Chapter 7, Part 2: Neural Networks

Advanced Topics in Statistical Machine Learning

Tom Rainforth

Hilary 2024

rainforth@stats.ox.ac.uk

Artificial Neural Networks

- An **artificial neural network** (or typically just neural network) is an equivalent, older, term for a deep learning model
- Historically, though, this tended to imply a more restricted class of models than the vast array of modern deep networks
  - Original motivations where different to the current intuitions of arbitrary differentiable frameworks and deep feature hierarchies
  - This can make the surrounding terminology at bit awkward: many redundant and overloaded terms
- The core **building blocks** for most modern models are still based on ideas and formulations from the 80s and early 90s
  - Even though the concept of deep learning is very general, the core components that are used to built up computational graphs tend to follow certain formulations and principles

Recap: Deep Learning as Multiple Layers of Features

A deep learning predictive model, $f_\theta : \mathcal{X} \mapsto \mathcal{Y}$, is given by a composition of functions

$$f_\theta = f^m_{\theta^m} \circ f^{m-1}_{\theta^{m-1}} \circ \cdots \circ f^2_{\theta^2} \circ f^1_{\theta^1}$$

We can also think of this as having multiple **layers** of features, $h^\ell \in \mathbb{R}^{d_\ell}$, such that $h^0 = x$, $h^m = f_\theta(x)$ and

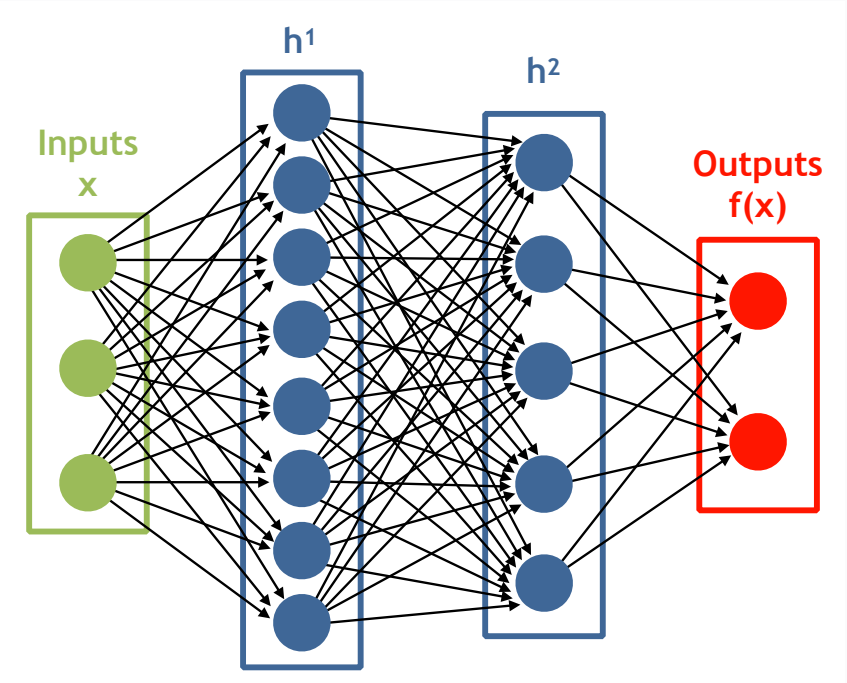
$$h^\ell = f^\ell_{\theta^\ell}\left(h^{\ell-1}\right)$$

Layers that are not inputs our outputs (i.e. $h^1, \ldots, h^{m-1}$) are known as **hidden**

We can extend this to general **computation graphs** by allowing individual h^ℓ to be split up into $h^\ell = \left[h^{\ell,1}, h^{\ell,2}, \ldots, h^{\ell,t_\ell}\right]$ where each $h^{\ell,j}$ need not depend on all $h^{\ell-1,1}, h^{\ell-1,2}, \ldots, h^{\ell-1,t_{\ell-1}}$.

