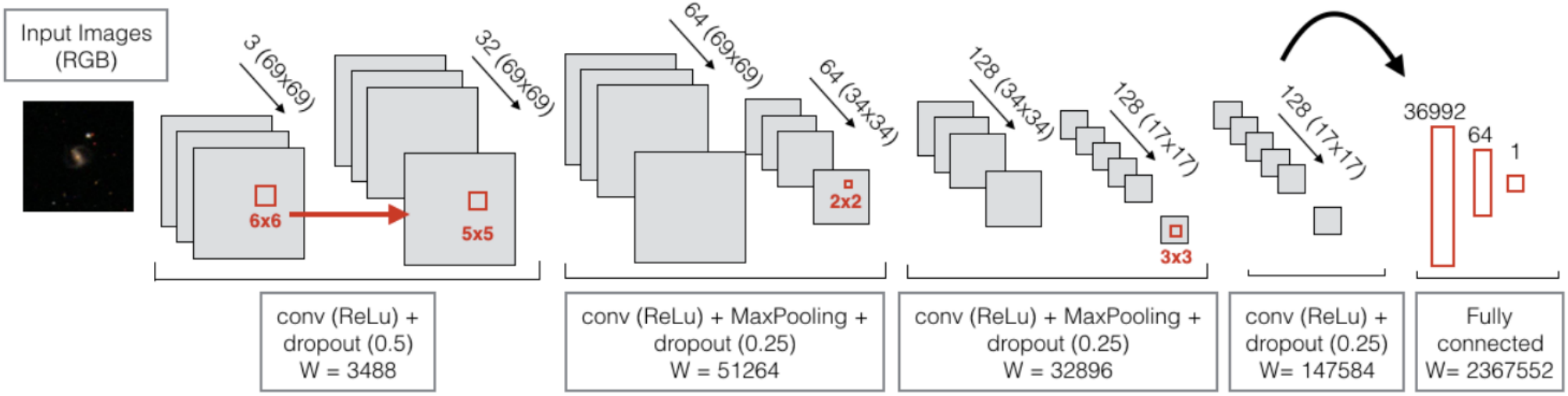
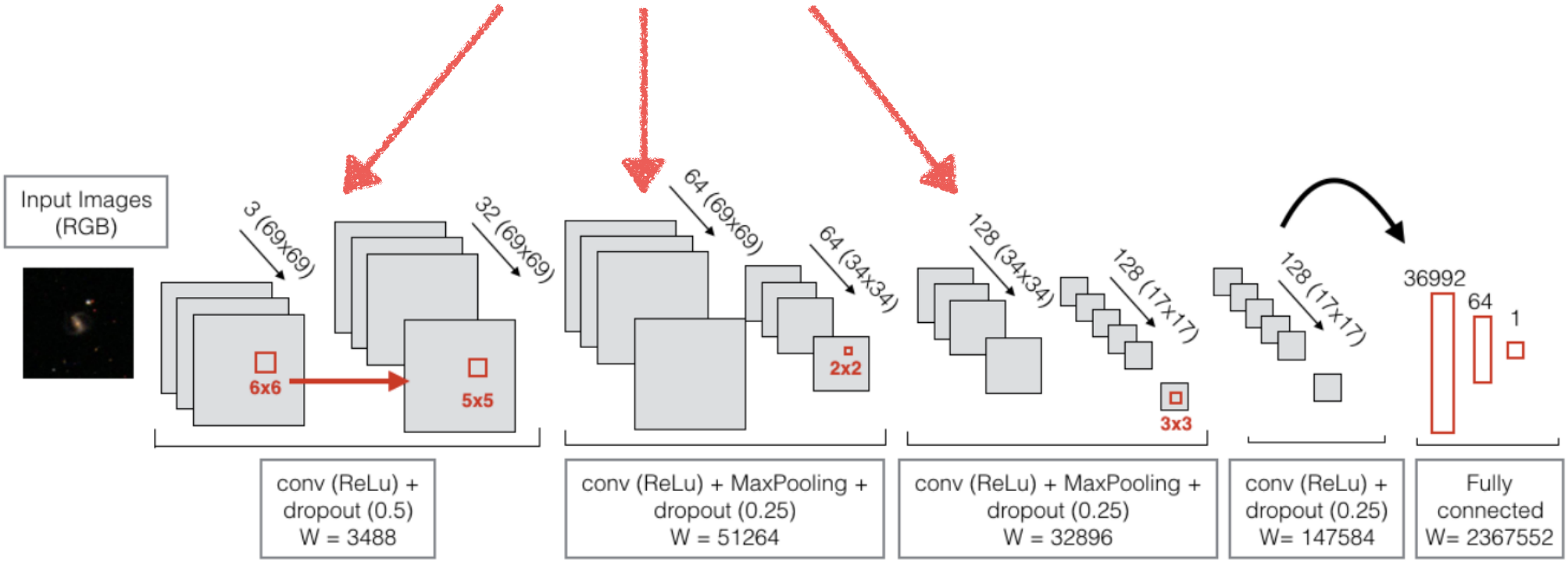
EXAMPLE OF VERY SIMPLE CNN



