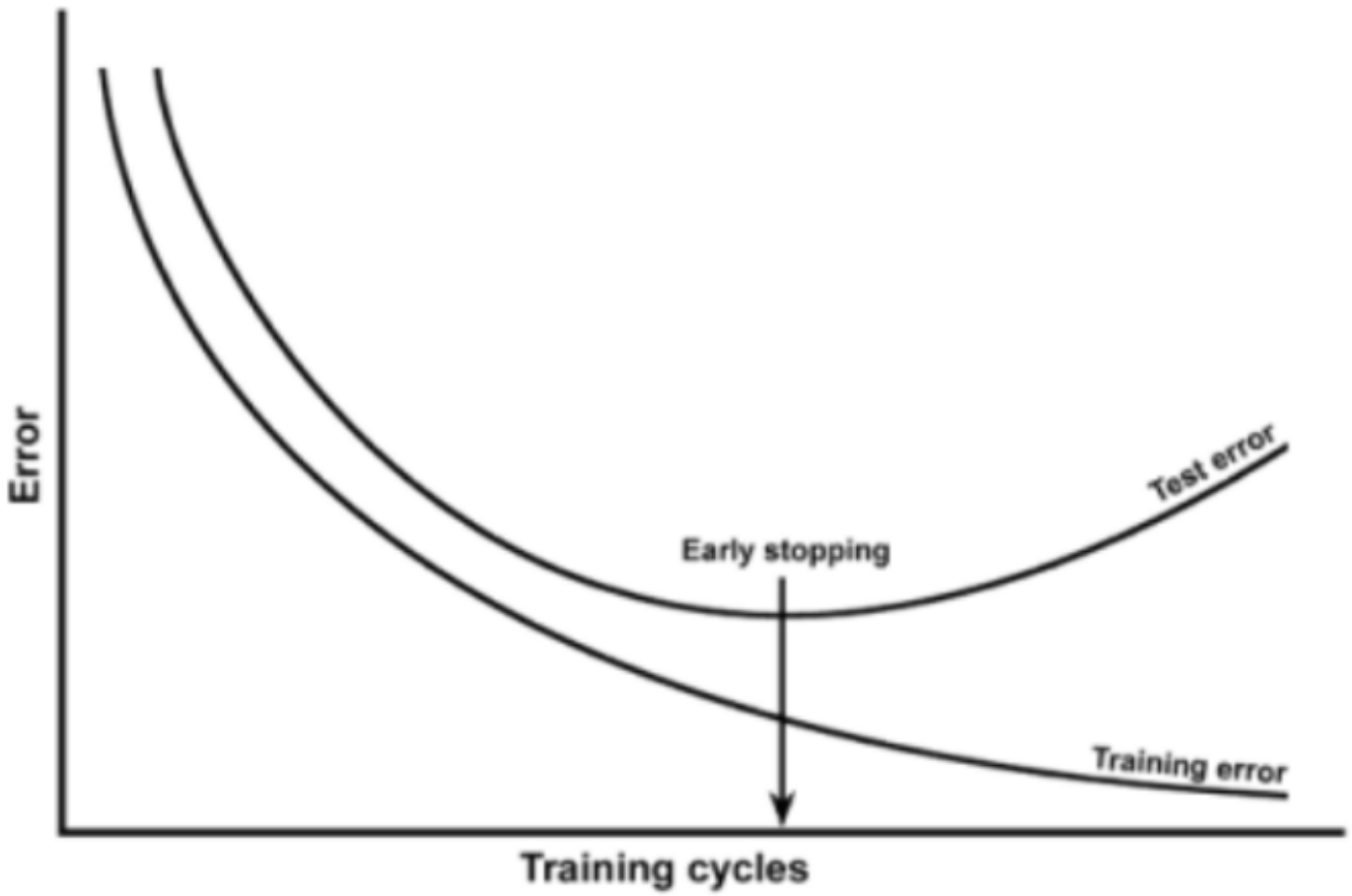
EARLY STOPPING IN KERAS

set patience (number of epochs to monitor)

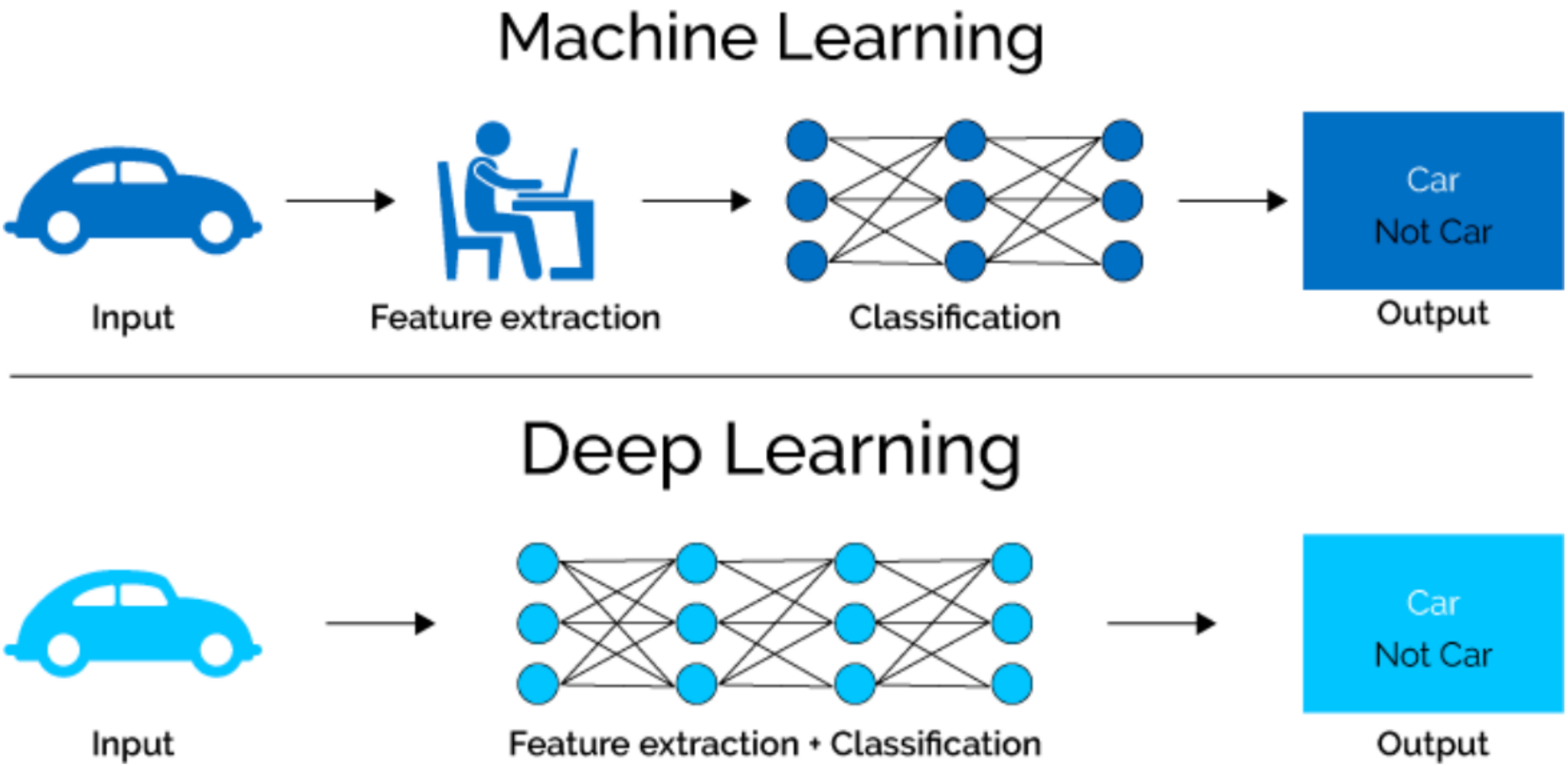
```
from keras.callbacks import EarlyStopping

patience_par=10
earlystopping = EarlyStopping( monitor='val_loss',patience =
patience_par,verbose=0,mode='auto' )
history = model.fit(X, Y, validation_split=0.33, epochs=150, batch_size=10,
verbose=0, callbacks=[earlystopping])
```

parameter to monitor [typically validation loss]

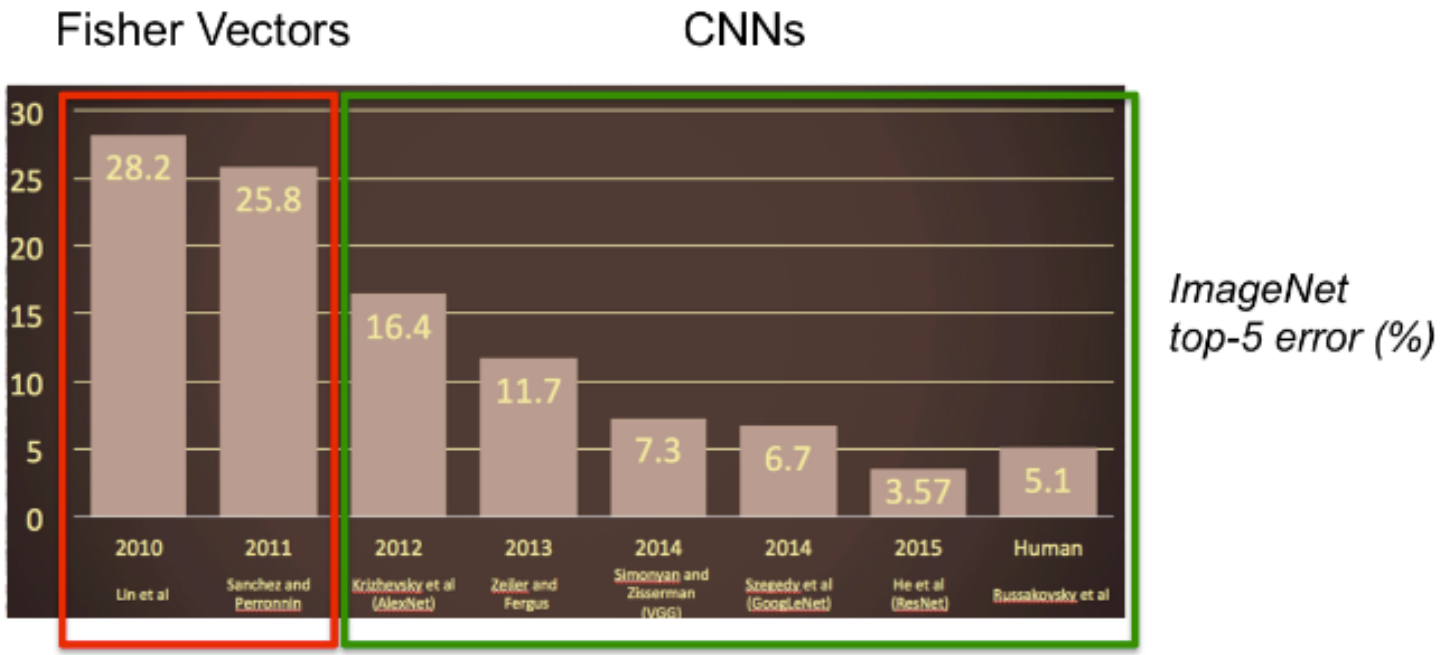
EARLY STOPPING HELPS ALSO PREVENTING OVERFITTING…



THIS IS A CHANGE OF PARADIGM!



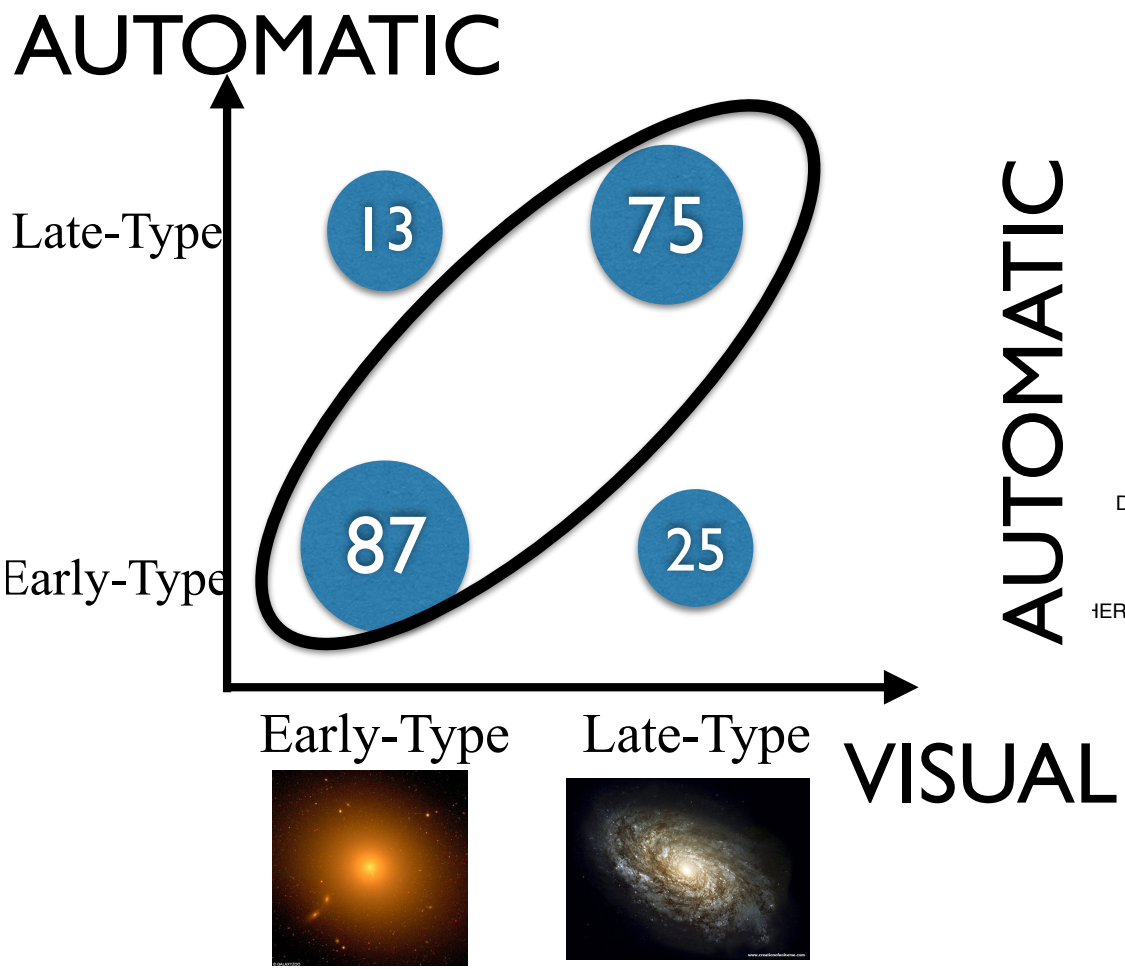
THIS IS A CHANGE OF PARADIGM!



ALSO FOR GALAXY MORPHOLOGY

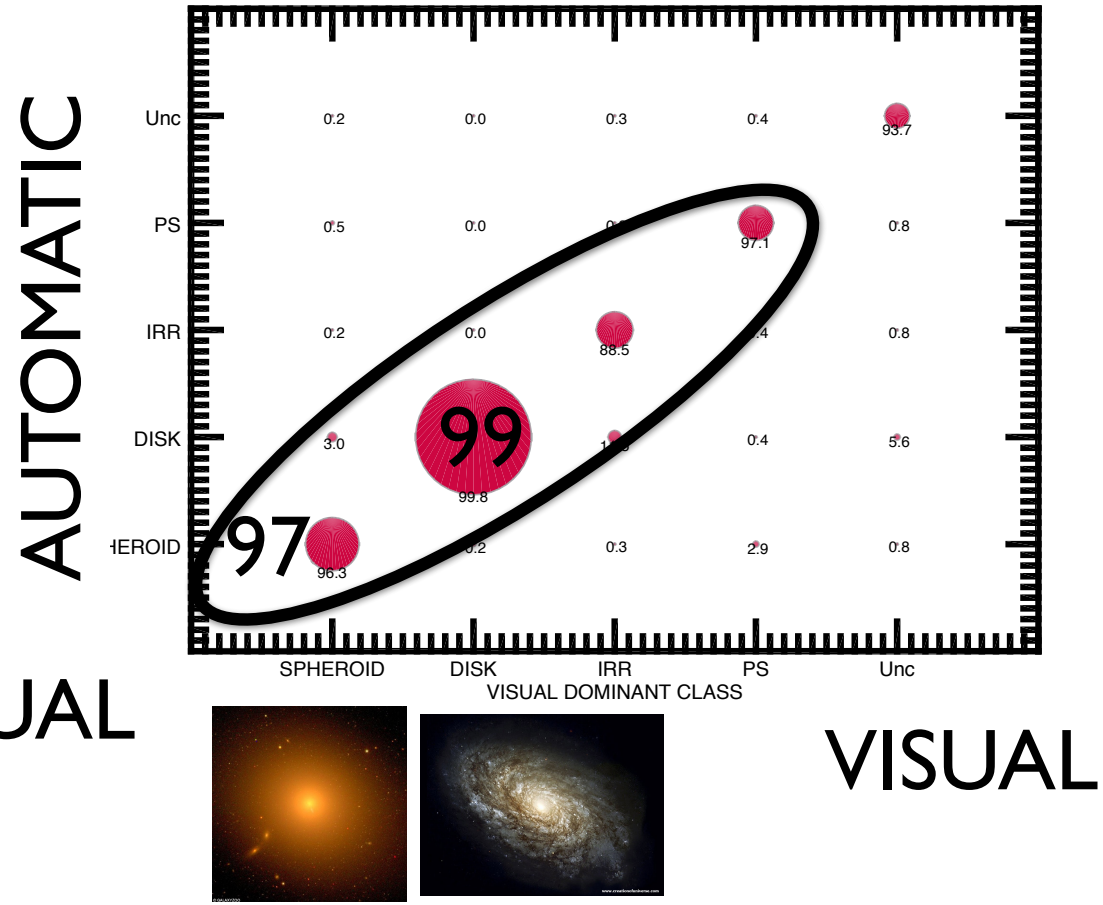
SVMs

[HUERTAS-COMPANY+14]



CNNs

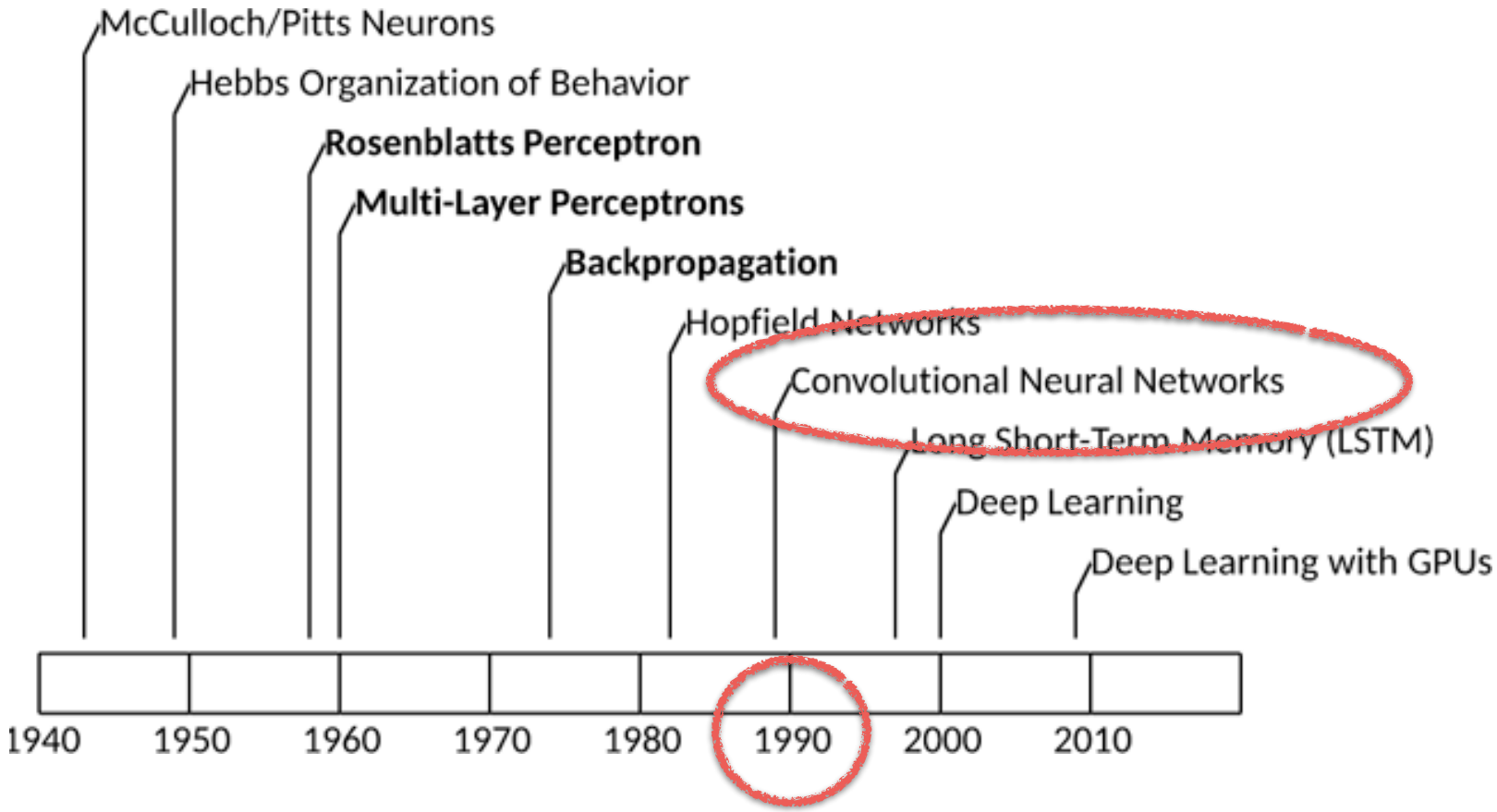
[HUERTAS-COMPANY+15b]



Name	type	AUROC	TPR_0	TPR_{10}	short description
CMU-DeepLens-ResNet-ground3	Ground-Based	0.98	0.09	0.45	CNN
CMU-DeepLens-Resnet-Voting	Ground-Based	0.98	0.02	0.10	CNN
LASTRO EPFL	Ground-Based	0.97	0.07	0.11	CNN
CAS Swinburne Melb	Ground-Based	0.96	0.02	0.03	CNN
AstrOmatic	Ground-Based	0.96	0.00	0.01	CNN
Manchester SVM	Ground-Based	0.93	0.22	0.35	SVM / Gabor
Manchester-NA2	Ground-Based	0.89	0.00	0.01	Human Inspection
ALL-star	Ground-Based	0.84	0.01	0.02	edges/gradiants and Logistic Reg.
CAST	Ground-Based	0.83	0.00	0.00	CNN / SVM
YattaLensLite	Ground-Based	0.82	0.00	0.00	SExtractor
LASTRO EPFL	Space-Based	0.93	0.00	0.08	CNN
CMU-DeepLens-ResNet	Space-Based	0.92	0.22	0.29	CNN
GAMOCLASS	Space-Based	0.92	0.07	0.36	CNN
CMU-DeepLens-Resnet-Voting	Space-Based	0.91	0.00	0.01	CNN
AstrOmatic	Space-Based	0.91	0.00	0.01	CNN
CMU-DeepLens-ResNet-aug	Space-Based	0.91	0.00	0.00	CNN
Kapteyn Resnet	Space-Based	0.82	0.00	0.00	CNN
CAST	Space-Based	0.81	0.07	0.12	CNN
Manchester1	Space-Based	0.81	0.01	0.17	Human Inspection
Manchester SVM	Space-Based	0.81	0.03	0.08	SVM / Gabor
NeuralNet2	Space-Based	0.76	0.00	0.00	CNN / wavelets
YattaLensLite	Space-Based	0.76	0.00	0.00	Arcs / SExtractor
All-now	Space-Based	0.73	0.05	0.07	edges/gradiants and Logistic Reg.
GAHEC IRAP	Space-Based	0.66	0.00	0.01	arc finder

Metcalf+18

WELL, BUT THIS IS AN "OLD" IDEA - WHY NOW?



