Agenda

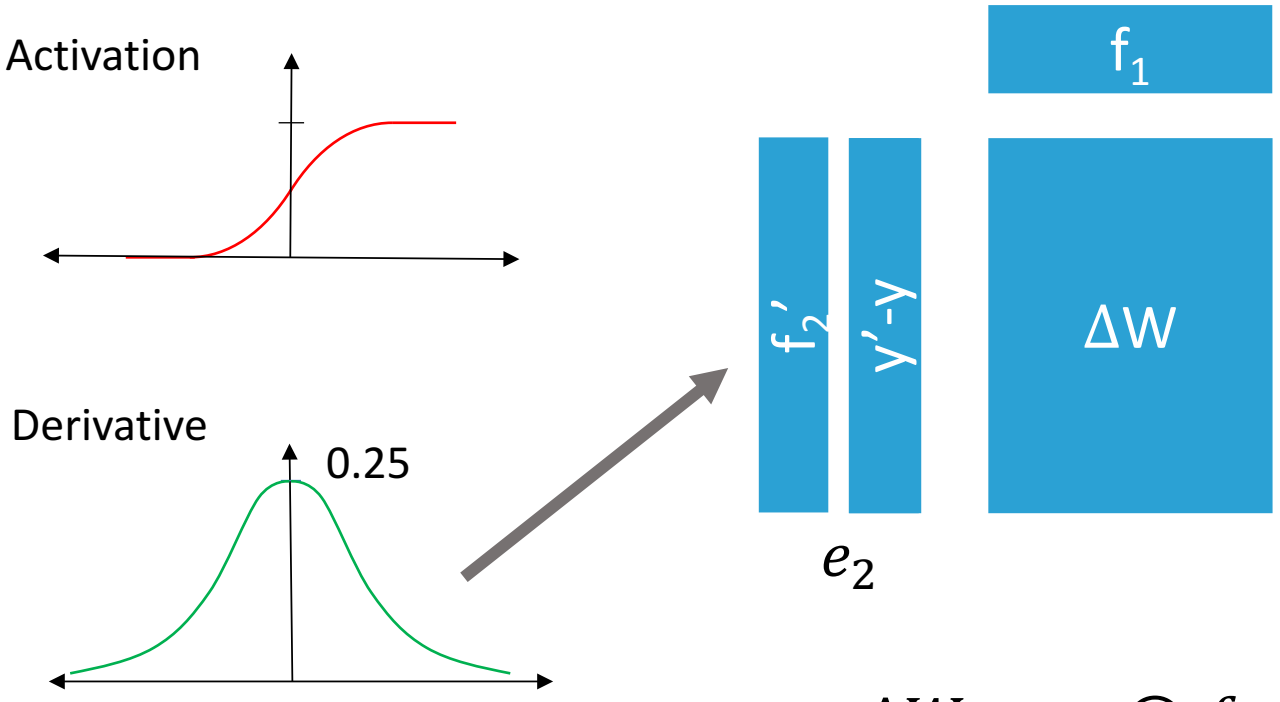
- History, continued
- Deep Learning
- TensorFlow

History of Neural Networks, Part 2

Three Problems

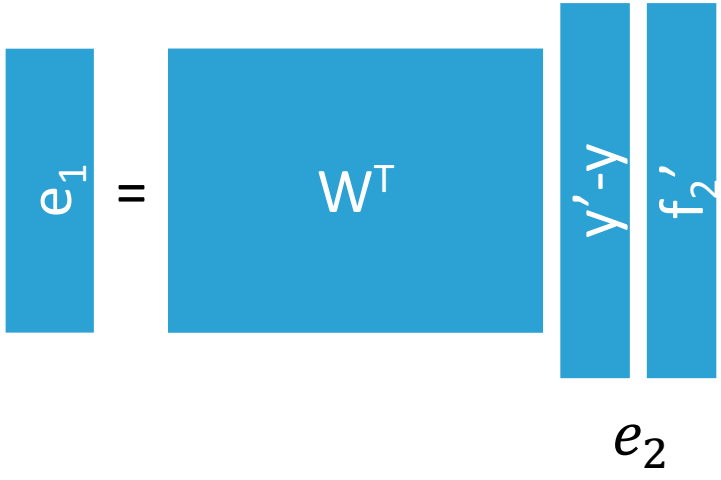
- Vanishing gradients in deep and recurrent networks
- High dimensional parameter spaces
- Internal representations

