Machine Learning for Phase Retrieval

(Optical Wavefront Sensing and Control)

Dr. Bruce H. Dean
NASA GSFC
Code 551

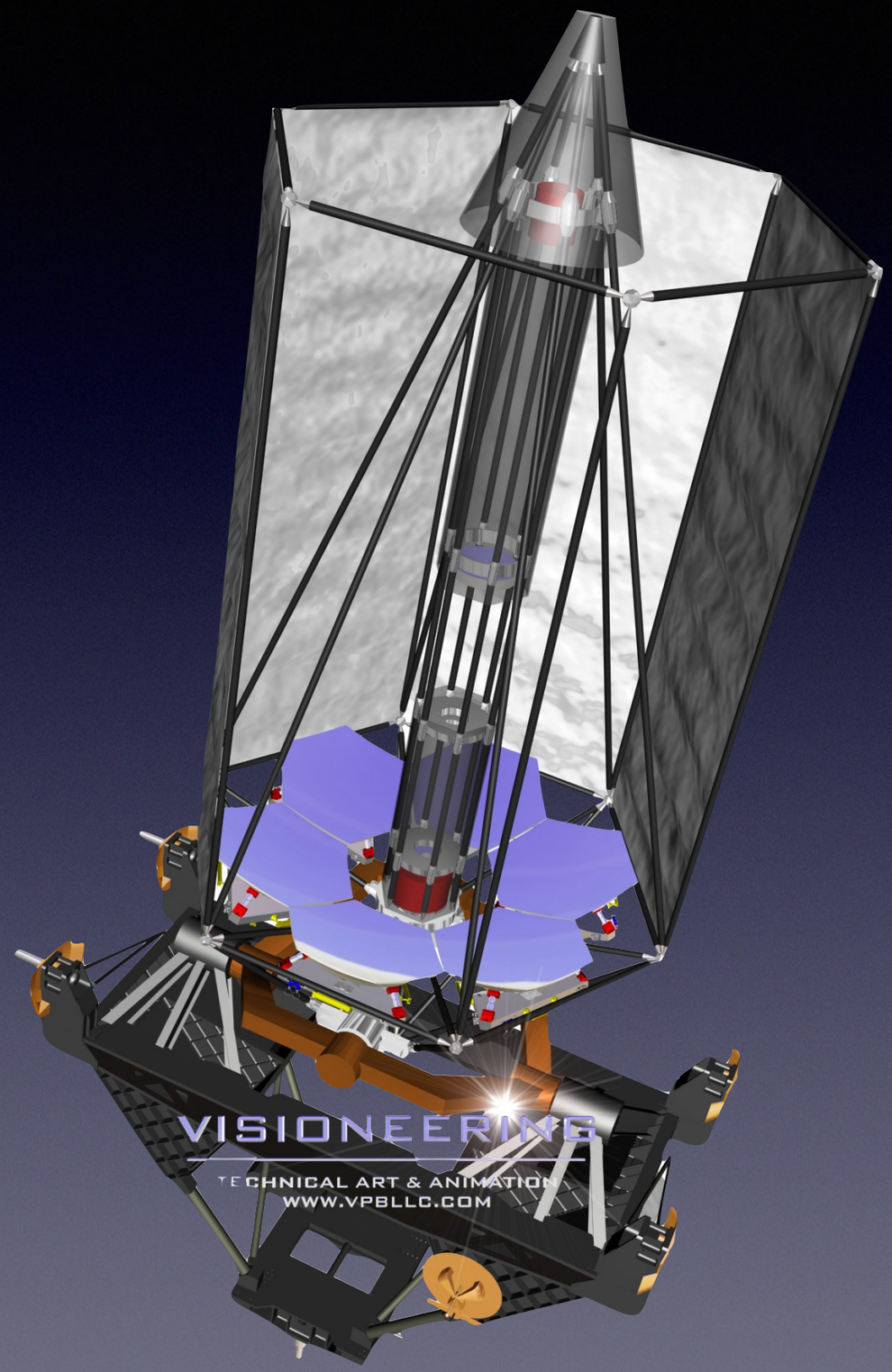
Application: Optical Systems

Adam Hinett

Application: Optical Systems

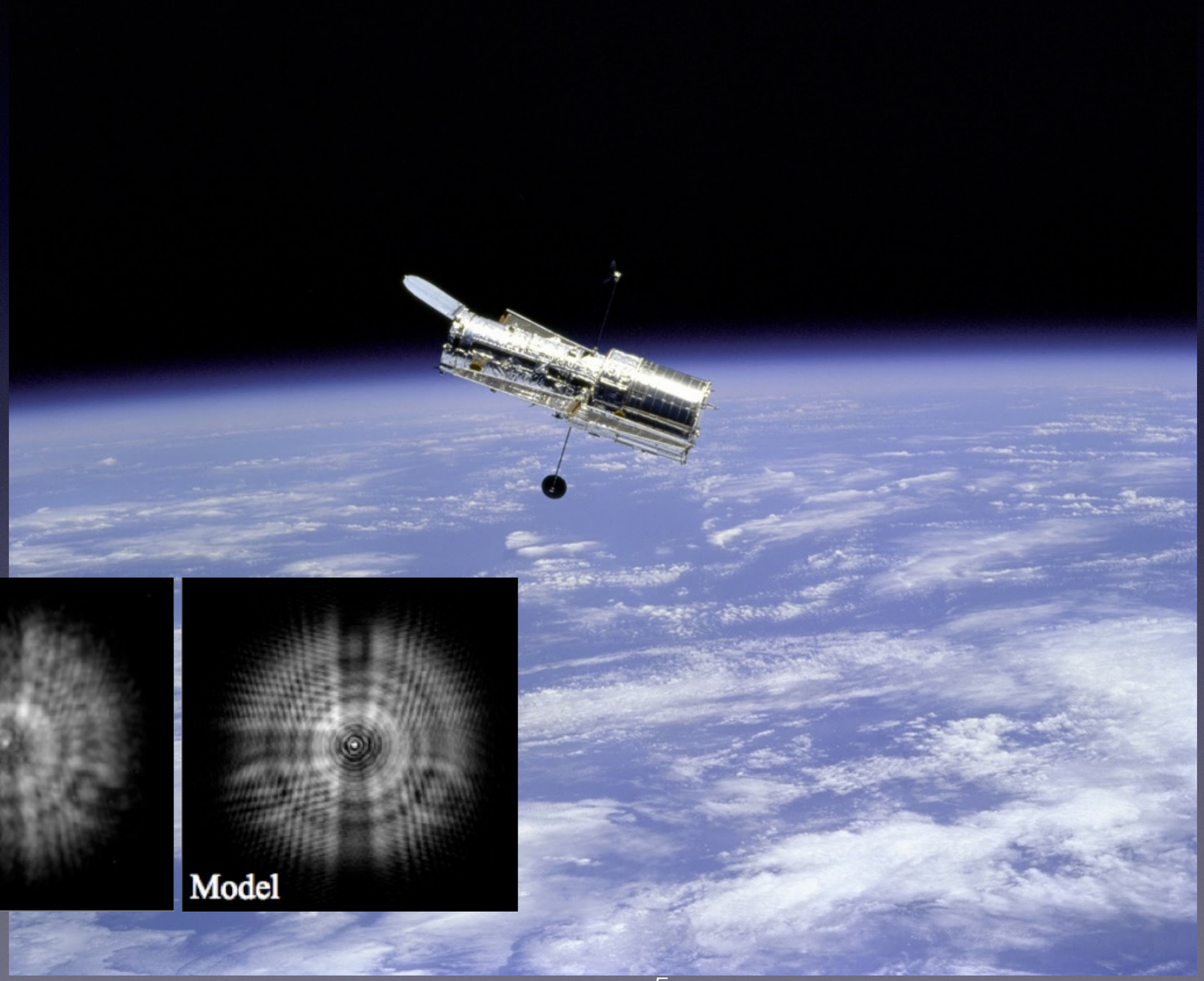
Adam Hinett

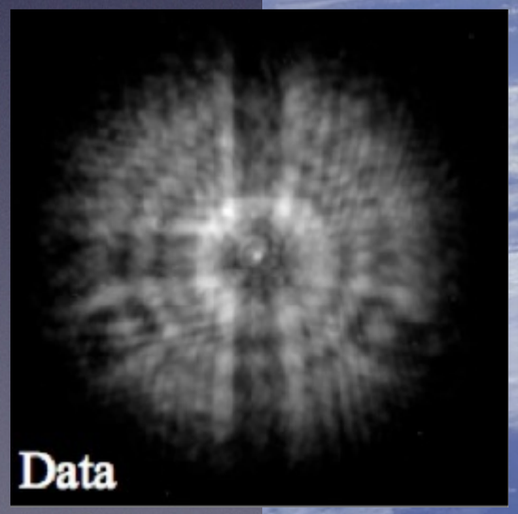
Space Telescopes



Phase Retrieval

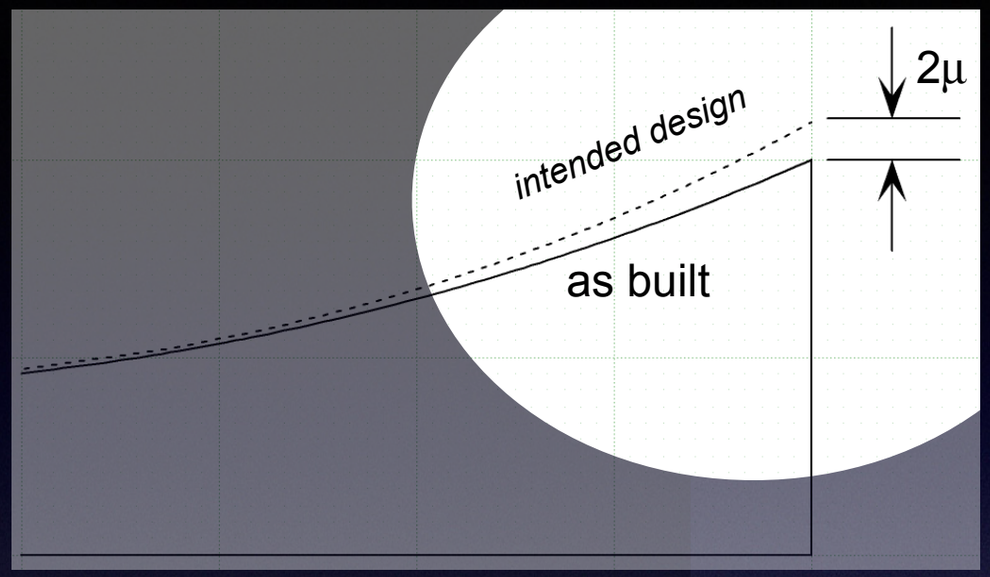
- COSTAR (corrective optics)



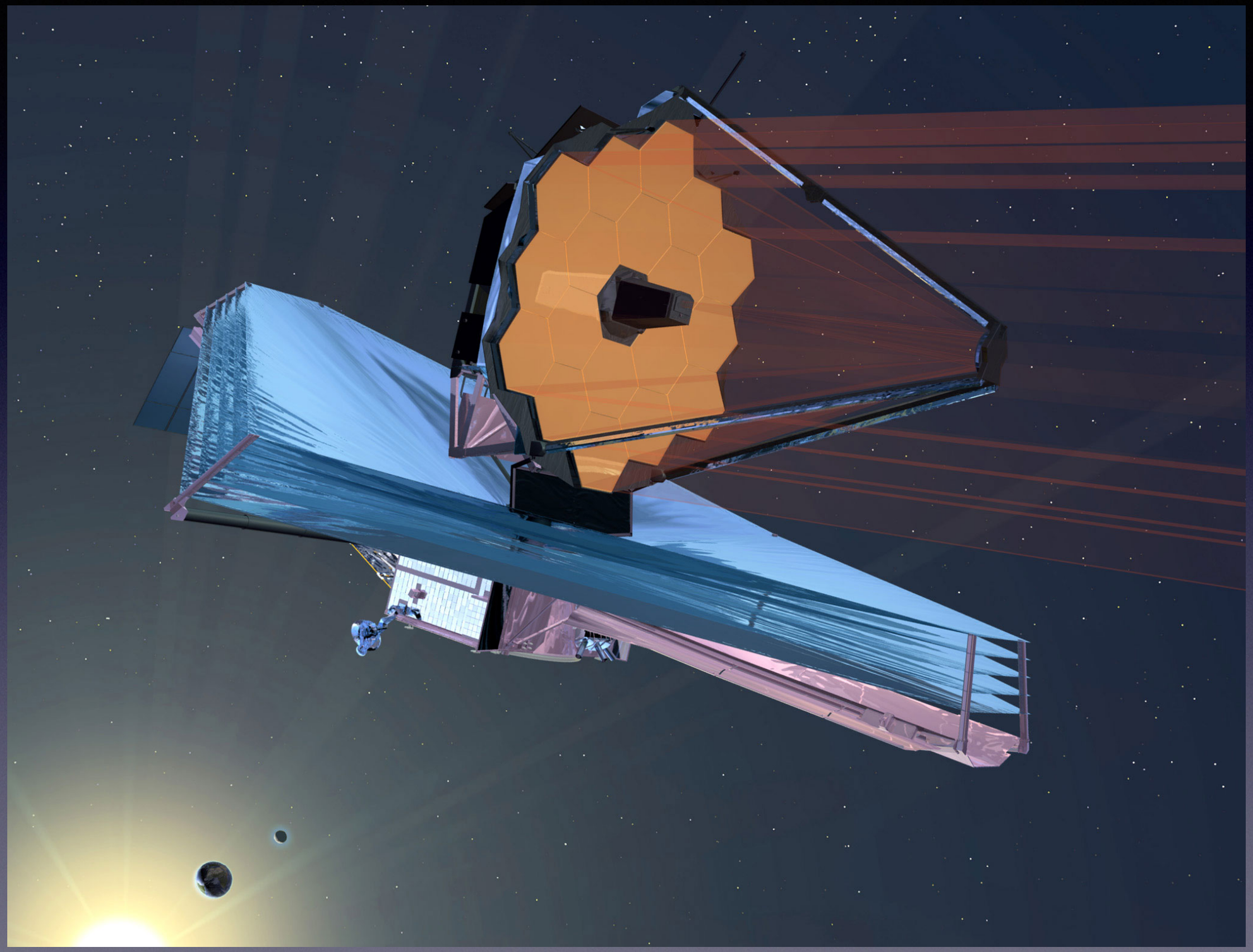


Phase Retrieval

- COSTAR (corrective optics)

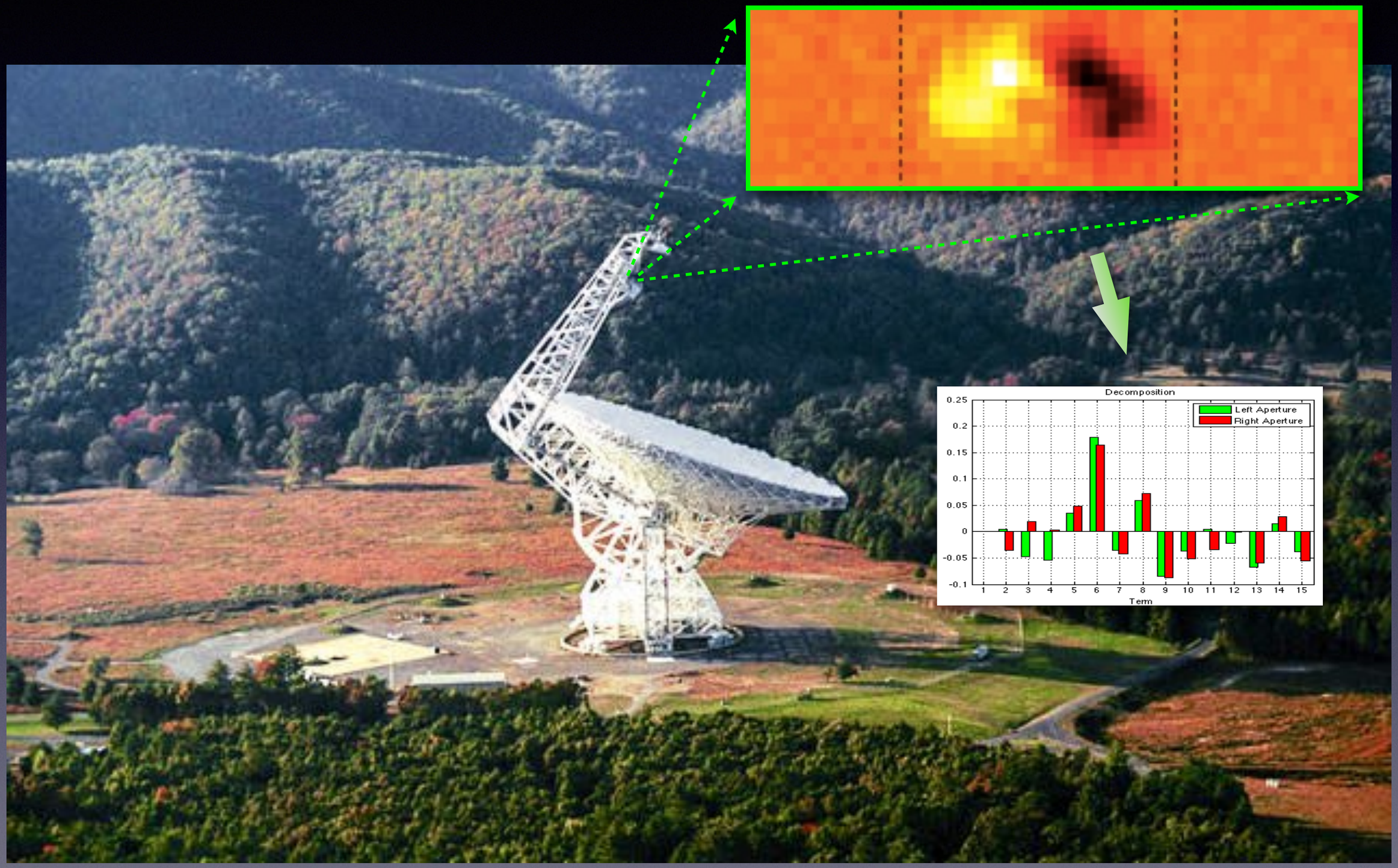


JWST



Cool example

Cool example

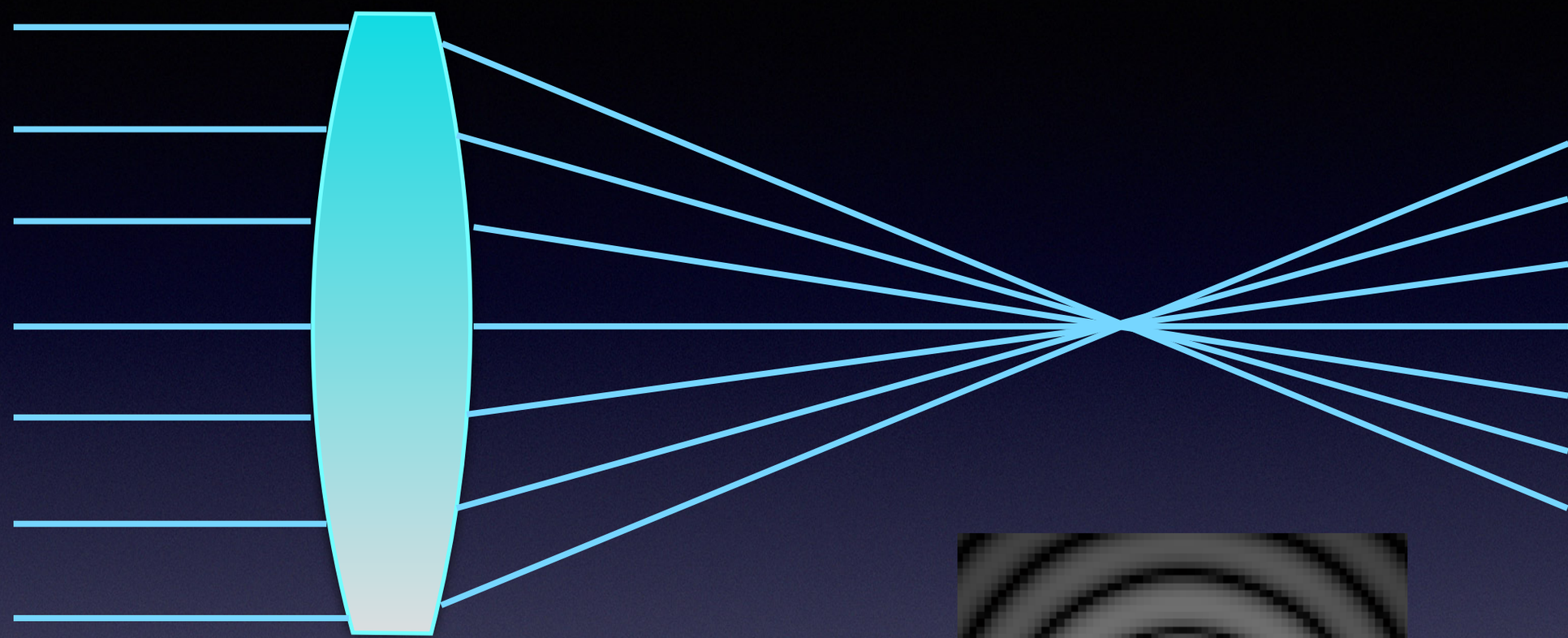


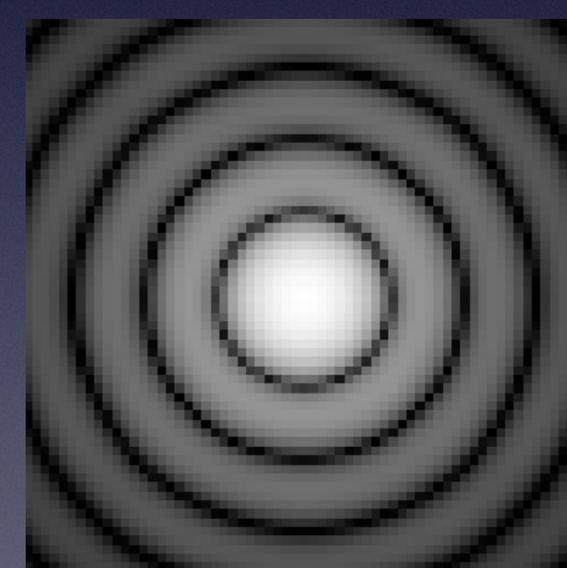
How can Machine Learning help?