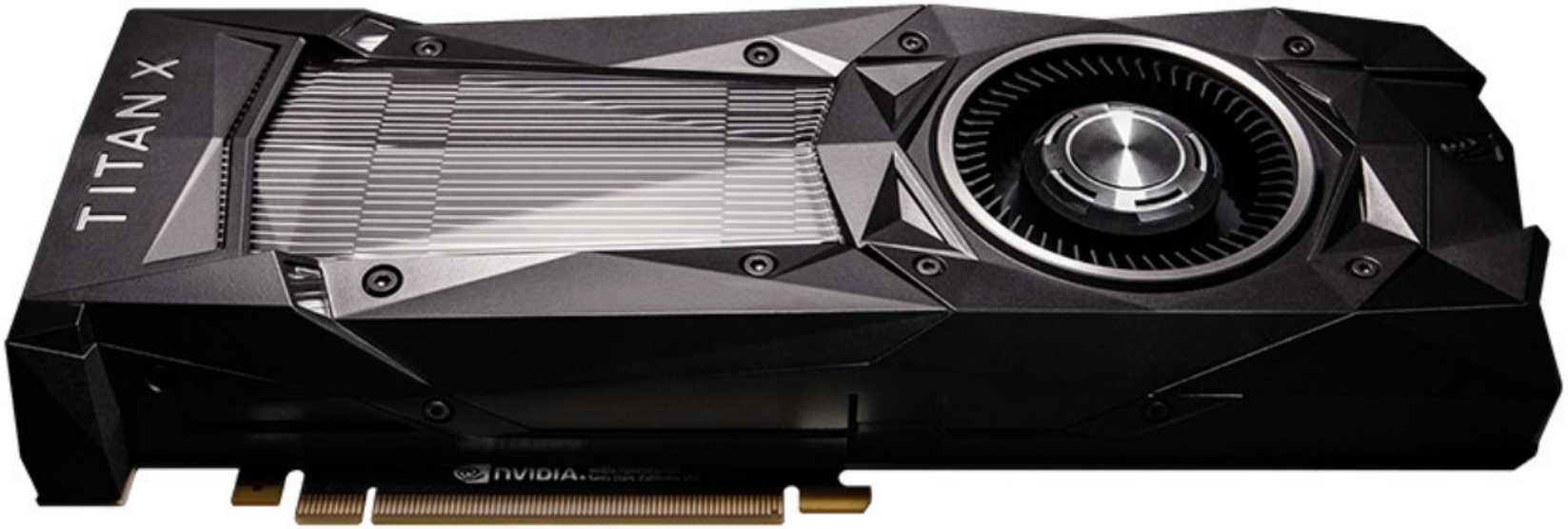
WELL, BUT THIS IS AN “OLD” IDEA - WHY NOW?

1 - MORE DATA TO TRAIN! DEEP NETWORKS HAVE A LARGE NUMBER OF PARAMETERS - THX TO SOCIAL MEDIA …

WELL, BUT THIS IS AN “OLD” IDEA - WHY NOW?

2 - GPUs - TRAINING OF THESE DEEP NETWORKS HAS REMAINED PROHIBITIVELY TIME CONSUMING WITH CPUs - THX TO VIDEO GAMES…

GPUs



NVIDA TITANX GPU

GPUs vs. CPUs

CPUs

FEWER CORES (~10x)

EACH CORE IS FASTER

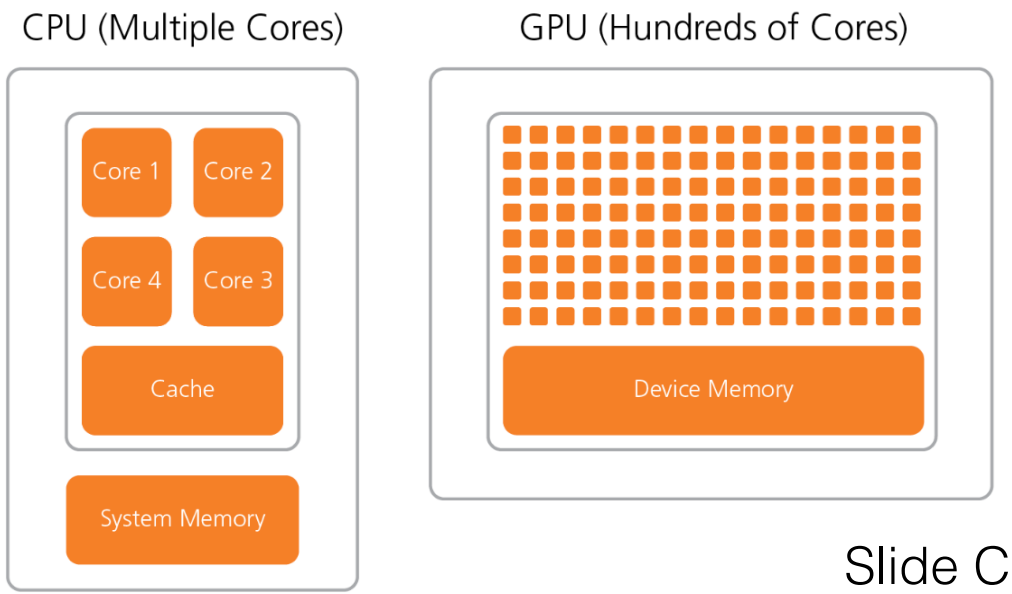
USEFUL FOR SEQUENTIAL TASKS

GPUs

MORE CORES (100x)

EACH CORE IS SLOWER

USEFUL FOR PARALLEL TASKS



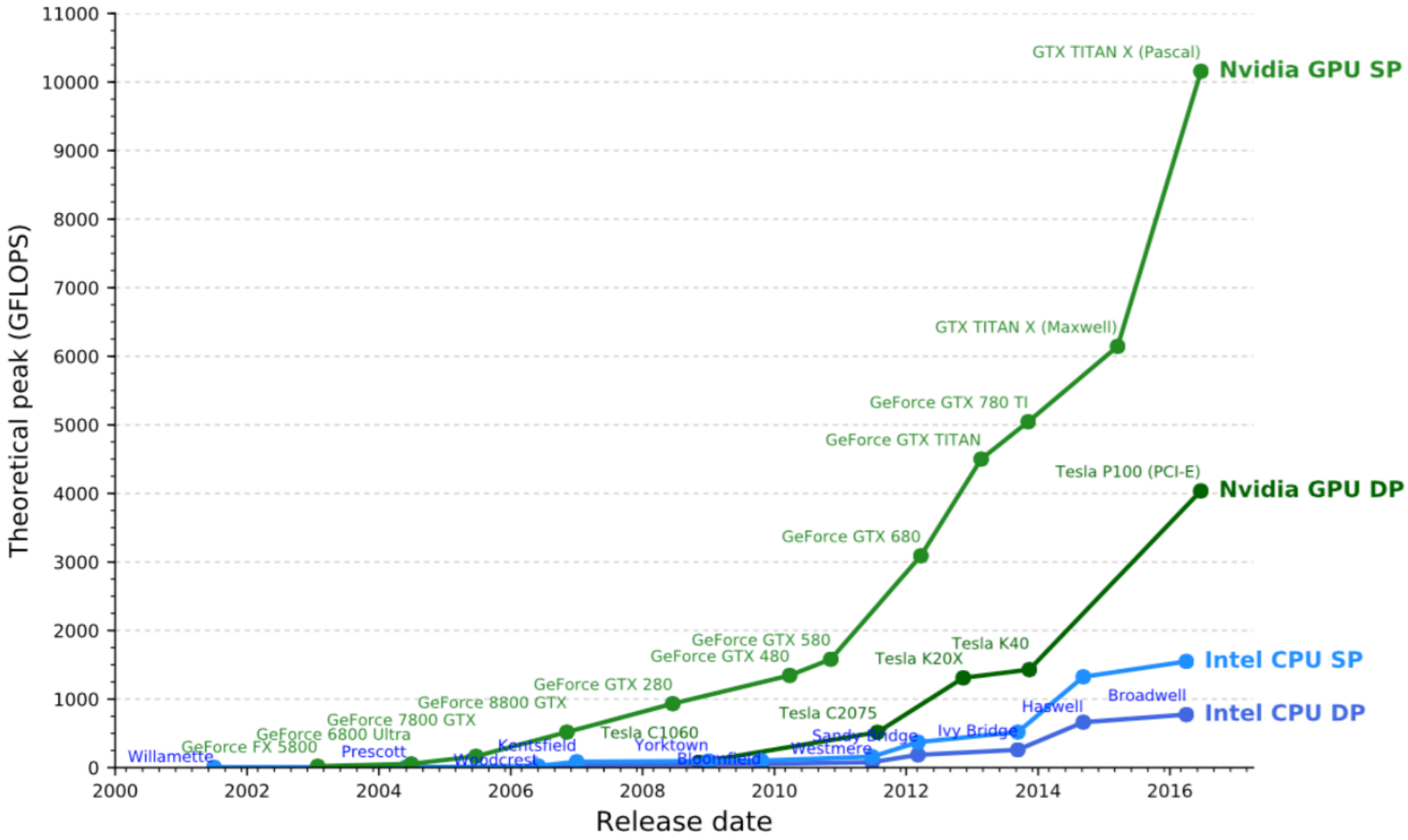
Slide Credit:

GPUs vs. CPUs

	# Cores	Clock Speed	Memory	Price
CPU (Intel Core i7-7700k)	4 (8 threads with hyperthreading)	4.4 GHz	Shared with system	$339
CPU (Intel Core i7-6950X)	10 (20 threads with hyperthreading)	3.5 GHz	Shared with system	$1723
GPU (NVIDIA Titan Xp)	3840	1.6 GHz	12 GB GDDR5X	$1200
GPU (NVIDIA GTX 1070)	1920	1.68 GHz	8 GB GDDR5	$399

Slide Credit:

GPUs vs. CPUs



Slide Credit:

GPUs vs. CPUs

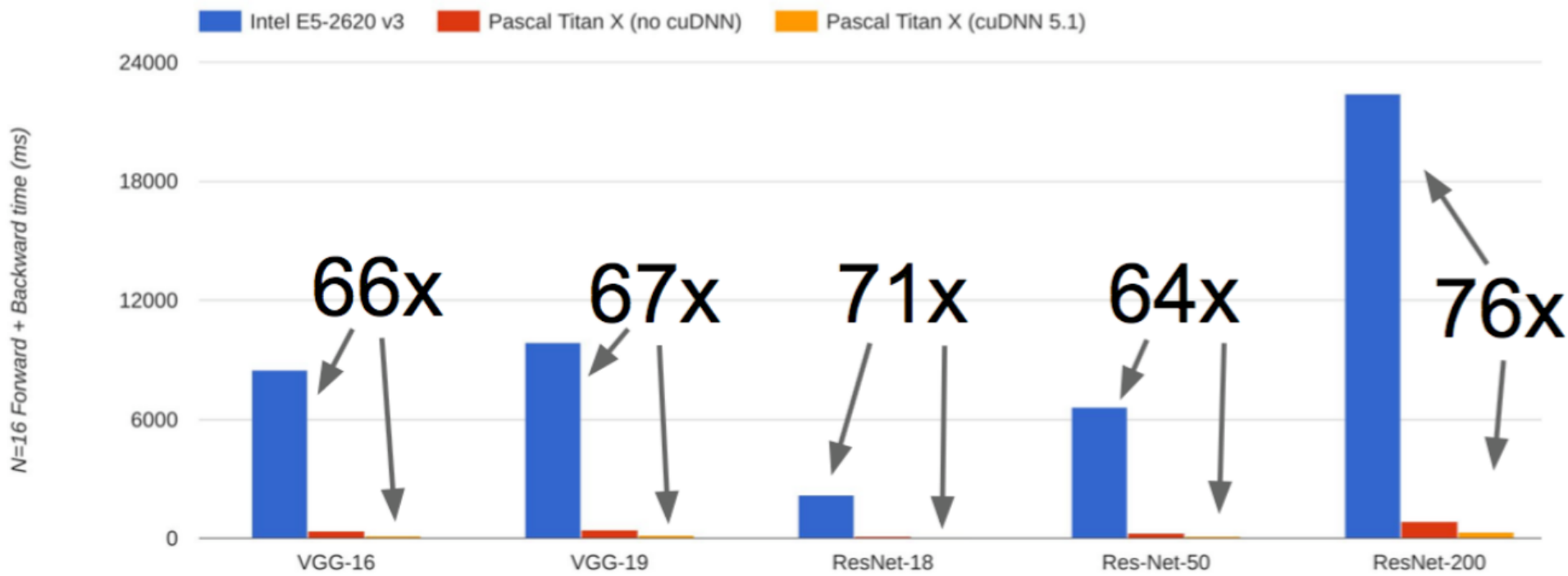
TABLE 7. COMPARATIVE EXPERIMENT RESULTS (TIME PER MINI-BATCH IN SECOND)

		Desktop CPU (Threads used)				Server CPU (Threads used)						Single GPU		
		1	2	4	8	1	2	4	8	16	32	G980	G1080	K80
FCN-S	Caffe	1.324	0.790	**0.578**	15.444	1.355	0.997	0.745	**0.573**	0.608	1.130	0.041	**0.030**	0.071
	CNTK	1.227	0.660	**0.435**	-	1.340	0.909	0.634	0.488	**0.441**	1.000	0.045	**0.033**	0.074
	TF	7.062	4.789	2.648	**1.938**	9.571	6.569	3.399	1.710	0.946	**0.630**	0.060	**0.048**	0.109
	MXNet	4.621	2.607	2.162	**1.831**	5.824	3.356	2.395	2.040	**1.945**	2.670	-	**0.106**	0.216
	Torch	1.329	0.710	**0.423**	-	1.279	1.131	0.595	0.433	**0.382**	1.034	0.040	**0.031**	0.070
AlexNet-S	Caffe	1.606	0.999	**0.719**	-	1.533	1.045	**0.797**	0.850	0.903	1.124	0.034	**0.021**	0.073
	CNTK	3.761	1.974	**1.276**	-	3.852	2.600	1.567	1.347	**1.168**	1.579	0.045	**0.032**	0.091
	TF	6.525	2.936	1.749	**1.535**	5.741	4.216	2.202	1.160	**0.701**	0.962	0.059	**0.042**	0.130
	MXNet	2.977	2.340	2.250	**2.163**	3.518	3.203	2.926	2.828	**2.827**	2.887	0.020	**0.014**	0.042
	Torch	4.645	2.429	**1.424**	-	4.336	2.468	1.543	1.248	**1.090**	1.214	0.033	**0.023**	0.070
RenNet-50	Caffe	11.554	7.671	**5.652**	-	10.643	8.600	6.723	**6.019**	6.654	8.220	-	**0.254**	0.766
	CNTK	-	-	-	-	-	-	-	-	-	-	0.240	**0.168**	0.638
	TF	23.905	16.435	10.206	**7.816**	29.960	21.846	11.512	6.294	**4.130**	4.351	0.327	**0.227**	0.702
	MXNet	48.000	46.154	44.444	**43.243**	57.831	57.143	54.545	54.545	**53.333**	55.172	0.207	**0.136**	0.449
	Torch	13.178	7.500	**4.736**	4.948	12.807	8.391	5.471	4.164	**3.683**	4.422	0.208	**0.144**	0.523
FCN-R	Caffe	2.476	1.499	**1.149**	-	2.282	1.748	1.403	1.211	1.127	**1.127**	0.025	**0.017**	0.055
	CNTK	1.845	0.970	0.661	**0.571**	1.592	0.857	0.501	0.323	**0.252**	0.280	0.025	**0.017**	0.053
	TF	2.647	1.913	1.157	**0.919**	3.410	2.541	1.297	0.661	0.361	**0.325**	0.033	**0.020**	0.063
	MXNet	1.914	1.072	0.719	**0.702**	1.609	1.065	0.731	0.534	0.451	**0.447**	0.029	**0.019**	0.060
	Torch	1.670	0.926	**0.565**	0.611	1.379	0.915	0.662	0.440	0.402	**0.366**	0.025	**0.016**	0.051
AlexNet-R	Caffe	3.558	2.587	**2.157**	2.963	4.270	3.514	3.381	**3.364**	4.139	4.930	0.041	**0.027**	0.137
	CNTK	9.956	7.263	**5.519**	6.015	9.381	6.078	4.984	**4.765**	6.256	6.199	0.045	**0.031**	0.108
	TF	4.535	3.225	1.911	**1.565**	6.124	4.229	2.200	1.396	1.036	**0.971**	**0.227**	0.317	0.385
	MXNet	13.401	12.305	12.278	**11.950**	17.994	17.128	16.764	**16.471**	17.471	17.770	0.060	**0.032**	0.122
	Torch	5.352	3.866	**3.162**	3.259	6.554	5.288	4.365	**3.940**	4.157	4.165	0.069	**0.043**	0.141
RenNet-56	Caffe	6.741	5.451	**4.989**	6.691	7.513	**6.119**	6.232	6.689	7.313	9.302	-	**0.116**	0.378
	CNTK	-	-	-	-	-	-	-	-	-	-	0.206	**0.138**	0.562
	TF	-	-	-	-	-	-	-	-	-	-	0.225	**0.152**	0.523
	MXNet	34.409	31.255	**30.069**	31.388	44.878	43.775	**42.299**	42.965	43.854	44.367	0.105	**0.074**	0.270
	Torch	5.758	3.222	**2.368**	2.475	8.691	4.965	3.040	**2.560**	2.575	2.811	0.150	**0.101**	0.301
LSTM	Caffe	-	-	-	-	-	-	-	-	-	-	-	-	-
	CNTK	0.186	0.120	**0.090**	0.118	0.211	0.139	0.117	0.114	**0.114**	0.198	0.018	**0.017**	0.043
	TF	4.662	3.385	1.935	**1.532**	6.449	4.351	2.238	1.183	0.702	**0.598**	0.133	**0.065**	0.140
	MXNet	-	-	-	-	-	-	-	-	-	-	0.089	**0.079**	0.149
	Torch	6.921	3.831	**2.682**	3.127	7.471	4.641	3.580	**3.260**	5.148	5.851	0.399	**0.324**	0.560

Note: The mini-batch sizes for FCN-S, AlexNet-S, ResNet-50, FCN-R, AlexNet-R, ResNet-56 and LSTM are 64, 16, 16, 1024, 1024, 128 and 128 respectively.

GPUs vs. CPUs

More benchmarks available here.



GPUs vs. CPUs

More benchmarks available here.

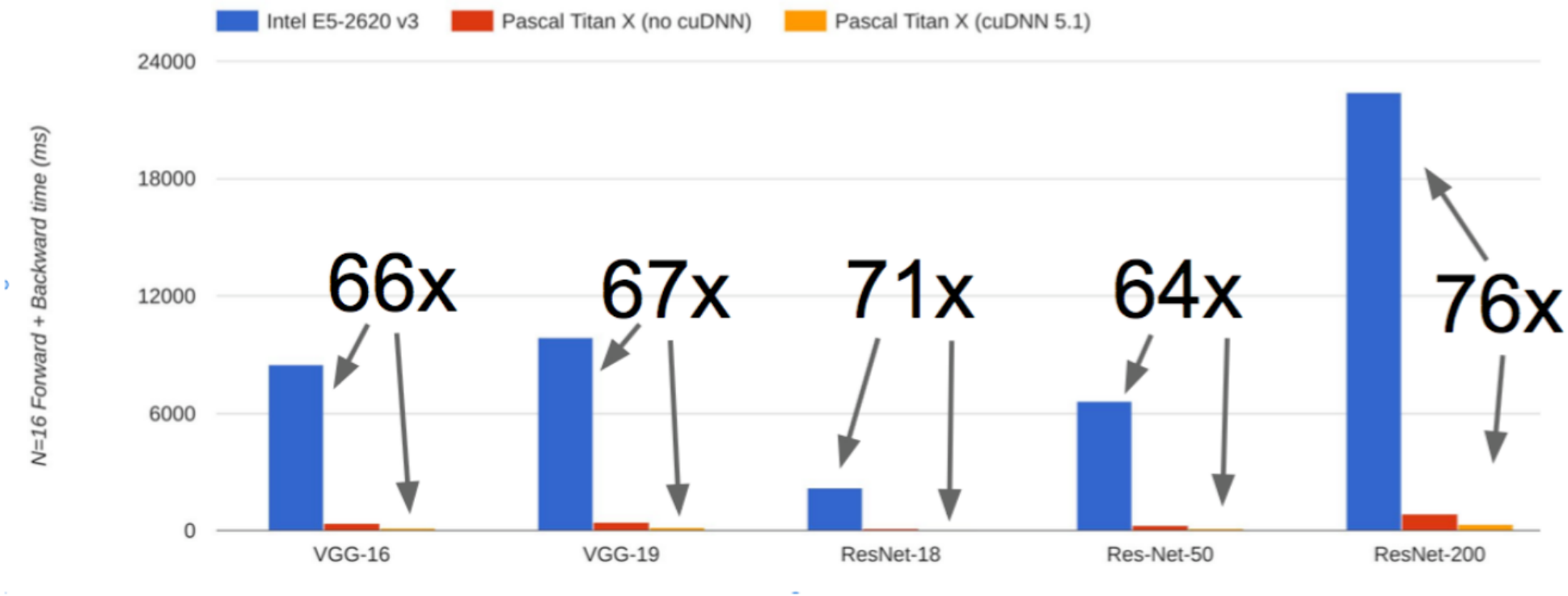


Figure credit: J. Johnson

GPUs vs. CPUs

More benchmarks available here.

