PART V: SOME PRACTICAL CONSIDERATIONS

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

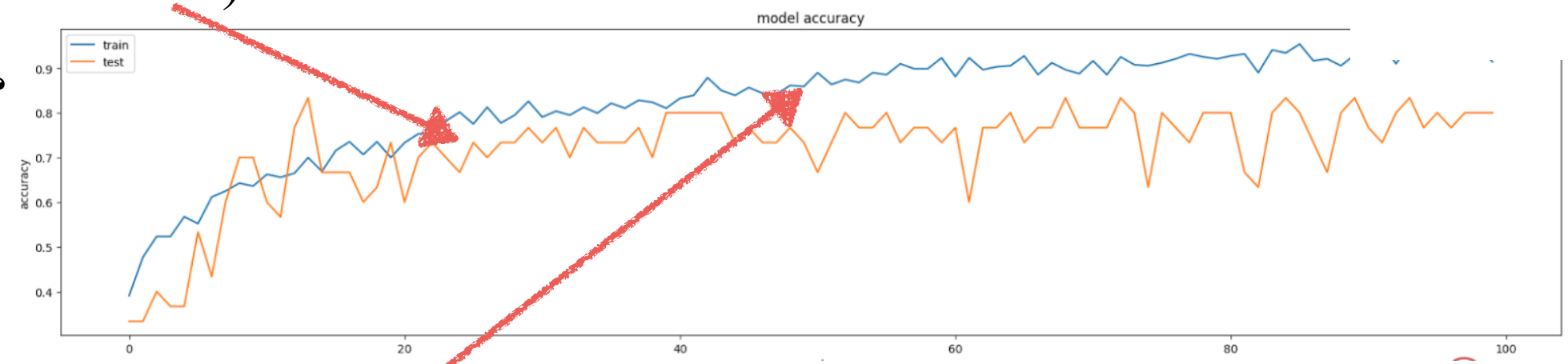
… “IF THE VALIDATION LOSS DOES NOT CHANGE MORE
THAN EPSILON OVER THE LAST N EPOCHS”…

KEEPING TRACK OF PERFORMANCE DURING TRAINING

THE LEARNING HISTORY

[MODEL ACCURACY AS A FUNCTION OF EPOCH]

(validation)



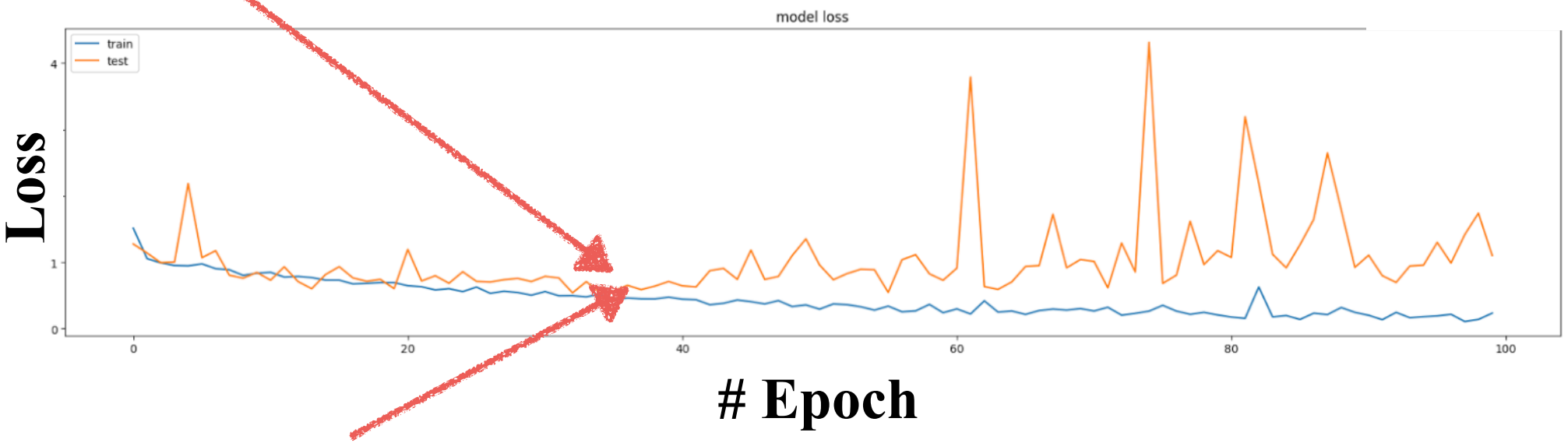
(training)

KEEPING TRACK OF PERFORMANCE DURING TRAINING

THE LEARNING HISTORY

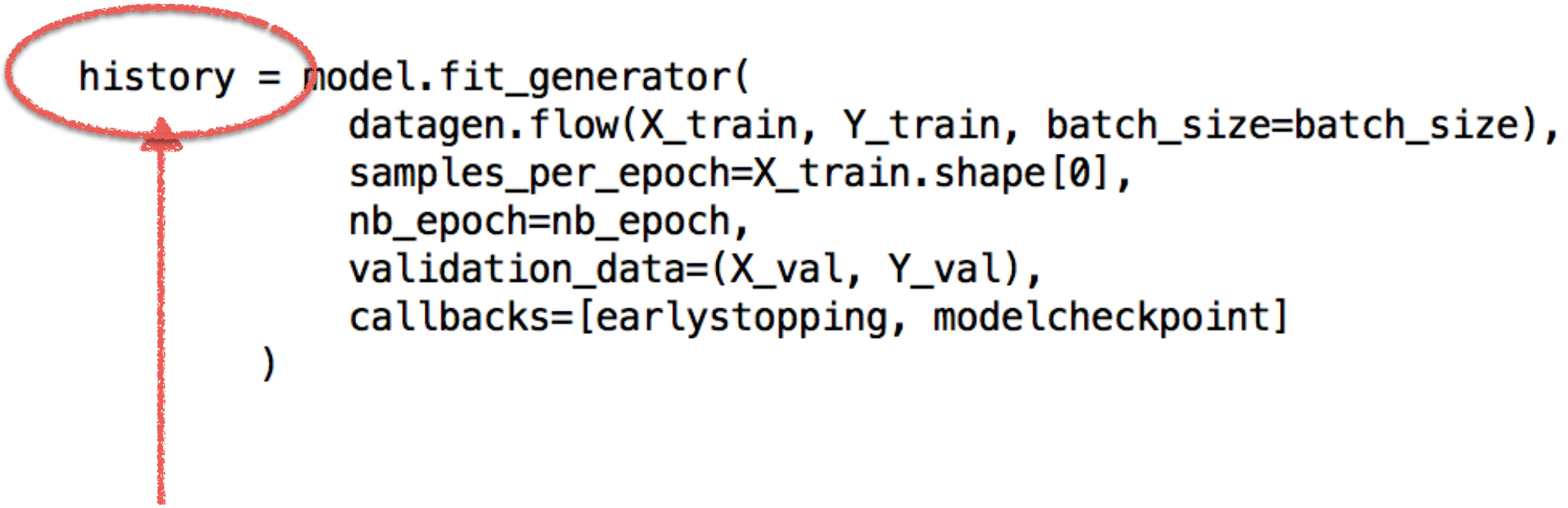
[MODEL LOSS AS A FUNCTION OF EPOCH]

(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

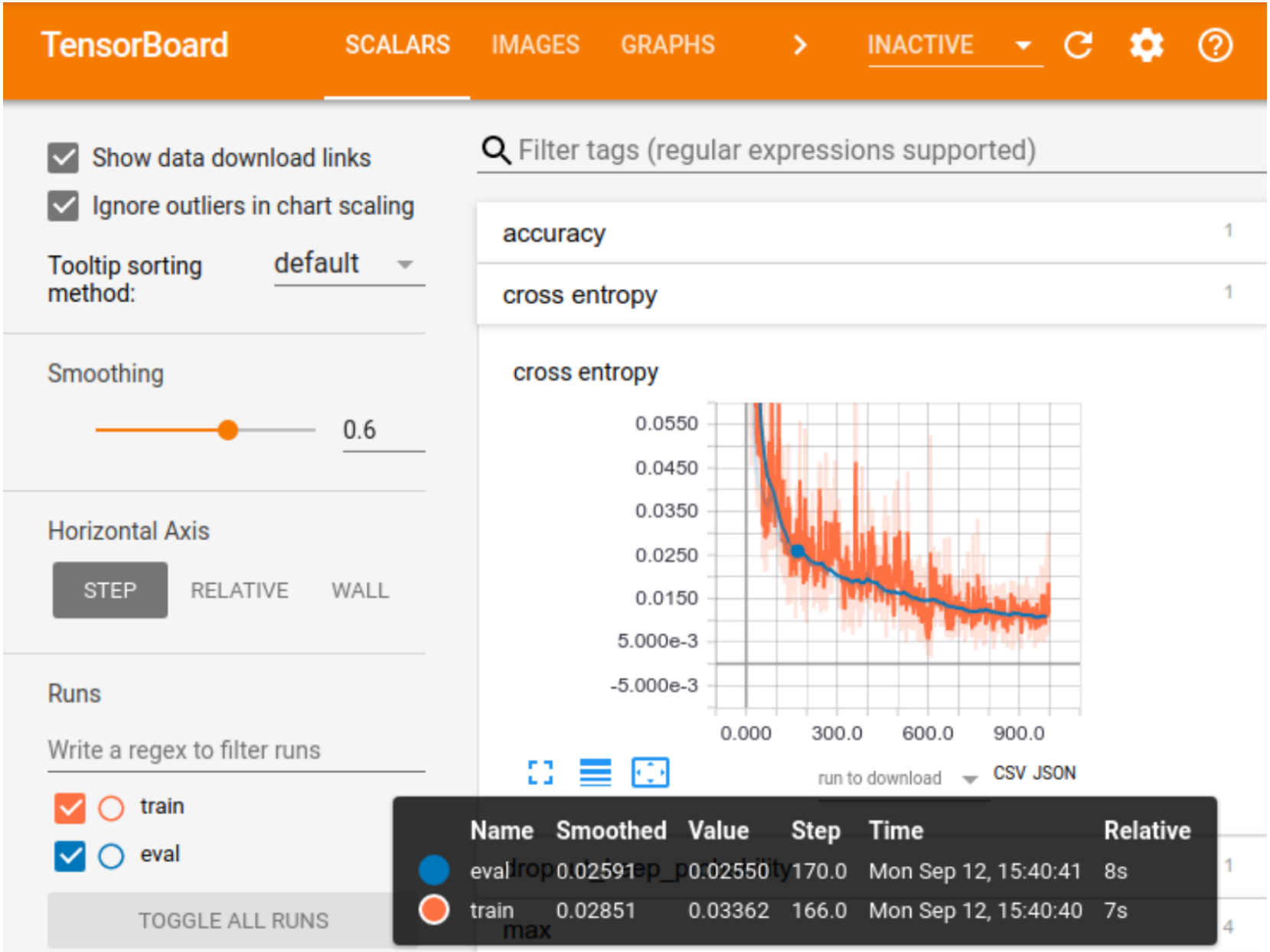
```
from keras.models import Sequential

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

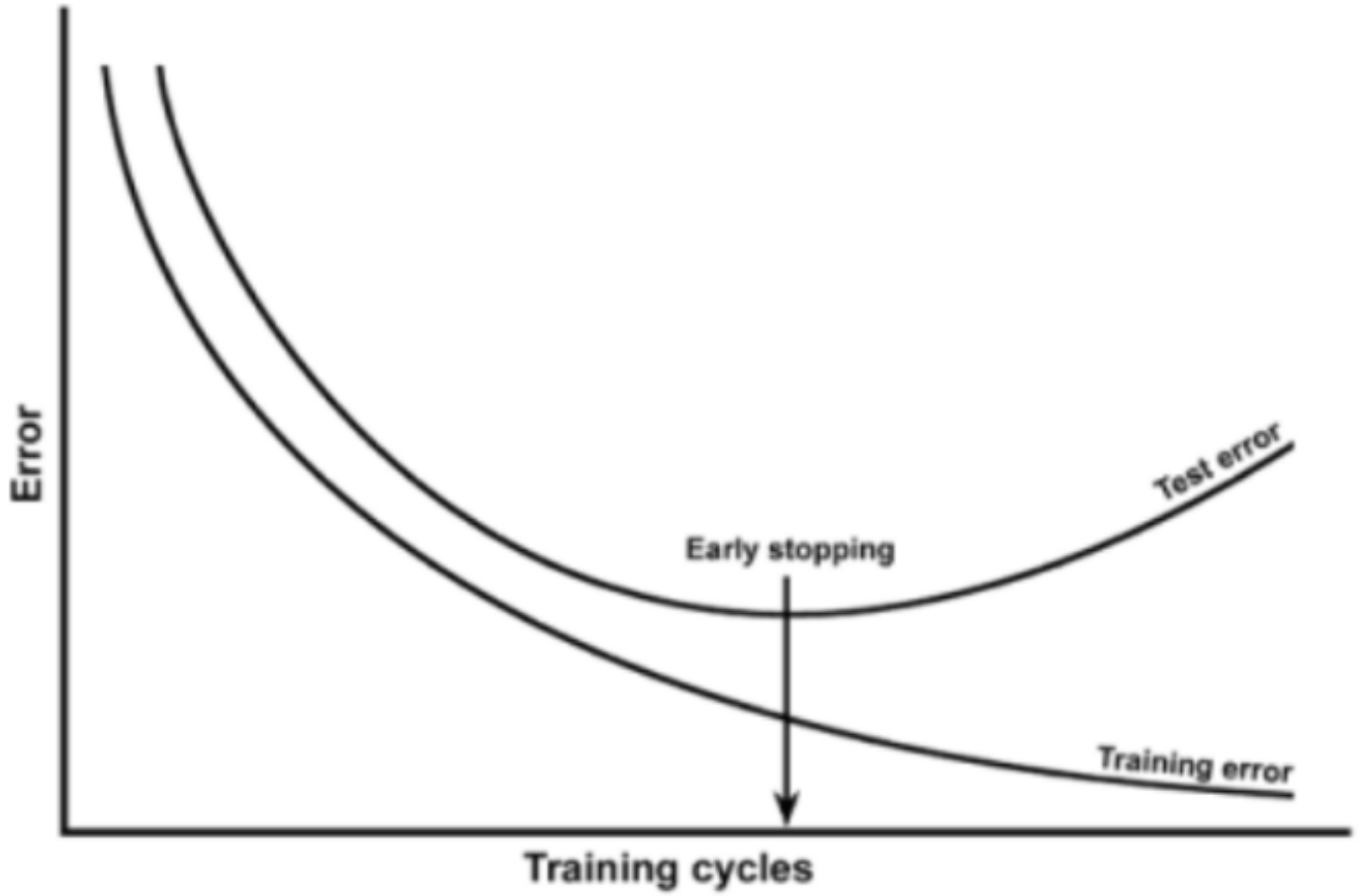
set patience (number of epochs to monitor)