Recall

$$\underbrace{y}_{\textit{labels}} = f(\underbrace{X}_{\textit{features}})$$

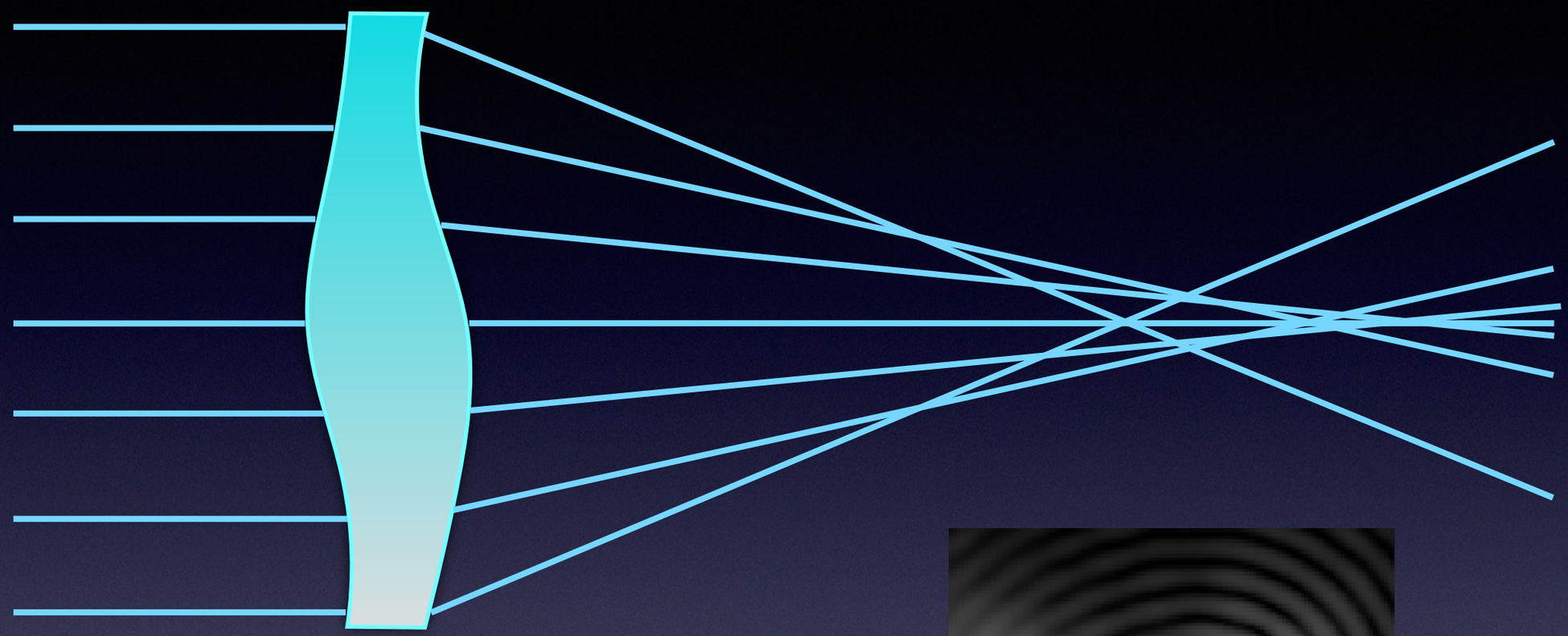
Visualization

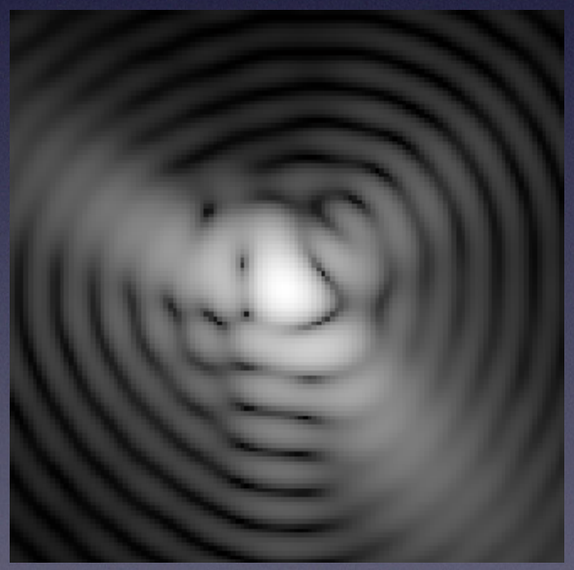




perfect image

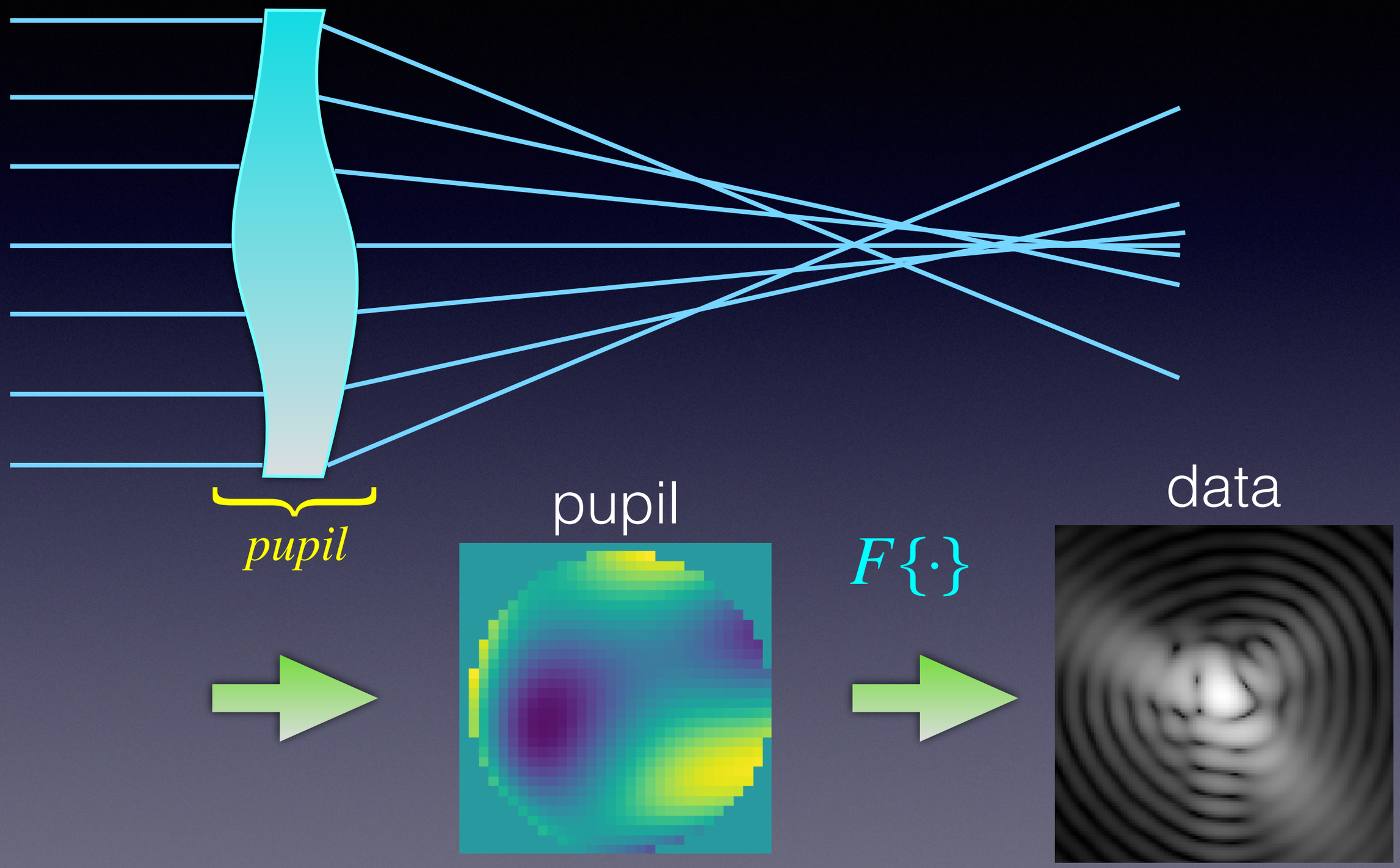
Annoying Imperfections



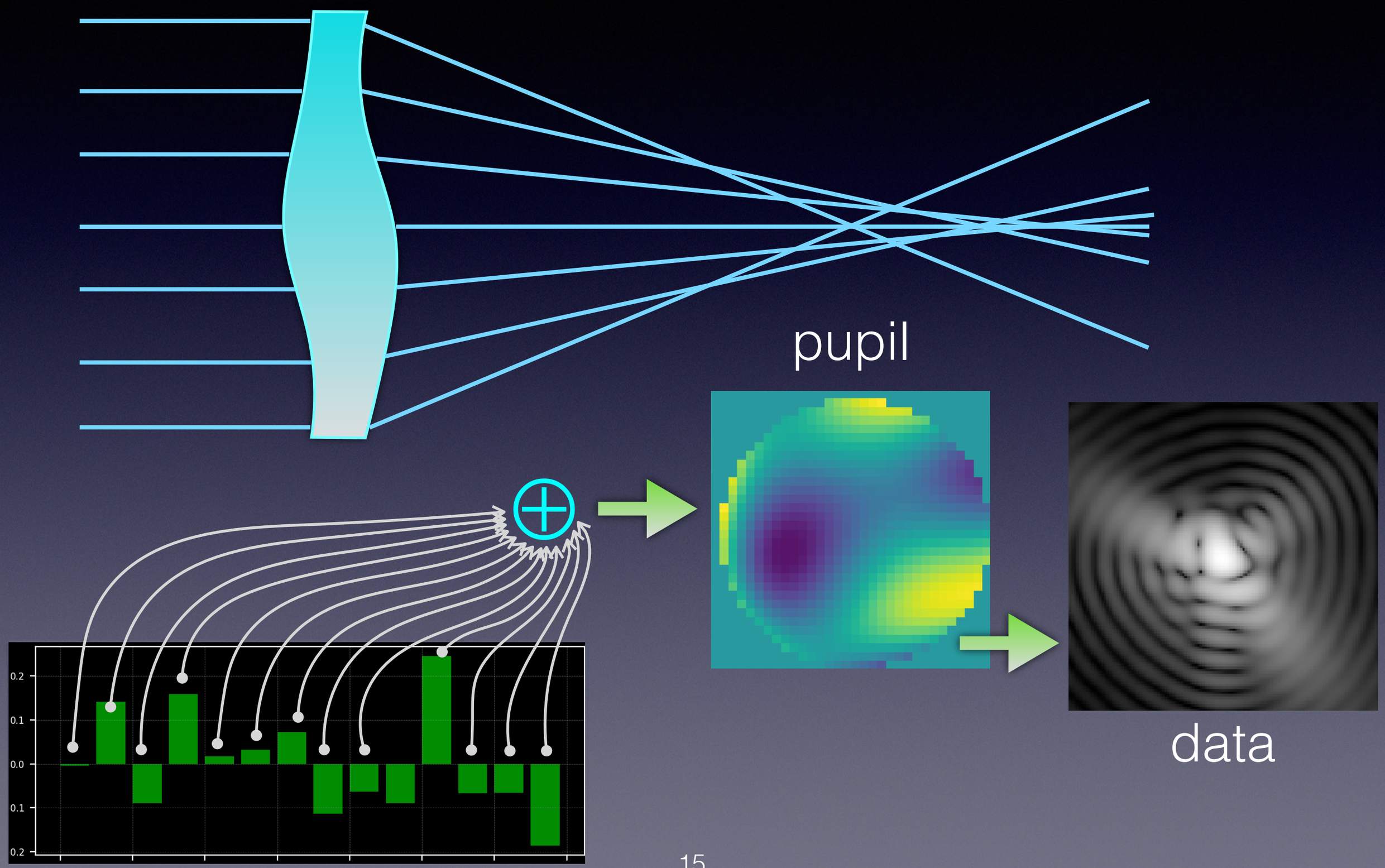


aberrated image

How to Quantify?



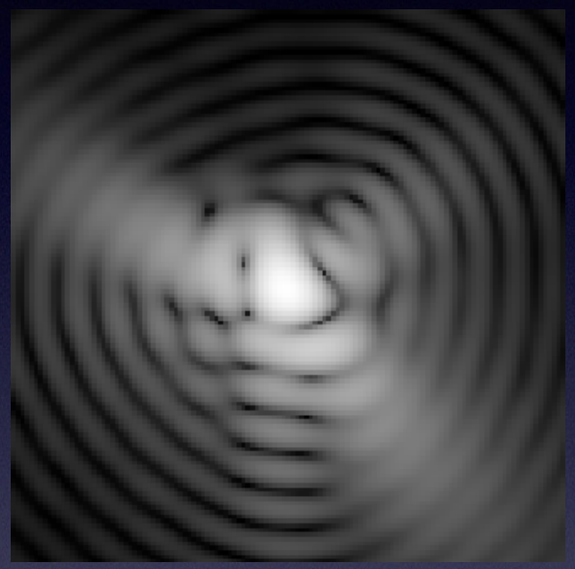
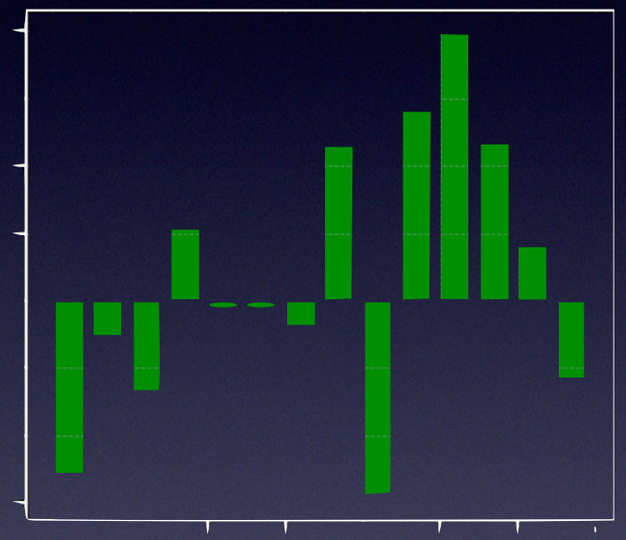
How to Quantify?



