

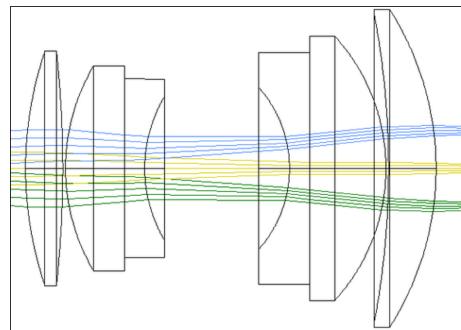
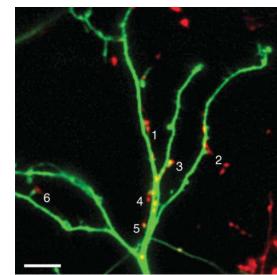
Lecture 19: Physical Layer Implementations and Troubleshooting

Machine Learning and Imaging

BME 590L
Roarke Horstmeyer

Summary of two models for image formation

- Interpretation #1: Radiation (*Incoherent*)
- Model: Rays

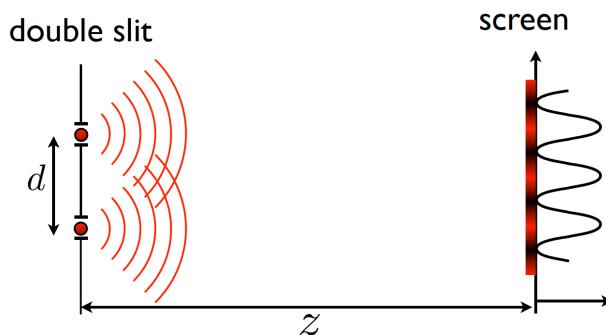
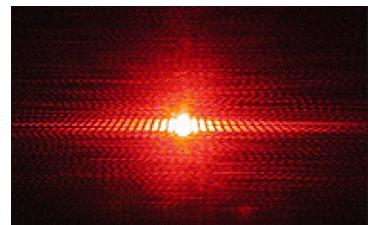


- Real, non-negative

$$I_s = H B S_0$$

- Sample absorption **S**
- Illumination brightness **B**
- Blur in **H**

- Interpretation #2: Electromagnetic wave (*Coherent*)
- Model: Waves



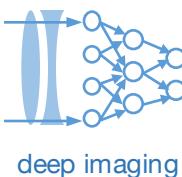
- Complex field

$$I_c = |H C S_c|^2$$

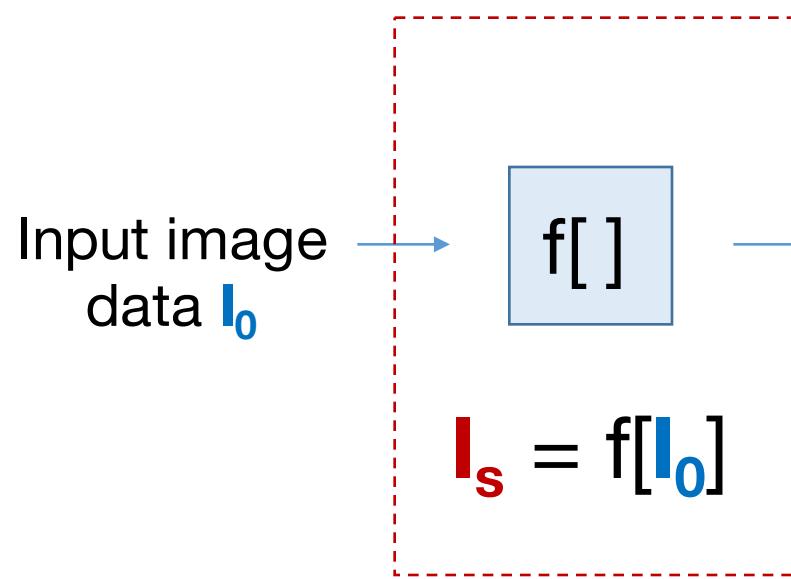
- Sample abs./phase **S**
- Illumination wave **B**
- Blur in **H**

Questions to address in this lecture

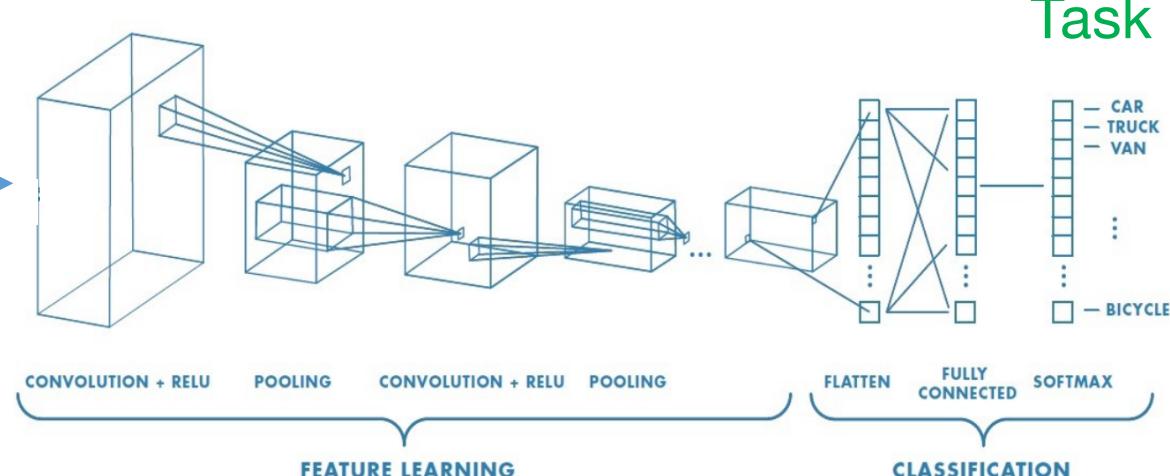
- Where and how should I implement my physical layer?
 - Simulation data
 - Experimental data
- How can I add some constraints to the physical weights that I'm optimizing?
- What are some common issues and pitfalls?



Physical Layers

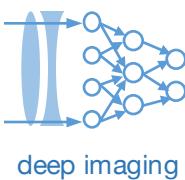


Digital Layers

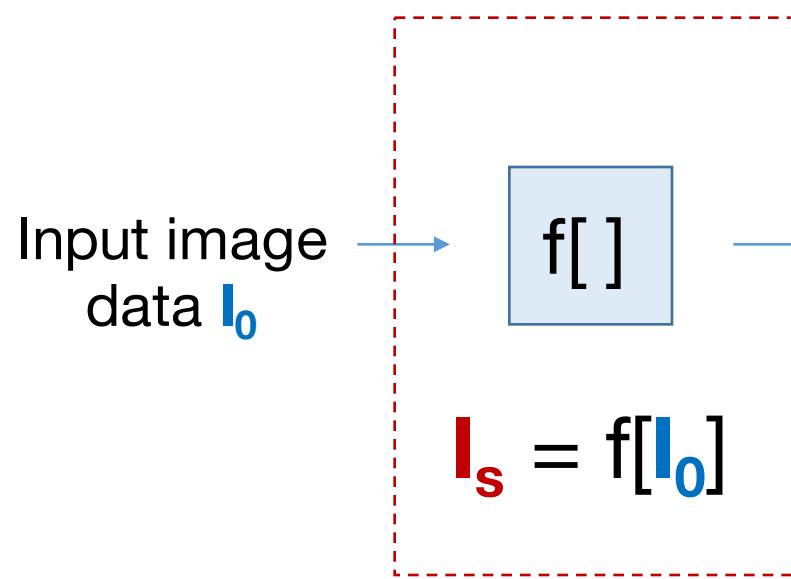


Task

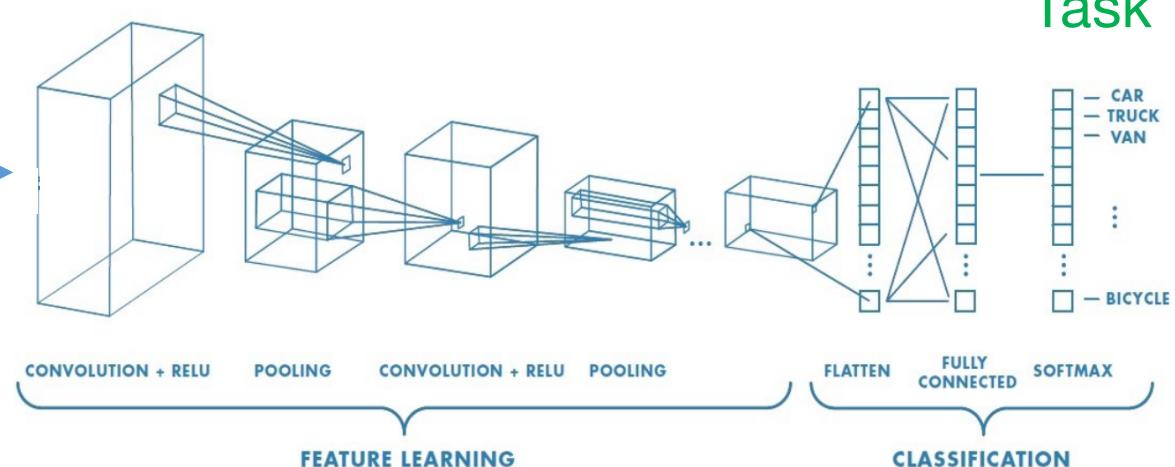
- Some Examples:
- Optimized illumination
 - Optimized sensor specifications
 - Number of measurements and locations
 - Radiation dosage, biomarkers