Vanishing Gradient



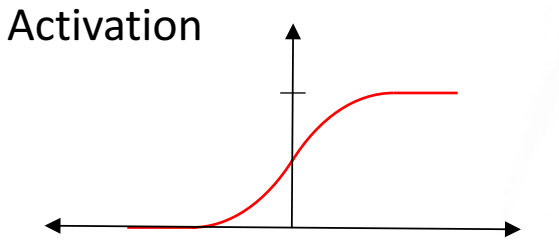
$$\Delta W = e_2 \otimes f_1$$

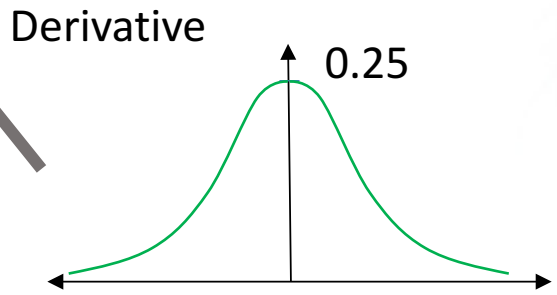
Vanishing Gradient



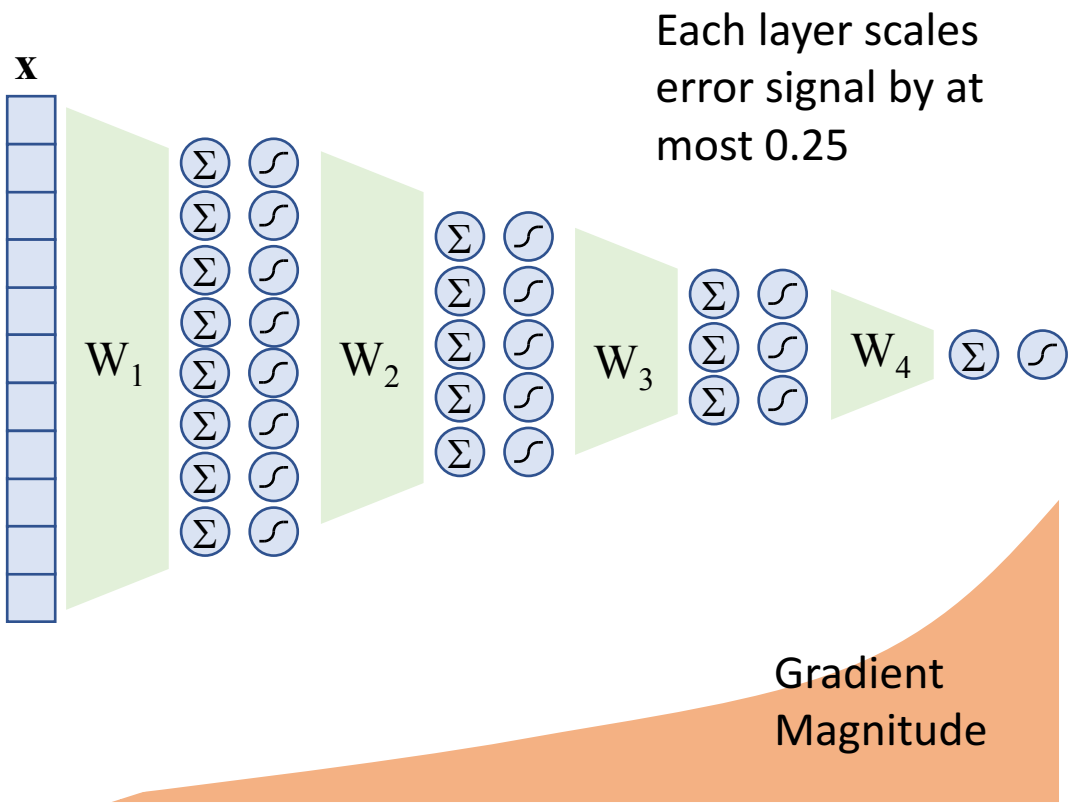
e_2

$$e_1 = W^T e_2$$





Vanishing Gradient



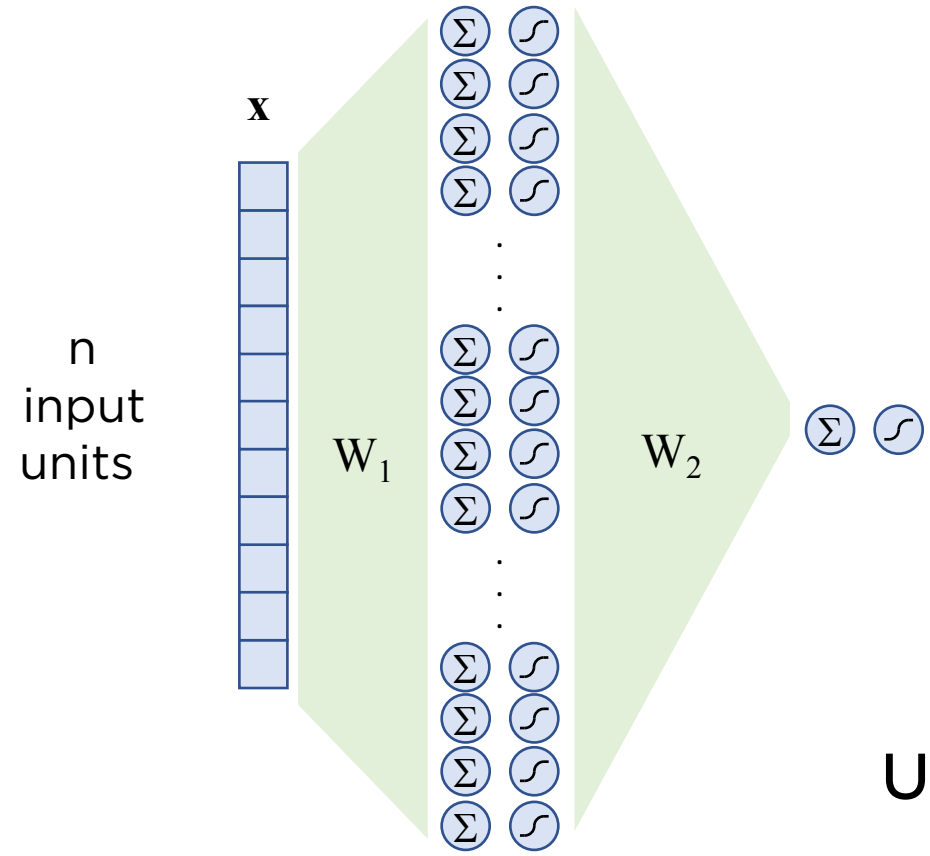
Each layer scales error signal by at most 0.25

Gradient Magnitude

Weights change slowly

Weights change quickly

High Dimensional Search Spaces



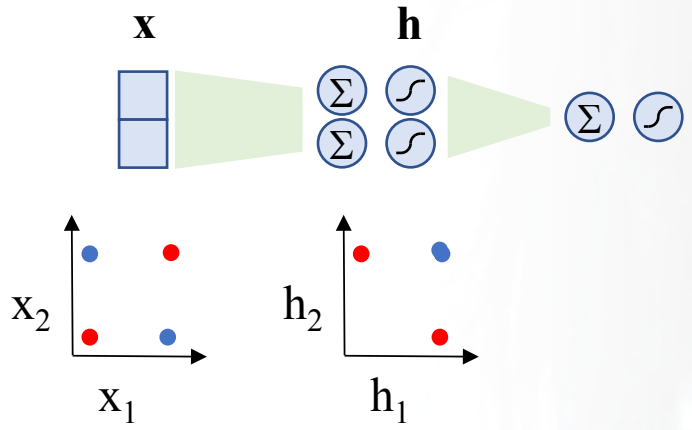
Universal Approximation Theorem

With enough hidden units
you can approximate anything

Internal Representations

- Activations of hidden units

- Totally dependent upon
- Training Set
- Architecture Selection (units in hidden layer)
- Random Initial Weights



- Hard to change poor/unlucky initial representational commitments

Ten Year Intermission

- Neural networks are black boxes
  - No explanations
- Hyperparameter search is hard
  - Complex interactions between learning rates, network sizes, randomization parameters
- Support Vector Machines
  - Return to linear transformations
  - Kernel Trick

Deep Learning

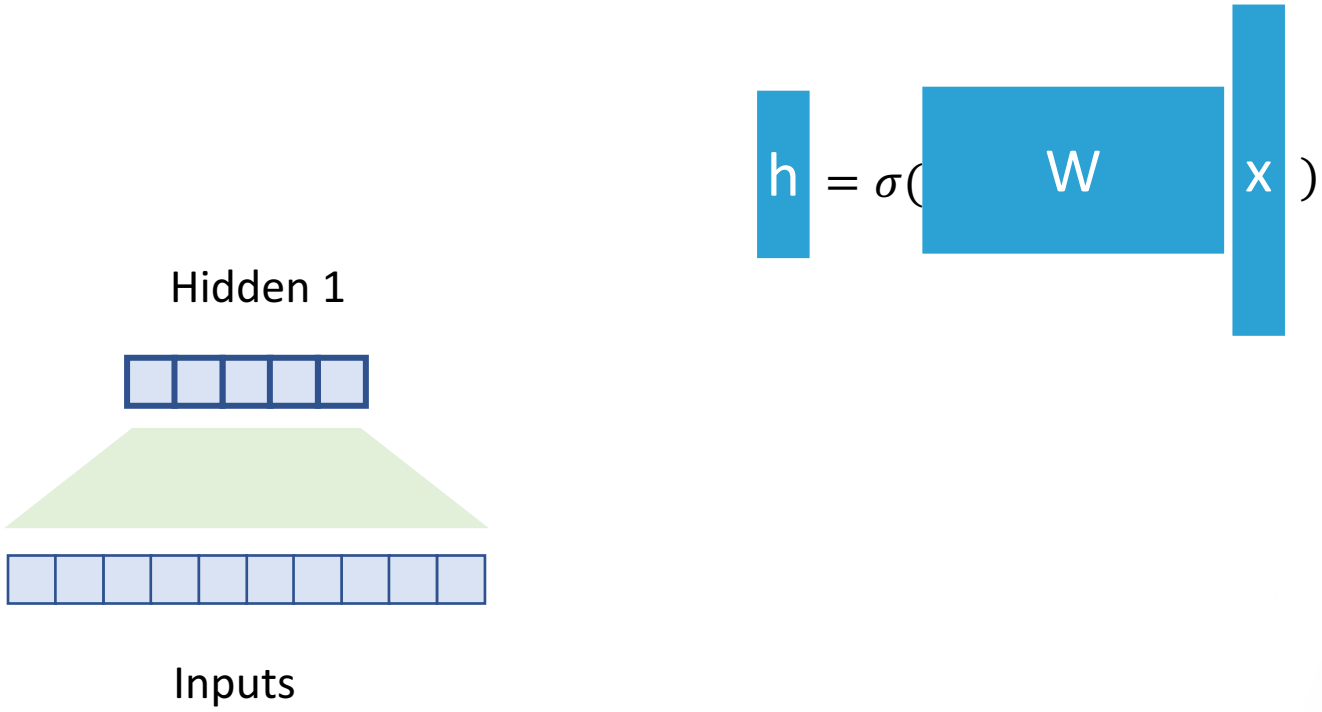
Question

How do you train deep networks with back propagation?

You don’t.

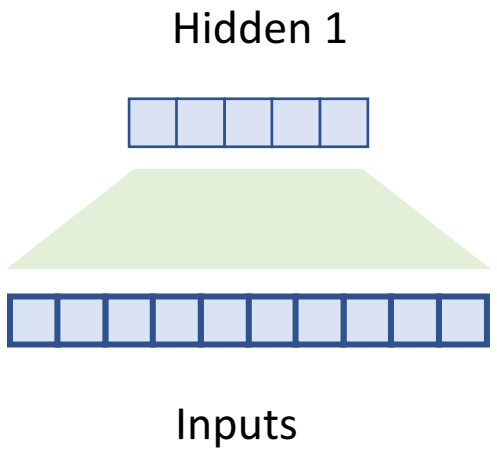
Deep Learning (2006)

Incrementally stack unsupervised networks
Restricted Boltzman Machines



Deep Learning (2006)

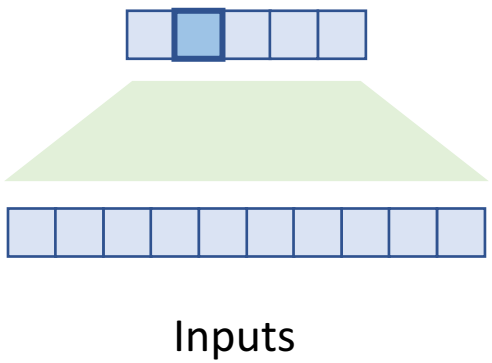
Incrementally stack unsupervised networks
Restricted Boltzman Machines



$$x = \sigma(\ W^T\ h\ )$$

Deep Learning (2006)