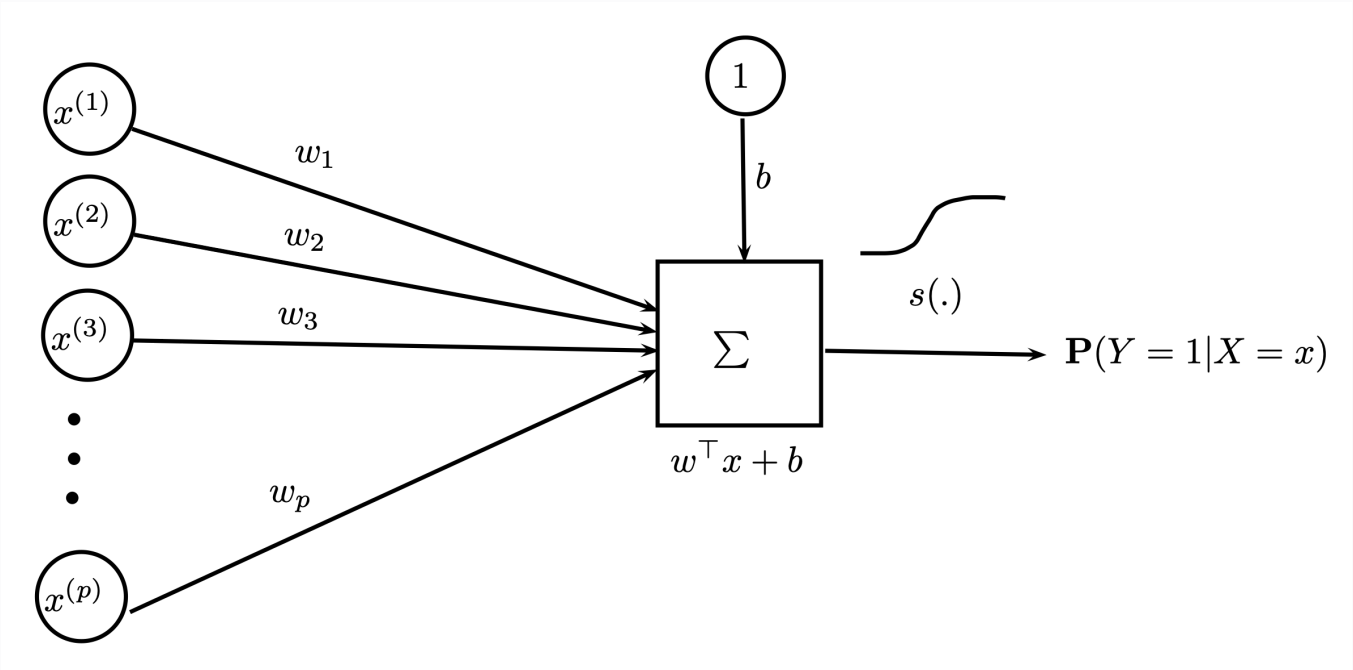
Neural Networks

- We can think of introducing a separate node, known as a **unit**, for each dimension h_j^ℓ of each layer h^ℓ
- Each unit is scalar valued and depends on some or all of the units in the previous layer: $h_j^\ell = f_j^\ell(h^{\ell-1})$
- These dependencies induce a **directed network** of connections between units in consecutive layers



Computational Graph for a Linear Classifier

Consider the following computational graph that corresponds to the linear classifier $s(w^T x + b) \approx P(Y = 1|X = x)$, where s is a fixed nonlinear function (e.g. $s(z) = 1/(1 + \exp(-z))$)



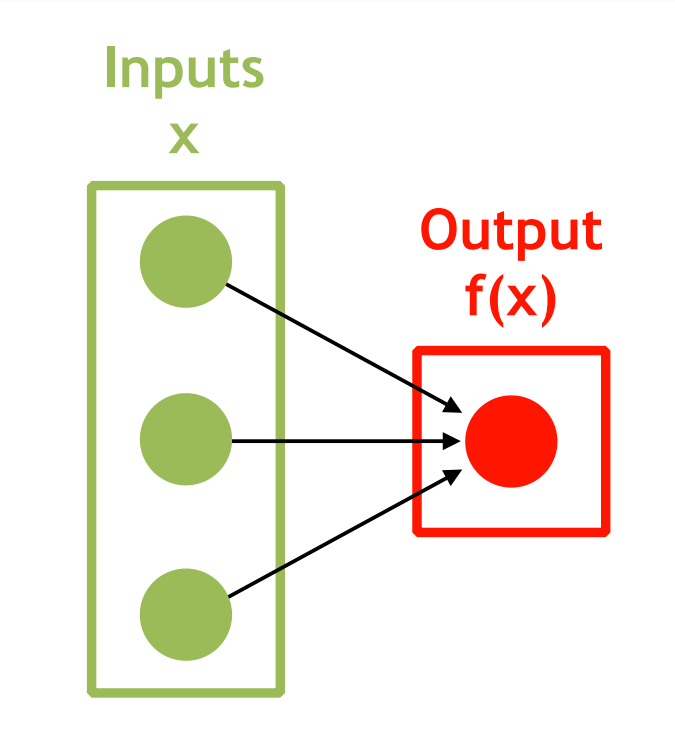
The model parameters are the **weight** vector w and scalar **bias** b

A Single Node Network

We can think of this as a neural network with no hidden units and a single output unit $f(x) = s(w^T x + b)$

Here s provides a **nonlinearity** and is known as an **activation function**, where $w^T x + b$ is a **pre-activation**[1]

Our node thus comprises of a nonlinear transformation of a weighted sum of the nodes at the last layer (which in this case is the inputs)



[1] The literature is infuriatingly contradictory on what to call the input and output of the activation function, with the term "activation" sometimes used to refer to each. Activation meaning output is the prevalent usage in deep learning circles and how I will occasionally use it, but the previous ML course used it to mean the input.