```
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])
```

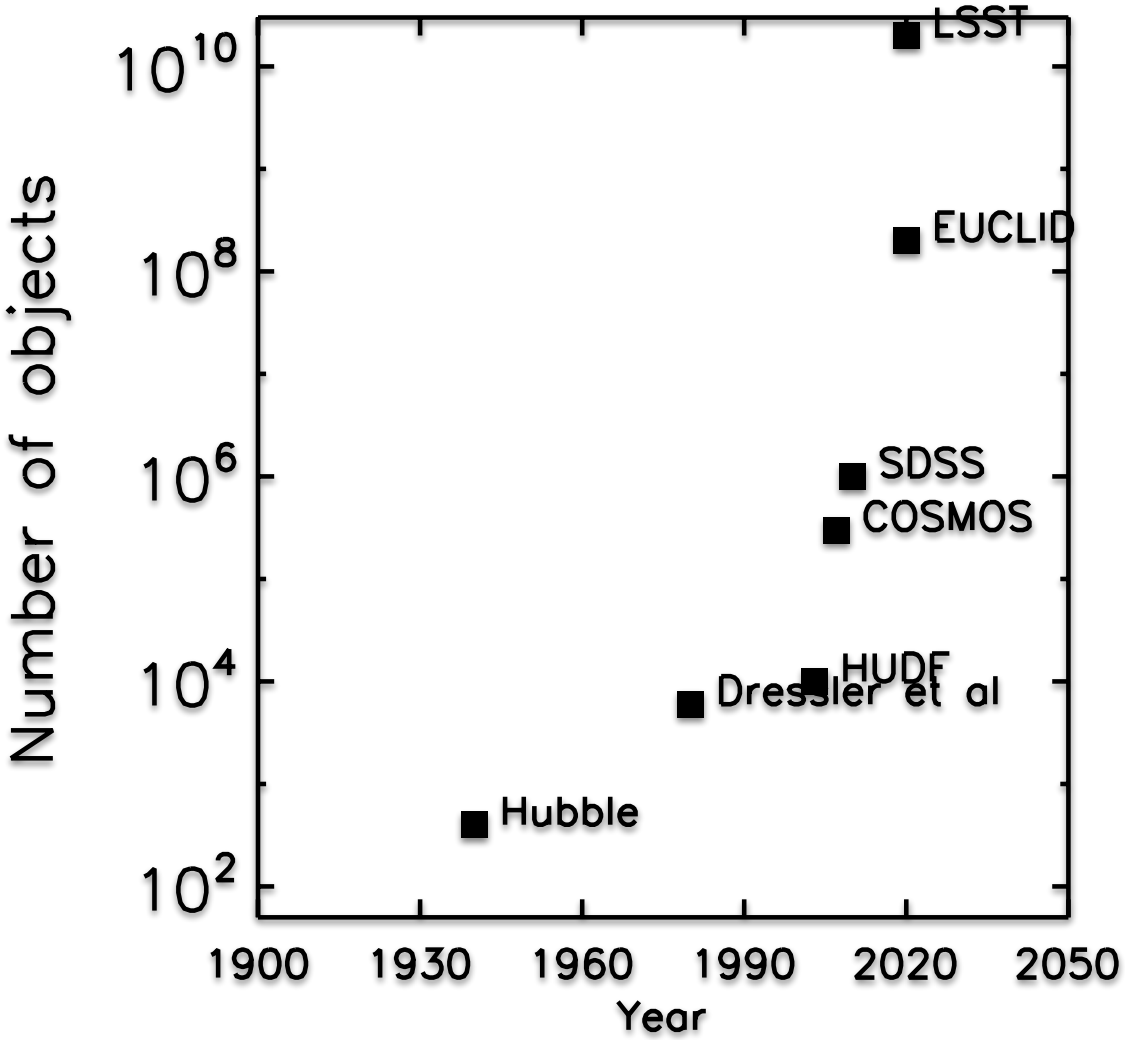
parameter to monitor [typically validation loss]

EARLY STOPPING HELPS ALSO PREVENTING OVERFITTING…



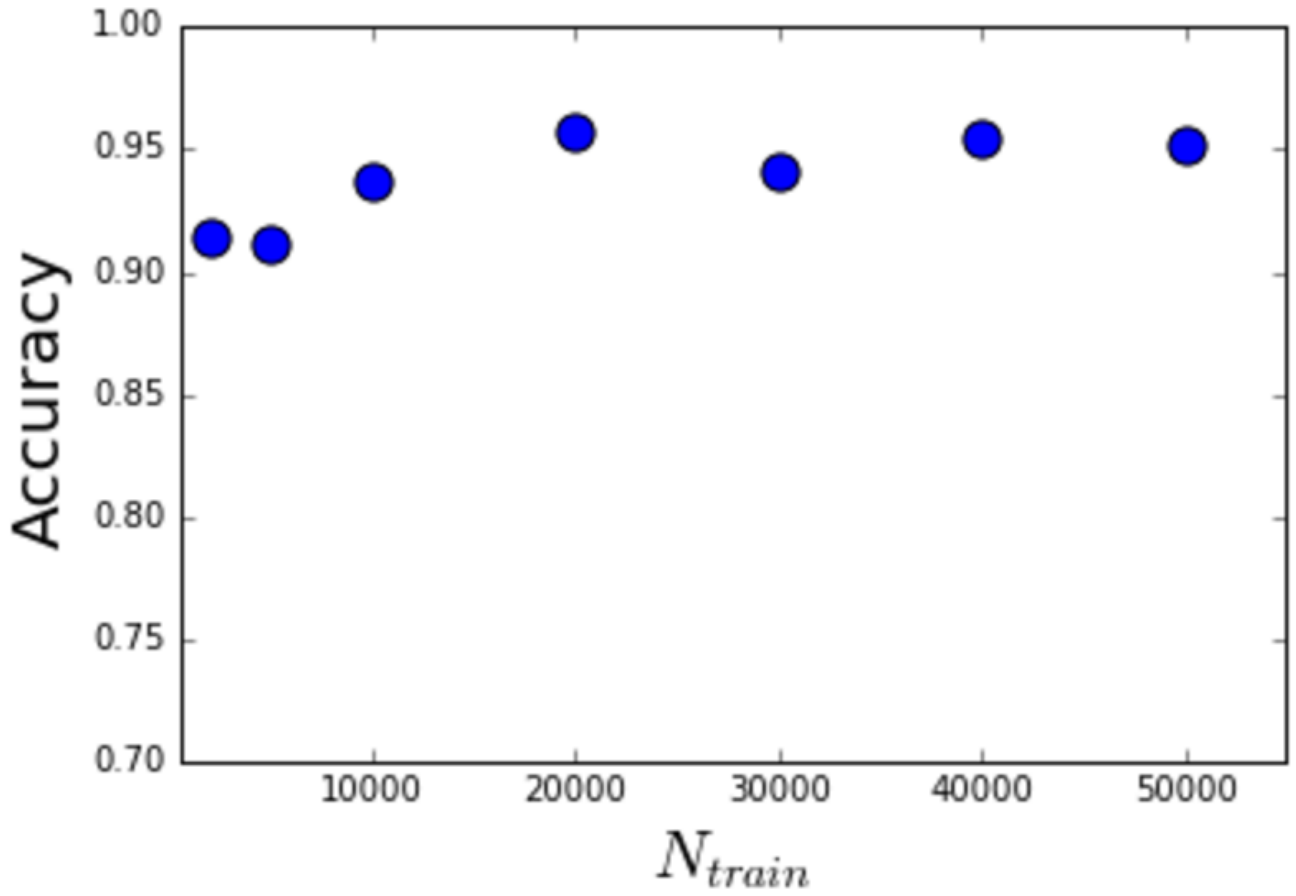
HOW LARGE DOES MY DATASET NEED TO BE?

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



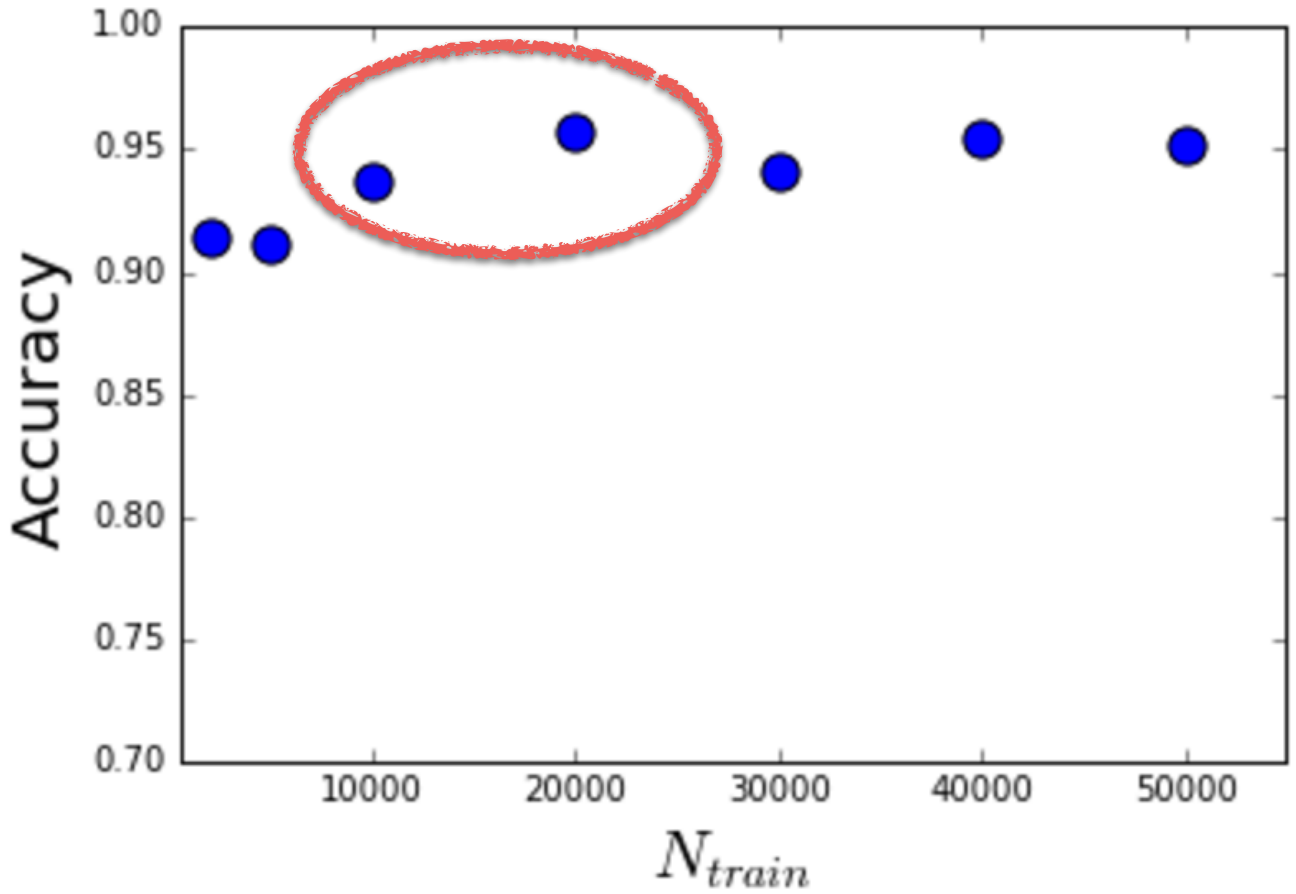
THE SIZE OF DATASETS
IS
INCREASING FAST BUT
WE DO
NOT ALWAYS HAVE
MILLIONS OF
LABELLED
EXAMPLES TO TRAIN
…

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…

SOME TRICKS FOR “SMALL” DATASETS

DATA AUGMENTATION

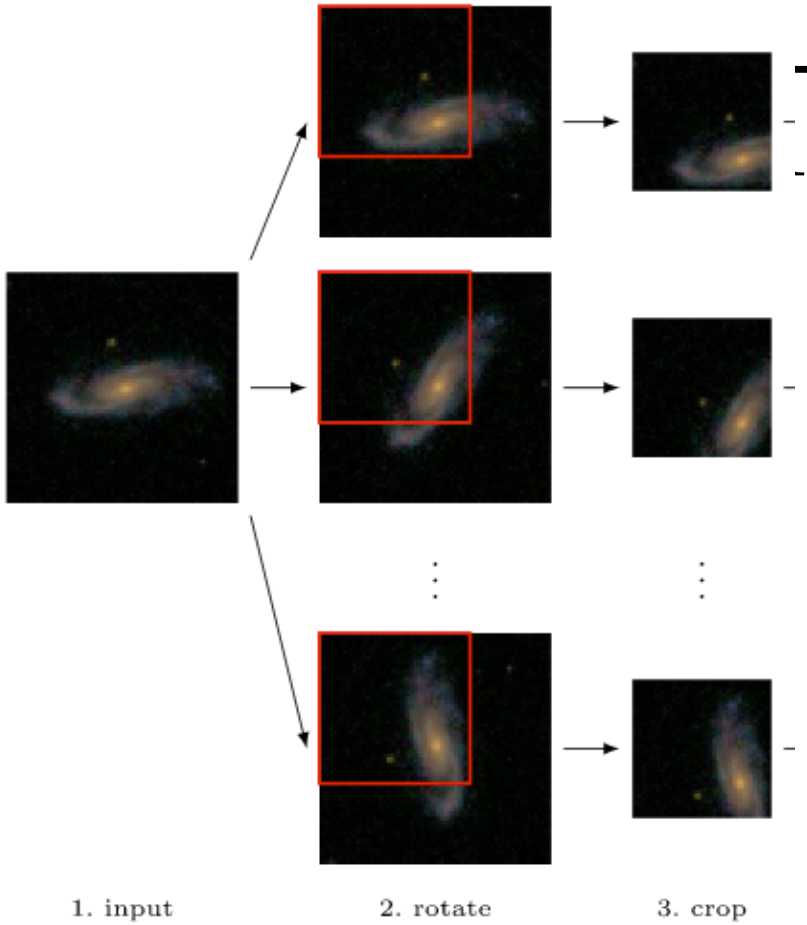
ANOTHER WAY TO REDUCE OVER-FITTING IS TO "AUGMENT" THE SIZE OF THE DATASET AVAILABLE FOR TRAINING

FOR MANY APPLICATIONS THE CLASSIFICATION SHOULD BE INDEPENDENT TO:

- TRANSALTIONS
- ROTATIONS
- SCALINGS

ETC…

DATA [illegible]TATION