Incrementally stack unsupervised networks
Restricted Boltzman Machines



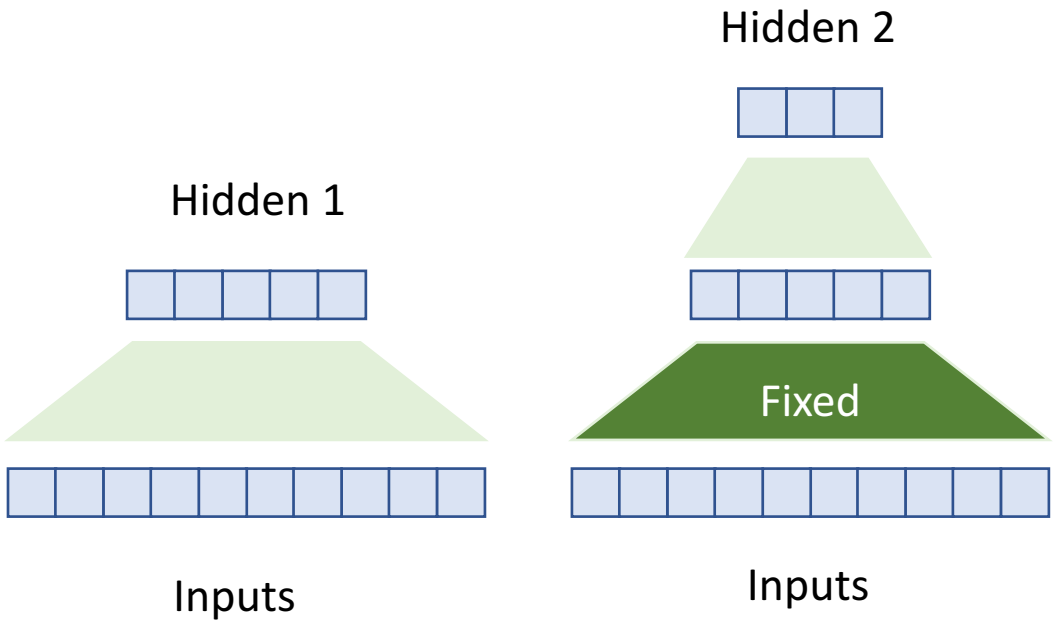
$$h = \sigma(W x)$$

$$x = \sigma(W^T h)$$

Update units randomly to avoid oscillations

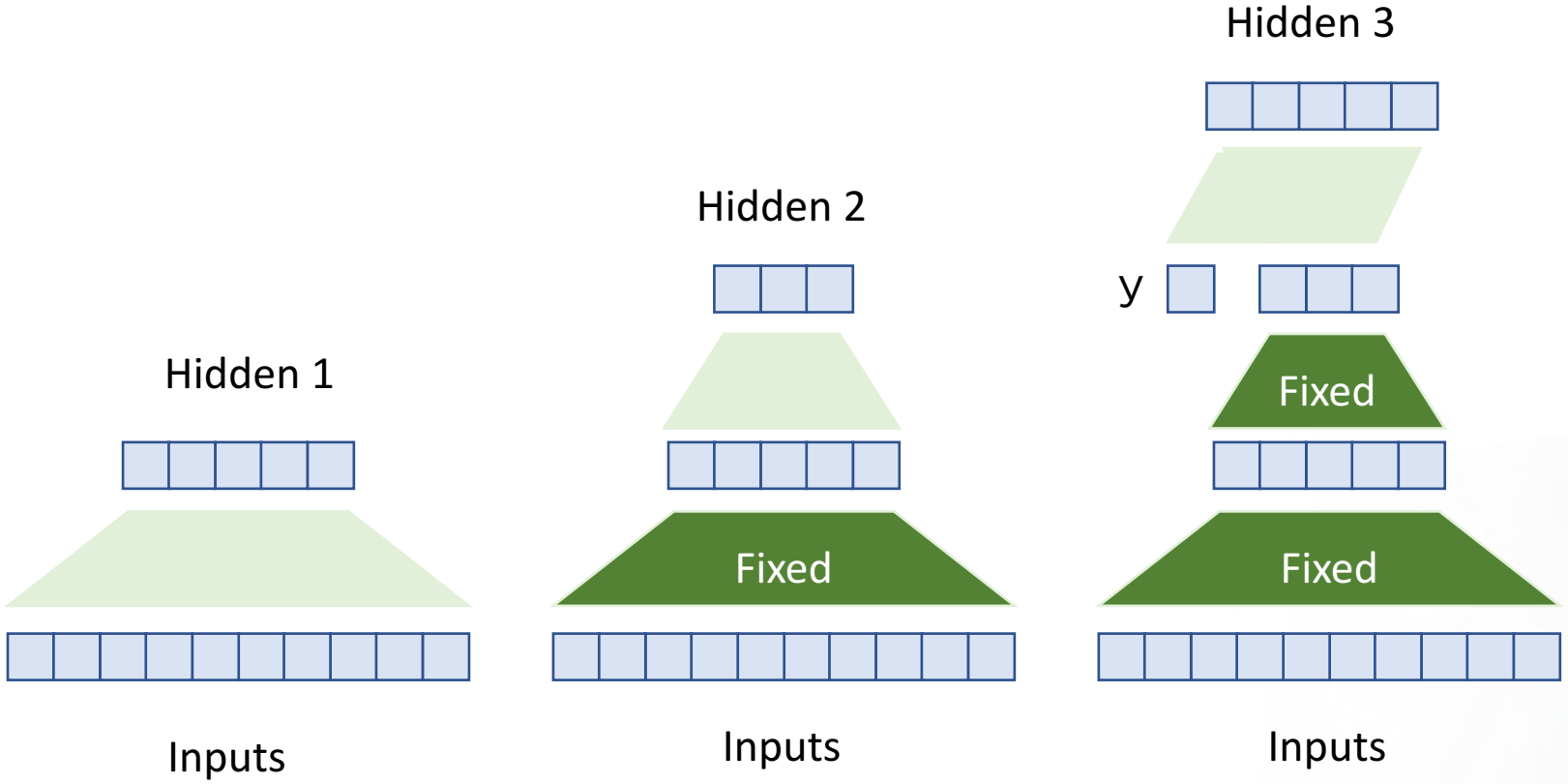
Deep Learning (2006)

Incrementally stack unsupervised networks
Restricted Boltzman Machines



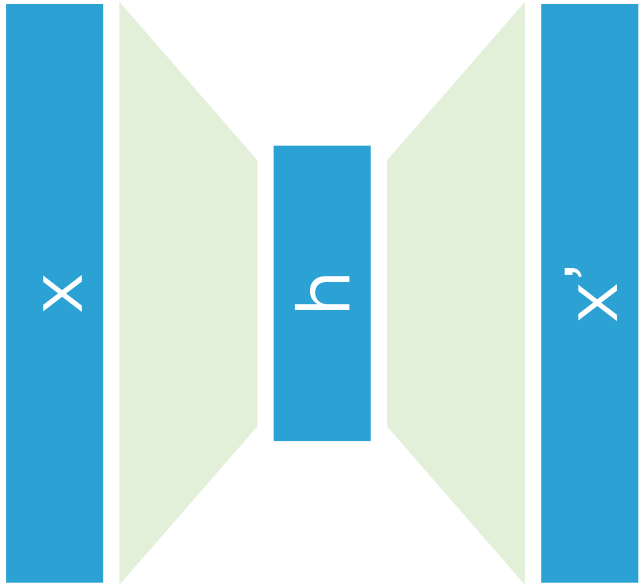
Deep Learning (2006)