Physical Layers

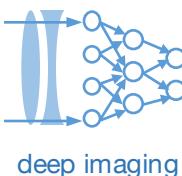


Digital Layers

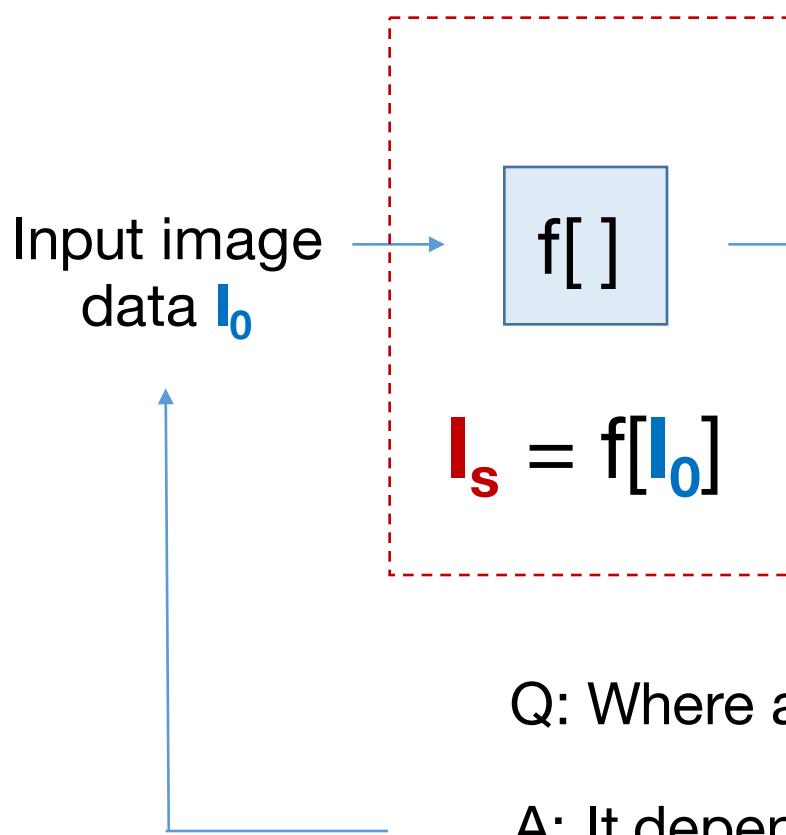


Task

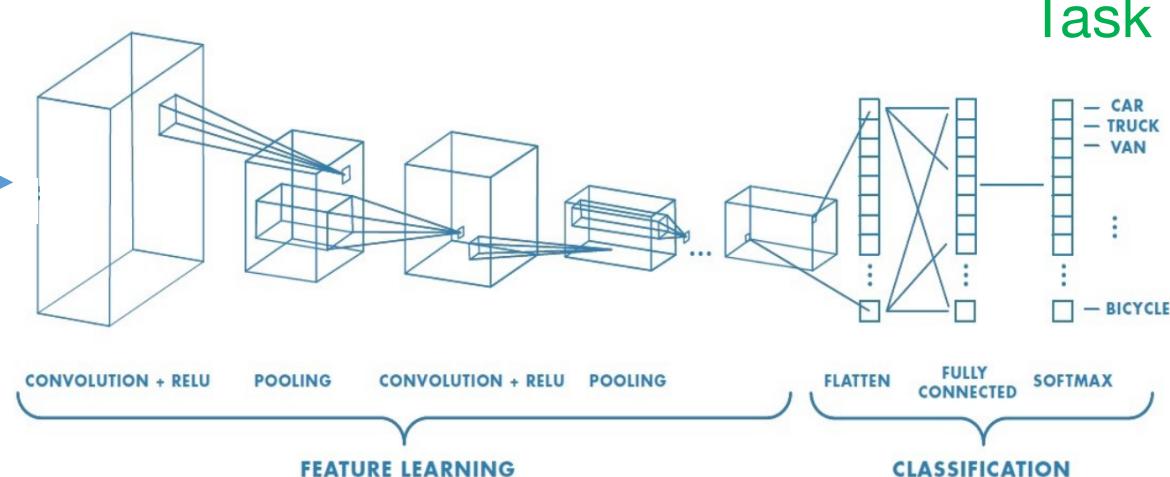
Q: Where and how should I implement my physical layer?



Physical Layers



Digital Layers

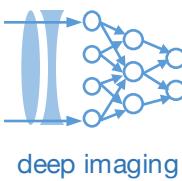


Task

Q: Where and how should I implement my physical layer?

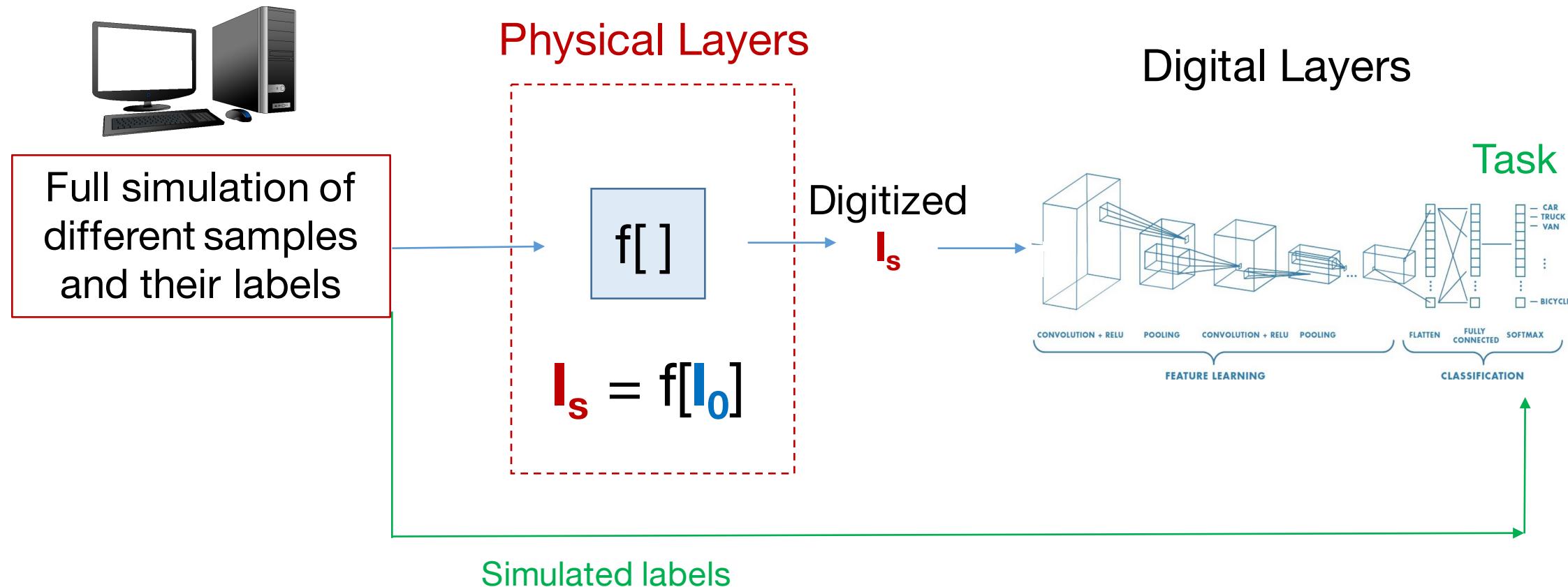
A: It depends on your data and implementation

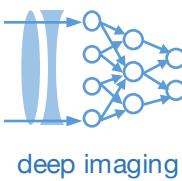
- Situation #1: Fully simulated physical layers
- Situation #2: Experimentally-driven physical layers



Situation #1: Fully simulated physical layers

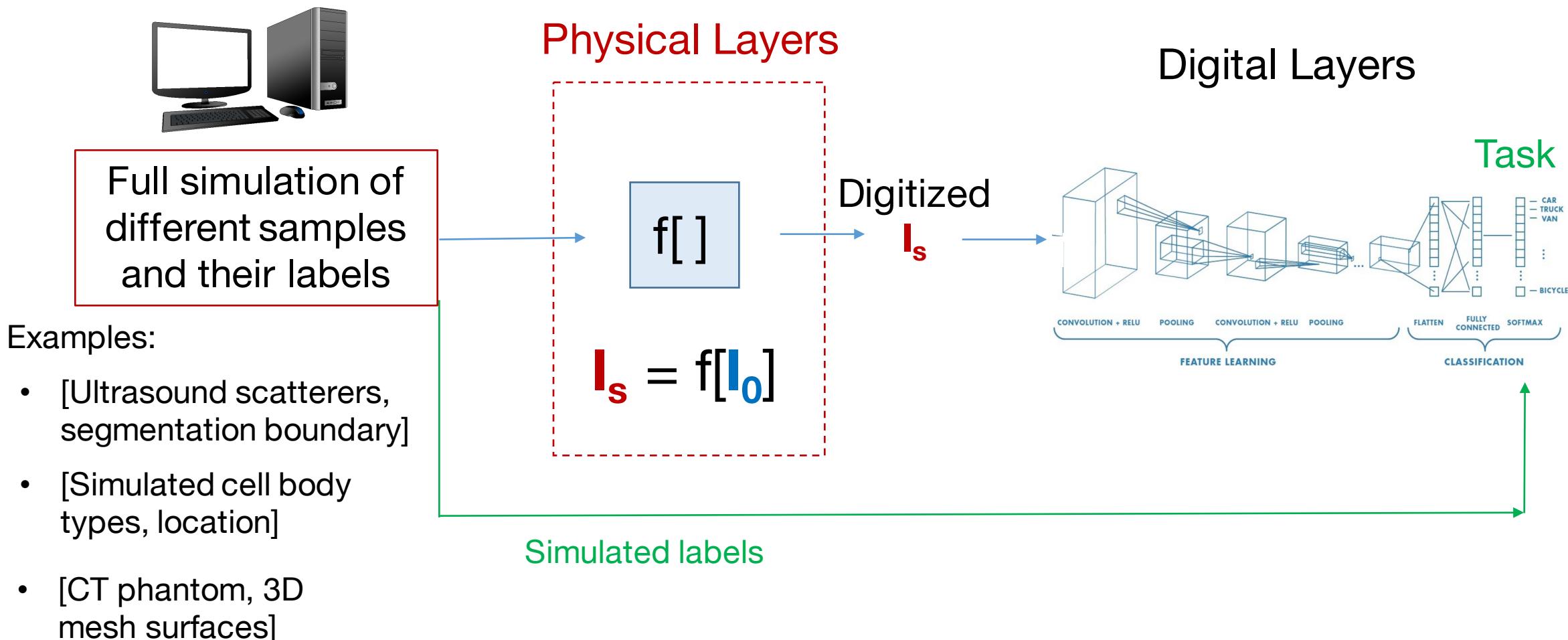
Option (a): Simulate the input images and the labels from scratch

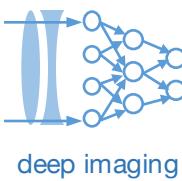




Situation #1: Fully simulated physical layers

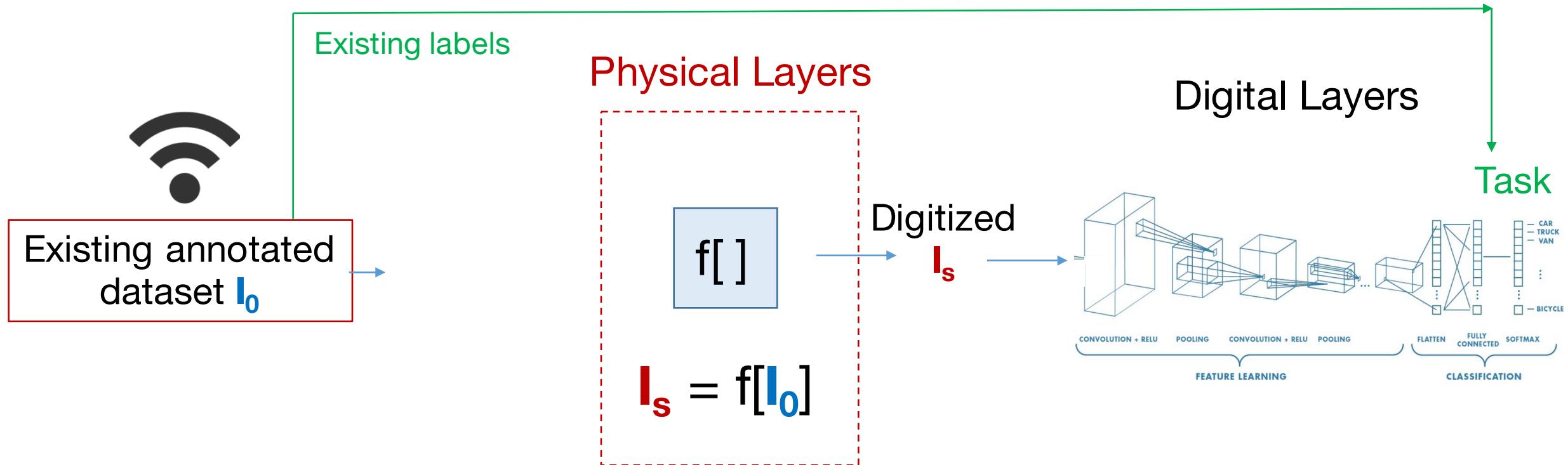
Option (a): Simulate the input images and the labels from scratch