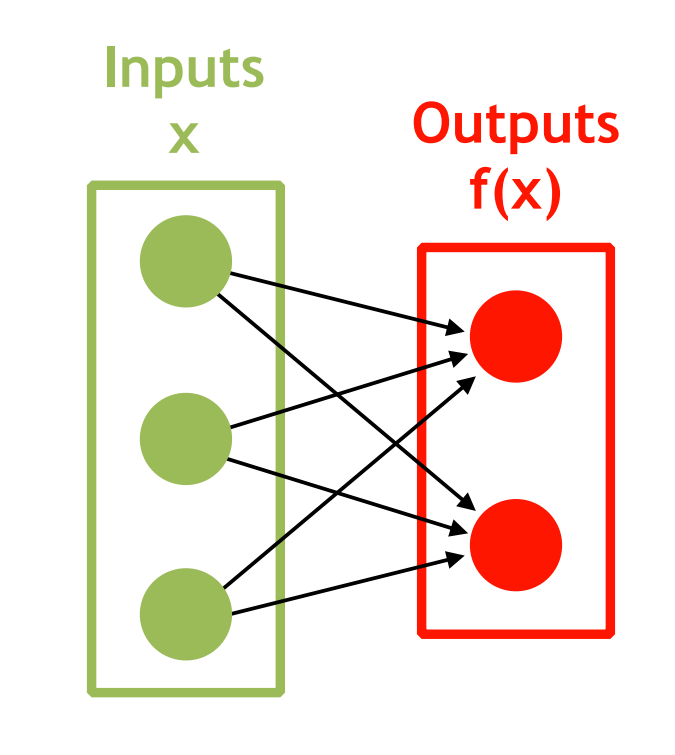
Multiple Outputs

We can also introduce additional output nodes

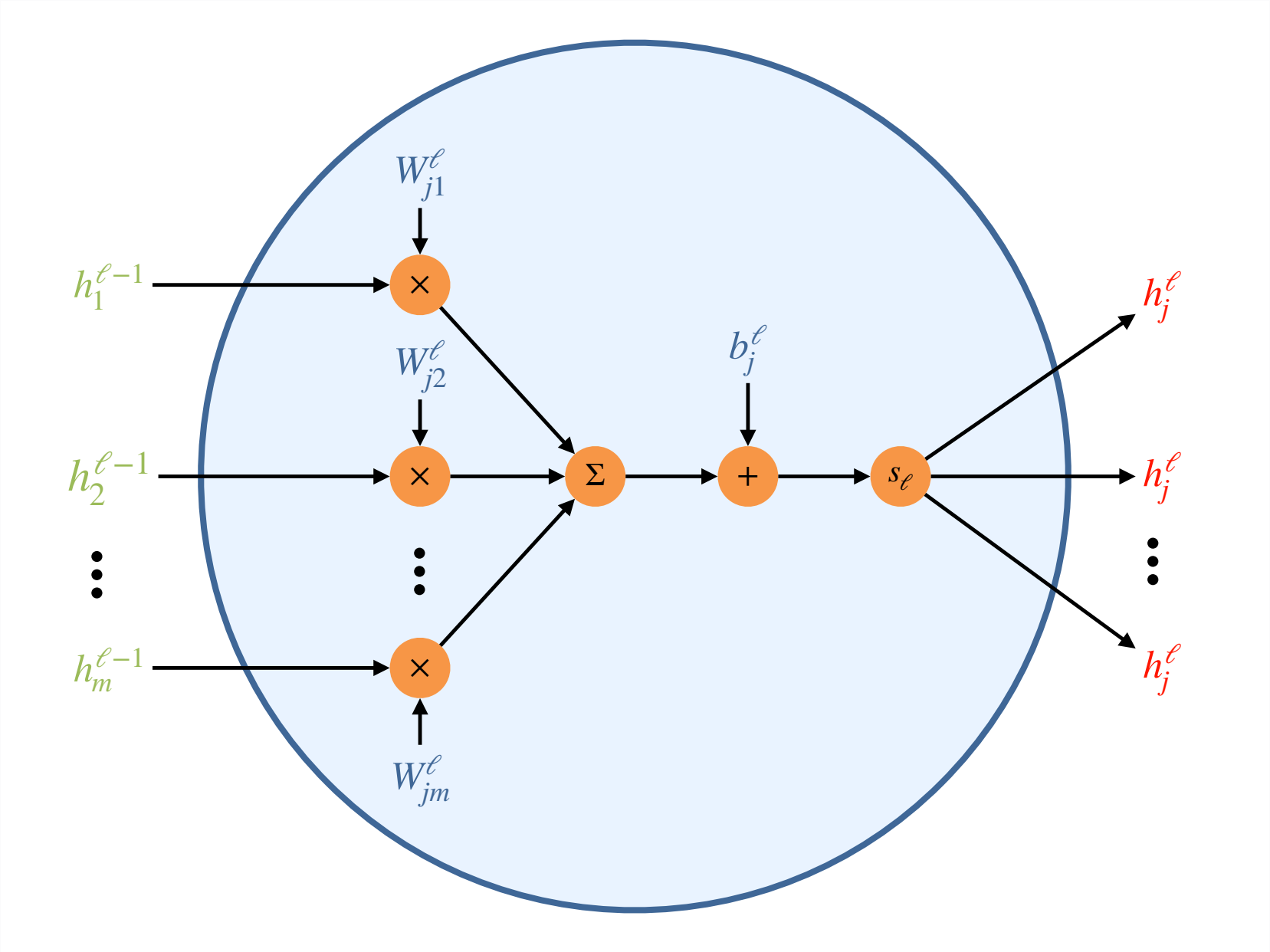
We now have $f(x) = s(Wx + b)$ where W is now a weight **matrix** and b a **vector** of biases (with $W \in \mathbb{R}^{2\times 3}$ and $b \in \mathbb{R}^2$ in the example)

Example: predicting probability of classes in multi–class classification

We can also think about the form of this output node as a building block for constructing more complicated networks