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 …

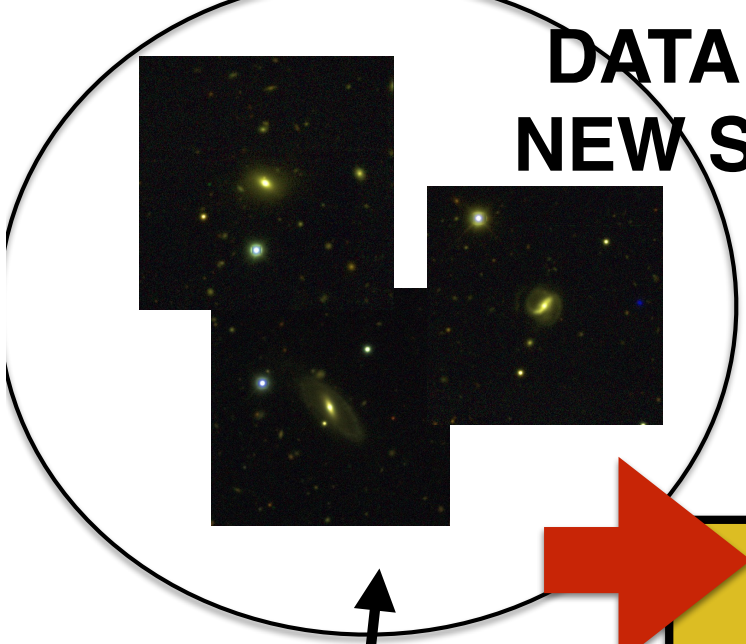
DOMAIN ADAPTATION
(or knowledge transfer)

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?

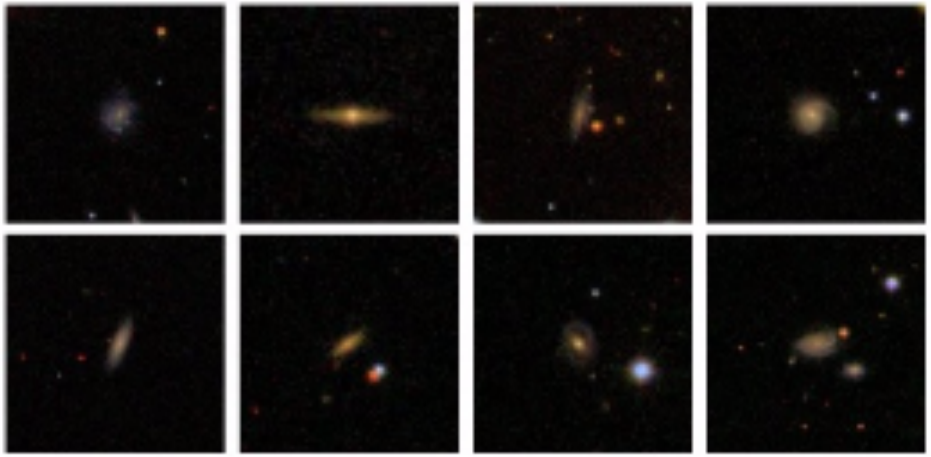
DEEP-LEARNING BASED MACHINE

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

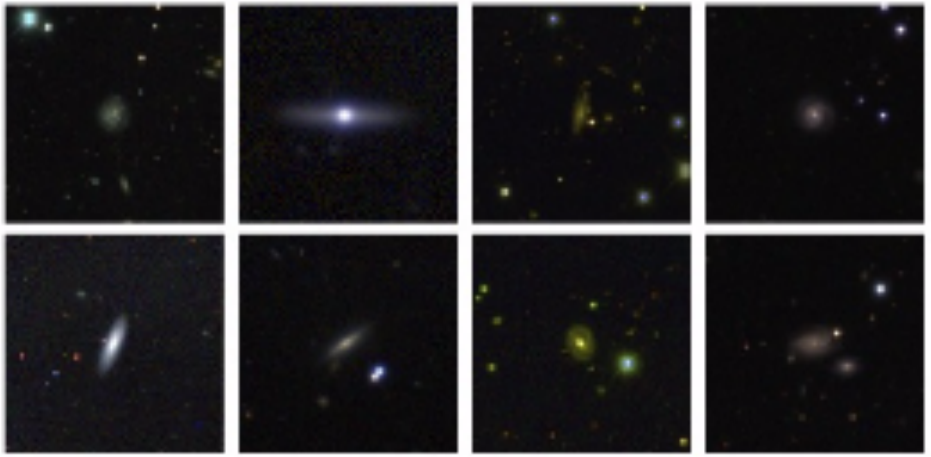
Transfer knowledge?

Human classifications from existing survey

Dominguez-Sanchez+18b



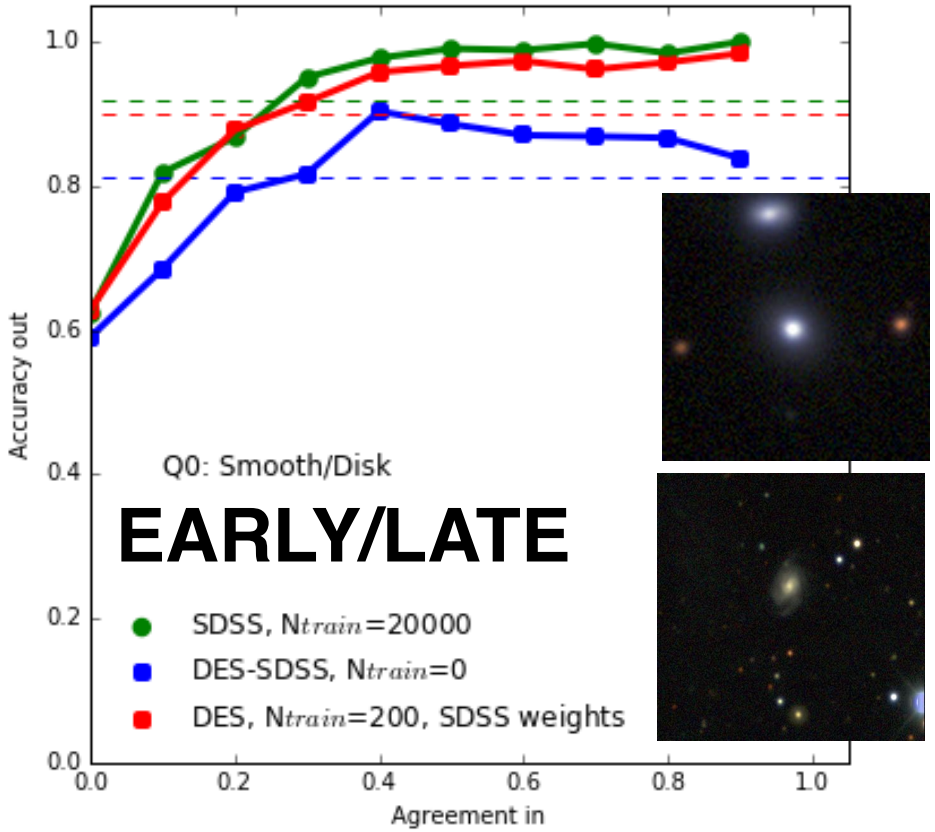
SDSS

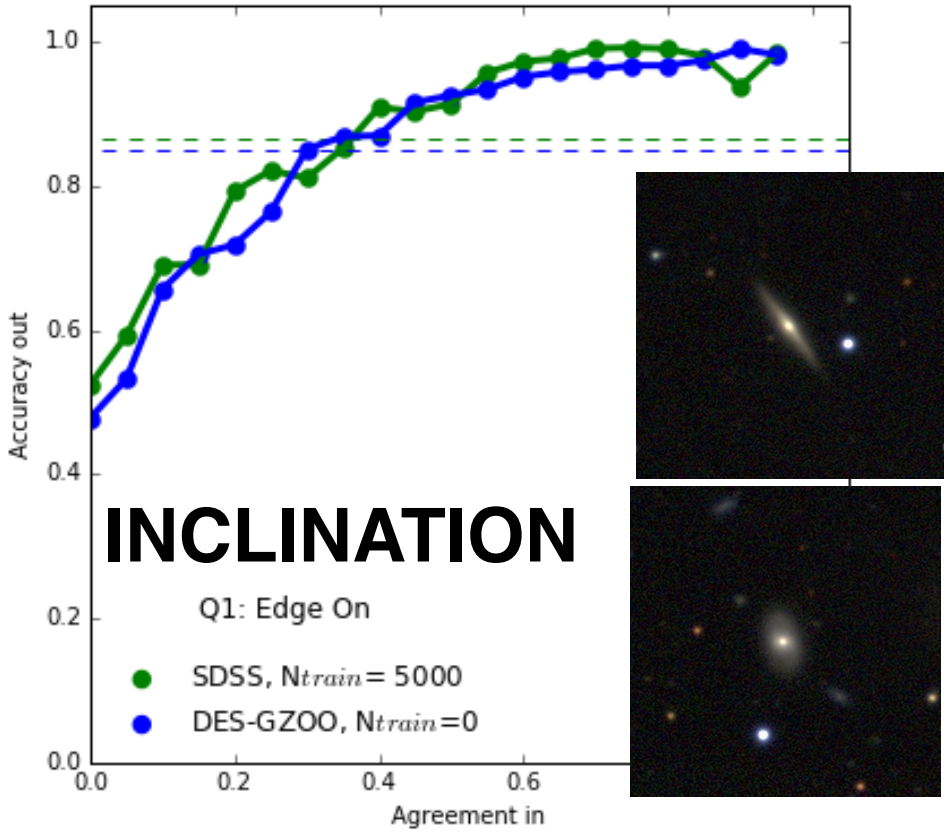


DES

Dominguez-Sanchez+18b

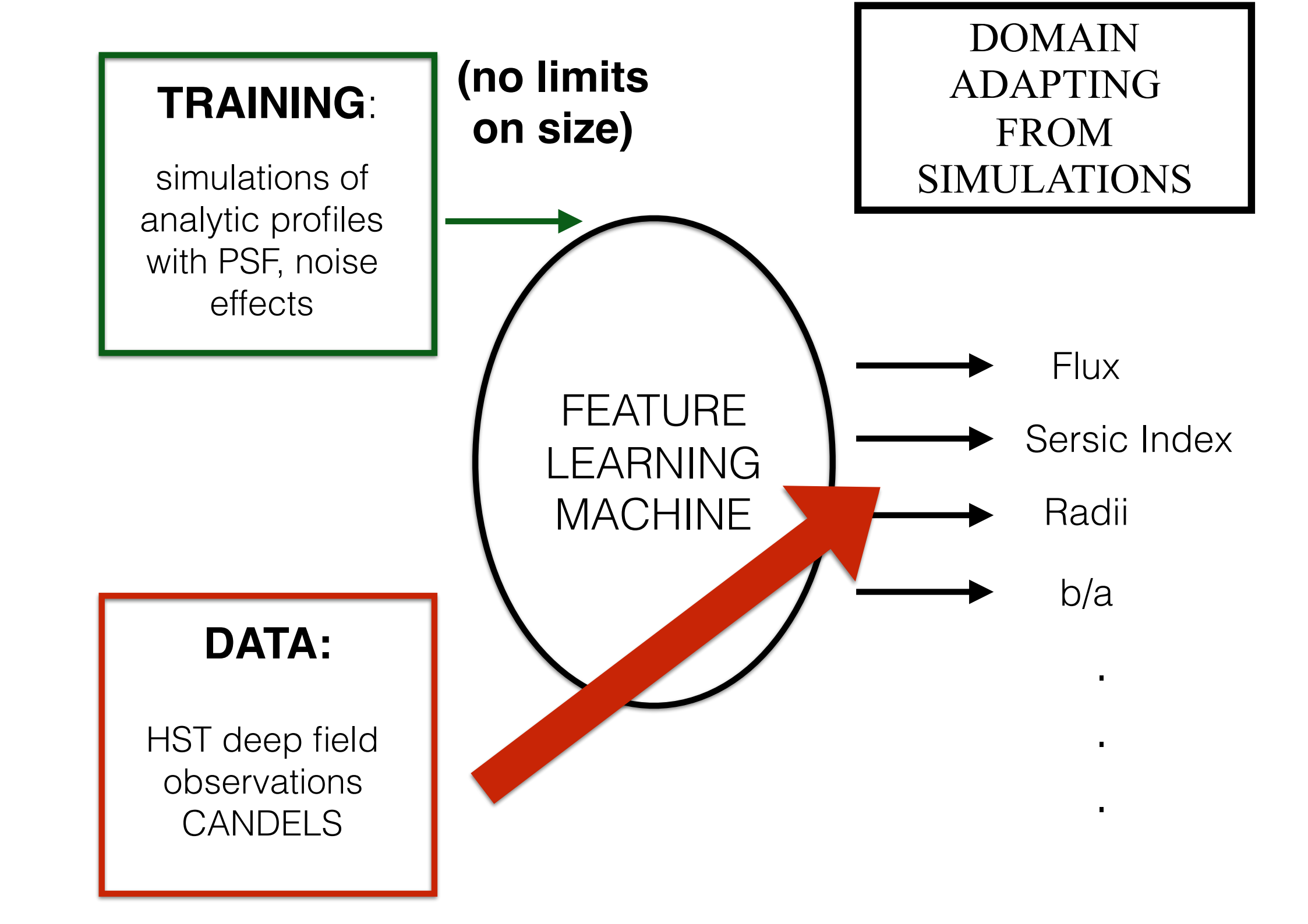
Knowledge Transfer from SDSS to DES

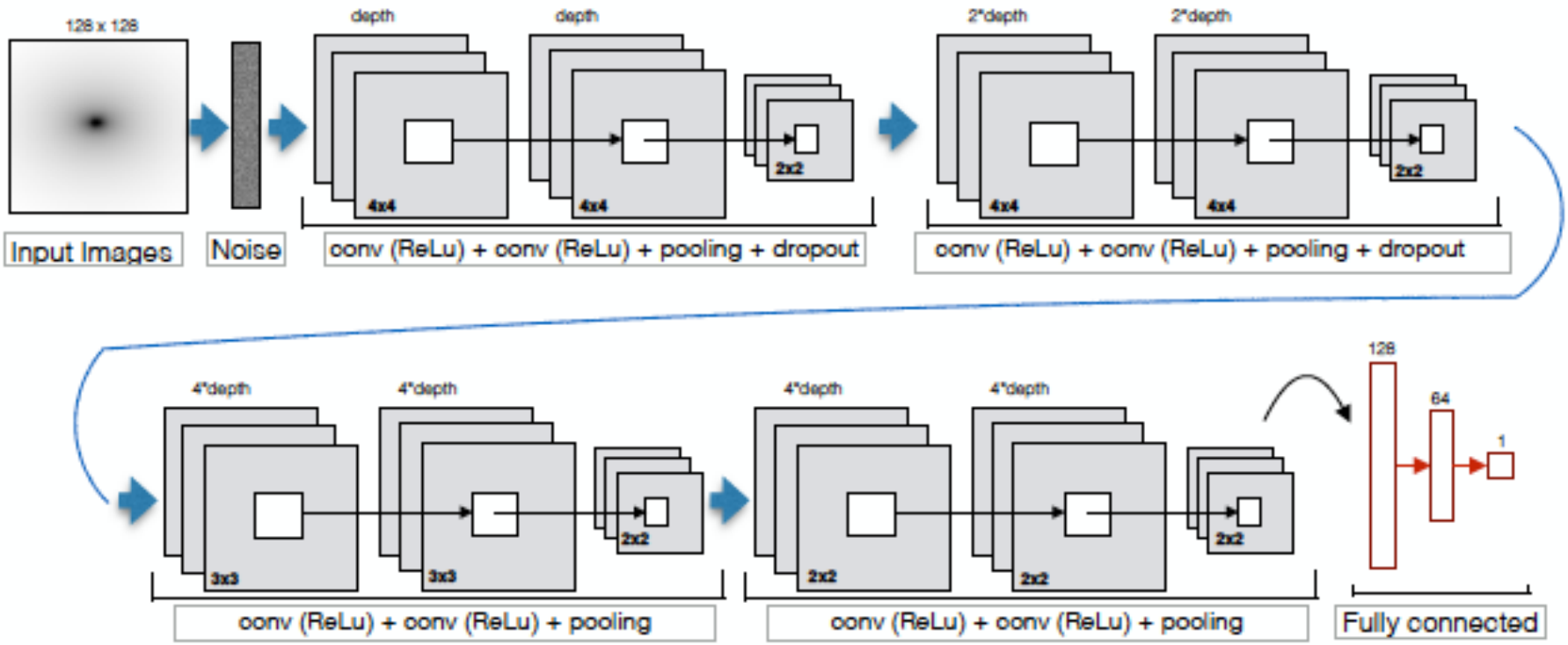




Only 200 (**1%!**) objects classified in DES are needed to reach an accuracy >90% if a machine trained on the SDSS is used

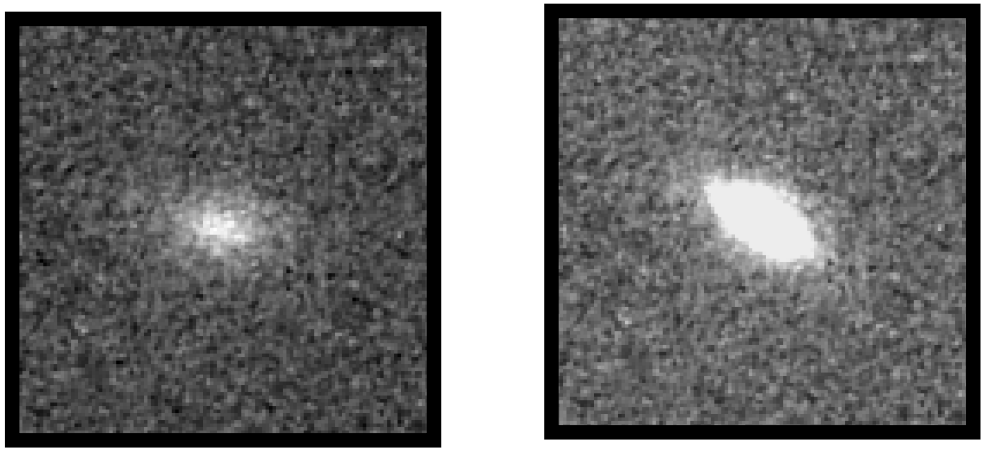
For some properties, i.e. EDGE-ON galaxies. No training at all is needed to go from SDSS to DES