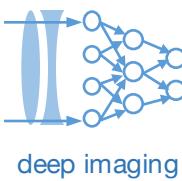




Situation #1: Fully simulated physical layers

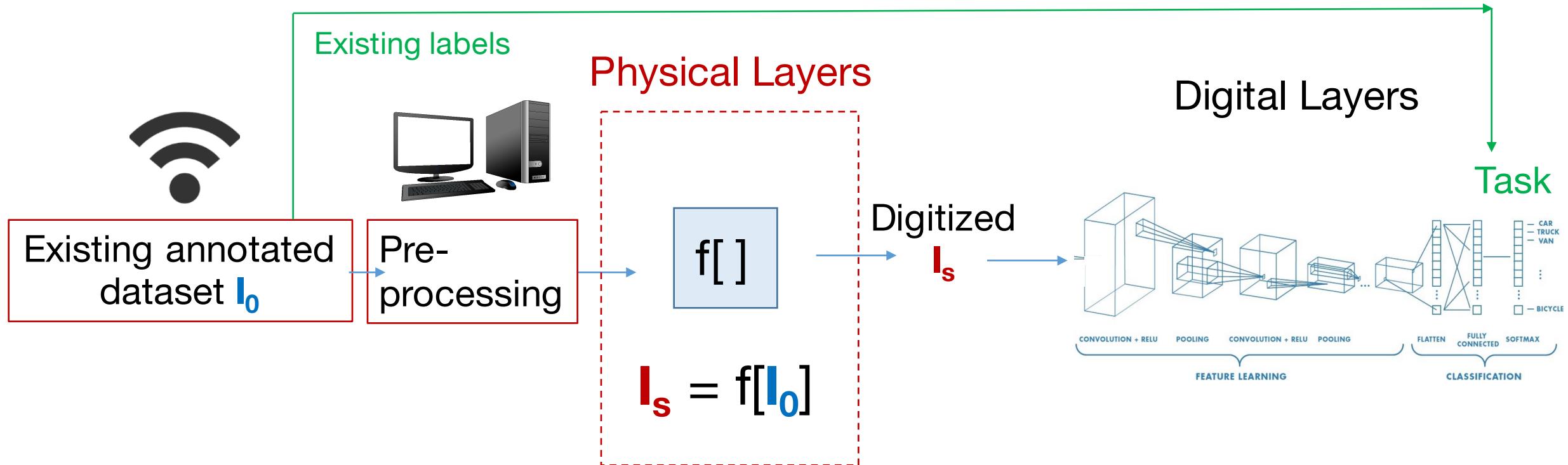
Option (b): Augment an existing dataset that you download

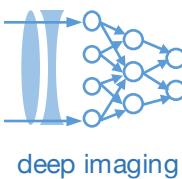




Situation #1: Fully simulated physical layers

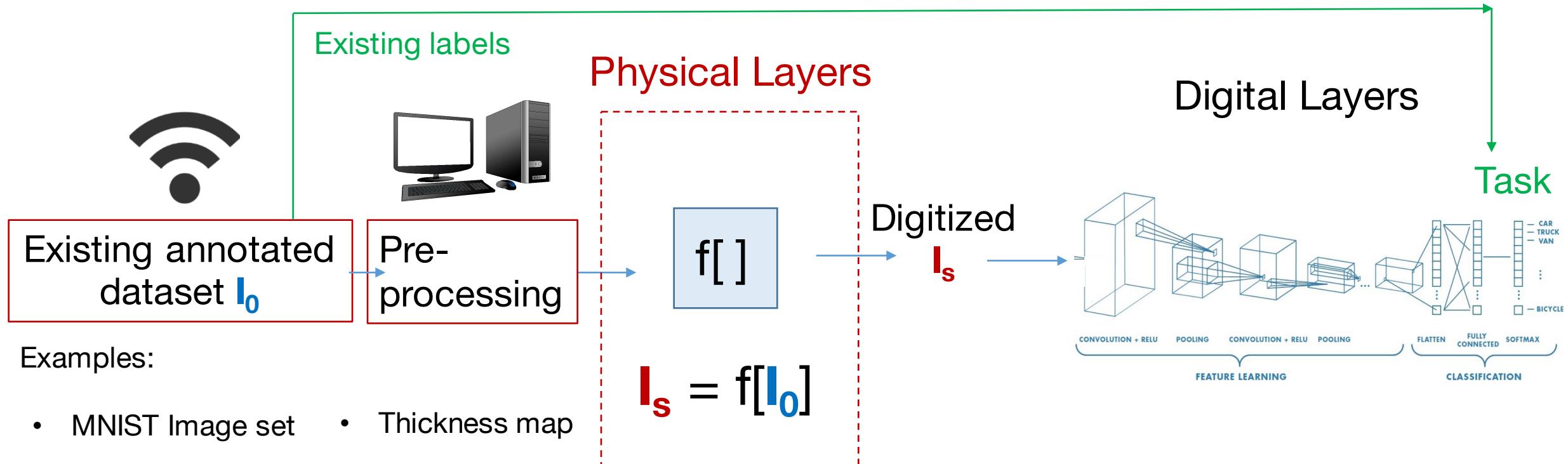
Option (b): Augment an existing dataset that you download





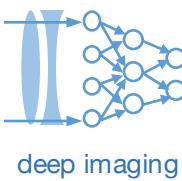
Situation #1: Fully simulated physical layers

Option (b): Augment an existing dataset that you download



Examples:

- MNIST Image set
- Segmented cells from Celltracker
- Segmented CT dataset from lab
- Thickness map
- Multispectral image stack
- Stitch together in a 3D composite



Situation #1: Fully simulated physical layers

Option (a) or Option (b): Choice on where and how to simulate/pre-process

**Simulation and/or
pre-processing**

Python/Matlab/other

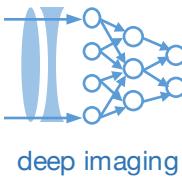


ML Optimization

Big dataset →  **TensorFlow**

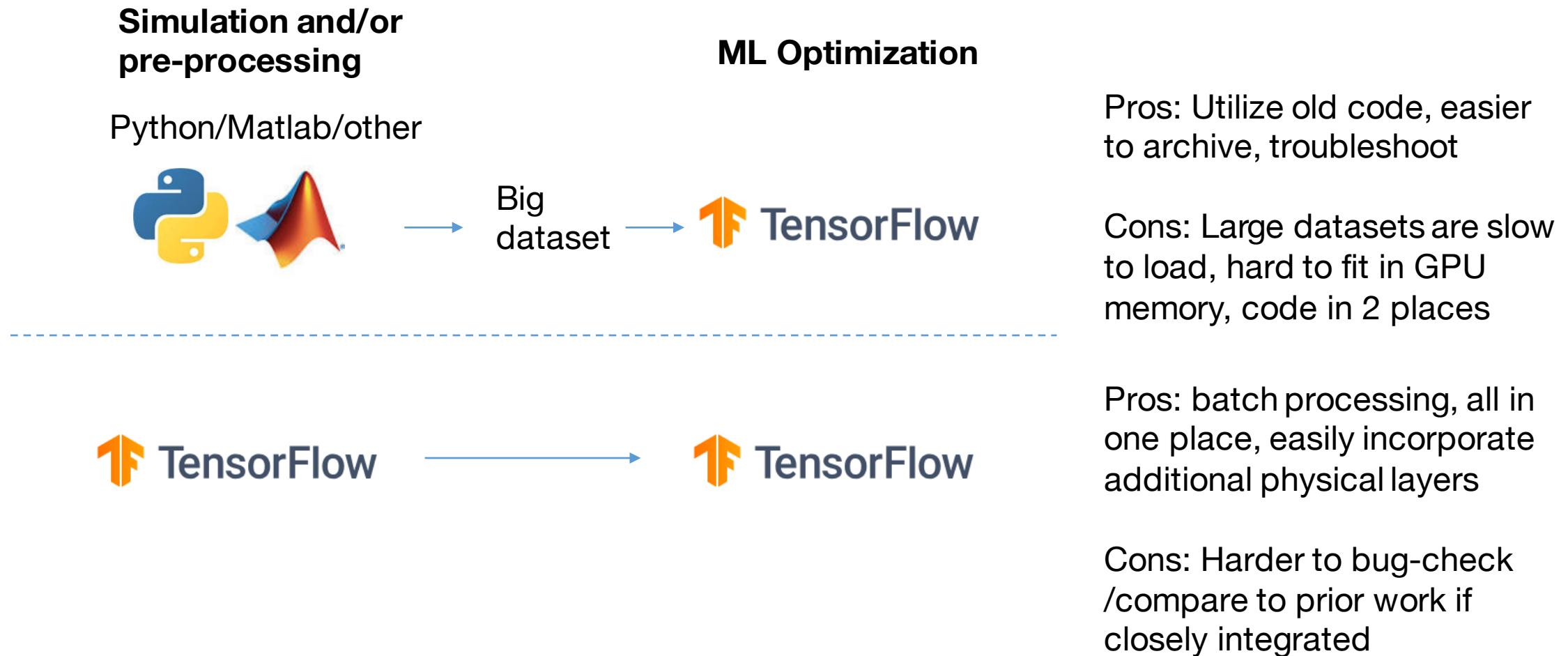
 **TensorFlow**

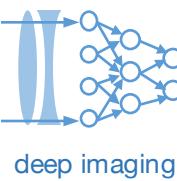
→  **TensorFlow**



Situation #1: Fully simulated physical layers

Option (a) or Option (b): Choice on where and how to simulate/pre-process

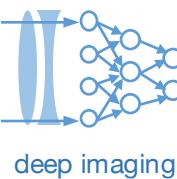




Aside on simulated data: Combining forward and inverse solvers

Forward problem: Start with the causes (objects in the real world) and compute the results (captured data)

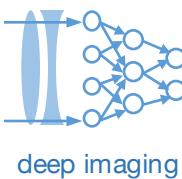
Inverse problem: Start with the results (captured data) and infer about the causes (objects in the real world)



Aside on simulated data: Combining forward and inverse solvers

Forward problem: Start with the causes (objects in the real world) and compute the results (captured data)
(Typically easy)

Inverse problem: Start with the results (captured data) and infer about the causes (objects in the real world)
(Typically hard)