Computational Graph for Single Hidden Unit



Fully–Connected Single Hidden Layer Network

We can significantly improve the predictive power of this model by introducing a **hidden layer**:

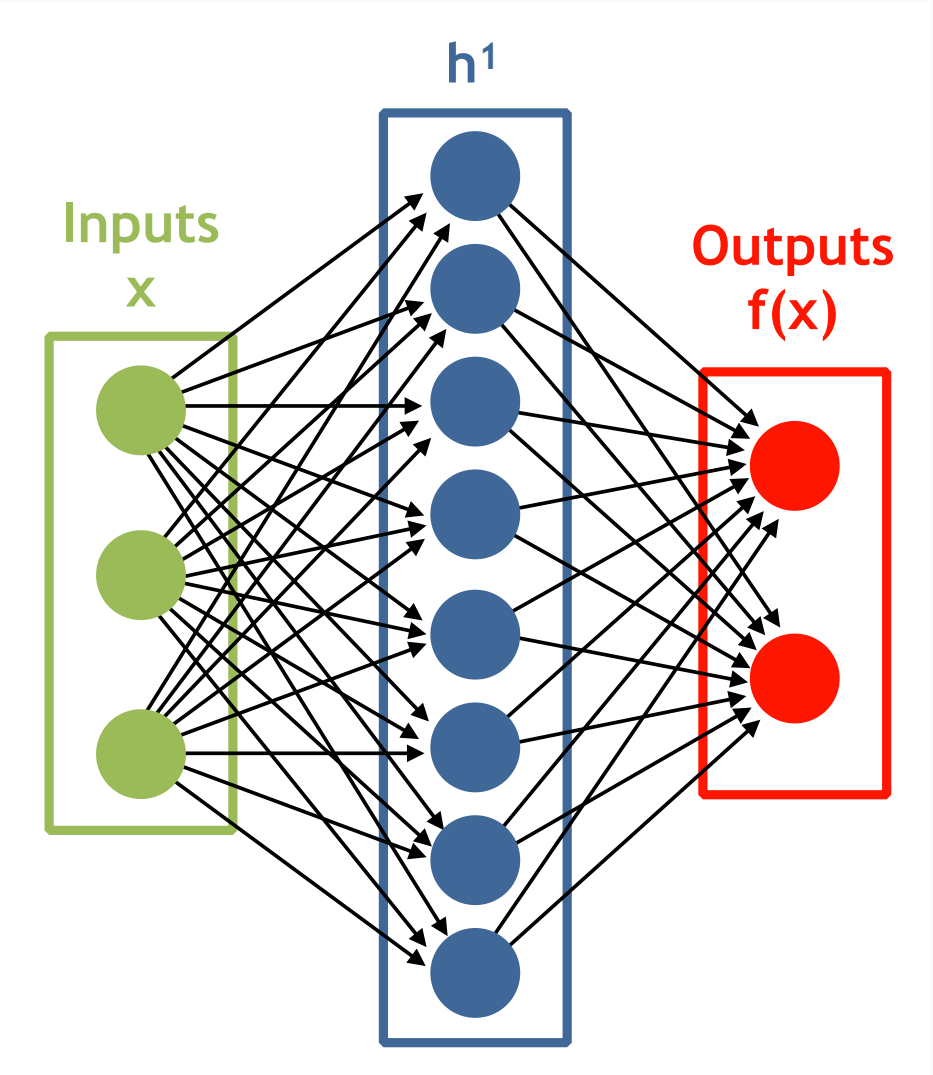
$$h^1 = s_1(W^1 x + b^1)$$

$$f(x) = s_2(W^2 h^1 + b^2)$$

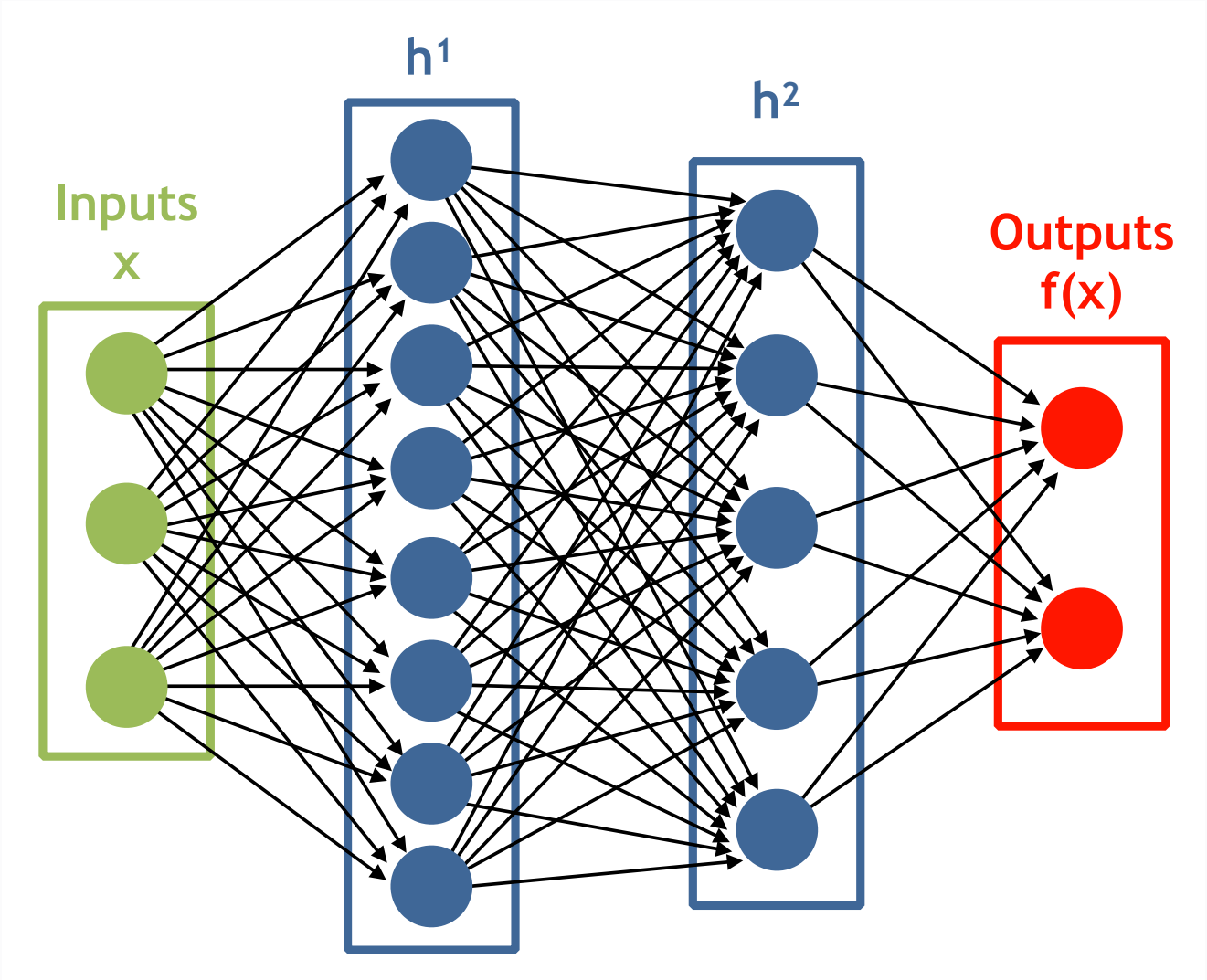
where s_1 and s_2 are fixed activation functions