Create Mapping

X

supervised: y

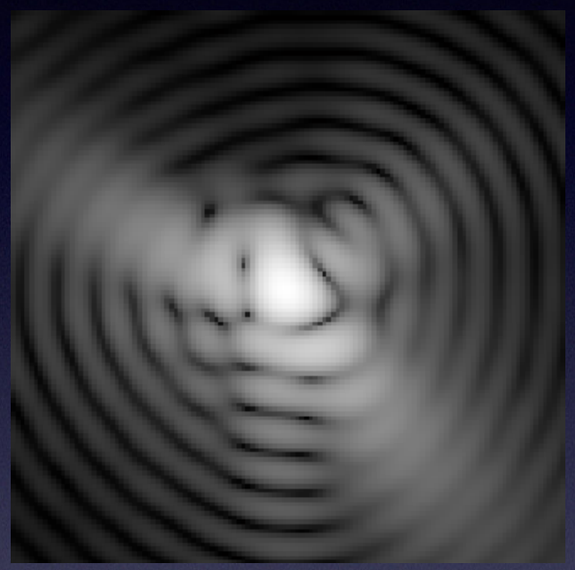
$f($  $)$ $\rightarrow$ 

features

Labels

Create Mapping

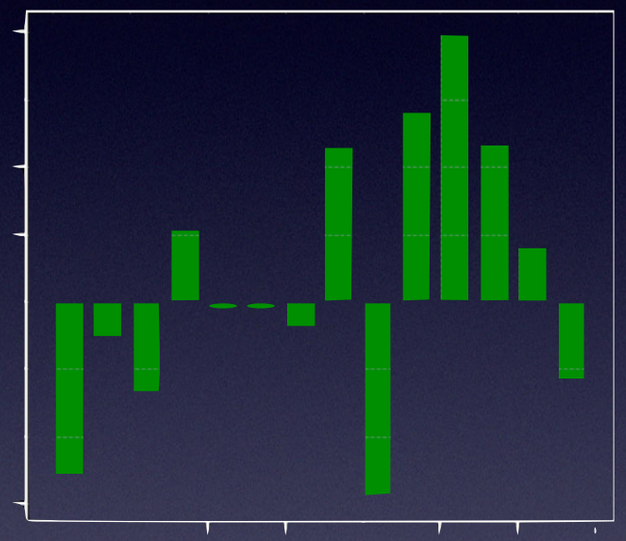
X

$f($  $)$

features

$\longrightarrow$

Y

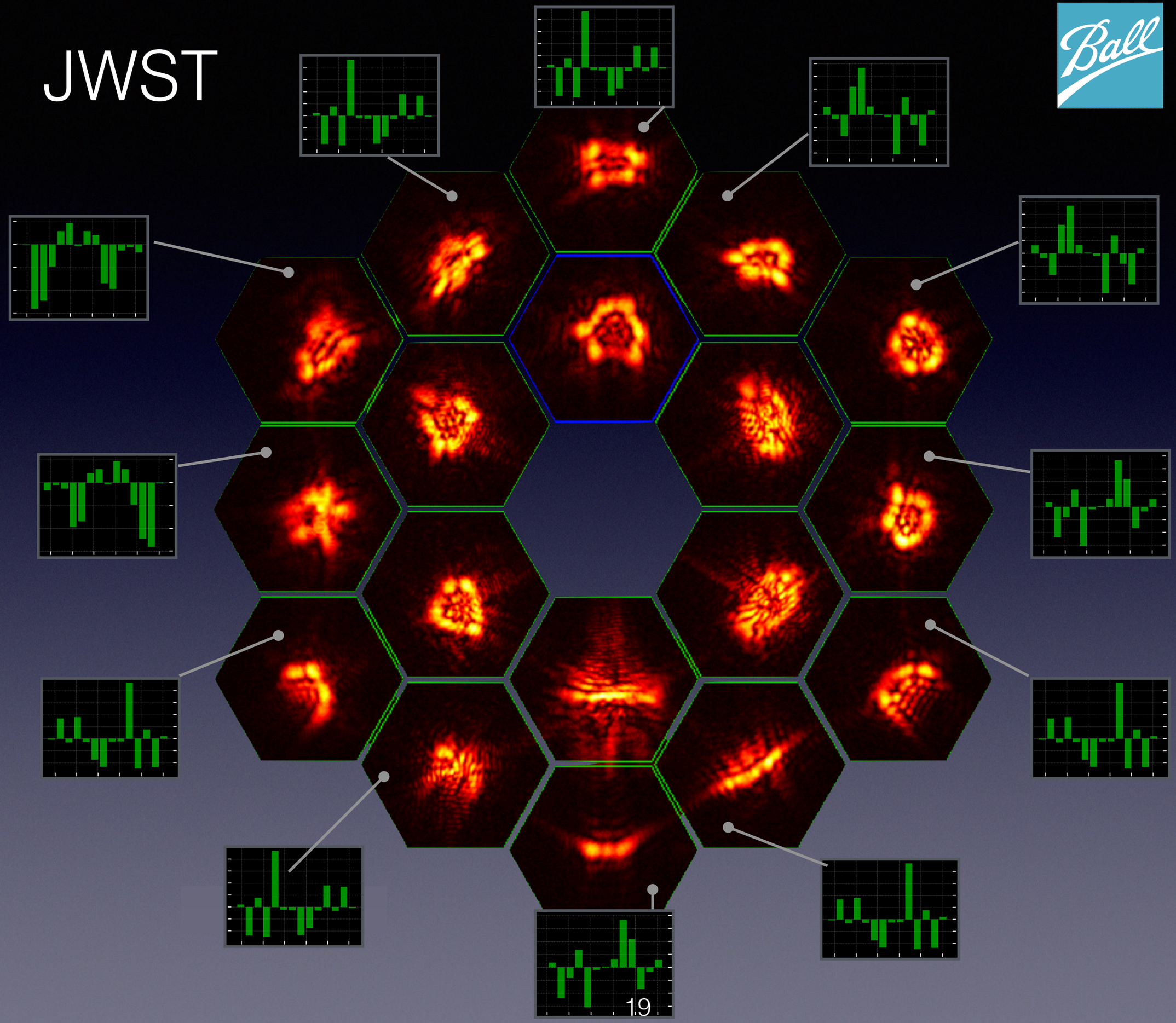


Labels

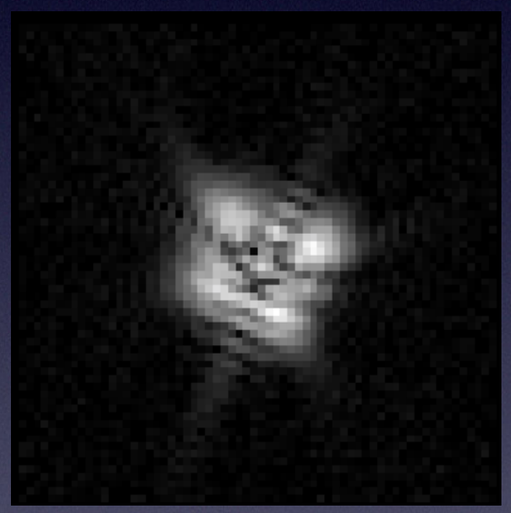
mapping from one set to another

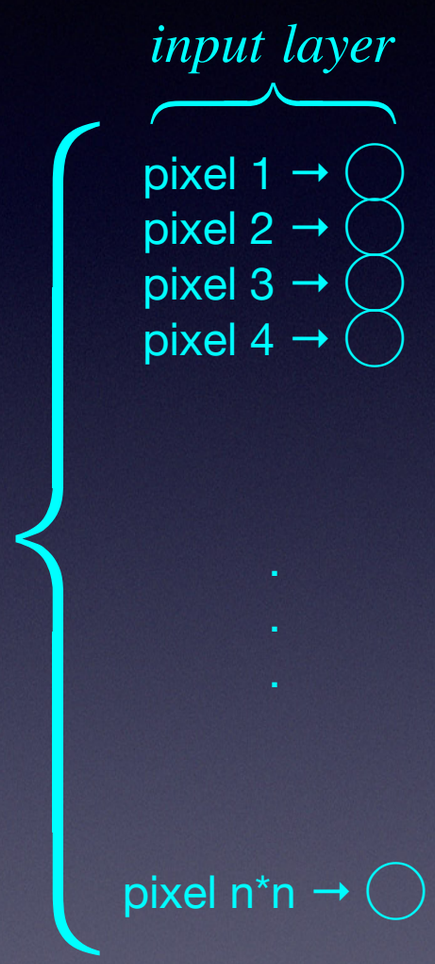
JWST

JWST

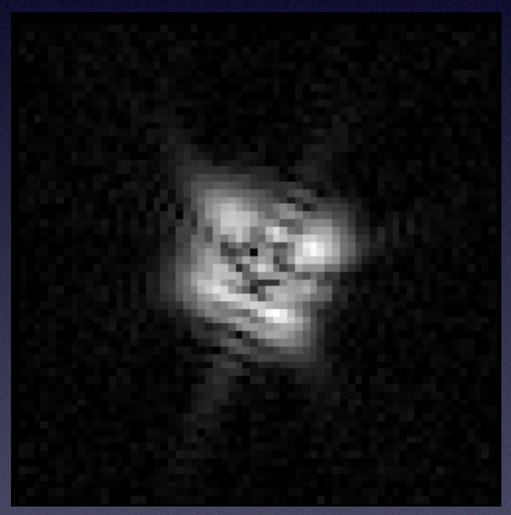


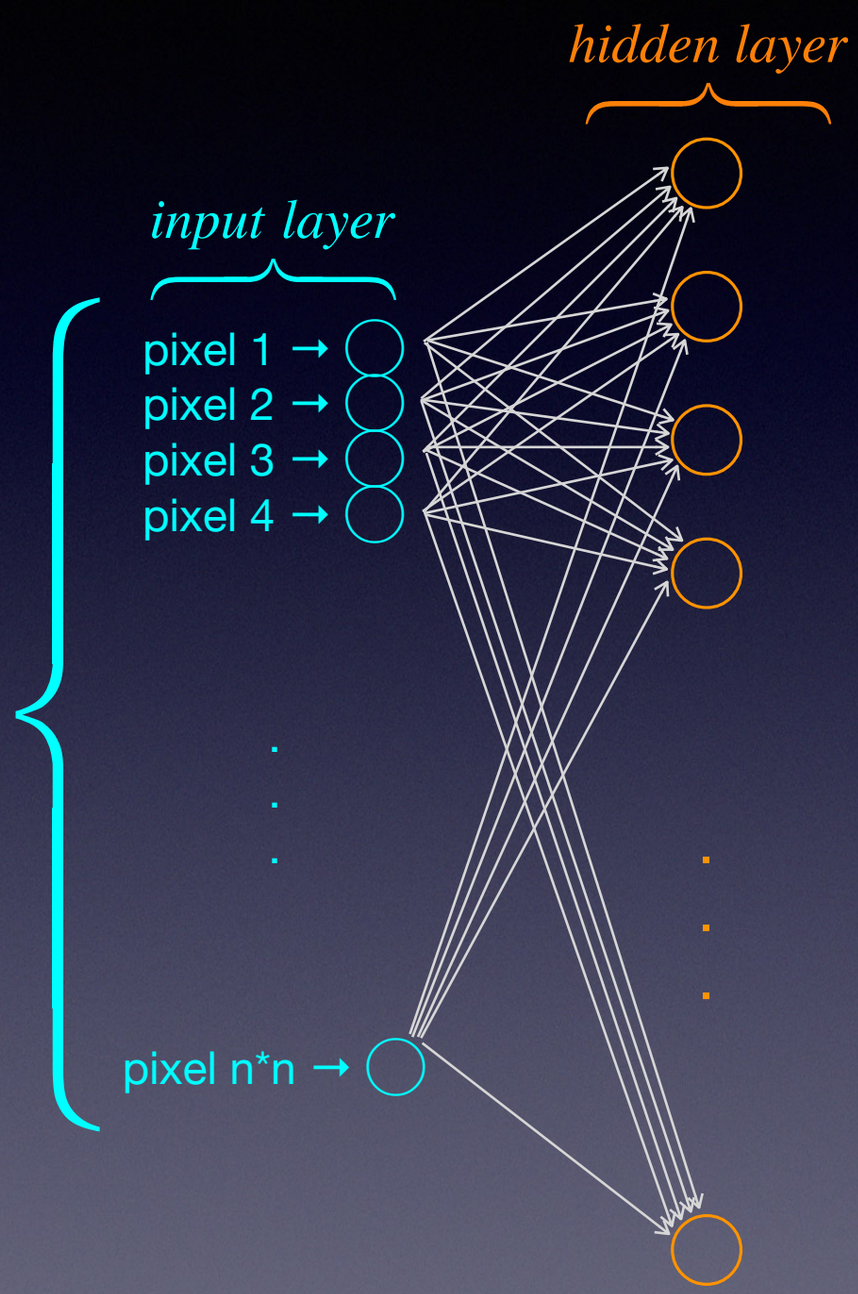
“Naive” Architecture



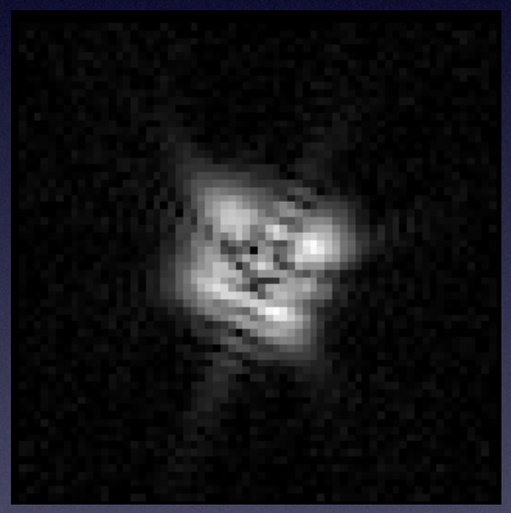


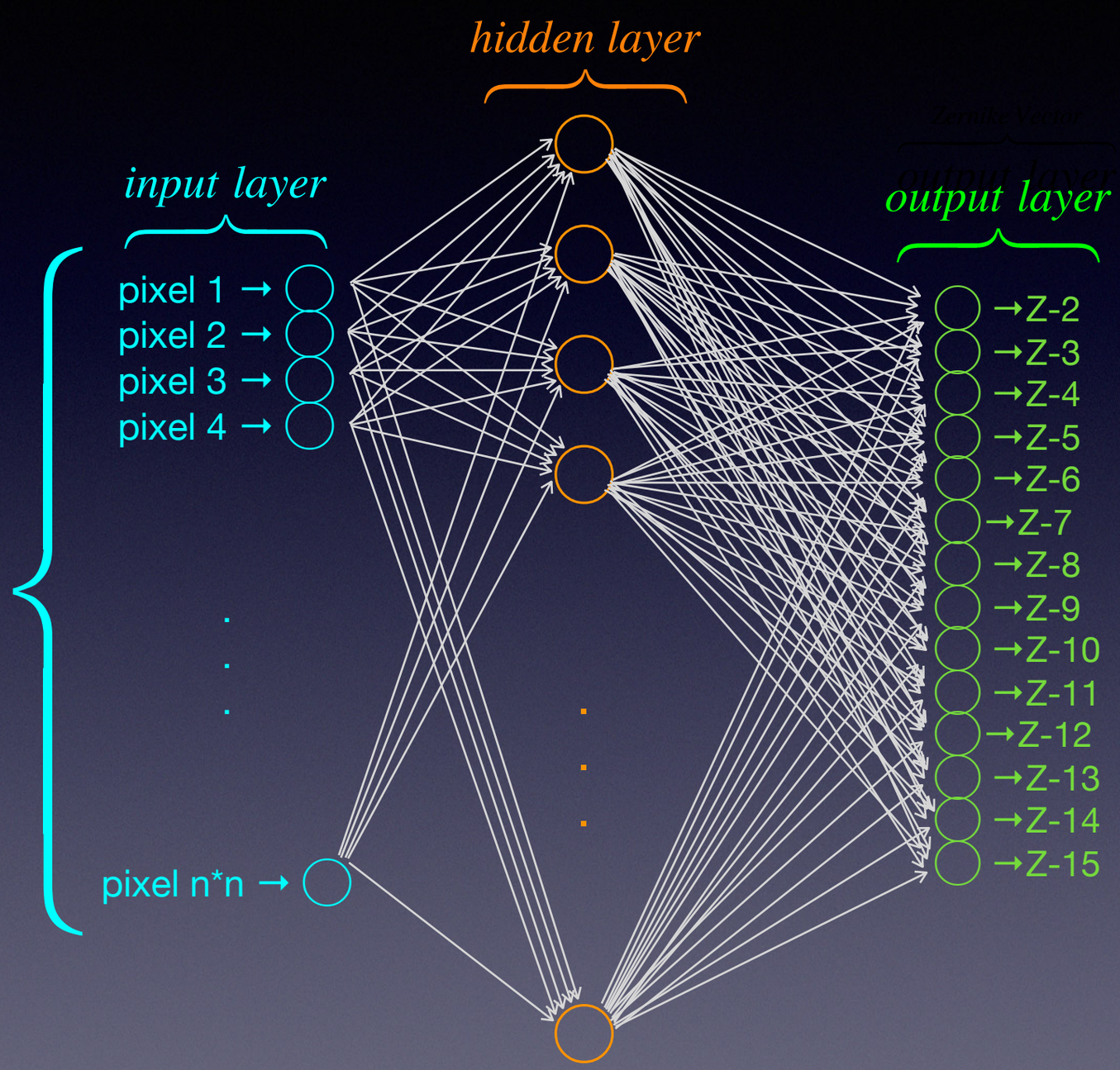
“Naive” Architecture





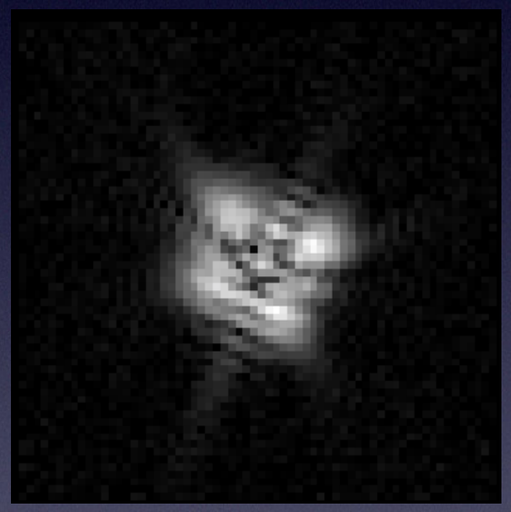
“Naive” Architecture

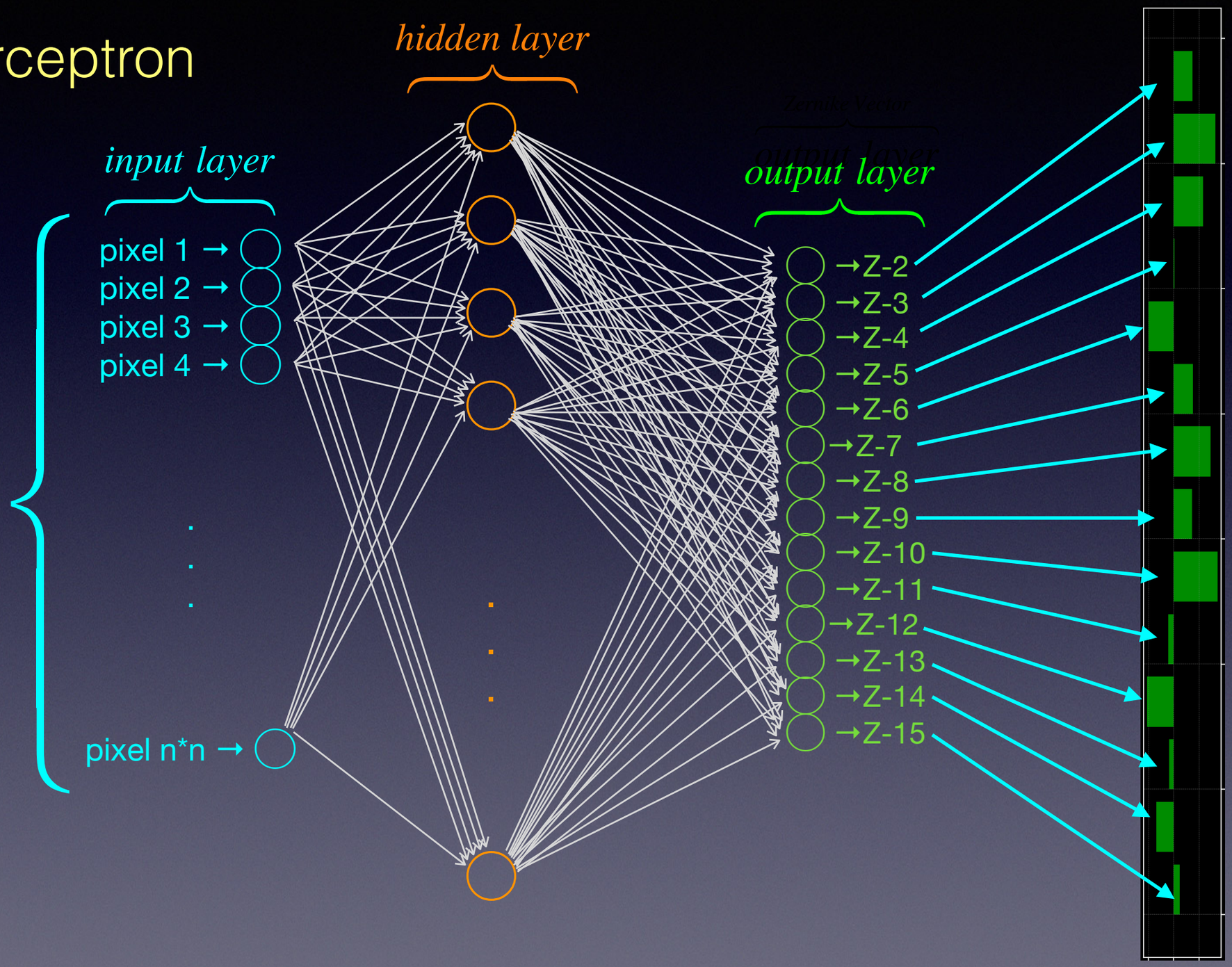




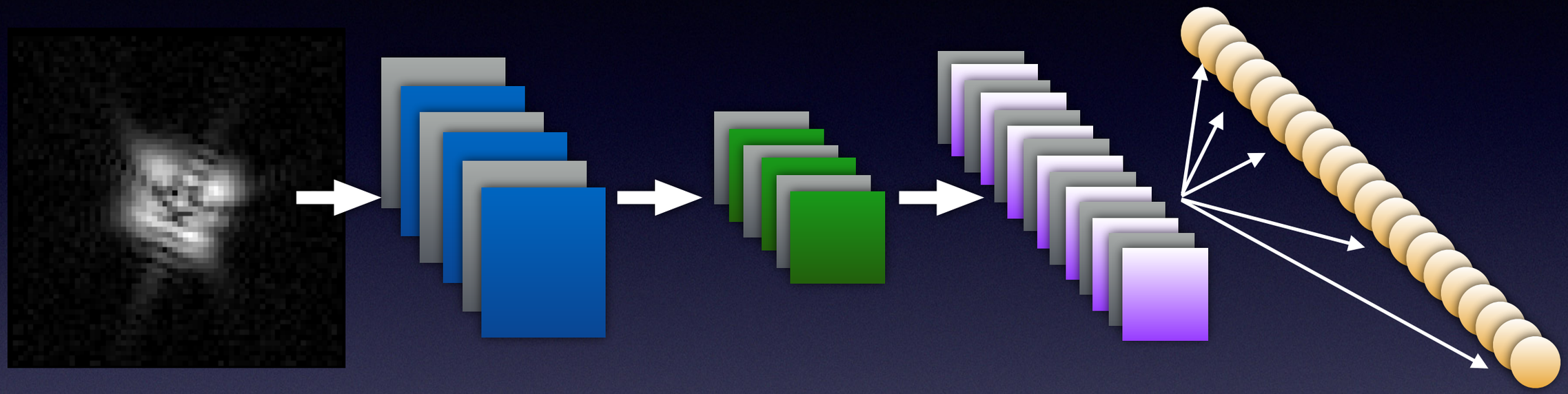
“Naive” Architecture

Multi-Layer-Perceptron

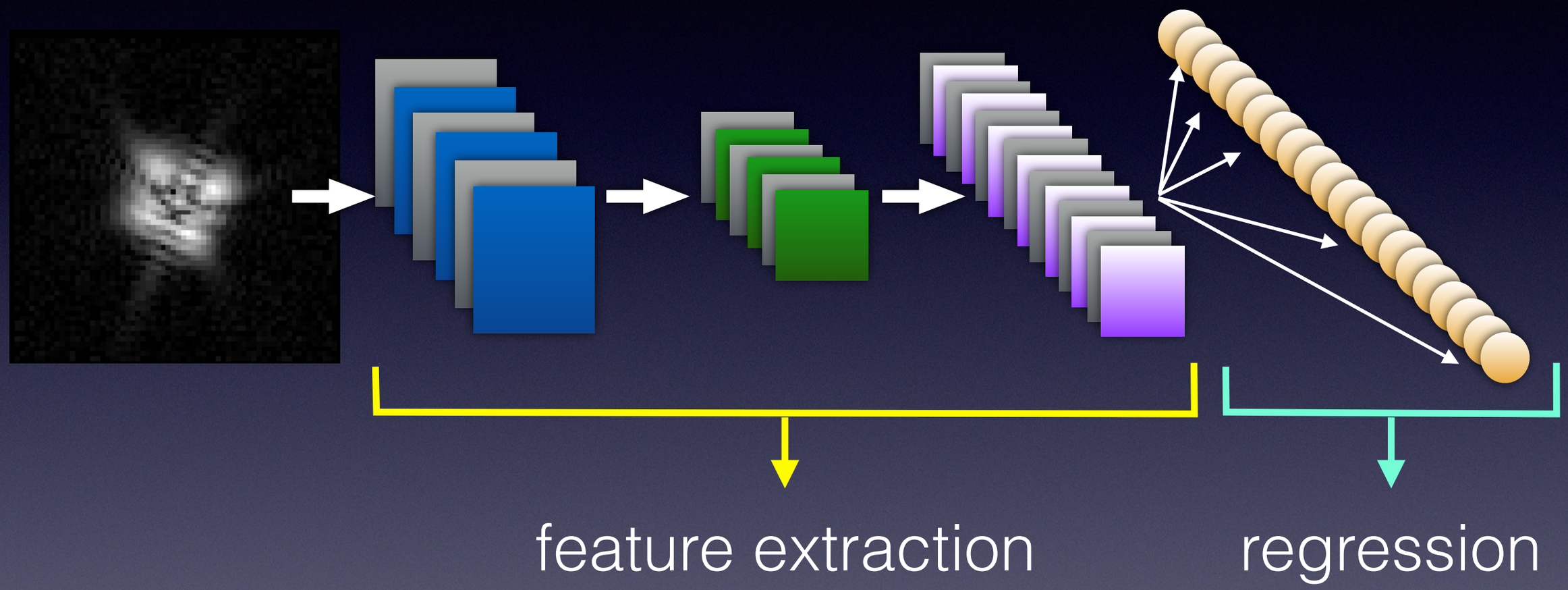




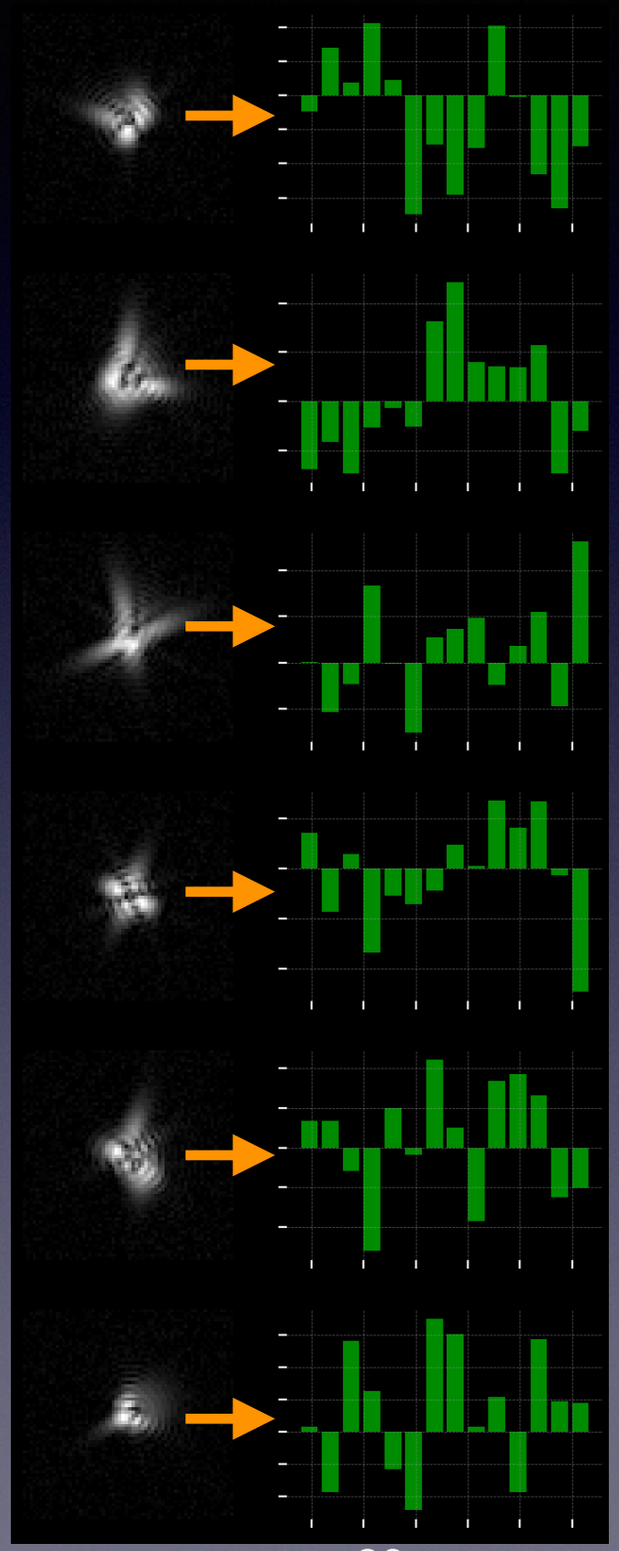
Convolutional Neural Net