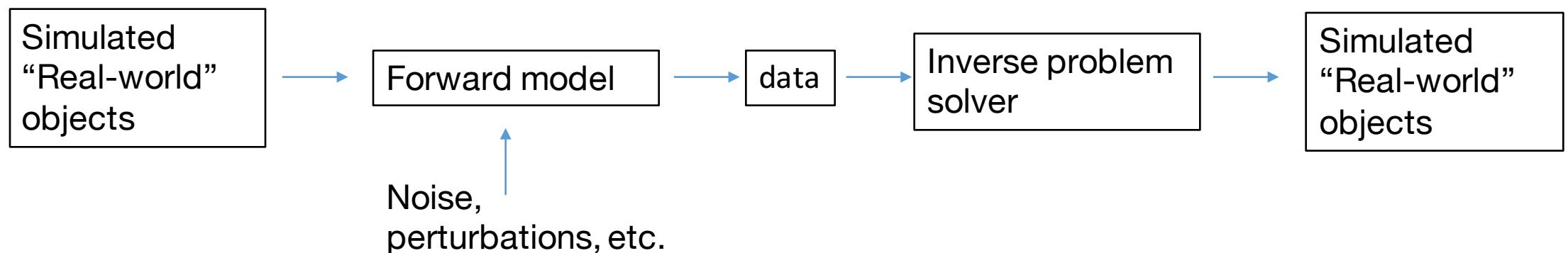


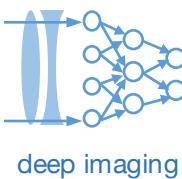
Aside on simulated data: Combining forward and inverse solvers

Forward problem: Start with the causes (objects in the real world) and compute the results (captured data)
(Typically easy)

Inverse problem: Start with the results (captured data) and infer about the causes (objects in the real world)
(Typically hard)

What I did in grad school to get ready for an experiment:



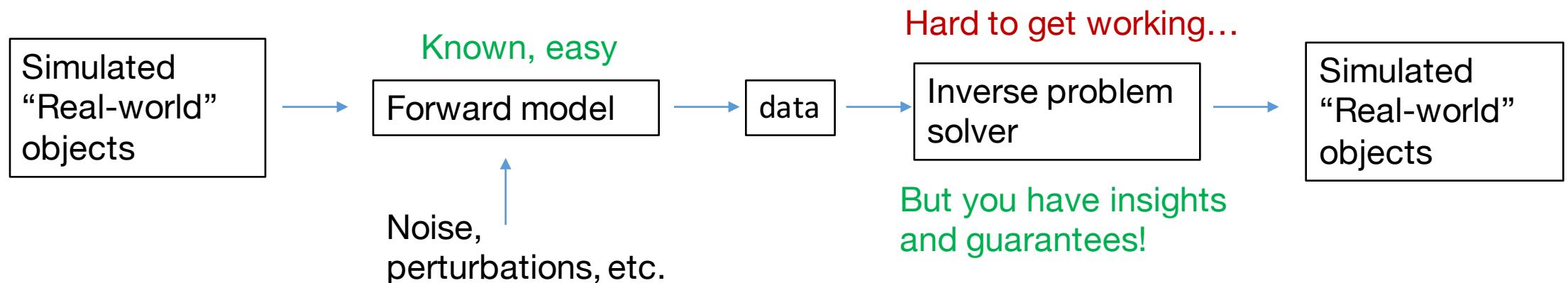


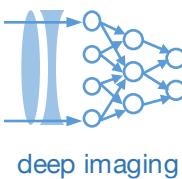
Aside on simulated data: Combining forward and inverse solvers

Forward problem: Start with the causes (objects in the real world) and compute the results (captured data)
(Typically easy)

Inverse problem: Start with the results (captured data) and infer about the causes (objects in the real world)
(Typically hard)

What I did in grad school to get ready for an experiment:



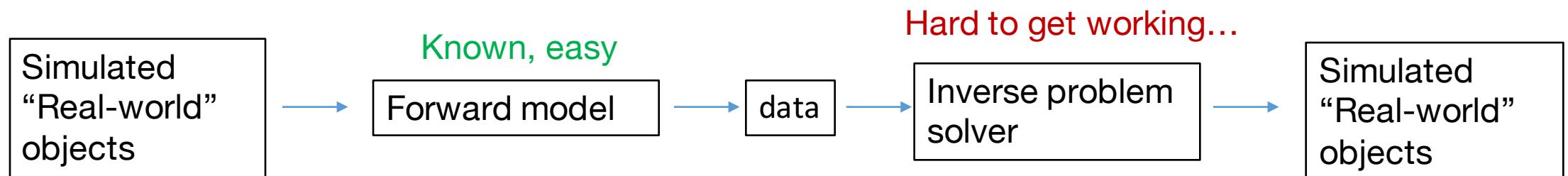


Aside on simulated data: Combining forward and inverse solvers

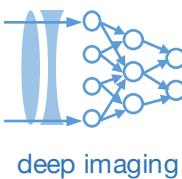
Forward problem: Start with the causes (objects in the real world) and compute the results (captured data)
(Typically easy)

Inverse problem: Start with the results (captured data) and infer about the causes (objects in the real world)
(Typically hard)

What I did in grad school to get ready for an experiment:



Classic examples: Inverse Radon Transform, US image reconstruction, image deblurring/denoising, diffraction tomography, phase retrieval, super-resolution (structured illumination, STORM/PALM), etc.

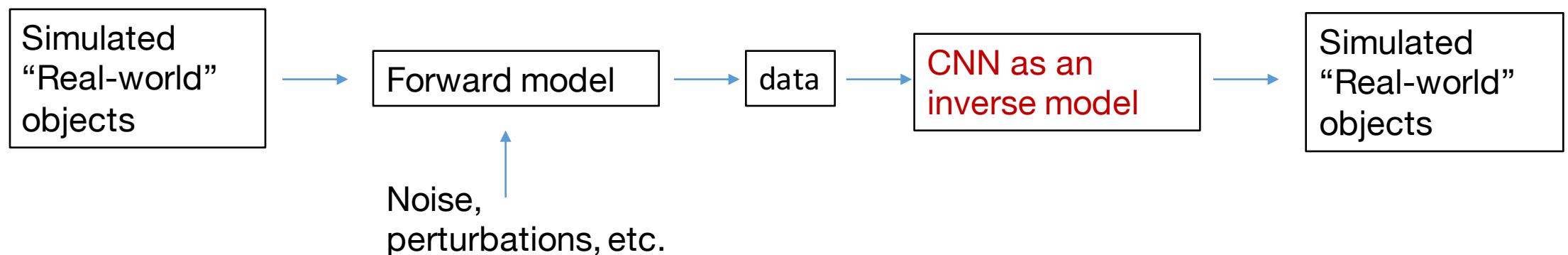


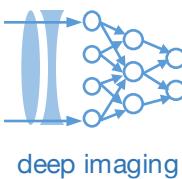
Aside on simulated data: Combining forward and inverse solvers

Forward problem: Start with the causes (objects in the real world) and compute the results (captured data)
(Typically easy)

Inverse problem: Start with the results (captured data) and infer about the causes (objects in the real world)
(Typically hard)

What you can do now with CNN's:



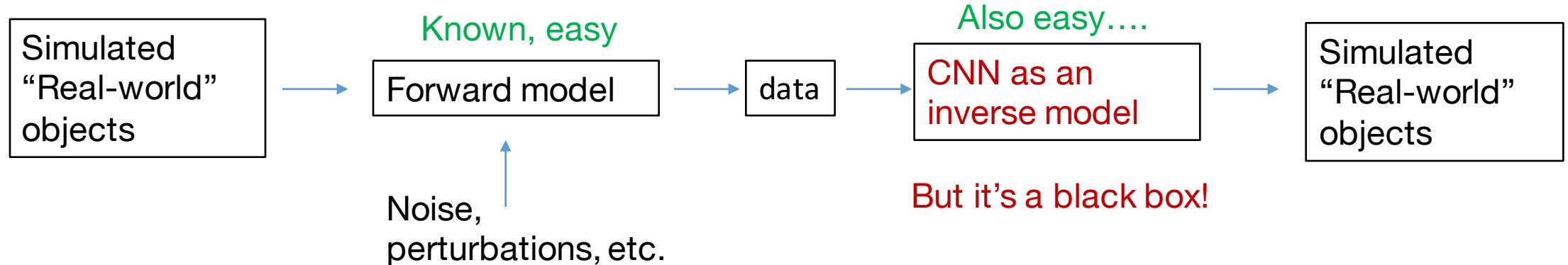


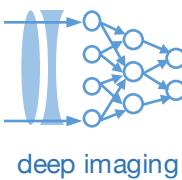
Aside on simulated data: Combining forward and inverse solvers

Forward problem: Start with the causes (objects in the real world) and compute the results (captured data)
(Typically easy)

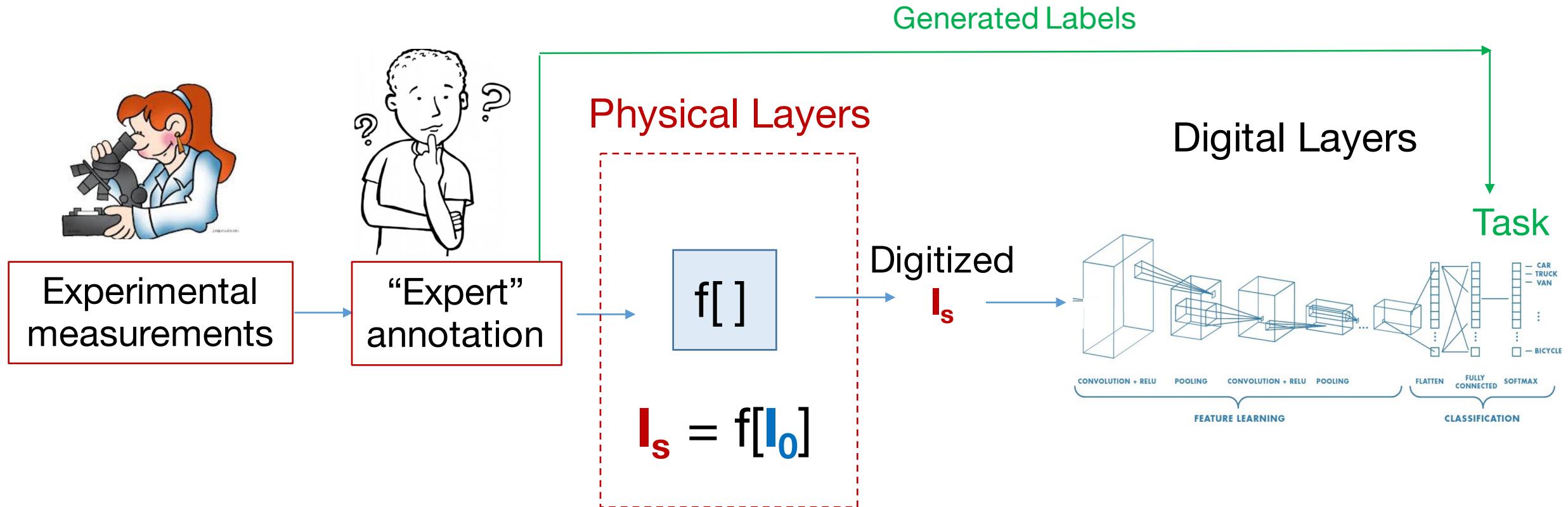
Inverse problem: Start with the results (captured data) and infer about the causes (objects in the real world)
(Typically hard)