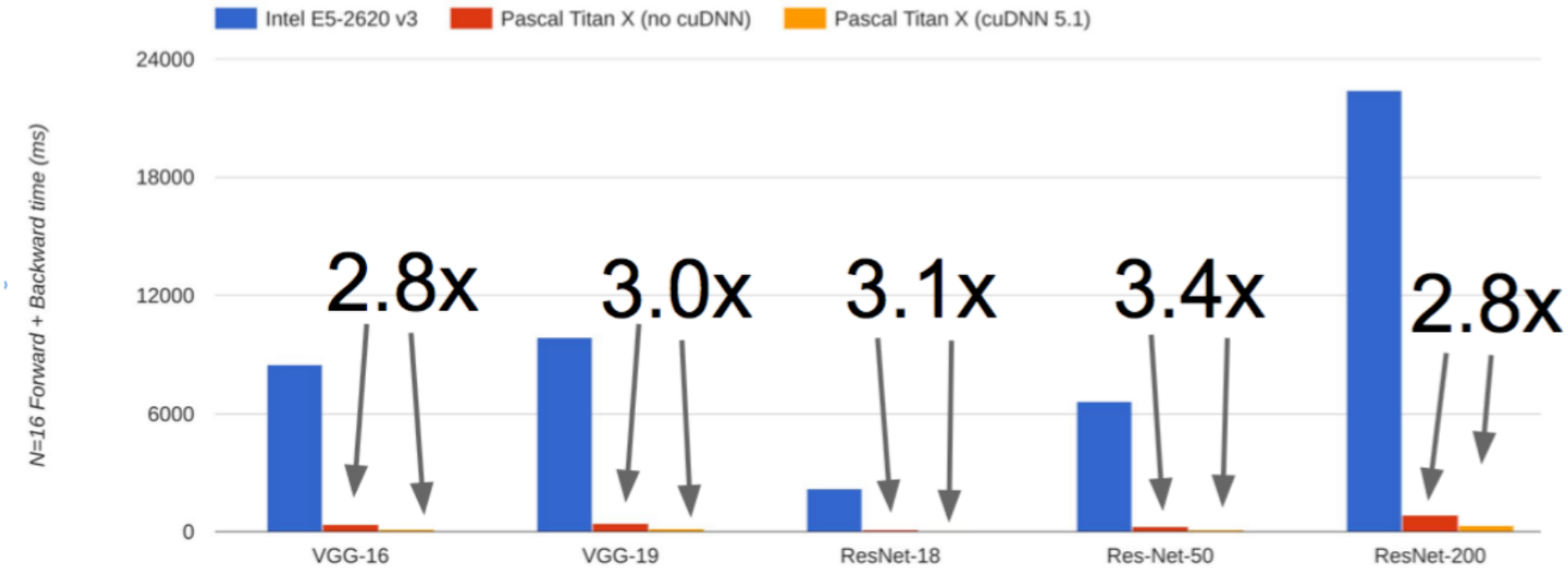
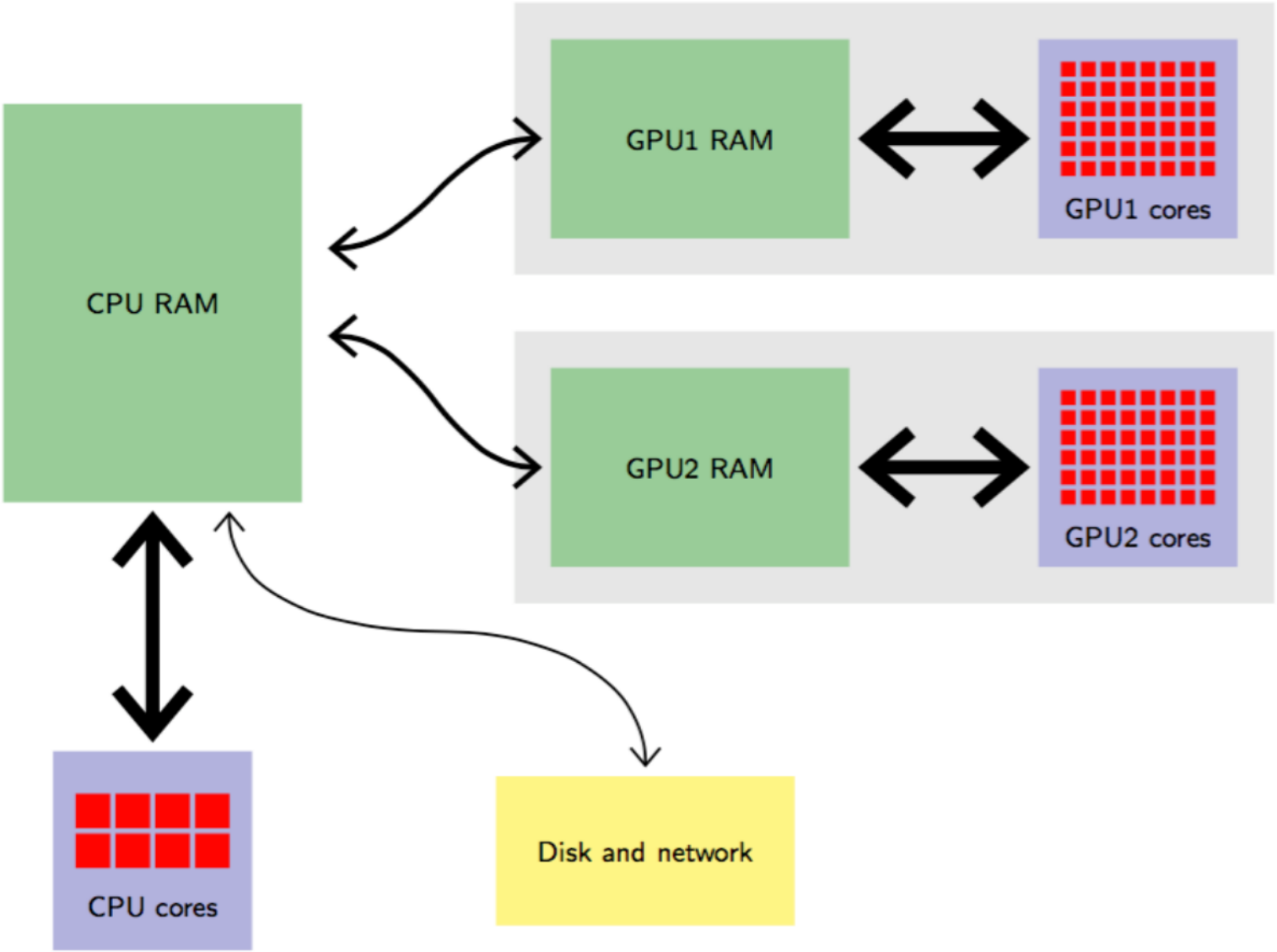


Figure credit: J. Johnson

System configuration



GPUs for deep learning

NVIDIA GPUs ARE PROGRAMMED THROUGH CUDA [Compute Unified Device Architecture]

ANOTHER ALTERNATIVE IS OPENCL, SUPPORTED BY SEVERAL MANUFACTURES, LESS INVESTMENT [Way less used]

CuDNN IS A LIBRARY FOR SPECIFIC DEEP LEARNING COMPUTATIONS ON NVIDIA GPUs

USING GPUs IN KERAS/ TENSORFLOW

TRANSPARENT USAGE - SAME CODE FOR CPU or GPU

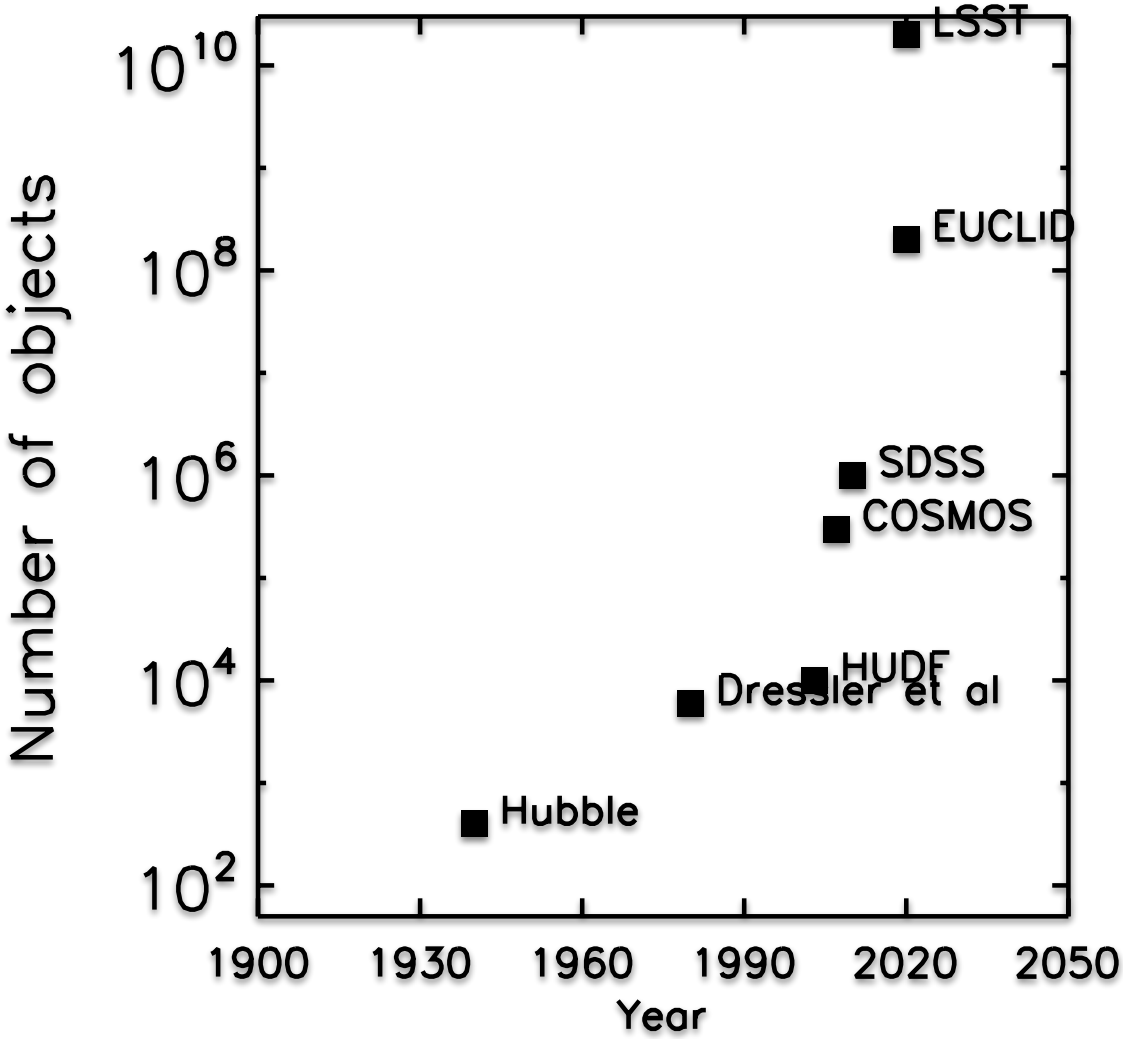
THE PRICE TO PAY?

1. LARGE NUMBER OF PARAMETERS IMPLIES LARGE DATASETS TO TRAIN

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

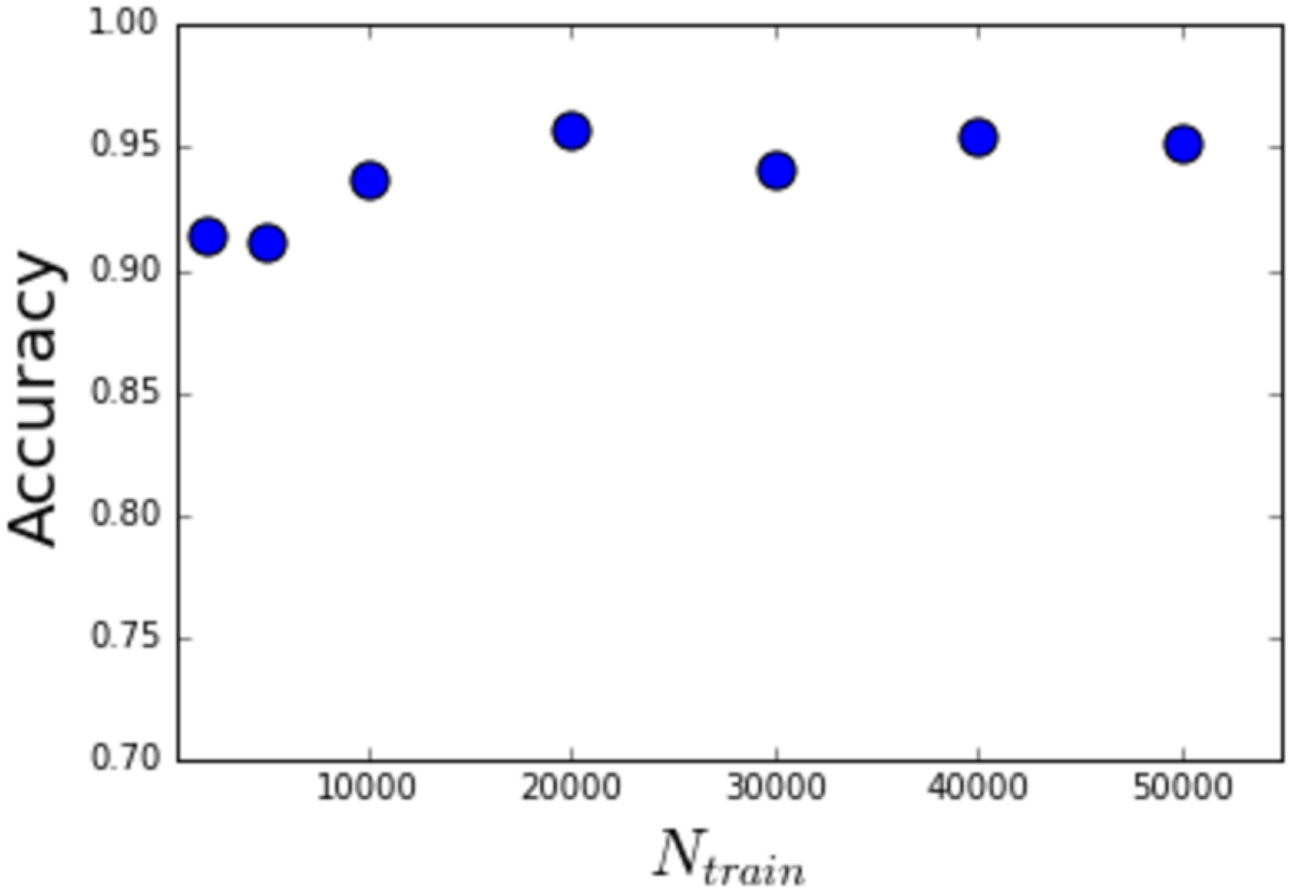
HOW LARGE DOES MY DATASET NEED TO BE?

DOES THAT MEAN THAT IF I DO NOT HAVE MILLIONS OF LABELLED EXAMPLES ALL THIS IS USELESS?



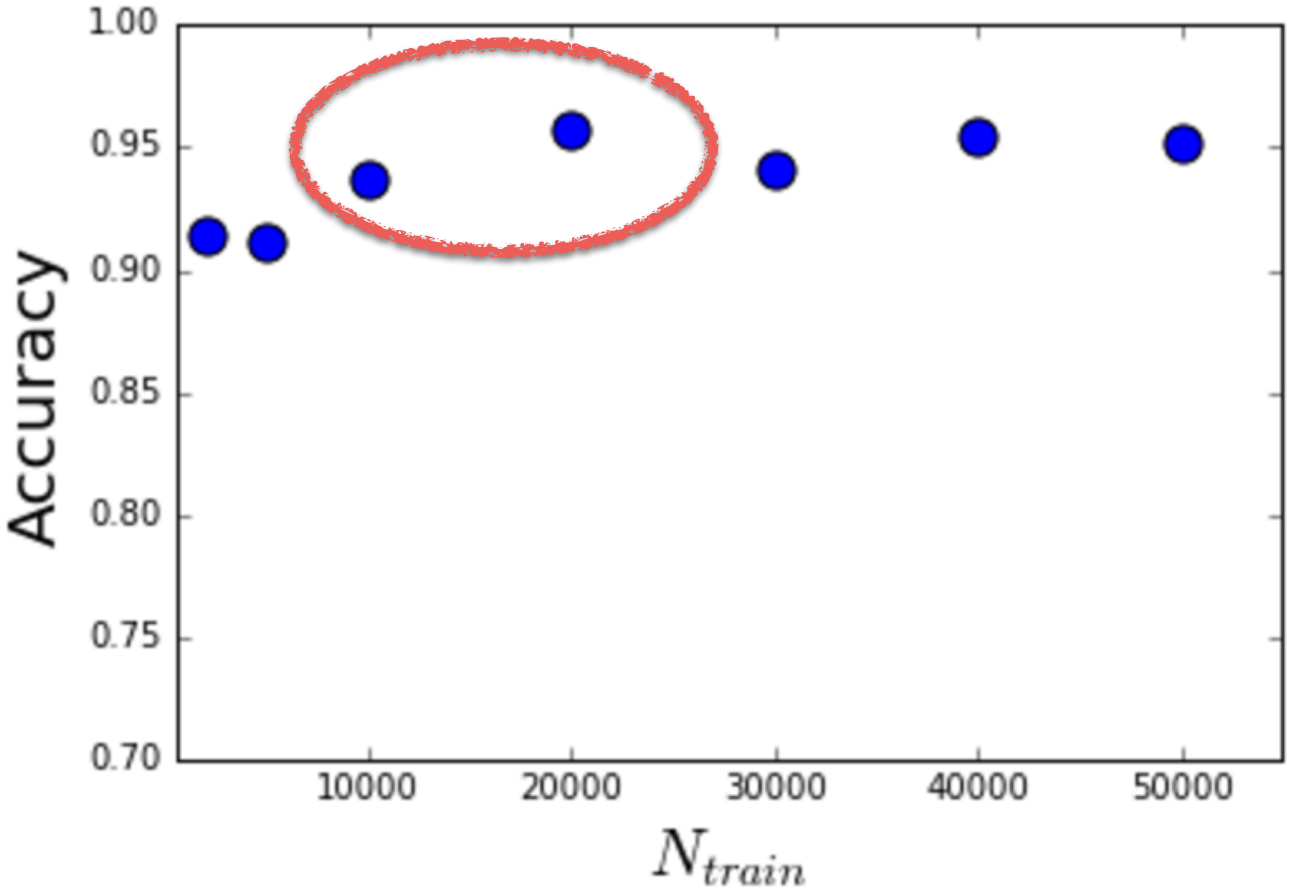
THE SIZE OF DATASETS
IS
INCREASING FAST BUT
WE DO
NOT ALWAYS HAVE
MILLIONS OF
LABELLED
EXAMPLES TO TRAIN
...

NOT REALLY…



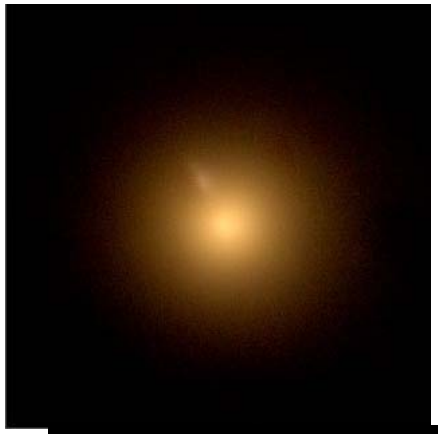
ACCURACY OF MORPHOLOGICAL CLASSIFICATIONS OF GALAXIES AS A FUNCTION OF THE SIZE OF THE TRAINING SET

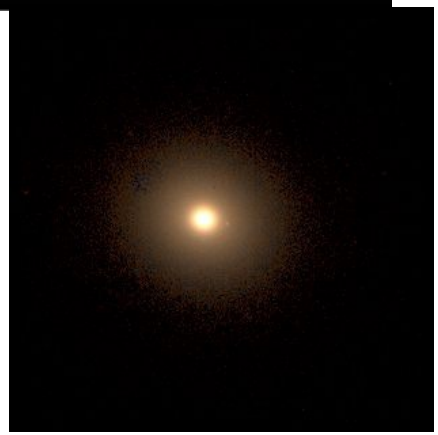
NOT REALLY…

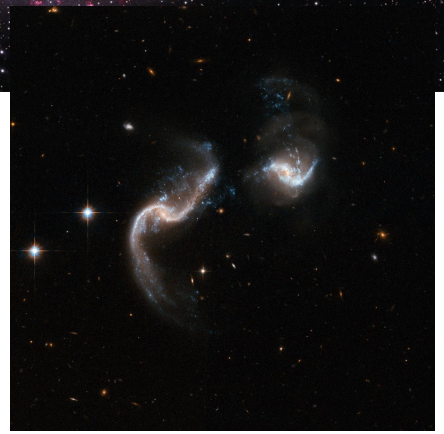


ACCURACY OF MORPHOLOGICAL CLASSIFICATIONS OF GALAXIES AS A FUNCTION OF THE SIZE OF THE TRAINING SET

WHY?







WHY?

ship dog deer bird ship cat dog dog
horse horse ship frog bird ship bird cat
automobile ship deer truck dog deer automobile horse

WHY?