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

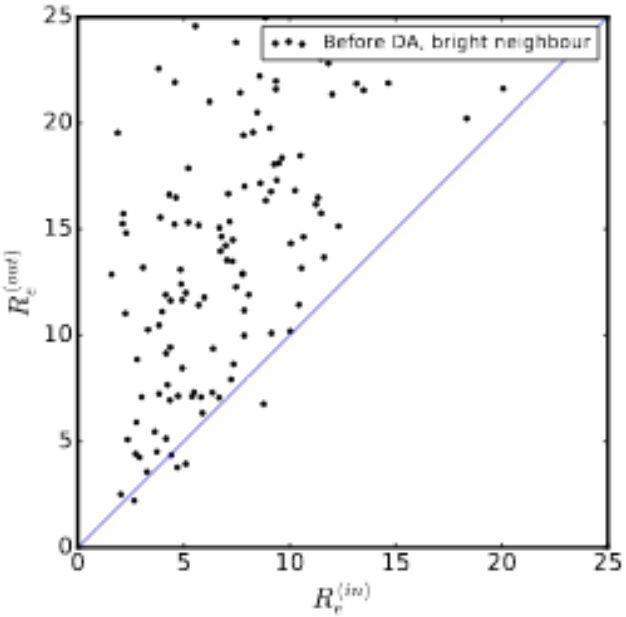
Standard analytic profiles

- 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!

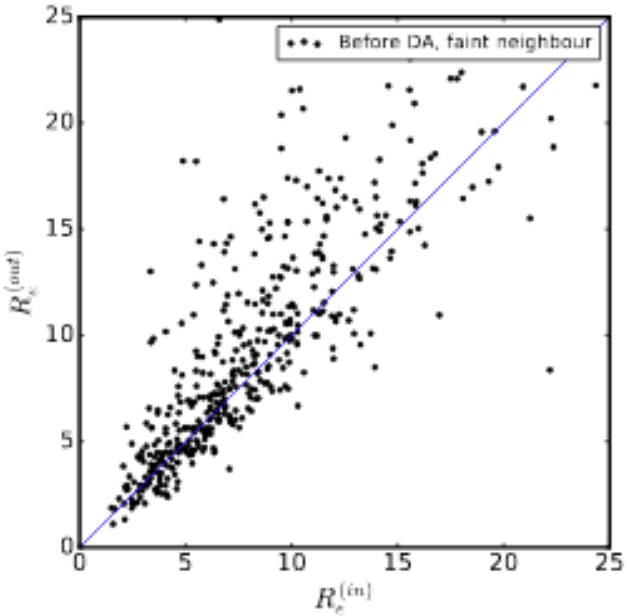


MORPHOMETRY OF REAL GALAXIES TRAINED ON ANALYTIC PROFILES

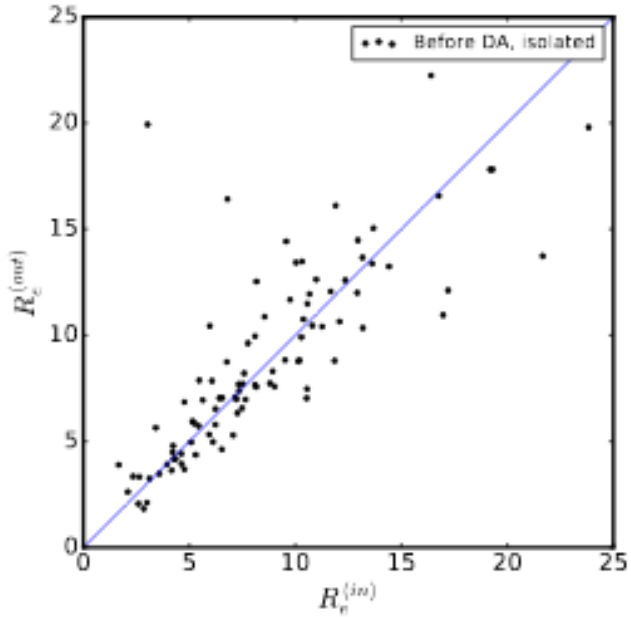
BRIGHT NEIGH. **FAINT NEIGH.** **ISOLATED**



(c) BDA bright neigbours



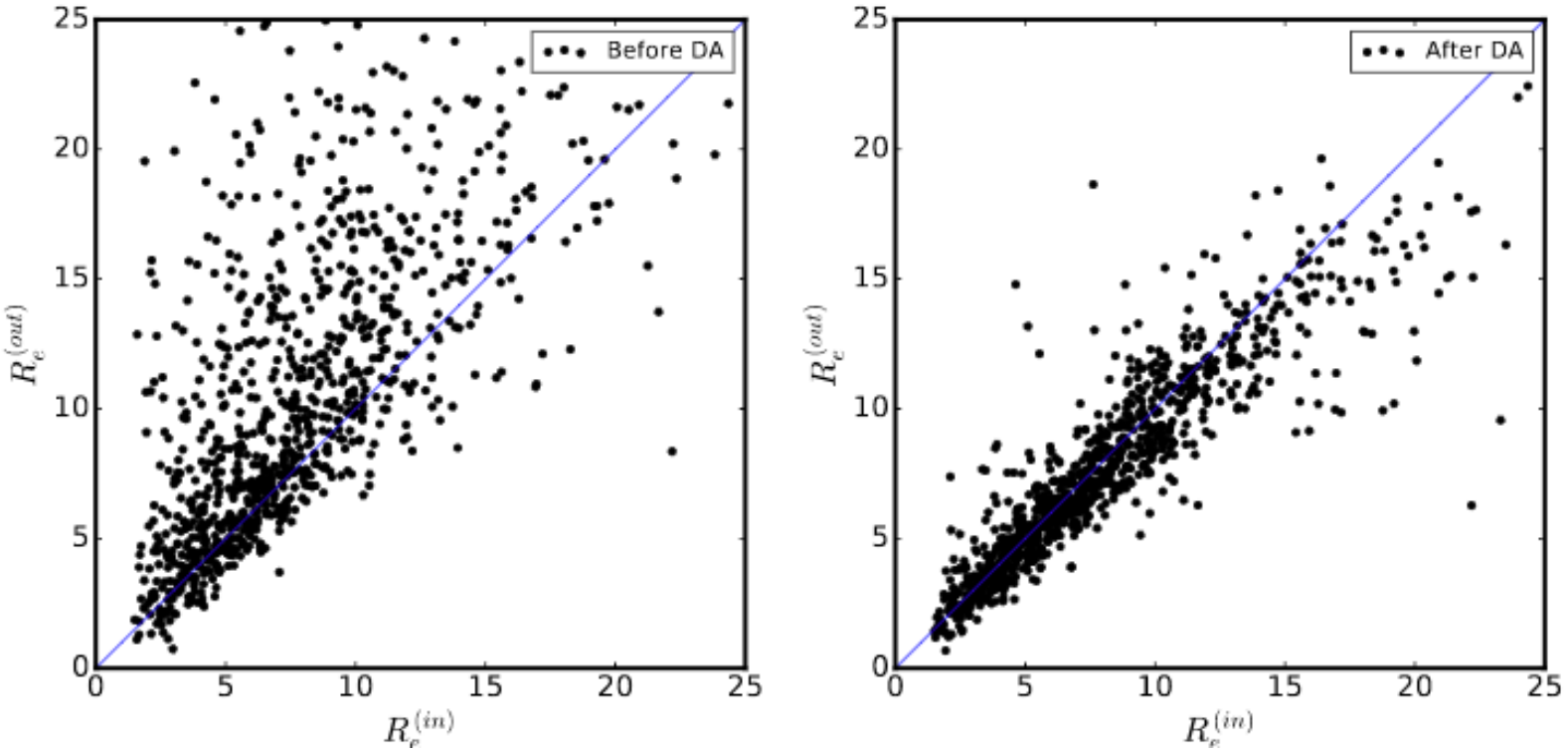
(d) BDA faint neigbours



(e) BDA isolated galaxies

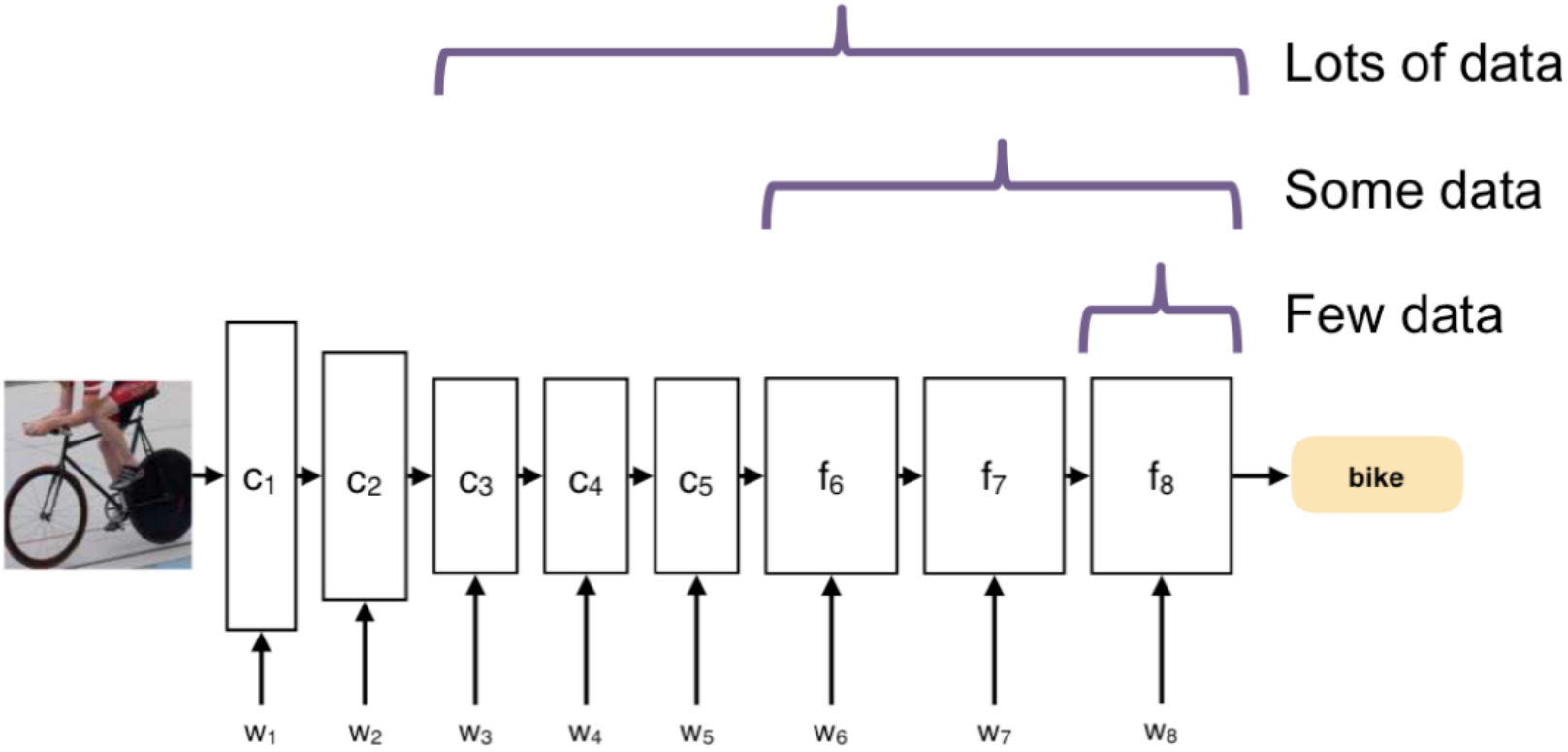
Tuccillo+18

DOMAIN ADAPTATION: 0.1% OF “REAL” GALAXIES



Tuccillo+18

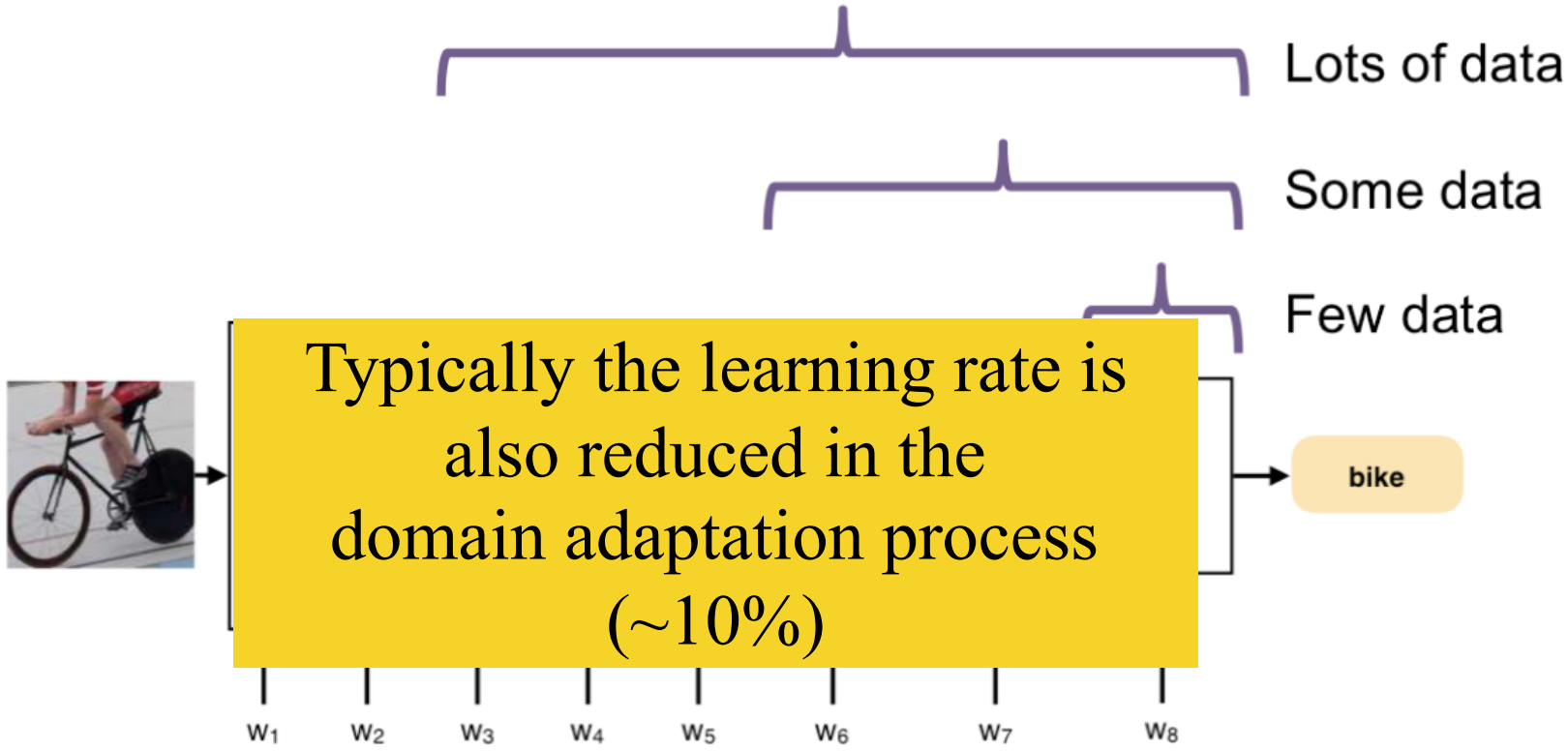
HOW MANY LAYERS “DOMAIN ADAPT” ?



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" ?



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")

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]
          )
```