Dominguez-Sanchez+18

EXAMPLE OF VERY SIMPLE CNN

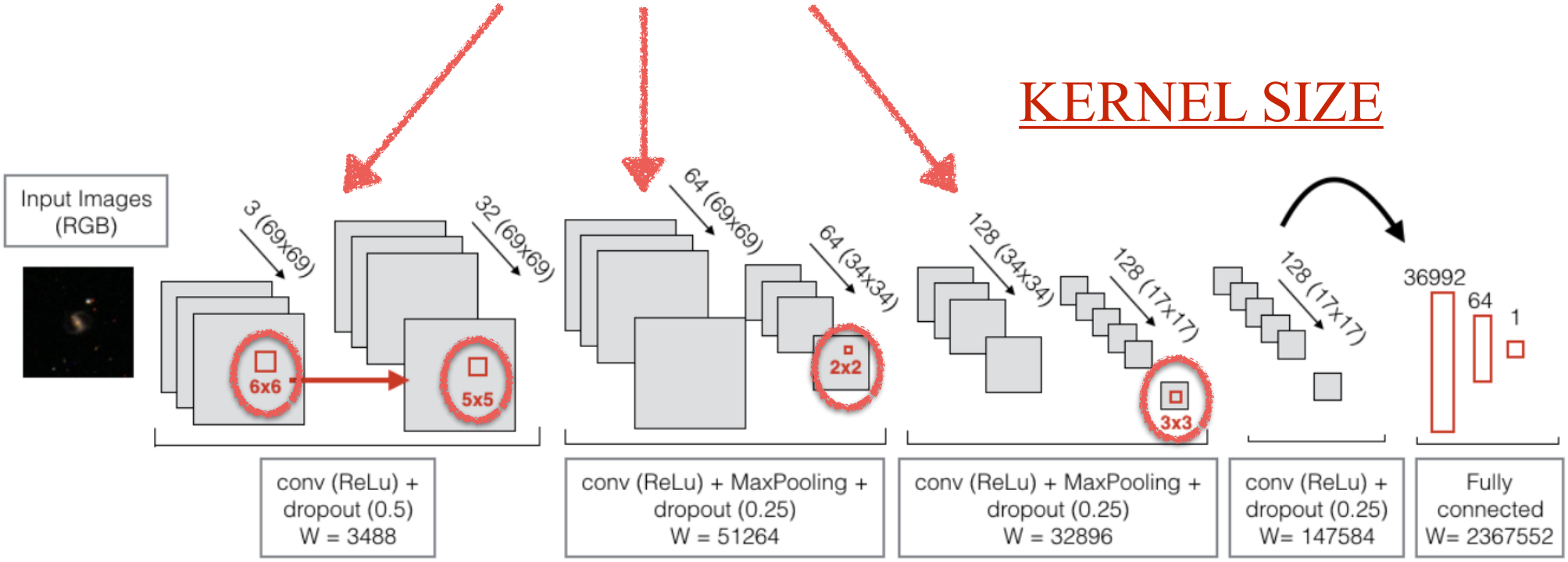
3 convolutional layers



Dominguez-Sanchez+18

EXAMPLE OF VERY SIMPLE CNN

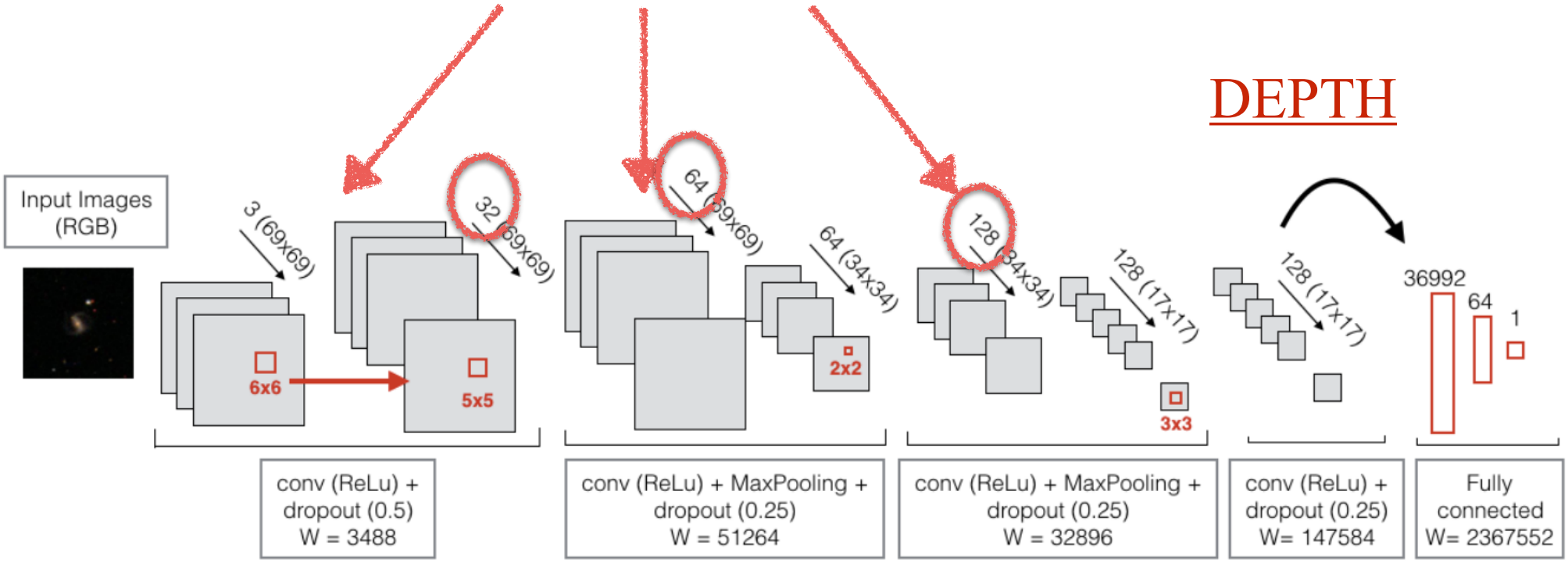
3 convolutional layers



Dominguez-Sanchez+18

EXAMPLE OF VERY SIMPLE CNN

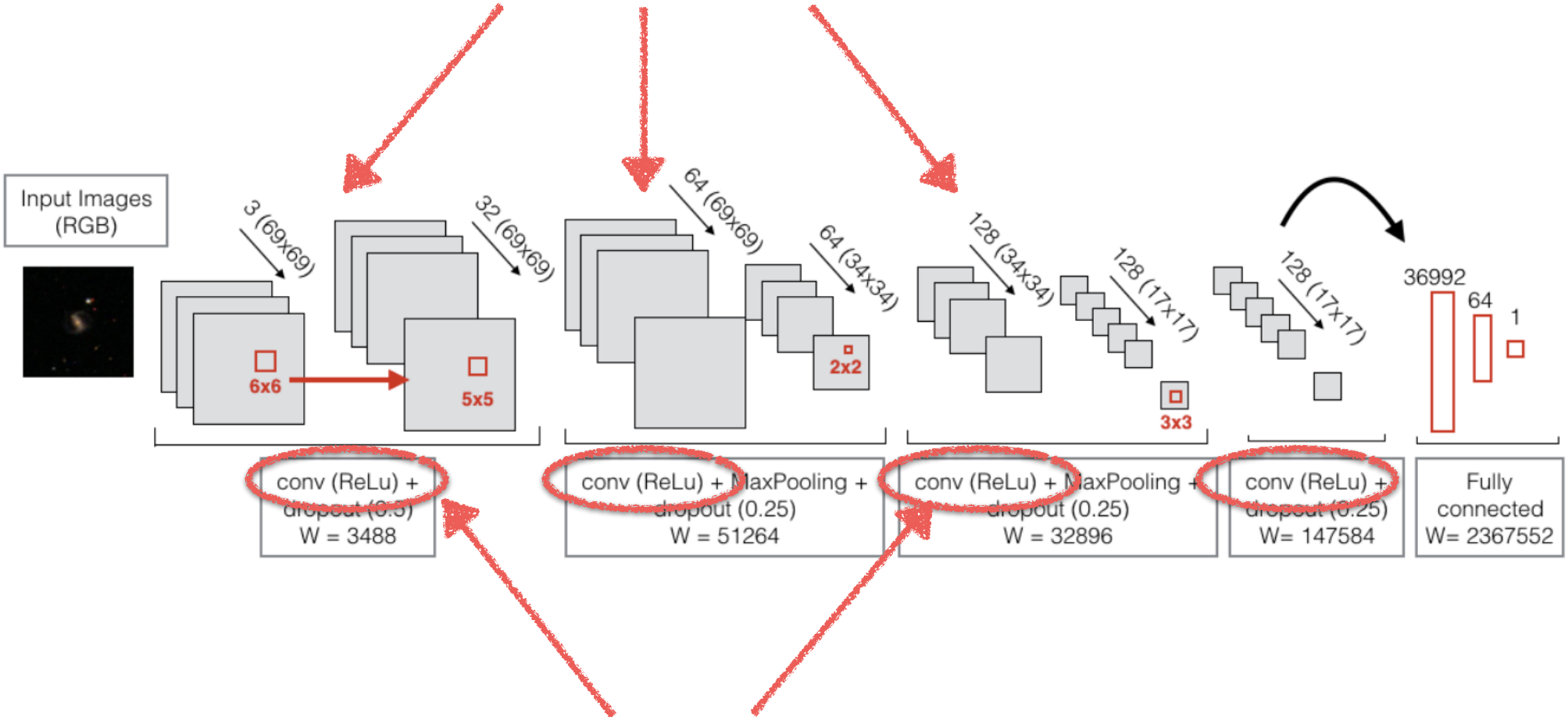
3 convolutional layers



Dominguez-Sanchez+18

EXAMPLE OF VERY SIMPLE CNN

3 convolutional layers

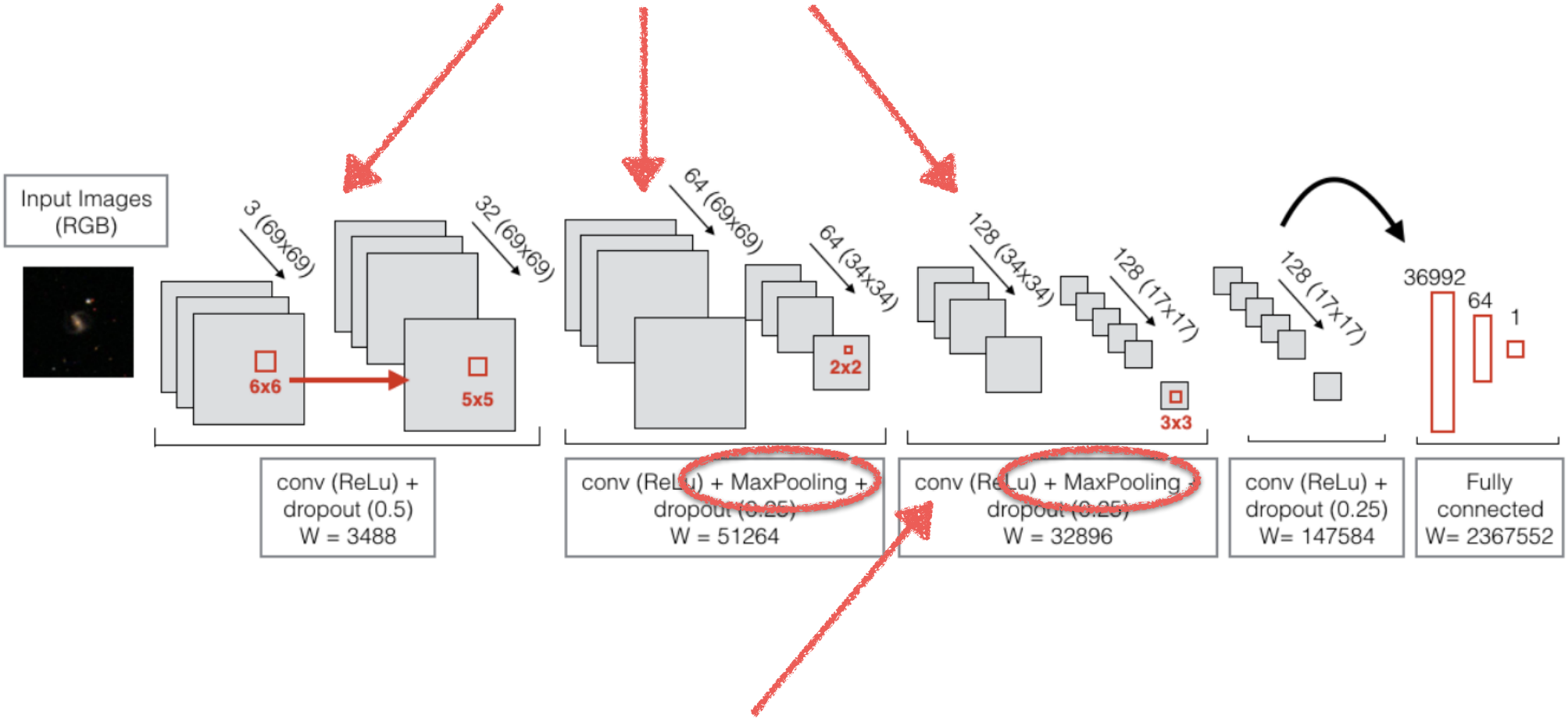


ReLu activation

Dominguez-Sanchez+18

EXAMPLE OF VERY SIMPLE CNN

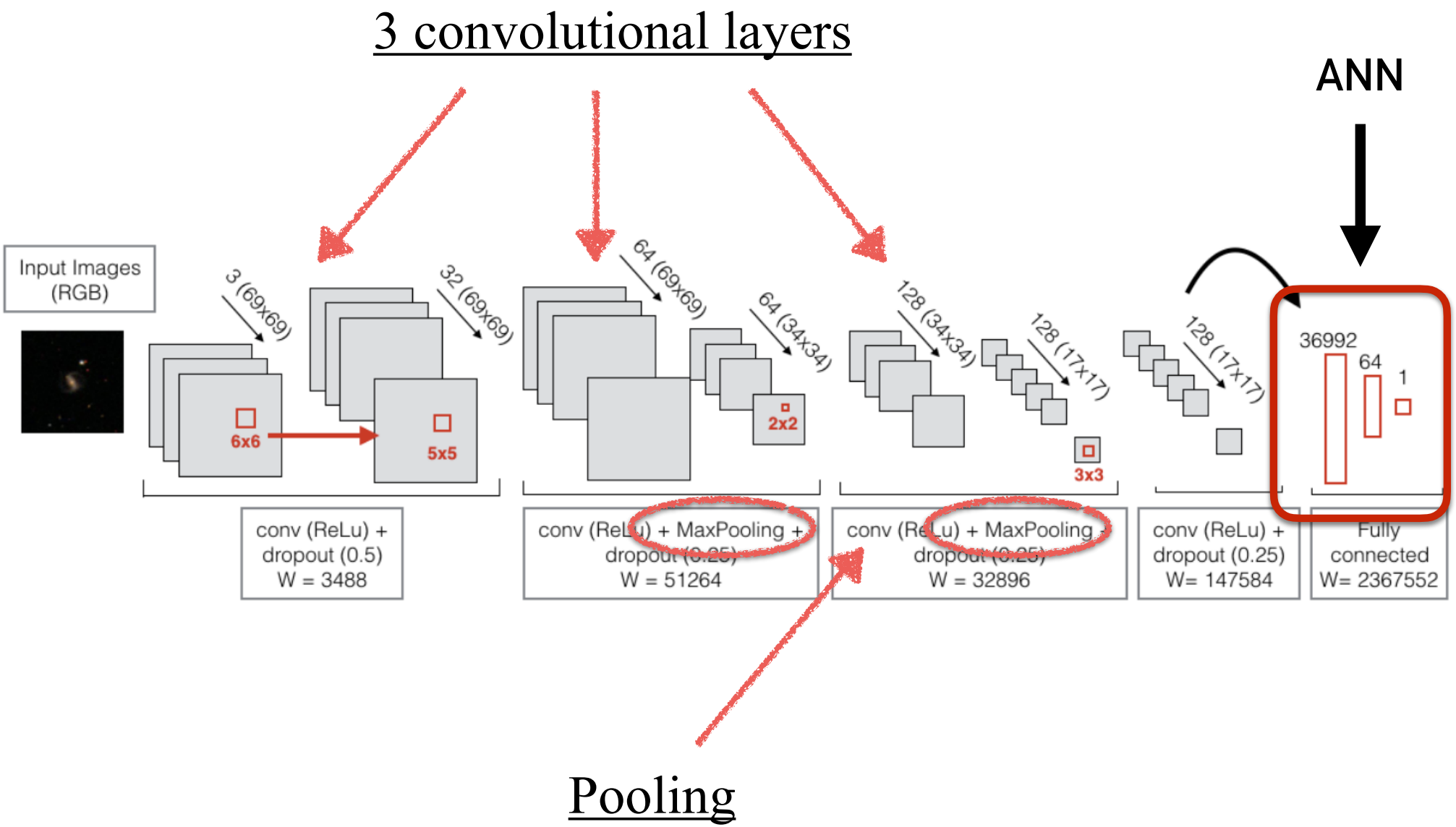
3 convolutional layers



Pooling

Dominguez-Sanchez+18

EXAMPLE OF VERY SIMPLE CNN

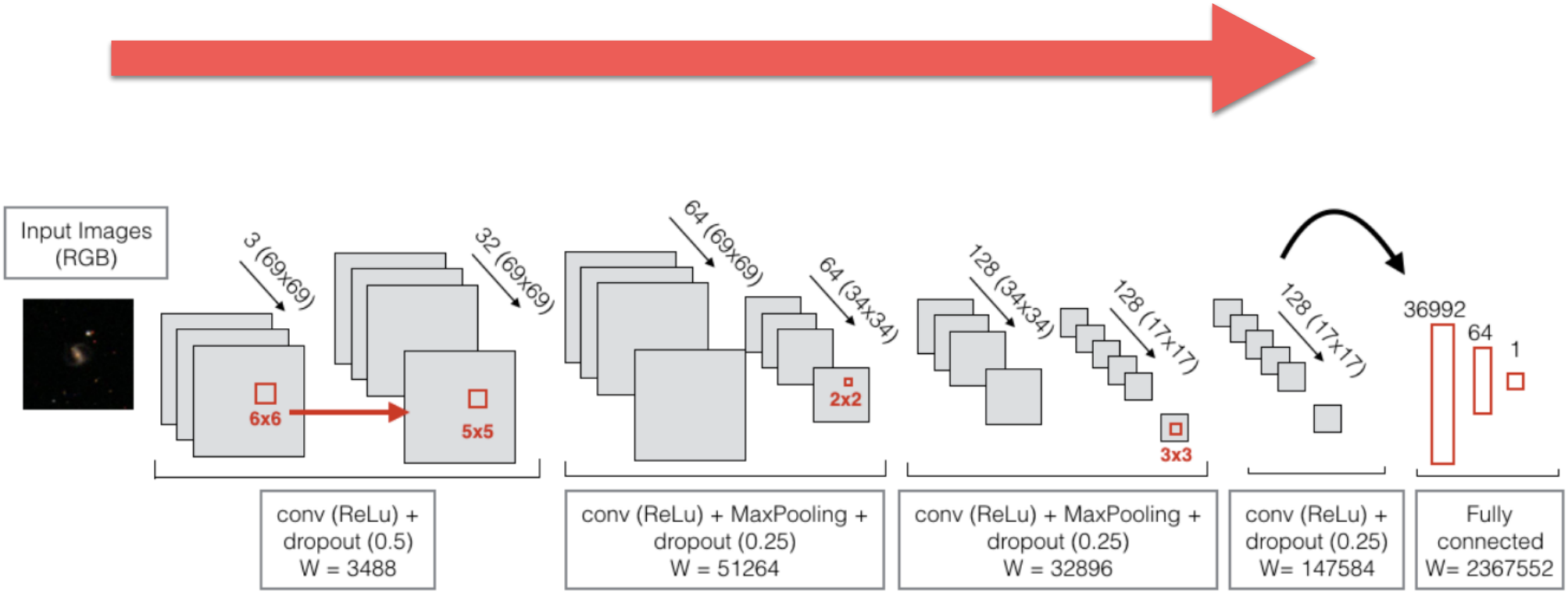


Dominguez-Sanchez+18

EXAMPLE OF VERY SIMPLE CNN

OVERALL:

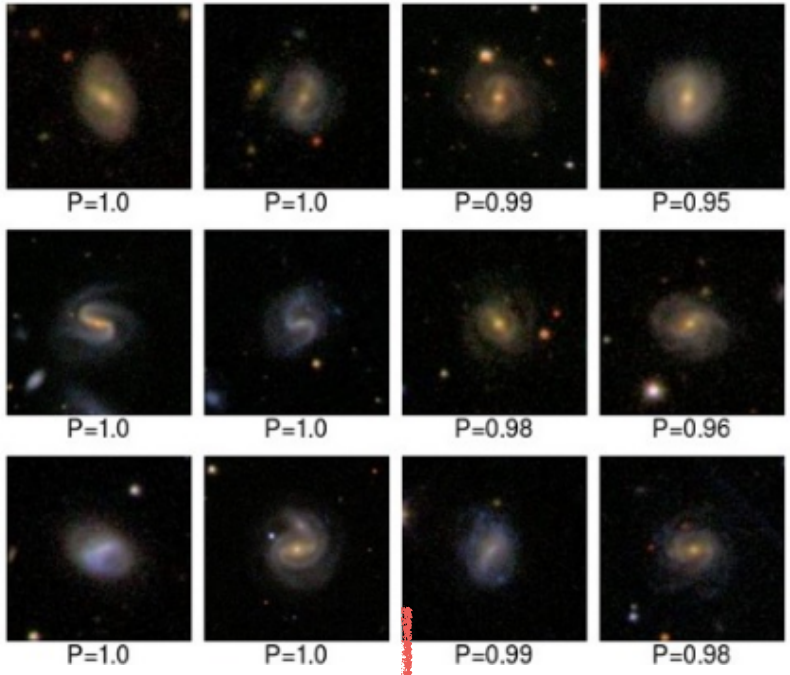
- decrease of tensor size
- increase of depth

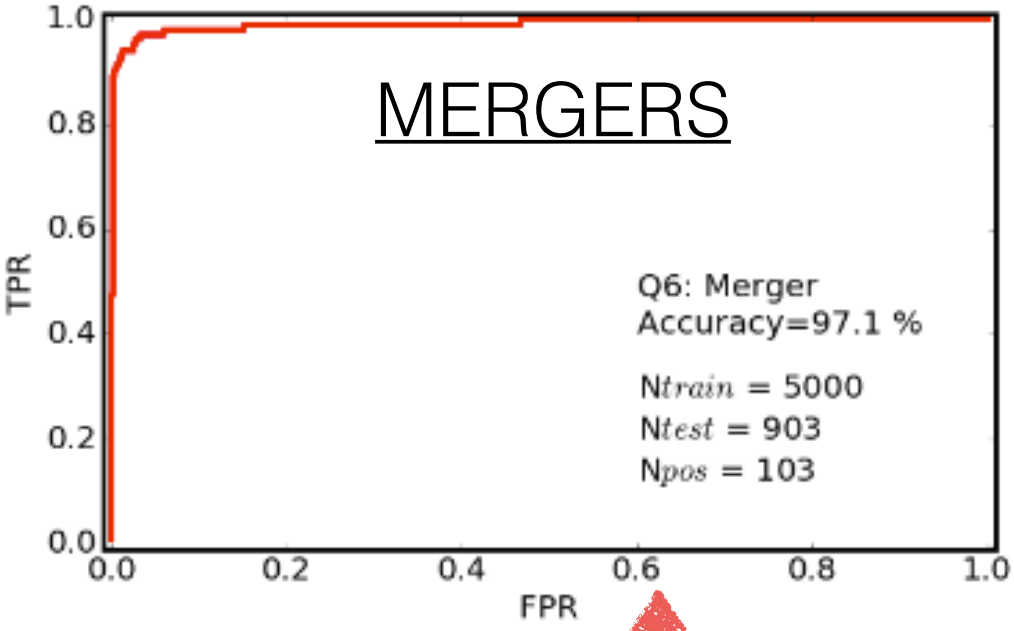


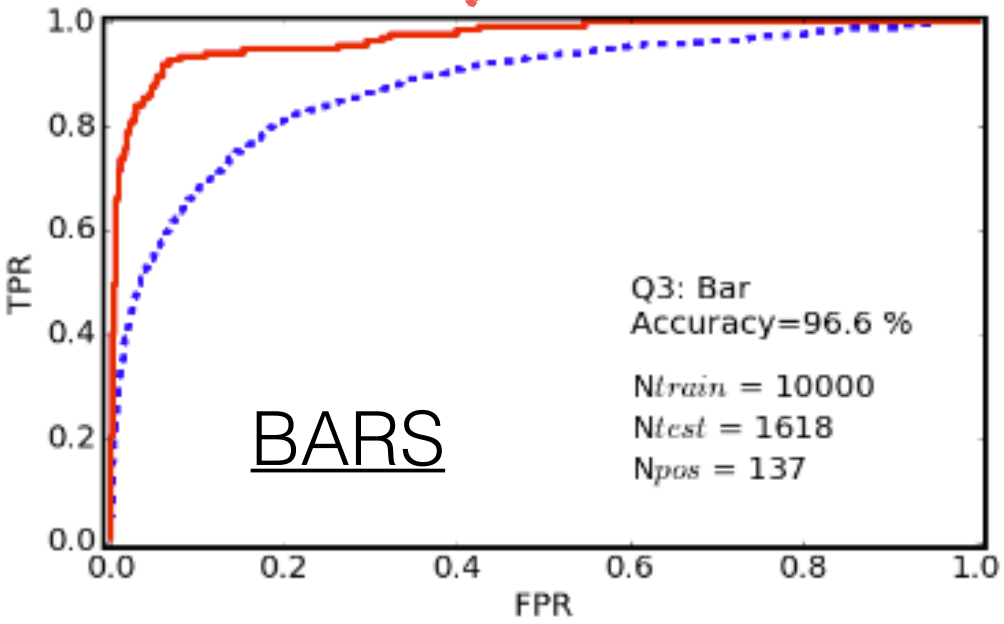
Dominguez-Sanchez+18

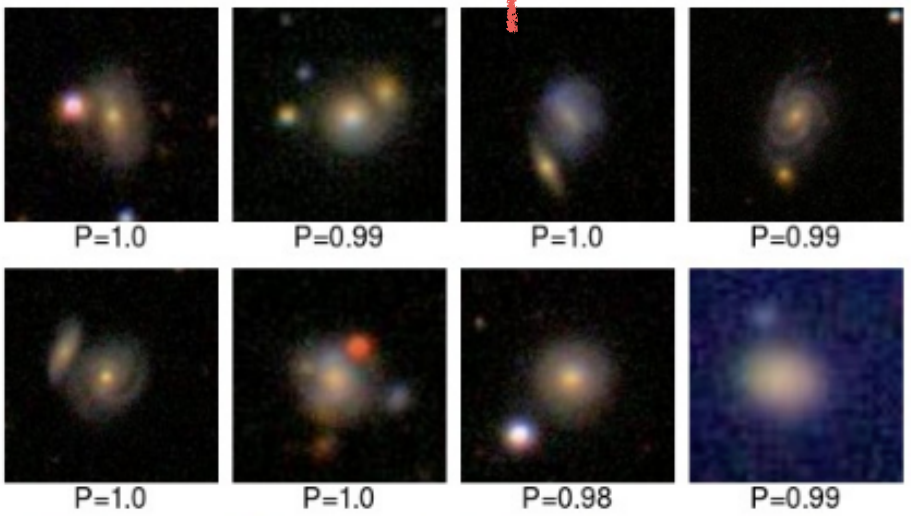
IMPLEMENTATION IN KERAS

```
#========= Model definition=======

#Convolutional Layers

model = Sequential()
model.add(Convolution2D(32, 6,6, border_mode='same',
                        input_shape=(img_channels, img_rows, img_cols)))
model.add(Activation('relu'))
model.add(Dropout(0.5))

model.add(Convolution2D(64, 5, 5, border_mode='same'))
model.add(Activation('relu'))

model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Convolution2D(128, 2, 2, border_mode='same'))
model.add(Activation('relu'))

model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Convolution2D(128, 3, 3, border_mode='same'))
model.add(Activation('relu'))

model.add(Dropout(0.25))

#Fully Connected start here
#-------------------------------#

model.add(Flatten())
model.add(Dense(64, activation='relu'))
model.add(Dropout(.5))
model.add(Dense(1, init='uniform', activation='sigmoid'))

print("Compilation...")

model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
```


P=1.0
P=1.0
P=0.99
P=0.95
P=1.0
P=1.0
P=0.98
P=0.96
P=1.0
P=1.0
P=0.99
P=0.98


MERGERS
Q6: Merger
Accuracy=97.1 %
Ntrain = 5000
Ntest = 903
Npos = 103
TPR
FPR
0.0
0.2
0.4
0.6
0.8
1.0

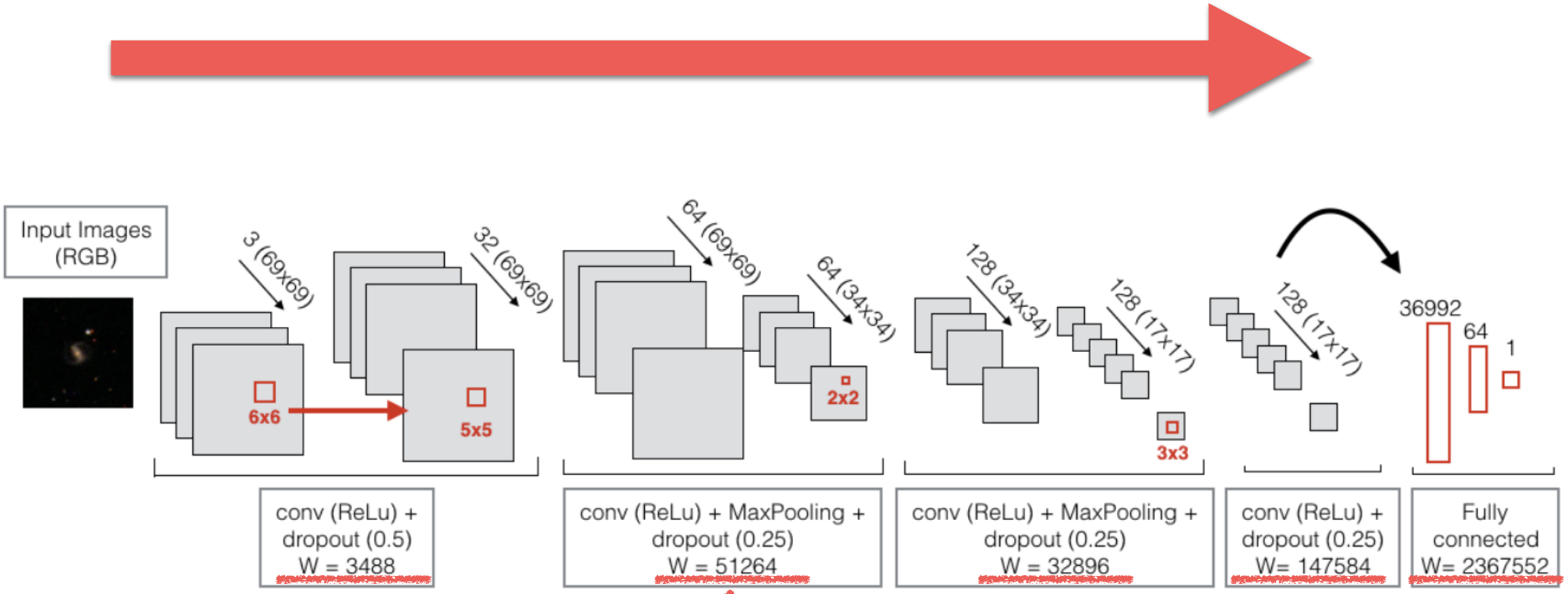

BARS
Q3: Bar
Accuracy=96.6 %
Ntrain = 10000
Ntest = 1618
Npos = 137
TPR
FPR
0.0
0.2
0.4
0.6
0.8
1.0


P=1.0
P=0.99
P=1.0
P=0.99
P=1.0
P=1.0
P=0.98
P=0.99

EXAMPLE OF VERY SIMPLE CNN

OVERALL:

- decrease of tensor size
- increase of depth



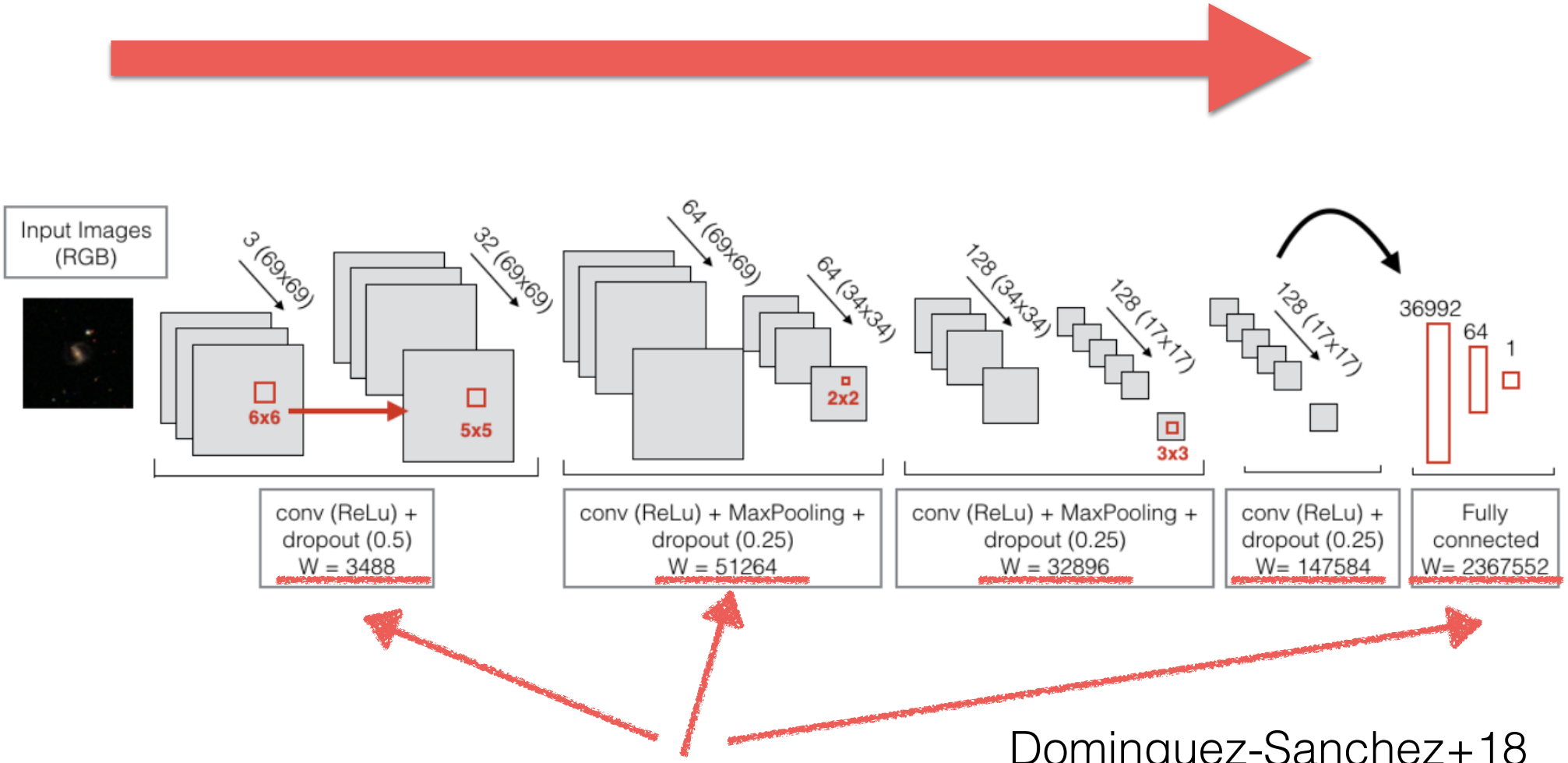
Dominguez-Sanchez+18

Number of parameters

EXAMPLE OF VERY SIMPLE CNN

OVERALL:

- decrease of tensor size
- increase of depth



Dominguez-Sanchez+18

2 million of parameters for this very simple network!

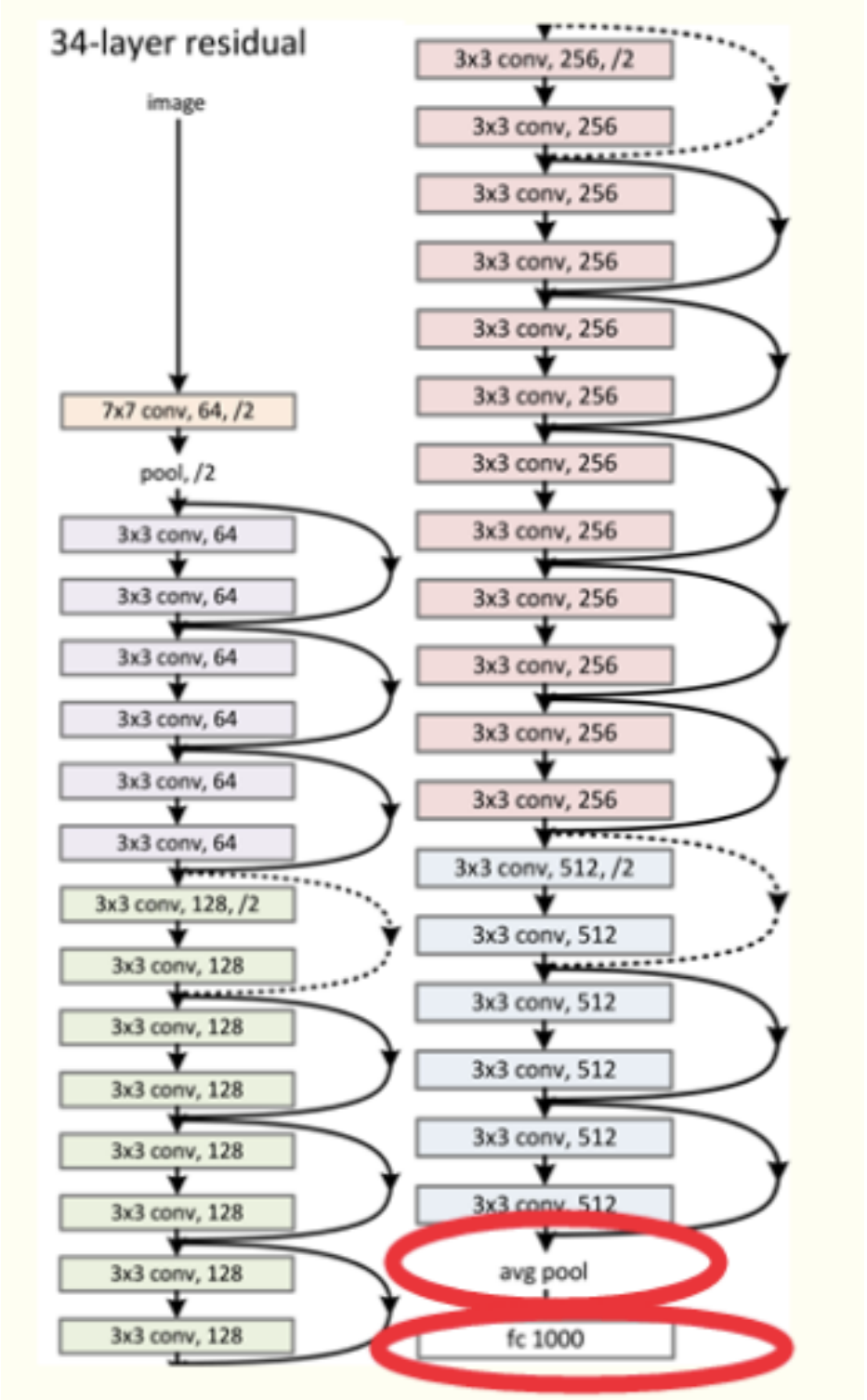
CHECKING THE NUMBER OF PARAMETERS / LAYERS WITH KERAS

model.summary()

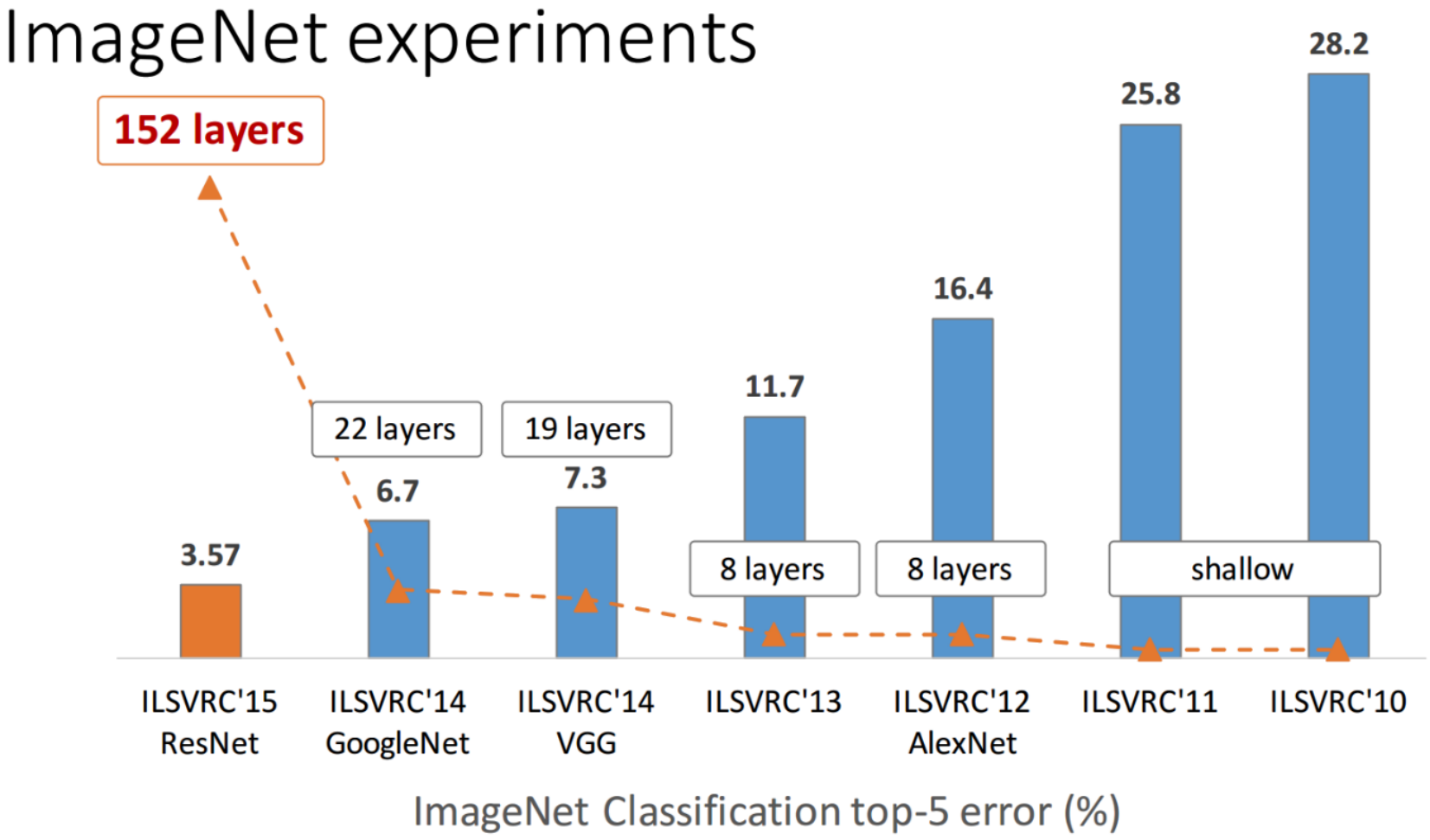
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 1, 16, 112, 112)   0
_________________________________________________________________
conv3d_1 (Conv3D)            (None, 16, 16, 112, 112)  448
_________________________________________________________________
batch_normalization_1 (Batch (None, 16, 16, 112, 112)  448
_________________________________________________________________
activation_1 (Activation)    (None, 16, 16, 112, 112)  0
_________________________________________________________________
max_pooling3d_1 (MaxPooling3 (None, 16, 8, 56, 56)     0
_________________________________________________________________
conv3d_2 (Conv3D)            (None, 32, 8, 56, 56)     13856
_________________________________________________________________
batch_normalization_2 (Batch (None, 32, 8, 56, 56)     224
_________________________________________________________________
activation_2 (Activation)    (None, 32, 8, 56, 56)     0
_________________________________________________________________
max_pooling3d_2 (MaxPooling3 (None, 32, 4, 28, 28)     0
_________________________________________________________________
conv3d_3 (Conv3D)            (None, 64, 4, 28, 28)     55360
_________________________________________________________________
batch_normalization_3 (Batch (None, 64, 4, 28, 28)     112
_________________________________________________________________
activation_3 (Activation)    (None, 64, 4, 28, 28)     0
_________________________________________________________________
max_pooling3d_3 (MaxPooling3 (None, 64, 2, 14, 14)     0
_________________________________________________________________
activation_12 (Activation)   (None, 64, 2, 14, 14)     0
=================================================================
Total params: 70,448
Trainable params: 70,056
Non-trainable params: 392
_________________________________________________________________
```

MODERN CNNs

RESNET



DEEPER TENDS TO BE BETTER…

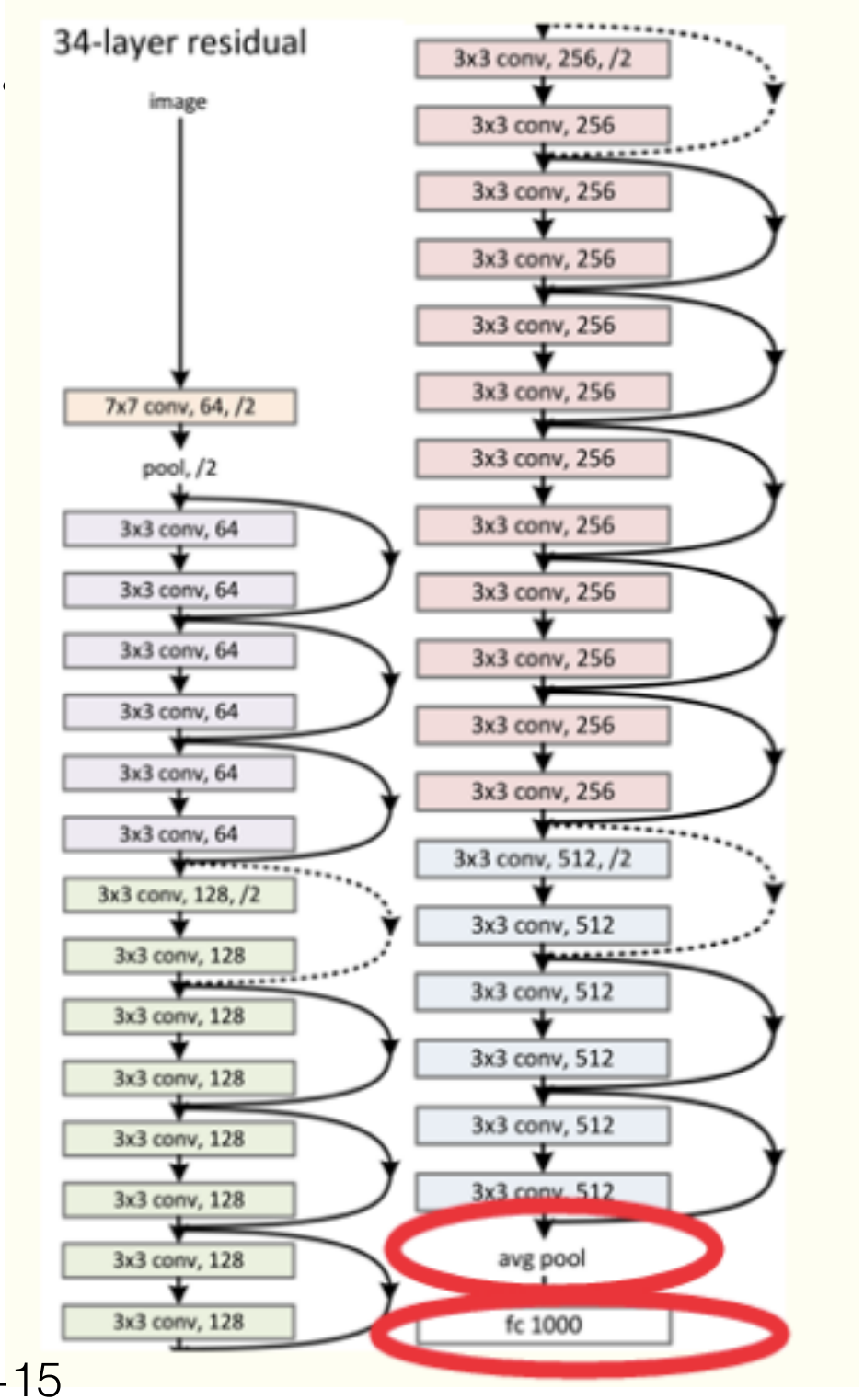


IN THE REAL LIFE…

RESNET

DO WE NEED TO GO THIS DEEP FOR ASTRONOMY APPLICATIONS?

[34 layers - authors explored up to 1202!]



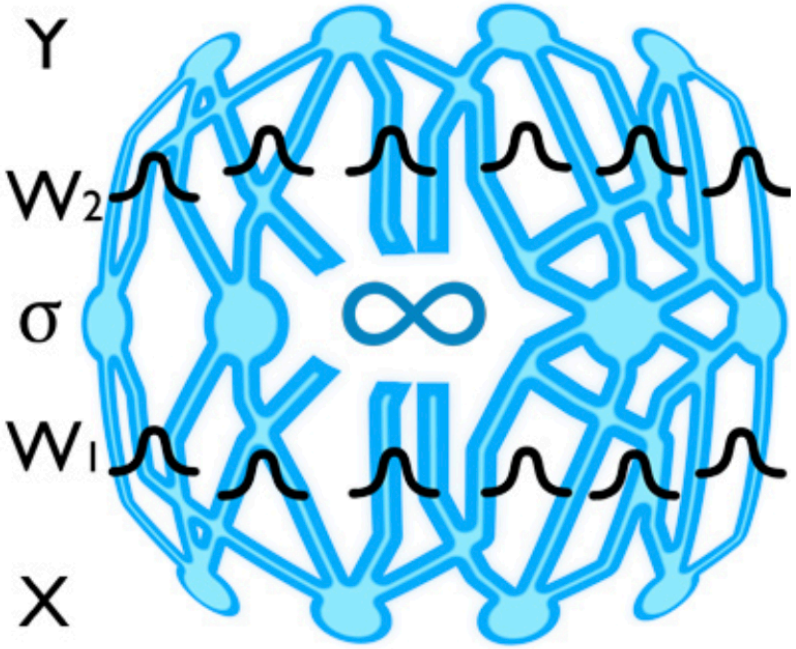
He+15

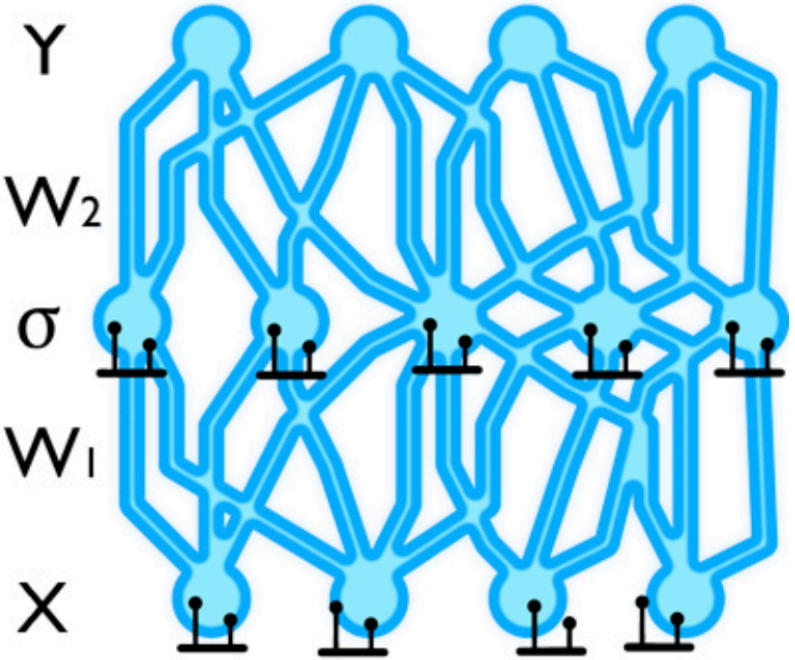
THE PROBLEMS OF GOING TOO DEEP…

CAPTURING THE MODEL UNCERTAINTY

NEURAL NETWORKS AS BAYESIAN MODELS

Denker&LEcun91, Neal+95, Graves+11, Kingma+15, **Gal+15…**





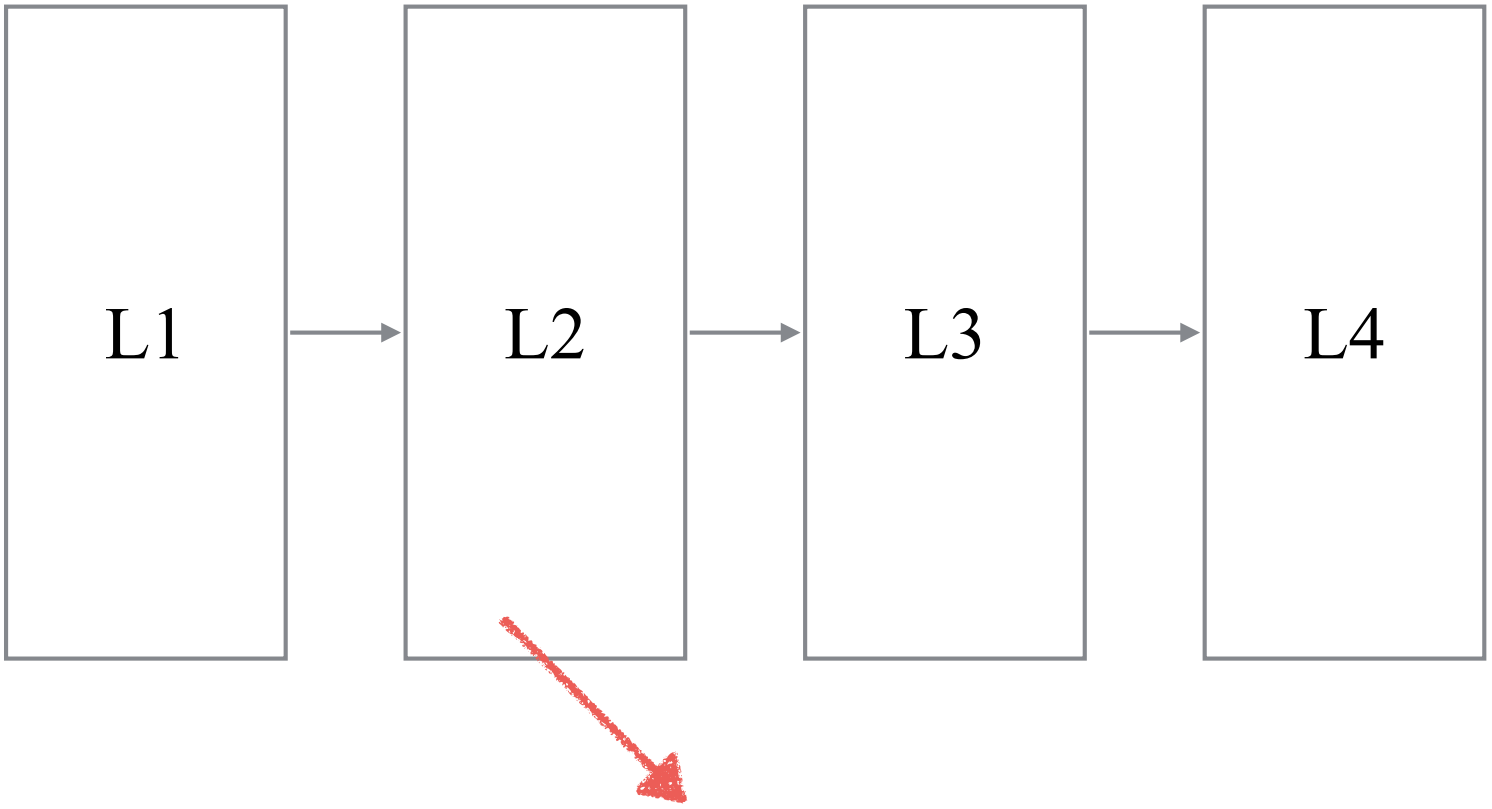
BNNs ADD A PRIOR DISTRIBUTION TO EACH WEIGHT - HARD TO TRAIN

GAL+15 SHOW THAT DROPOUT CAN BE USED TO ESTIMATE UNCERTAINTY

IMPLEMENTATION IN KERAS / TENSORFLOW