For hidden units, it is customary that s is an **element-wise** nonlinearity

Parameters: W^1, W^2, b^1, b^2

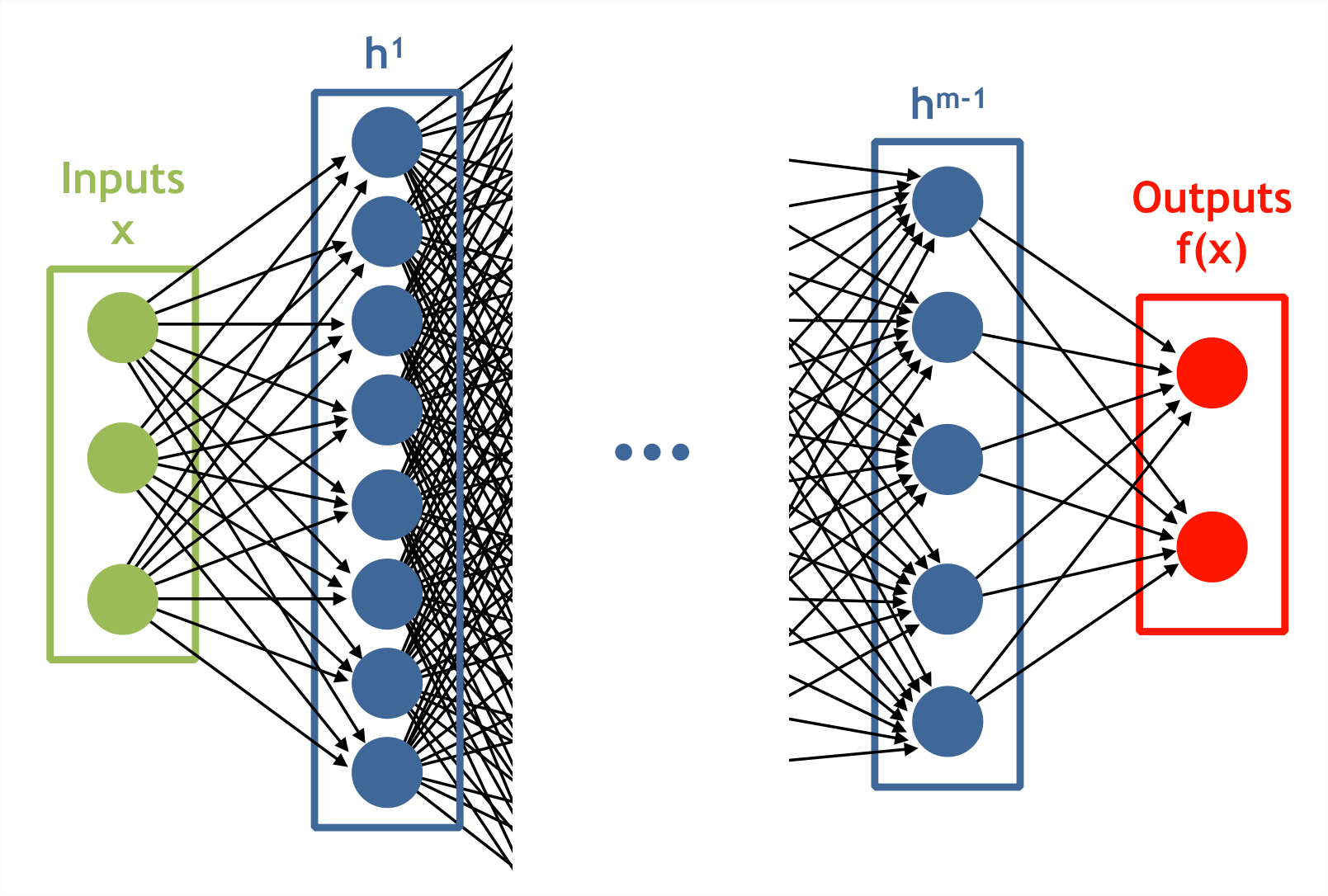


Additional Hidden Layer



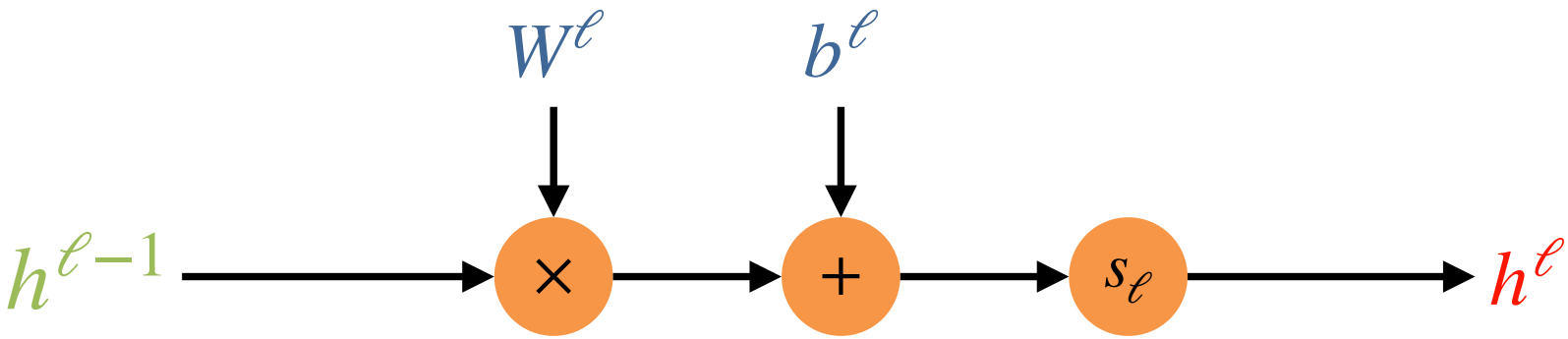
$$h^1 = s_1(W^1 x + b^1) \quad h^2 = s_2(W^2 h^1 + b^2) \quad f(x) = s_3(W^3 h^2 + b^3)$$