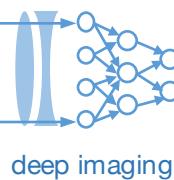
What you can do now with CNN's:





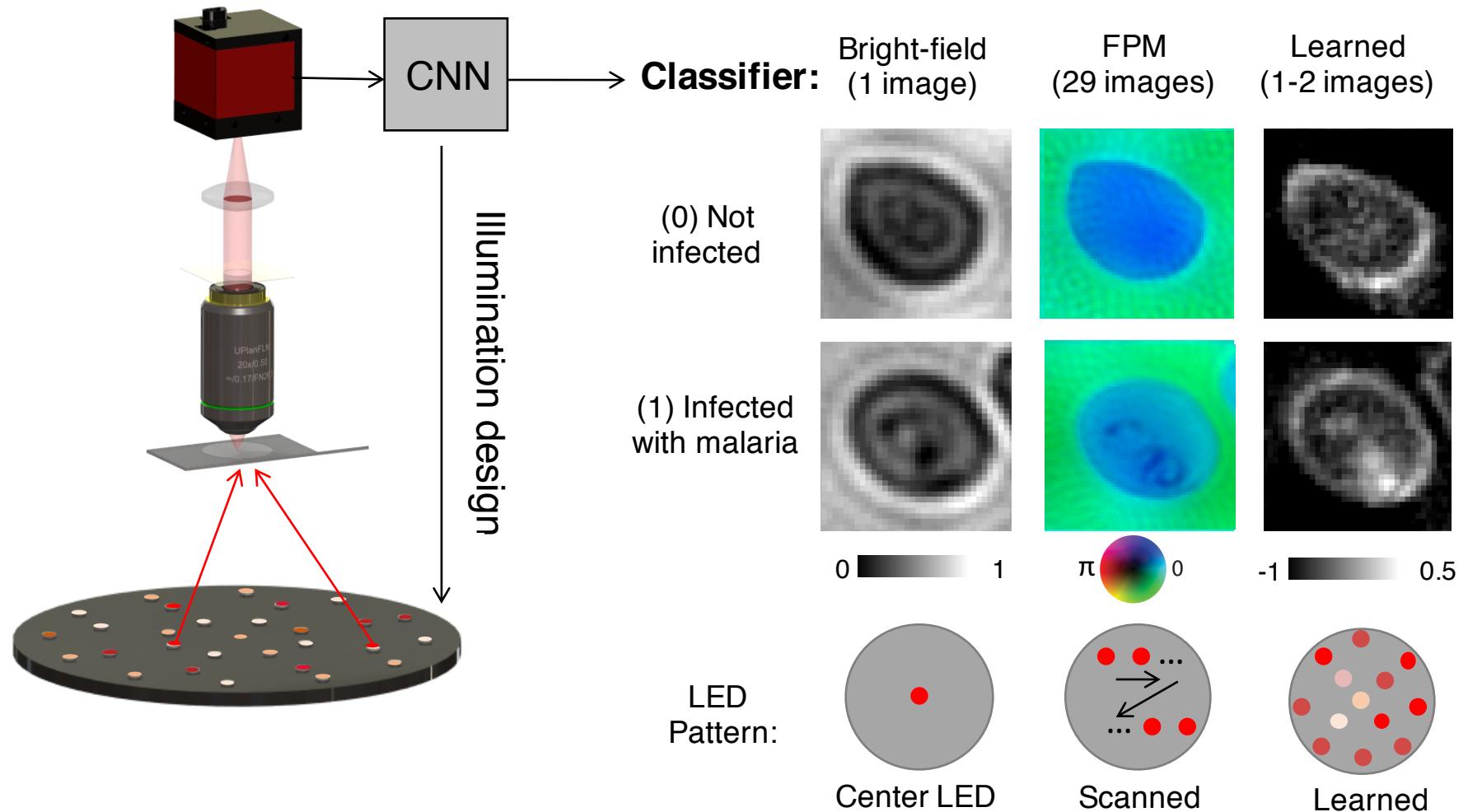
Situation #2: Experimentally-driven physical layers





Situation #2: Experimentally-driven physical layers

Example: CNN-Optimized illumination for classification of malaria:



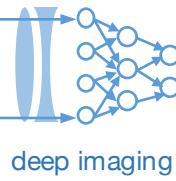
R. Horstmeyer et al., "Convolutional neural networks that teach microscopes how to image," arXiv (2017)

Classification Accuracy:

75%

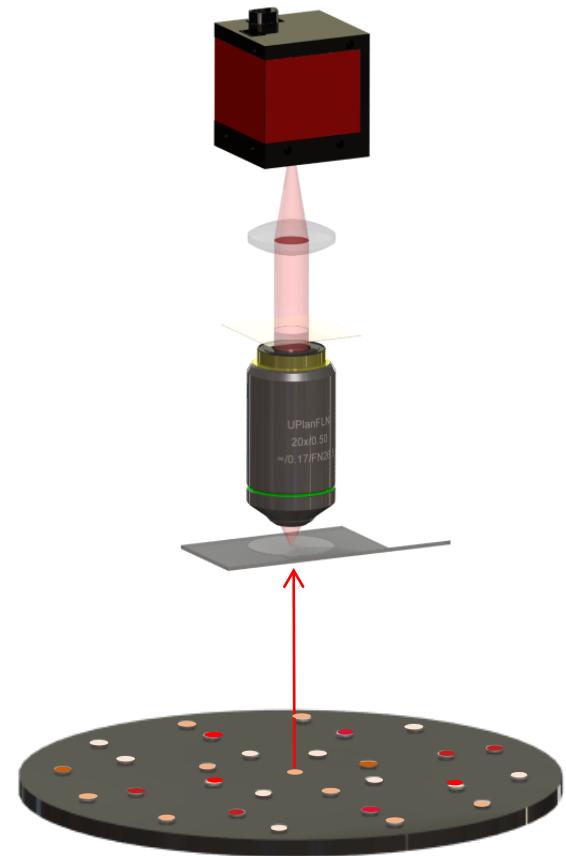
98.5%

97%



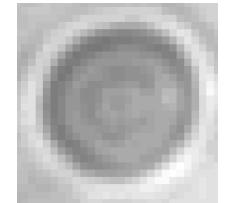
Situation #2: Experimentally-driven physical layers

Example: CNN-Optimized
illumination for
classification of malaria:

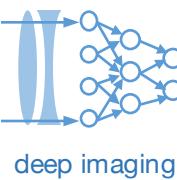


Data set for physical layer optimization:

Turn on LED 1, capture image 1:

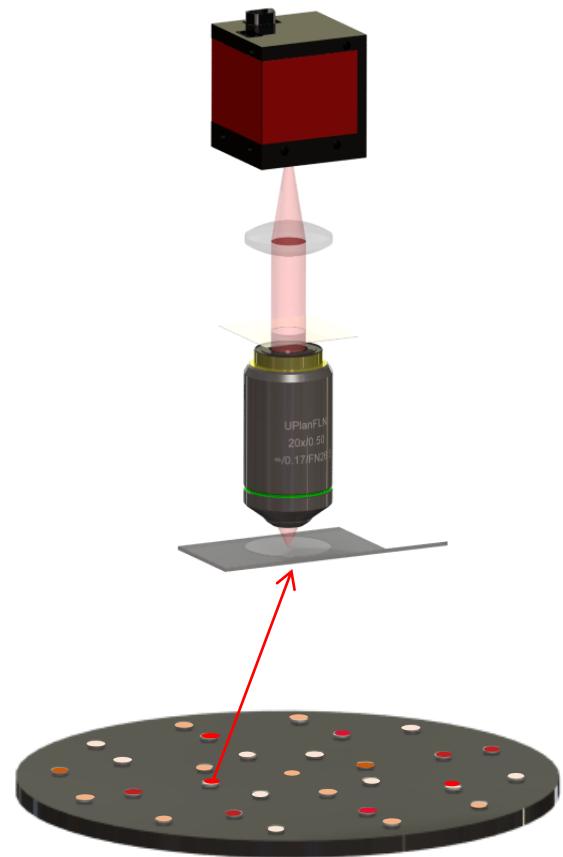


R. Horstmeyer et al., "Convolutional neural networks that teach microscopes how to image," arXiv (2017)



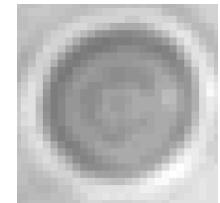
Situation #2: Experimentally-driven physical layers

Example: CNN-Optimized
illumination for
classification of malaria:

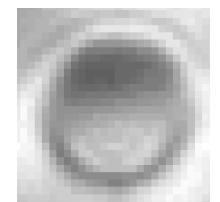


Data set for physical layer optimization:

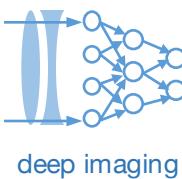
Turn on LED 1, capture image 1:



Turn on LED 1, capture image 2:

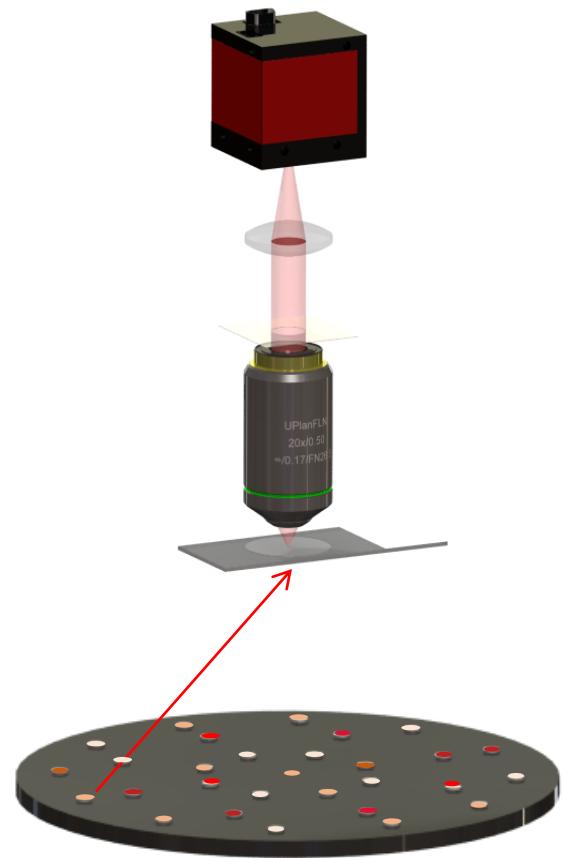


R. Horstmeyer et al., "Convolutional neural networks that teach microscopes how to image," arXiv (2017)



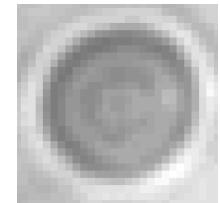
Situation #2: Experimentally-driven physical layers

Example: CNN-Optimized
illumination for
classification of malaria:

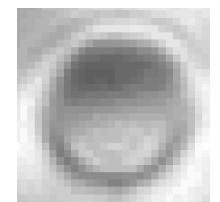


Data set for physical layer optimization:

Turn on LED 1, capture image 1:

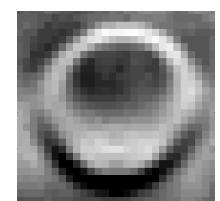


Turn on LED 1, capture image 2:



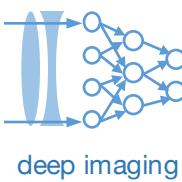
⋮

Turn on LED 32, capture image 32:



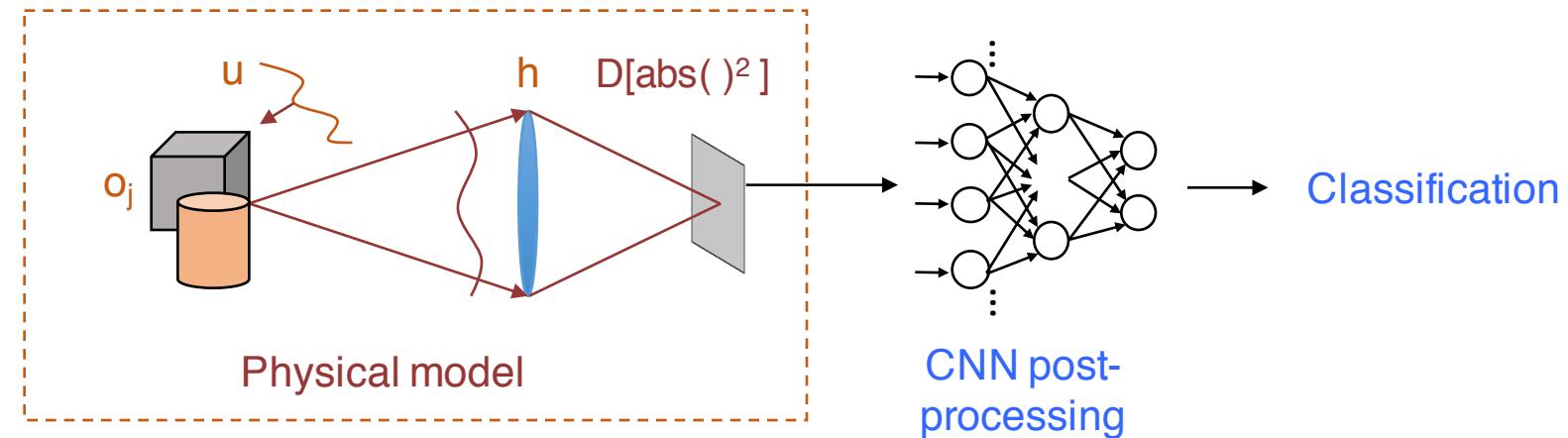
Save all 32 images (96 with 3 colors)

R. Horstmeyer et al., "Convolutional neural networks that teach microscopes how to image," arXiv (2017)



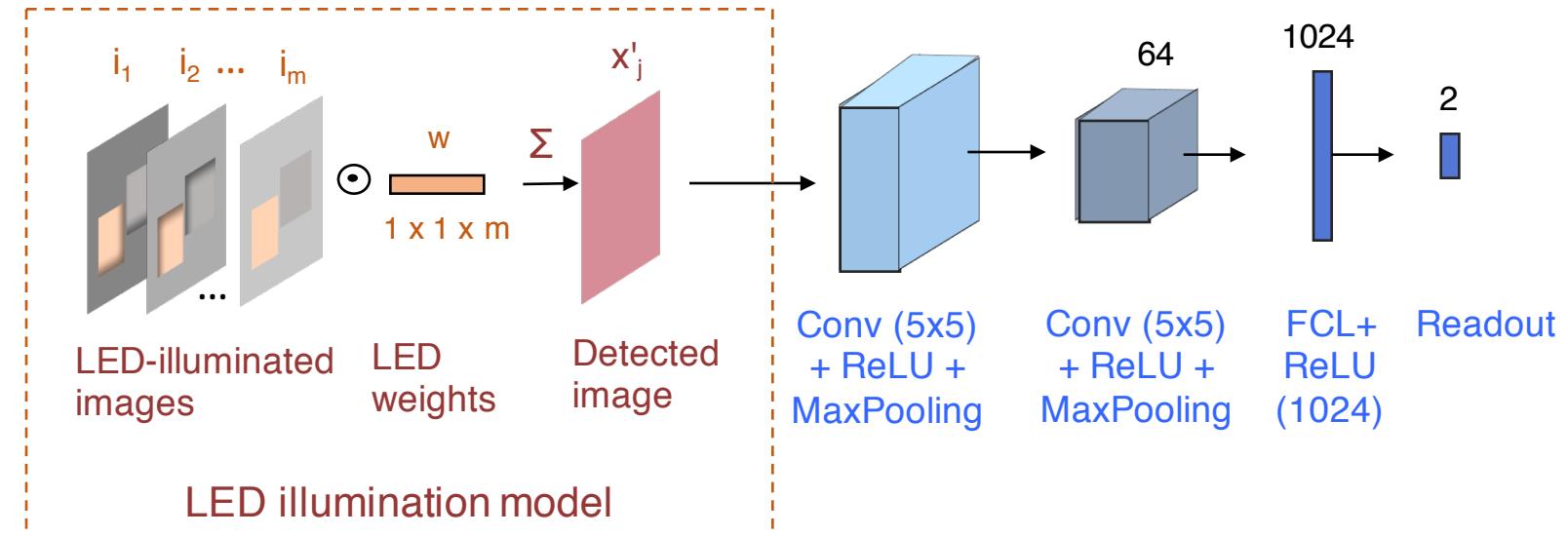
Situation #2: Experimentally-driven physical layers

Example: CNN-Optimized illumination for classification of malaria:

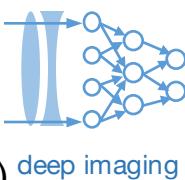


Physical layer:

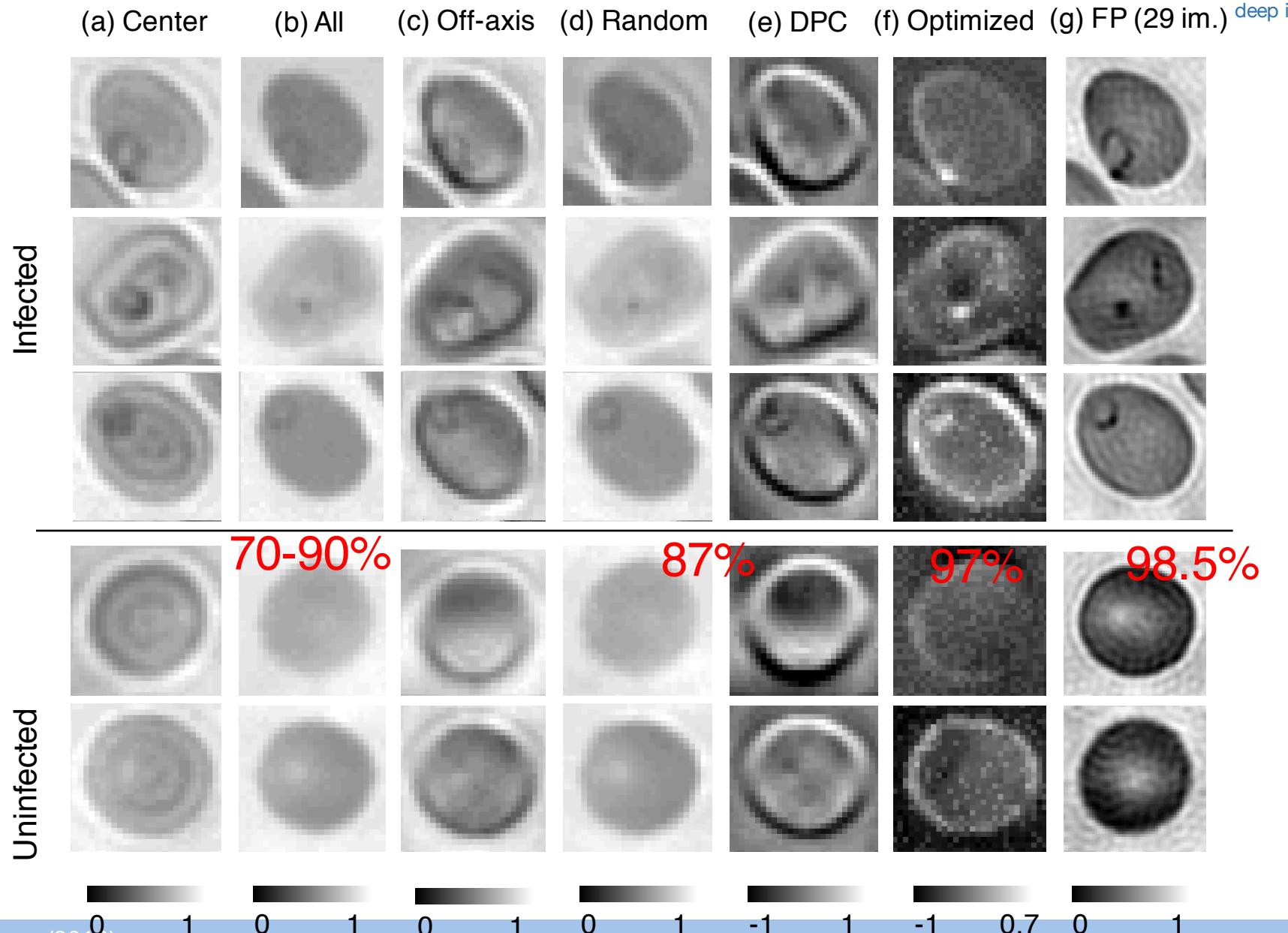
$$I_s = \sum w_j I_j$$

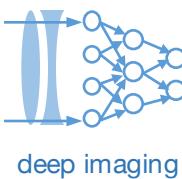


Situation #2: Experimentally-driven physical layers

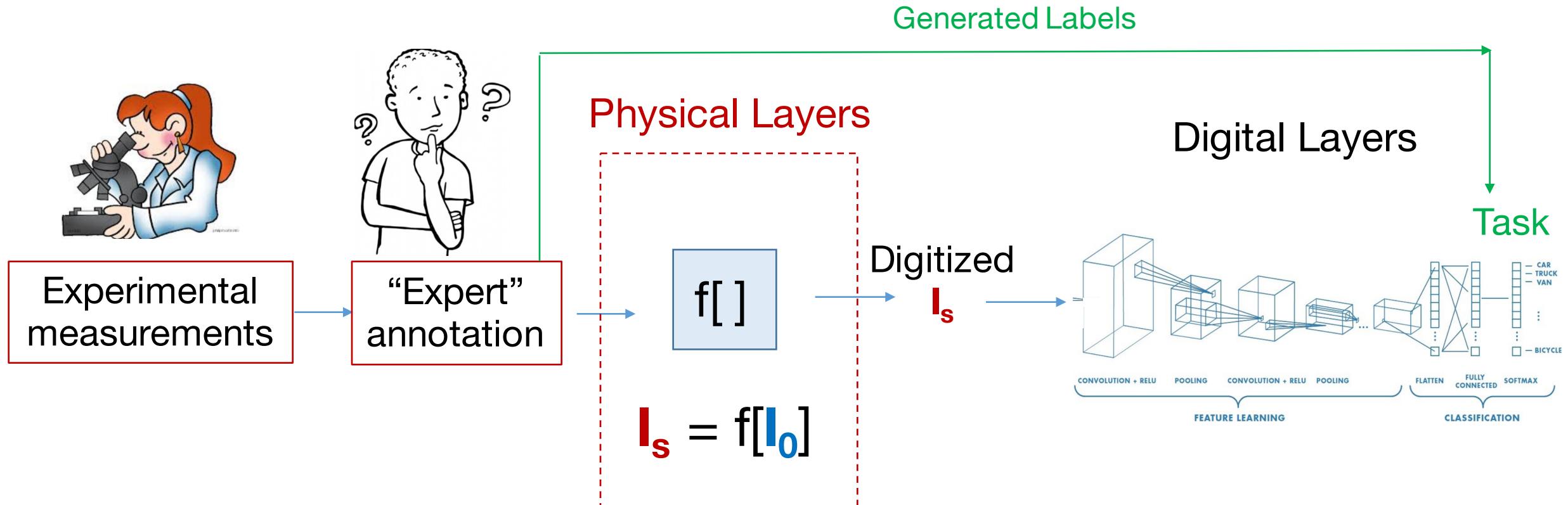


Example: CNN-Optimized
illumination for
classification of malaria:





Situation #2: Experimentally-driven physical layers



Pro's of experimental measurements: Don't need to worry about making your simulations match the setup! (HUGE WIN)

Con's of experimental measurements: You'll need to label them, limited access to desired sample information, often need to exploit some fundamental physical property

How can I add some constraints to my physical weights?

Without any constraints, weights can be any real (or complex) number. What if you physically can't realize any real or physical number?

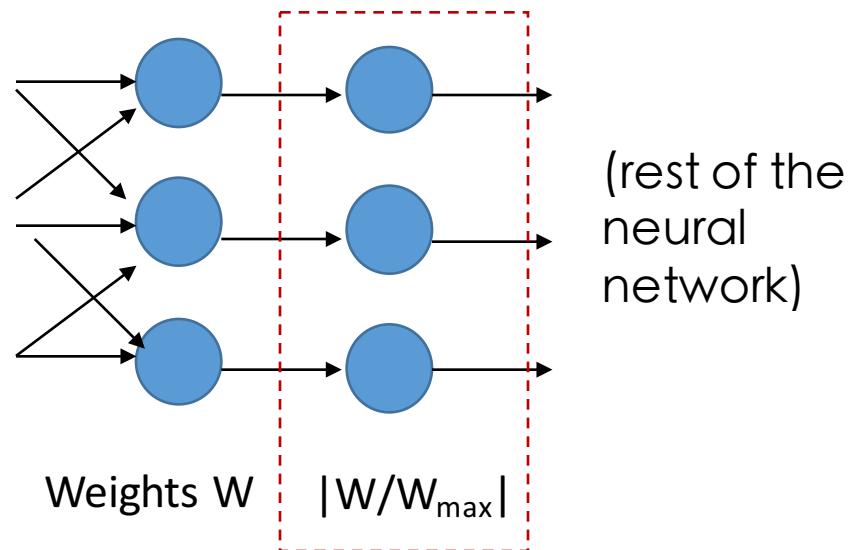
Example: Constrain weights to be non-negative values less than one

How can I add some constraints to my physical weights?

Without any constraints, weights can be any real (or complex) number. What if you physically can't realize any real or physical number?

Example: Constrain weights to be non-negative values less than one

Solution: add constraint as an extra “differentiable” layer (operation)

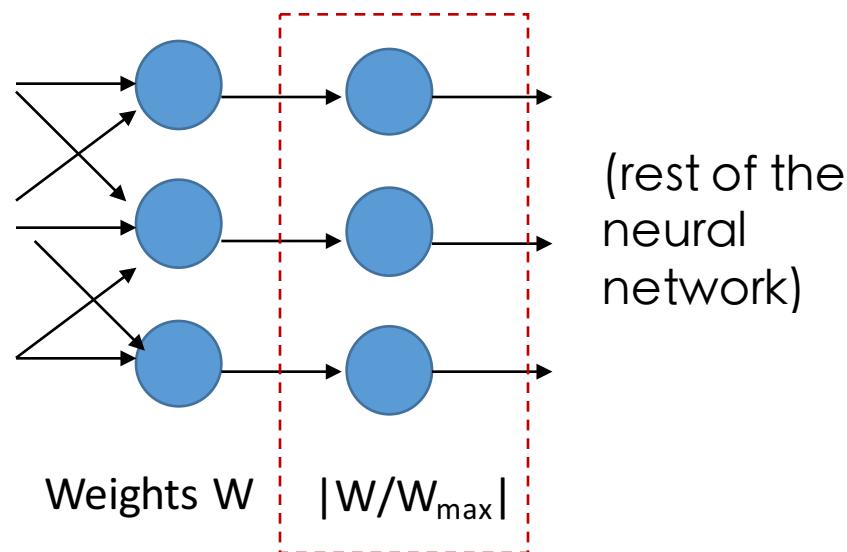


How can I add some constraints to my physical weights?

Without any constraints, weights can be any real (or complex) number. What if you physically can't realize any real or physical number?

Example: Constrain weights to be non-negative values less than one

Solution: add constraint as an extra “differentiable” layer (operation)



Pros:

- Easy to implement
- Constraints are obvious

Cons:

- Not always a well-behaved derivative

How can I add some constraints to my physical weights?

Without any constraints, weights can be any real (or complex) number. What if you physically can't realize any real or physical number?

Example: Constrain weights to be either 0 or 1

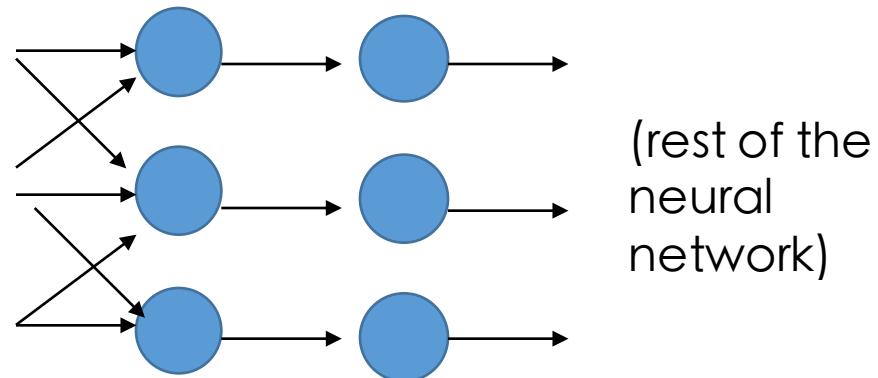
Solution: Perform annealing with a temperature parameter

How can I add some constraints to my physical weights?

Without any constraints, weights can be any real (or complex) number. What if you physically can't realize any real or physical number?

Example: Constrain weights to be either 0 or 1

Solution: Perform annealing with a temperature parameter



$$\text{Weights } w \quad I(n) = \text{Soft-max} [\alpha_t w(n)]$$

Increase α with iteration number

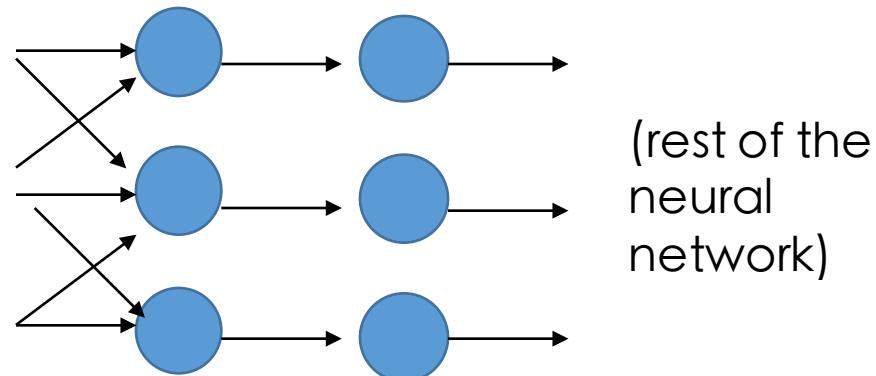
$$\text{Soft-max}(x) = \exp(-x) / \sum \exp(-x)$$

How can I add some constraints to my physical weights?

Without any constraints, weights can be any real (or complex) number. What if you physically can't realize any real or physical number?

Example: Constrain weights to be either 0 or 1

Solution: Perform annealing with a temperature parameter



Weights w $I(n) = \text{Soft-max} [\alpha_t w(n)]$

Increase α with iteration number

Pros:

- Works pretty well
- Flexibly address convergence issues

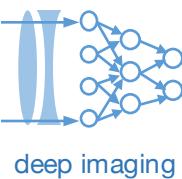
Cons:

- A bit sensitive

$$\text{Soft-max}(x) = \exp(-x) / \sum \exp(-x)$$

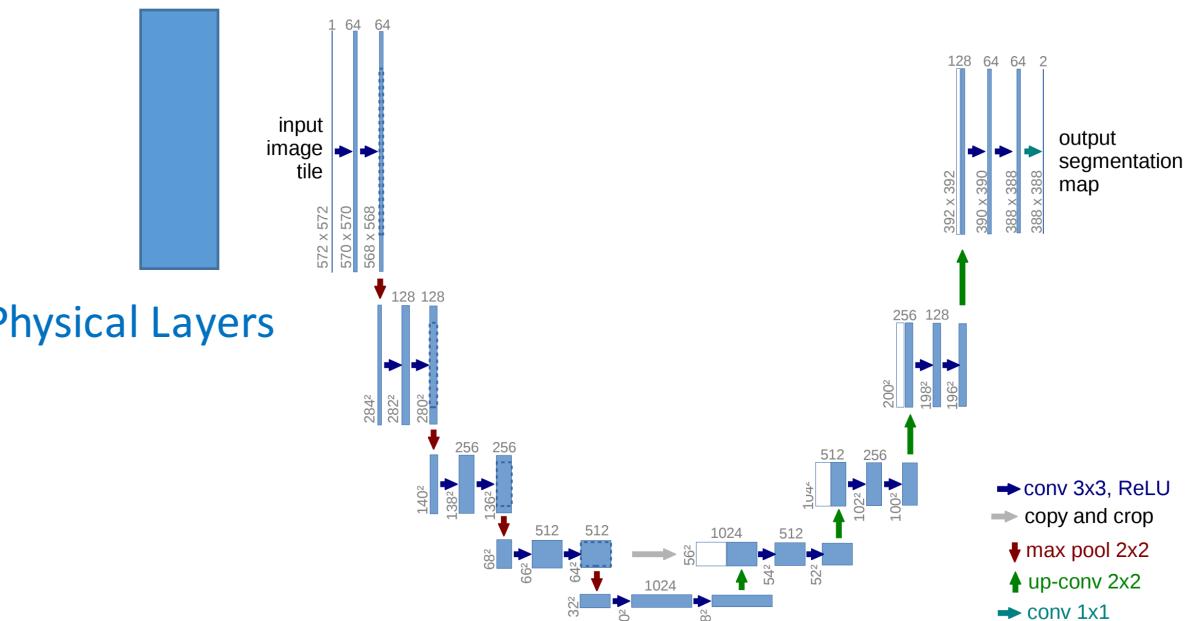
What are some common issues and pitfalls with physical layers?