IMAGES USED IN ASTROPHYSICS ARE SIMPLE…
ship
dog
deer
bird
ship
cat
dog
dog
horse
horse
ship
frog
bird
ship
bird
cat
automobile
ship
deer
truck
dog
deer
automobile
horse

TRICKS FOR “SMALL” DATASETS

DATA AUGMENTATION

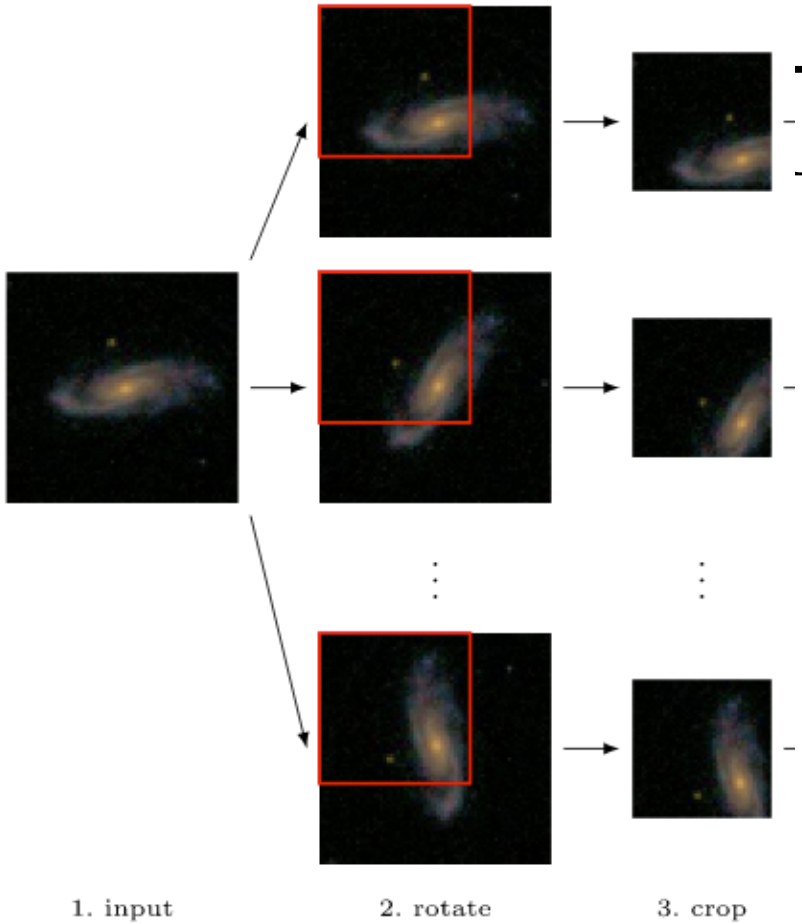
ANOTHER WAY TO REDUCE OVER-FITTING IS TO "AUGMENT" THE SIZE OF THE DATASET AVAILABLE FOR TRAINING

FOR MANY APPLICATIONS THE CLASSIFICATION SHOULD BE INDEPENDENT TO:

- TRANSALTIONS
- ROTATIONS
- SCALINGS

ETC…

DATA ATION



Dieleman+15

FOR MANY APPLICATIONS THE CLASSIFICATION SHOULD BE INDEPENDENT TO:

- TRANSALTIONS
- ROTATIONS
- SCALINGS

ETC…

DATA AUGMENTATION

FOR MANY APPLICATIONS THE CLASSIFICATION SHOULD BE INDEPENDENT TO:

- TRANSALTIONS
- ROTATIONS
- SCALINGS

ETC…

DATA AUGMENTATION

CAN BE DONE “ON THE FLY” DURING THE TRAINING PHASE

KERAS IMPLEMENTATION:

DEFINES THE RANGE OF PERTURBATIONS APPLIED TO IMAGES DURING TRAINING

```
datagen = ImageDataGenerator(
    featurewise_center=False,
    samplewise_center=False,
    featurewise_std_normalization=False,
    samplewise_std_normalization=False,
    zca_whitening=False,
    rotation_range=45,
    width_shift_range=0.05,
    height_shift_range=0.05,
    horizontal_flip=True,
    vertical_flip=True,
    zoom_range=[0.75,1.3])
```

DATA AUGMENTATION

SOMETIMES ADDING NOISE INCREASES THE CLASSIFICATION / REGRESSION ACCURACY!

STANDARD DEVIATION OF GAUSSIAN NOISE

```
model = Sequential()
model.add(GaussianNoise(0.01,input_shape=(img_channels, img_rows, img_cols)))
```

A LAYER OF GAUSSIAN NOISE ADDED IN THE MODEL

DOMAIN ADAPTATION (or knowledge transfer)

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

DOMAIN ADAPTATION (or knowledge transfer)

THE CONVOLUTIONAL PART OF A CNN IS
A FEATURE EXTRACTOR

IN THAT RESPECT, THEY ARE VERY FLEXIBLE …

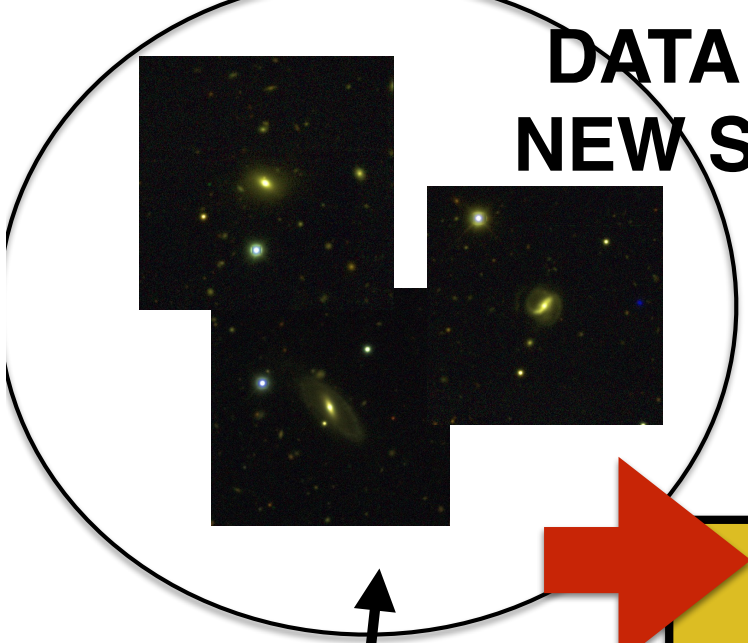
DOMAIN ADAPTATION (or knowledge transfer)

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?

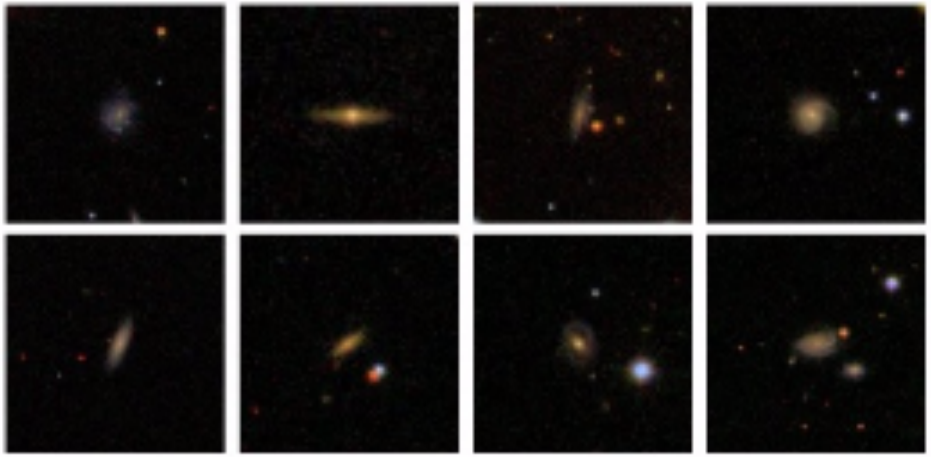
DEEP-LEARNING BASED MACHINE

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

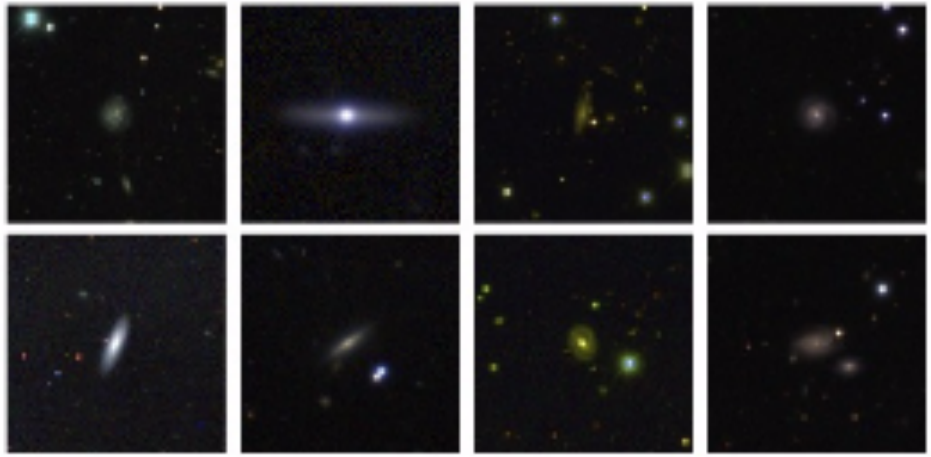
Transfer knowledge?

Human classifications from existing survey

Dominguez-Sanchez+18b



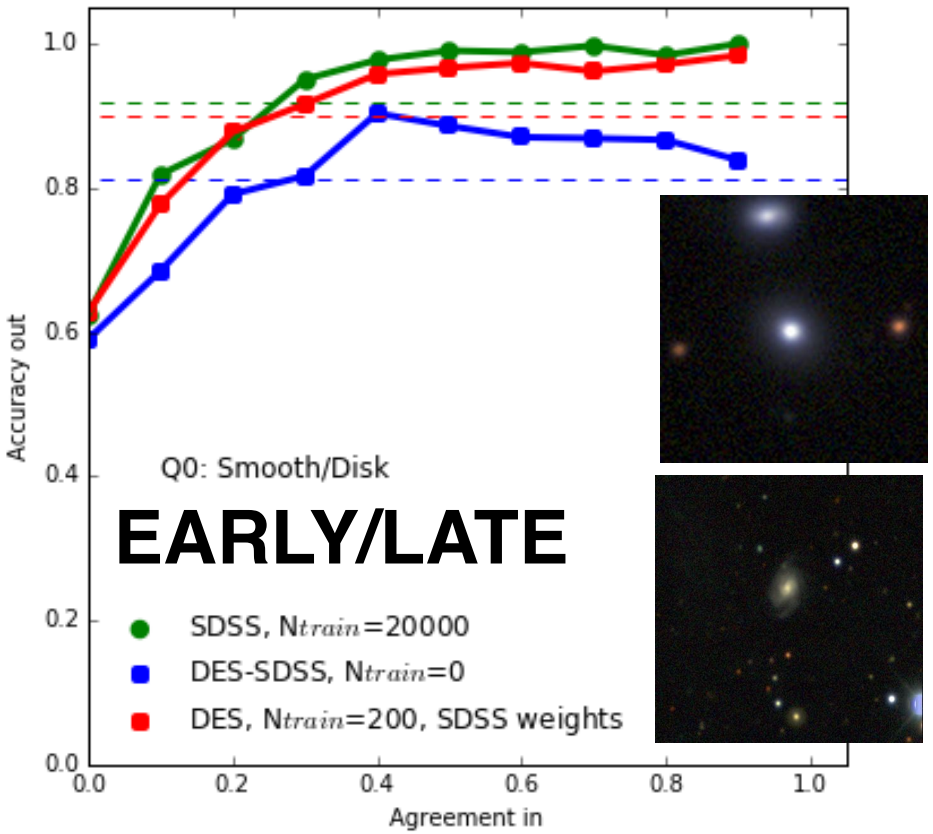
SDSS

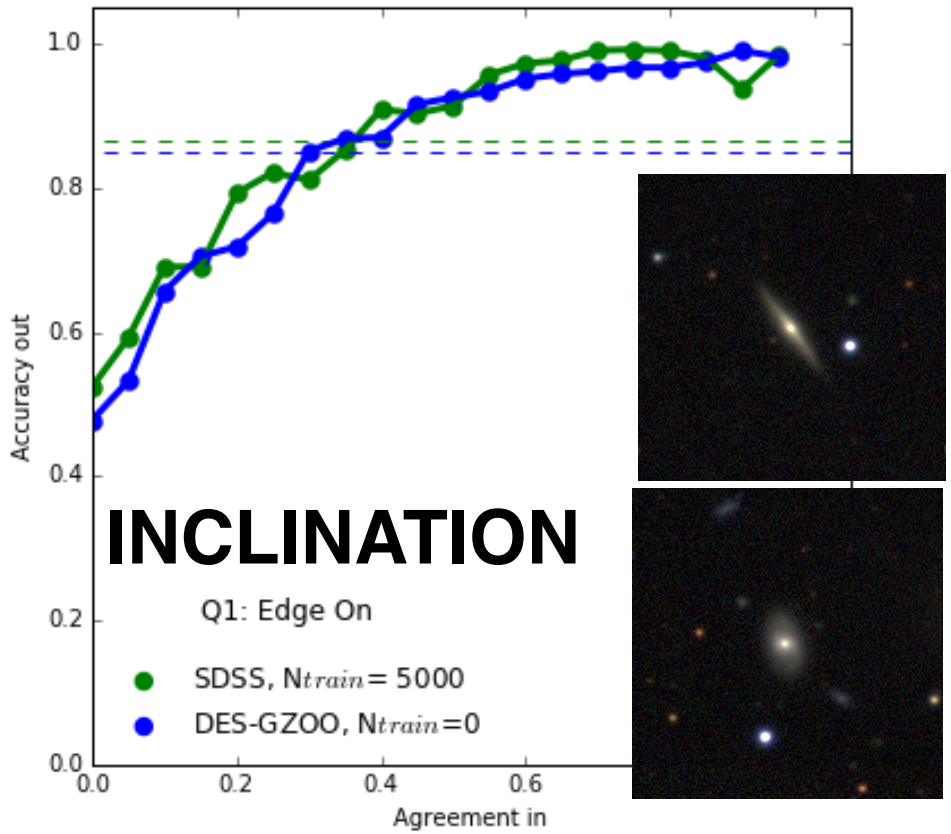


DES

Dominguez-Sanchez+18b

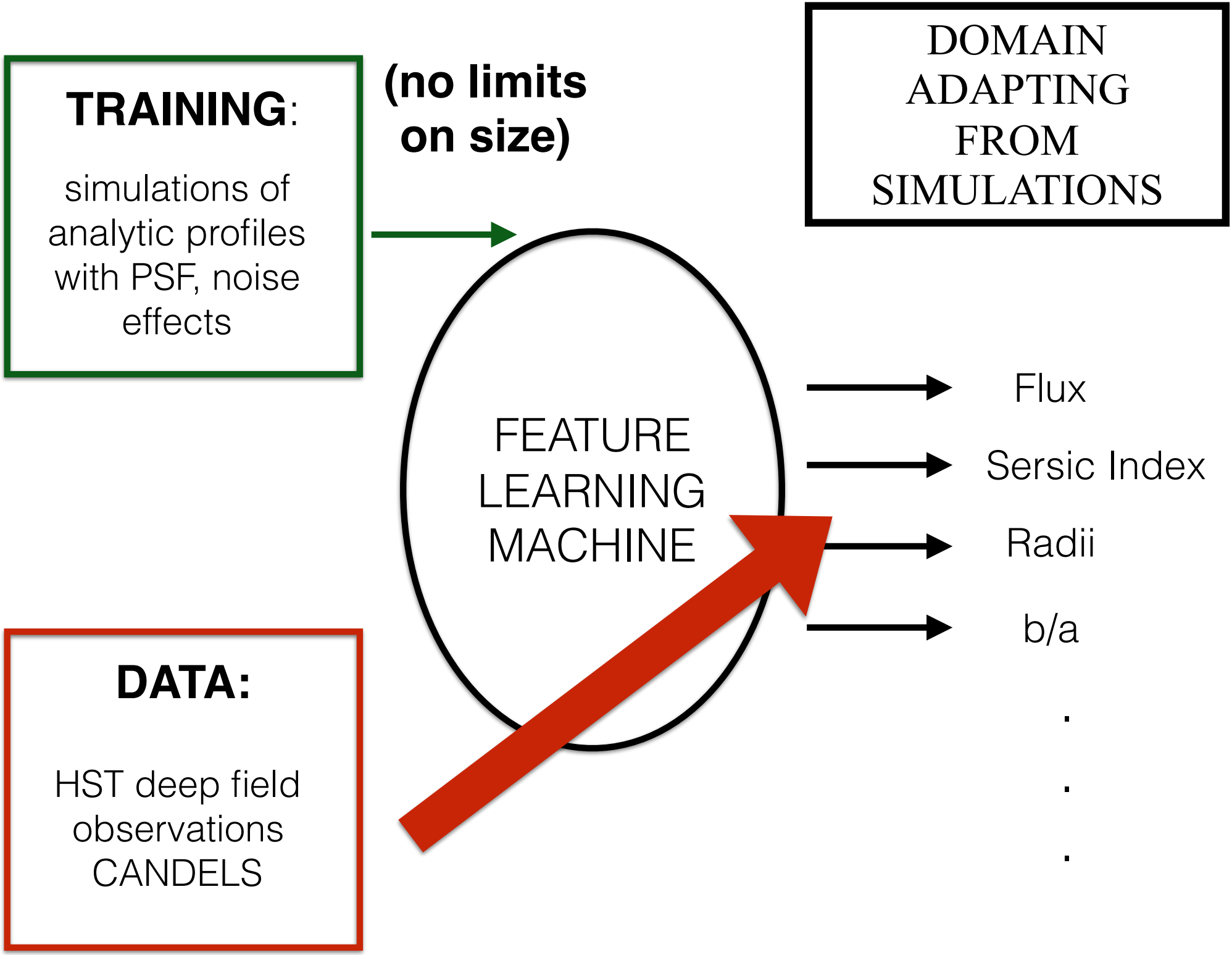
Knowledge Transfer from SDSS to DES

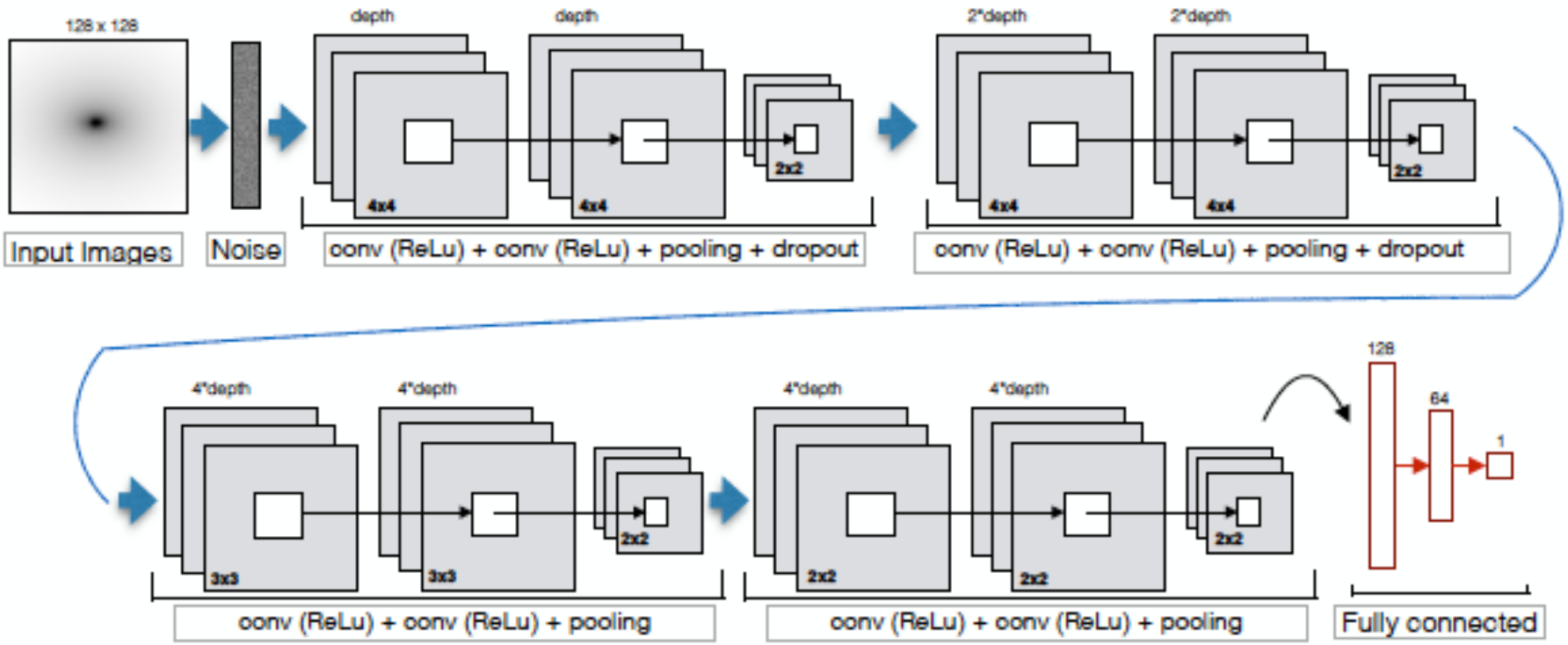




Only 200 (**1%!**) objects classified in DES are needed to reach an accuracy >90% if a machine trained on the SDSS is used

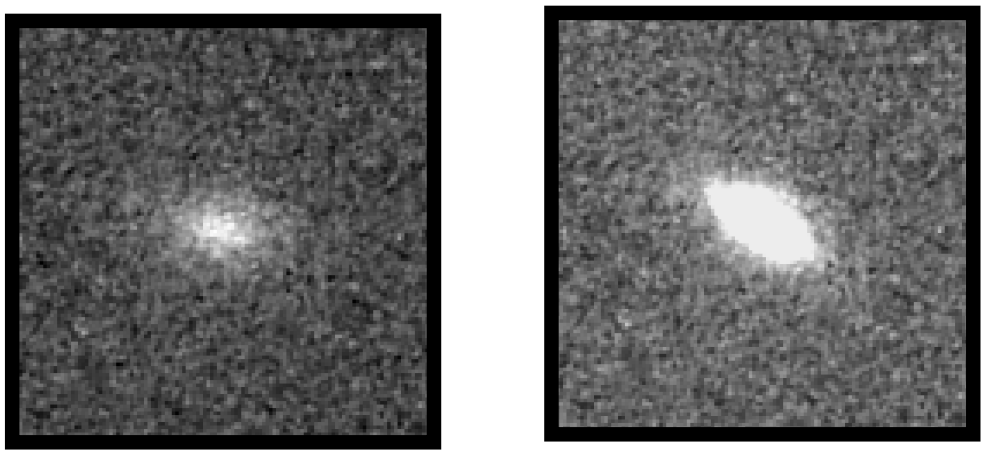
For some properties, i.e. EDGE-ON galaxies. No training at all is needed to go from SDSS to DES