Incrementally stack unsupervised networks
Restricted Boltzman Machines



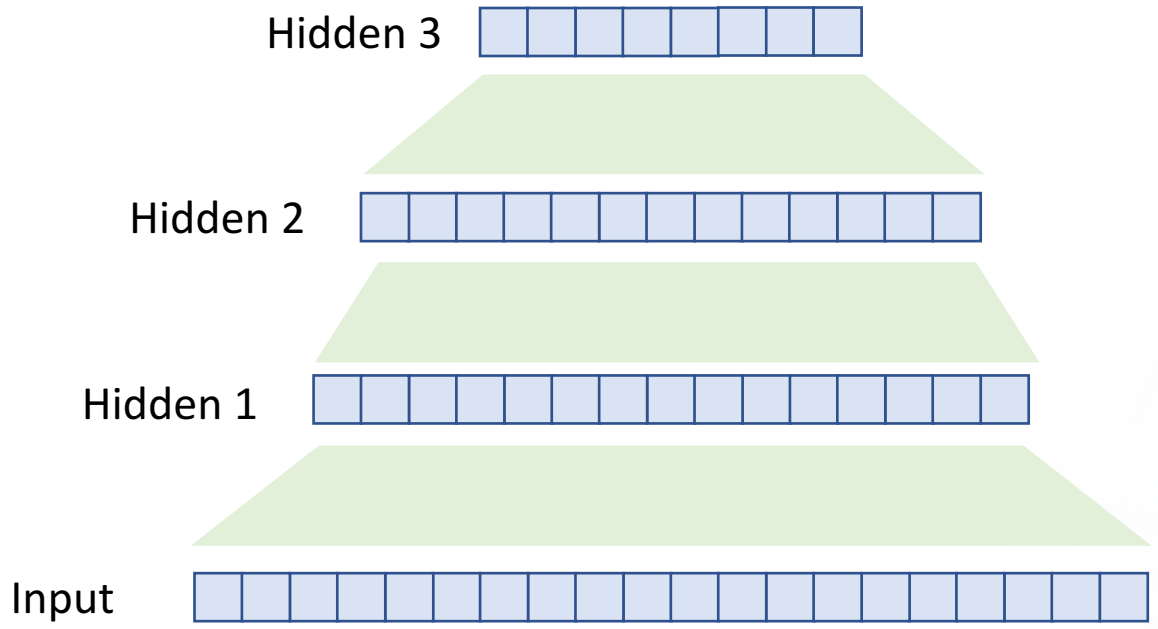
Autoencoders

Input = Output
Capture distribution of data set
Bottleneck forces dimensionality reduction



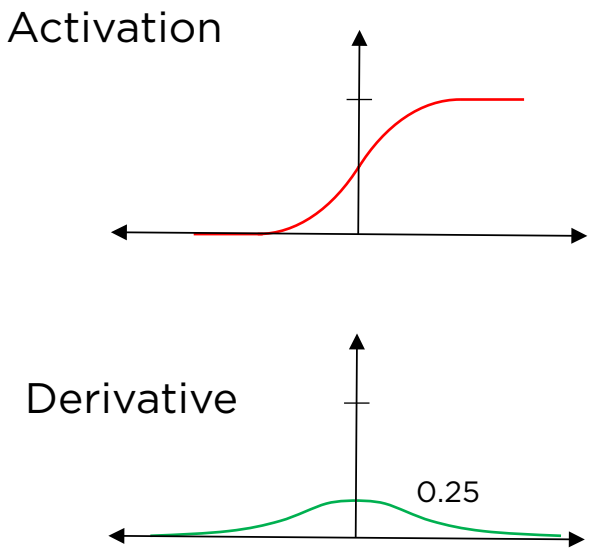
Deep Belief Networks

Stacking autoencoders creates
layers of abstractions of *input space*



Deep Learning (2010)

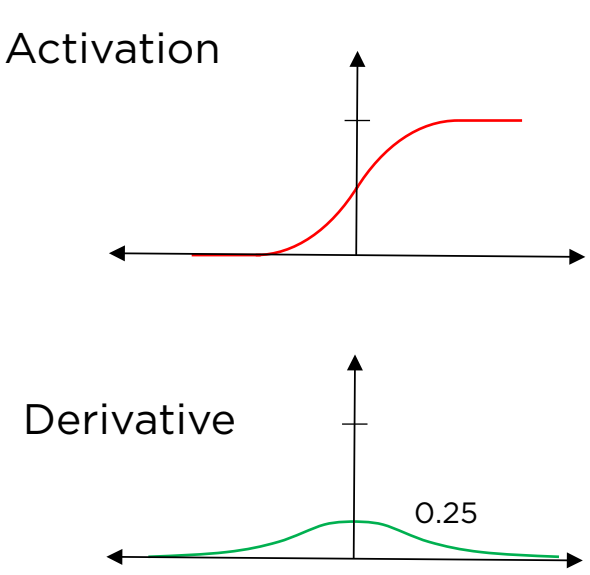
Why deep networks can’t learn



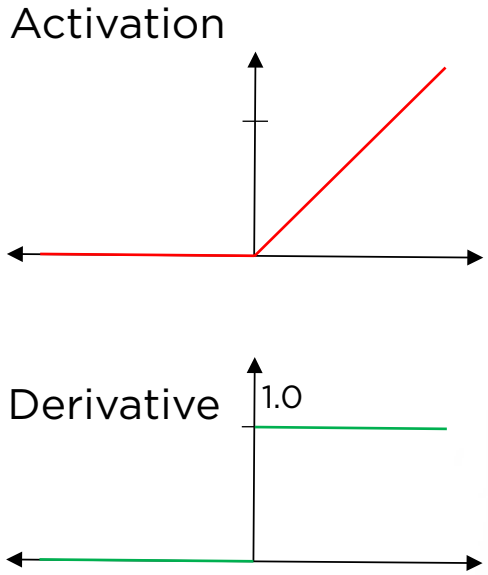
Sigmoid
Vanishing Gradient

Deep Learning (2010)

Now deep networks can learn



Sigmoid
Vanishing Gradient



Rectified Linear Unit
Gated Gradient

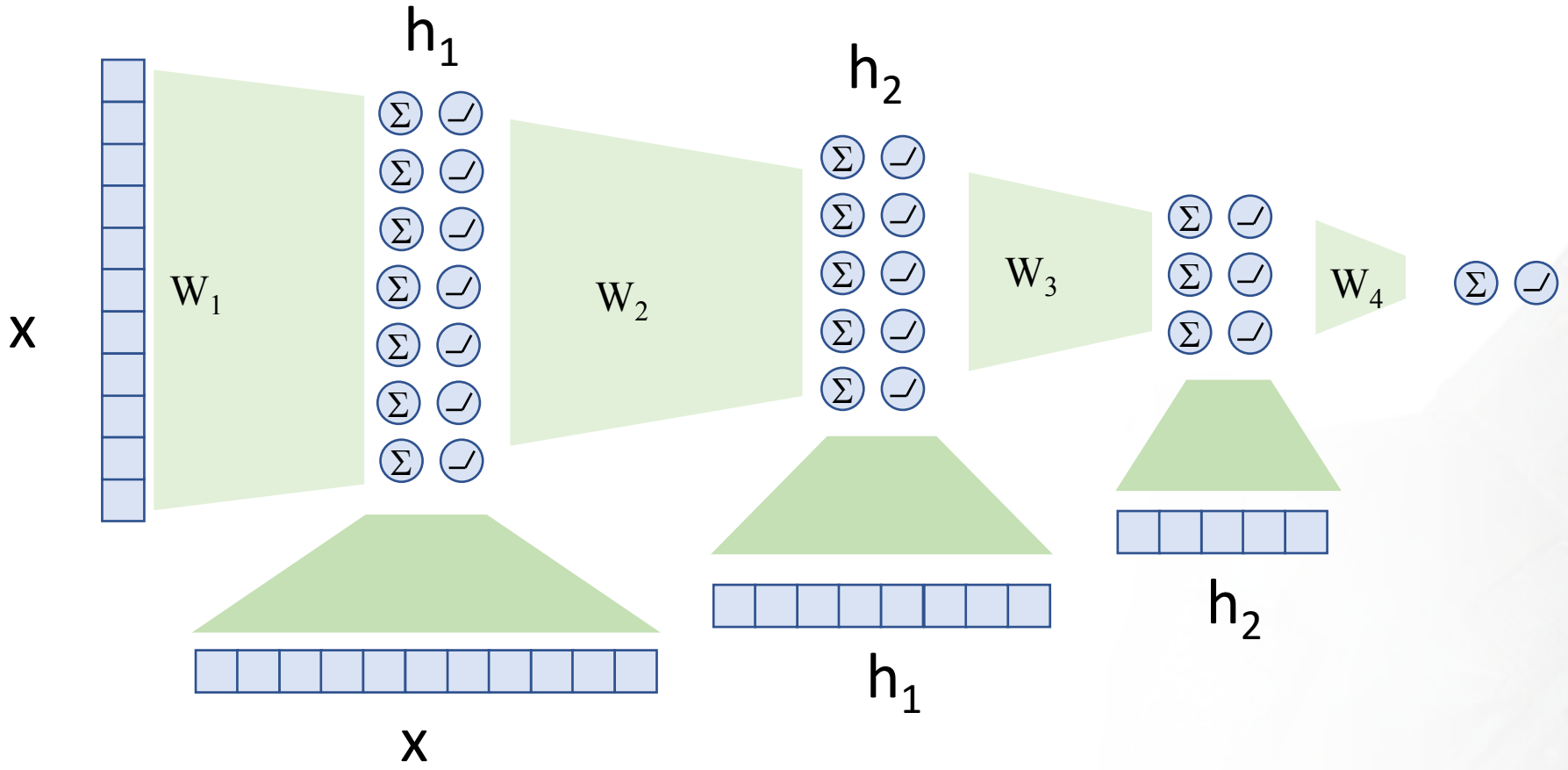
Solved Problems

- Internal Representations
  - Autoencoder constraints
  - Input space knowledge
- Vanishing Gradient
  - Rectified Linear Unit (ReLU)
  - Functional knowledge