load weights of previous training

start training

KERAS IMPLEMENTATION

"PARTIAL" DOMAIN ADAPTATION

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

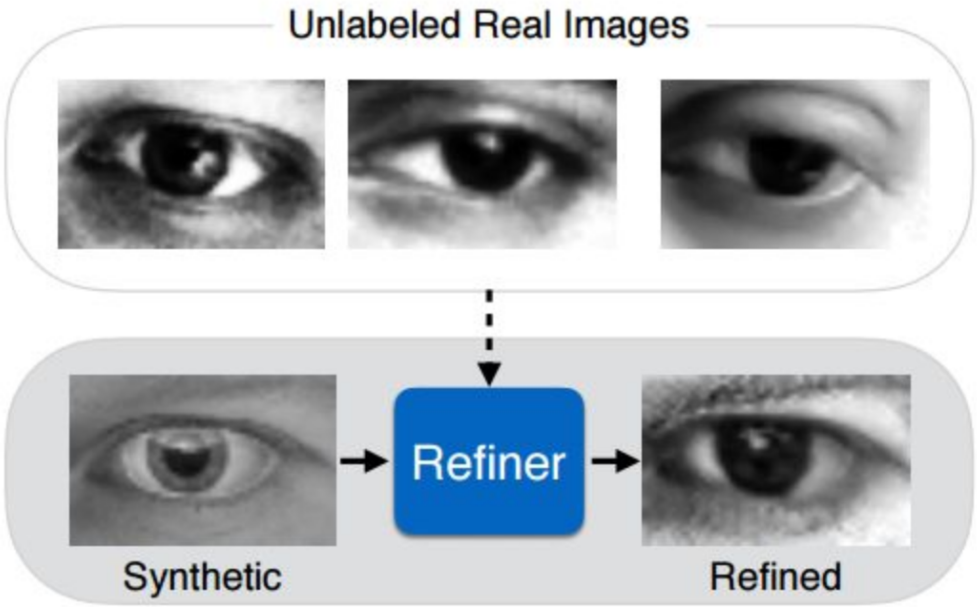
```
keras.optimizers.Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=None, decay=0.0, amsgrad=False)
```

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

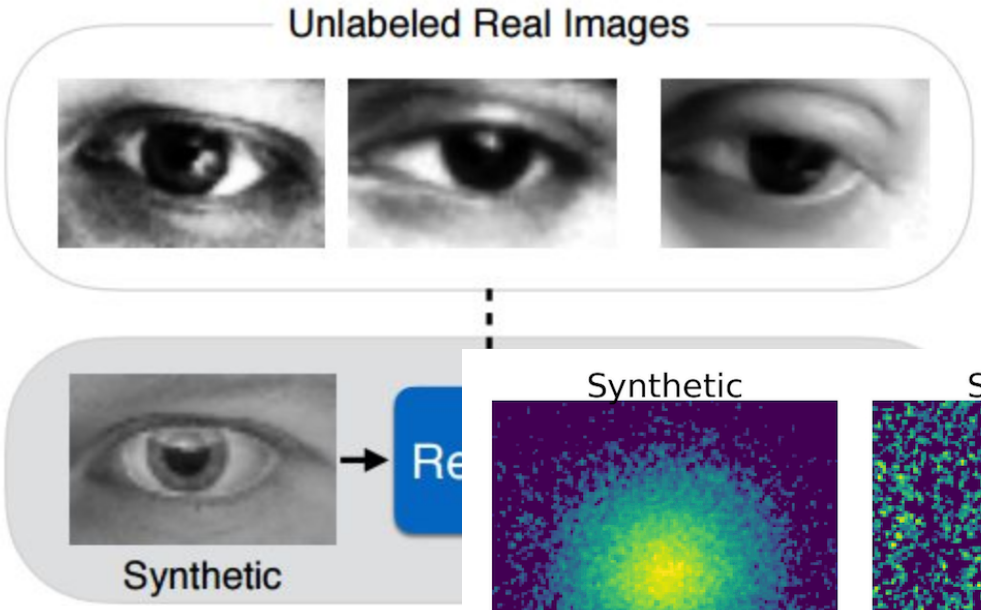
```
all_layers = model.layers
for i in range(model.layers.index(mid_start)):
    all_layers[i].trainable = False
```

Freeze some layers

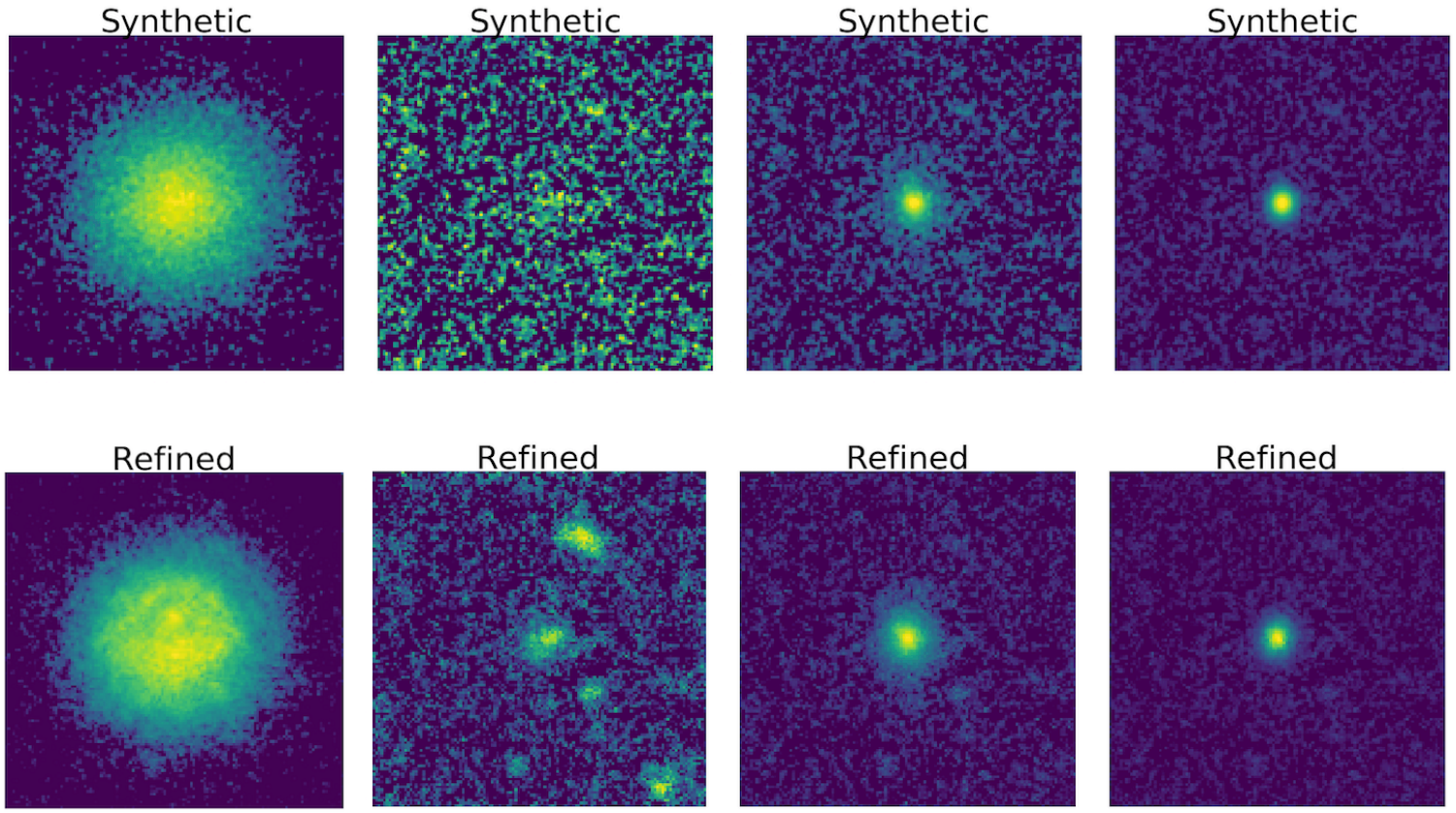
USE GANs?



USE GANs?



VERY PRELIMINAR WORK IN CANDELS



MARGALEF,MHC+ in prep

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

VISUALIZING CNNs
[what happens inside a CNN?]

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

ESSENTIALLY 2 APPROACHES

PERTURBATION

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

BACK-PROPAGATION

THE BASIC IDEA IS TO REVERSE THE NETWORK TO

OCCLUSION SENSITIVITY

INCEPTIONISM

ESSENTIALLY 2 APPROACHES

PERTURBATION

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

BACK-PROPAGATION

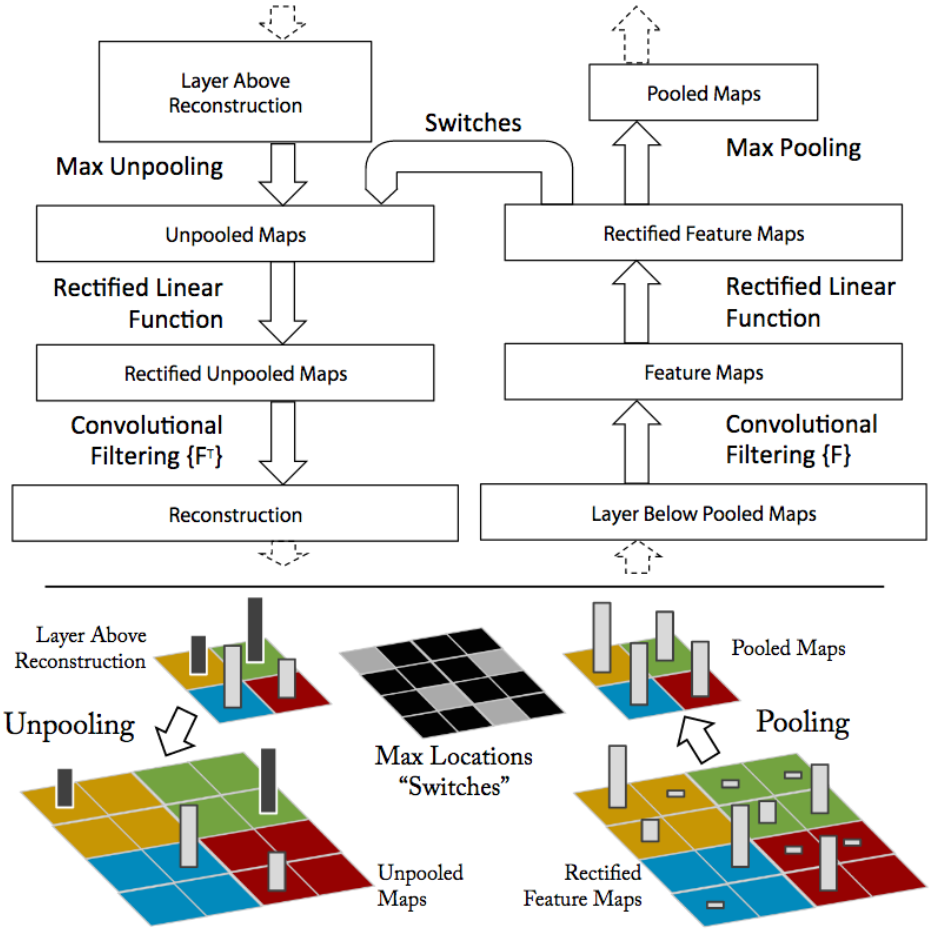
THE BASIC IDEA IS TO REVERSE THE NETWORK TO