TRAINING:
simulations of analytic profiles with PSF, noise effects
(no limits on size)
DOMAIN ADAPTING FROM SIMULATIONS
FEATURE LEARNING MACHINE
Flux
Sersic Index
Radii
b/a
.
.
.
DATA:
HST deep field observations CANDELS


128 x 128
Input Images
Noise
depth
depth
4x4
4x4
2x2
conv (ReLu) + conv (ReLu) + pooling + dropout
2*depth
2*depth
4x4
4x4
2x2
conv (ReLu) + conv (ReLu) + pooling + dropout
4*depth
4*depth
3x3
3x3
2x2
conv (ReLu) + conv (ReLu) + pooling
4*depth
4*depth
2x2
2x2
2x2
conv (ReLu) + conv (ReLu) + pooling
128
64
1
Fully connected

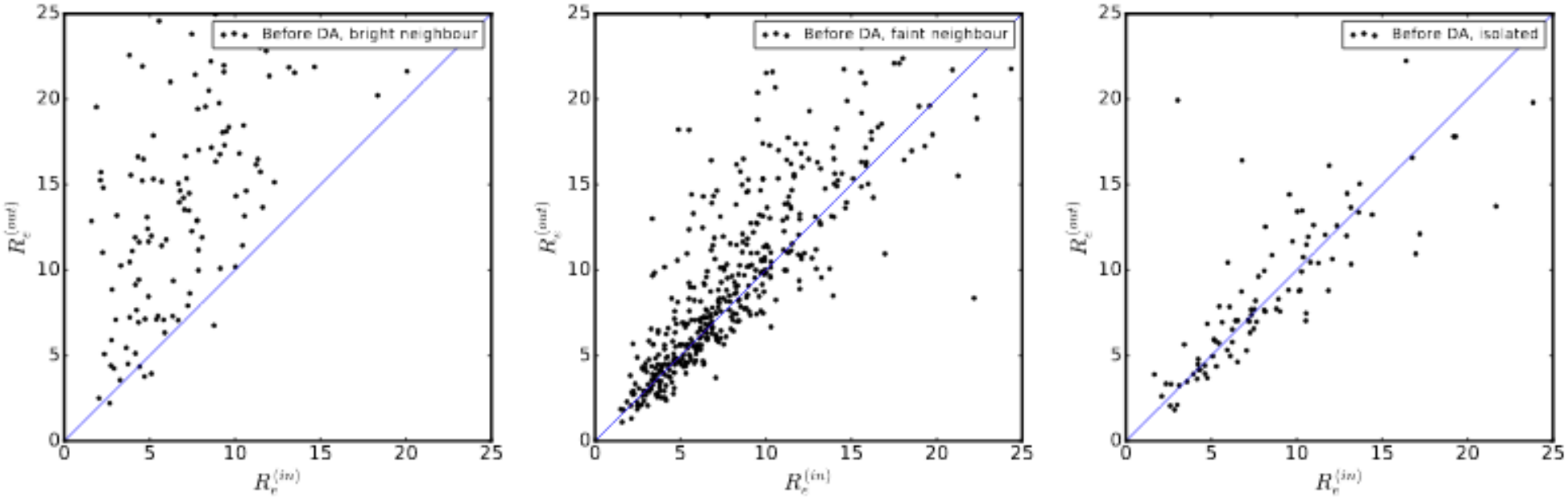
Standard analytic profiles

- 100.000-300.000 galaxies [GALSIM]
  - Real HST background added + PSF (F160)
  - Random distribution of parameters (uniform):
    - 18<Mag<24, 0<BT<1, <Nb<, Nd=1, 0.2<log(rb)<1.3, 0.2<log(rd)<1.5, 0.05<eb<0.95, 0.05<ed<0.95, 0<PA<180
  - 64*64 stamps
  - **FULLY IDEALISTIC -**
    NO COMPANIONS
    NO IRREGULARS
    NO CLUMPY!



MORPHOMETRY OF REAL GALAXIES TRAINED ON ANALYTIC PROFILES

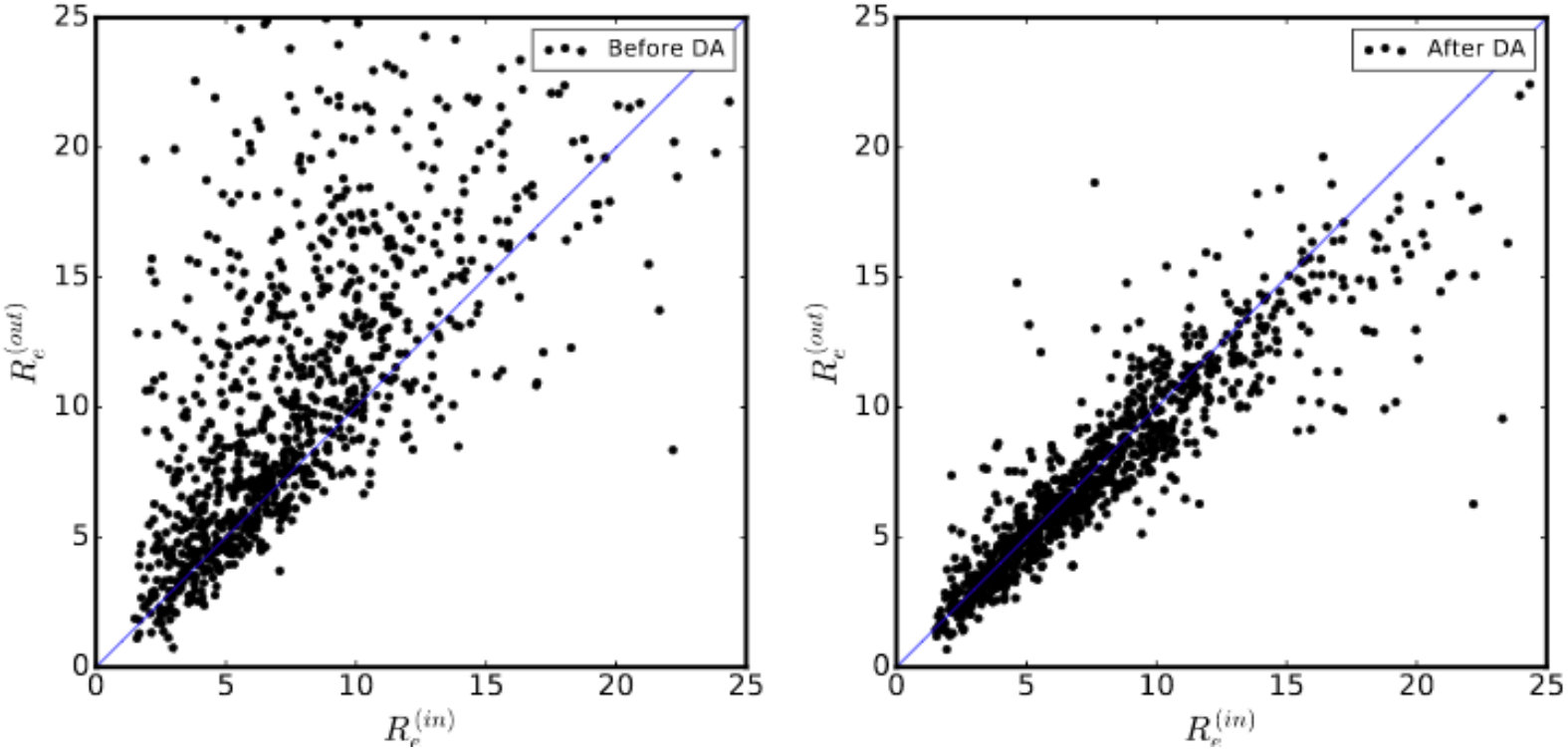
BRIGHT NEIGH. **FAINT NEIGH.** **ISOLATED**



(c) BDA bright neigbours (d) BDA faint neigbours (e) BDA isolated galaxies

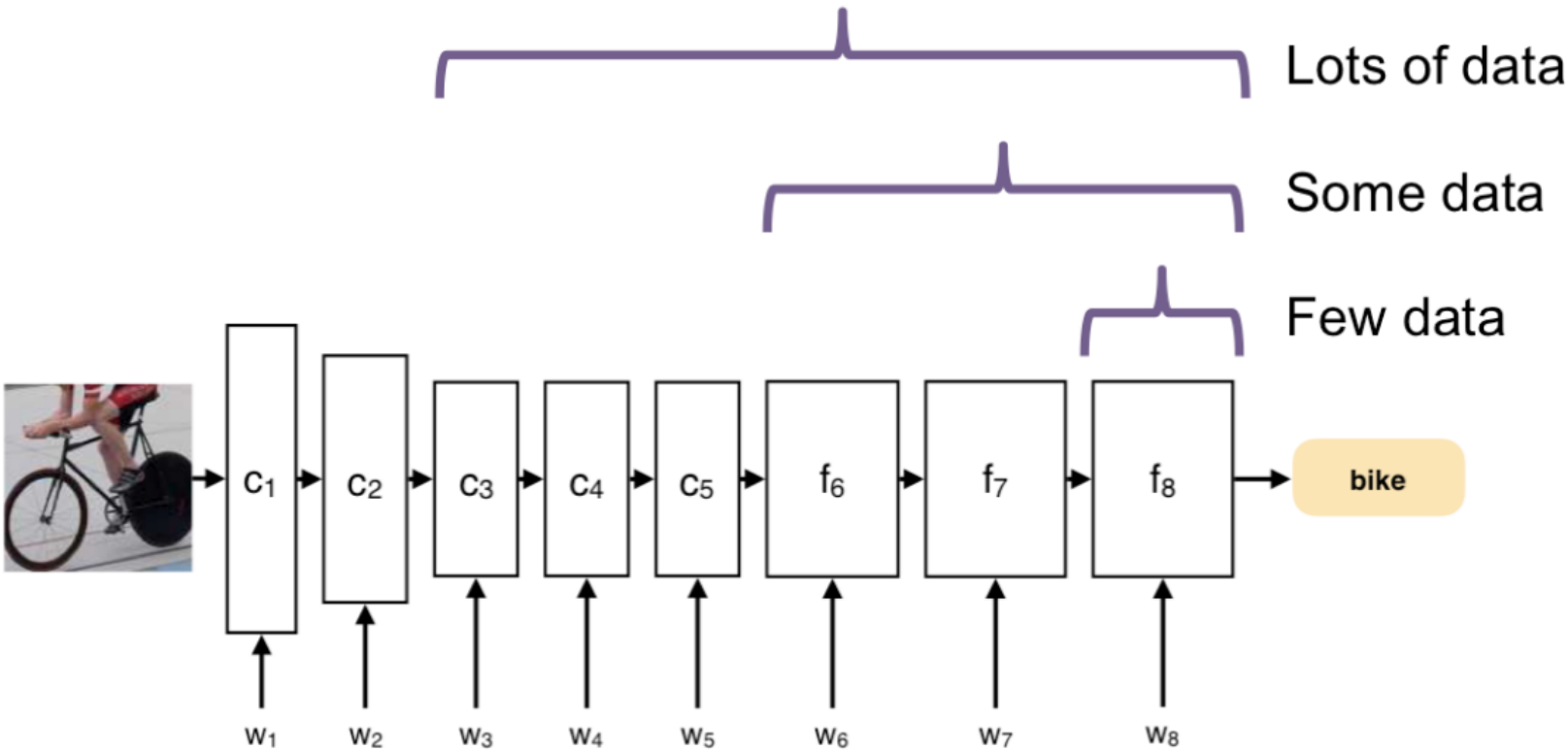
Tuccillo+18

DOMAIN ADAPTATION: 0.1% OF “REAL” GALAXIES



Tuccillo+18

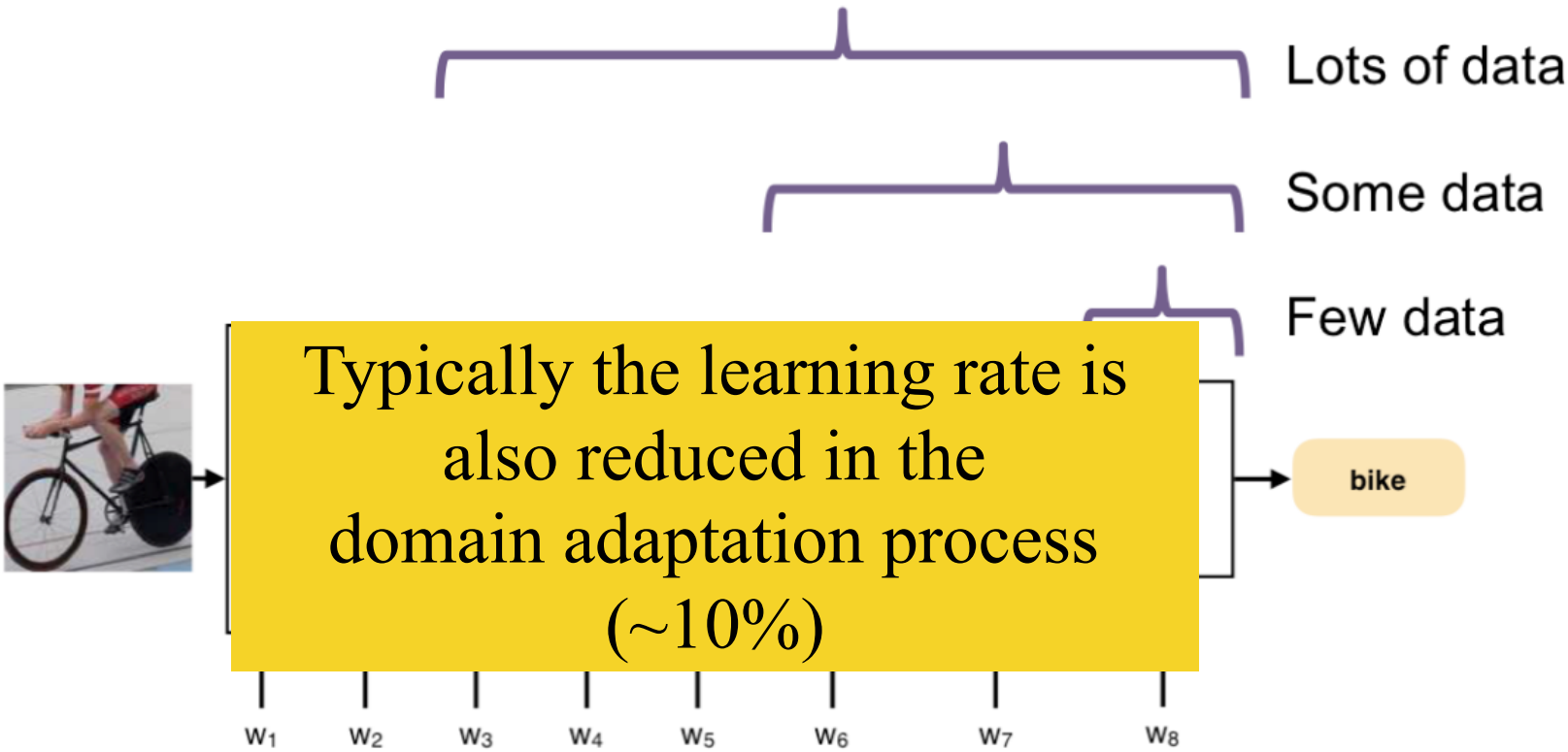
HOW MANY LAYERS "DOMAIN ADAPT" ?



DEPENDING ON HOW MUCH SIMILAR BOTH DATASETS ARE:

- FINE-TUNE ONLY FULLY CONNECTED LAYERS
- FINE-TUNE A FEW LAYERS
- FINE-TUNE ALL LAYERS

HOW MANY LAYERS "DOMAIN ADAPT" ?



Typically the learning rate is also reduced in the domain adaptation process (~10%)

DEPENDING ON HOW MUCH SIMILAR BOTH DATASETS ARE:

- FINE-TUNE ONLY FULLY CONNECTED LAYERS
- FINE-TUNE A FEW LAYERS
- FINE-TUNE ALL LAYERS

KERAS IMPLEMENTATION

"FULL" DOMAIN ADAPTATION

learning rate

```
model.save_weights(test_name+".hd5",overwrite=True)
```

```
keras.optimizers.Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=None, decay=0.0, amsgrad=False)
```

```
model.load_weights(model_name+".hd5")
```

load weights of previous training

start training

```
history = model.fit_generator(
            datagen.flow(X_train, Y_train, batch_size=batch_size),
            samples_per_epoch=X_train.shape[0],
            nb_epoch=nb_epoch,
            validation_data=(X_val, Y_val),
            callbacks=[earlystopping, modelcheckpoint]
        )
```

KERAS IMPLEMENTATION

"PARTIAL" DOMAIN ADAPTATION

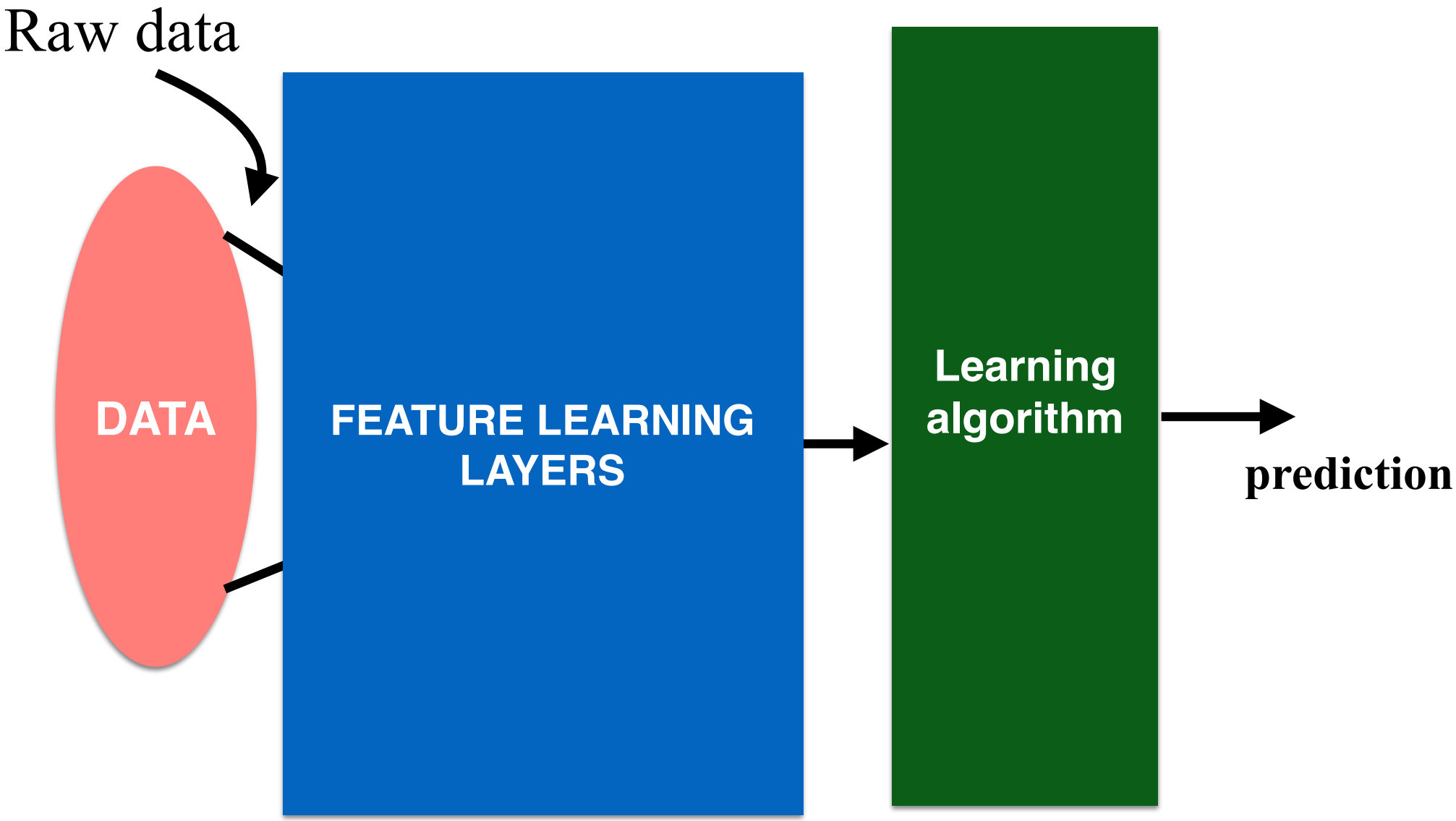
```
model.save_weights(test_name+".hd5",overwrite=True)
```

```
keras.optimizers.Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=None, decay=0.0, amsgrad=False)
```

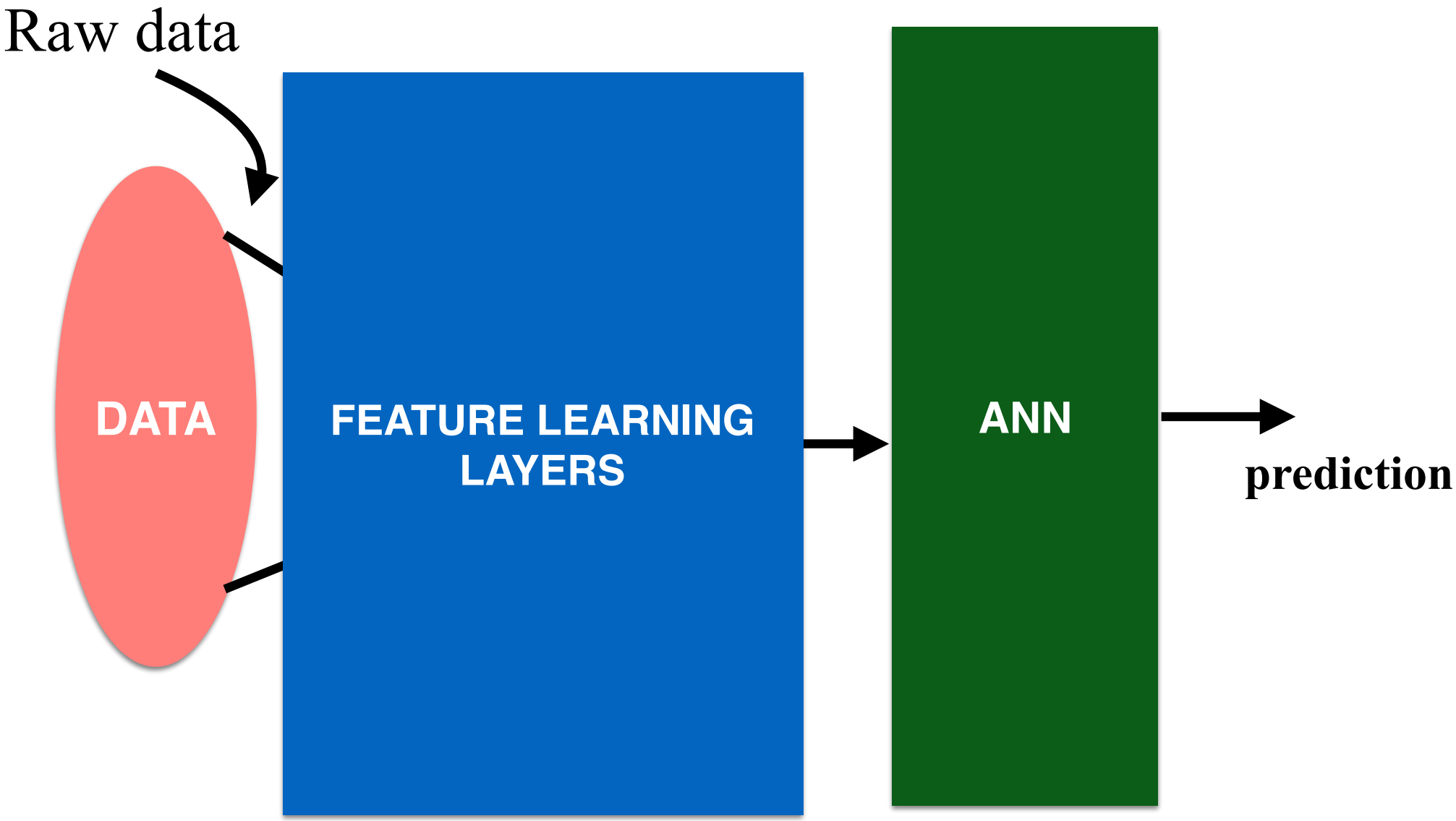
```
model.load_weights(model_name+".hd5")
```

```
all_layers = model.layers
for i in range(model.layers.index(mid_start)):
    all_layers[i].trainable = False
```