Convolutional Neural Net

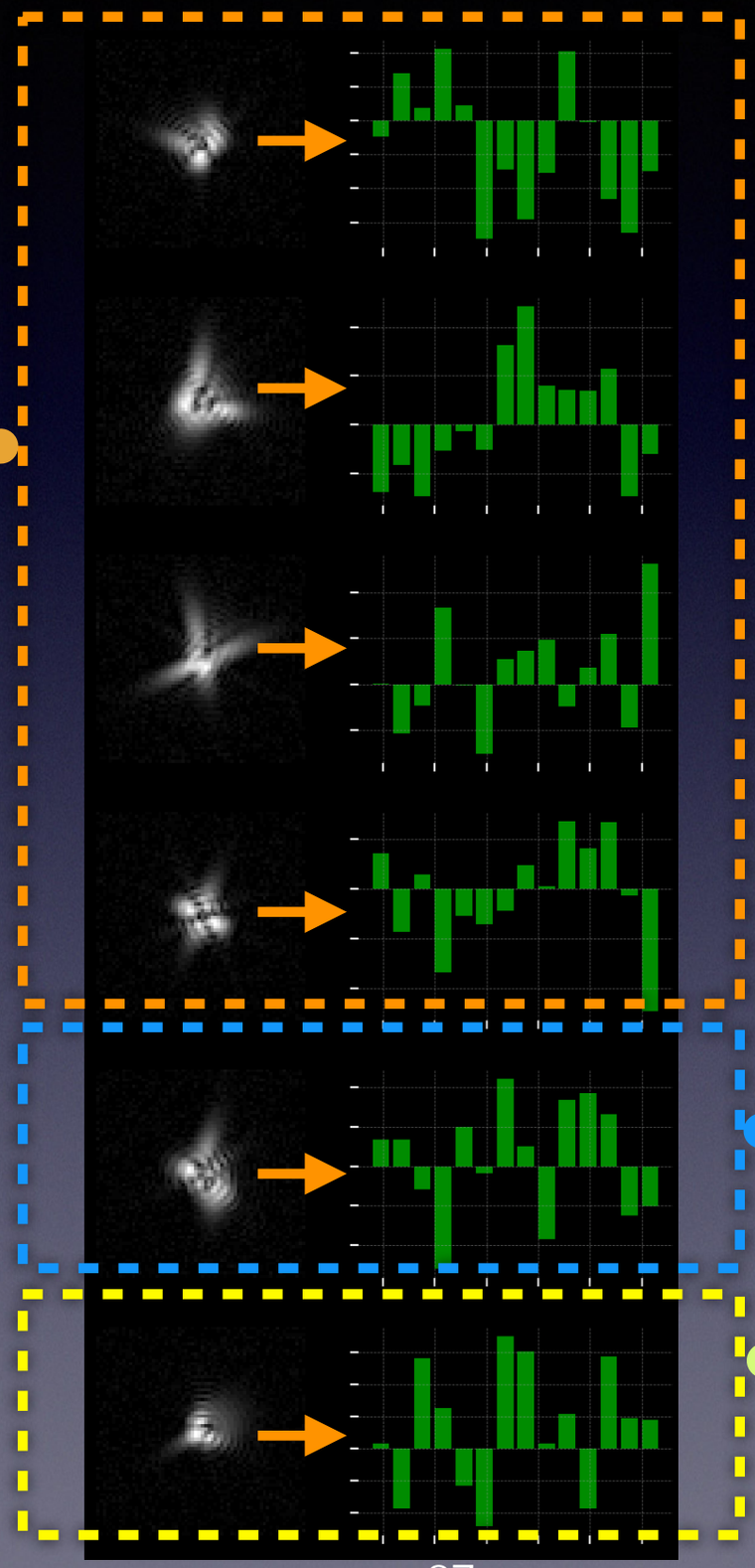


Space Telescope Training

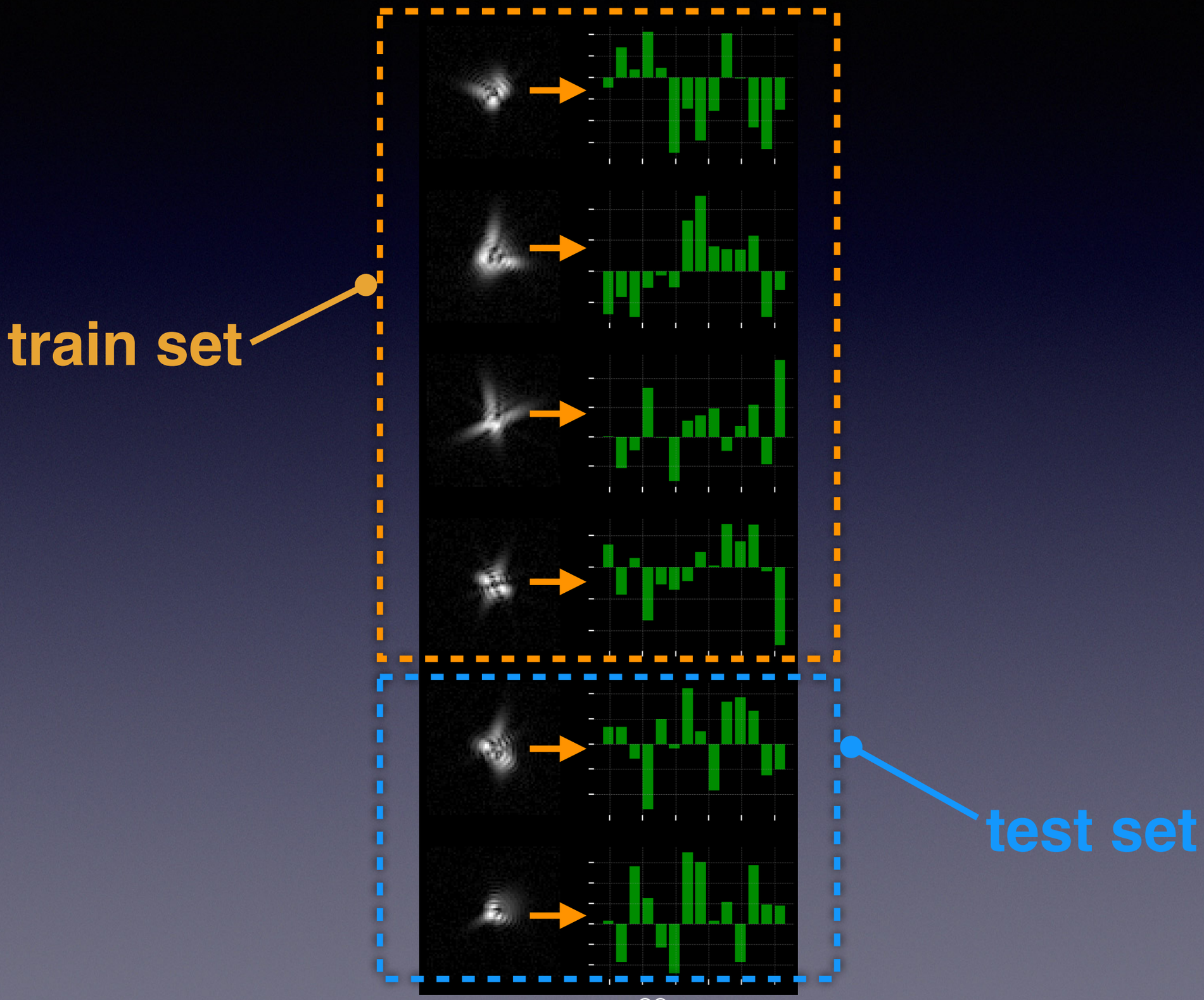


1000's of supervised learning examples

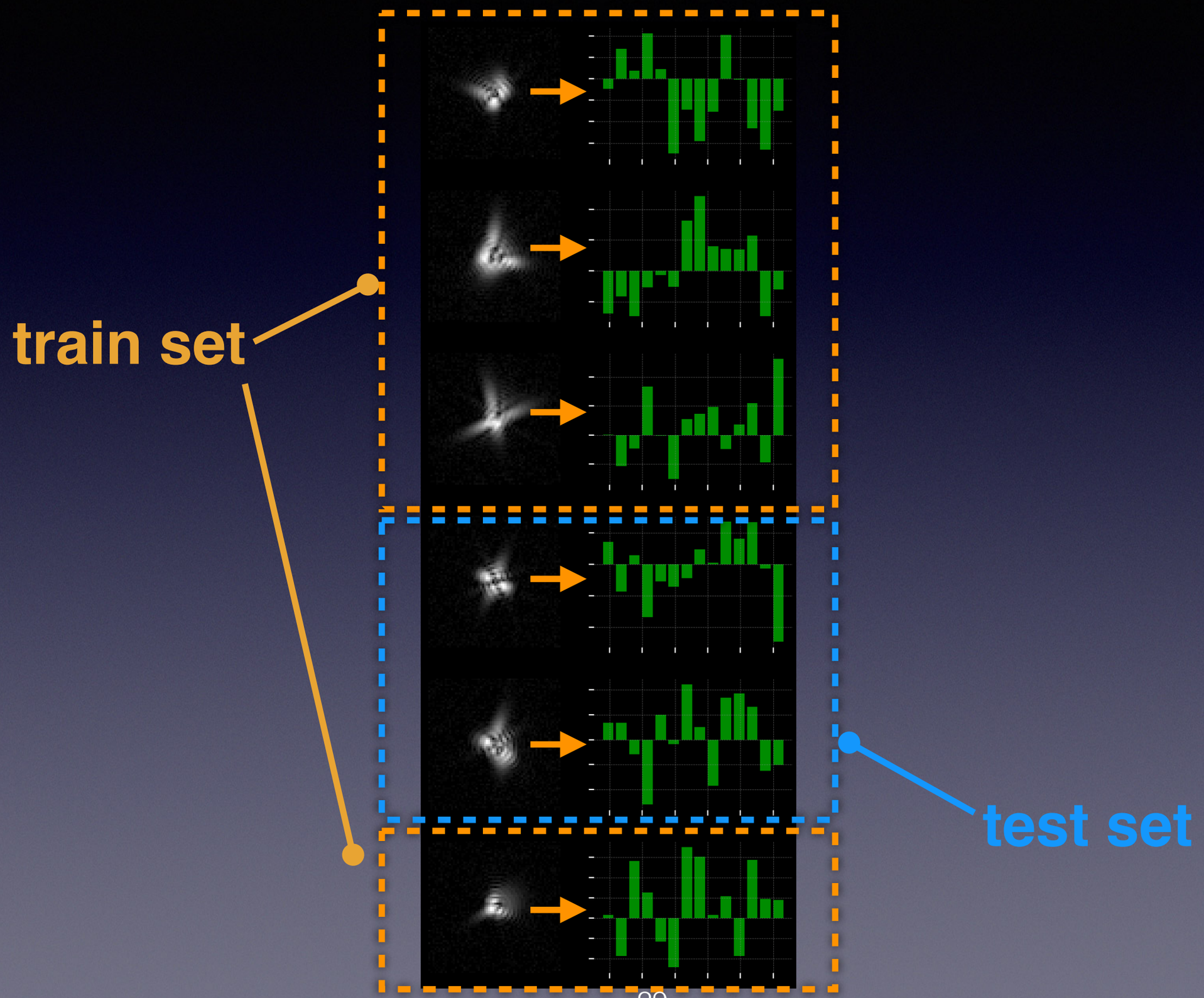
Space Telescope Training



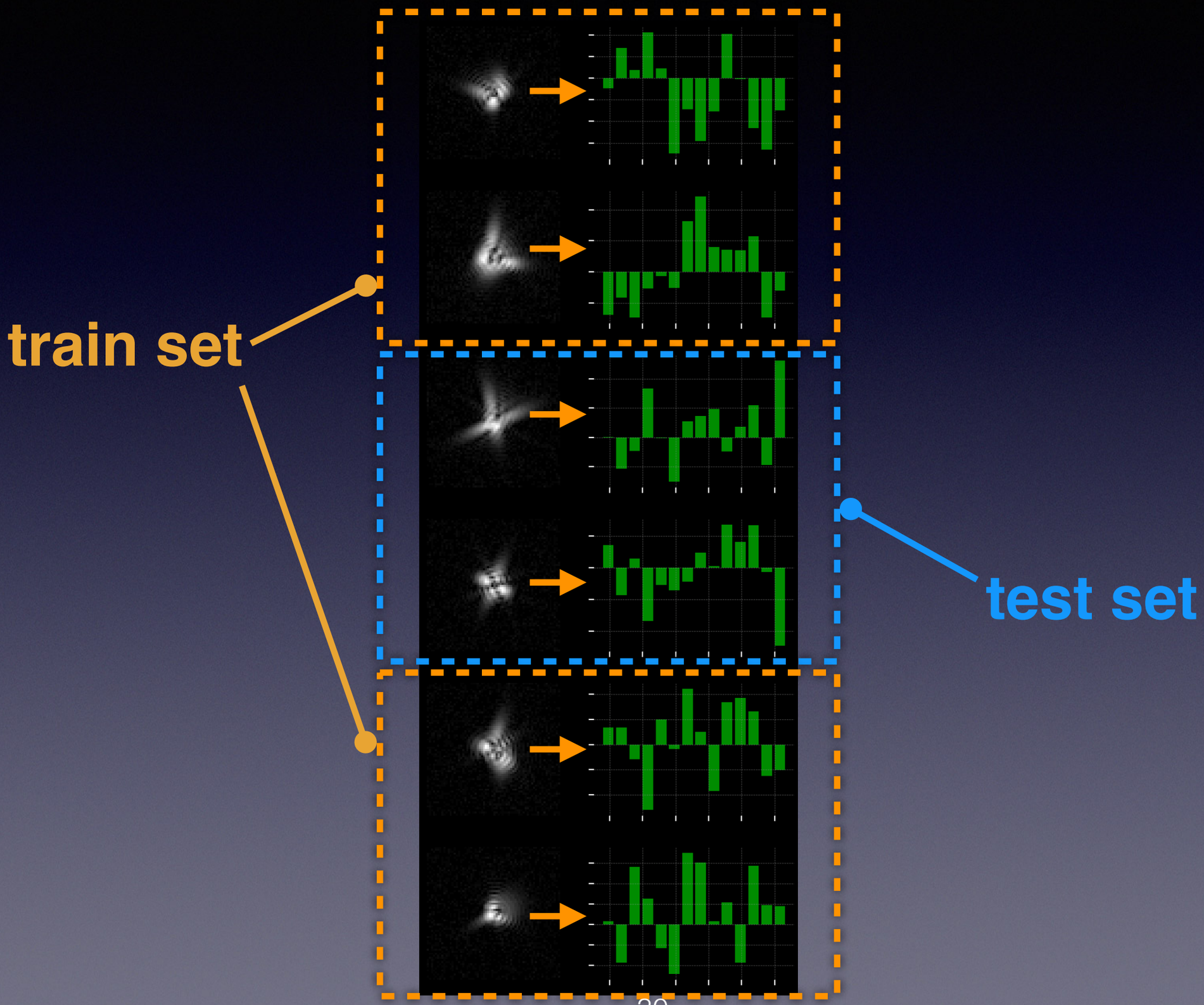
Gold Standard: k-fold cross validation



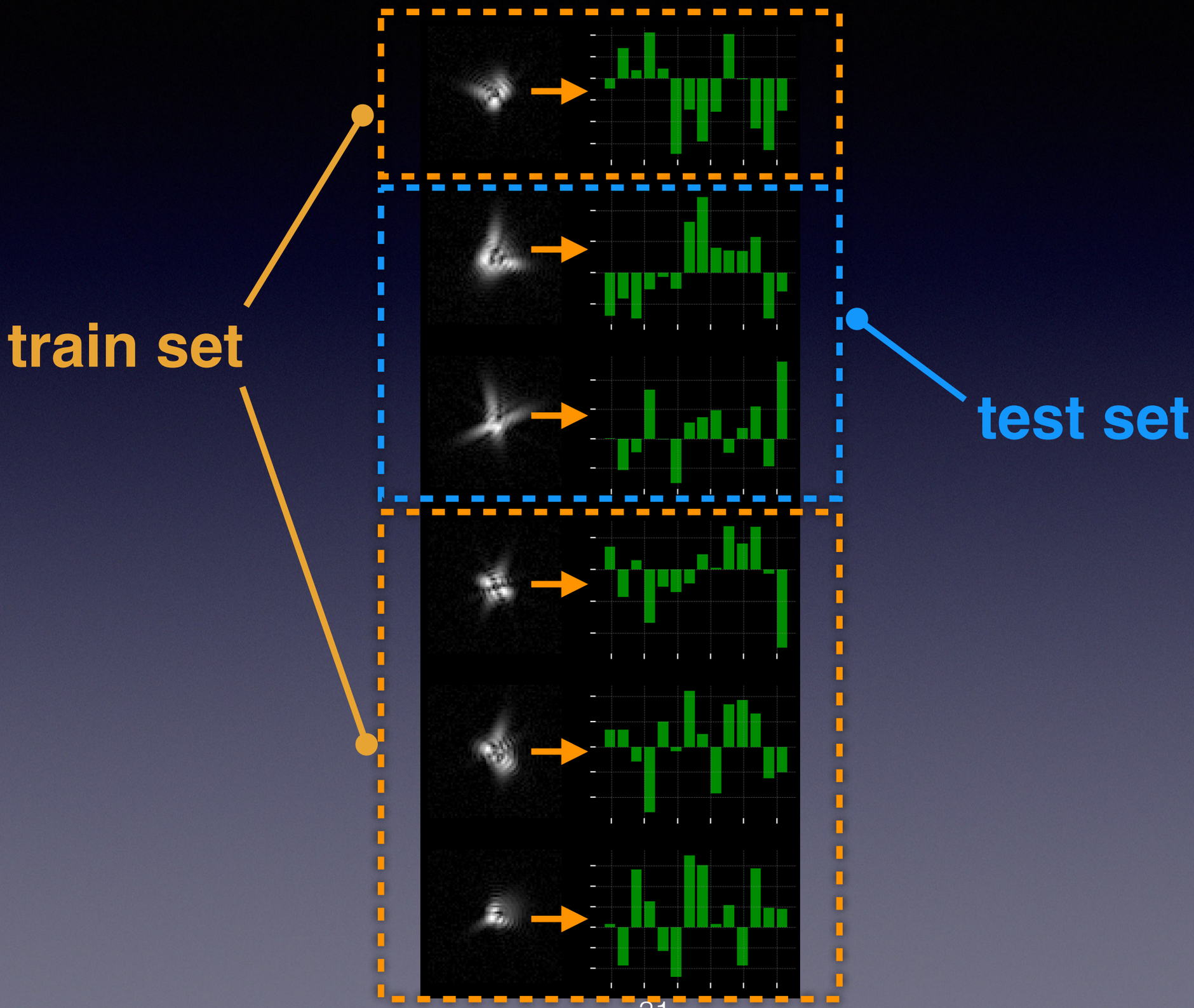
Gold Standard: k-fold cross validation



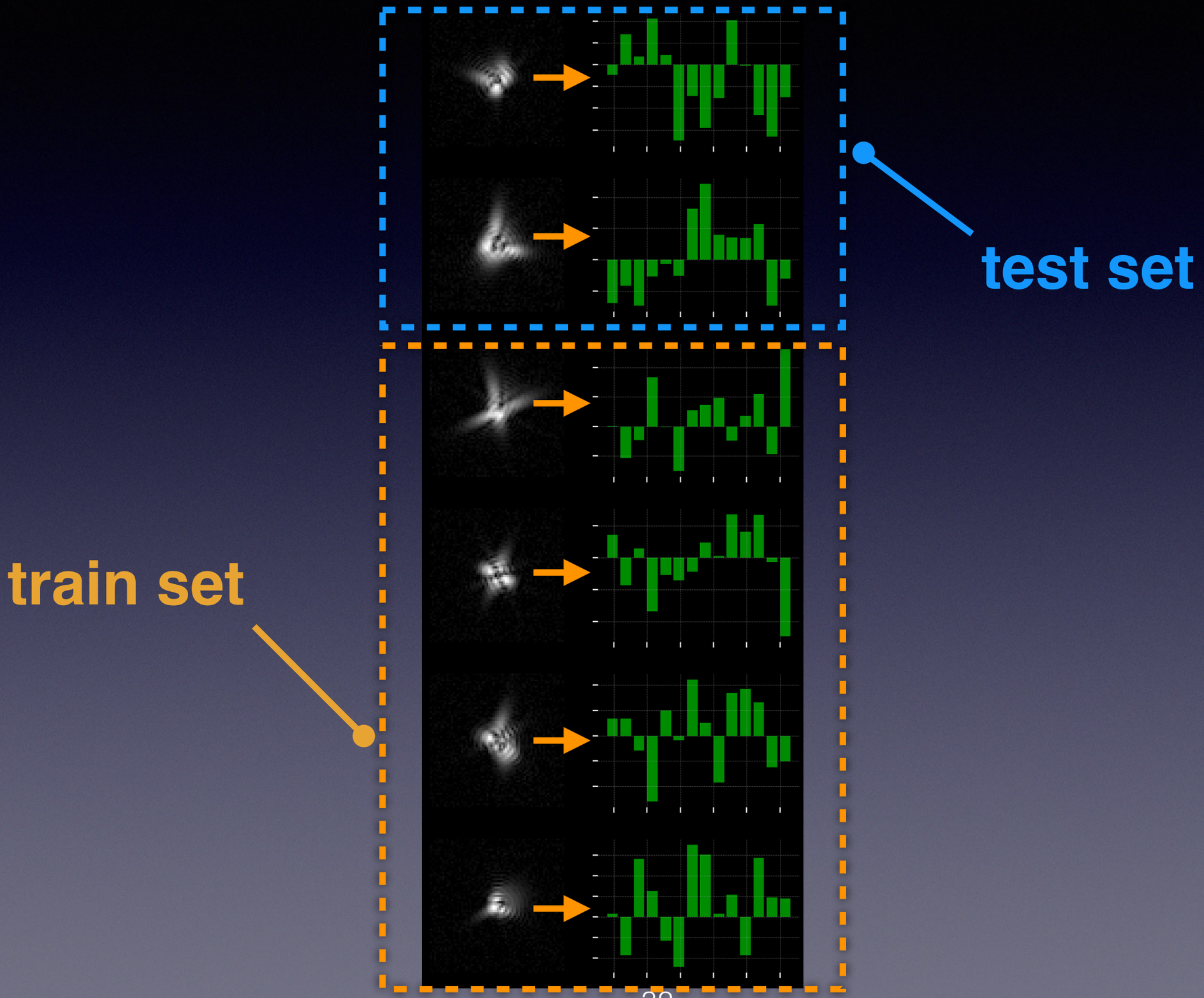
Gold Standard: k-fold cross validation



Gold Standard: k-fold cross validation

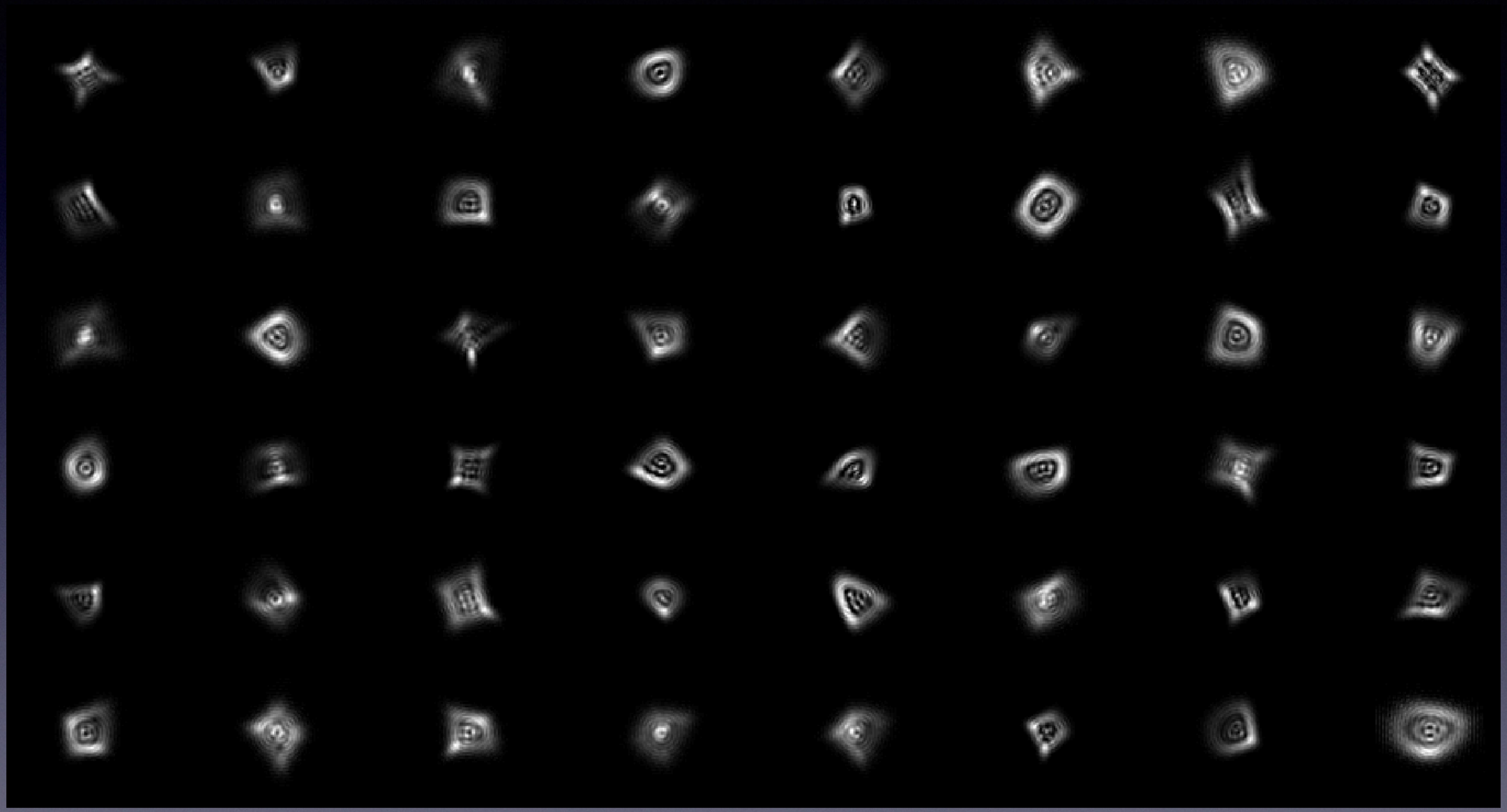


Gold Standard: k-fold cross validation

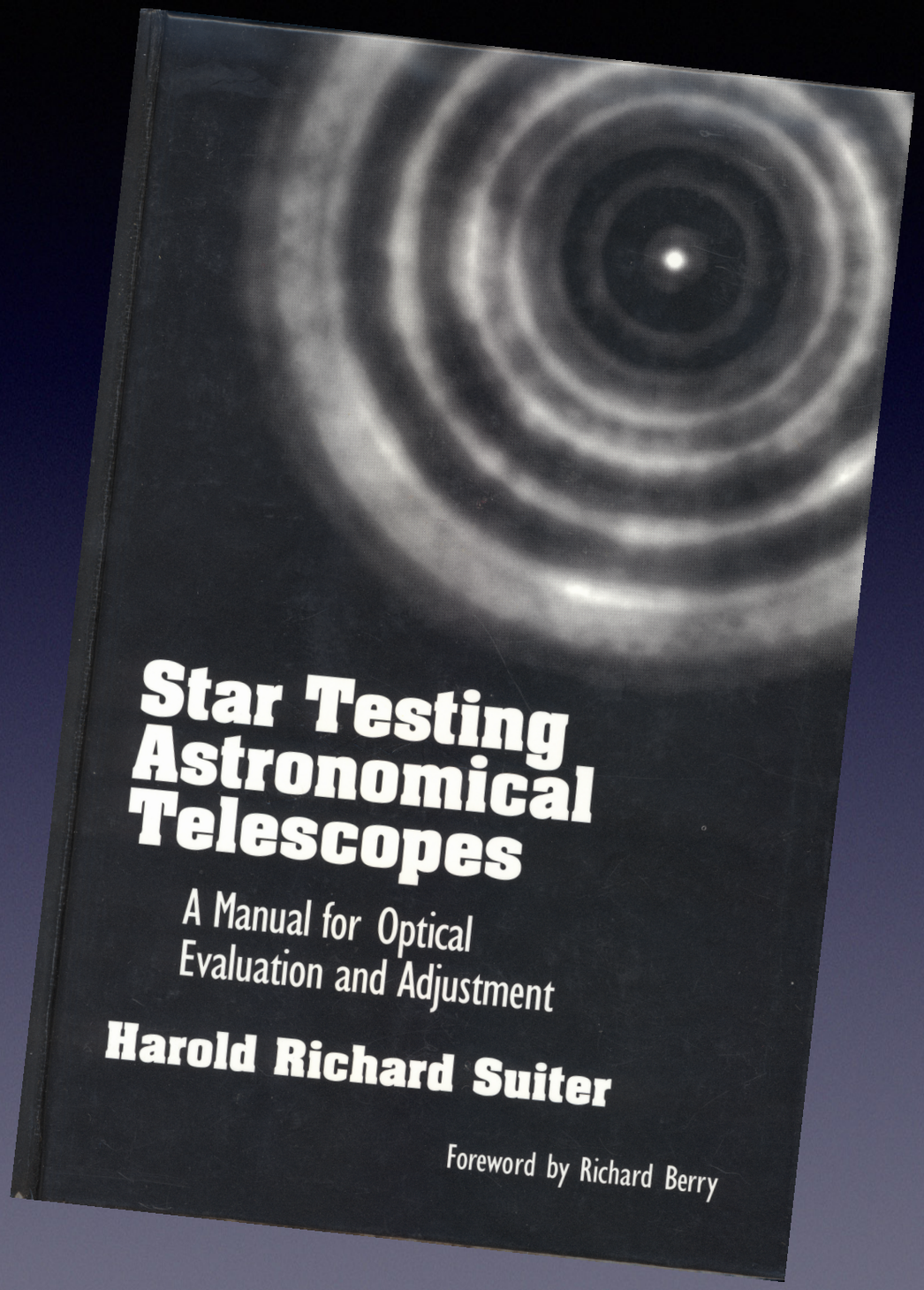


Pattern Recognition

ability to generalize

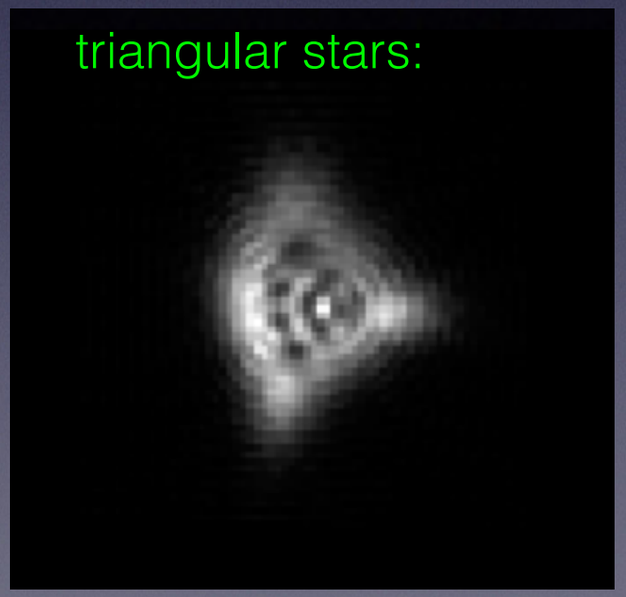


Star Testing



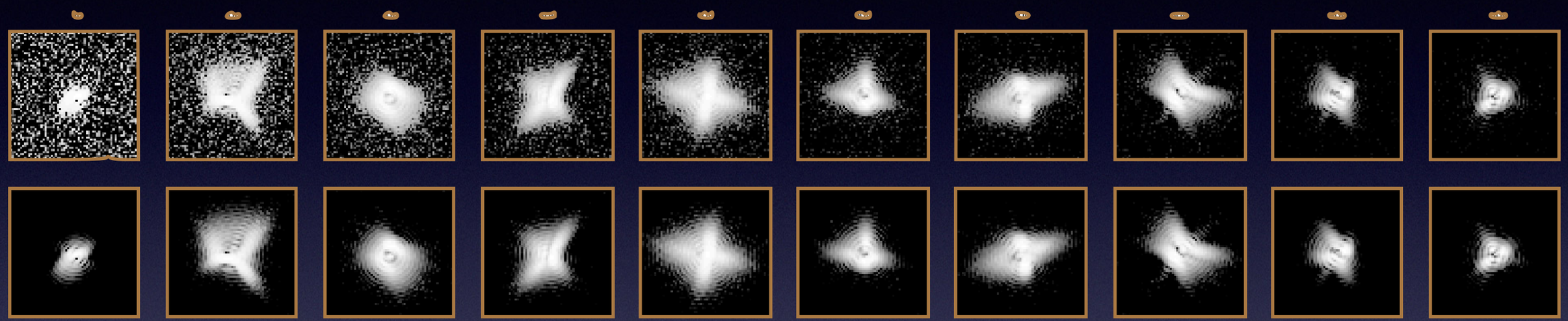
based on pattern recognition

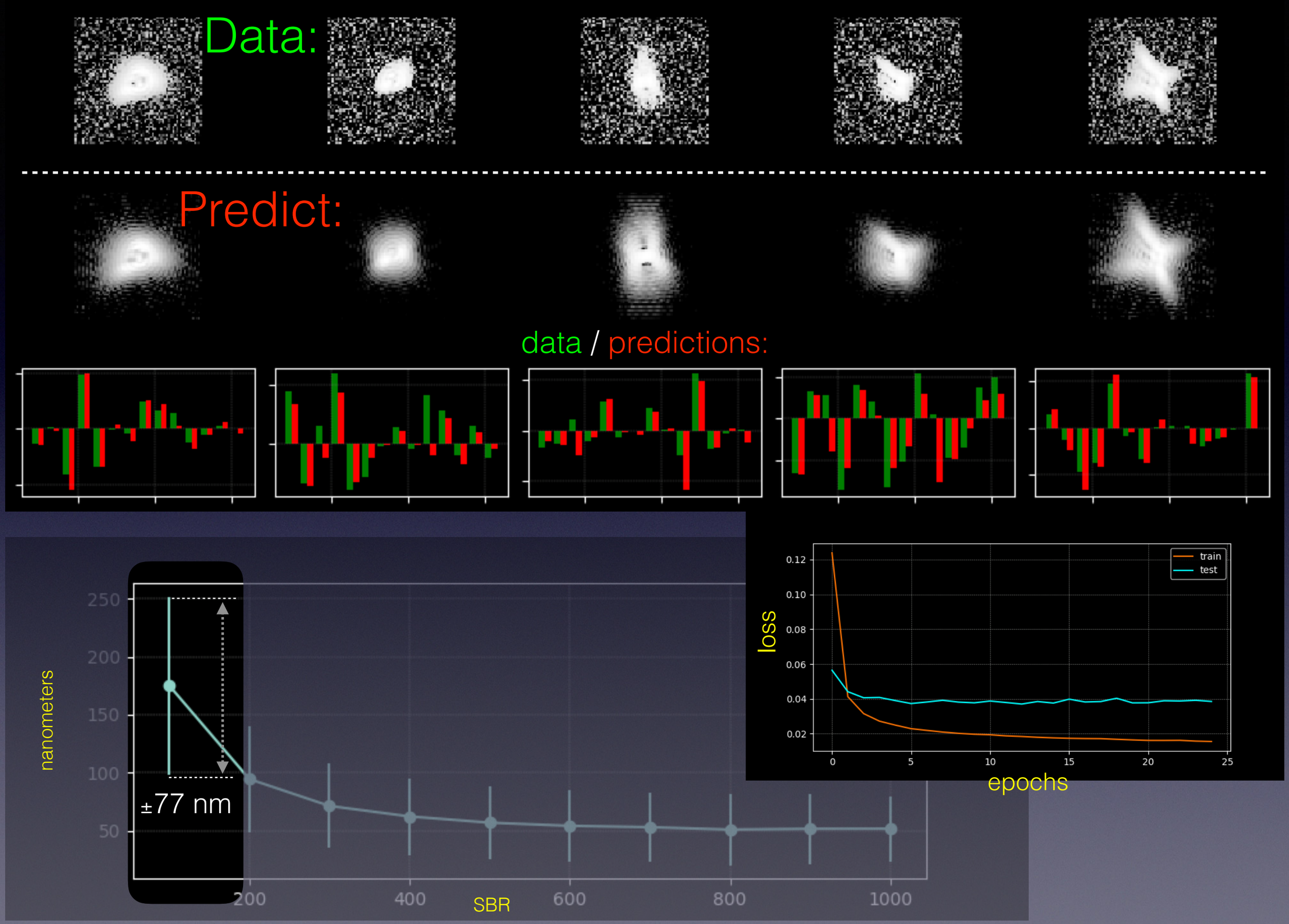
Pinched Optics
(stress in mount)