OCCLUSION SENSITIVITY

INCEPTIONISM

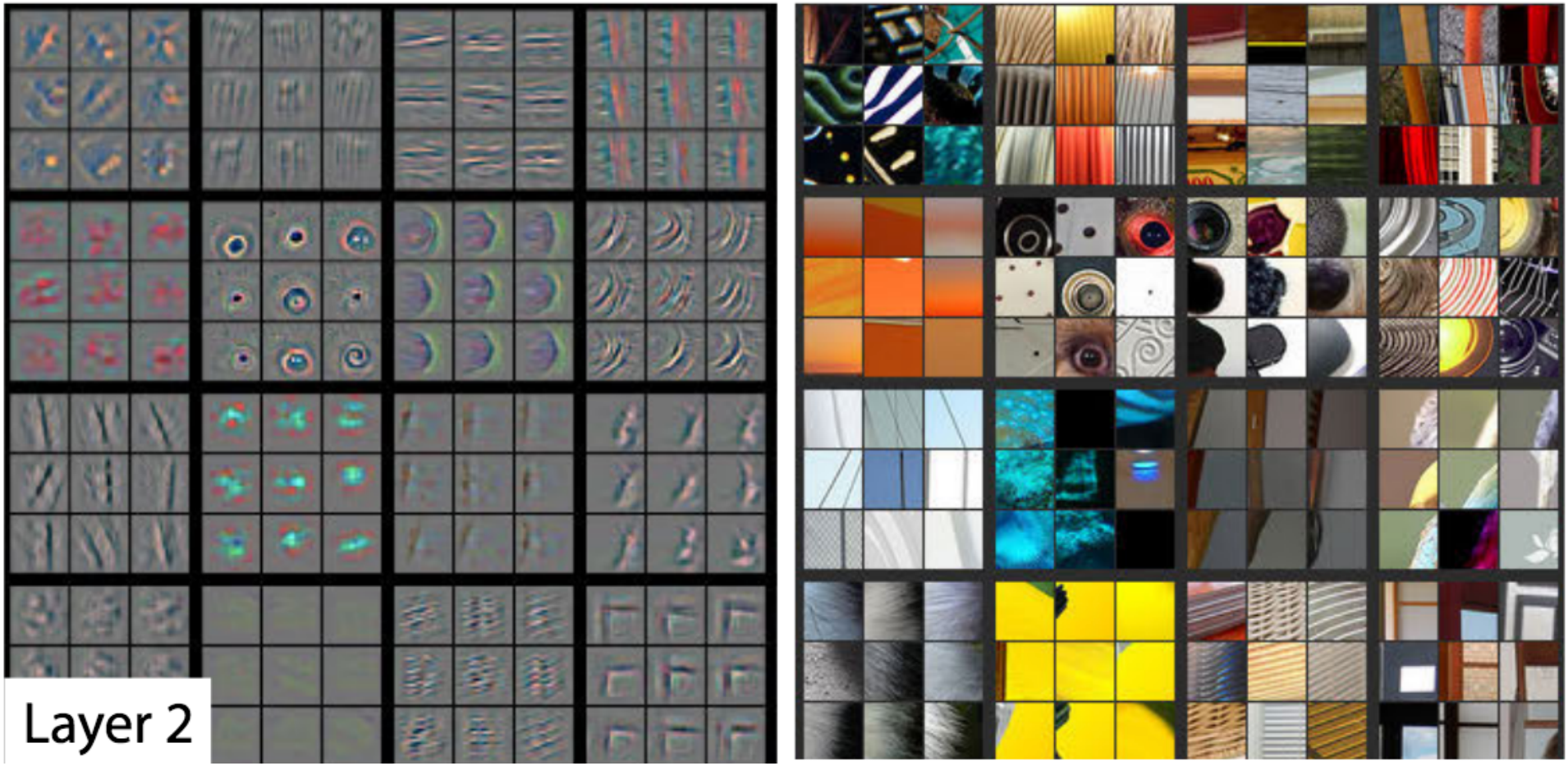
INTEGRATED GRADIENTS

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



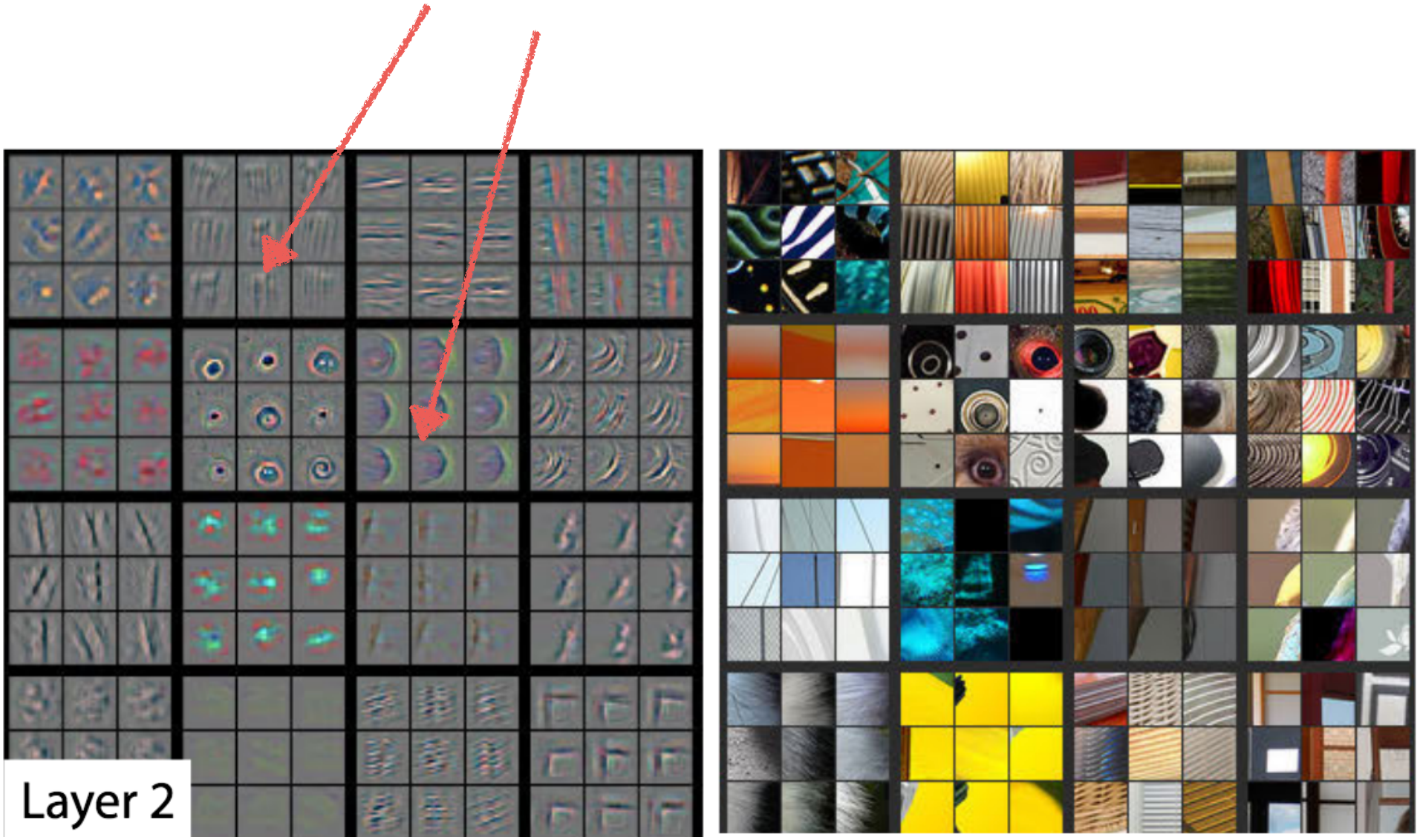
IT ALLOWS TO SEE
WHICH
REGIONS OF THE INPUT
GENERATED
A MAXIMUM RESPONSE
IN A NEURON