Now we can build deep functional networks

Deep Functional Networks

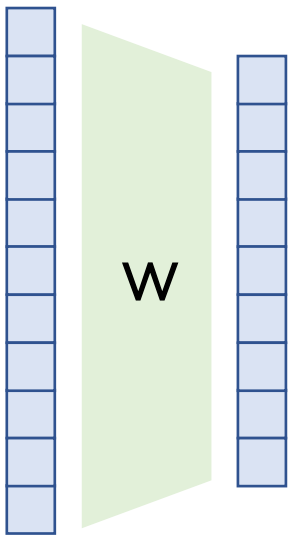
Can be trained with back propagation and autoencoder constraints



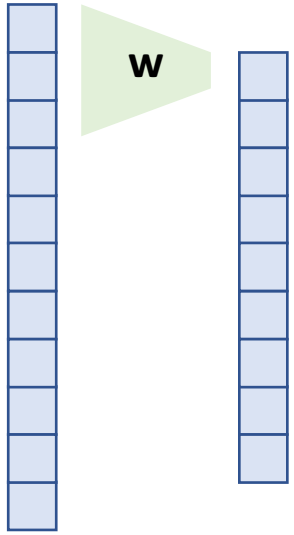
Convolutions

Reducing the number of weights

Spatial distribution of a simple function



O(nm) weights

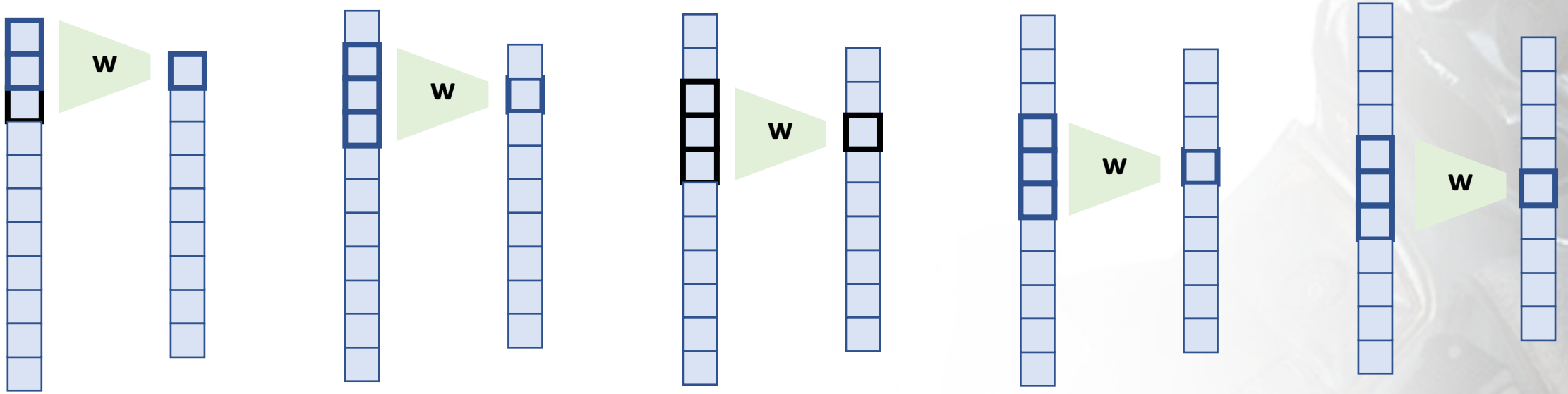


O(k) weights

Convolutions

Reducing the number of weights

Spatial distribution of a simple function

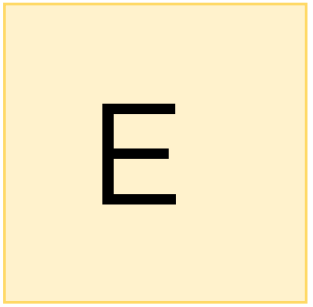


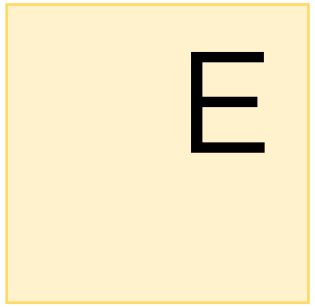
O(k) weights

Applied m times

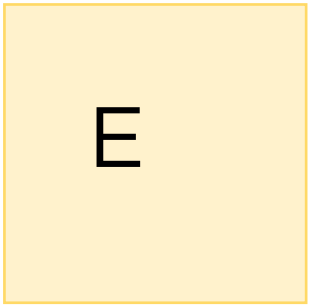