Fully Connected Deep Neural Network

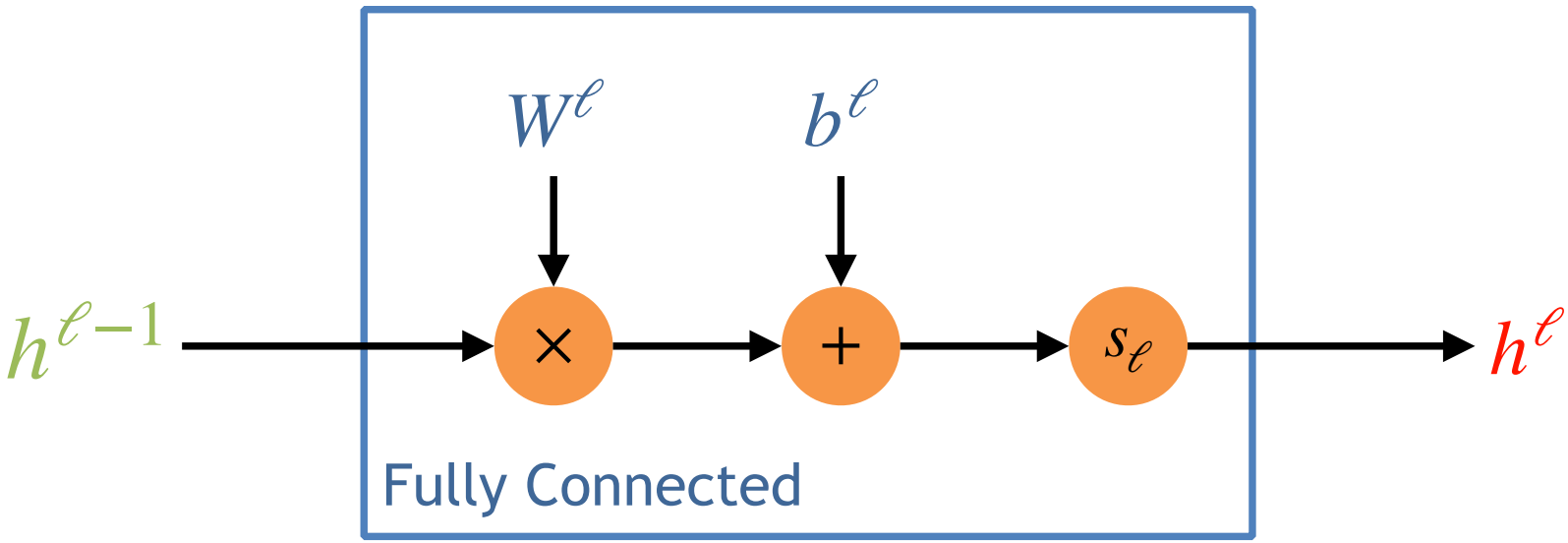


$$h^0 = x \qquad h^\ell = s_\ell(W^\ell h^{\ell-1} + b^\ell) \qquad f(x) = h^m$$

Computational Graph for a Fully Connected Layer



Computational Graph for a Fully Connected Layer



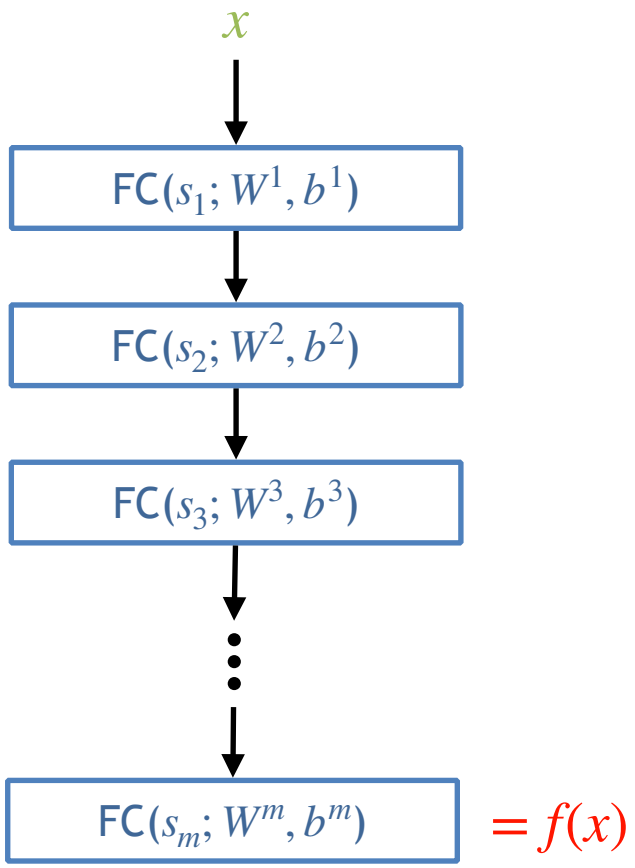