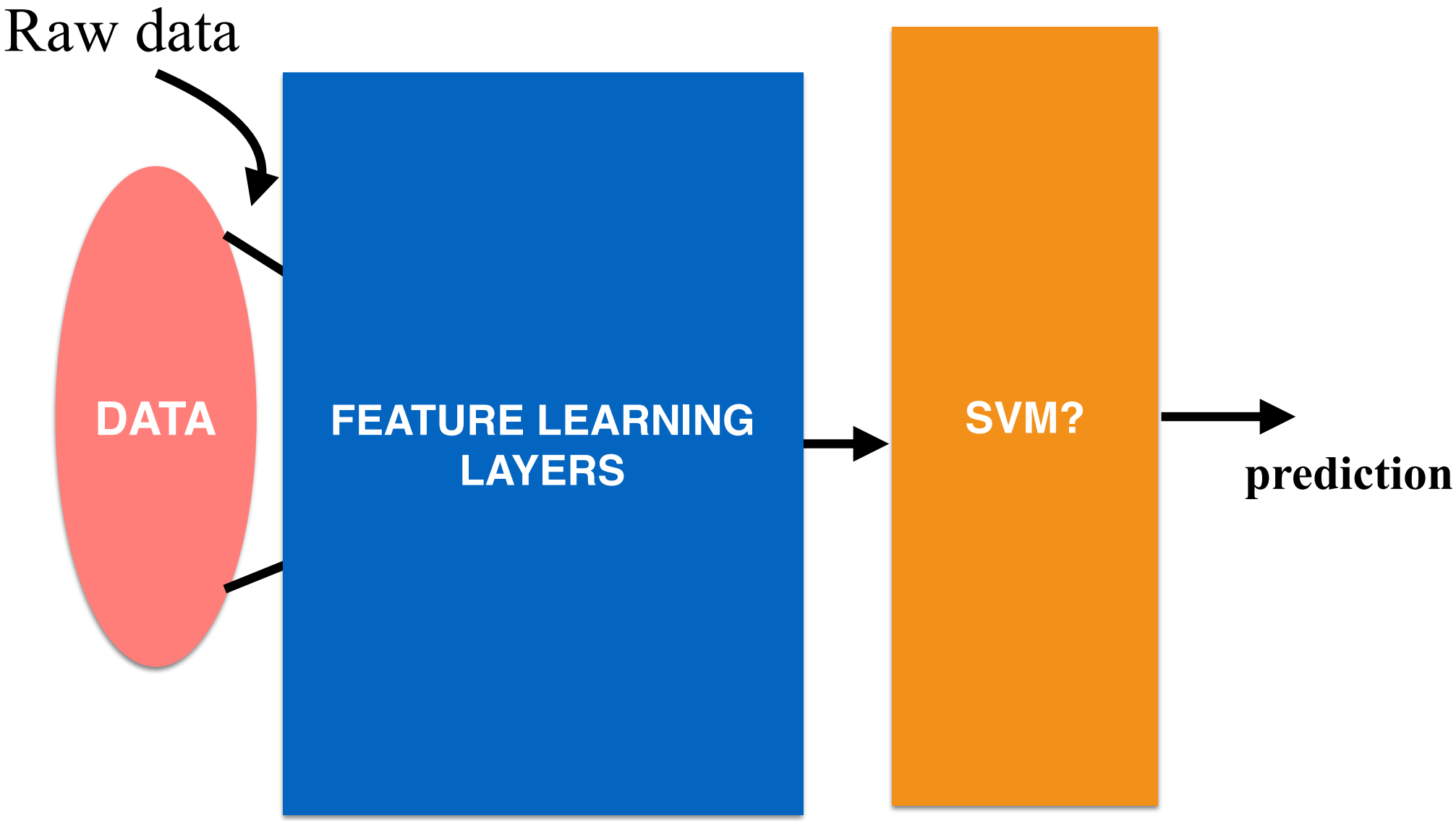
Freeze some layers


Raw data
DATA
FEATURE LEARNING LAYERS
Learning algorithm
prediction

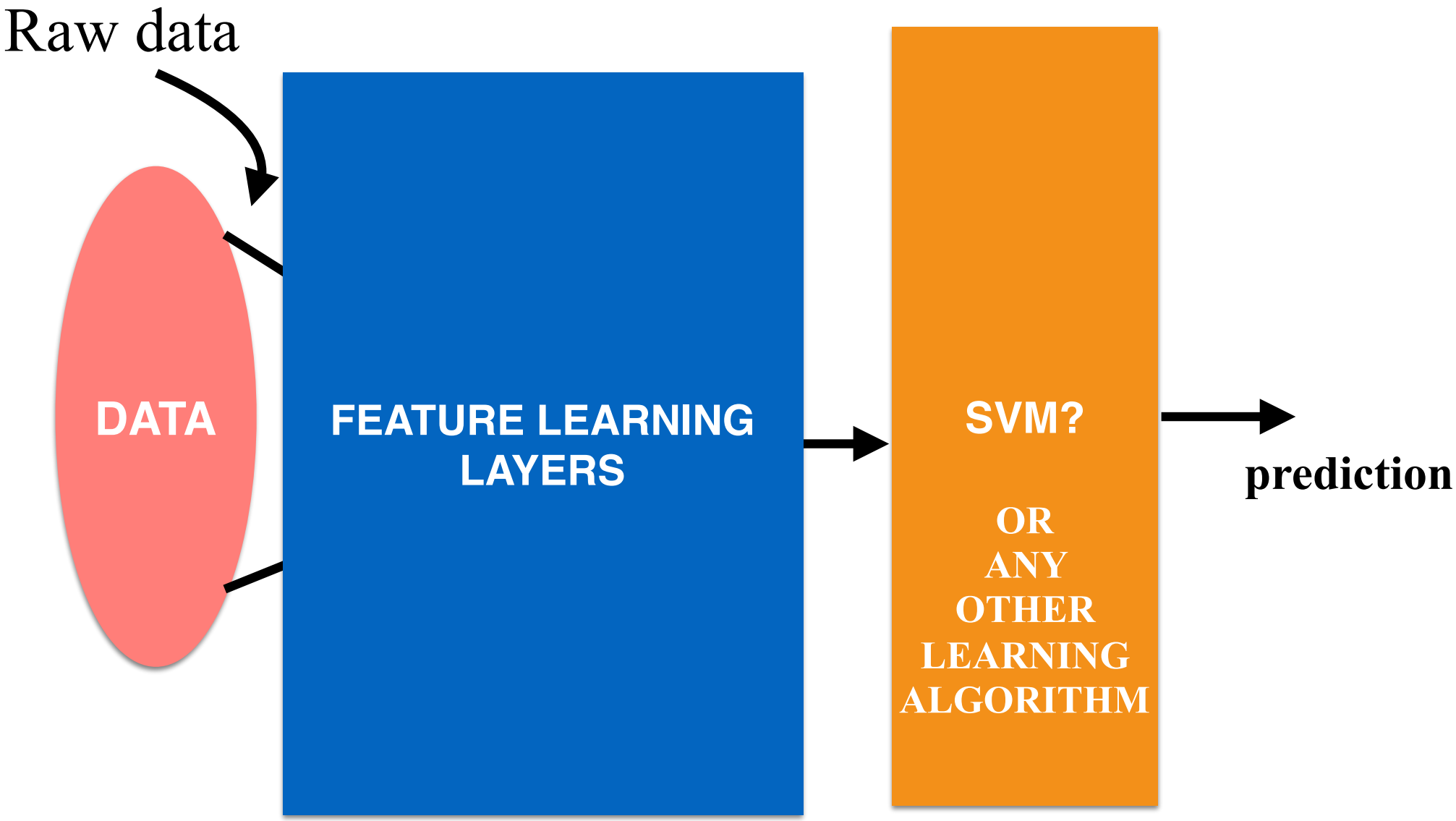
THE LEARNING ALGORITHM CAN BE CHANGED



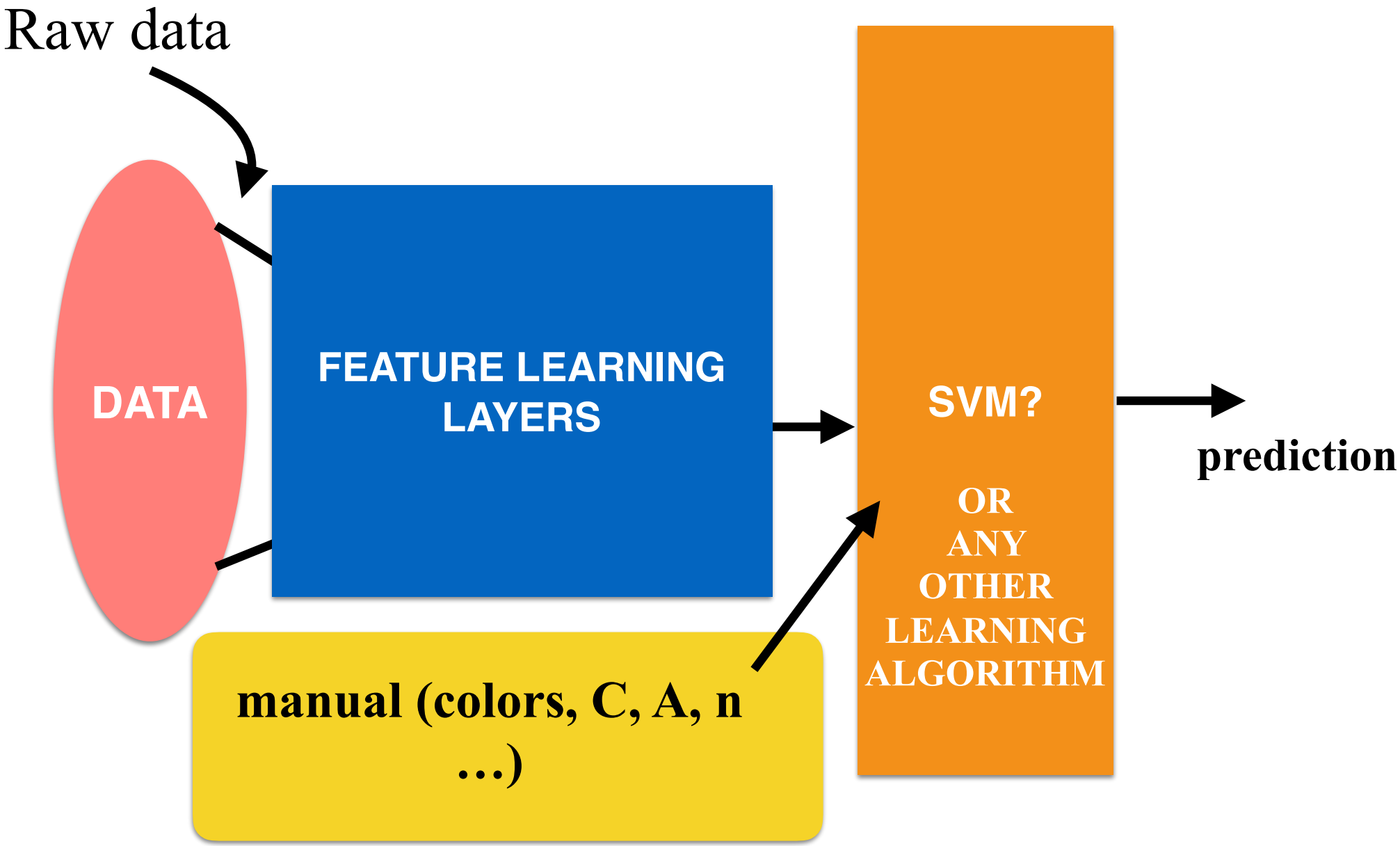
THE LEARNING ALGORITHM CAN BE CHANGED



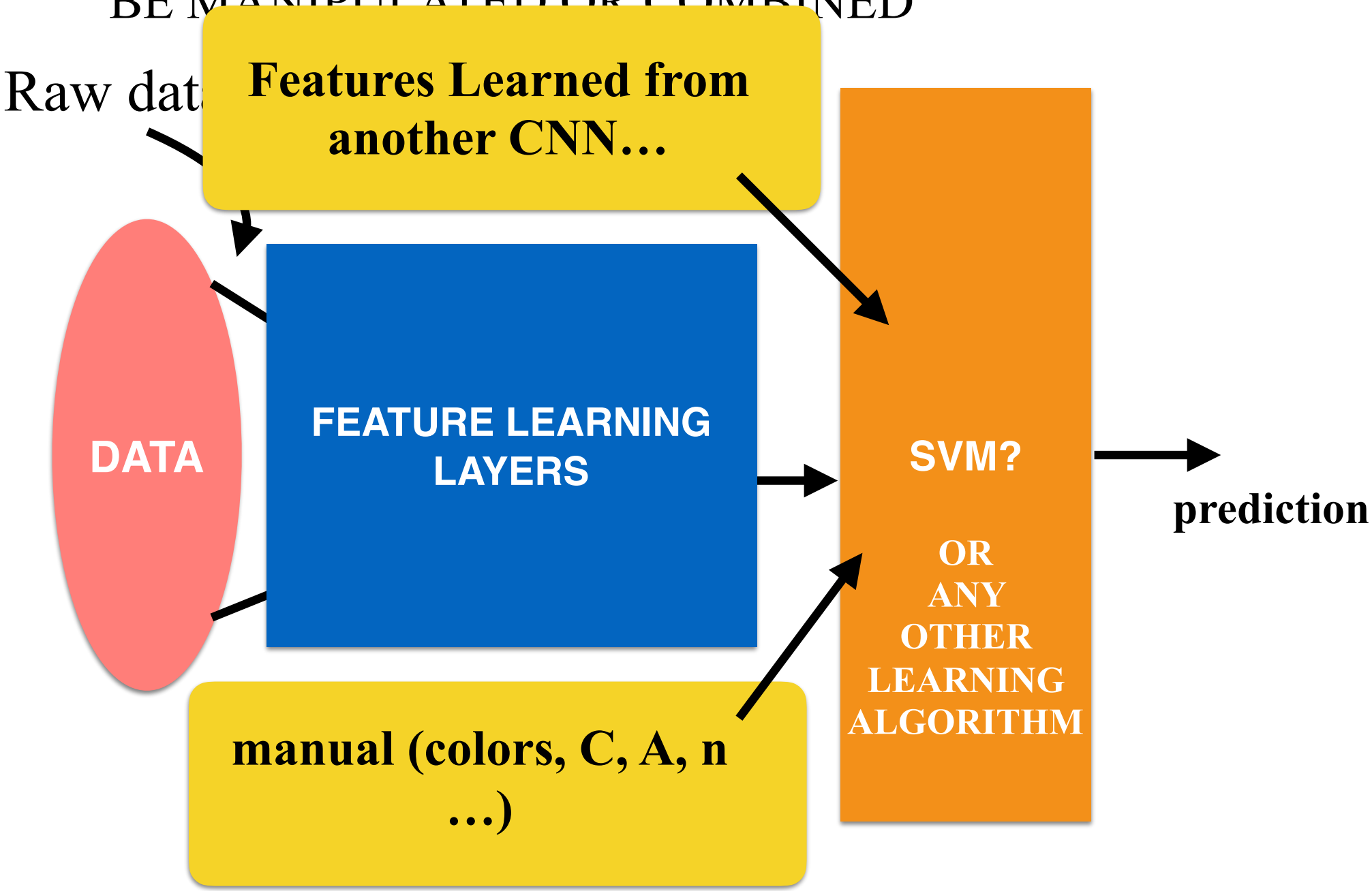
THE LEARNING ALGORITHM CAN BE CHANGED



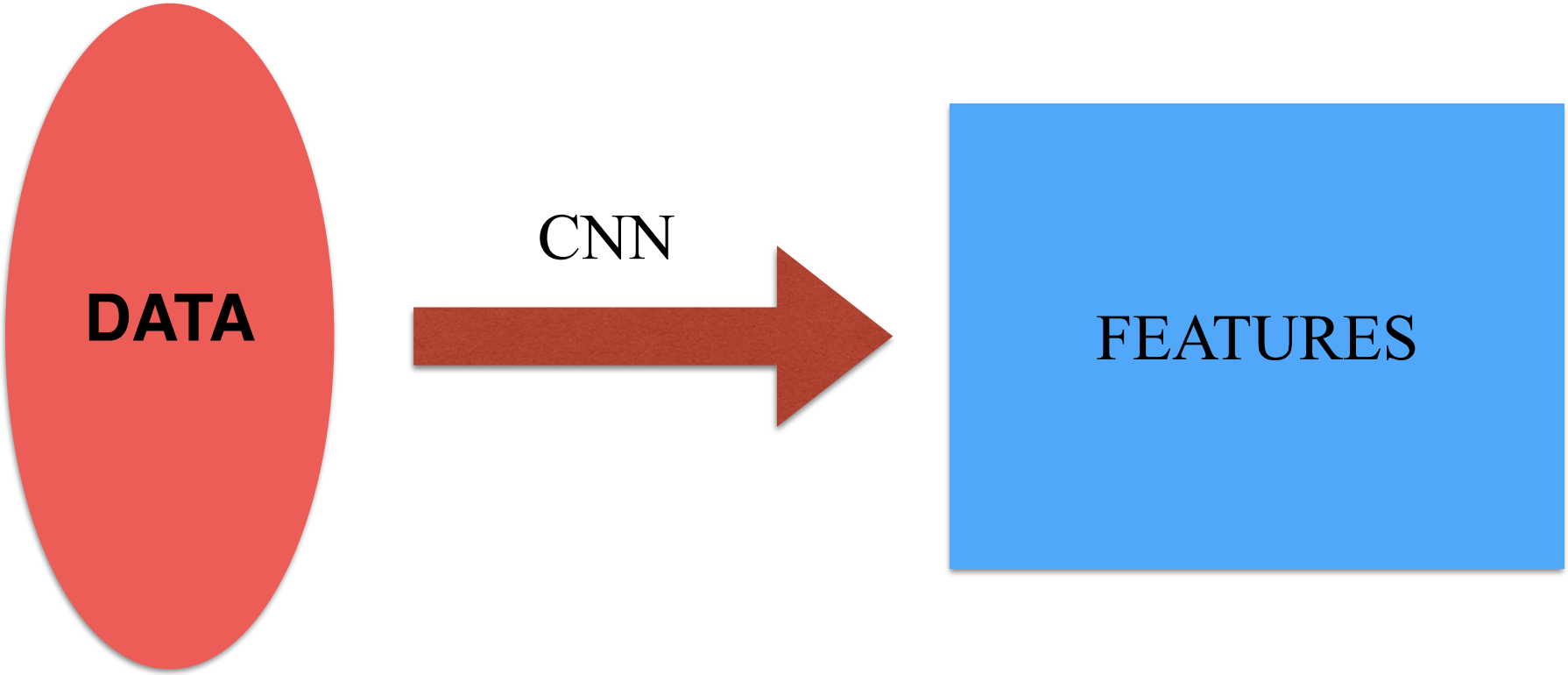
THE FEATURES CAN BE MANIPULATED OR COMBINED