Convolutions

Capture invariances

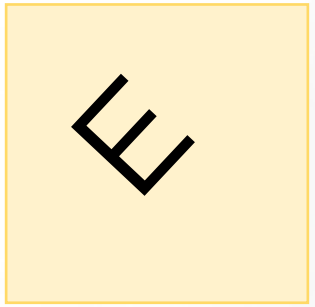




Translation



Scale

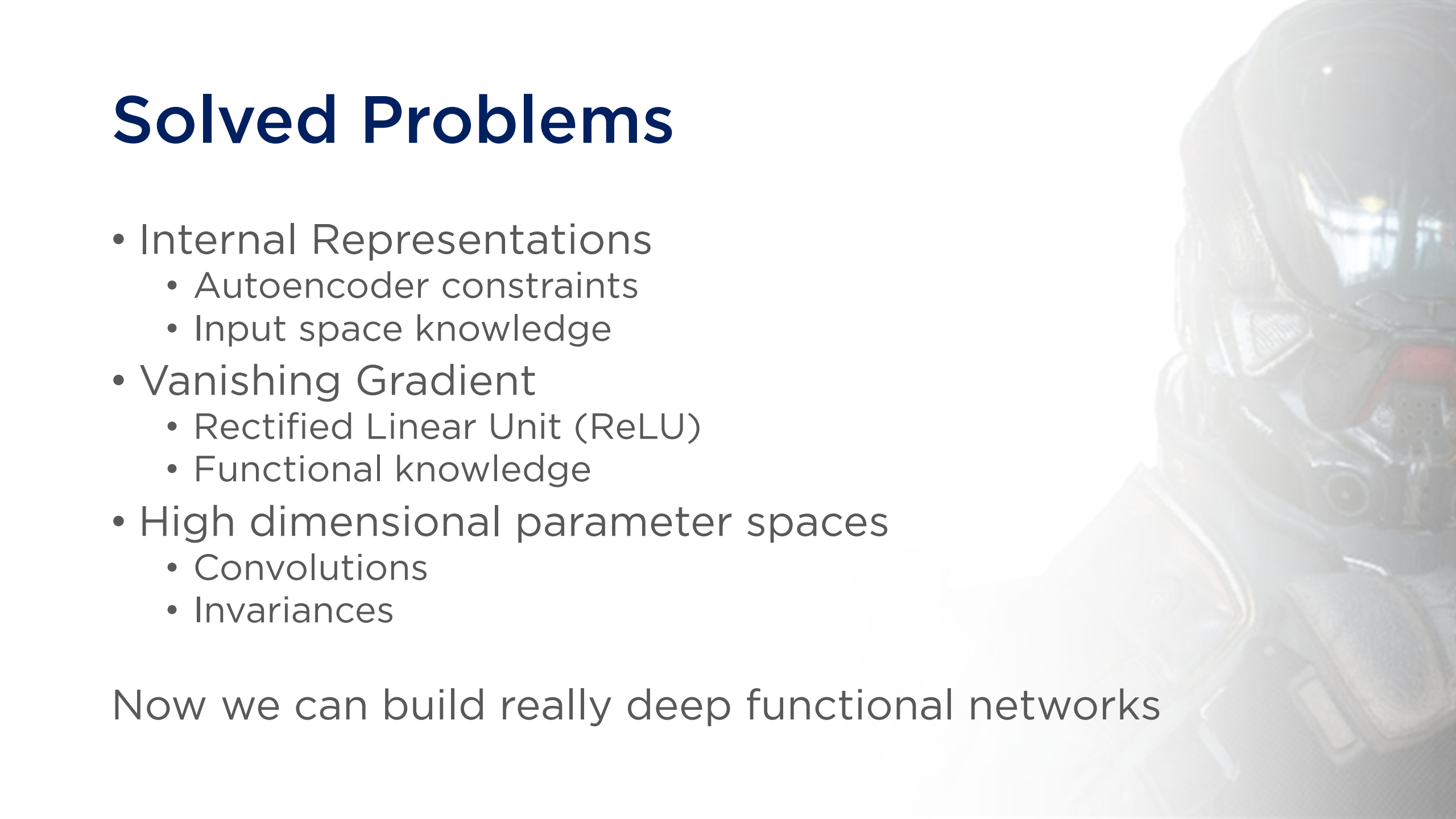


Rotation

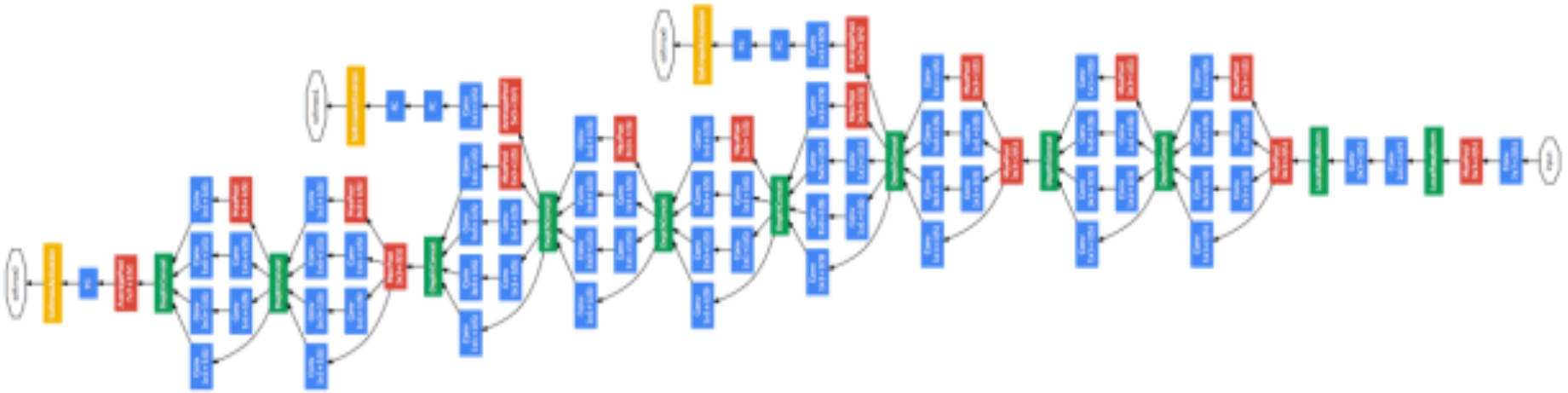
Solved Problems

- Internal Representations
  - Autoencoder constraints
  - Input space knowledge
- Vanishing Gradient
  - Rectified Linear Unit (ReLU)
  - Functional knowledge
- High dimensional parameter spaces
  - Convolutions
  - Invariances

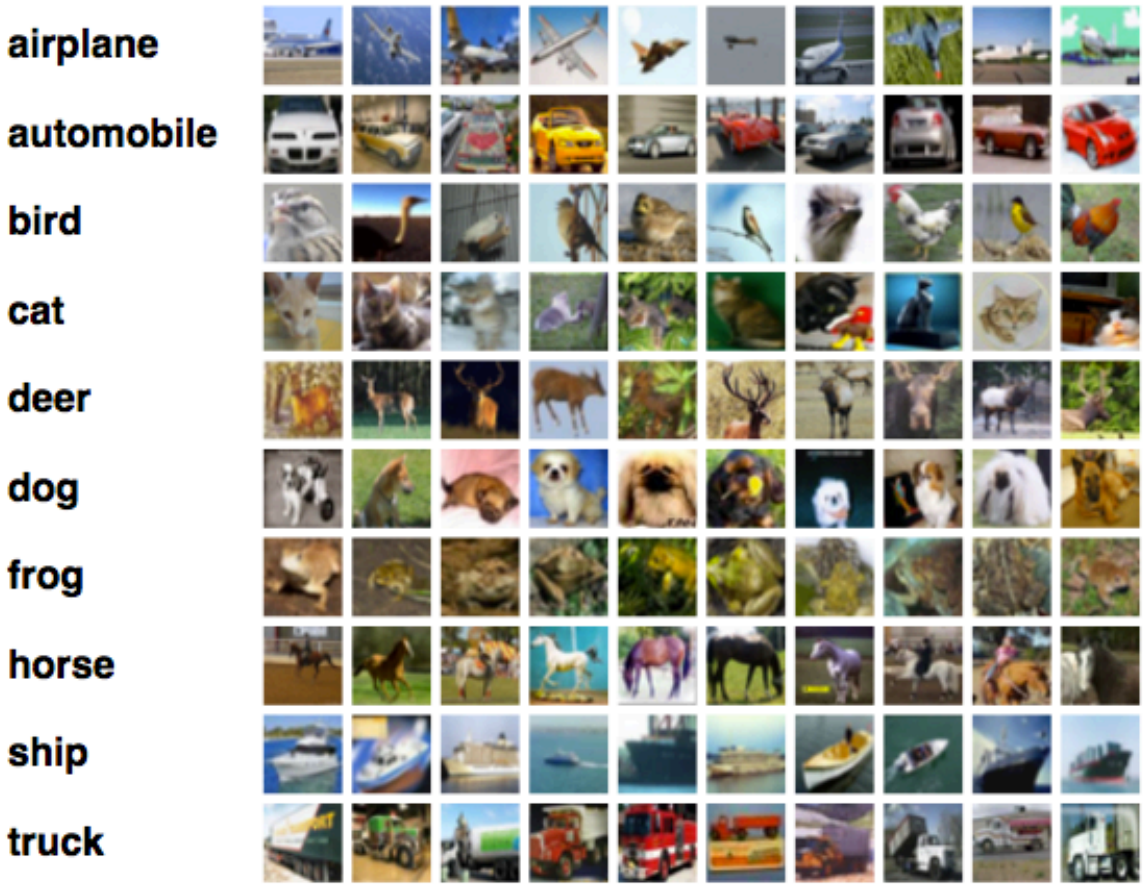
Now we can build really deep functional networks



Deep Learning



Neural Networks Are Really Great!