Zeiler+14



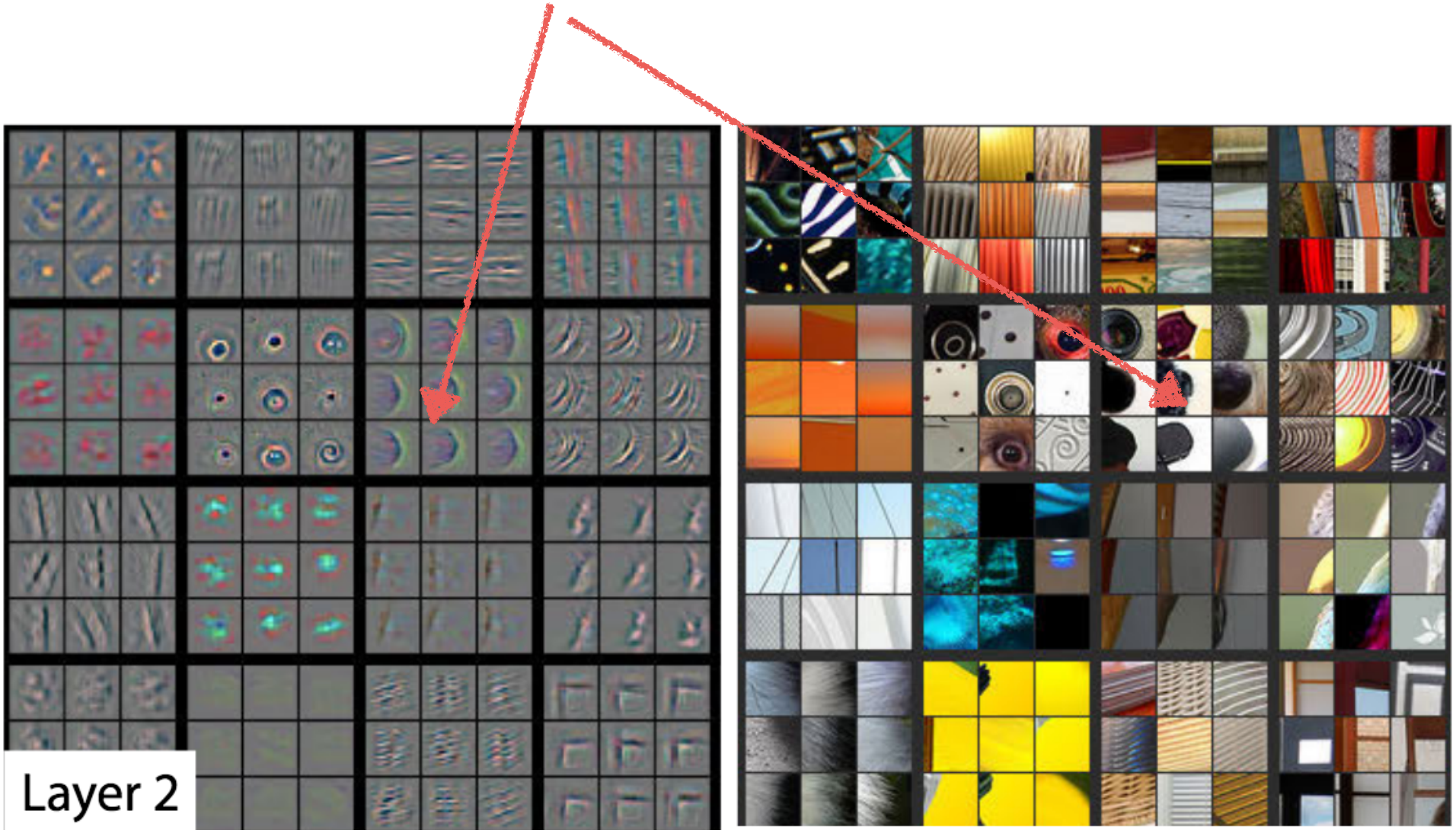
Zeiler+14

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



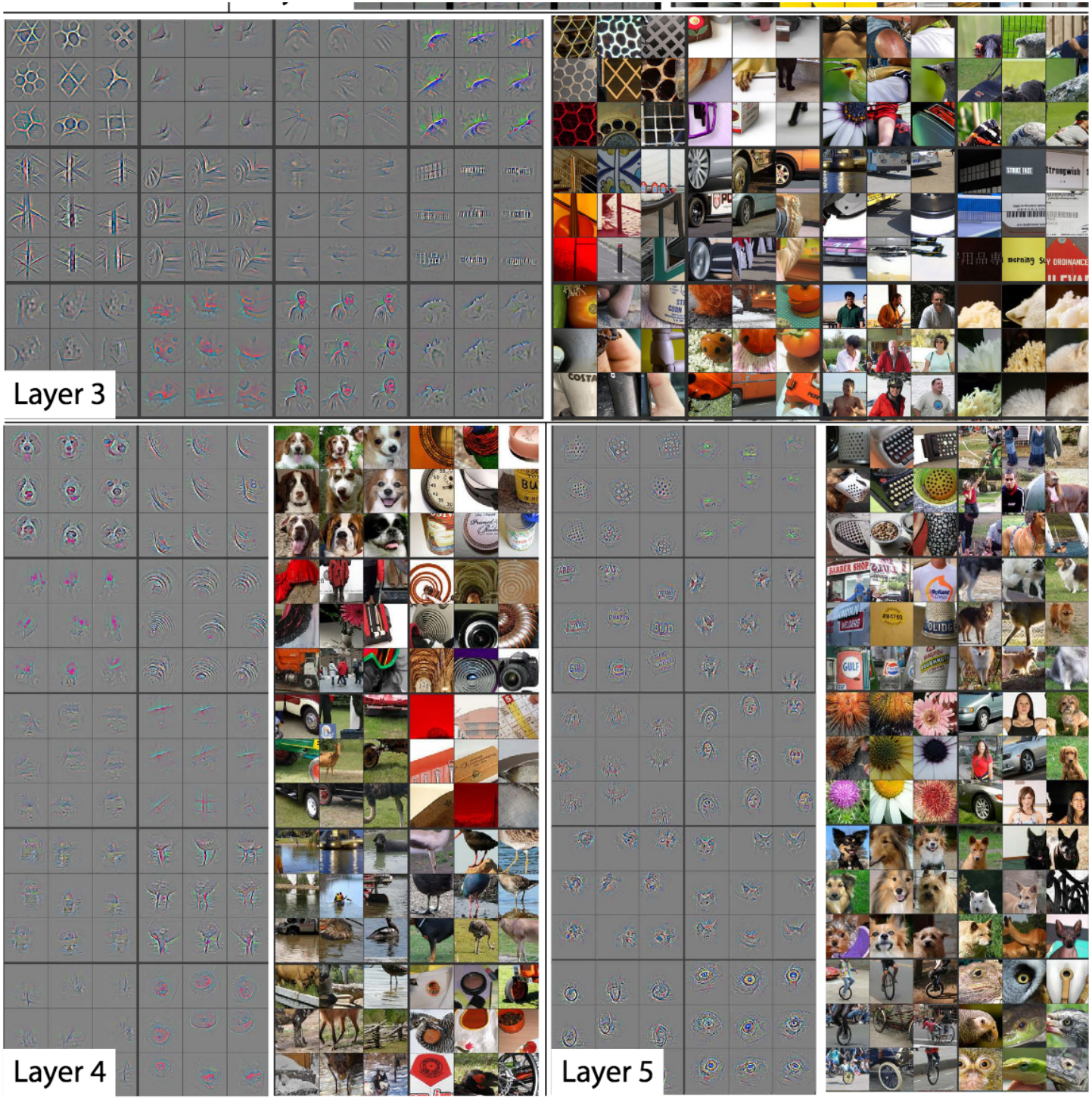
Zeiler+14

THE CORRESPONDING REGIONS OF IMAGES THAT GENERATED THE MAXIMUM RESPONSE



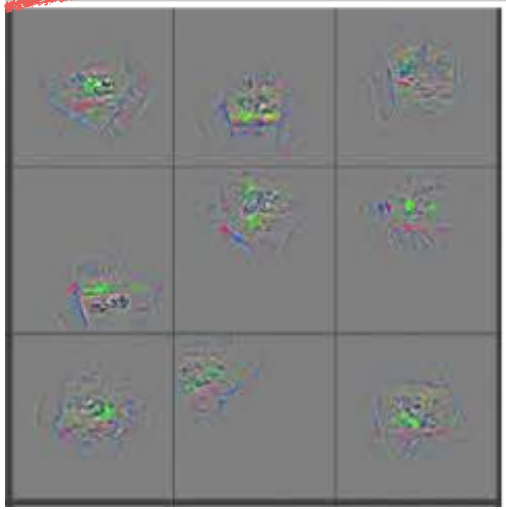
Zeiler+14

CAN BE REPEATED FOR DEEPER LAYERS ALTHOUGH IT BECOMES LESS INTUITIVE

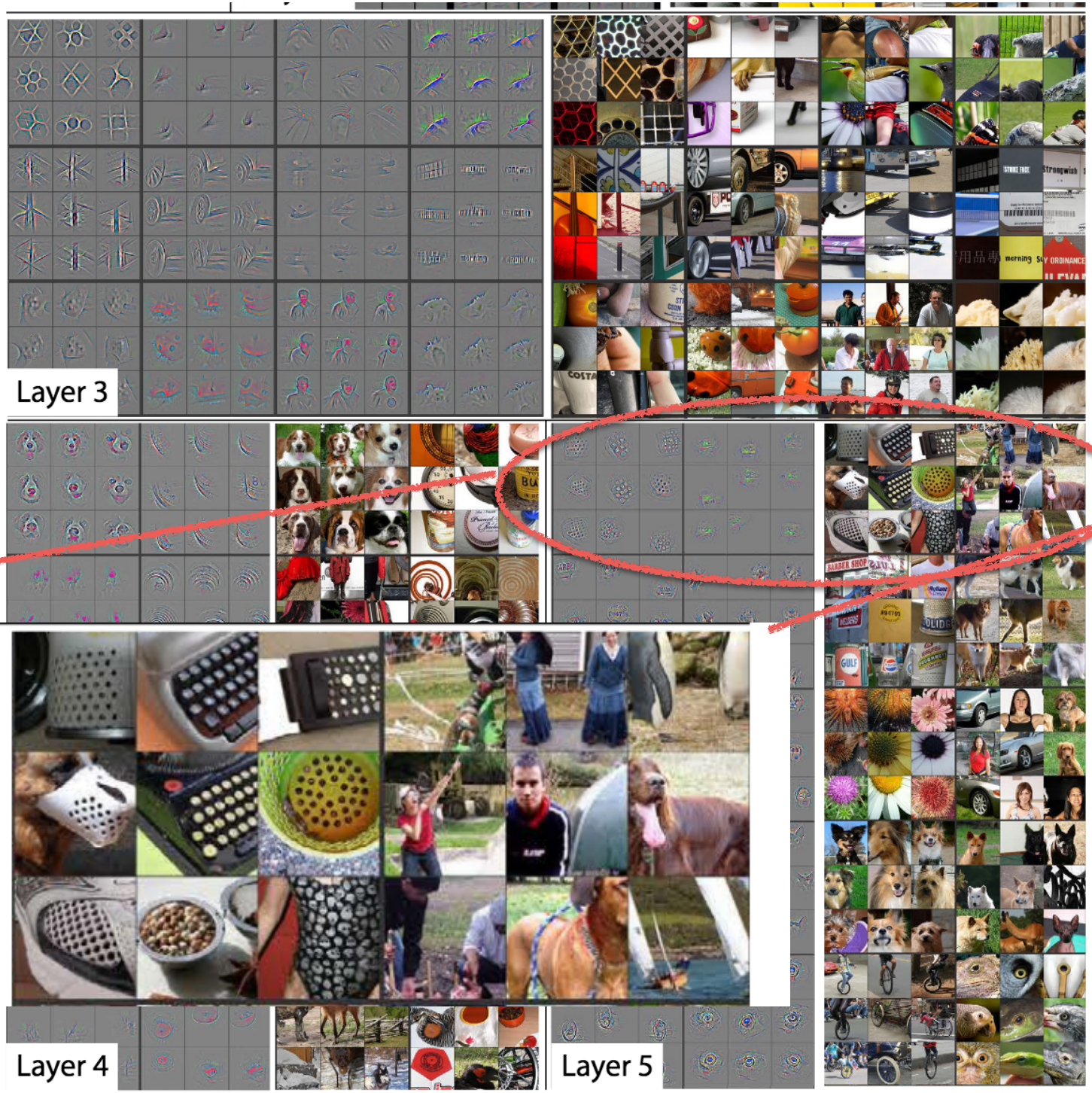


Zeiler+14

CAN BE REPEATED FOR DEEPER LAYERS ALTHOUGH IT BECOMES LESS



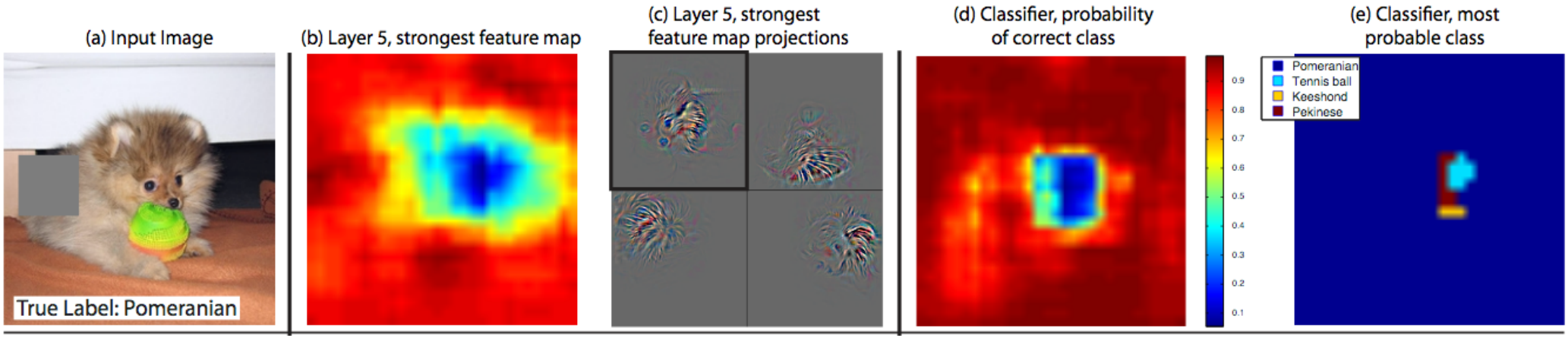
Zeiler+14



OCCLUSION SENSITIVITY 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>



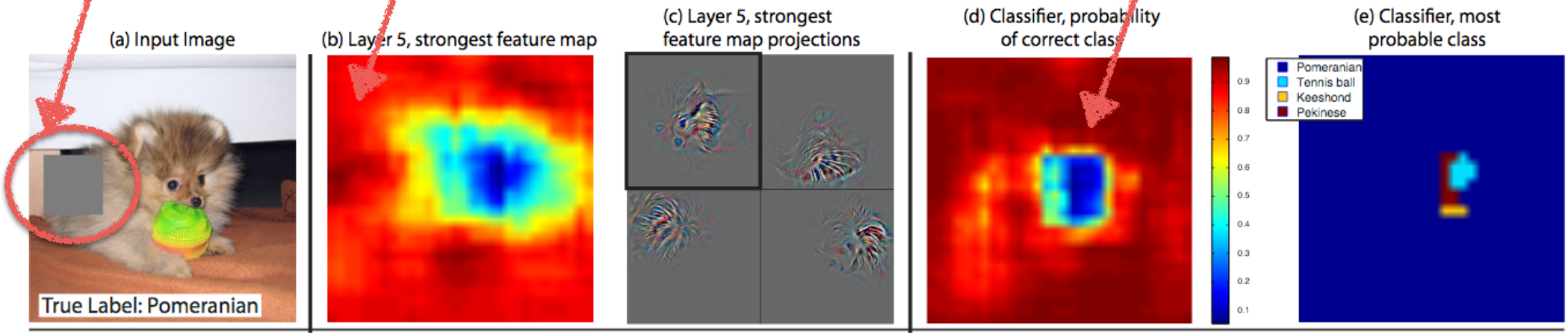
Zeiler+14

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 - DEEP DREAM

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

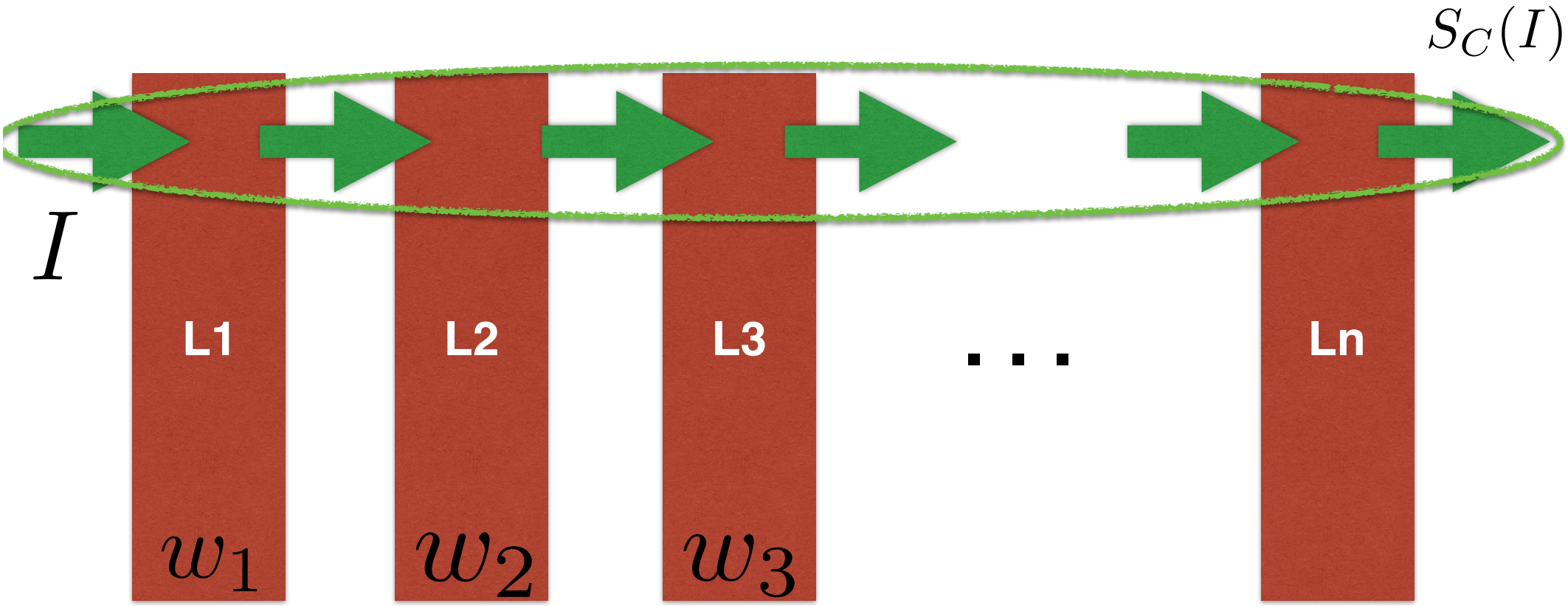
image I

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

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