```
#========== Model definition=======

#Convolutional Layers

model = Sequential()
model.add(Convolution2D(32, 6,6, border_mode='same',
                        input_shape=(img_channels, img_rows, img_cols)))
model.add(Activation('relu'))
model.add(Dropout(0.5))

model.add(Convolution2D(64, 5, 5, border_mode='same'))
model.add(Activation('relu'))

model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Convolution2D(128, 2, 2, border_mode='same'))
model.add(Activation('relu'))

model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Convolution2D(128, 3, 3, border_mode='same'))
model.add(Activation('relu'))

model.add(Dropout(0.25))

#Fully Connected start here
#-------------------------------#

model.add(Flatten())
model.add(Dense(64, activation='relu'))
model.add(Dropout(.5))
model.add(Dense(1, init='uniform', activation='sigmoid'))

print("Compilation...")

model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
```

CONVNET OR CNN

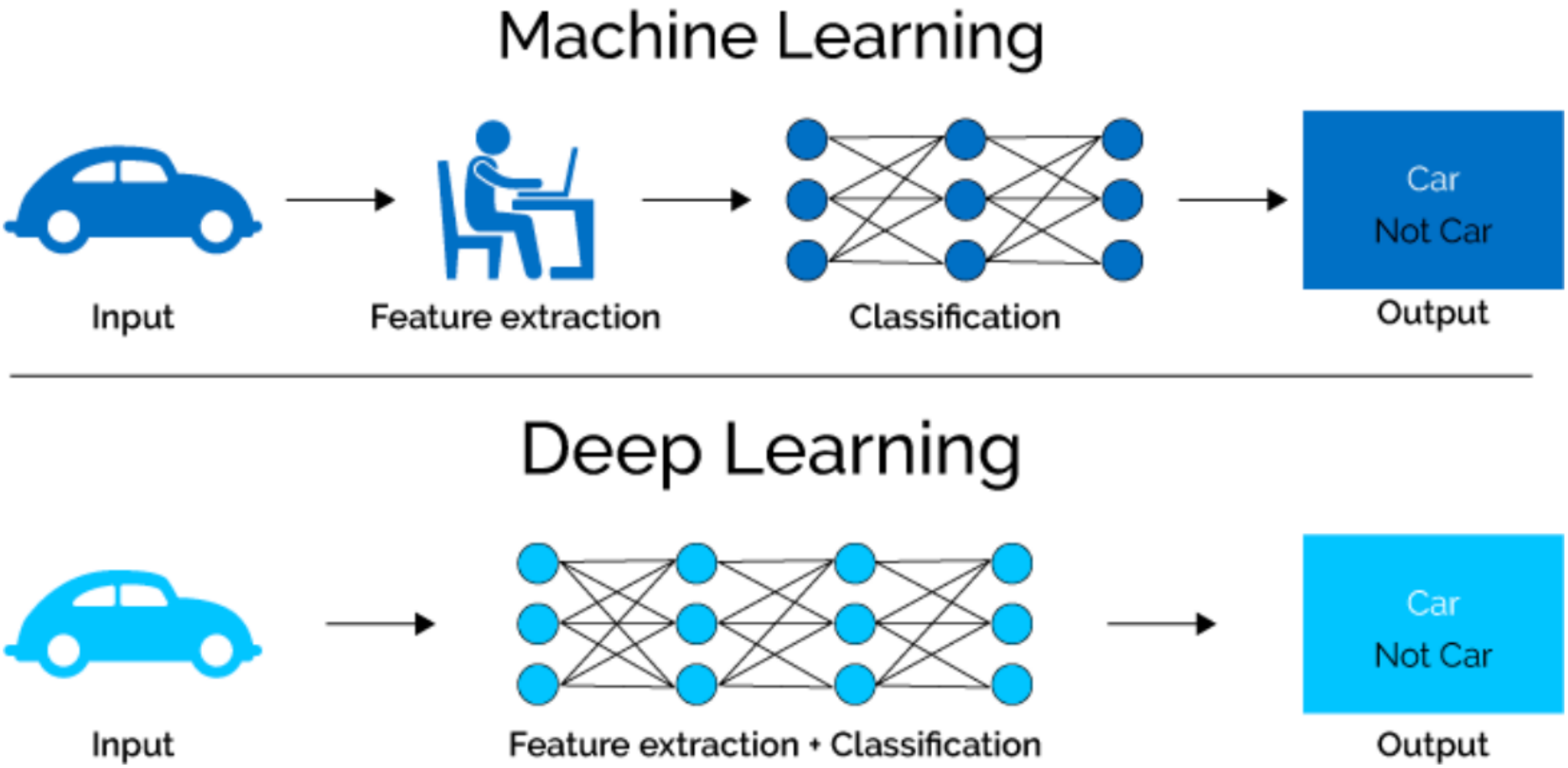


EACH BLOCK TYPICALLY MADE OF:

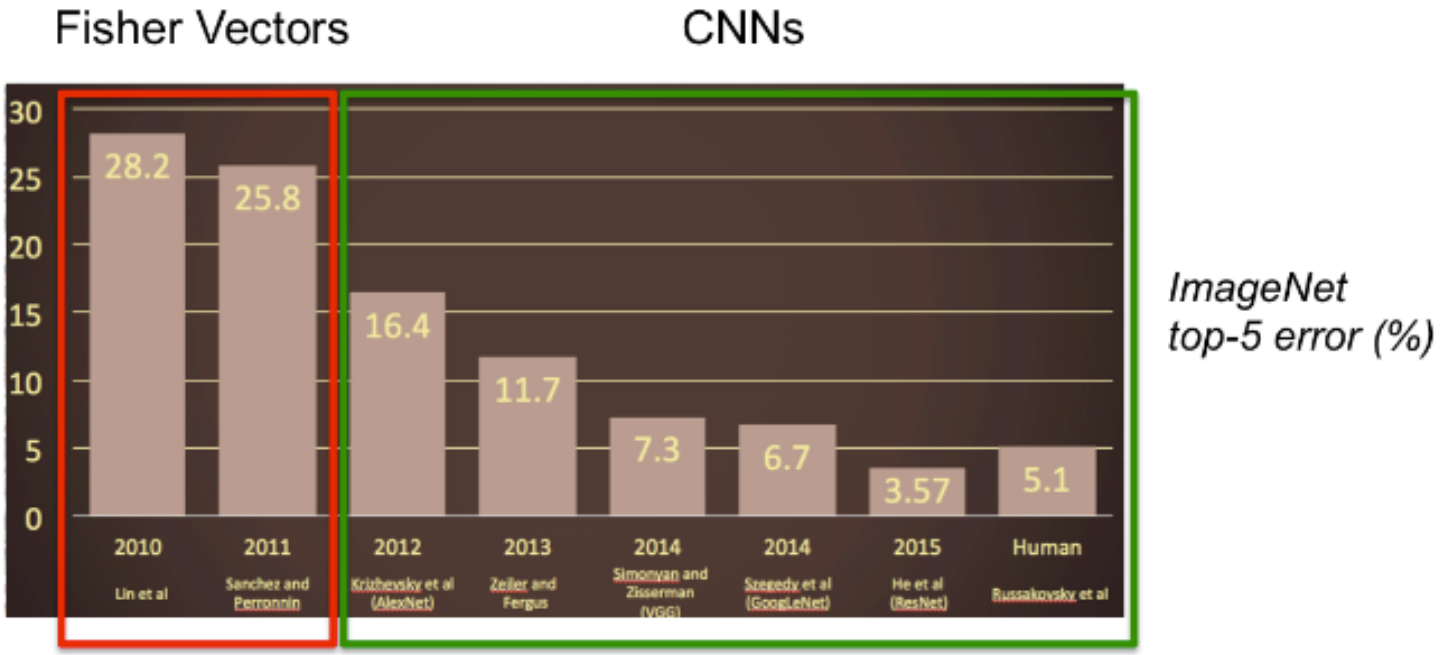
CONV | ACTIVATION | POOLING

(+dropout for training)

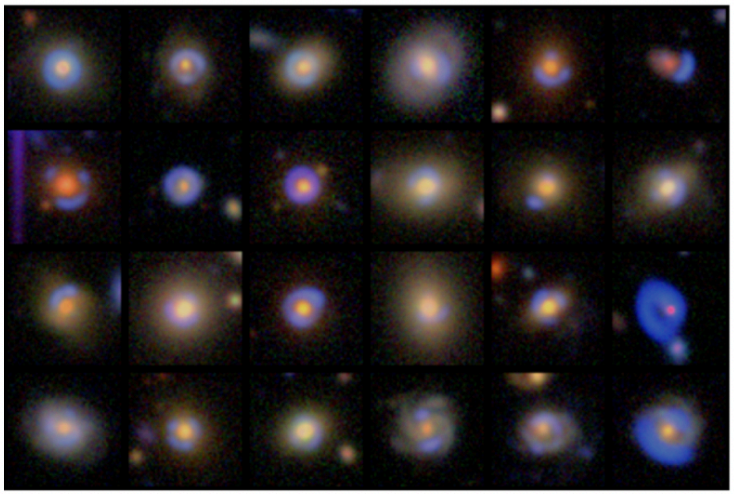
THIS IS A CHANGE OF PARADIGM!



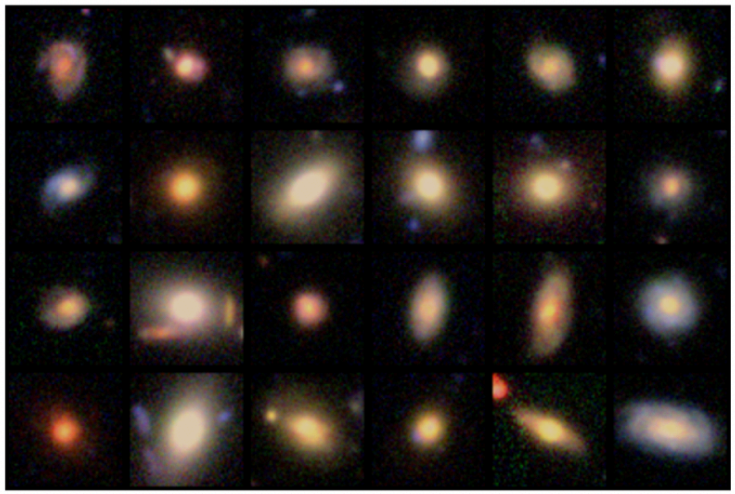
THIS IS A CHANGE OF PARADIGM!



1. Classification



LENS



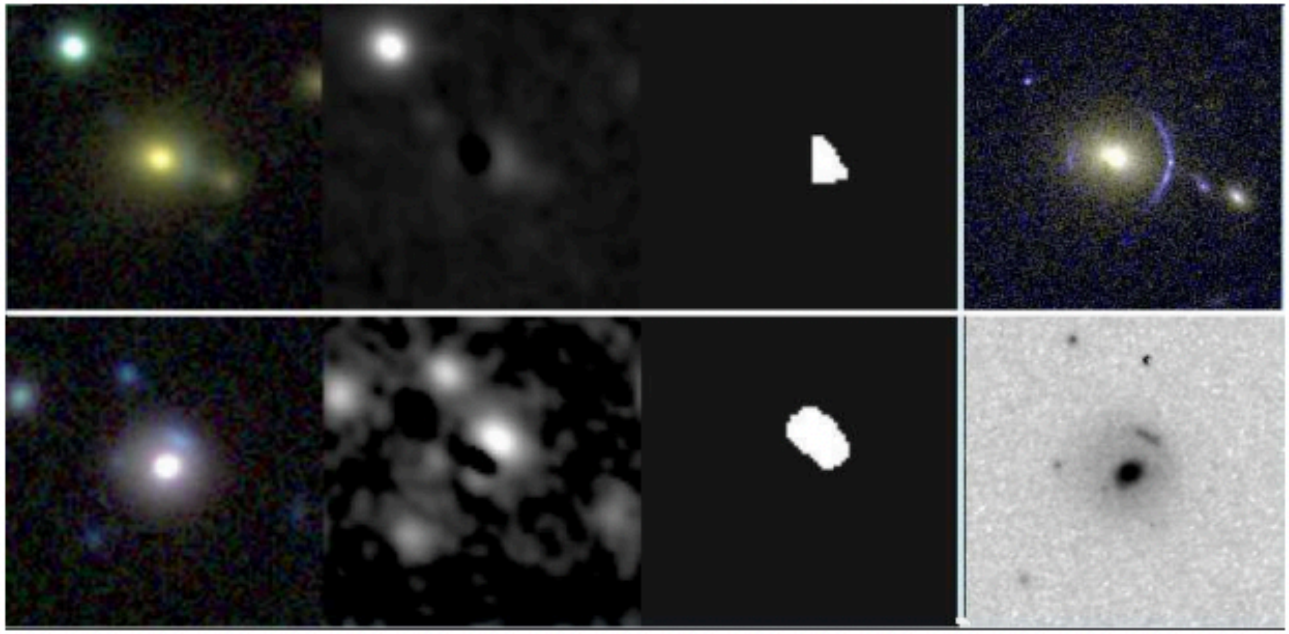
NON-LENS

Detection of Strong Lenses
Valuable information of
Dark Matter properties

Future surveys will increase the samples by orders of magnitude.

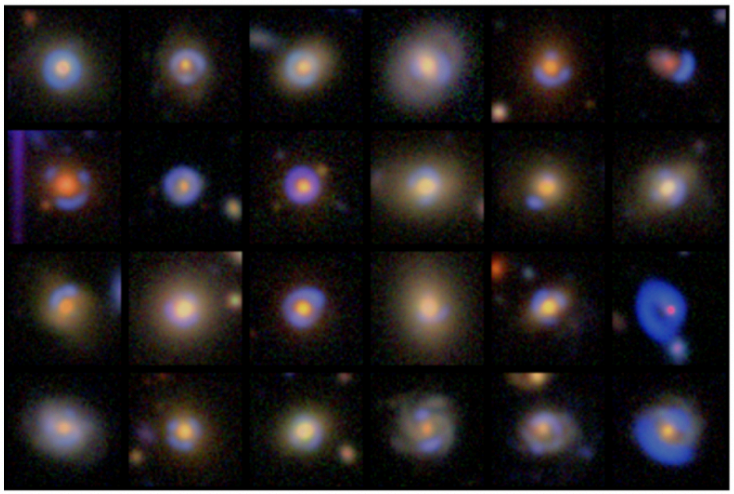
Jacobs+17

"Pre Deep Learning" Approach

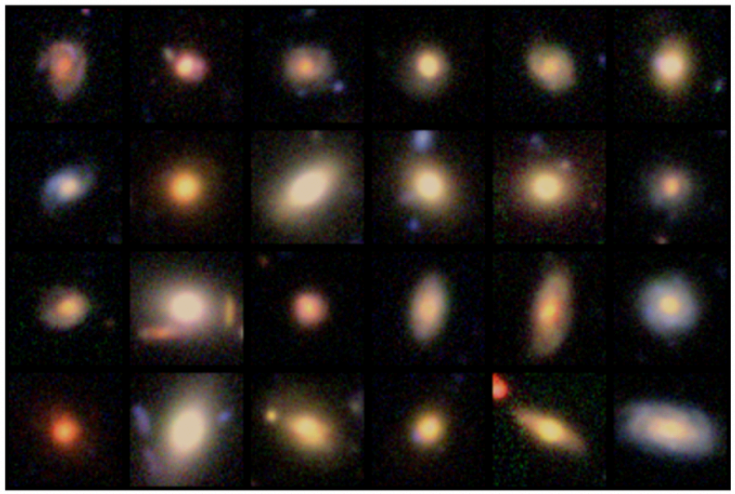


Gavazzi+17

1. Classification

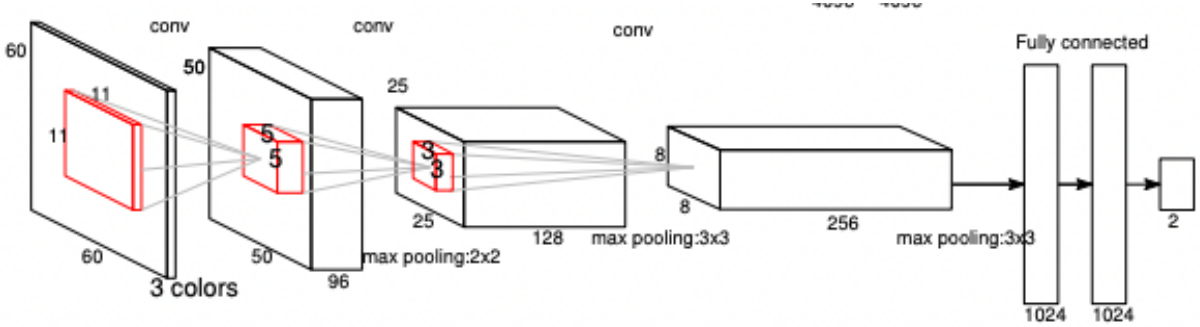


LENS

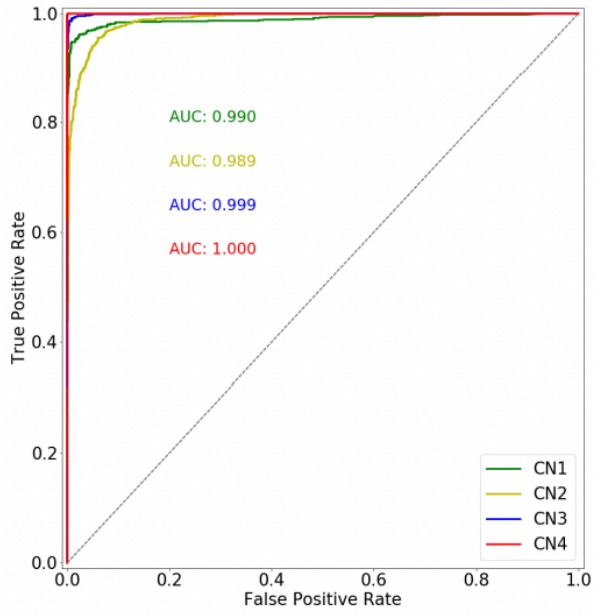


NON-LENS

Jacobs+17



It illustrates the change of paradigm from an algorithmic centric focus to a purely data driven approach to data



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

1. Classification

CNN based classifications reach unprecedented accuracy

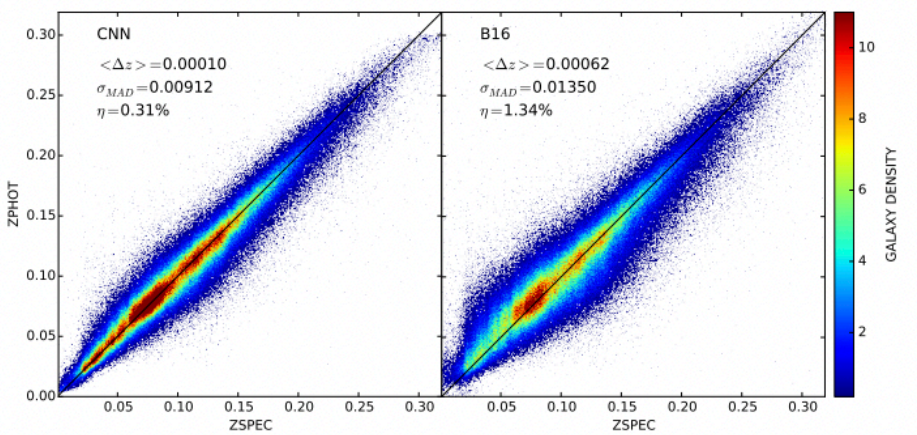
Name	type	AUROC	TPR_0	TPR_{10}	short description