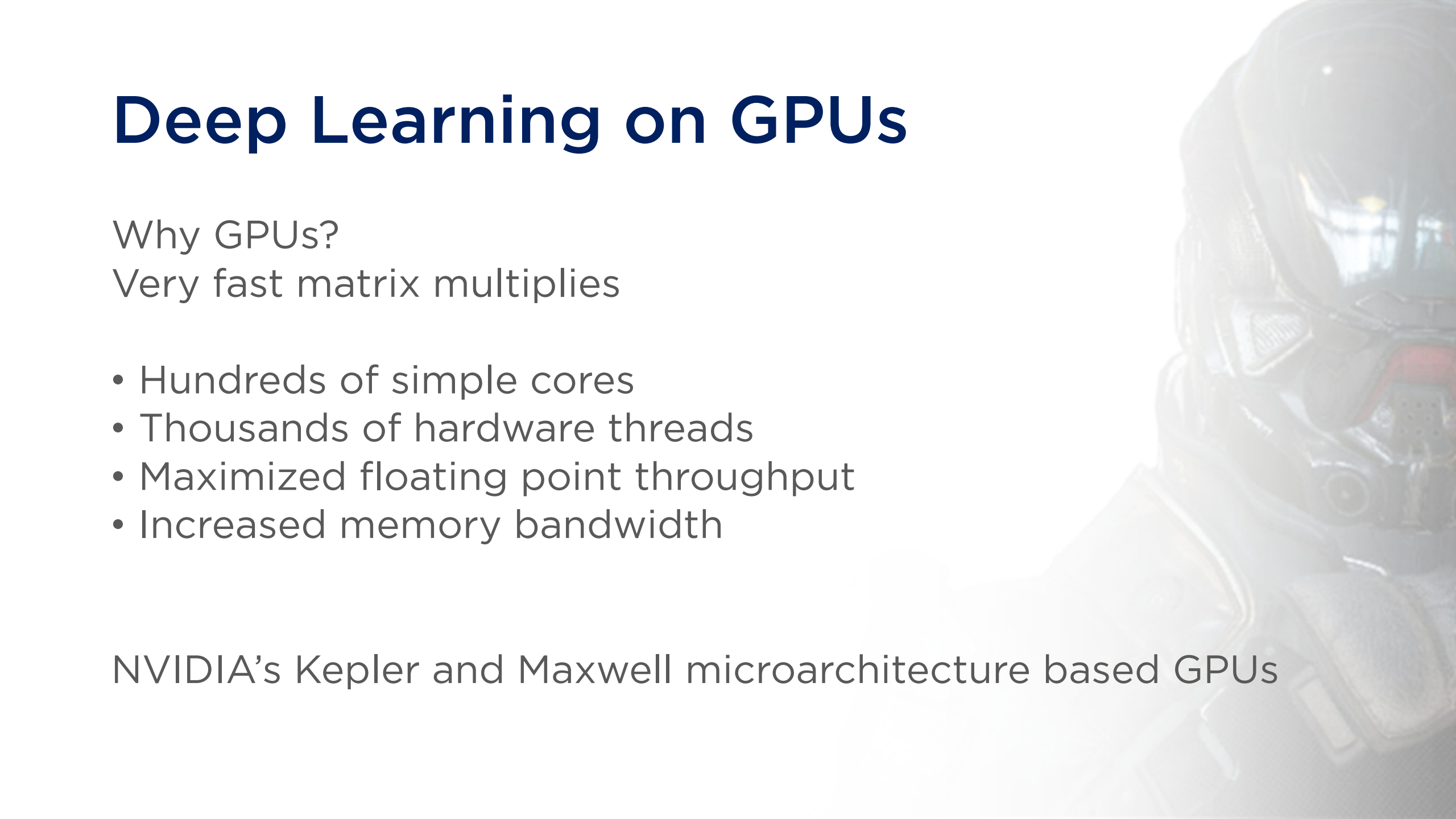


Deep Learning on GPUs

Why GPUs?
Very fast matrix multiplies

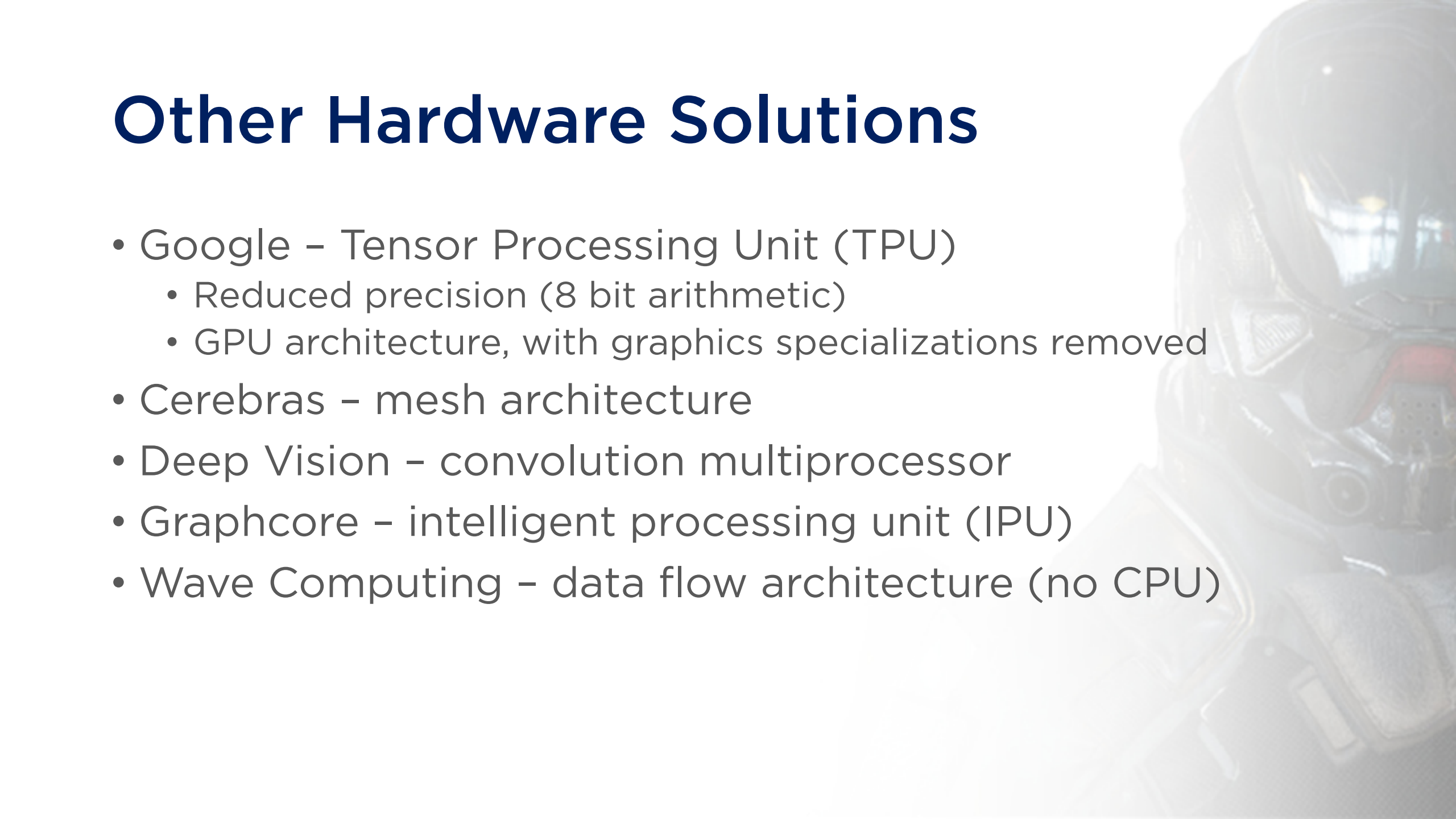
- Hundreds of simple cores
- Thousands of hardware threads
- Maximized floating point throughput
- Increased memory bandwidth

NVIDIA’s Kepler and Maxwell microarchitecture based GPUs



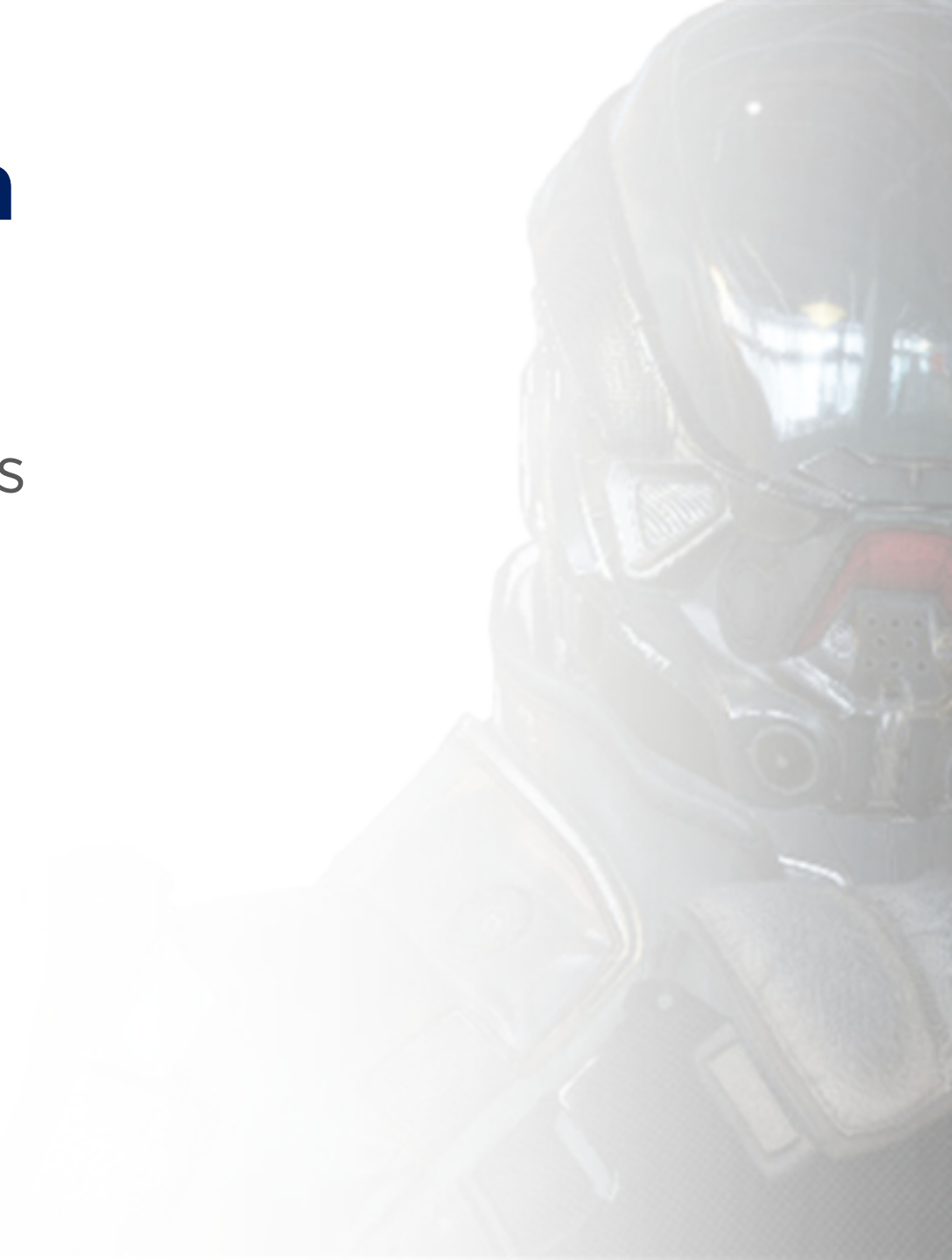
Other Hardware Solutions

- Google – Tensor Processing Unit (TPU)
  - Reduced precision (8 bit arithmetic)
  - GPU architecture, with graphics specializations removed
- Cerebras – mesh architecture
- Deep Vision – convolution multiprocessor
- Graphcore – intelligent processing unit (IPU)
- Wave Computing – data flow architecture (no CPU)