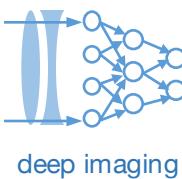
- Most common issue – you have a bug in your CNN!
 - Solution: “Disable” physical layer (set to constant), and get network to work!
 - Good practice: always compare performance with and without physical layer
- Another common challenge - vanishing gradients



What are some common issues and pitfalls with physical layers?

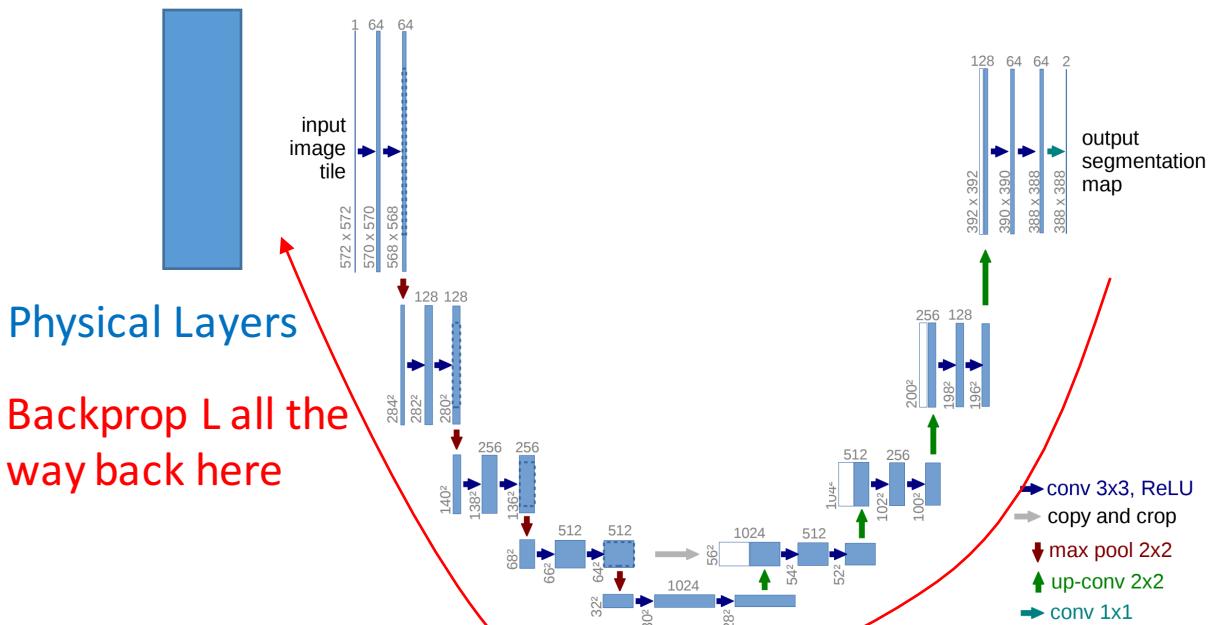
- Most common issue – you have a bug in your CNN!
 - Solution: “Disable” physical layer (set to constant), and get network to work!
- Another common challenge - vanishing gradients

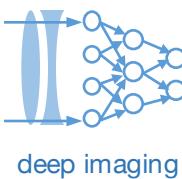




What are some common issues and pitfalls with physical layers?

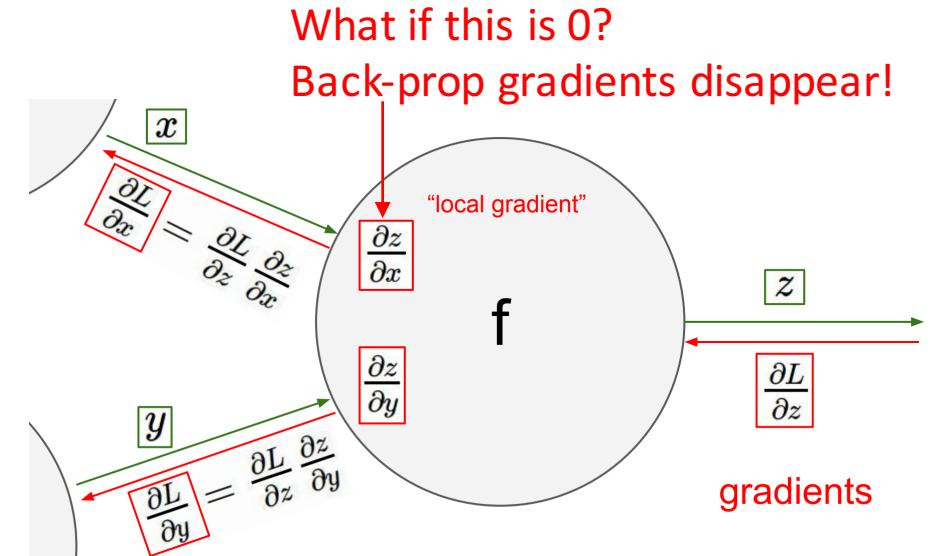
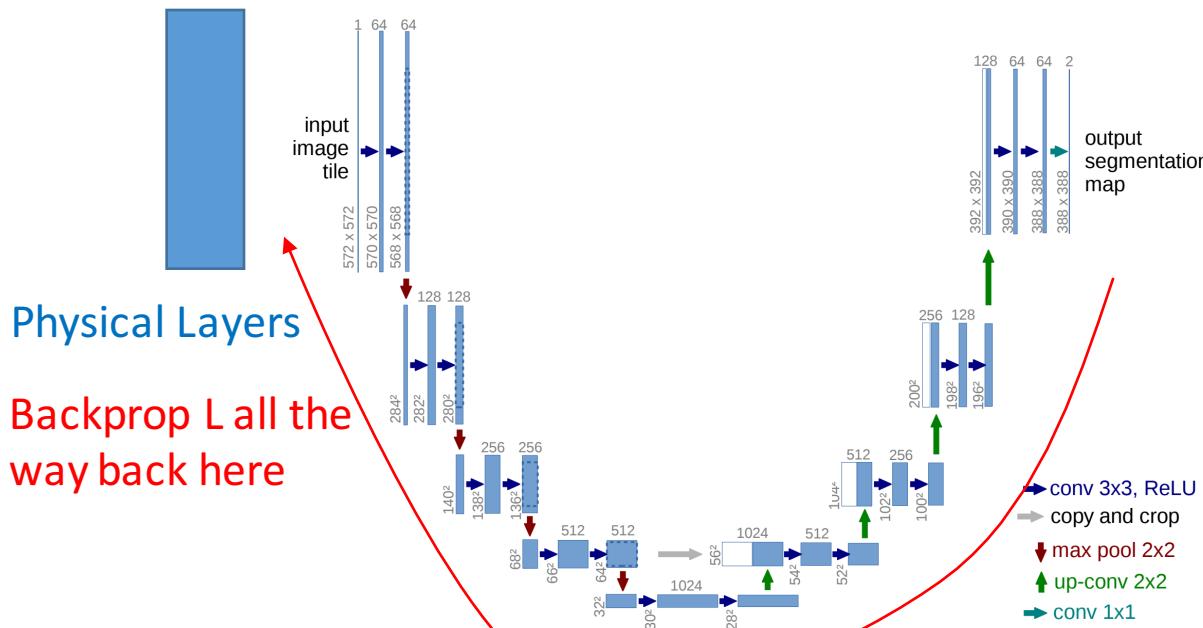
- Most common issue – you have a bug in your CNN!
 - Solution: “Disable” physical layer (set to constant), and get network to work!
- Another common challenge - vanishing gradients





What are some common issues and pitfalls with physical layers?

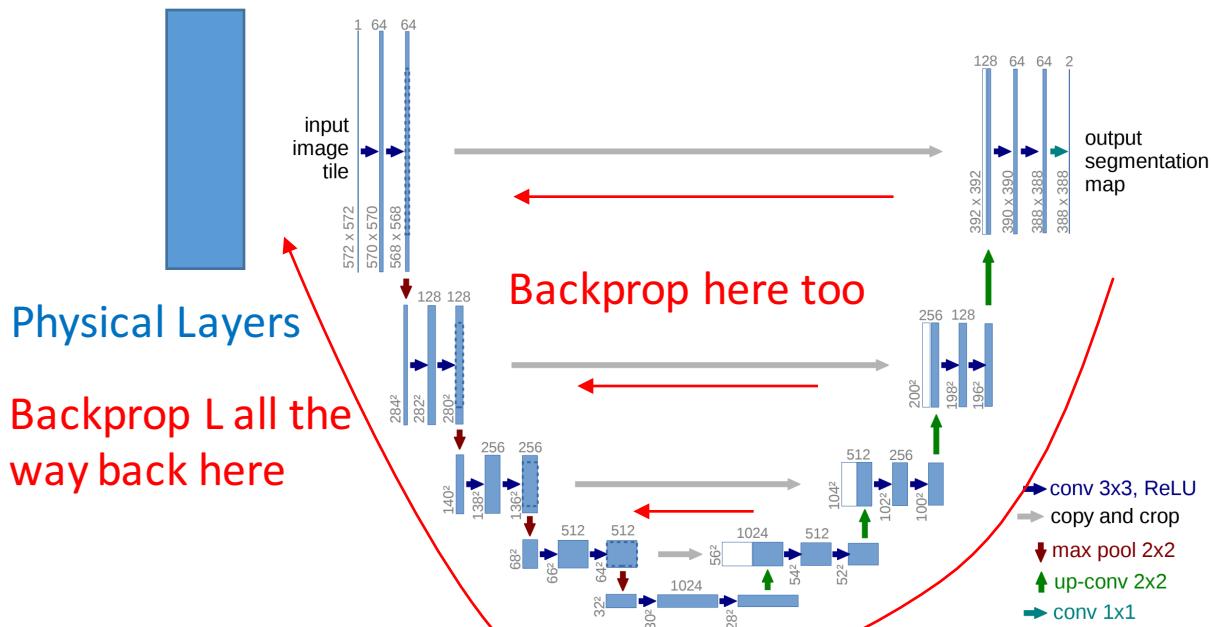
- Most common issue – you have a bug in your CNN!
 - Solution: “Disable” physical layer (set to constant), and get network to work!
- Another common challenge - vanishing gradients



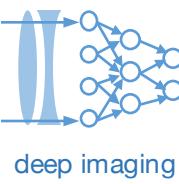
From Stanford CS231n

What are some common issues and pitfalls with physical layers?

- Most common issue – you have a bug in your CNN!
 - Solution: “Disable “ physical layer (set to constant), and get network to work!
- Another common challenge - vanishing gradients



Solution: Introduce skipped connections



Discussion group time!

1. Introduce your project (3 min)
2. Describe one (or more) potential challenges that you anticipate (2 min)
3. Discuss possible solutions to this challenge (3-5 min)
4. Switch!