CMU-DeepLens-Resnet-ground3	Ground-Based	0.98	0.09	0.45	CNN
CMU-DeepLens-Resnet-Voting	Ground-Based	0.98	0.02	0.10	CNN
LASTRO EPFL	Ground-Based	0.97	0.07	0.11	CNN
CAS Swinburne Melb	Ground-Based	0.96	0.02	0.08	CNN
AstrOmatic	Ground-Based	0.96	0.00	0.01	CNN
Manchester SVM	Ground-Based	0.93	0.22	0.35	SVM / Gabor
Manchester2	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+19

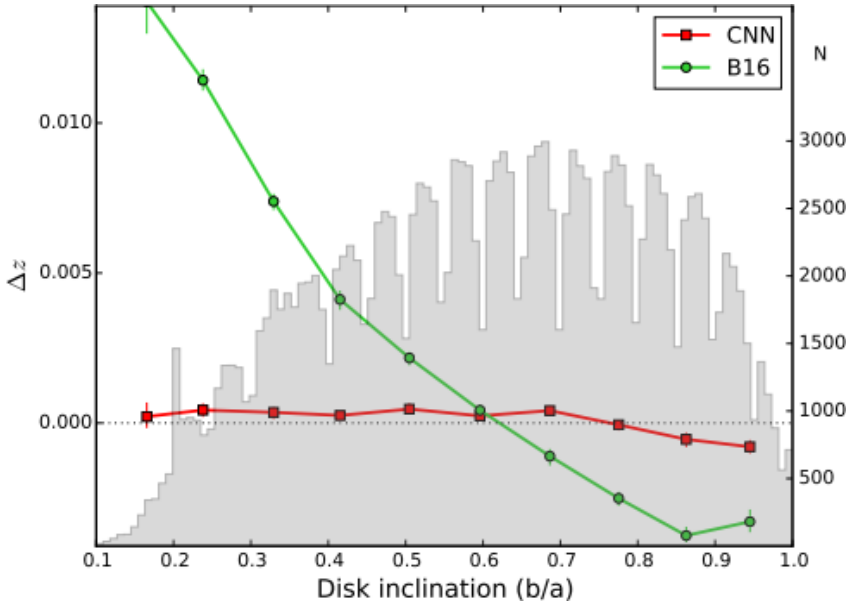
Dieleman+15, Huertas-Company+15, Aniyan+17, Charnock+17, Gieseke+17, Jacobs+17, Petrillo+17, Schawinski+17, Alhassan+18, Dominguez-Sanchez+18, George+18, Hinners+18, Lukic+18, Moss+18, Razzano+18, Schaefer+18, Allen+19, Burke+19, Carrasco-Davis+19, Chatterjee+19, Davies+19, Dominguez-Sanchez+19, Fusse+19, 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

Photometric Redshifts

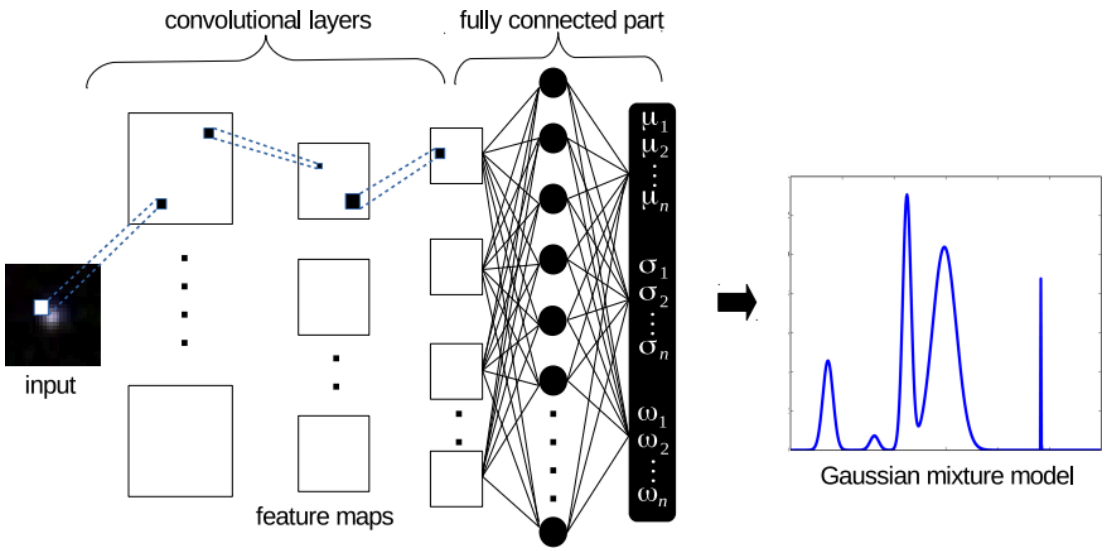
Deep Learning **Classical approach**



Geometric Effects are automatically considered (beyond photometry)



Pasquet+18

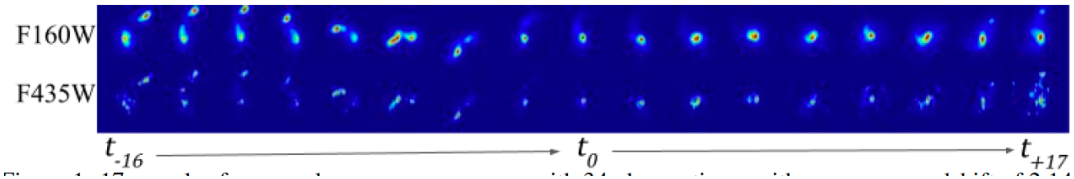


Disanto+18

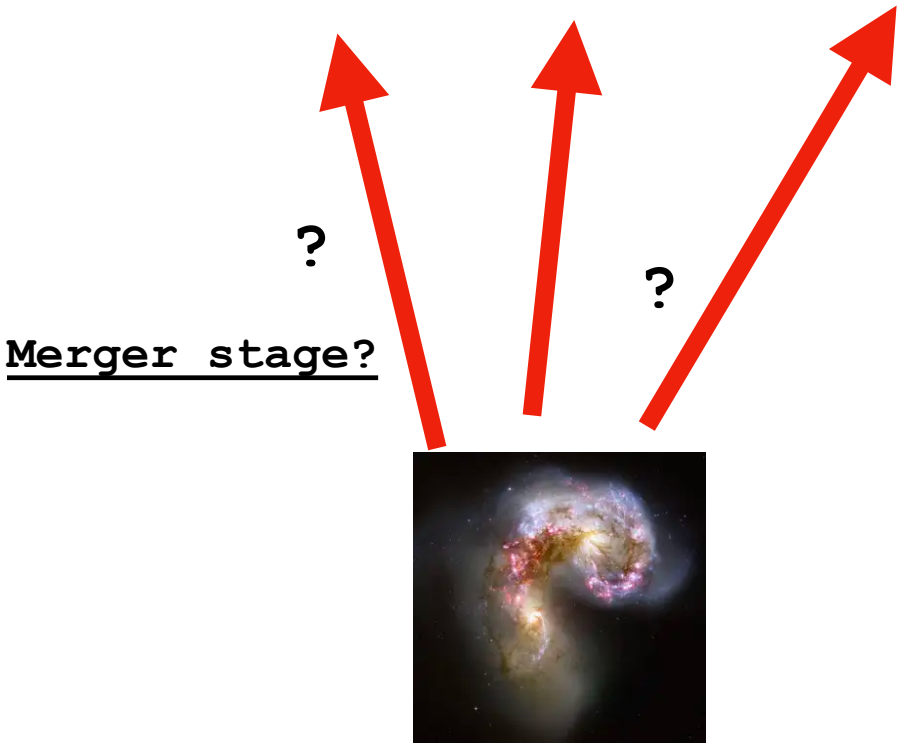
Uncertainty quantification through Mixture Density Networks

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

Mergers of Galaxies



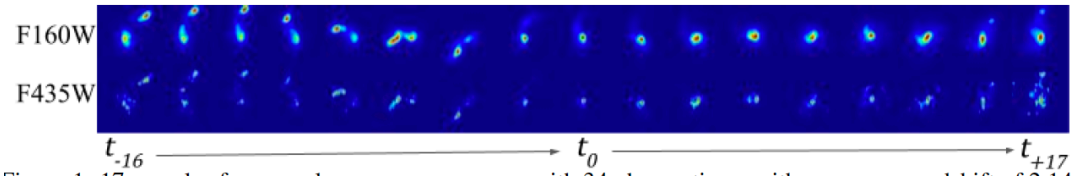
Merger of galaxies sequence from cosmological simulations



Neural Networks to find relations between observables and physical processes

Ntampaka+16, Huertas-Company+18, Schawinski+18, Armitage+19, Bottrell+19, Calderon+19, Diaz+19, Ho+19, Pearson+19, Cai+20, Ferreira+20, Hansen+20, Kodi Ramanah+20, Wu+20, Yan+20, Gupta+20, de Andres+21, Bickley+21, Ciprijanovic+21, Ho+21, Koppula+21, von Marttens+21, Shao+21, Villanueva-Domingo+21, Whithney+21, Holwerda+21

Mergers of Galaxies

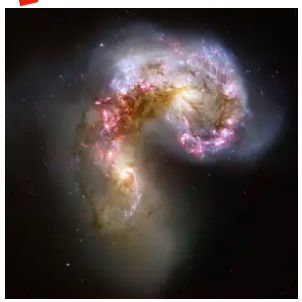


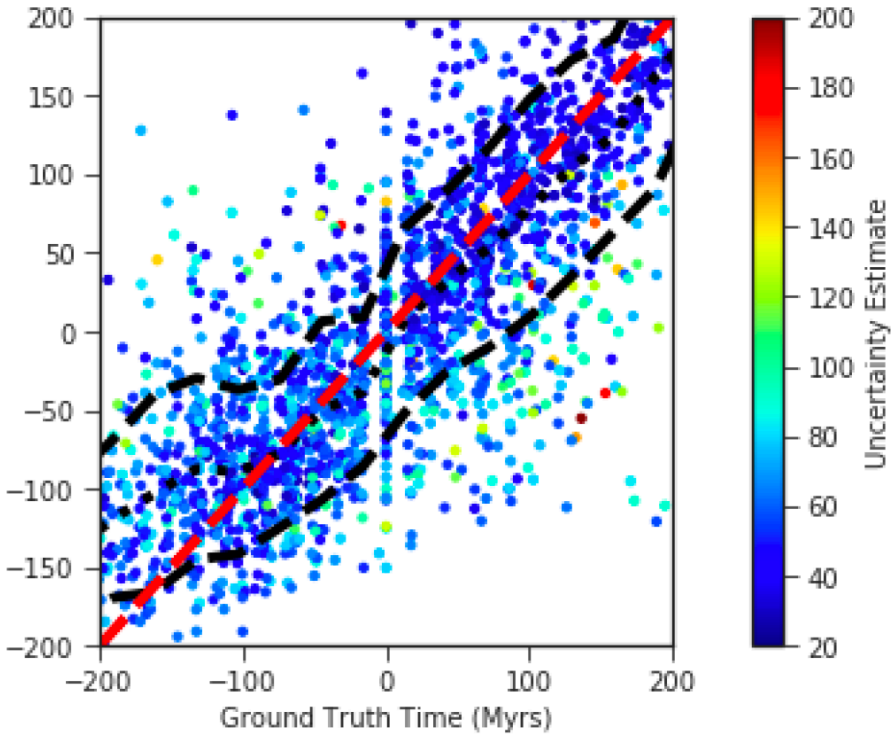
Merger of galaxies sequence from cosmological simulations

?

?

Merger stage?



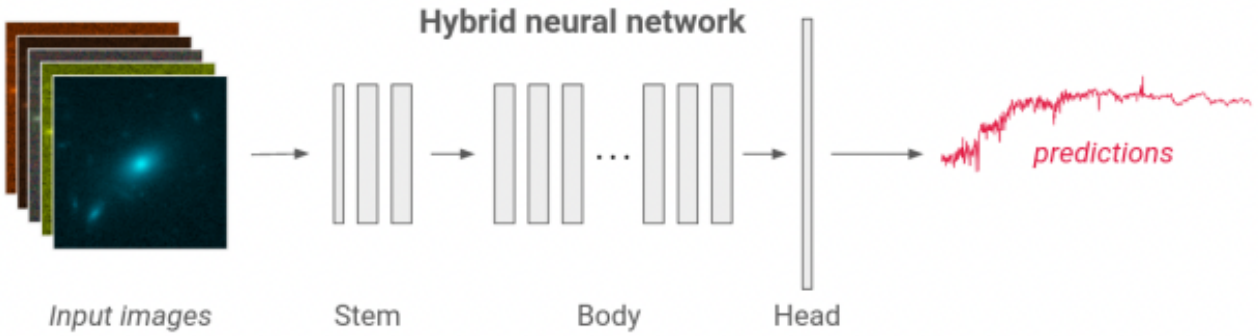


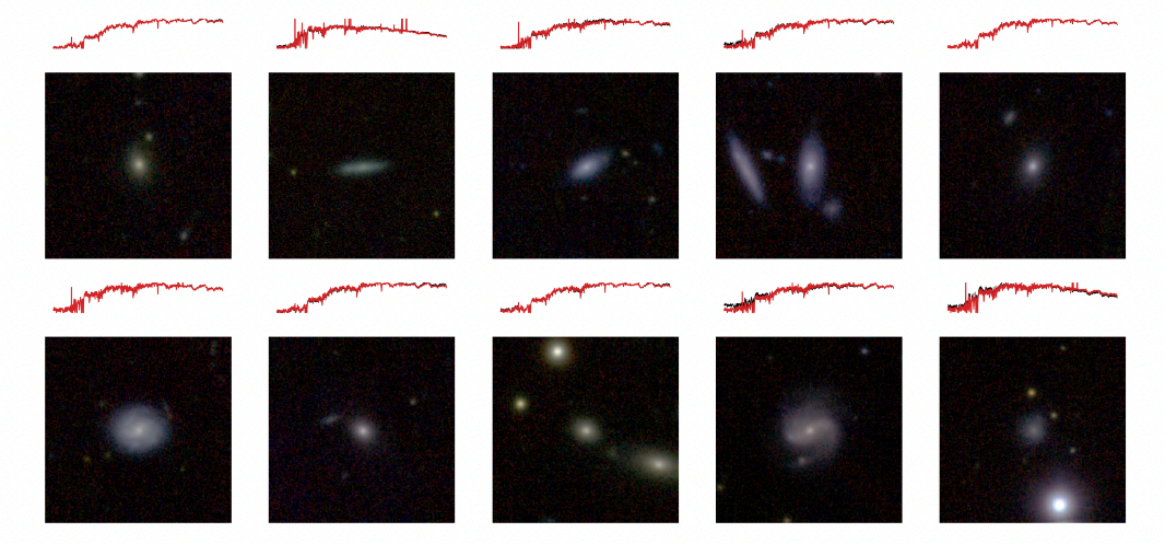
Koppula+21

Neural Networks to find relations between observables and physical processes