Don’t Forget Big Data

- Lots of data to work with
- Millions of images and documents
- Capturing distributions easier



Are We There Yet?

Deep Learning Applications

- Speech Recognition
- Alpha Go
- Self Driving Vehicles
- Text Analysis
- Image Analysis

Large input and/or output representations (e.g. images)

Transformation invariances on inputs (e.g. translation/scale/rotation)

Nothing But Matrix Multiplication

Easy

Hard

Stirring The Pile

The deep learning algorithms find a set of weights for the network, but

The model selection problem has not gone away

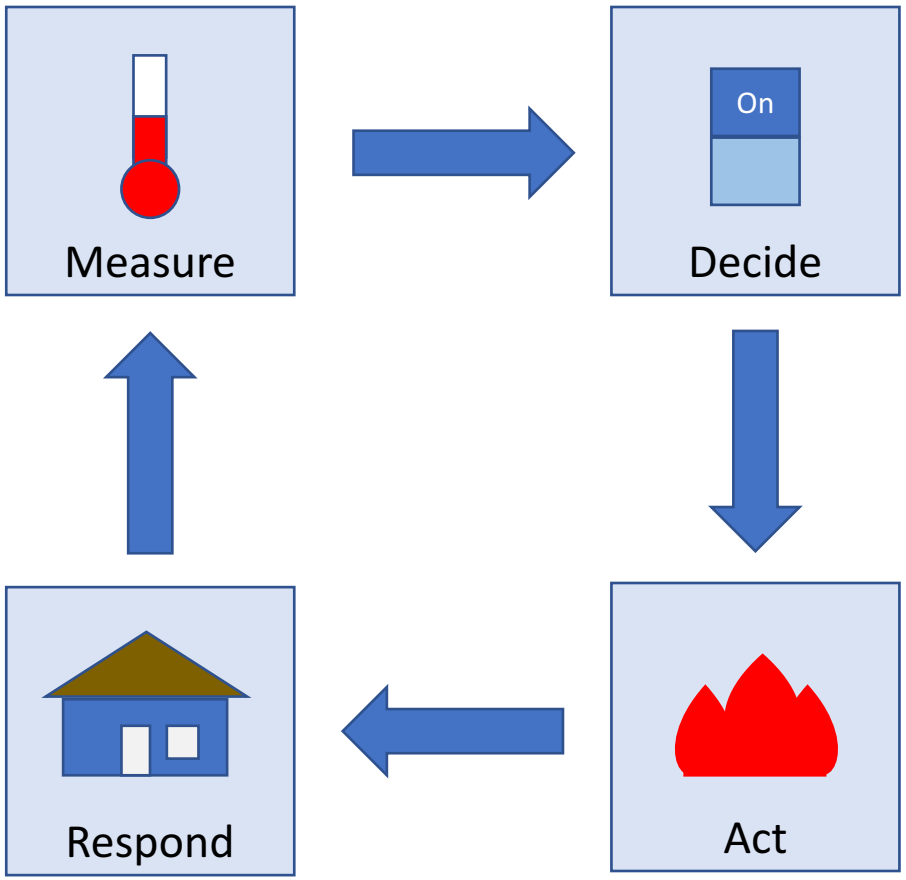
Developers still have to

- Specify the data flow graph
- Identify type of processing occurring at each node and layer
- Supply more hyper-parameters
  - convolution size, hidden units, activation functions, etc.

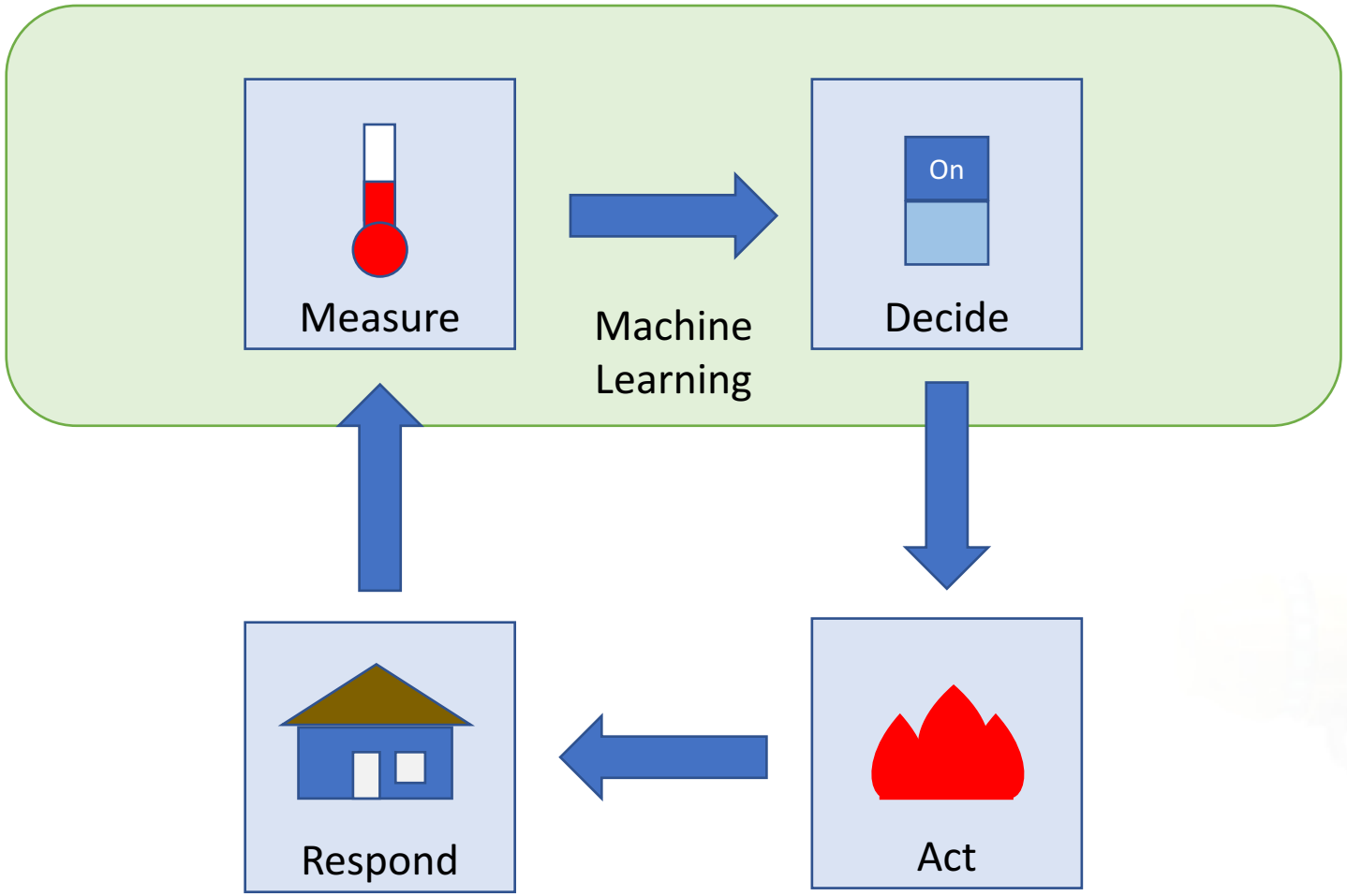
Back To The Cursed Loop

```
do
   select p from possible hyperparameters
   model = train(data, p)
until model is acceptable
```

Systems and Models



Systems and Models



TensorFlow Demo

MNIST Data

Next Steps