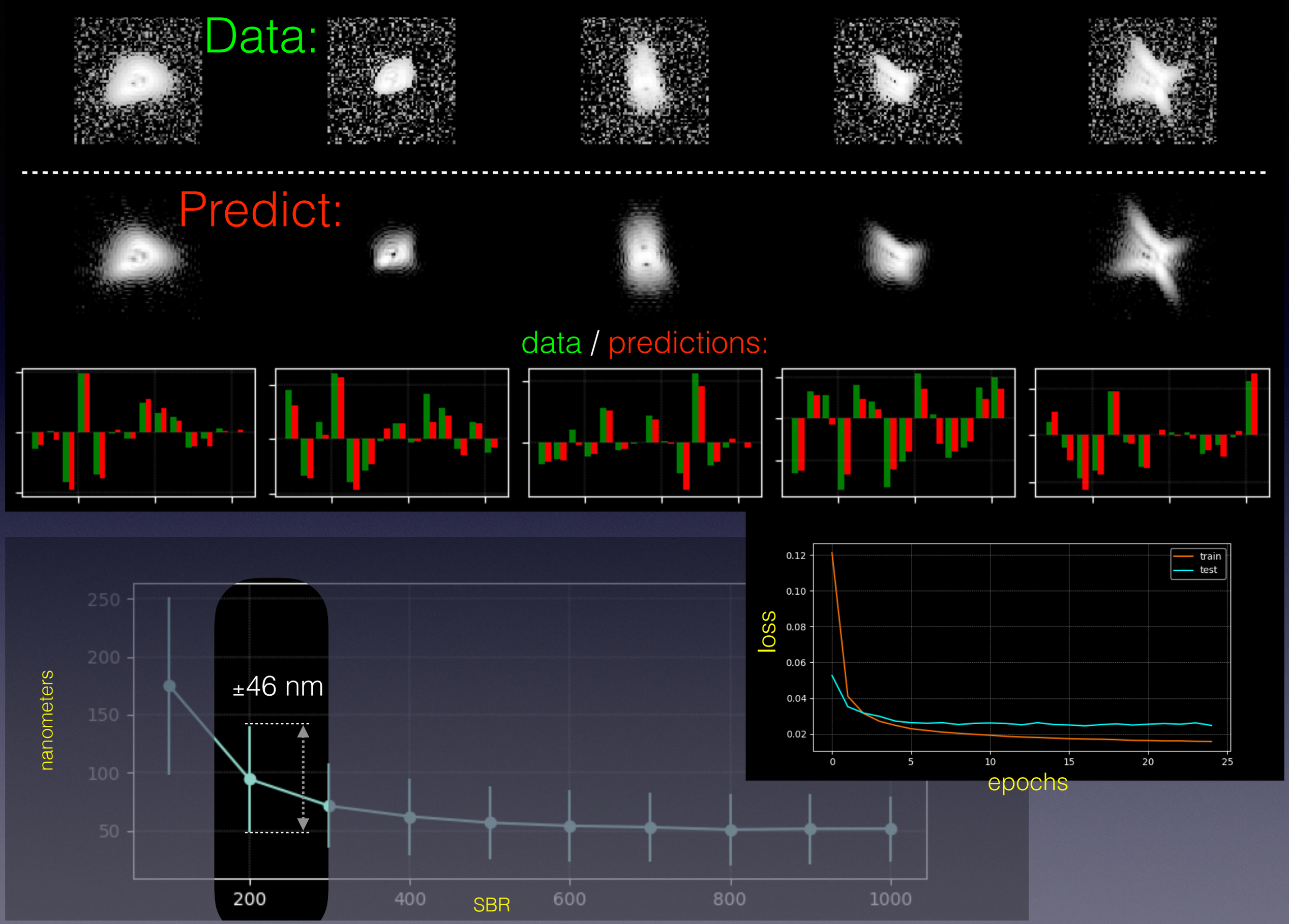


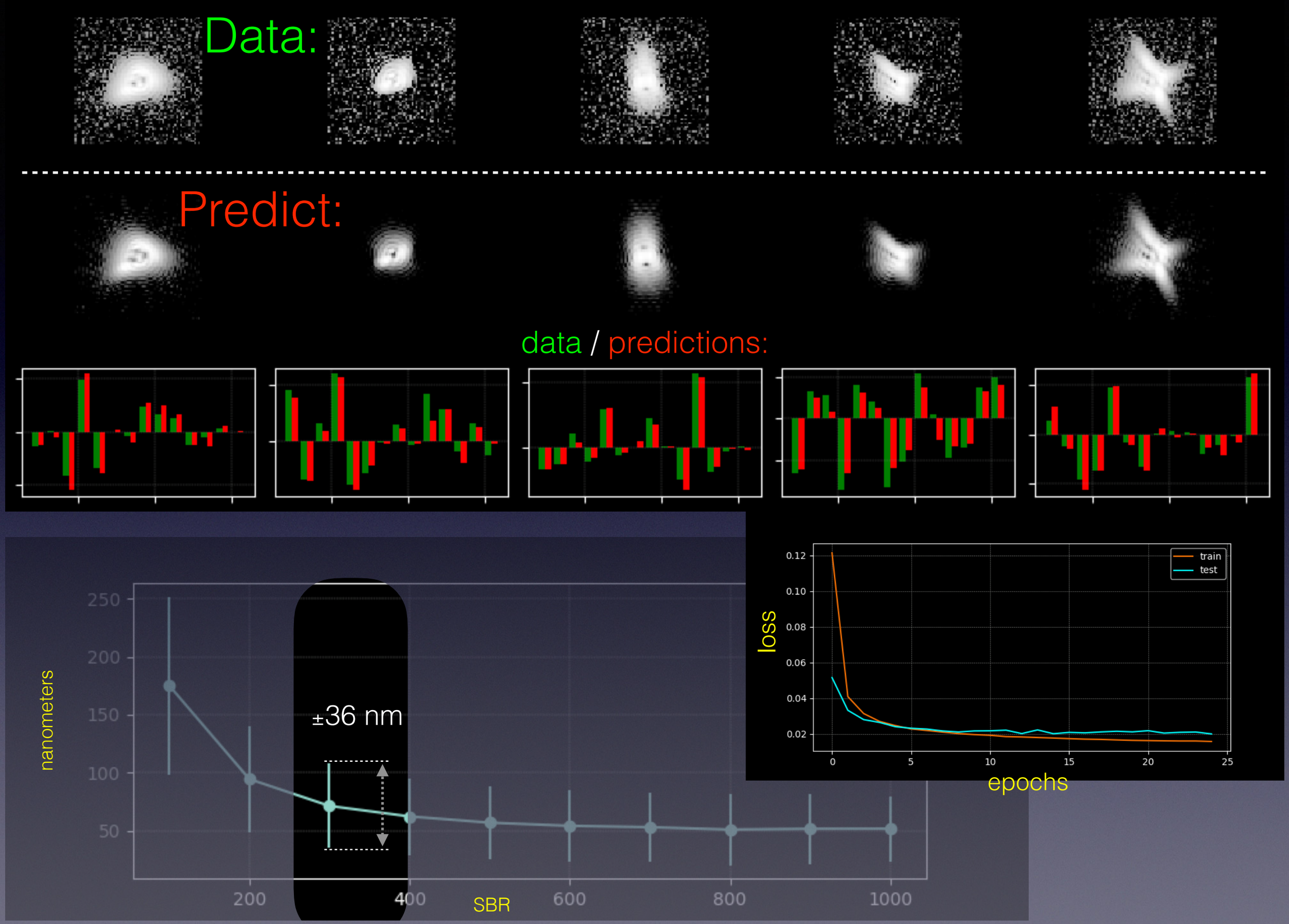
Generalization

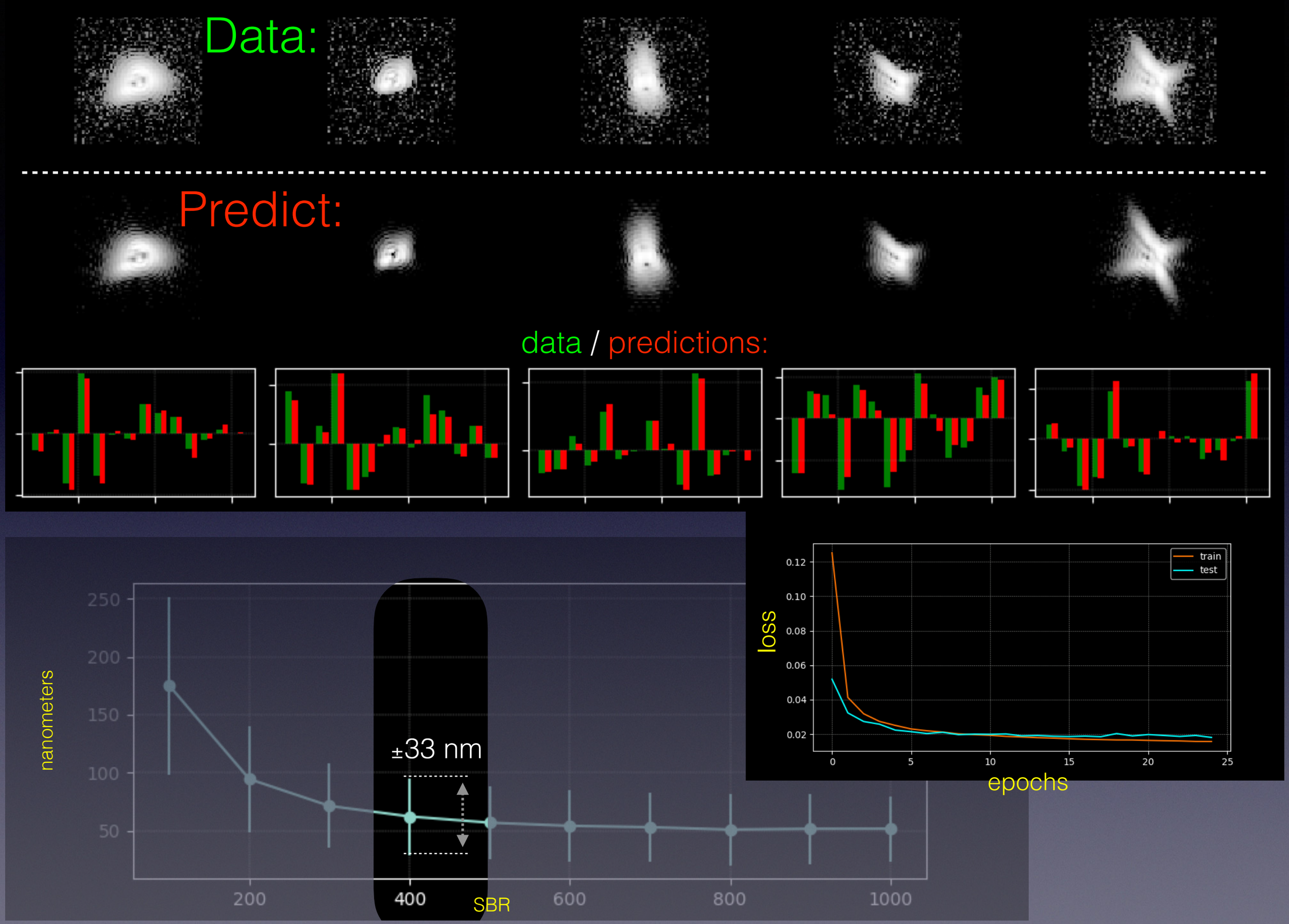
Signal-Noise-Ratio →

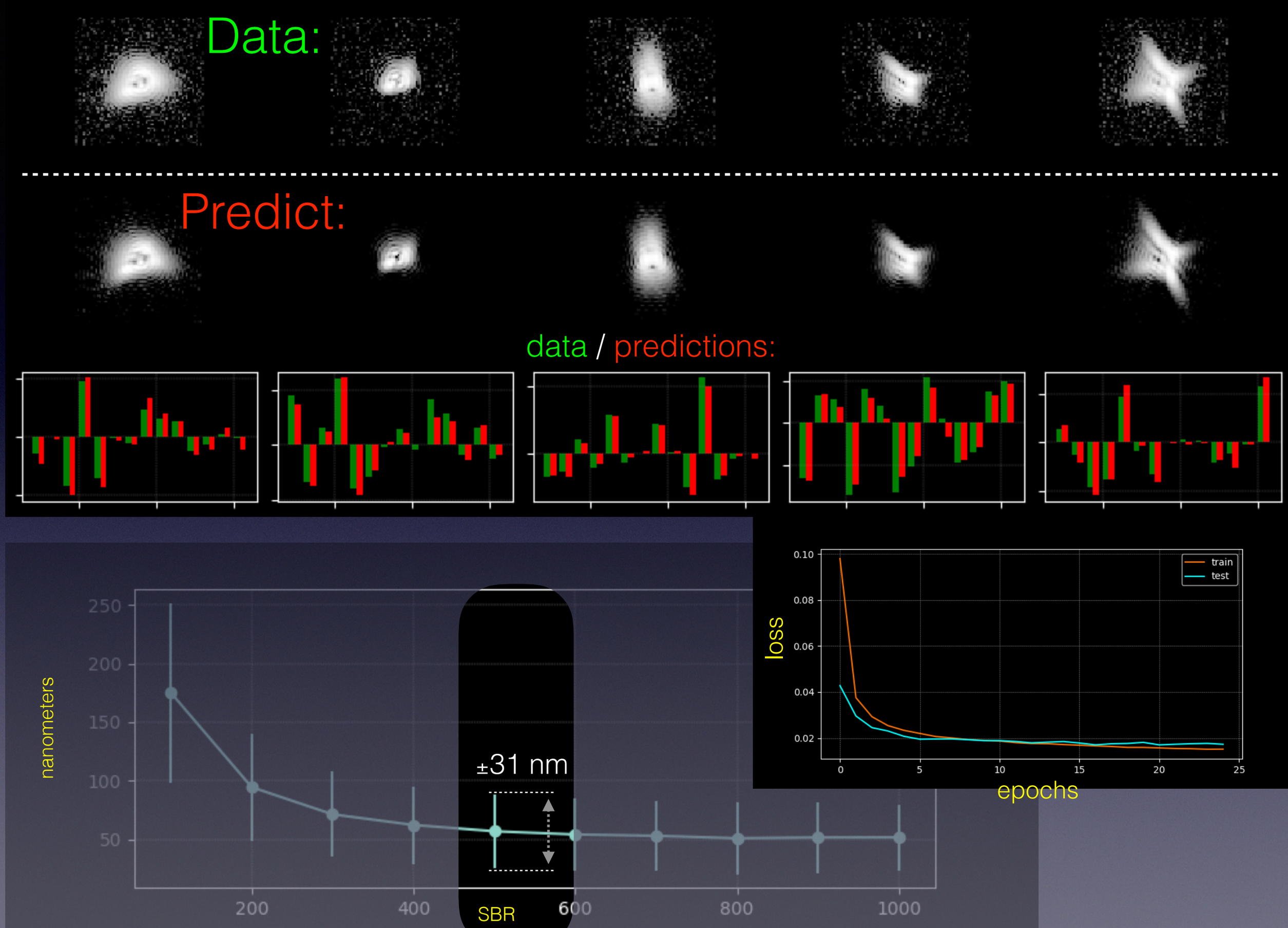


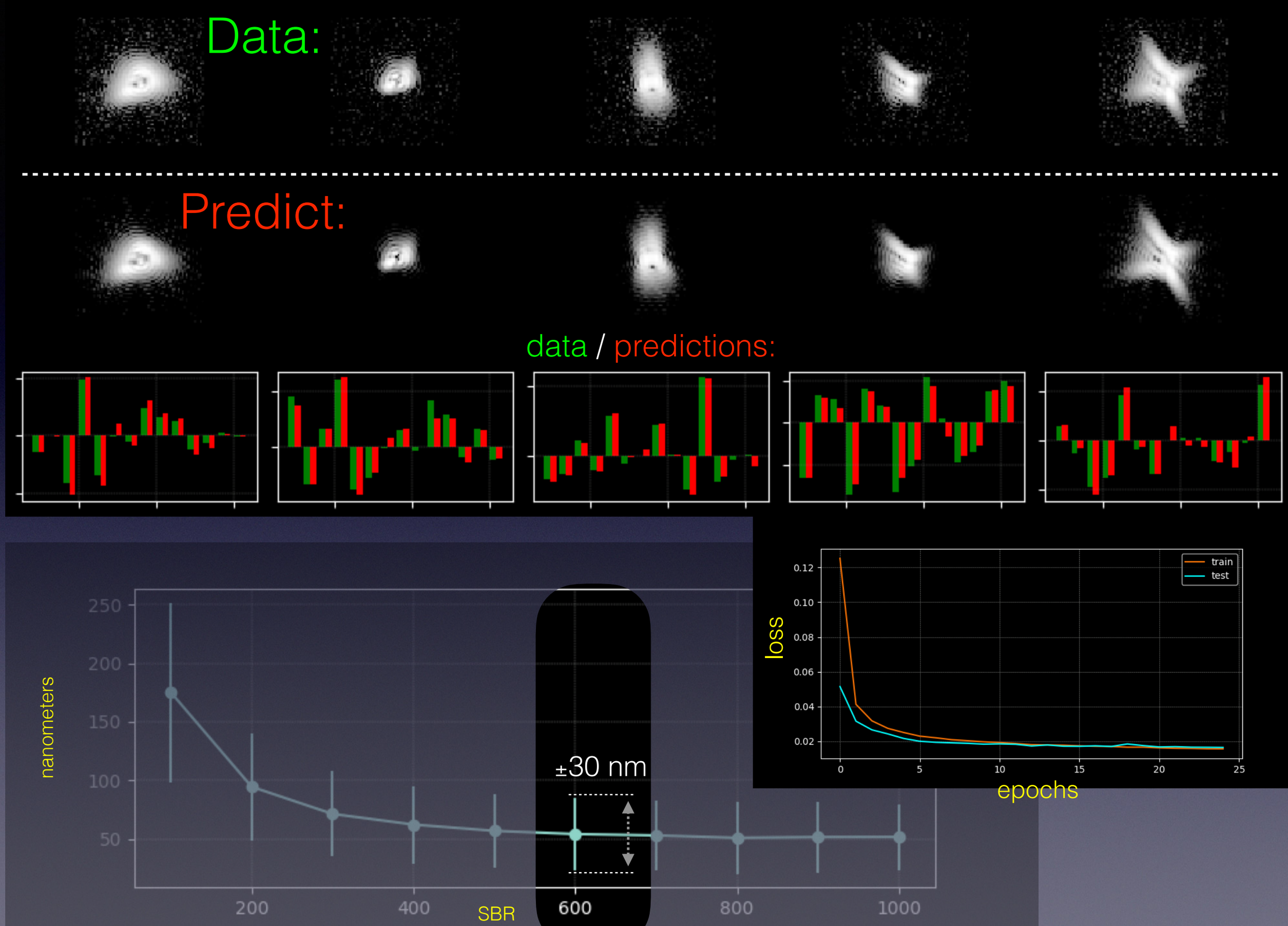

Data:
Predict:
data / predictions:
nanometers
±77 nm
250
200
150
100
50
200
400
600
800
1000
SBR
loss
epochs
train
test
0.12
0.10
0.08
0.06
0.04
0.02
0
5
10
15
20
25

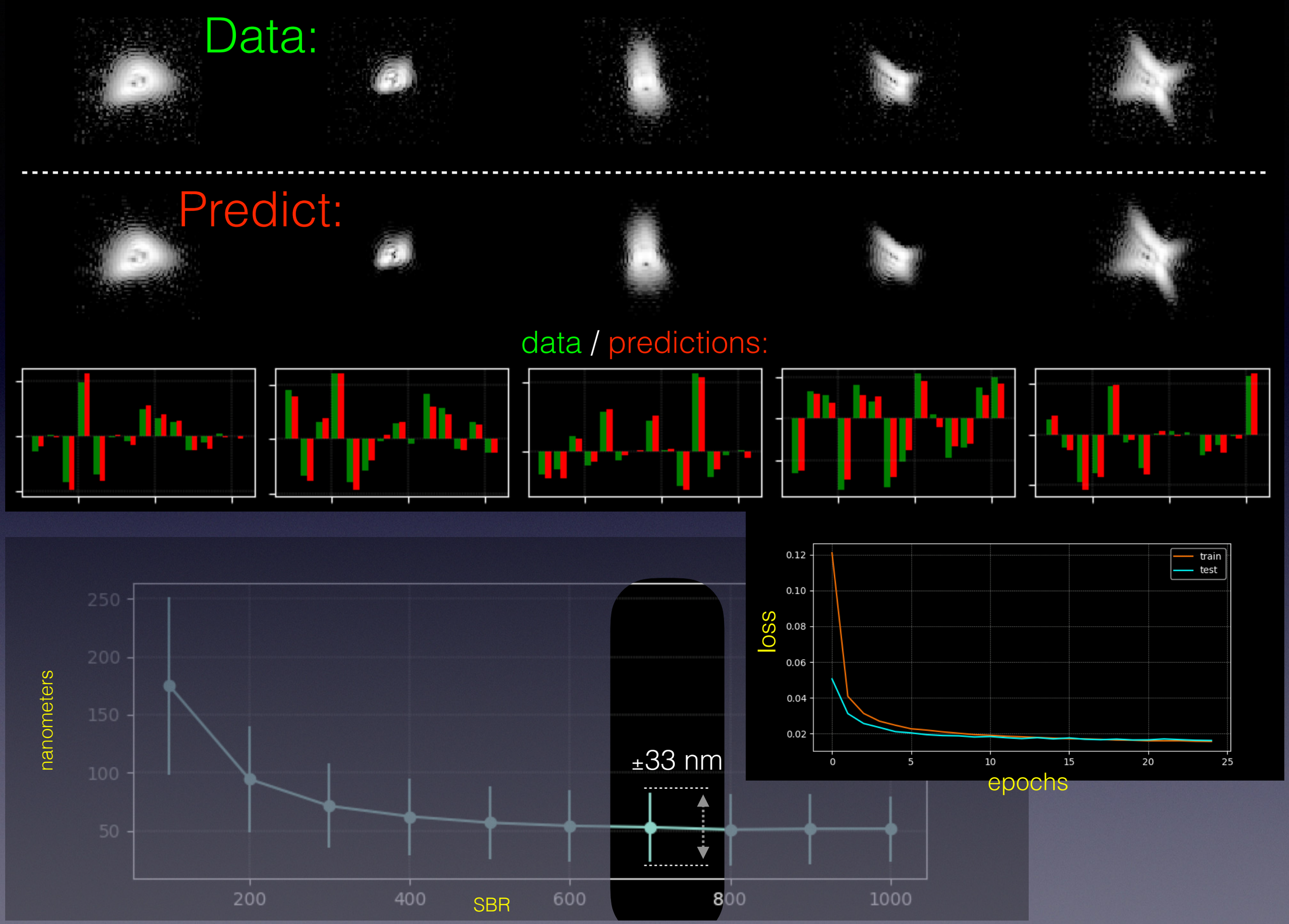

Data:
Predict:
data / predictions:
nanometers
250
200
150
100
50
±46 nm
200
400
SBR
600
800
1000
loss
0.12
0.10
0.08
0.06
0.04
0.02
0
5
10
15
20
25
epochs
train
test

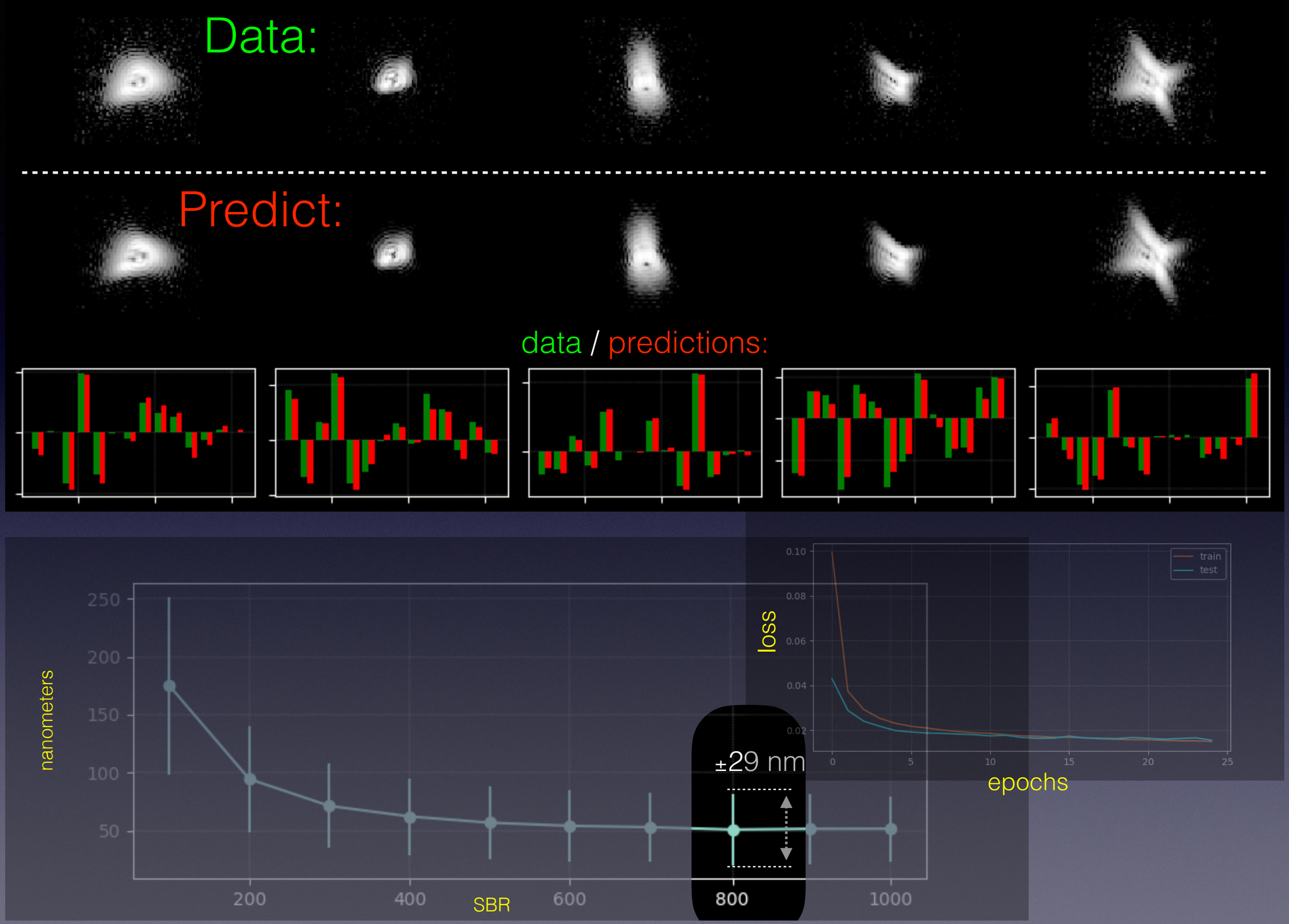

Data:
Predict:
data / predictions:
nanometers
250
200
150
100
50
±36 nm
200
400
600
800
1000
SBR
loss
0.12
0.10
0.08
0.06
0.04
0.02
0
5
10
15
20
25
epochs
train
test

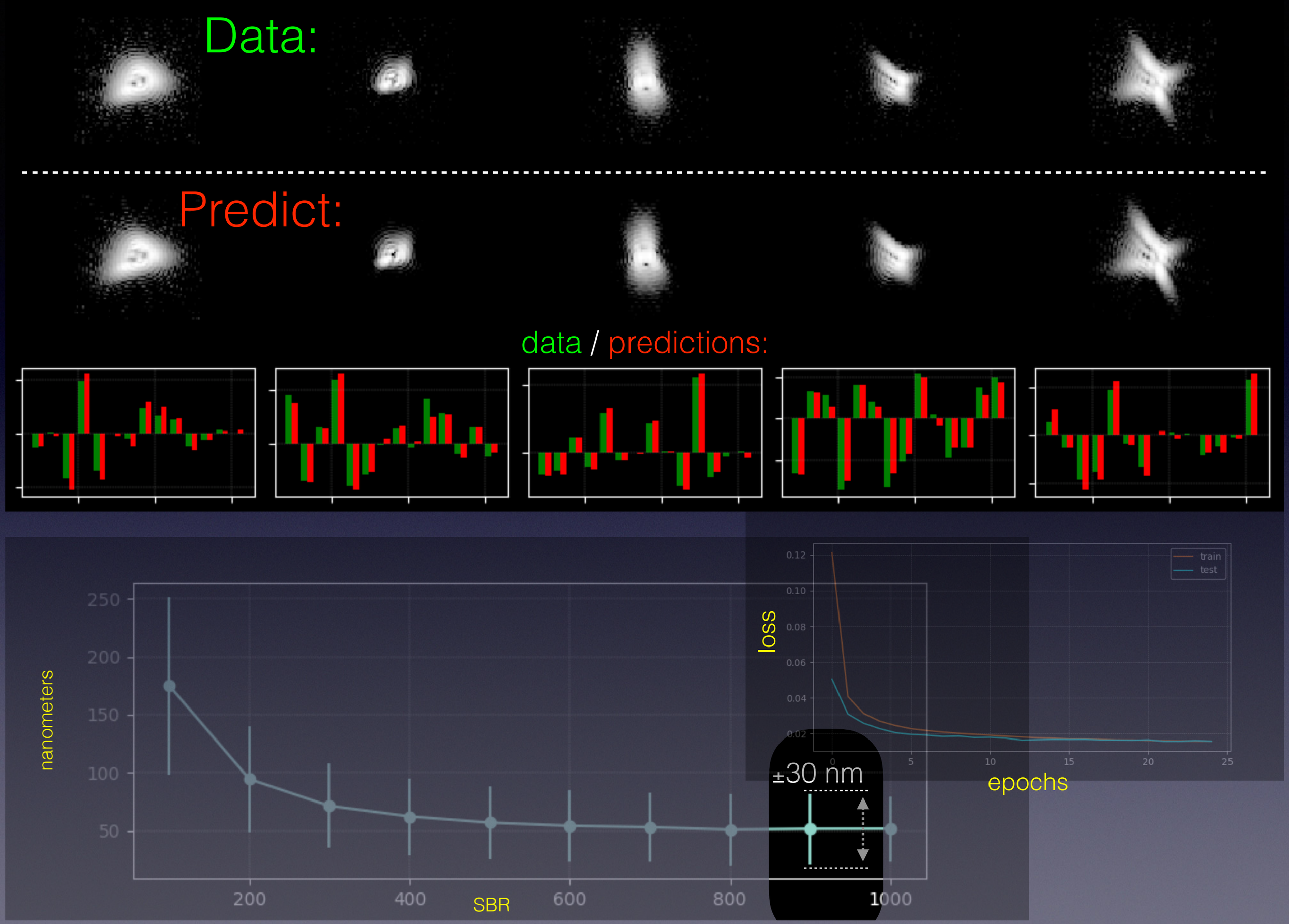

Data:
Predict:
data / predictions:
nanometers
250
200
150
100
50
200
400
600
800
1000
±33 nm
SBR
loss
epochs
train
test
0.12
0.10
0.08
0.06
0.04
0.02
0
5
10
15
20
25

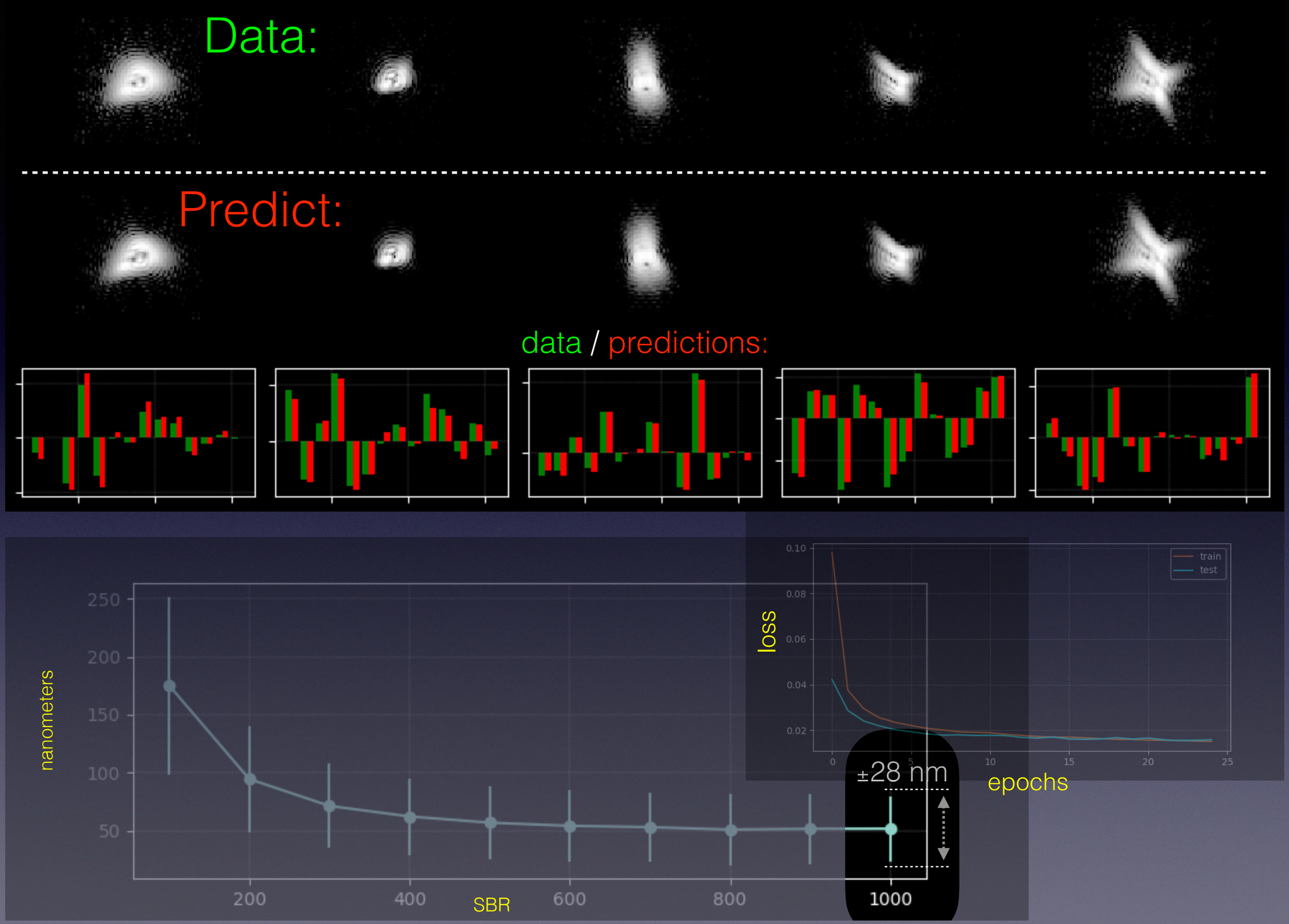

Data:
Predict:
data / predictions:
nanometers
250
200
150
100
50
200
400
600
800
1000
±31 nm
SBR
loss
0.10
0.08
0.06
0.04
0.02
0
5
10
15
20
25
epochs
train
test