Computational Graph for a Fully Connected Layer

Multi-layer Perceptrons

An alternative name for a full connected neural network with multiple layers is a **multi-layer perceptron** (MLP)

We can form a computation graph for an MLP using FC blocks

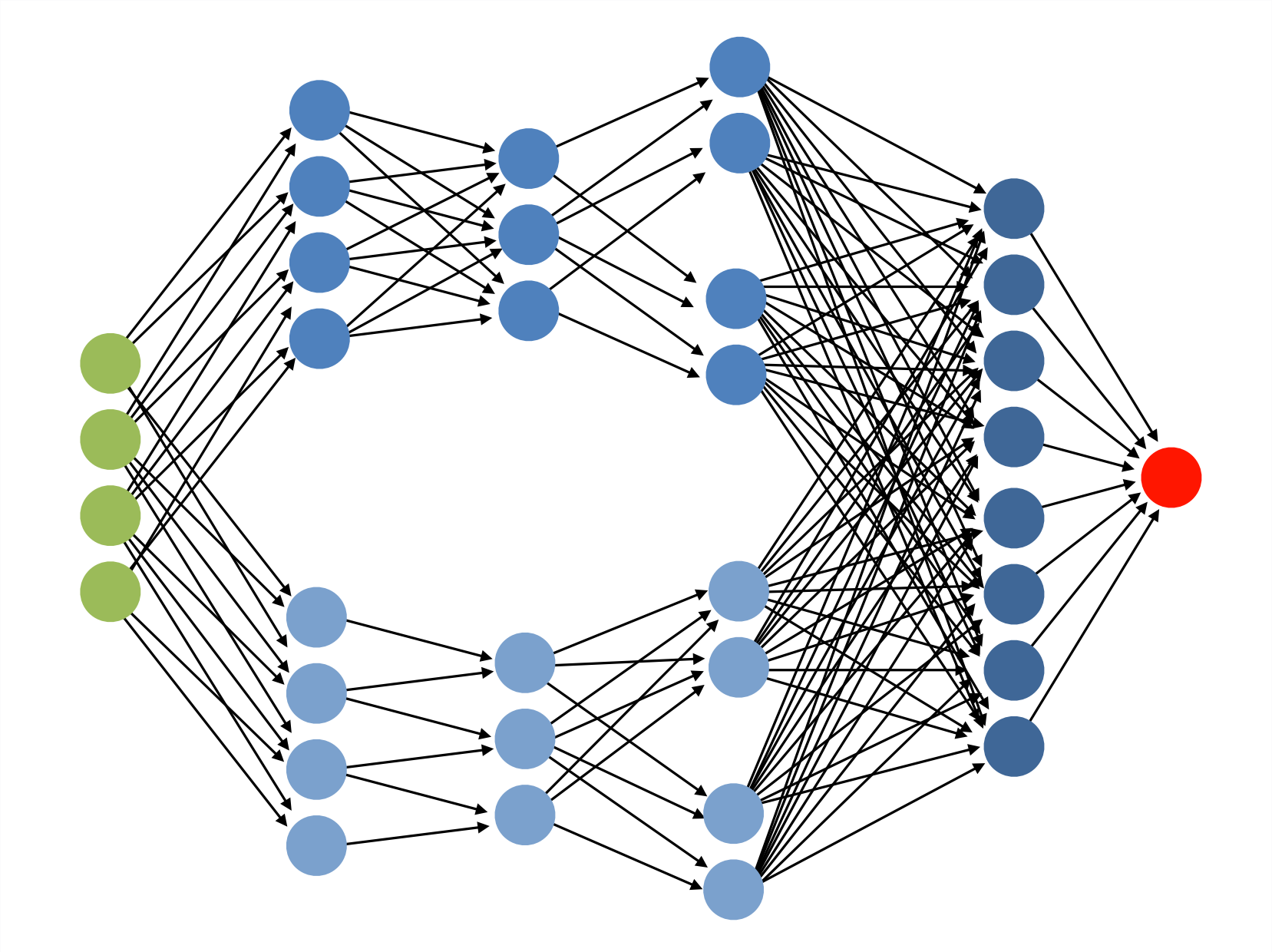
We can also use an MLP itself as a computation block