Simonyan+14

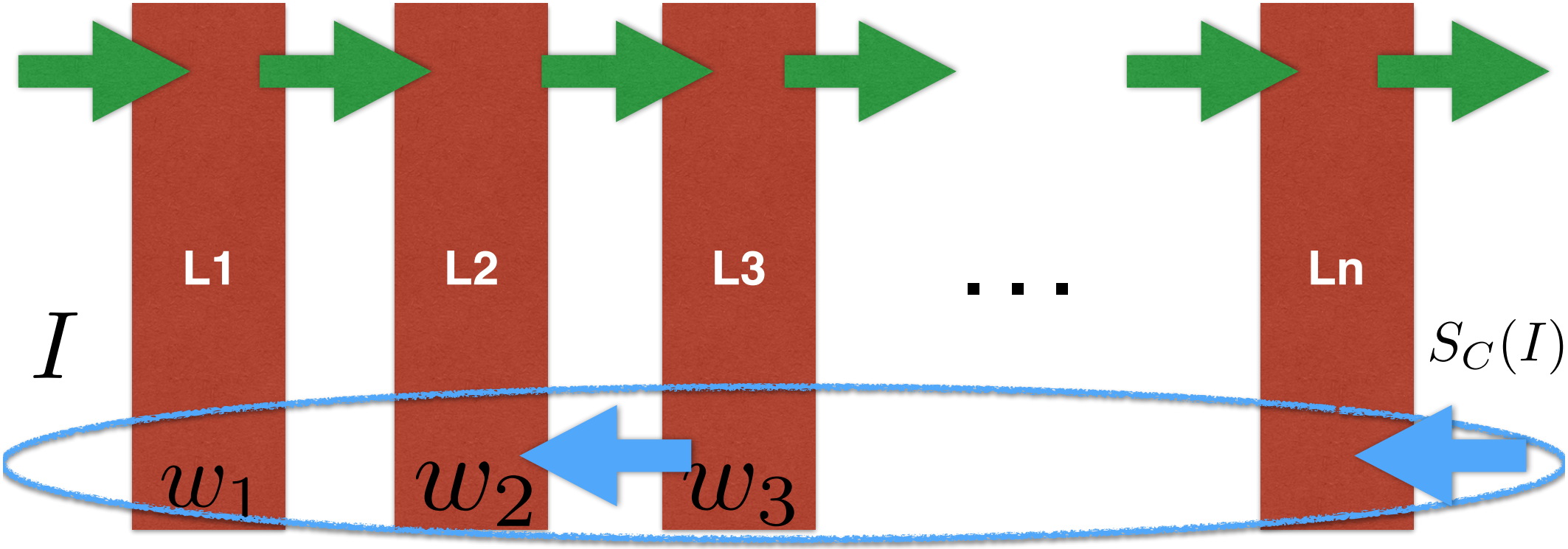
INCEPTIONISM - DEEP DREAM



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

Simonyan+14

INCEPTIONISM - DEEP DREAM

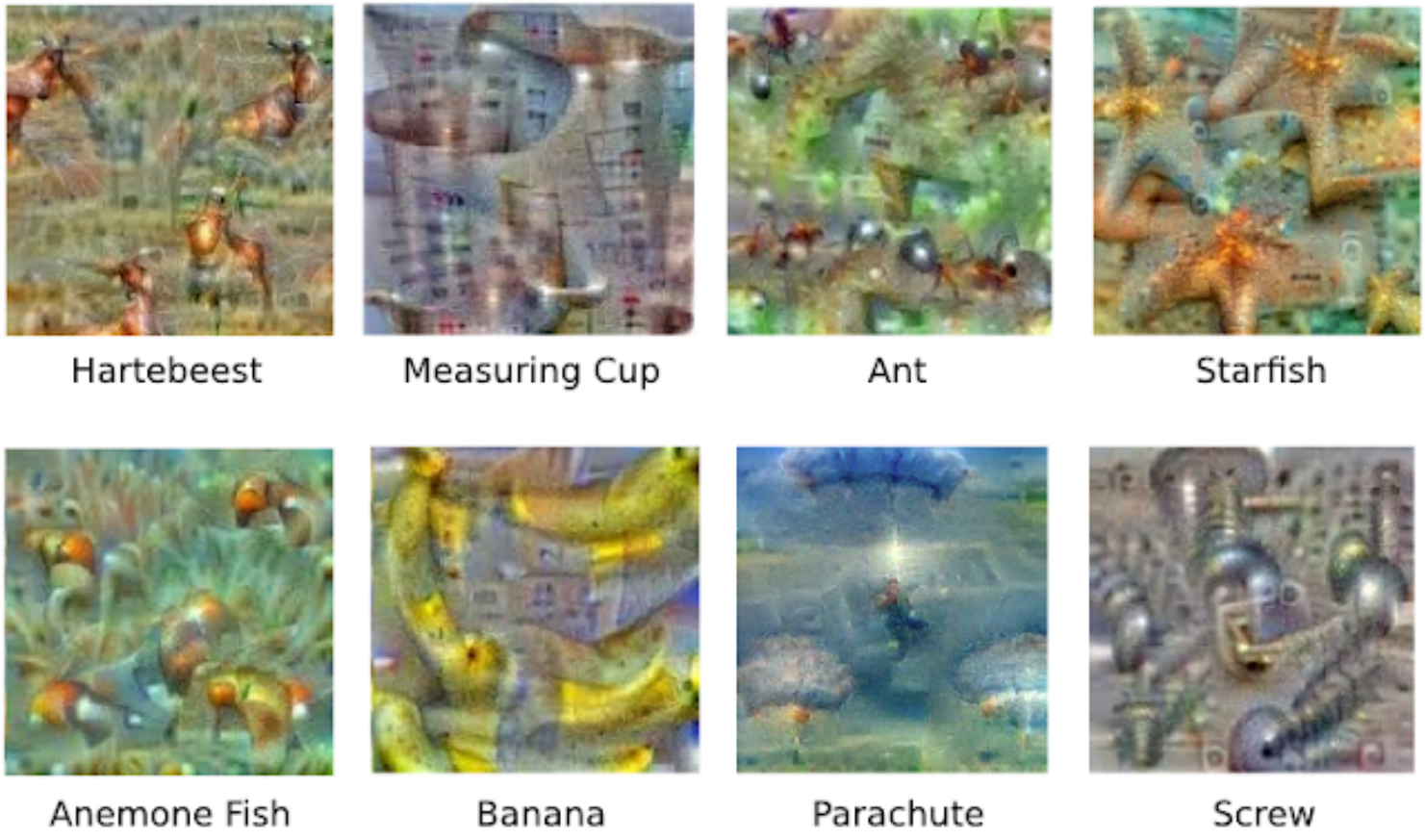


DURING THE RECONSTRUCTION PHASE, I IS LEARNT TRHOUG BACKPROPAGATION KEEPING THE WEIGHTS FIXED

Simonyan+14

INCEPTIONISM - DEEP DREAM

RESULTS REVEAL INTERESTING INFORMATION ON HOW THE NETWORKS BUILD REPRESENTATIONS OF OBJECTS



INCEPTIONISM - DEEP DREAM

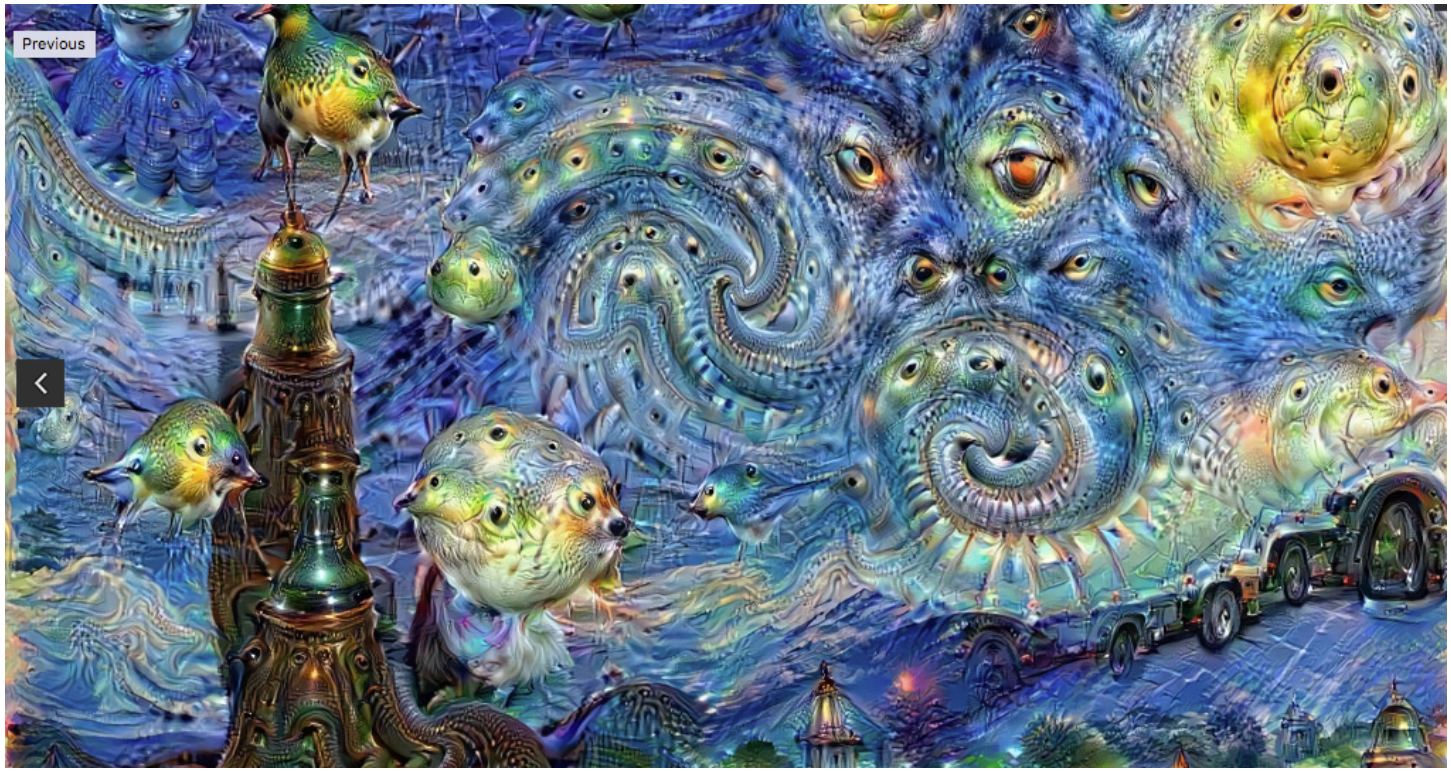
RESULTS REVEAL INTERESTING INFORMATION ON HOW THE NETWORKS BUILD REPRESENTATIONS OF OBJECTS

SOME STRANGE CASES…

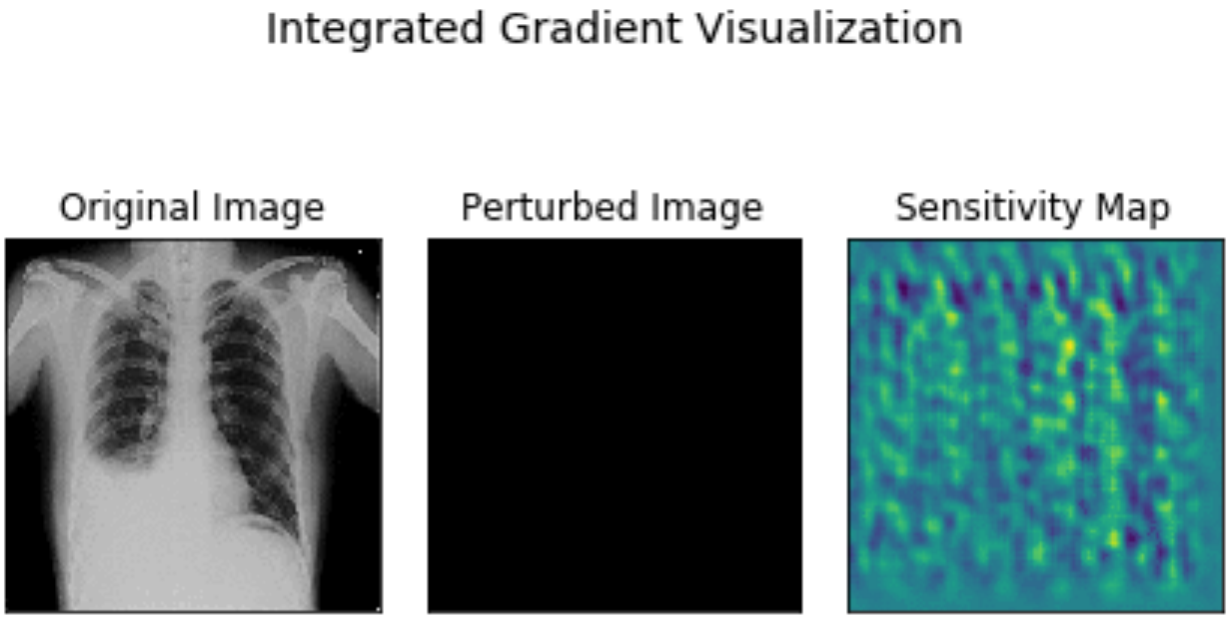
DEEP DREAM

https://deepdreamgenerator.com/

IT HAS NOW BECOME A SORT OF ART?

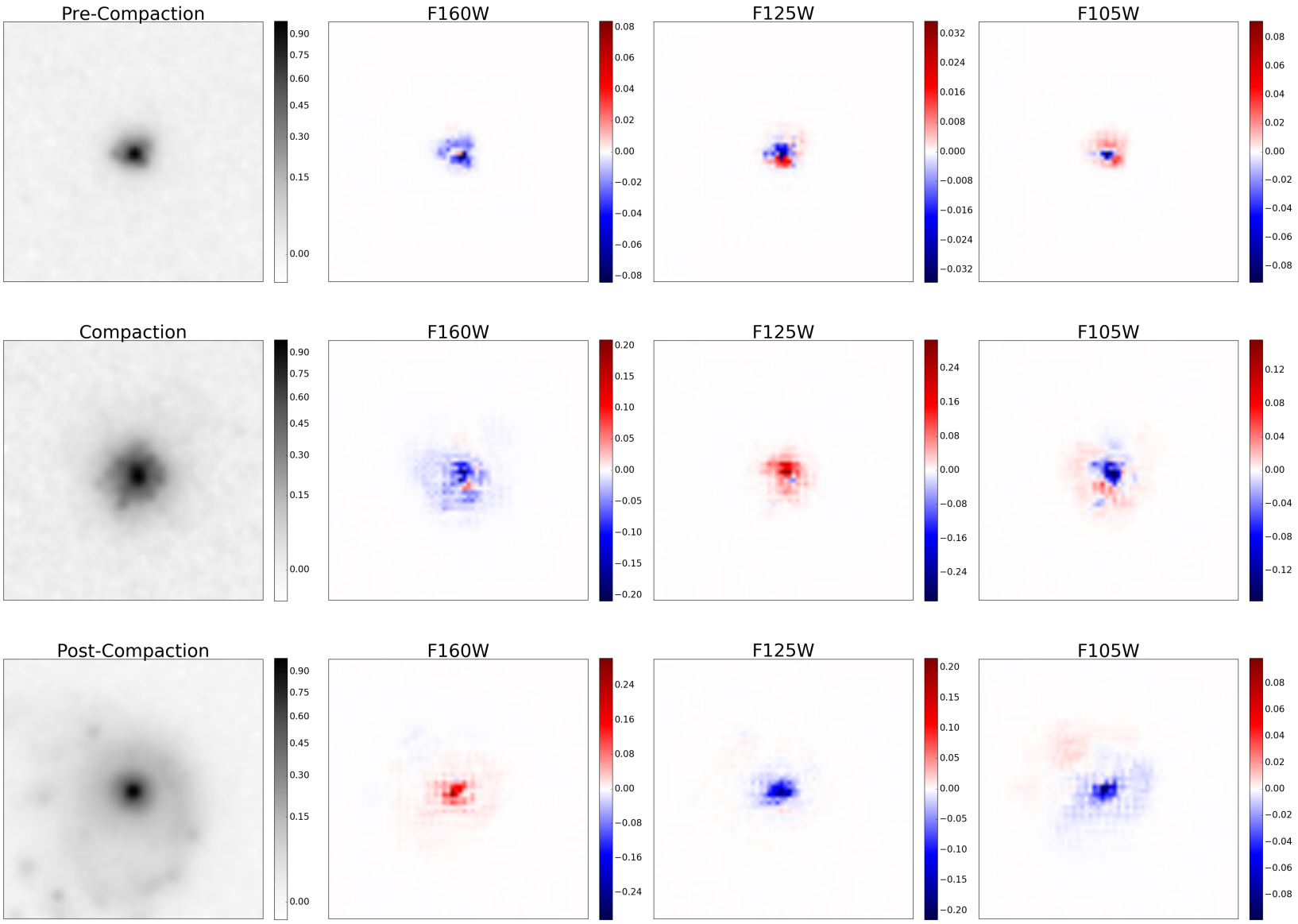


INTEGRATED GRADIENTS



Sundararajan+17

INTEGRATED GRADIENTS



INTEGRATED GRADIENTS

KERAS IMPLEMENTATION:
https://github.com/hiranumn/IntegratedGradients