Data:
Predict:
data / predictions:
nanometers
250
200
150
100
50
200
400
SBR
600
800
1000
±30 nm
loss
0.12
0.10
0.08
0.06
0.04
0.02
0
5
10
15
20
25
epochs
train
test


Data:
Predict:
data / predictions:
nanometers
250
200
150
100
50
200
400
600
800
1000
SBR
±33 nm
loss
0.12
0.10
0.08
0.06
0.04
0.02
0
5
10
15
20
25
epochs
train
test


Data:
Predict:
data / predictions:
nanometers
250
200
150
100
50
200
400
SBR
600
800
1000
±29 nm
loss
0.10
0.08
0.06
0.04
0.02
0
5
10
15
20
25
epochs
train
test


Data:
Predict:
data / predictions:
nanometers
250
200
150
100
50
200
400
600
800
1000
SBR
±30 nm
loss
0.12
0.10
0.08
0.06
0.04
0.02
0
5
10
15
20
25
epochs
train
test

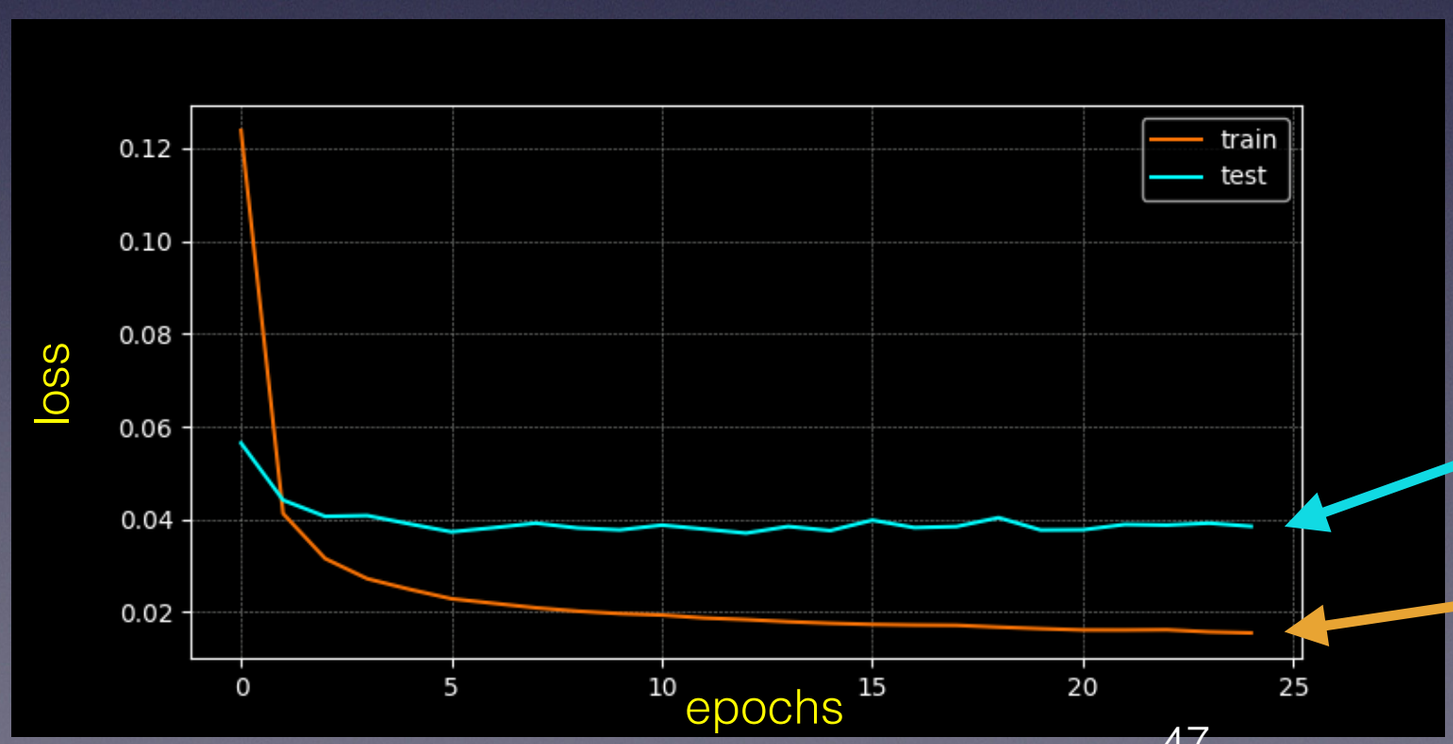

Data:
Predict:
data / predictions:
nanometers
250
200
150
100
50
200
400
600
800
1000
SBR
±28 nm
loss
0.10
0.08
0.06
0.04
0.02
0
5
10
15
20
25
epochs
train
test

Performance

Underfit Optimal Overfit

Performance

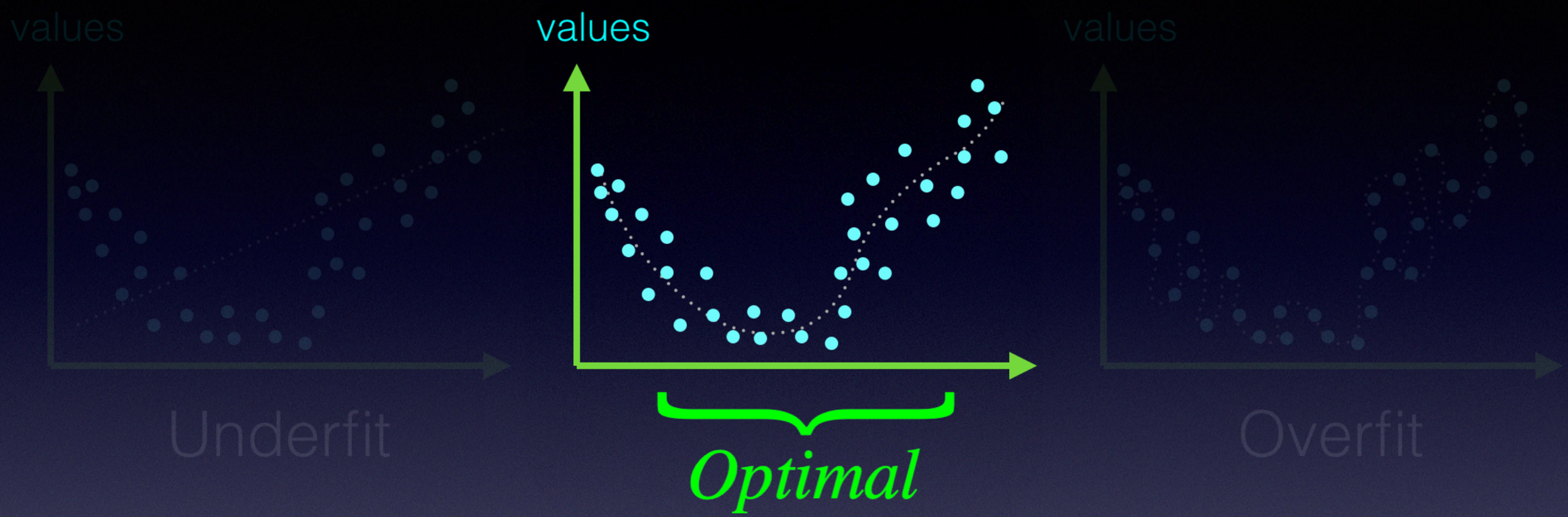
Underfit Optimal Overfit

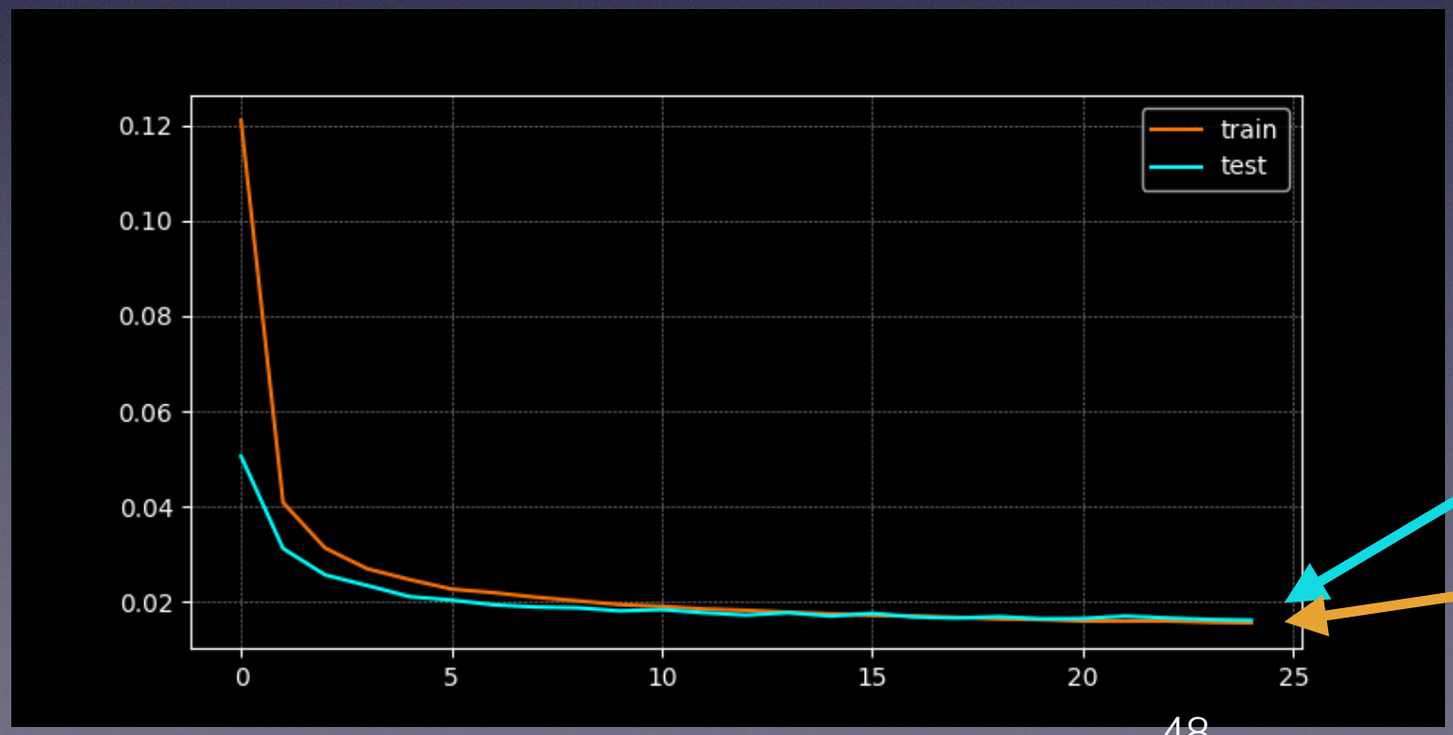


error for test set larger

error for train set smaller

Performance

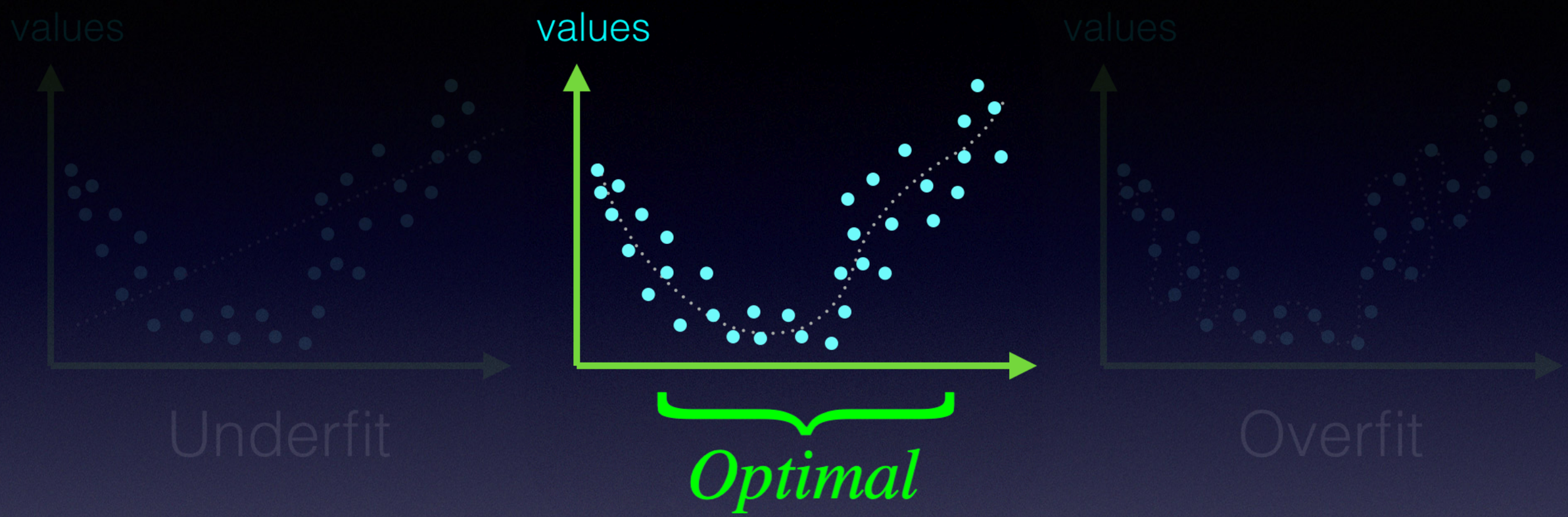


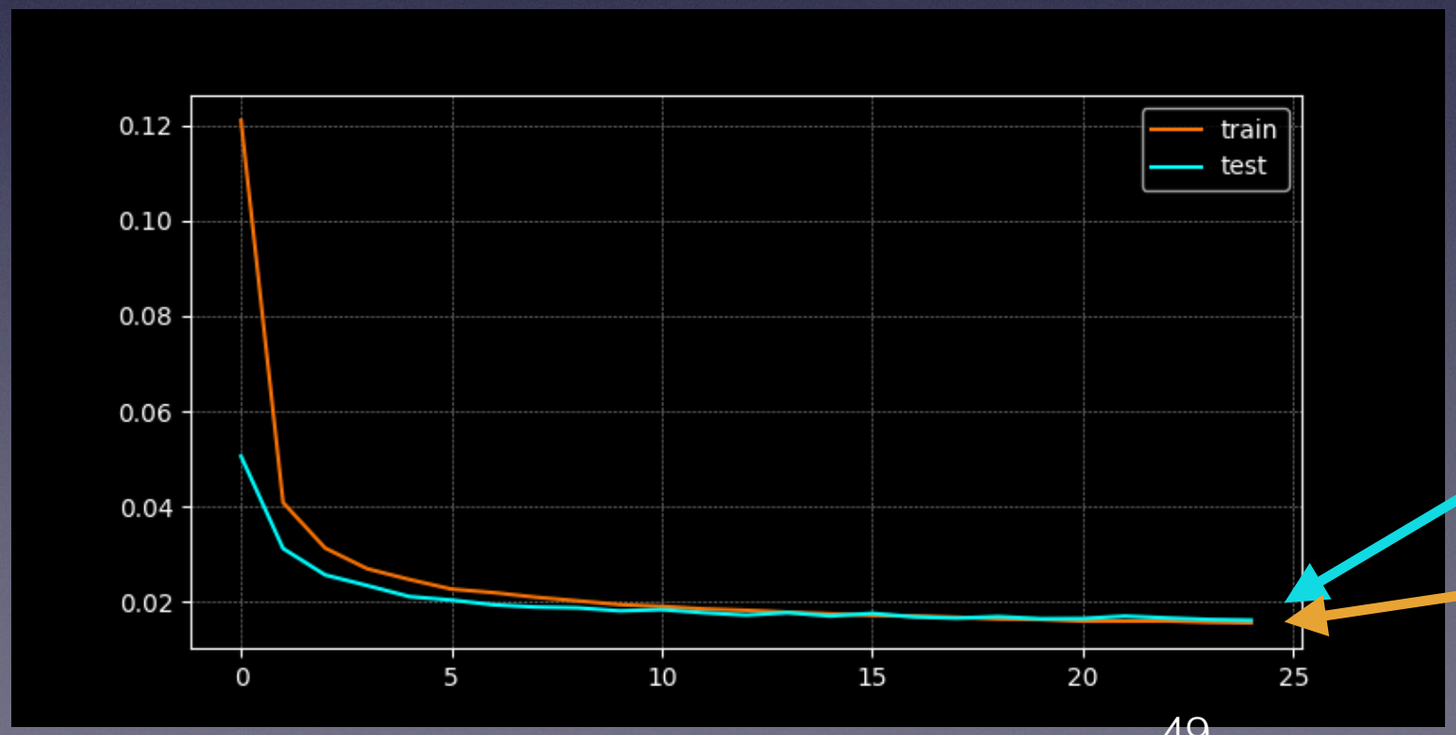


error for test set

error for train set

Performance





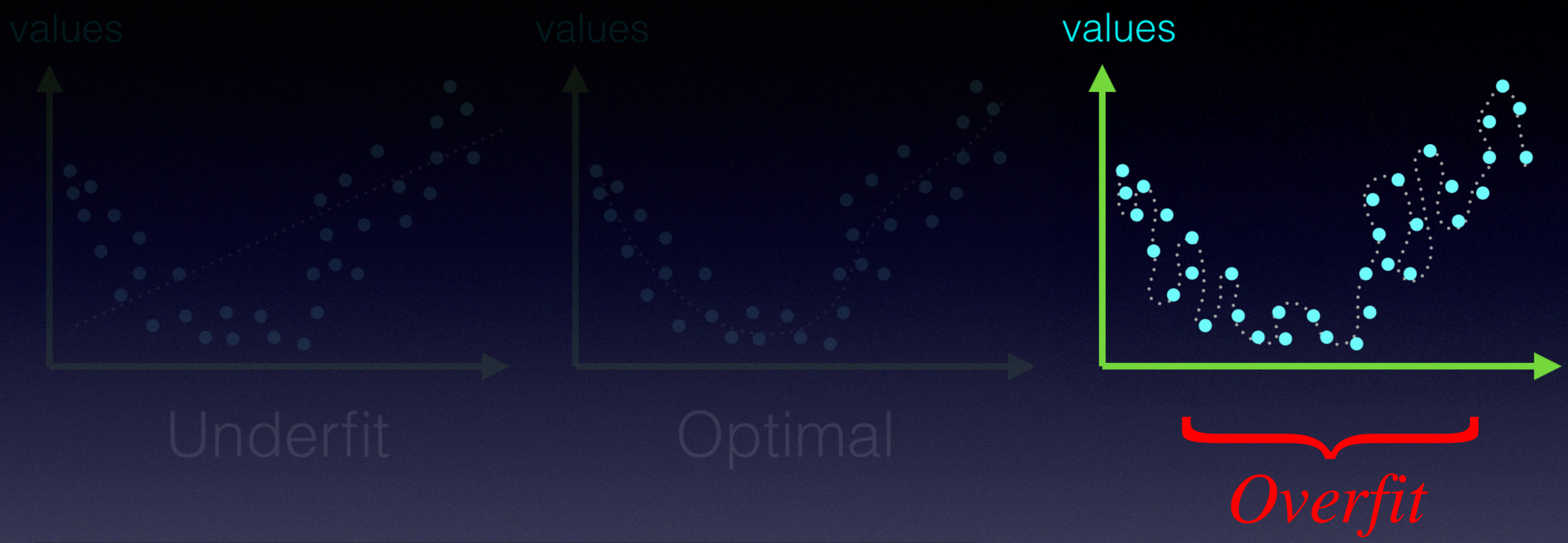
Best for Generalization

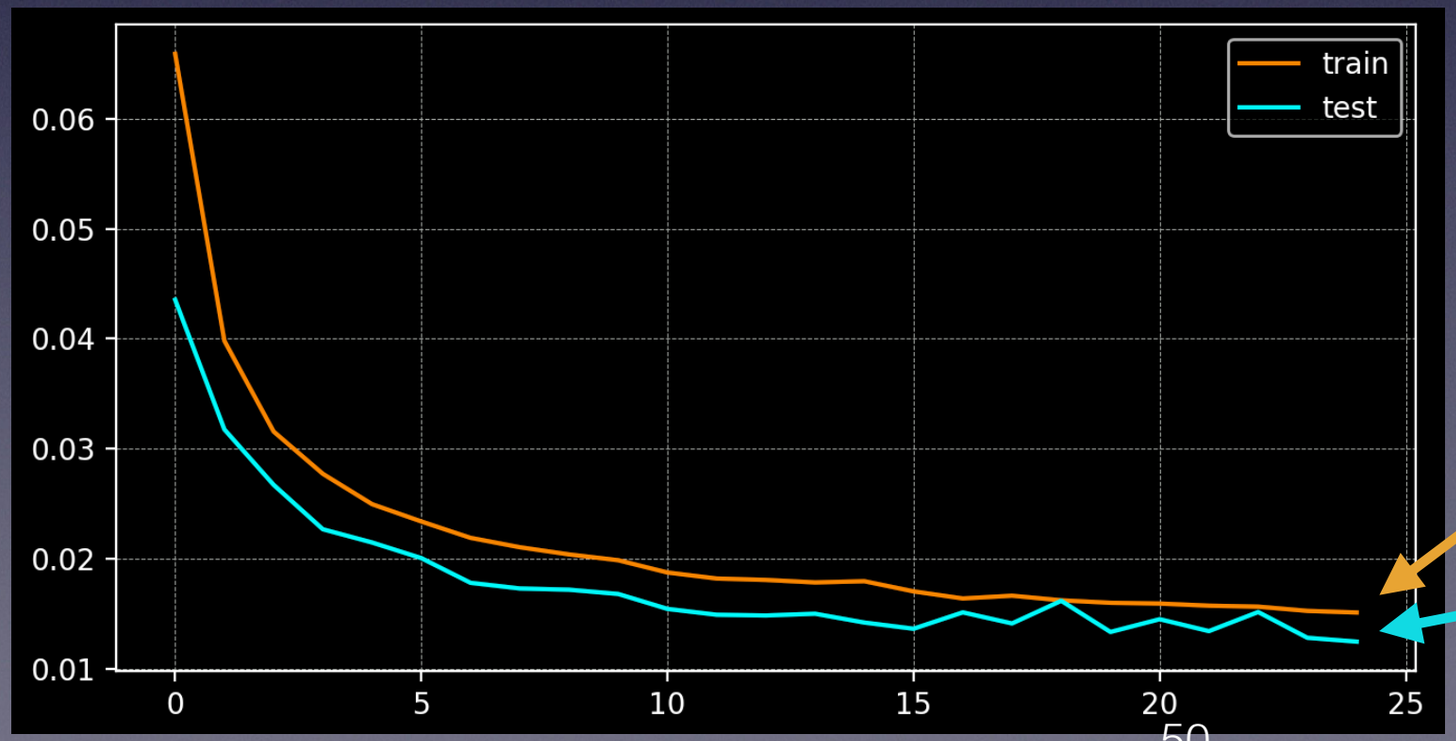
error for test set

error for train set

Performance

Memorization (look-up table)