THE FEATURES CAN
BE MANIPULATED OR COMBINED

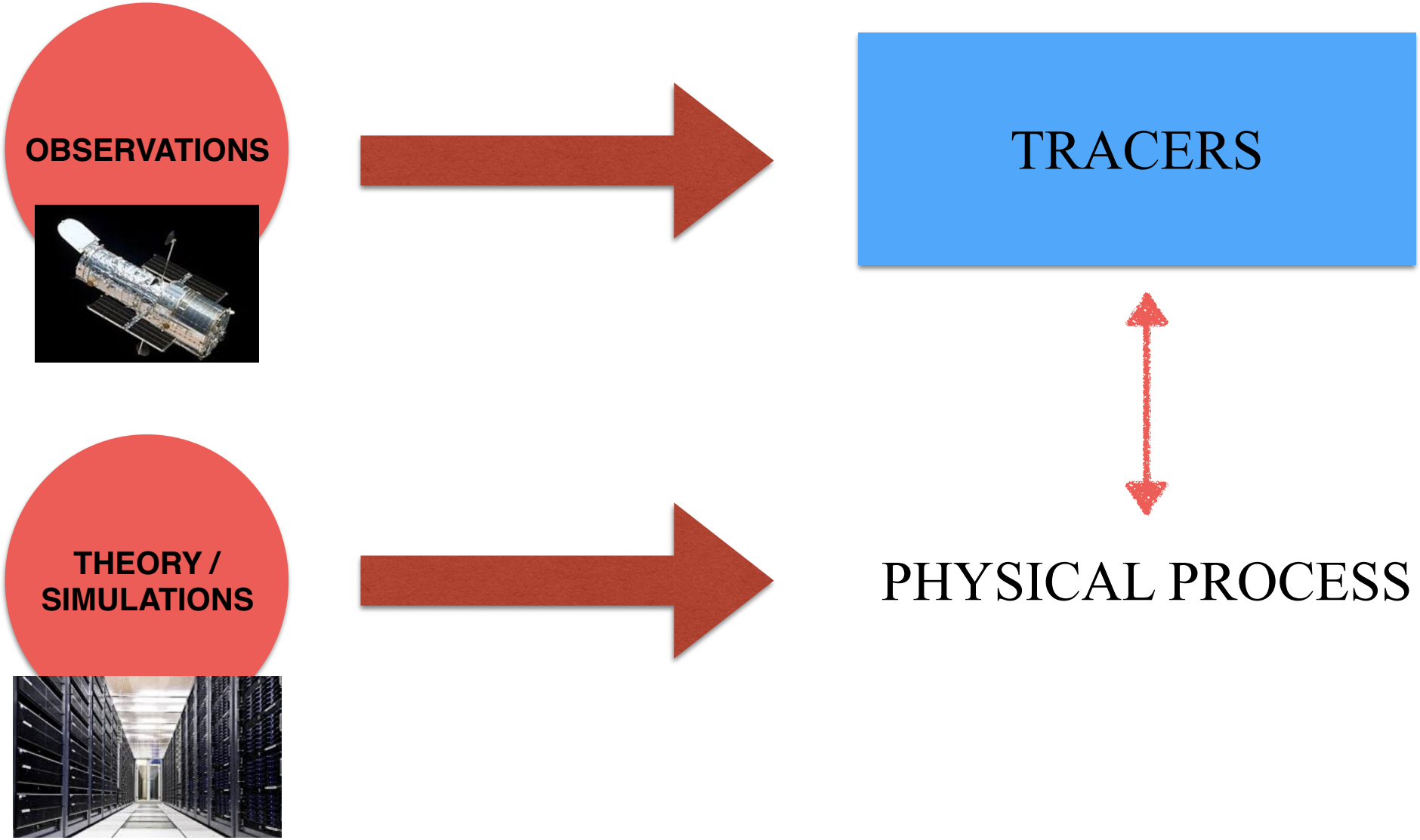


CNNs AS FEATURE EXTRACTOR



SO, CNNs BLINDLY EXTRACT FEATURES / TRACERS FROM DATA

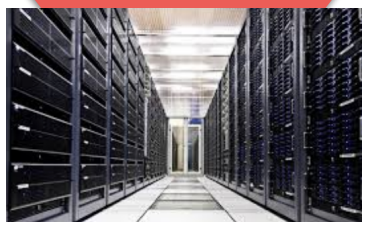
IN ASTRONOMY…



IN ASTRONOMY…

OBSERVATIONS

THEORY / SIMULATIONS



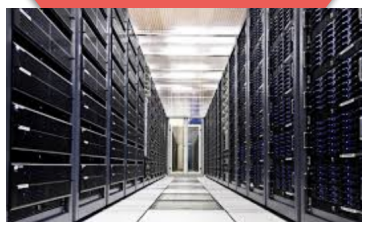
TRACERS

PHYSICAL PROCESS

IN ASTRONOMY…

OBSERVATIONS

THEORY / SIMULATIONS



TRACERS

[tidal tales, SFR enhancement, asymmetries …]

PHYSICAL PROCESS

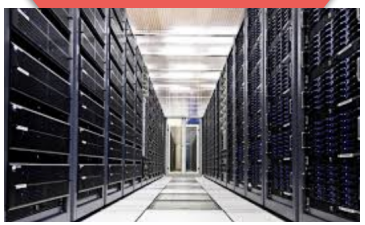
IN ASTRONOMY…

OBSERVATIONS

TRACERS

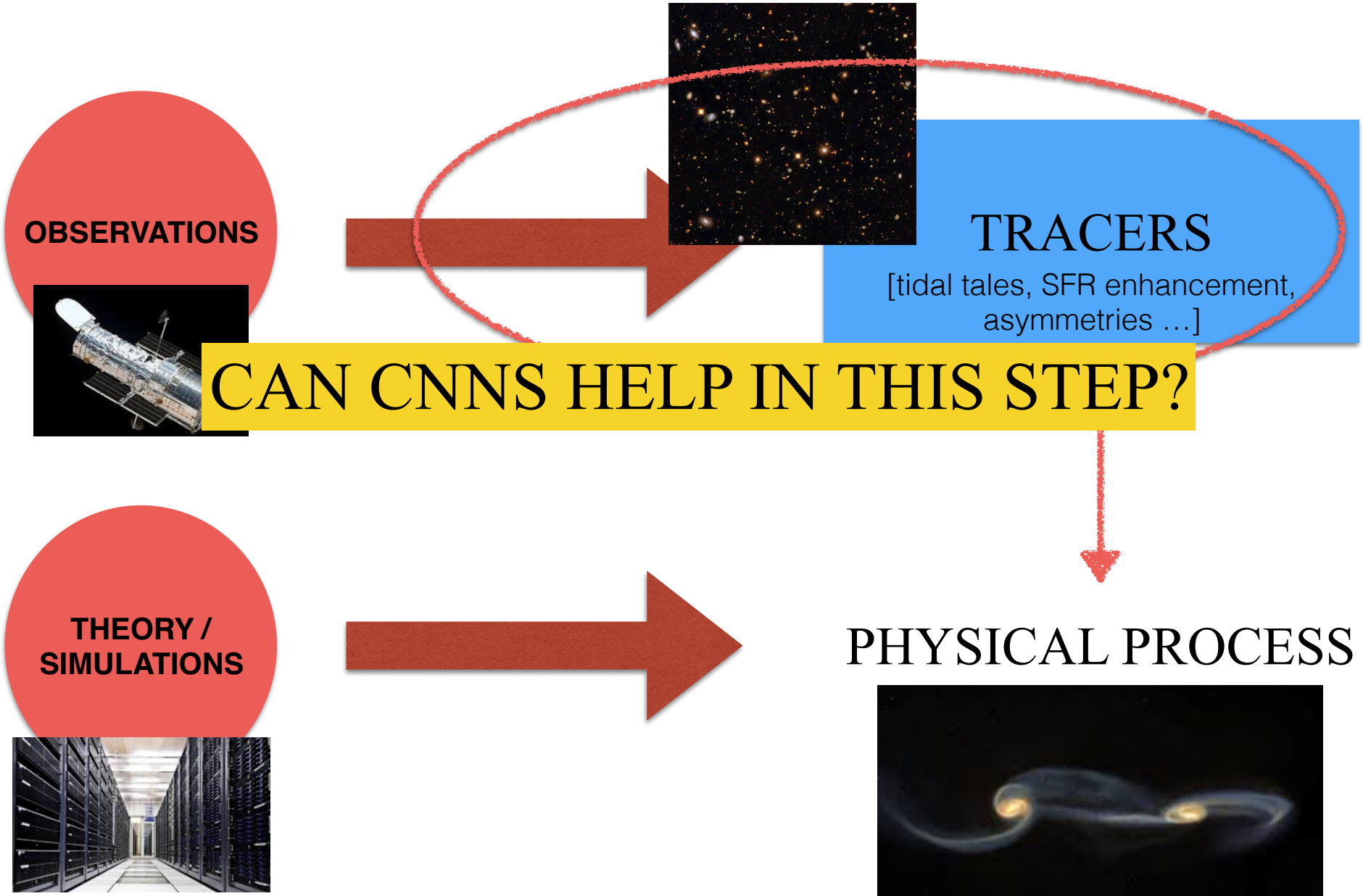
[tidal tales, SFR enhancement, asymmetries …]

THEORY / SIMULATIONS

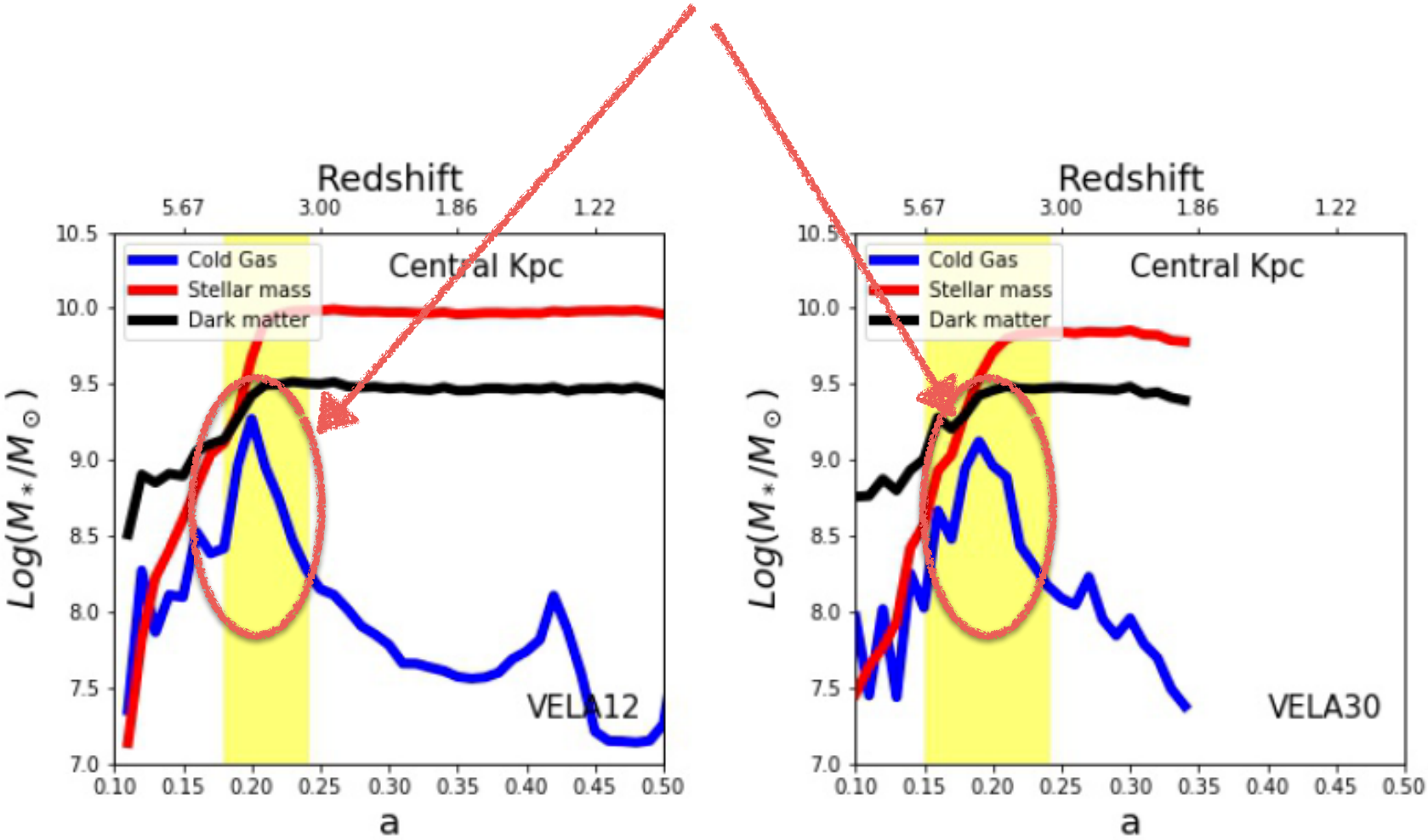


PHYSICAL PROCESS

IN ASTRONOMY…

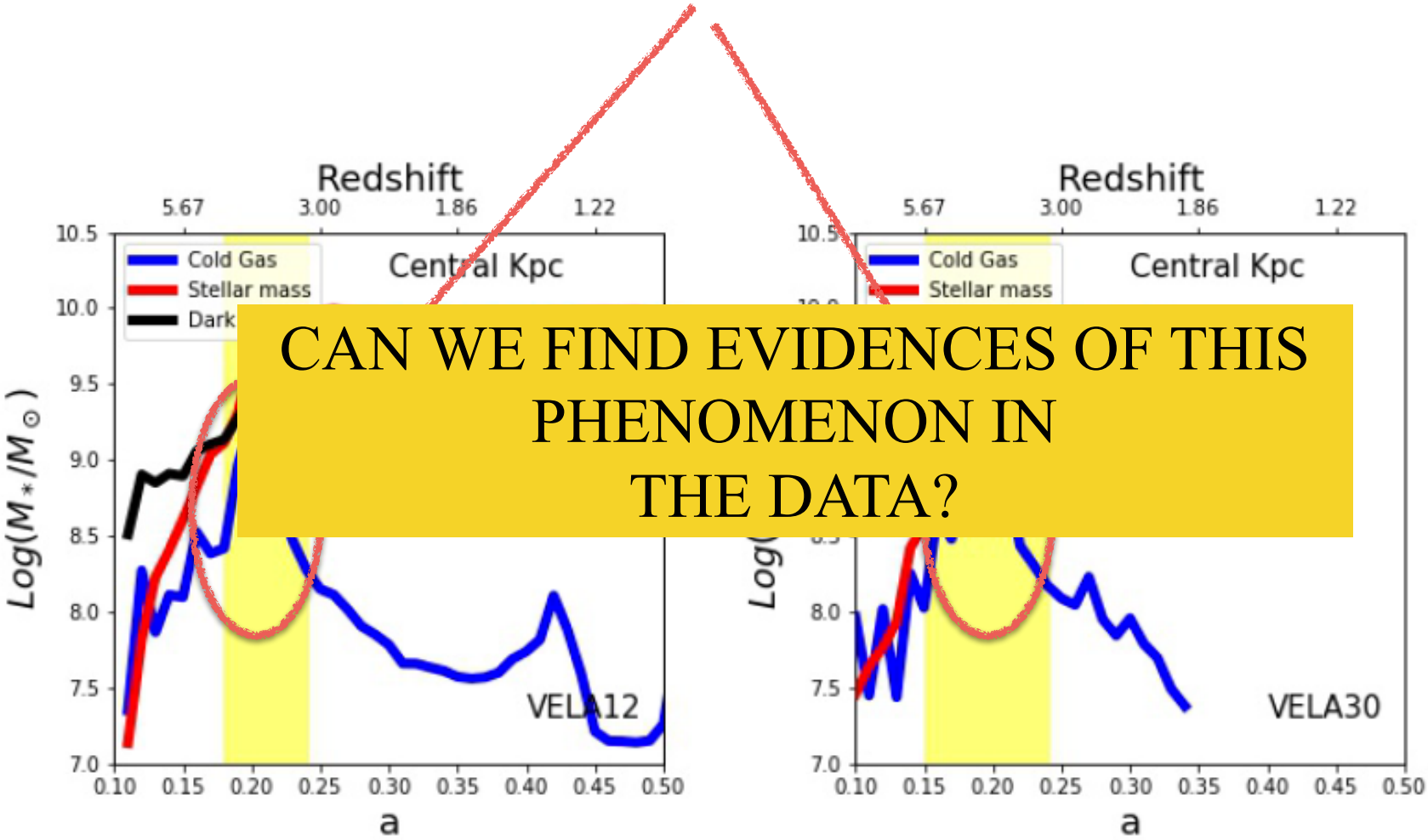


SIMULATIONS PREDICT THAT MANY GALAXIES GO THROUGH A CENTRAL "BLUE NUGGET" PHASE



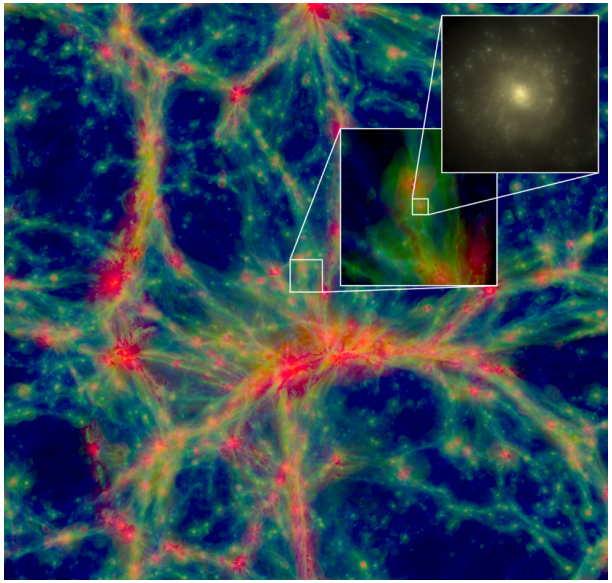
Huertas-Company+18

SIMULATIONS PREDICT THAT MANY GALAXIES GO THROUGH A CENTRAL "BLUE NUGGET" PHASE

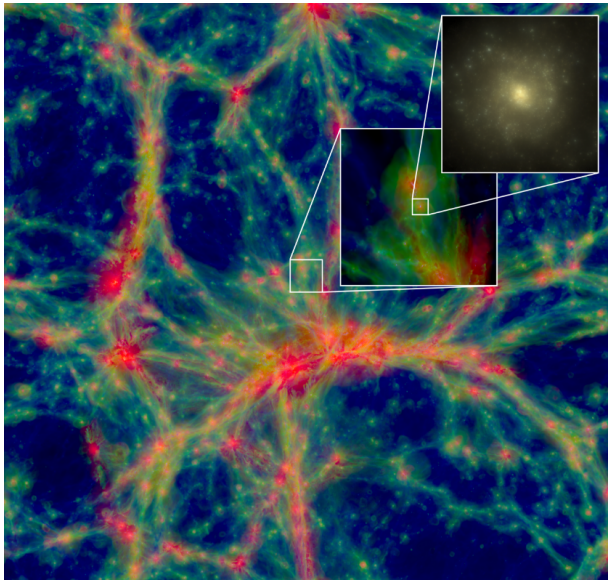


CAN WE FIND EVIDENCES OF THIS PHENOMENON IN THE DATA?

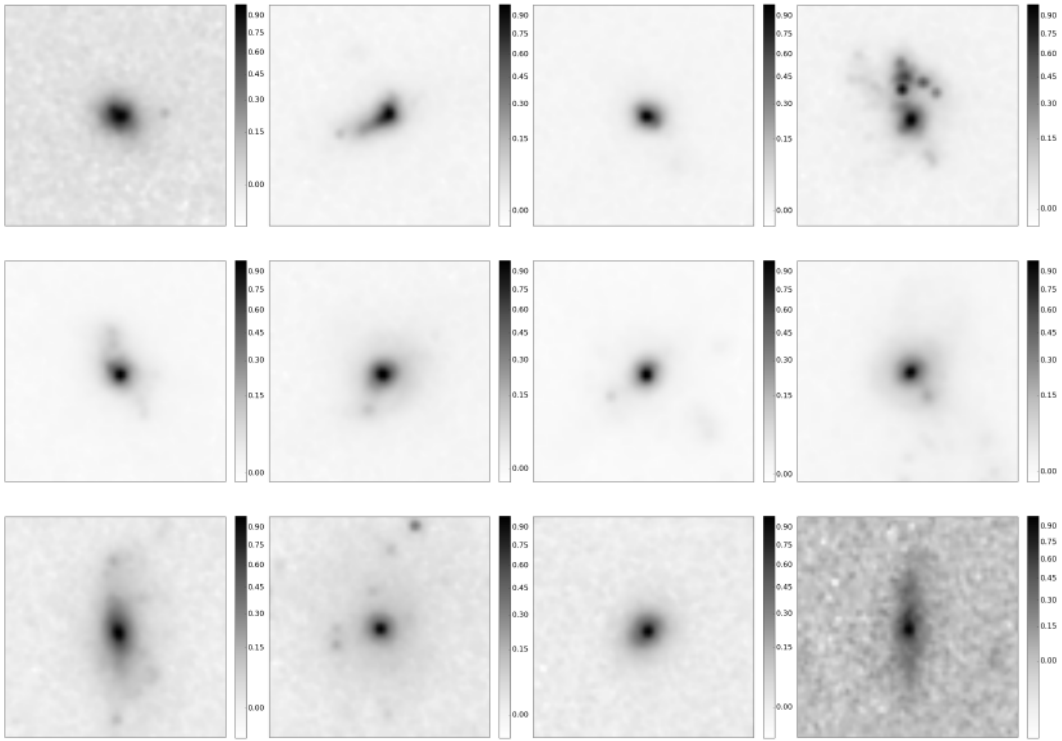
Huertas-Company+18



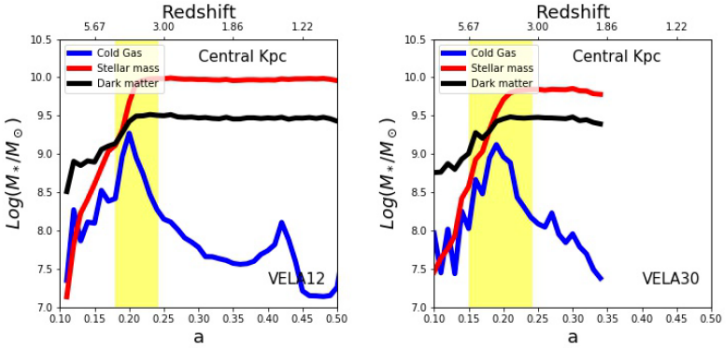
[COSMOLOGICAL SIMULATION]



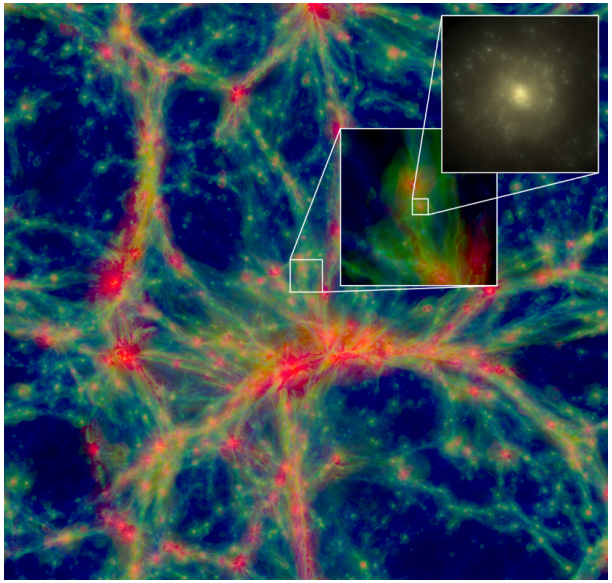
[COSMOLOGICAL SIMULATION]