Ntampaka+16, Huertas-Company+18, Schawinski+18, Armitage+19, Bottrell+19, Calderon+19, Diaz+19, Ho+19, Pearson+19, Cai+20, Ferreira+20, Hansen+20, Kodi Ramanah+20, Wu+20, Yan+20, Gupta+20, de Andres+21, Bickley+21, Ciprijanovic+21, Ho+21, Koppula+21, von Marttens+21, Shao+21, Villanueva-Domingo+21, Whithney+21, Holwerda+21

Virtual Observatory



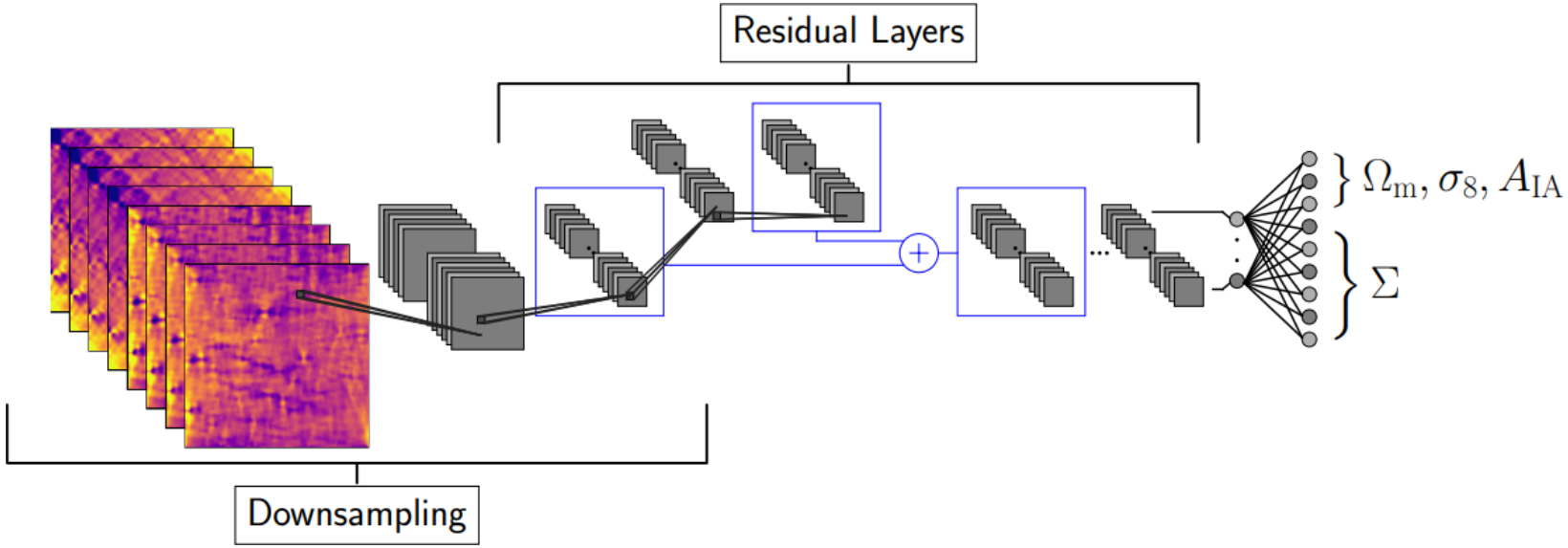


Neural Networks to learn complex mapping between observables

Wu+21

Ntampaka+16, Huertas-Company+18, Schawinski+18, Armitage+19, Bottrell+19, Calderon+19, Diaz+19, Ho+19, Pearson+19, Cai+20, Ferreira+20, Hansen+20, Kodi Ramanah+20, Wu+20, Yan+20, Gupta+20, de Andres+21, Bickley+21, Ciprijanovic+21, Ho+21, Koppula+21, von Marttens+21, Shao+21, Villanueva-Domingo+21, Whithney+21, Holwerda+21

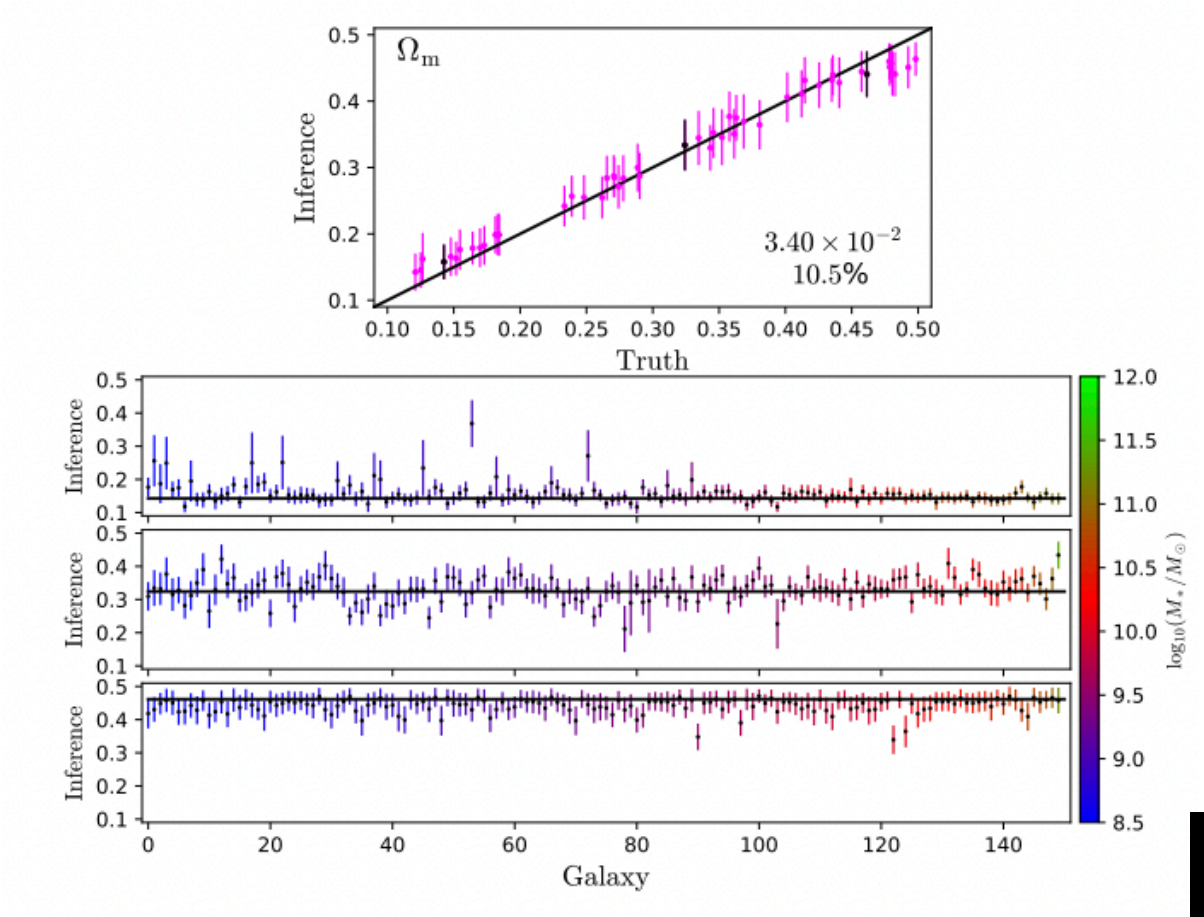
Deep Learning for Cosmological Inference

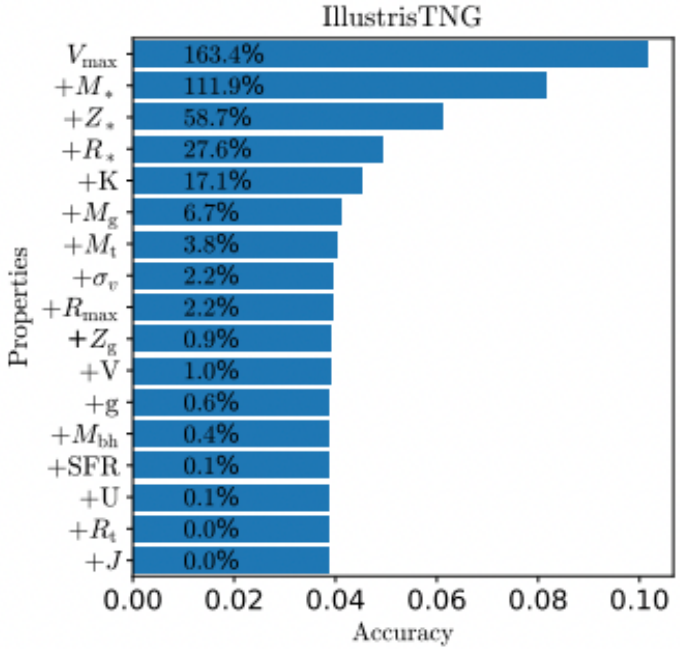


Motivation: Generalize comparison of observations with theory, beyond basic summary statistics

Neural Networks are used as efficient feature extractors

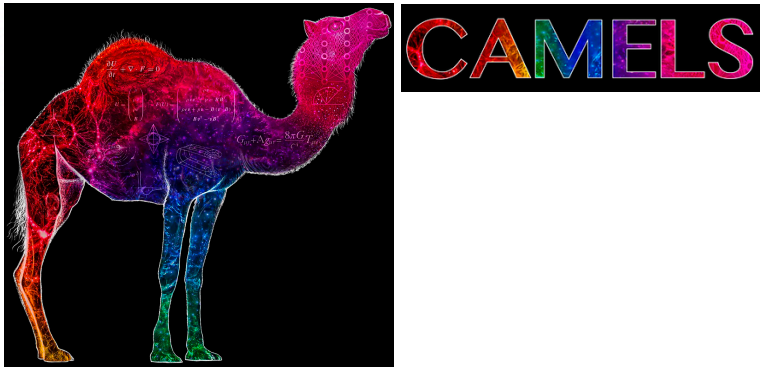
Cosmological Inference





Villaescusa-Navarro+22

Neural Networks to learn new relations between models and observables

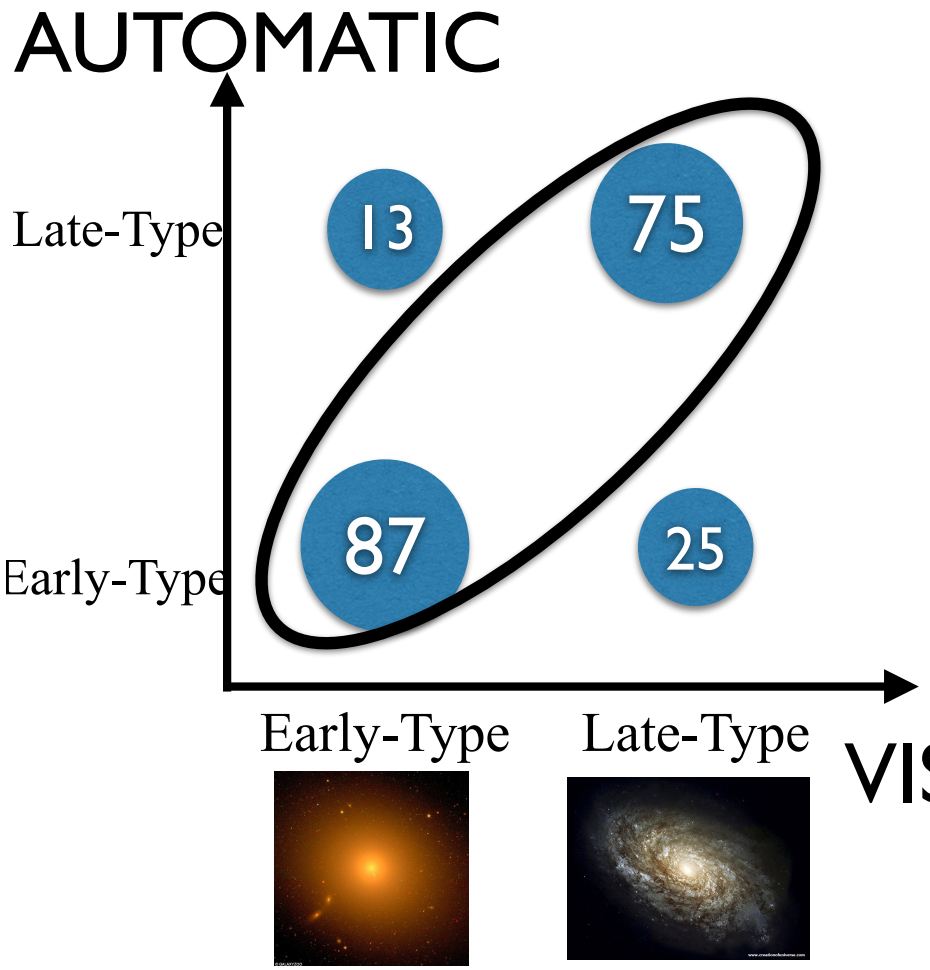


Ravanbakhsh+17, Brehmer+19, Ribli+19, Pan+19, Ntampaka+19, Alexander+20, Arjona+20, Coogan+20, Escamilla-Rivera+20, Hortua+20, Vama+20, Vernardos+20, Wang+20, Mao+20, Arico+20, Villaescusa_navarro+20, Singh+20, Park+21, Modi+21, Villaescusa-Navarro+21ab, Moriwaki+21, DeRose+21, Makinen+21, Villaescusa-Navaroo+22

ALSO FOR GALAXY MORPHOLOGY

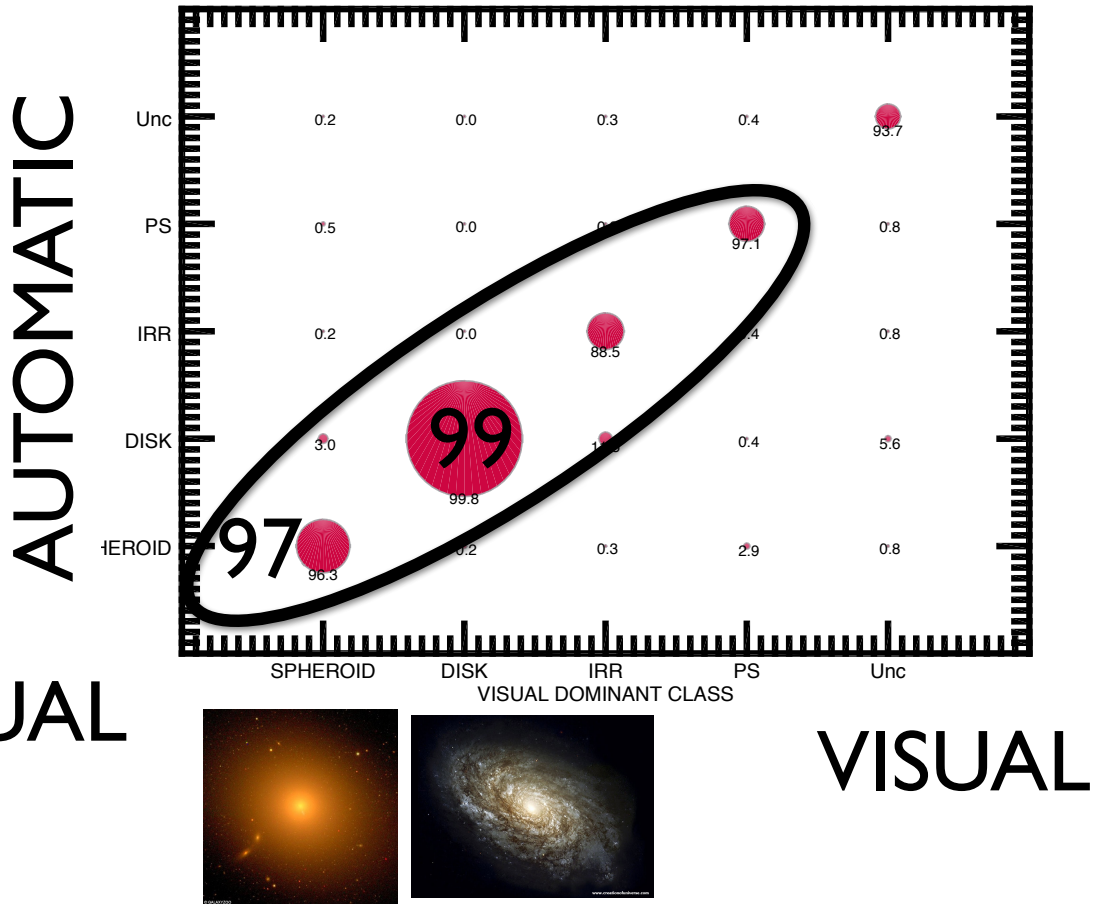
SVMs

[HUERTAS-COMPANY+14]

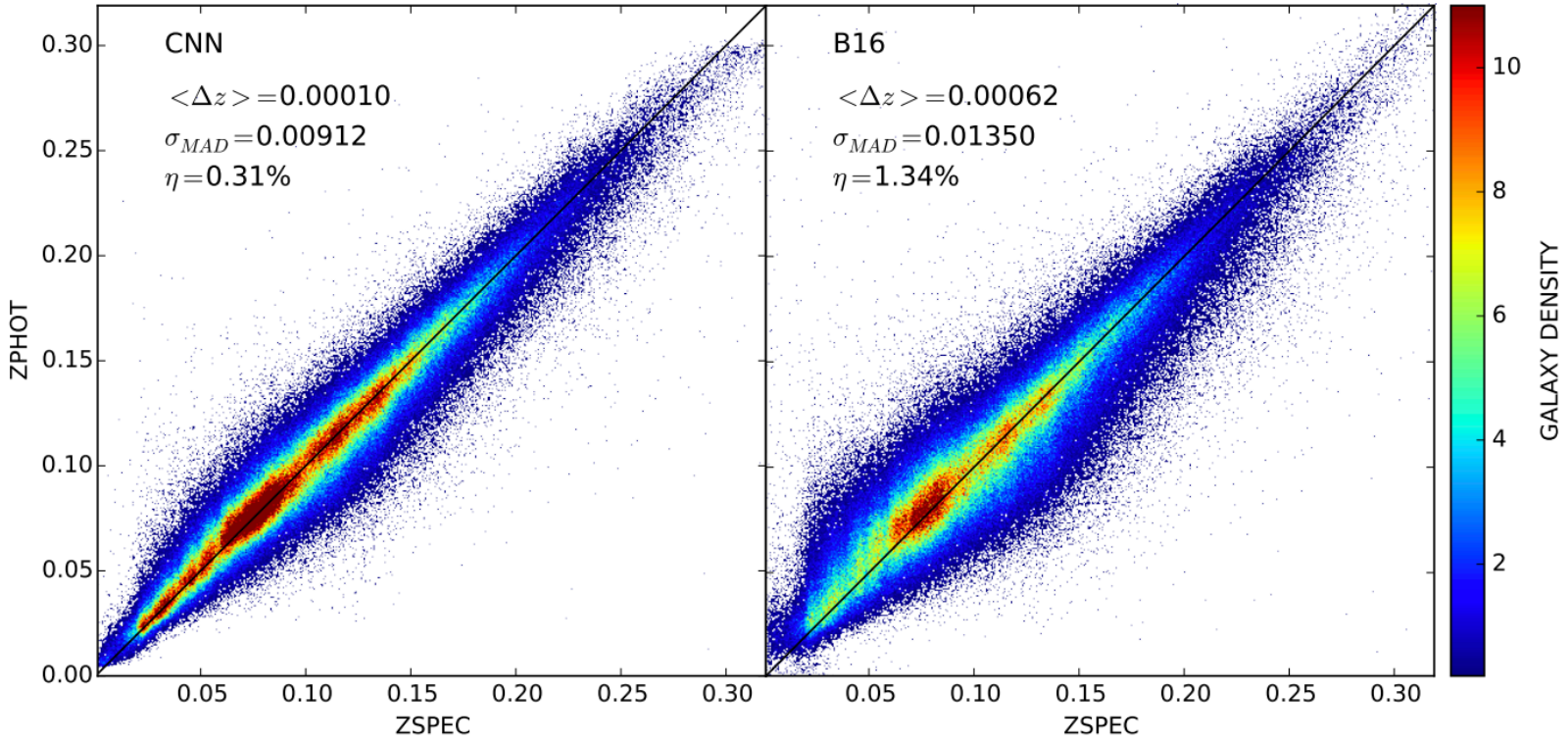


CNNs

[HUERTAS-COMPANY+15b]



PHOTOMETRIC REDSHIFTS



Pasquet+18

AUTOMATICALLY COMBINING MORPHOLOGY AND COLOR FOR PHOTOZ ESTIMATION

Making deep neural networks more robust

DATA AUGMENTATION

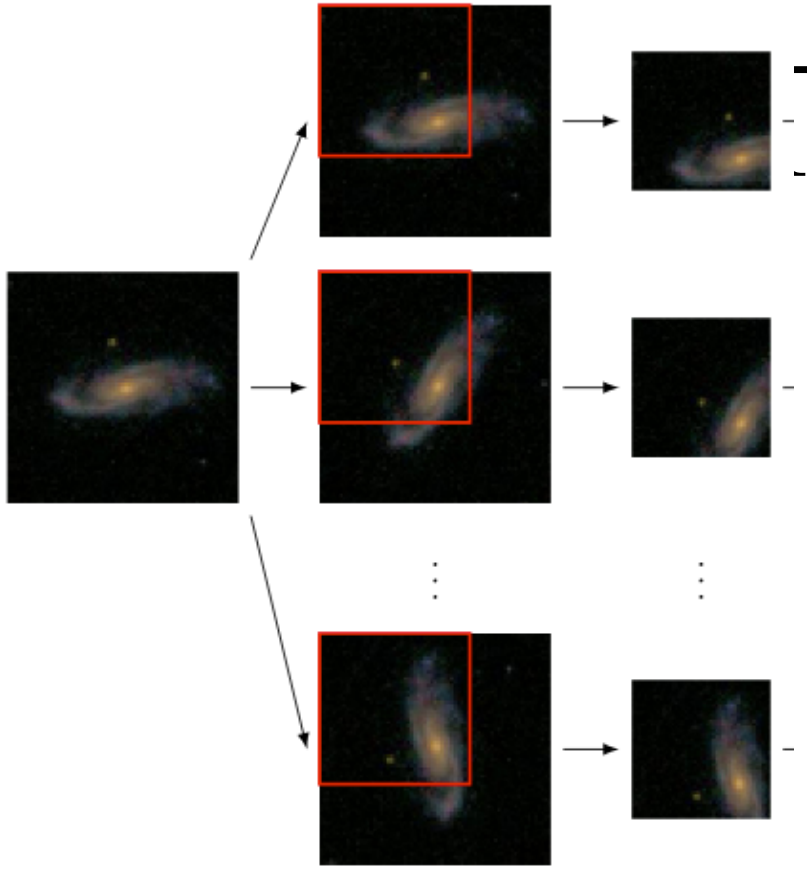
CNNs are invariants to translations

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:

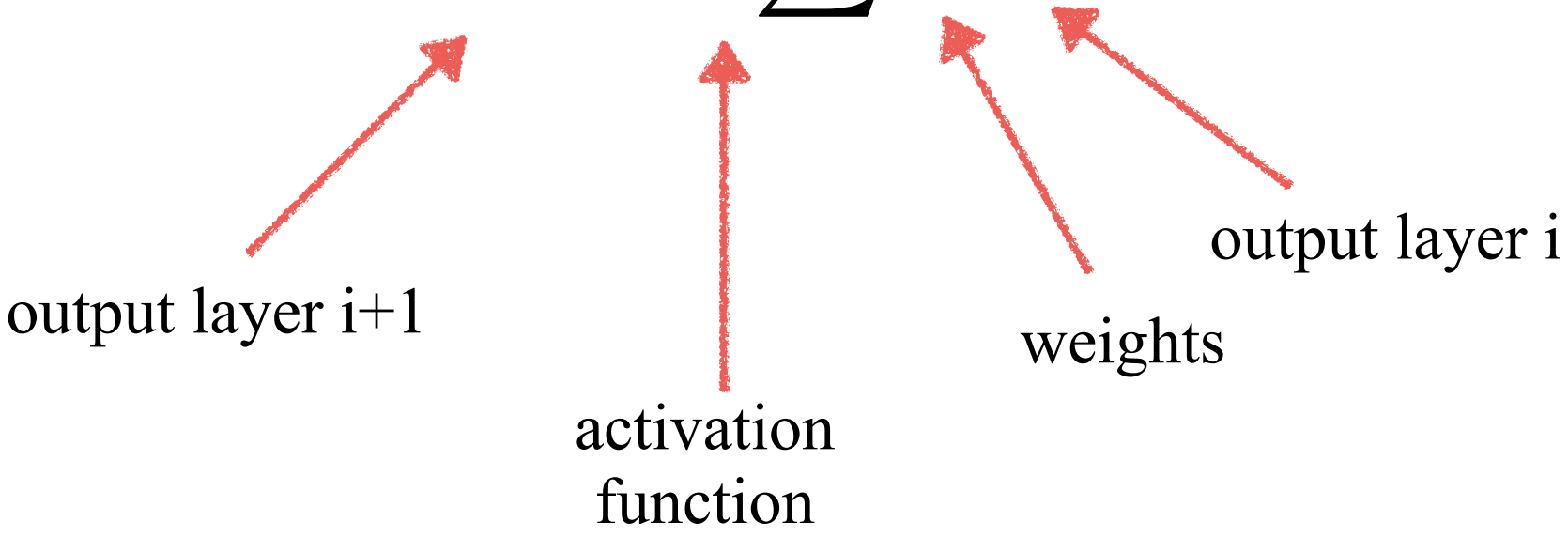
DEFINES THE RANGE OF PERTURBATIONS APPLIED TO IMAGES DURING TRAINING