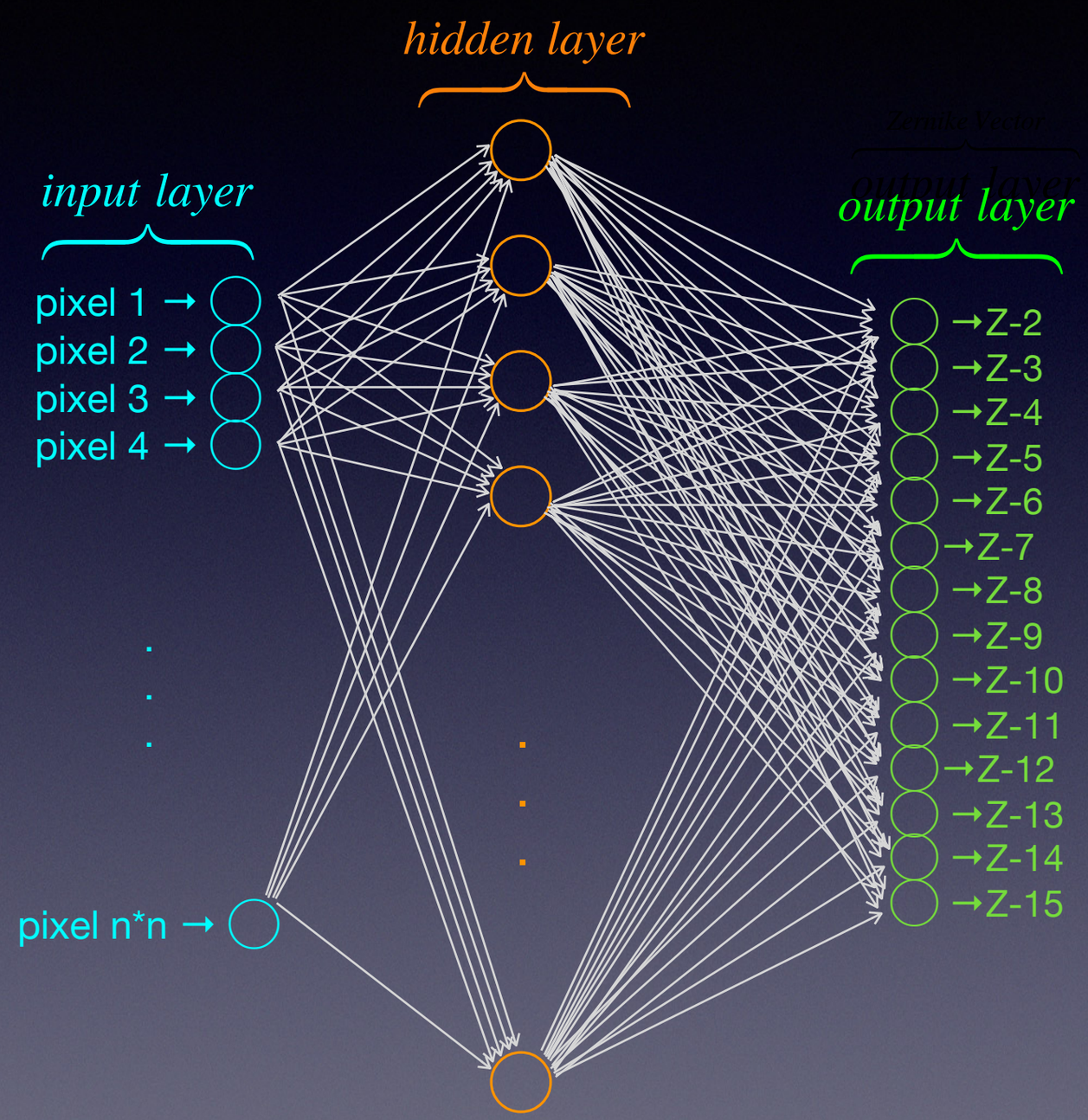


error for train set

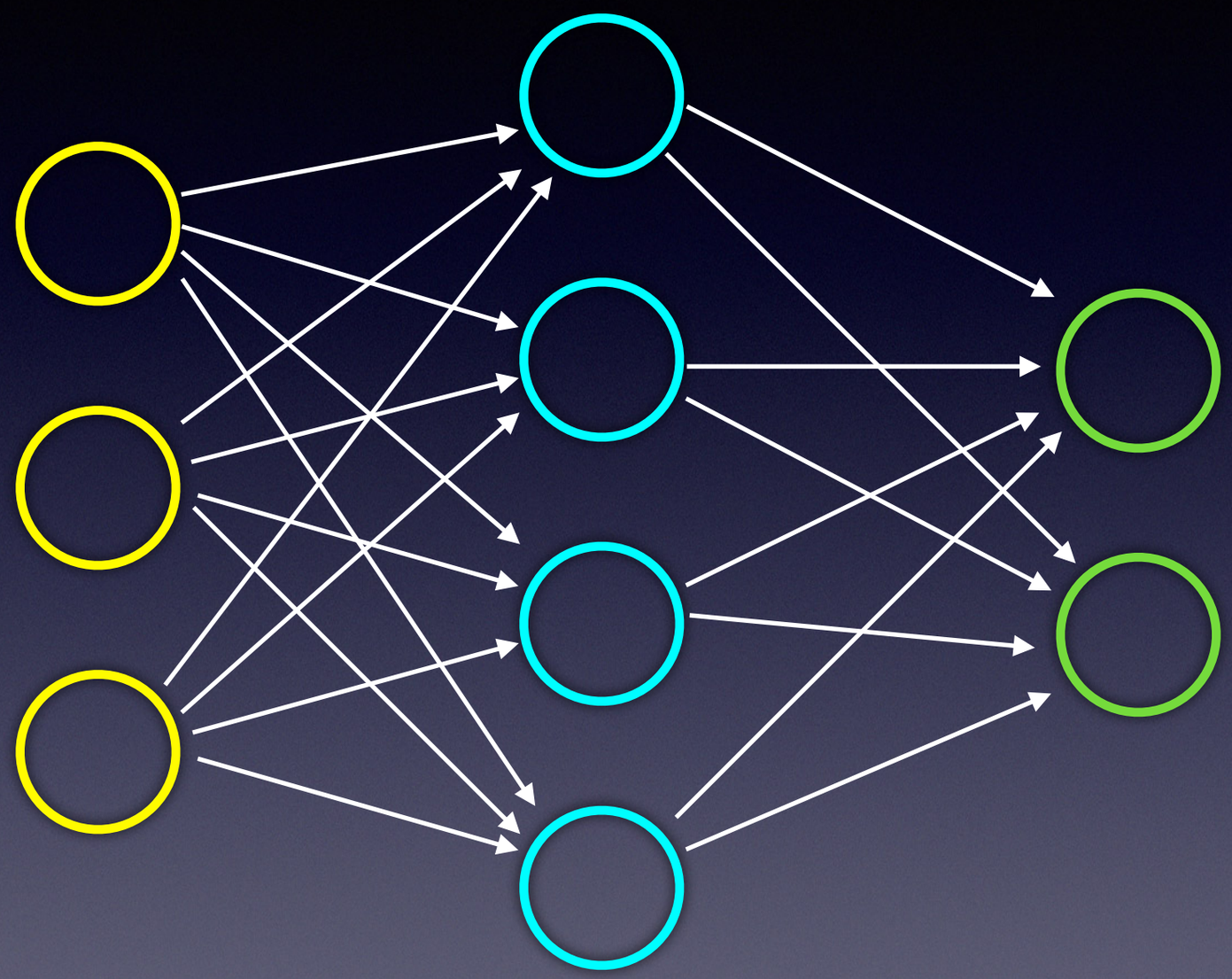
error for test set

Hyperparameter Optimization

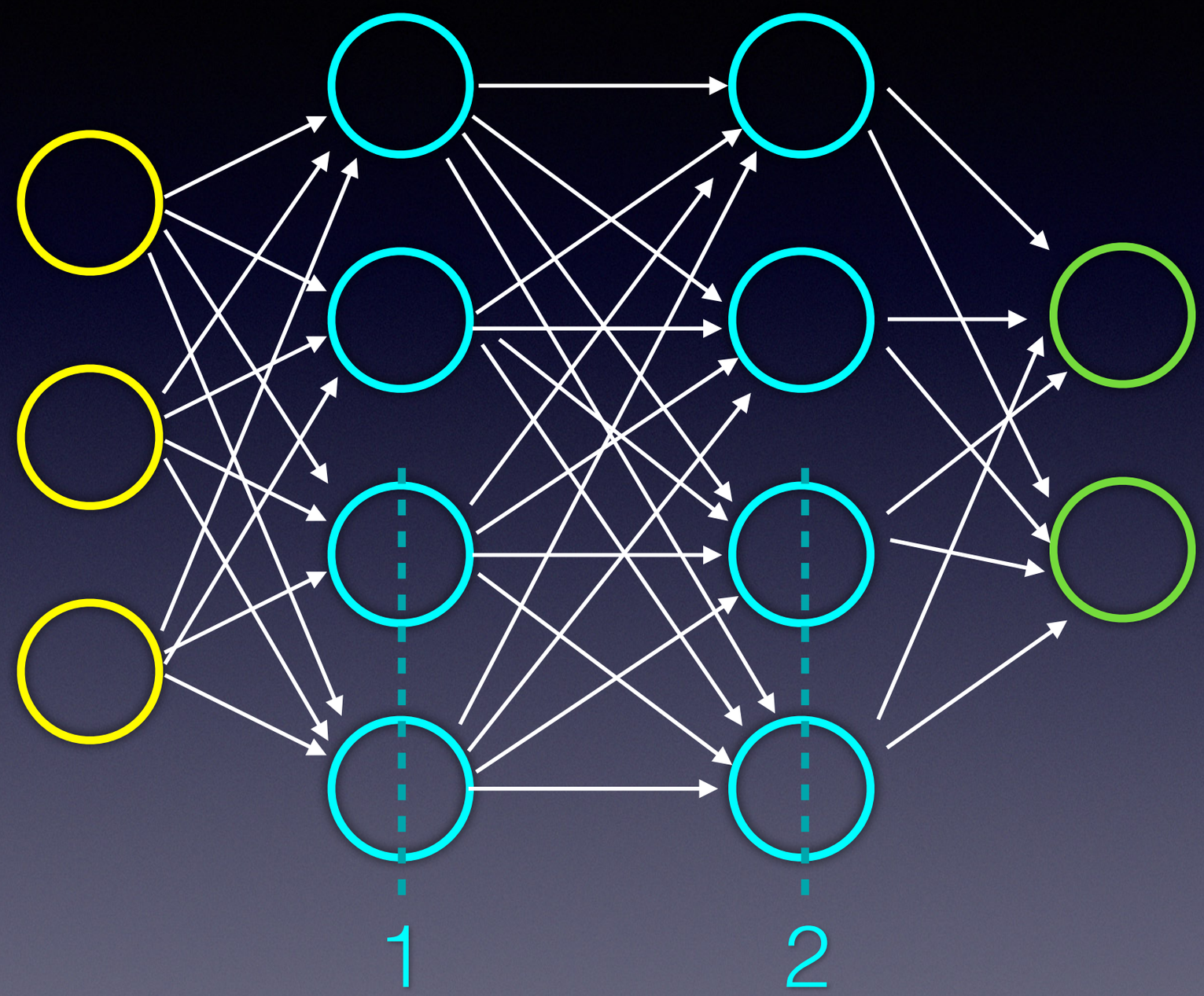
“Naive” Architecture



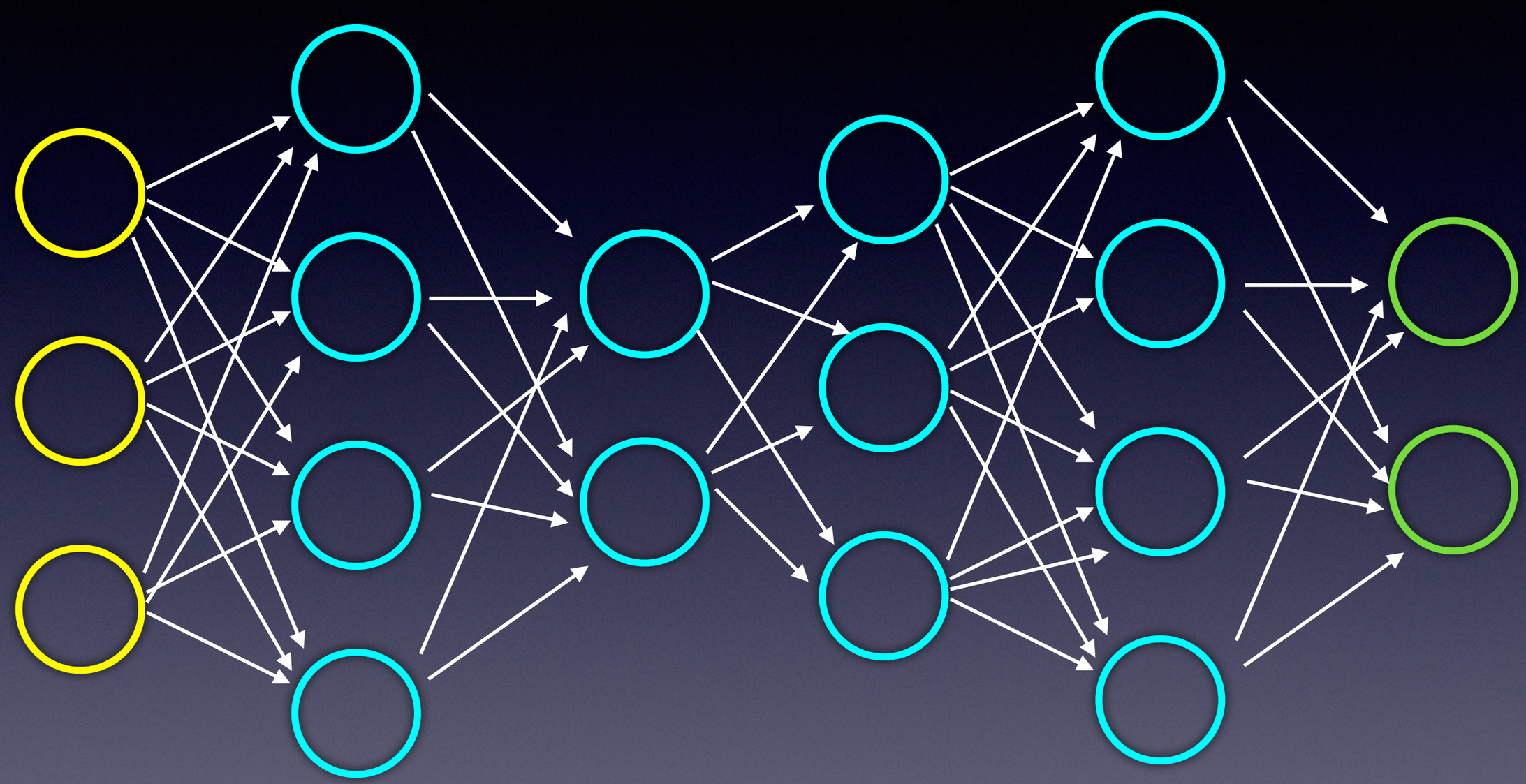
Many Choices



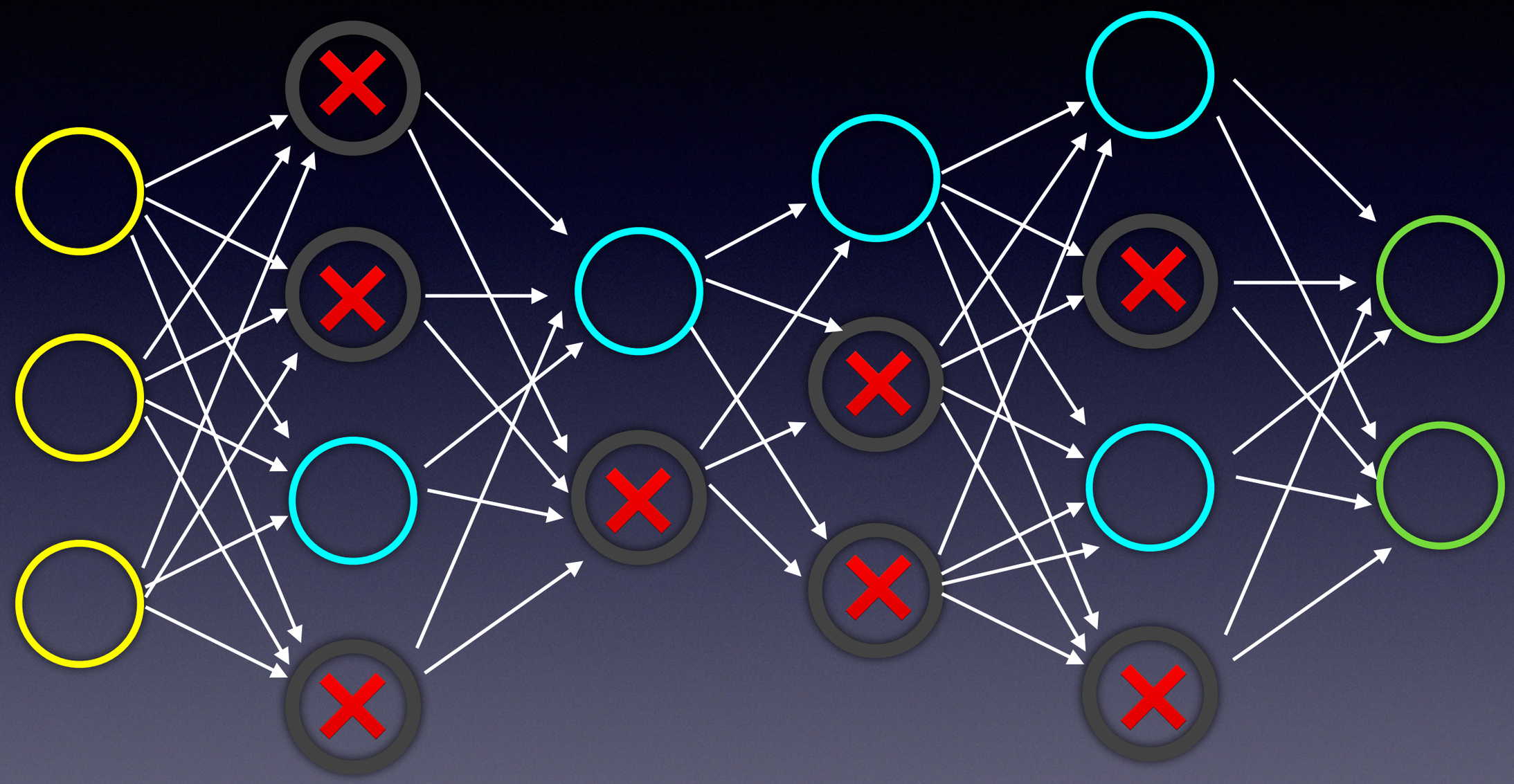
Hidden Layers



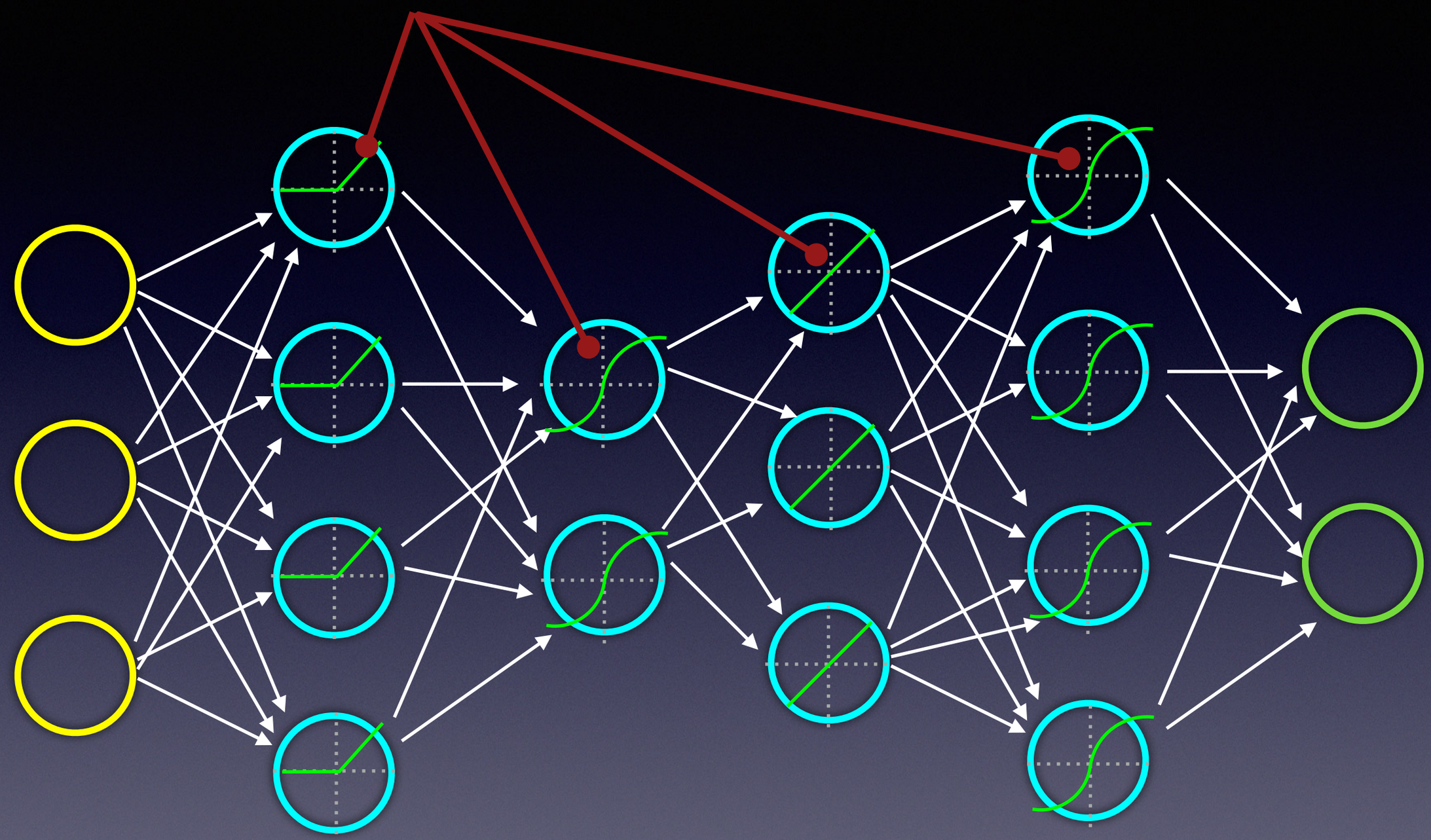
Nodes, Layers



Dropout



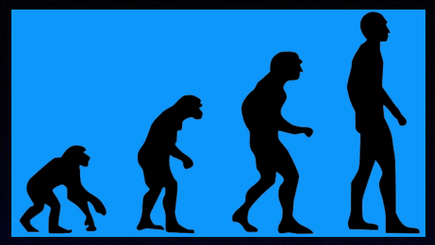
Activation Functions

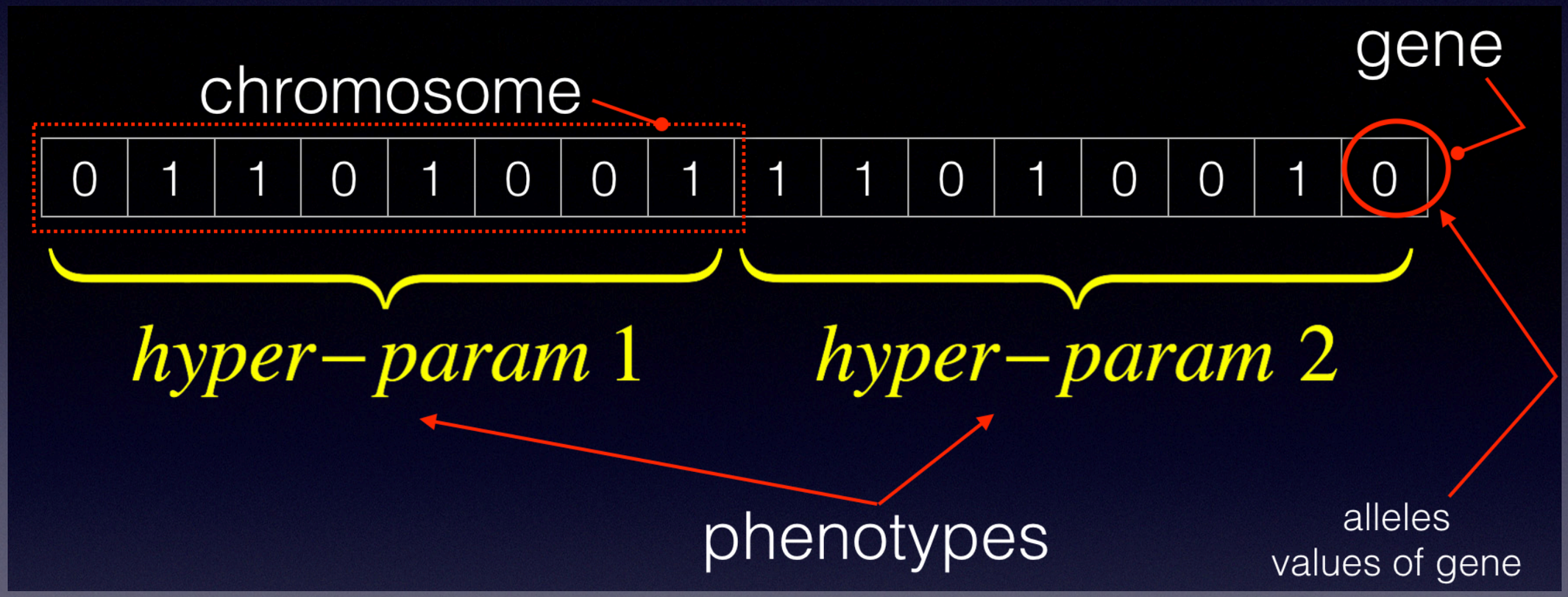


How to Solve for Best Hyperparameters?

How to Solve for Best Hyperparameters?

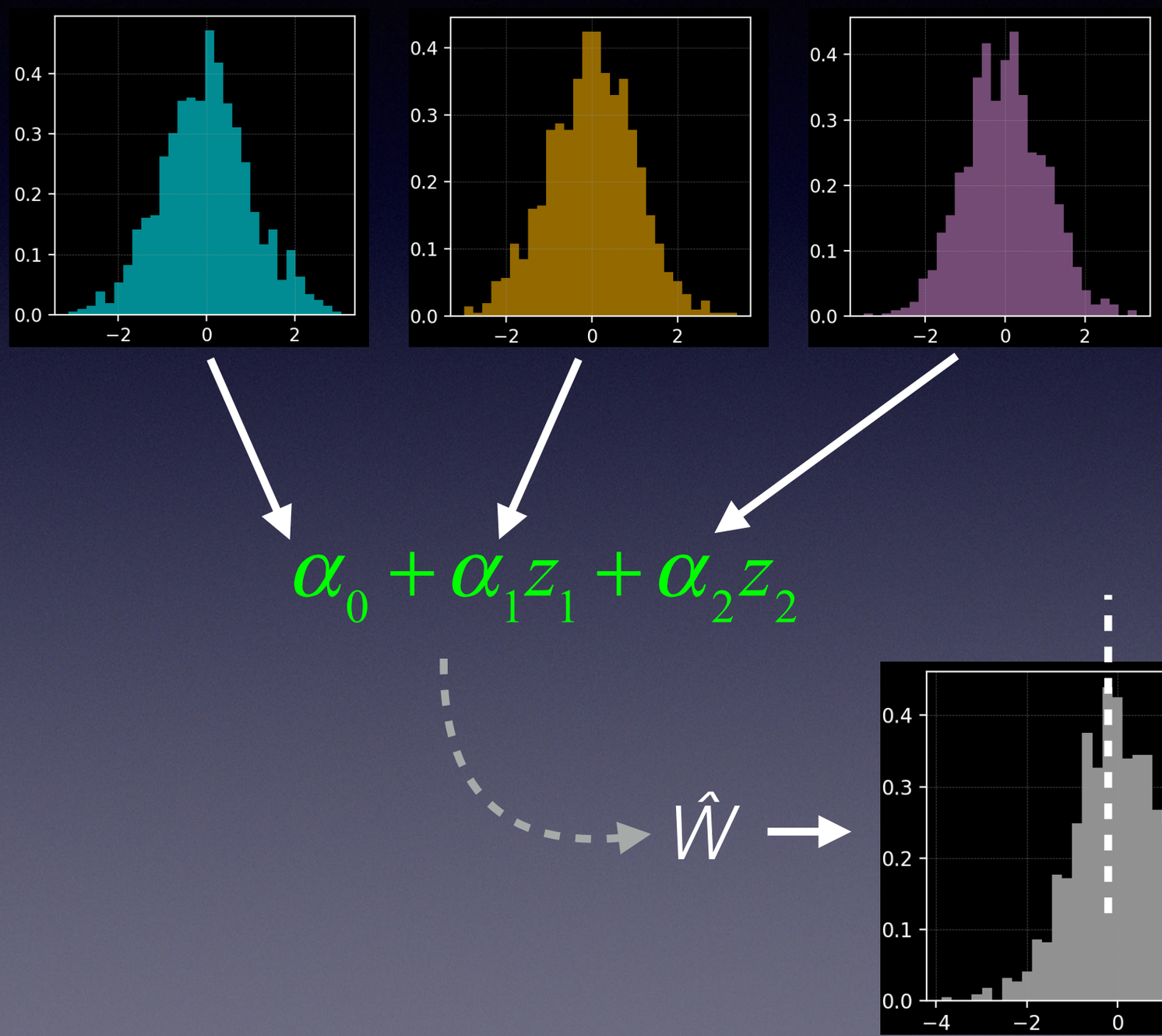
Genetic Algorithm:



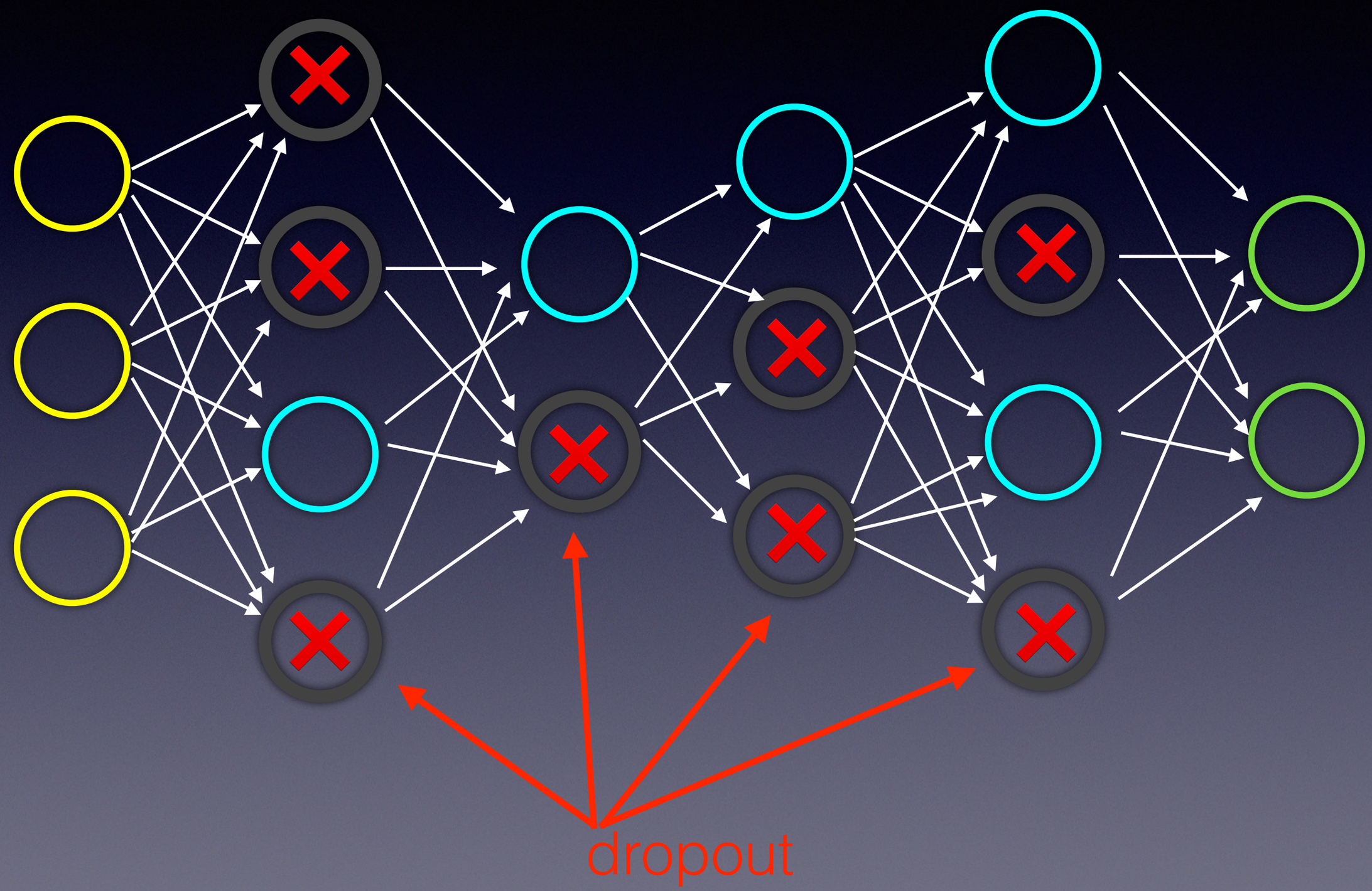


How to Solve for Best Hyperparameters?