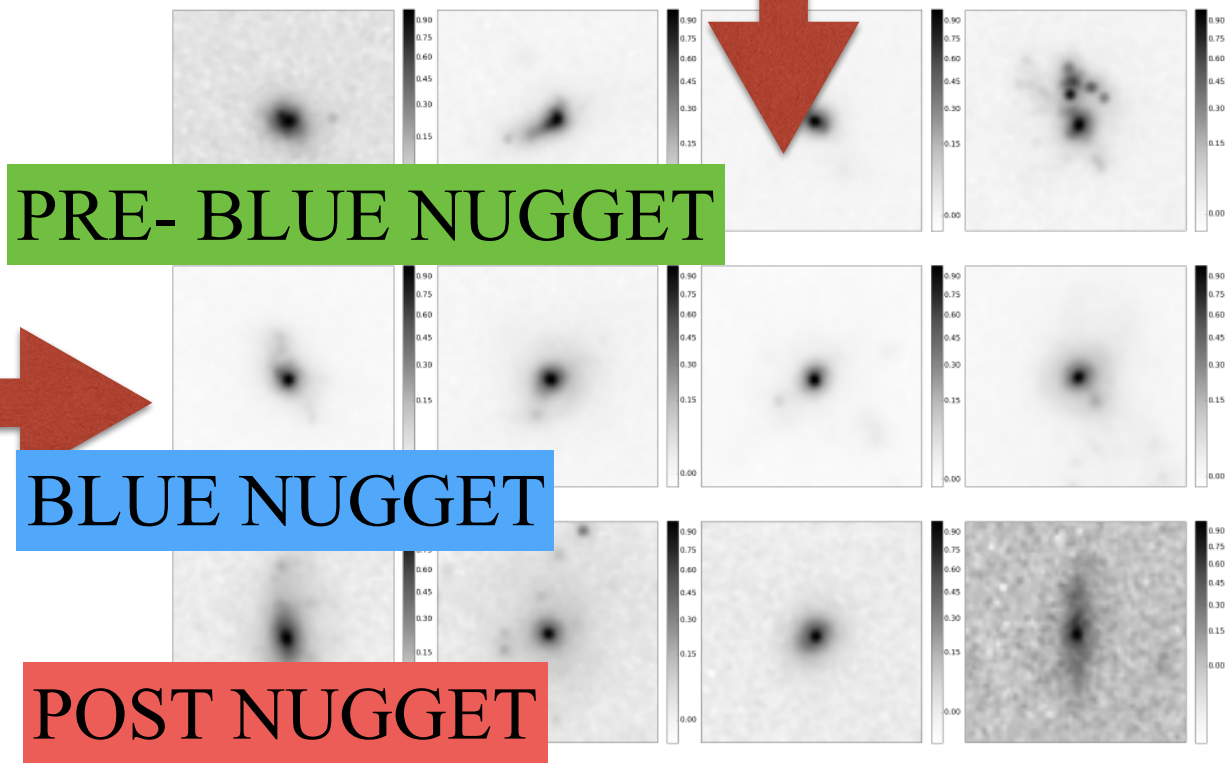
GENERATE MOCK IMAGES AS OBSERVED BY HST



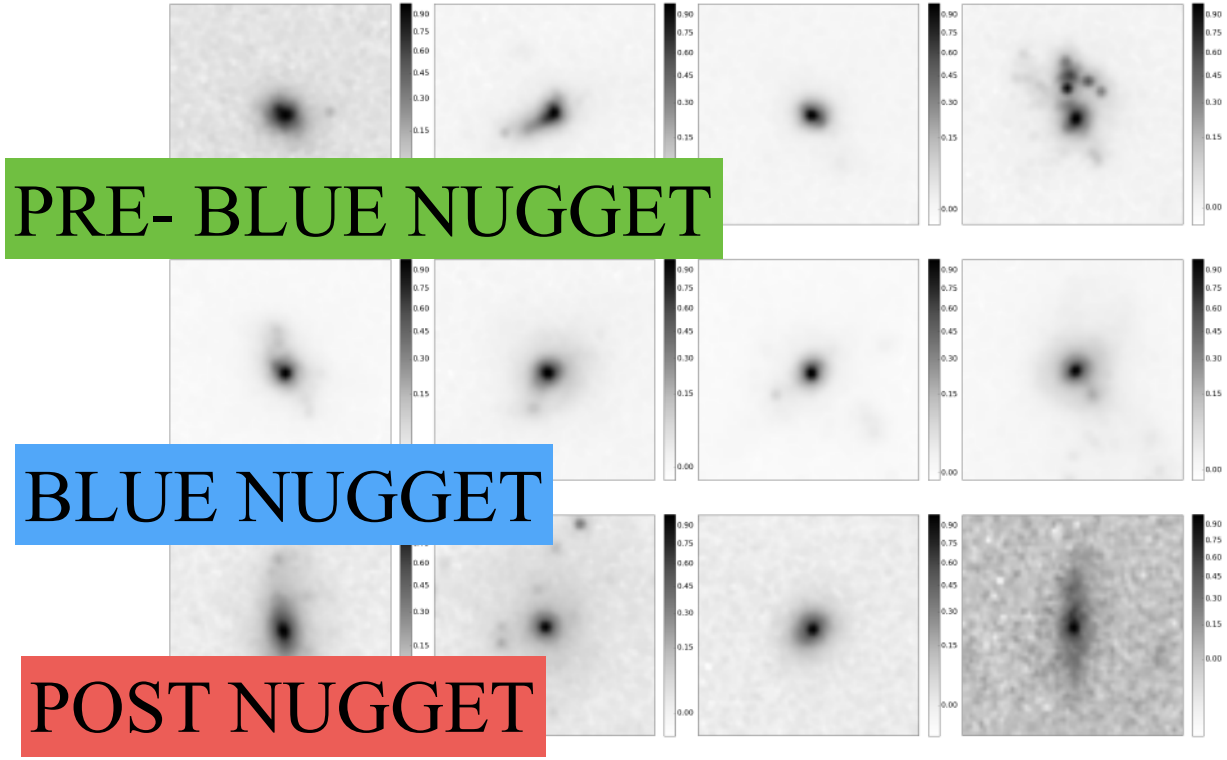
USE THE FORMATION HISTORY OF EACH GALAXY TO LABEL IMAGES …



[COSMOLOGICAL SIMULATION]



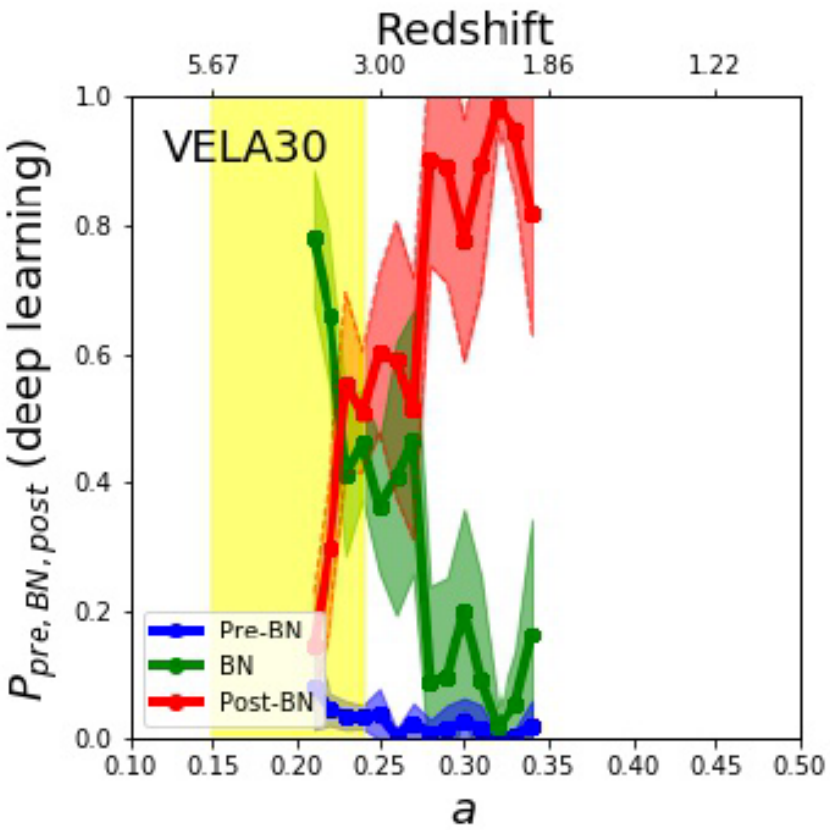
GENERATE MOCK IMAGES AS OBSERVED BY HST

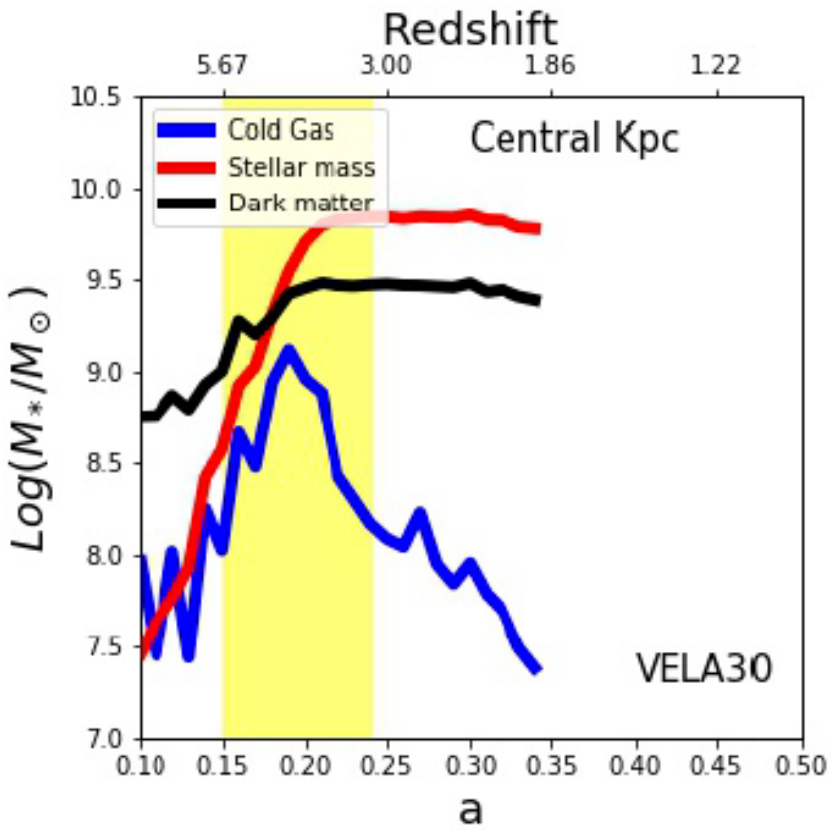


CNN

ARE THERE ANY DISTINGUISHABLE FEATURES GIVEN THE OBSERVATIONS?

YES!



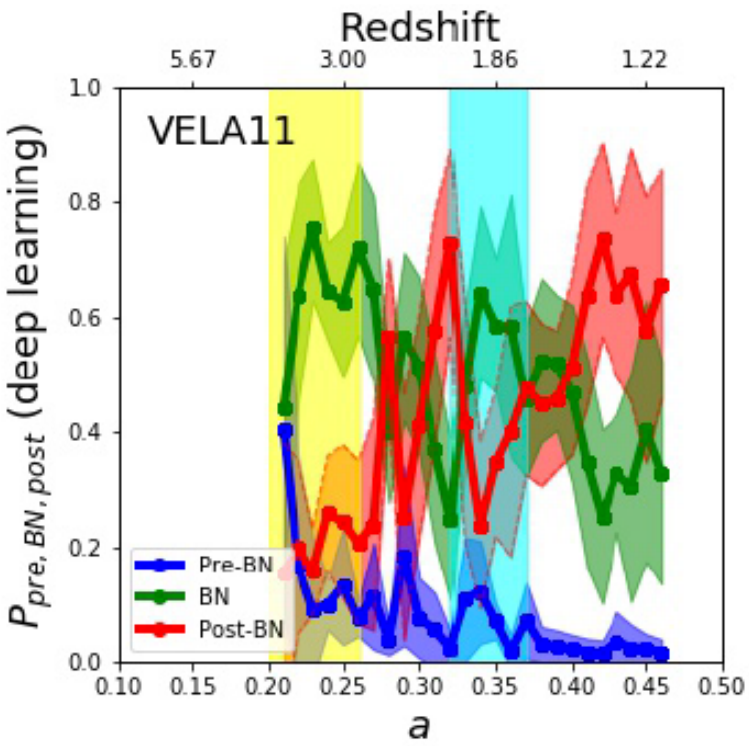


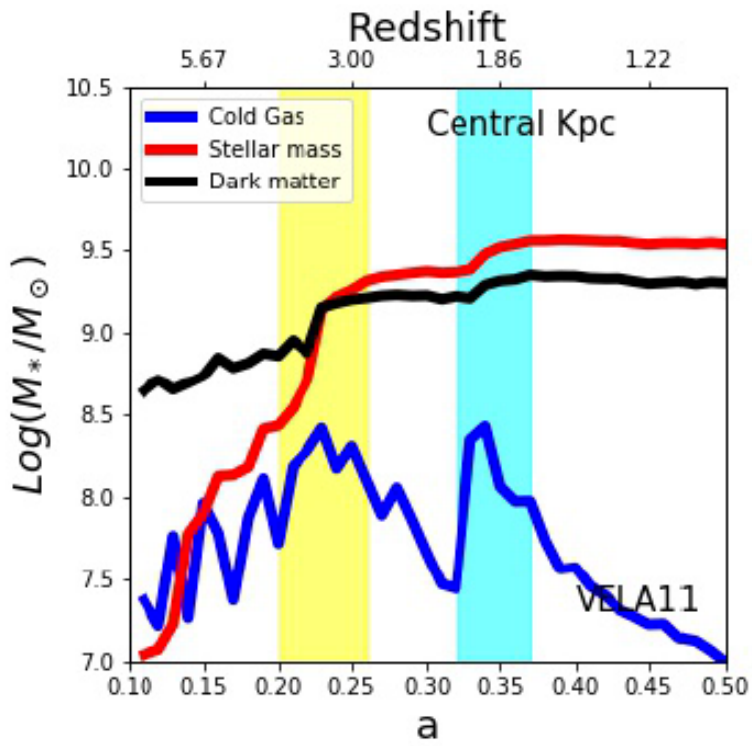
DEEP LEARNING

SIMULATION

Huertas-Company+18

YES!



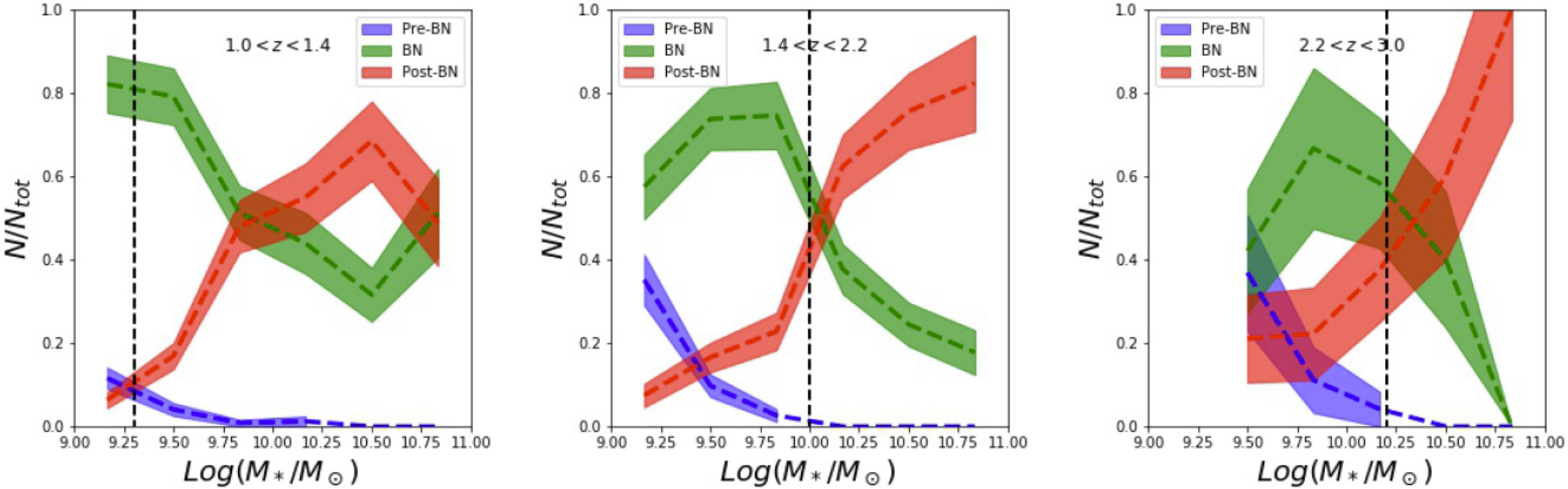


DEEP LEARNING

SIMULATION

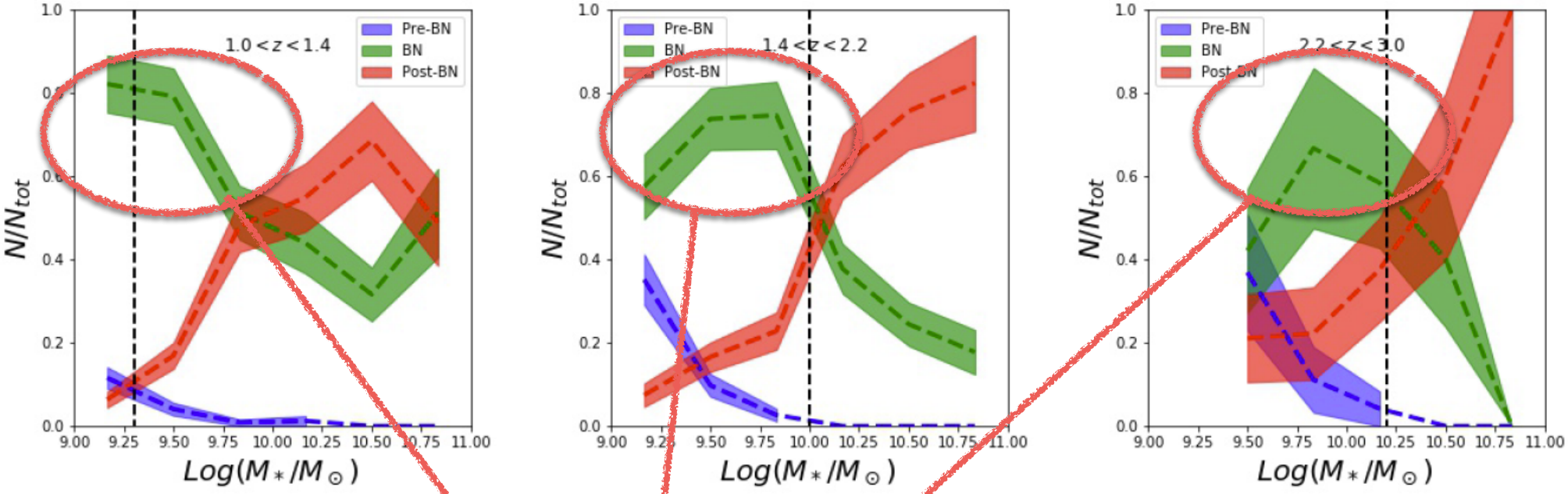
Huertas-Company+18

AND THEN WE CAN LOOK FOR THESE TRACERS IN THE DATA…



Huertas-Company+18

AND THEN WE CAN LOOK FOR THESE TRACERS IN THE DATA…



DETECTION OF A CHARACTERISTIC MASS FOR THE BLUE NUGGET PHASE

Huertas-Company+18

HYPER PARAMETER SEARCH

HOW DO I SETUP MY CNN?

UNFORTUNATELY, THERE IS NO MAGIC RECIPE

HOW DO I SETUP MY CNN?

UNFORTUNATELY, THERE IS NO MAGIC RECIPE

SOME GENERAL ADVICES:

- START WITH SOMETHING THAT WORKS
- CAPACITY INCREASES WITH LAYERS, CHANNELS, RECEPTIVE FIELD
- USE PRIOR KNOWLEDGE OF THE EXPECTED SCALE CONTAINING MEANINGFUL INFORMATION
- GRID SEARCH…

CREDIT

HOW DO I SETUP MY CNN?

TYPICAL PARAMETERS

NUMBER OF HIDDEN LAYERS?

- START WITH FEW LAYERS AND INCREASE COMPLEXITY IF NEEDED
- ADD MORE LAYERS, CHECK PERFORMANCE
- ADD MORE NEURONS, CHECK PERFORMANCE

CREDIT

HOW DO I SETUP MY CNN?

TYPICAL PARAMETERS

ACTIVATION FUNCTION

- PRIORITY TO ReLU
- TRY EVENTUALLY LeakyReLu, PreLU

CREDIT

HOW DO I SETUP MY CNN?

TYPICAL PARAMETERS

OPTIMIZER

- NOT REALLY A RULE FOR THIS…
- SGD, ADAM THE ONES I MOSTLY USED. ADAM MORE ROBUST TO “NAN” LOSSES

CREDIT

HOW DO I SETUP MY CNN?

TYPICAL PARAMETERS

LEARNING RATE

- THIS IS ONE OF THE MOST TWEAKED PARAMETERS
- THE LEARNING RATE SHOULD BE LARGE AT THE BEGINNING AND SMALL TOWARDS THEN WHEN CLOSER TO THE MINIMUM

CREDIT

HOW DO I SETUP MY CNN?

TYPICAL PARAMETERS

LEARNING RATE

- THIS IS ONE OF THE MOST TWEAKED PARAMETERS
- THE LEARNING RATE SHOULD BE LARGE AT THE BEGINNING AND SMALL TOWARDS THEN WHEN CLOSER TO THE MINIMUM

CREDIT

HOW DO I SETUP MY CNN?

LEARNING RATE

- THE DECAY OF THE LEARNING RATE CAN BE SET :
  - STEP DECAY [DECAY BY FIX AMOUNT EVERY FEW EPOCHS]
  - EXPONENTIAL DECAY $\lambda = \lambda_0 e^{-kt}$
  - INVERSE DECAY $\lambda = \frac{\lambda_0}{1 + kt}$

CREDIT

HOW DO I SETUP MY CNN?

LEARNING RATE

- THE DECAY OF THE LEARNING RATE CAN BE SET :
  - STEP DECAY [DECAY BY FIX AMOUNT EVERY FEW EPOCHS]
  - EXPONENTIAL DECAY $\lambda = \lambda_0 e^{-kt}$
  - INVERSE DECAY $\lambda = \frac{\lambda_0}{1 + kt}$

CREDIT

- WHEN OPTIMIZING THE NETWORK
  - FIRST PERFORM A COARSE SEARCH, USUALLY RECYCLING SOMETHING THAT WORKED, RUNE FEW EPOCH
  - THE, LONGER TRAINING, FINER SEARCH