```
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])
```

VANISHING / EXPLODING GRADIENT PROBLEM

REMEMBER THAT:

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



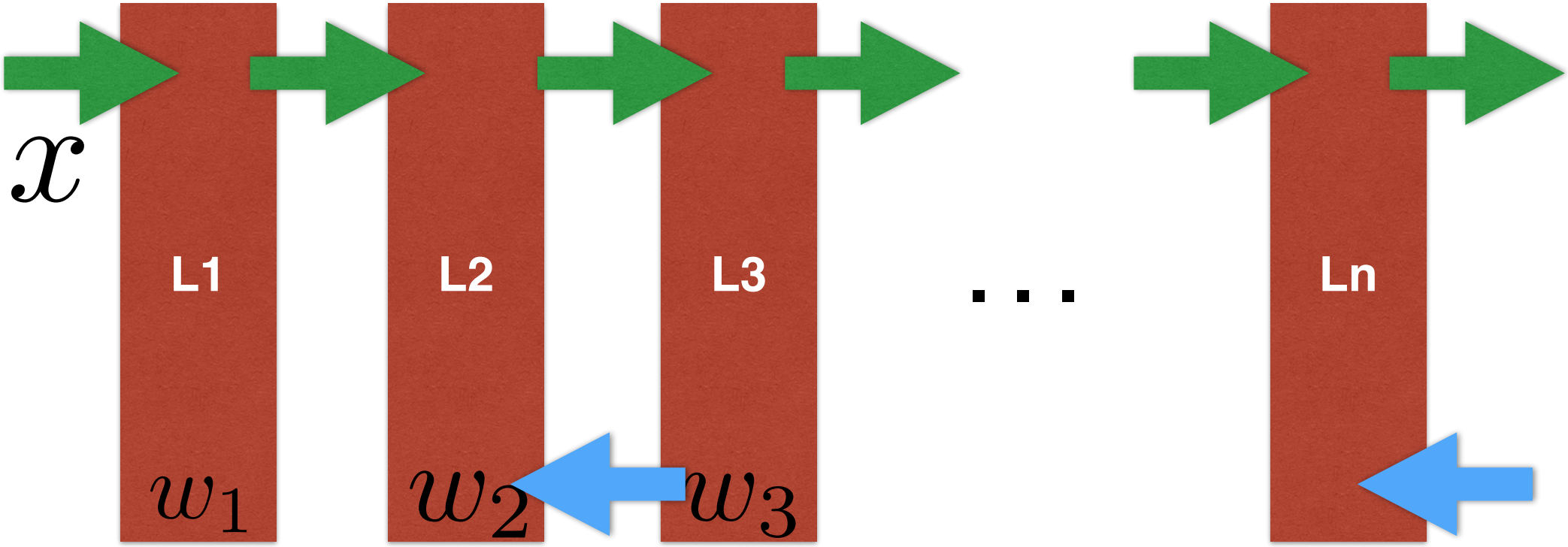
VANISHING / EXPLODING GRADIENT PROBLEM

WITH MANY LAYERS:

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

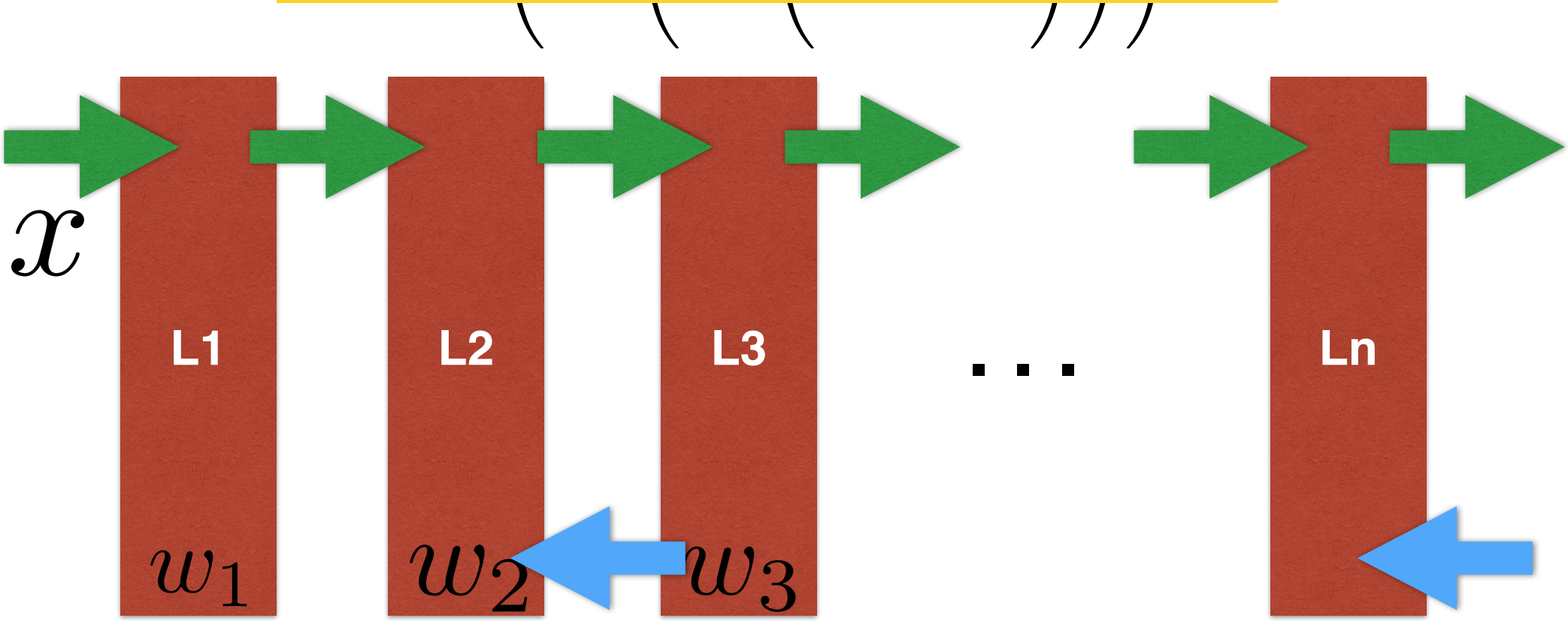
VANISHING/EXPLODING GRADIENT PROBLEM

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

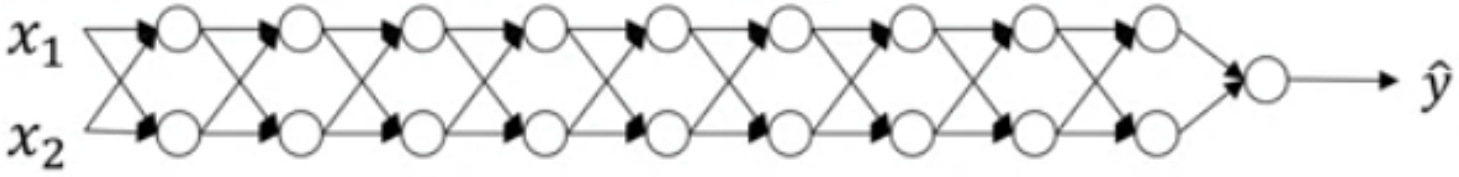


VANISHING/EXPLODING GRADIENT PROBLEM

TRAINING BECOMES UNSTABLE
VERY SLOW OR NO CONVERGENCE



VANISHING/EXPLODING GRADIENT PROBLEM



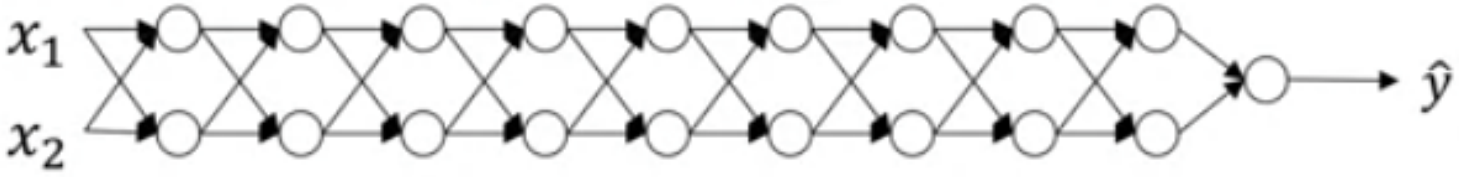
IF WE ASSUME AN IDENTITY ACTIVATION FUNCTION:

$$\hat{y} = x \prod_n w_i$$

with:

$$w_i = \begin{pmatrix} w_i^0 & 0 \\ 0 & w_i^1 \end{pmatrix} \qquad x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

VANISHING/EXPLODING GRADIENT PROBLEM



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

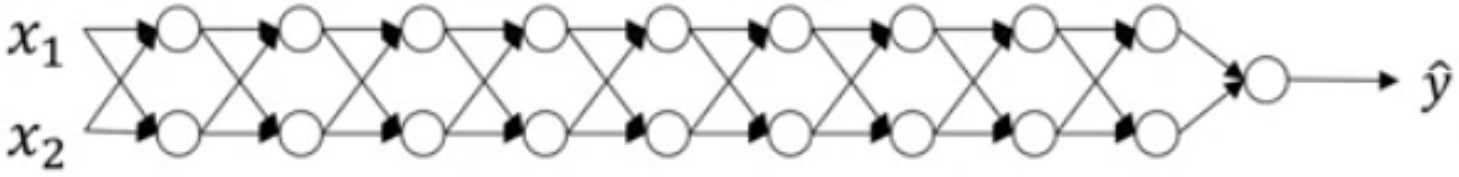
$$x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

IF WEIGHTS ARE ALL INITIALIZED TO VALUES <<1:

$$w_i^L \to 0$$

VANISHING GRADIENT

VANISHING/EXPLODING GRADIENT PROBLEM



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

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

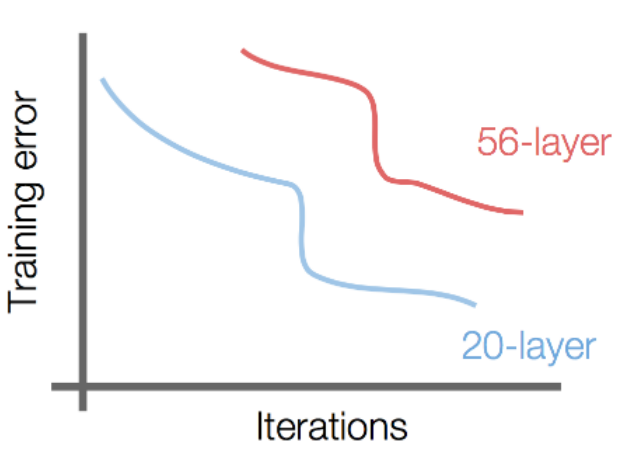
IF WEIGHTS ARE ALL INITIALIZED TO VALUES >1:

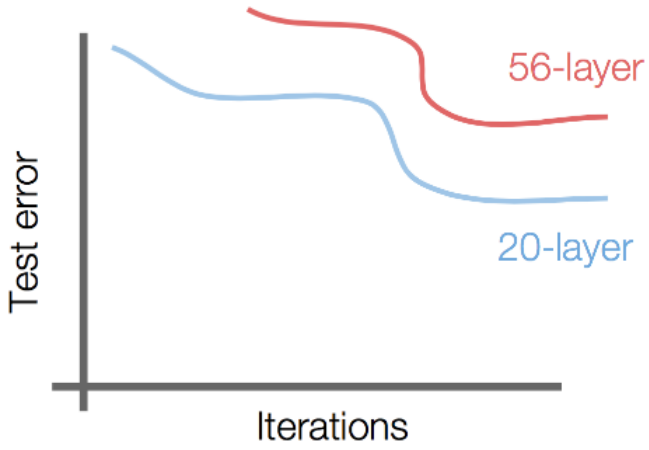
$$w_i^L \to \infty$$

EXPLODING GRADIENT

VANISHING/EXPLODING GRADIENT PROBLEM

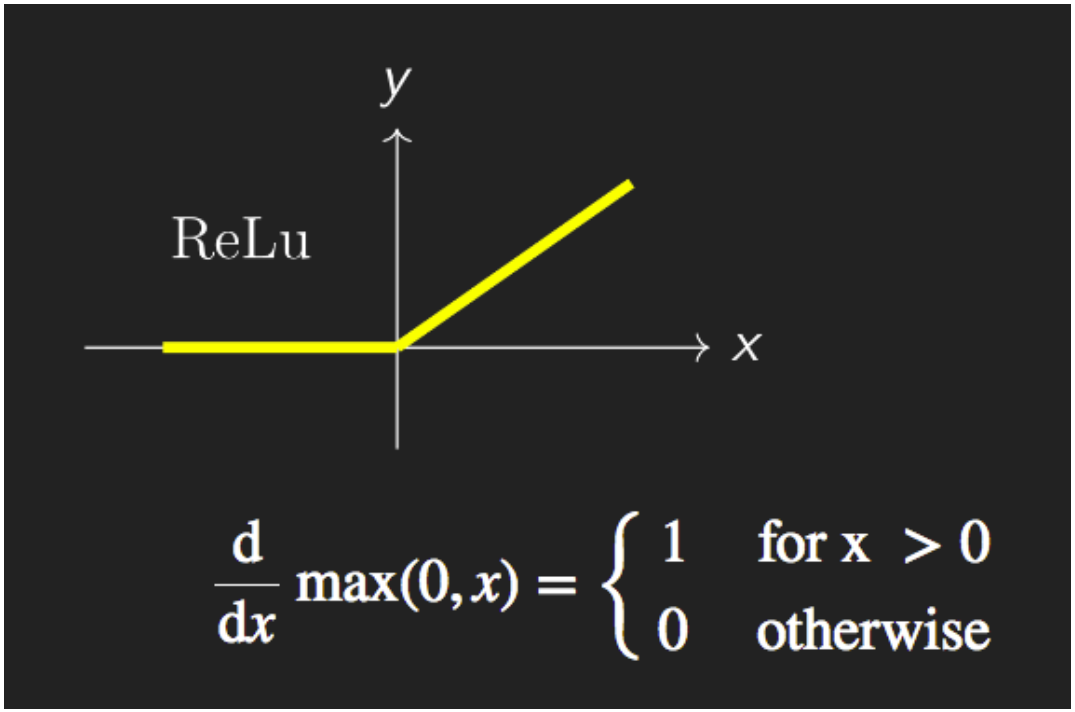
TRAINING BECOMES UNSTABLE
VERY SLOW OR NO CONVERGENCE





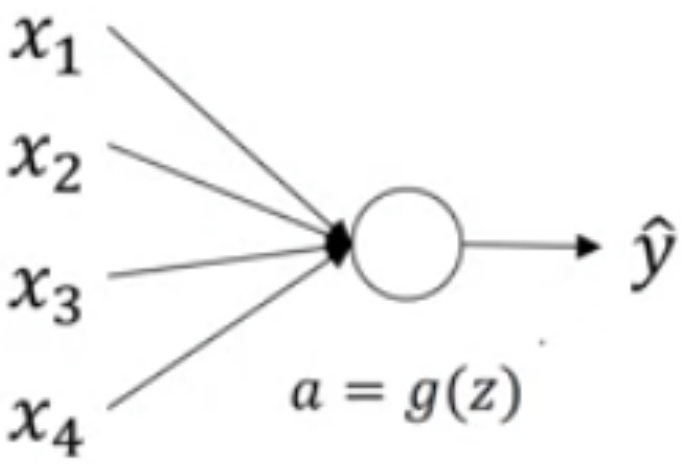
VANISHING GRADIENT PROBLEM

THE ReLU ACTIVATION FUNCTION HELPS PREVENTING VANISHING GRADIENTS



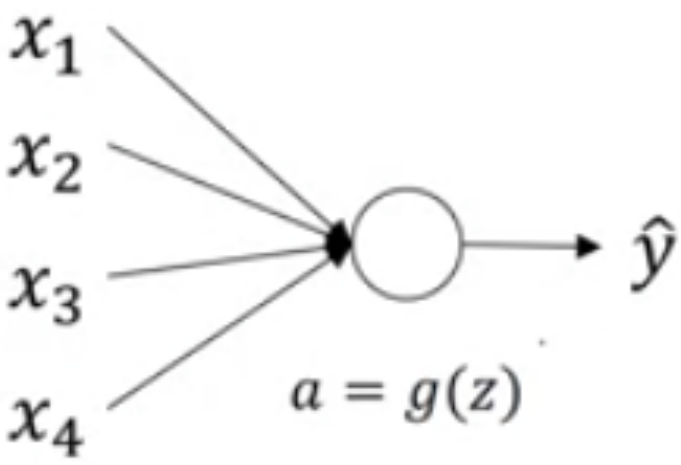
MOST WIDELY USED IN DEEP NETWORKS

WEIGHT INITIALIZATION IS A KEY POINT…



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

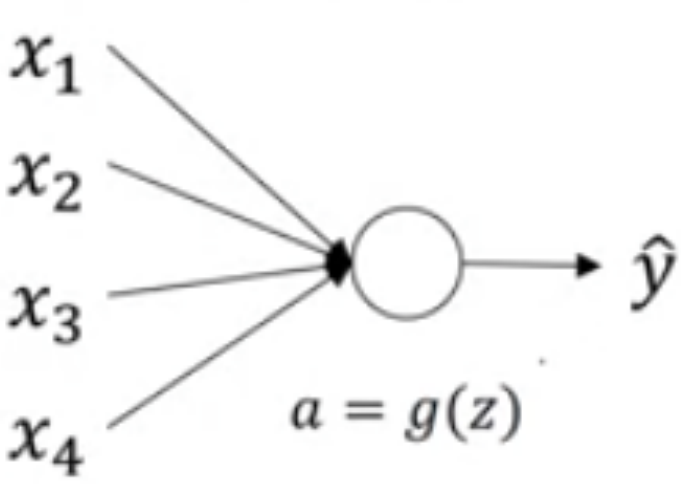
WEIGHT INITIALIZATION IS A KEY POINT…



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

THE LARGER n, THE SMALLER WEIGHTS SHOULD BE…

WEIGHT INITIALIZATION IS A KEY POINT…



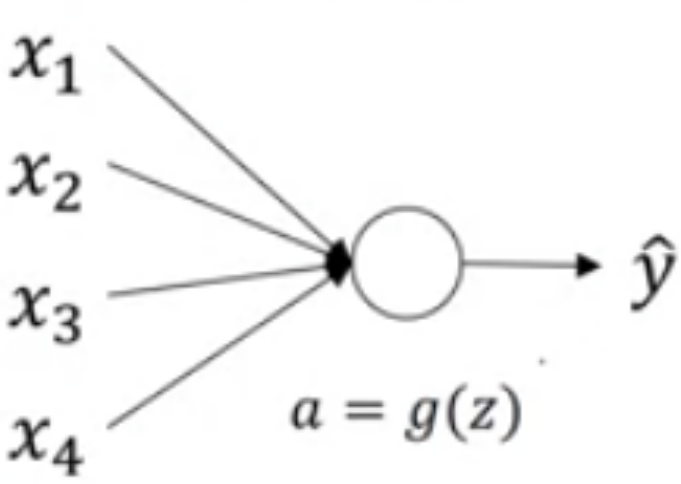
$$z = x_1 w_1 + x_2 w_2 + \ldots + x_n w_n$$

THE LARGER n, THE SMALLER WEIGHTS SHOULD BE…

variance

ONE SIMPLE SOLUTION: $\sigma^2(w_i) = \frac{1}{n}$ ← 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

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

WEIGHT INITIALIZATION IS A KEY POINT…

For ReLU activation functions we typically use:

$$\sigma^2(w_i) = \frac{2}{n}$$

[He initialization, He+15]

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:

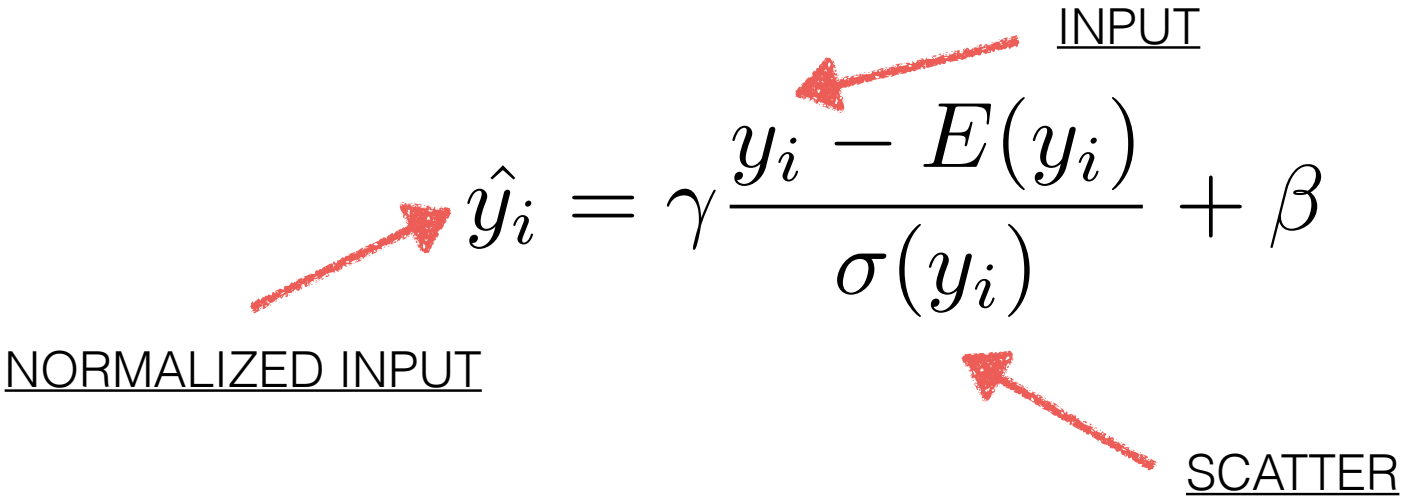
`keras.initializers` → https://keras.io/initializers/

BATCH NORMALIZATION

[SZEGEDY+15]

ANOTHER 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



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]

IN KERAS, IT IS IMPLEMENTED AS AN ADDITIONAL LAYER

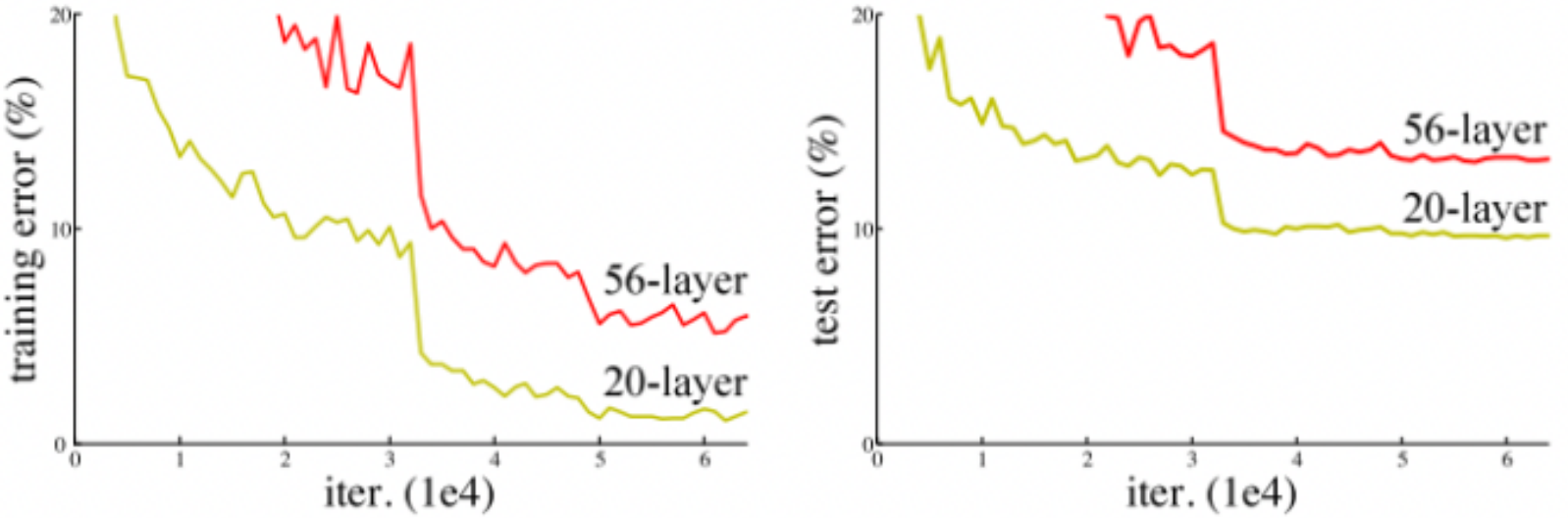
BatchNormalization [source]