Bayesian Optimization:

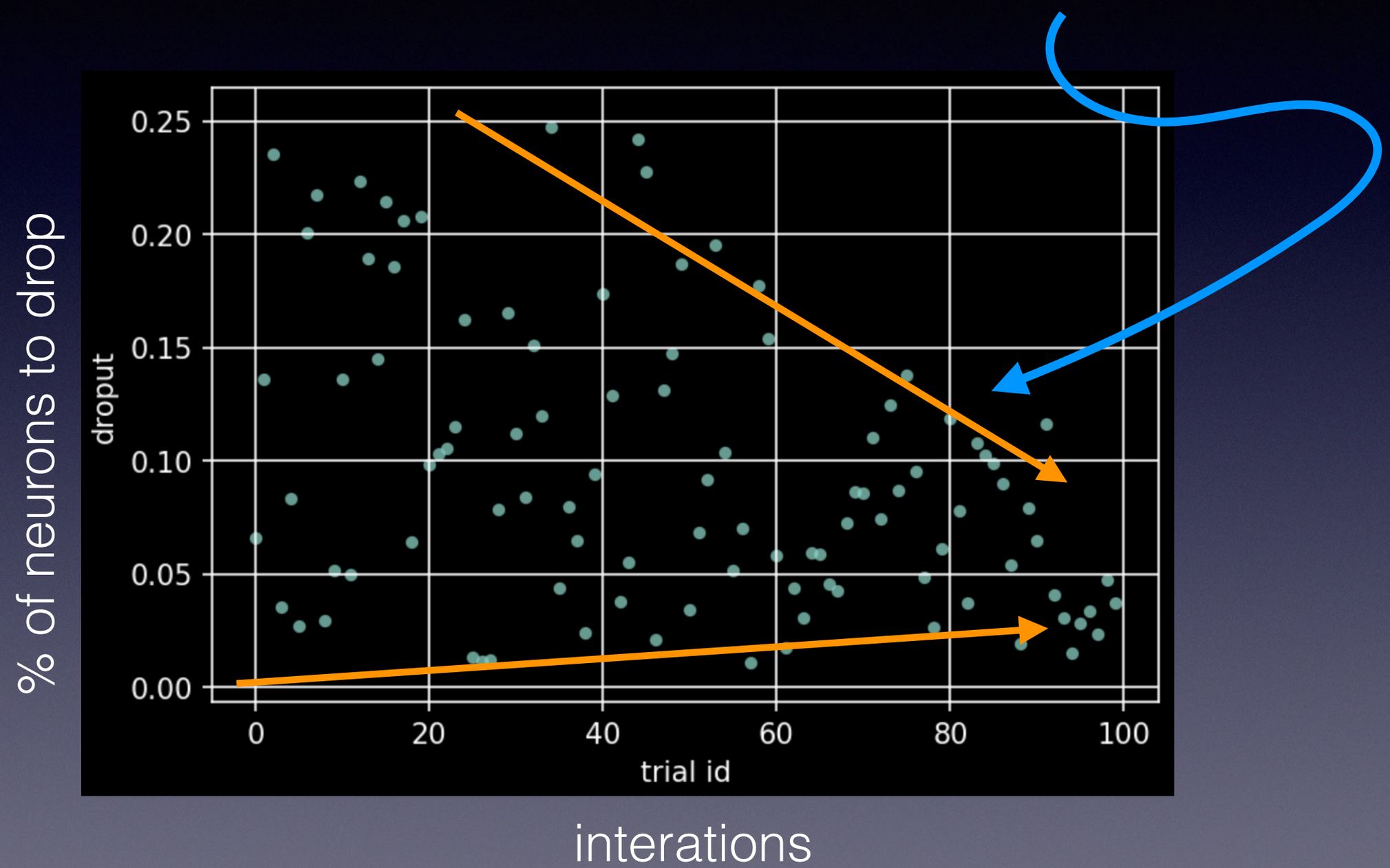


Bayesian Dropout Optimization



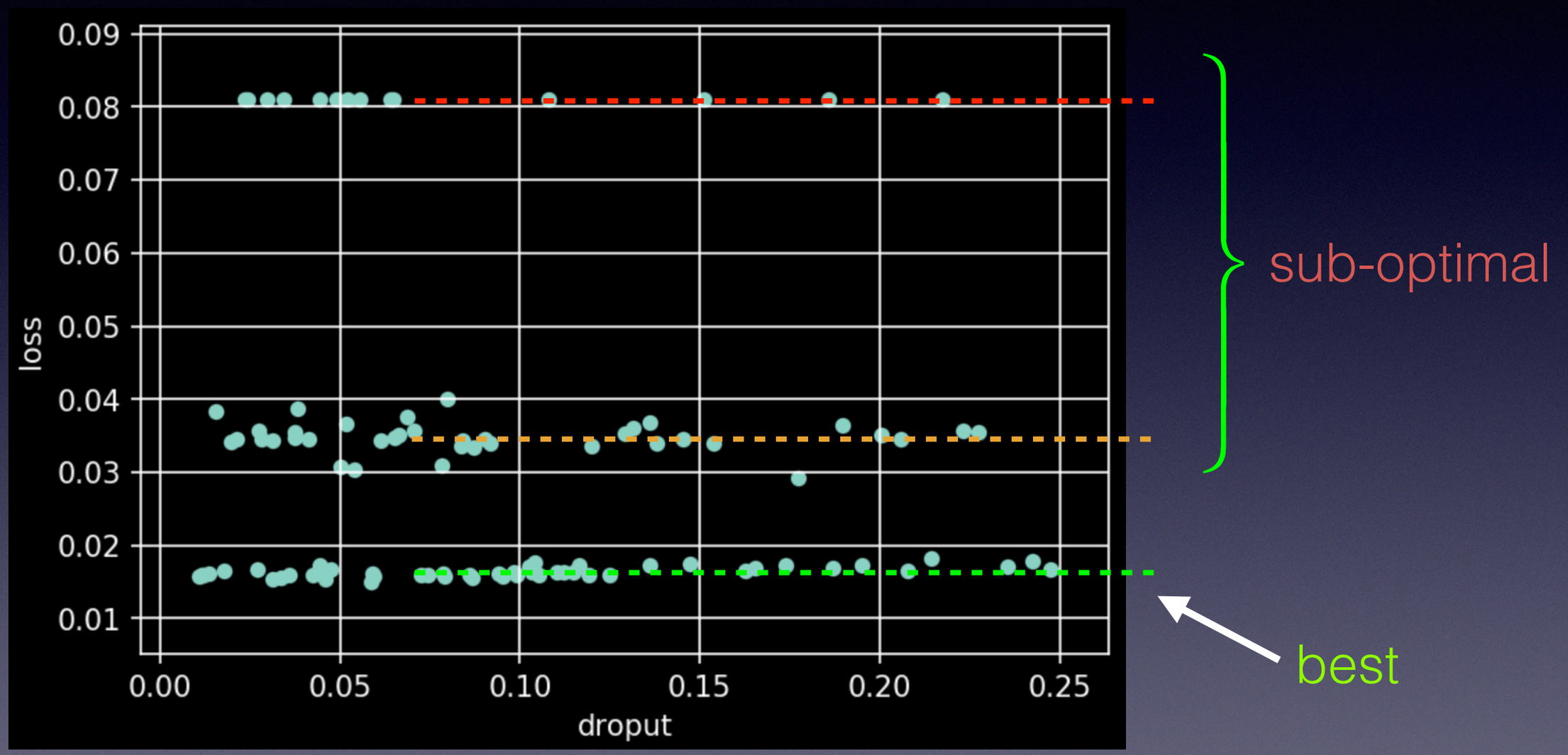
Bayesian Dropout Optimization

zeroing-in: range is narrowing



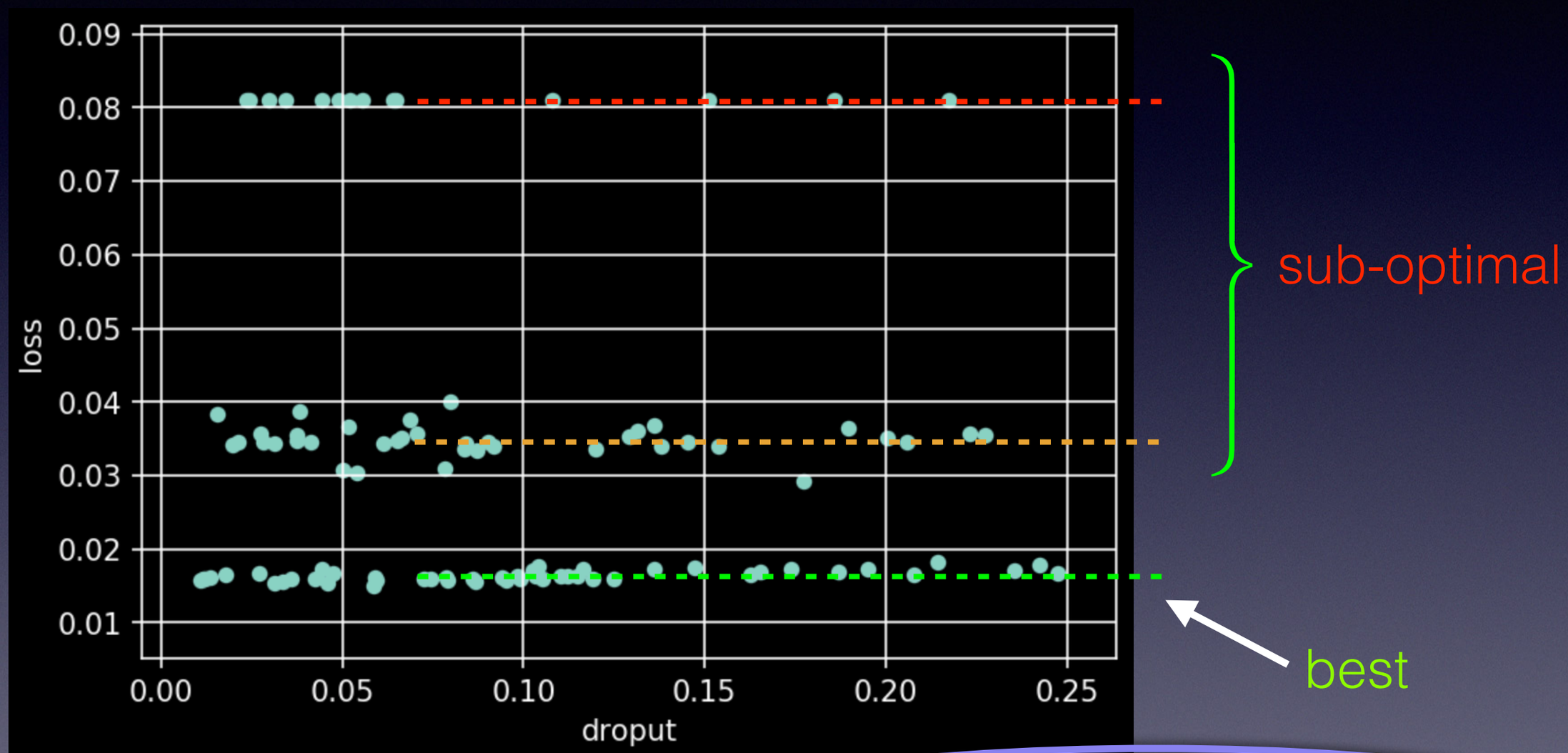
Bayesian Dropout Optimization

3-families of architectures begin to cluster:



Bayesian Optimization

3-families of architectures begin to cluster:



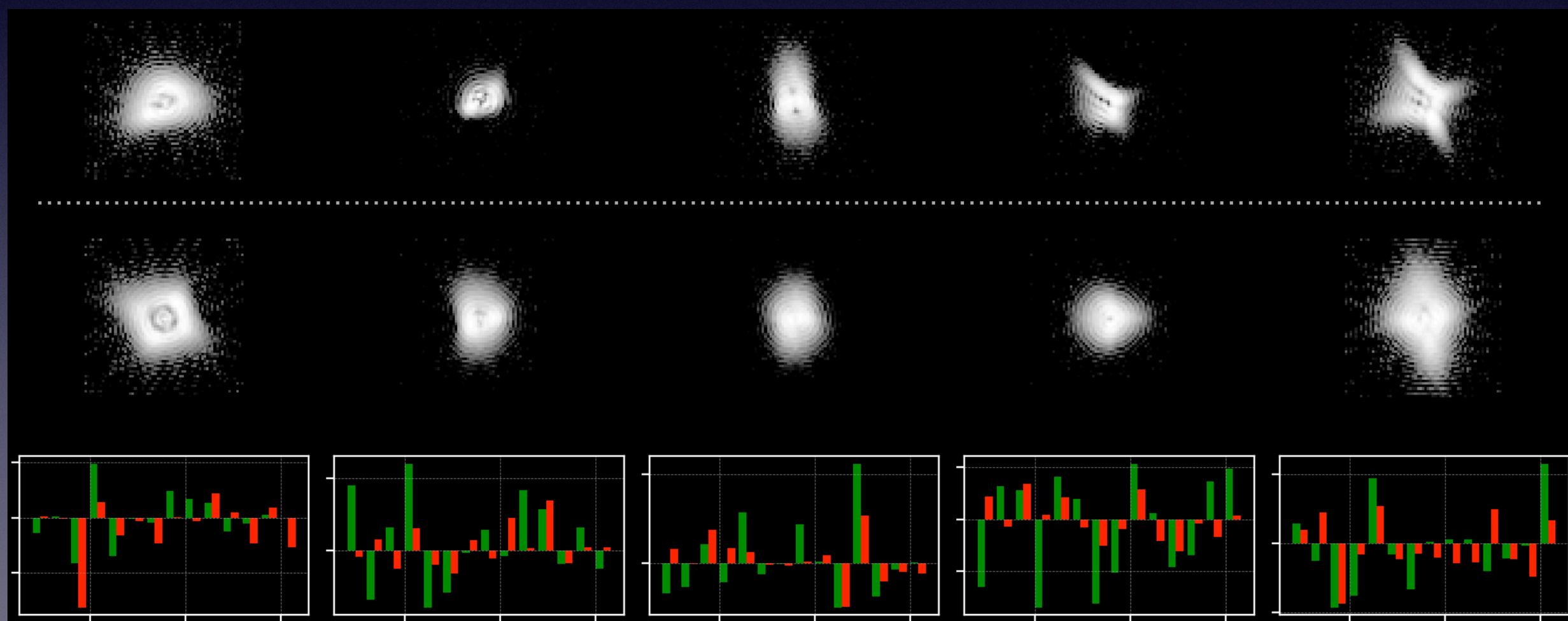
dropout	activate-1	activate-2	init-1	init-2
0.059	relu	tanh	Xavier	Lecun

Results

<u>Multi-Layer Perceptron</u>:

Mean RMS Error:

$\lambda/10$ = 205 nm ± 60 nm

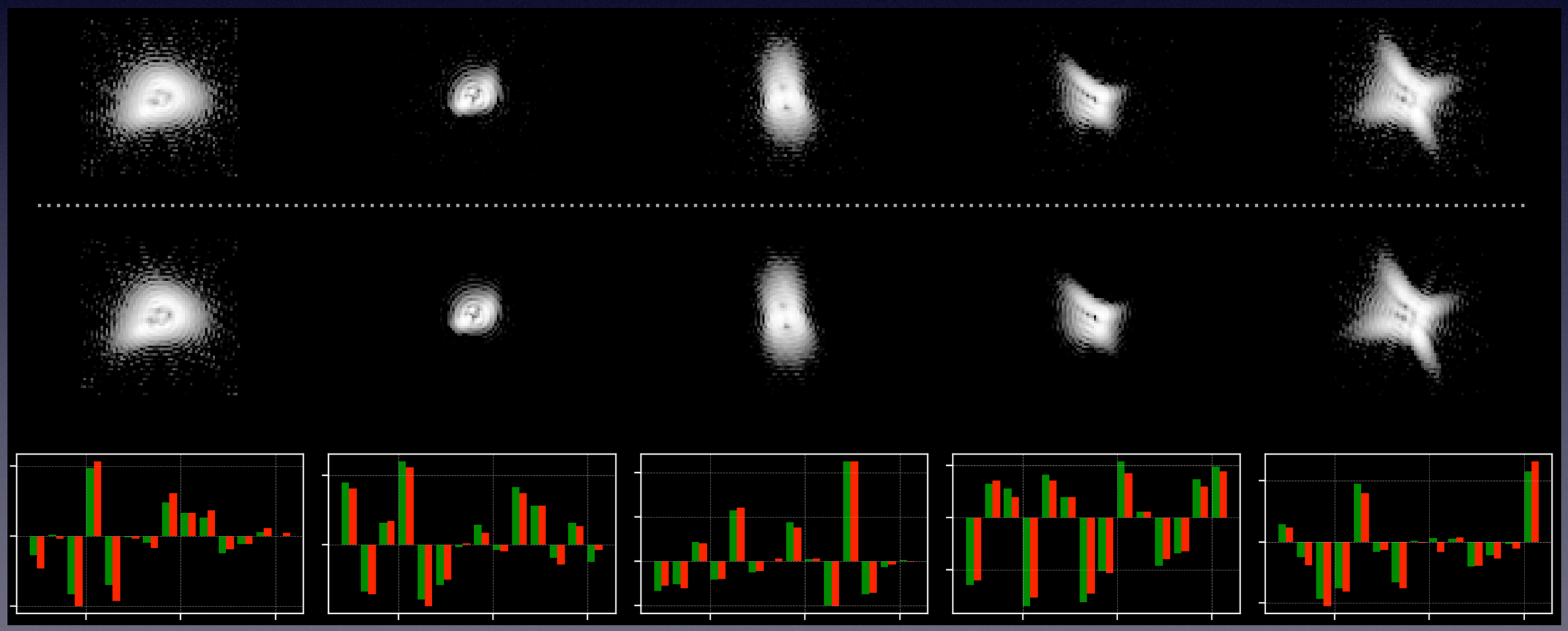


Results

Multi-Layer Perceptron (optimized):

Mean RMS Error:
$\lambda/48$ = 44 nm ± 25 nm

≈365%, ≈140%

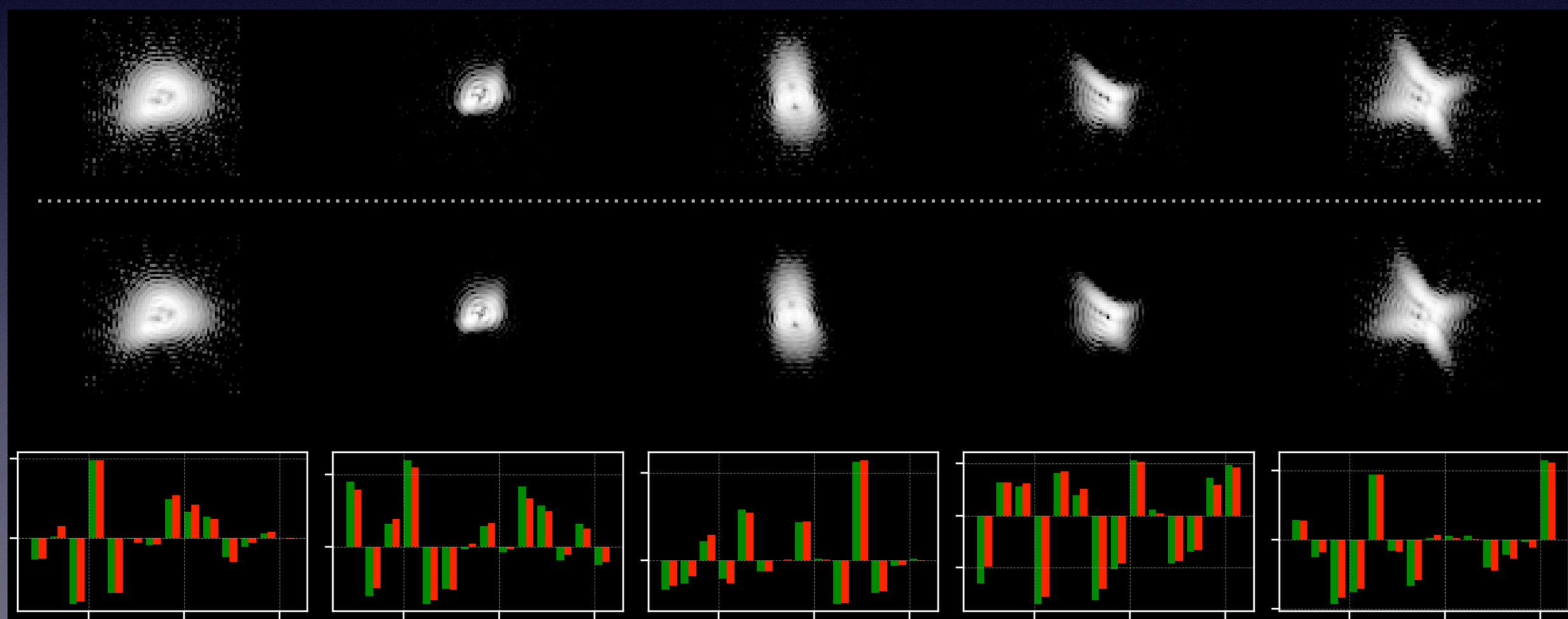


Results

Convolutional Neural Net:

Mean RMS Error:
$\lambda/64$ = 33 nm ± 16 nm

≈33%, ≈60%



Tuning is Important

MLP: Default

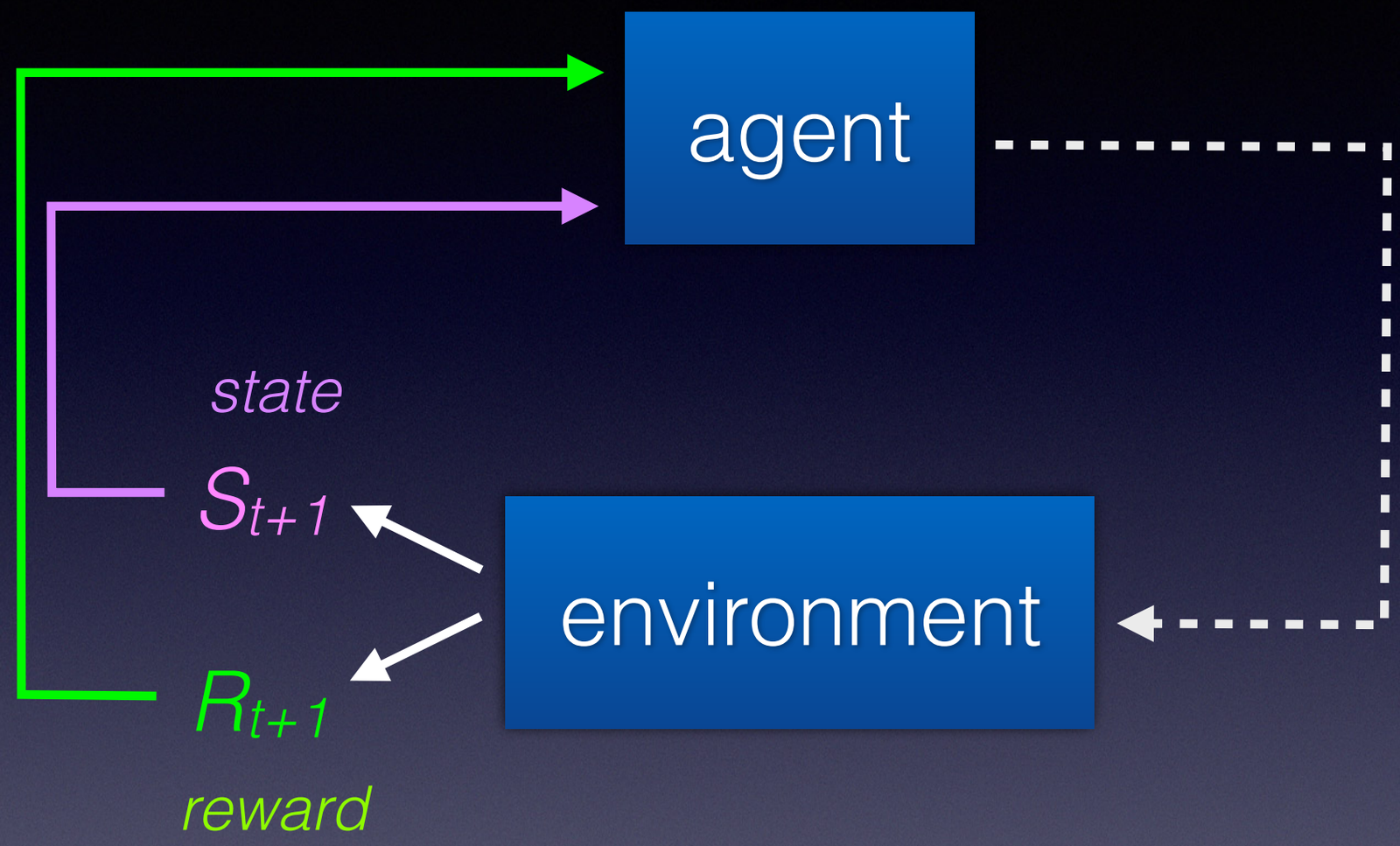
MLP: Optimized

Δ improved by ≈ 365%

CNN:

Δ improved by: ≈ 33%

Reinforcement Learning



When a “strategy” or competing resources might be important

Summary

- Direct Solve methods: overall more accurate
- ML can provide better starting points, robust against noise
- ML can be Fast
- In our application: <u>Unlimited training data</u>
- ML can solve for multiple params, same pipeline
- Hyperparameter tuning is essential