```
keras.layers.BatchNormalization(axis=-1, momentum=0.99, epsilon=0.001, center=True, scale=True, beta_initializer
```

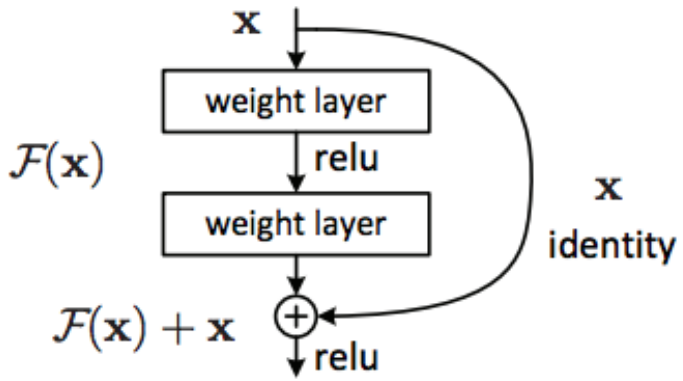
Batch normalization layer (Ioffe and Szegedy, 2014).

Normalize the activations of the previous layer at each batch, i.e. applies a transformation that maintains the mean activation close to 0 and the activation standard deviation close to 1.

Arguments

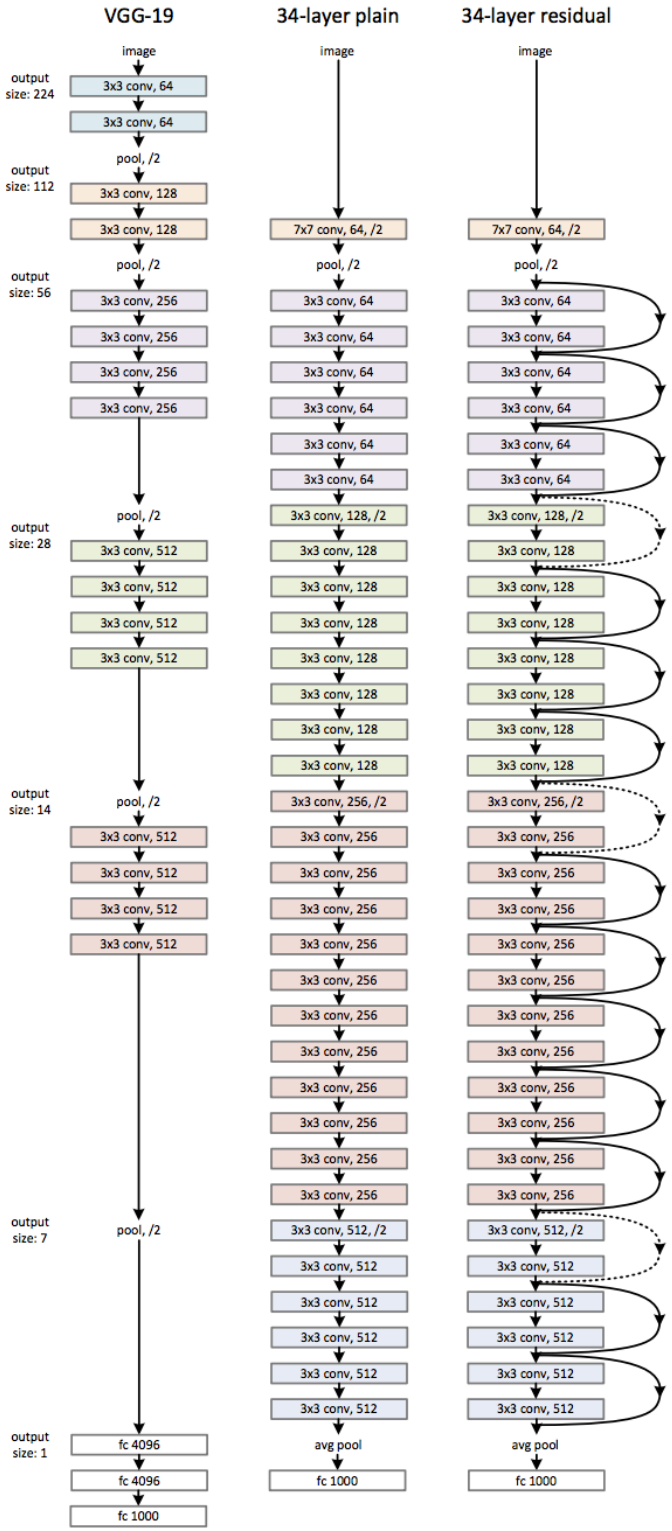

training error (%)
20
10
0
0
1
2
3
4
5
6
iter. (1e4)
56-layer
20-layer
test error (%)
20
10
0
0
1
2
3
4
5
6
iter. (1e4)
56-layer
20-layer

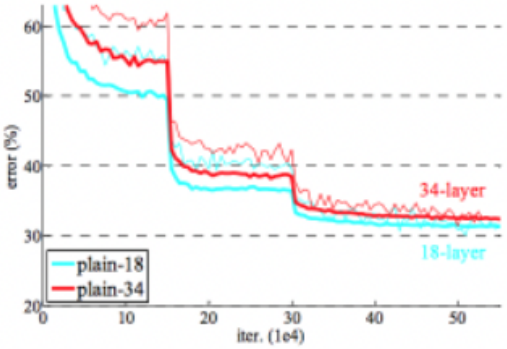
Residual Network

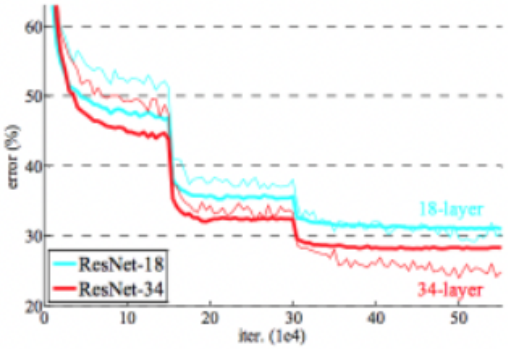


Enables an uninterrupted gradient flow

Learns the residual mapping instead of direct mapping


VGG-19
34-layer plain
34-layer residual
image
output size: 224
3x3 conv, 64
3x3 conv, 64
pool, /2
output size: 112
3x3 conv, 128
3x3 conv, 128
pool, /2
output size: 56
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
pool, /2
output size: 28
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
pool, /2
output size: 14
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
pool, /2
output size: 7
fc 4096
fc 4096
fc 1000
output size: 1
7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 128, /2
3x3 conv, 128
3x3 conv, 256, /2
3x3 conv, 256
3x3 conv, 512, /2
3x3 conv, 512
avg pool
fc 1000


error (%)
iter. (1e4)
plain-18
plain-34
34-layer
18-layer


error (%)
iter. (1e4)
ResNet-18
ResNet-34
18-layer
34-layer