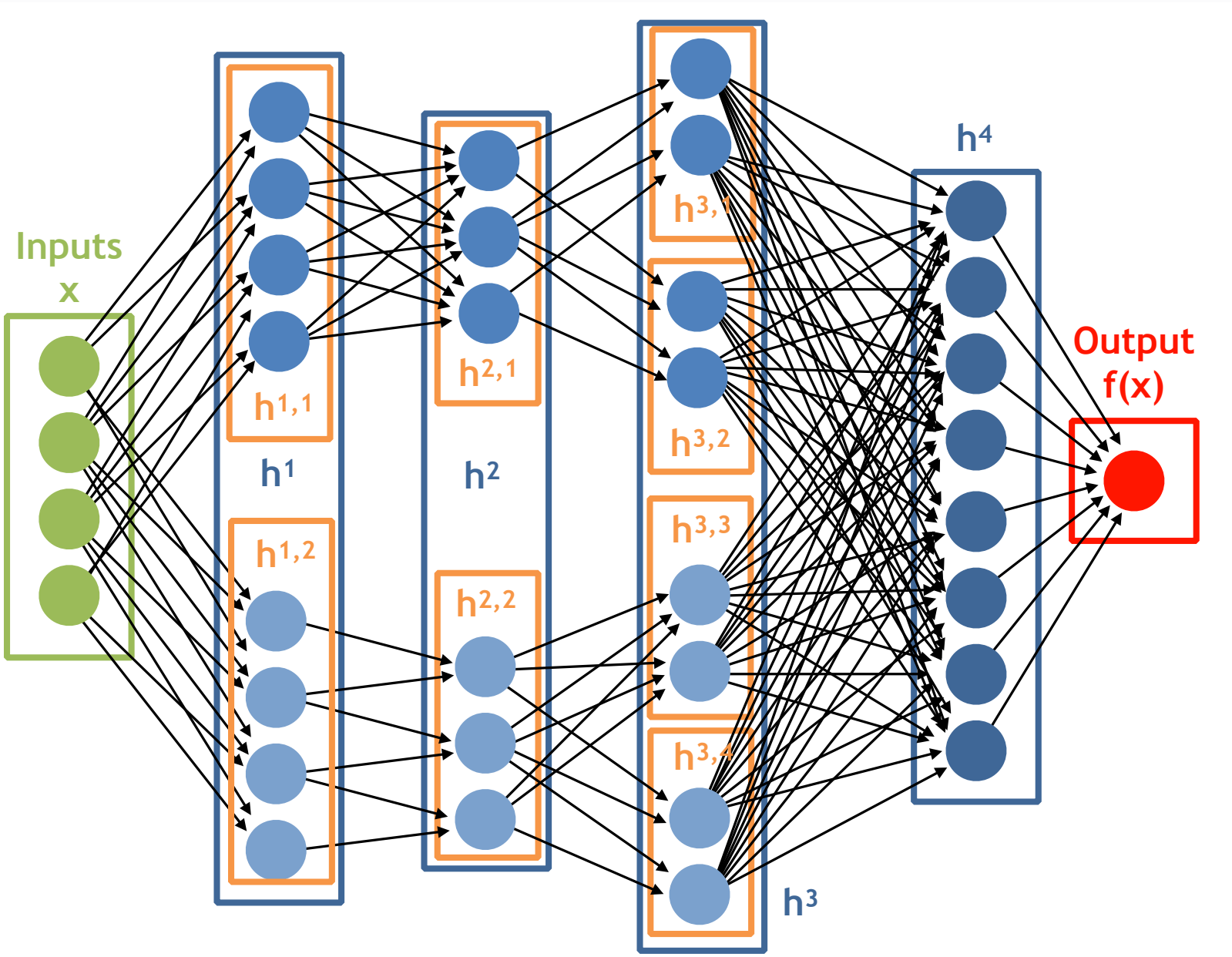
Demo

http://playground.tensorflow.org/

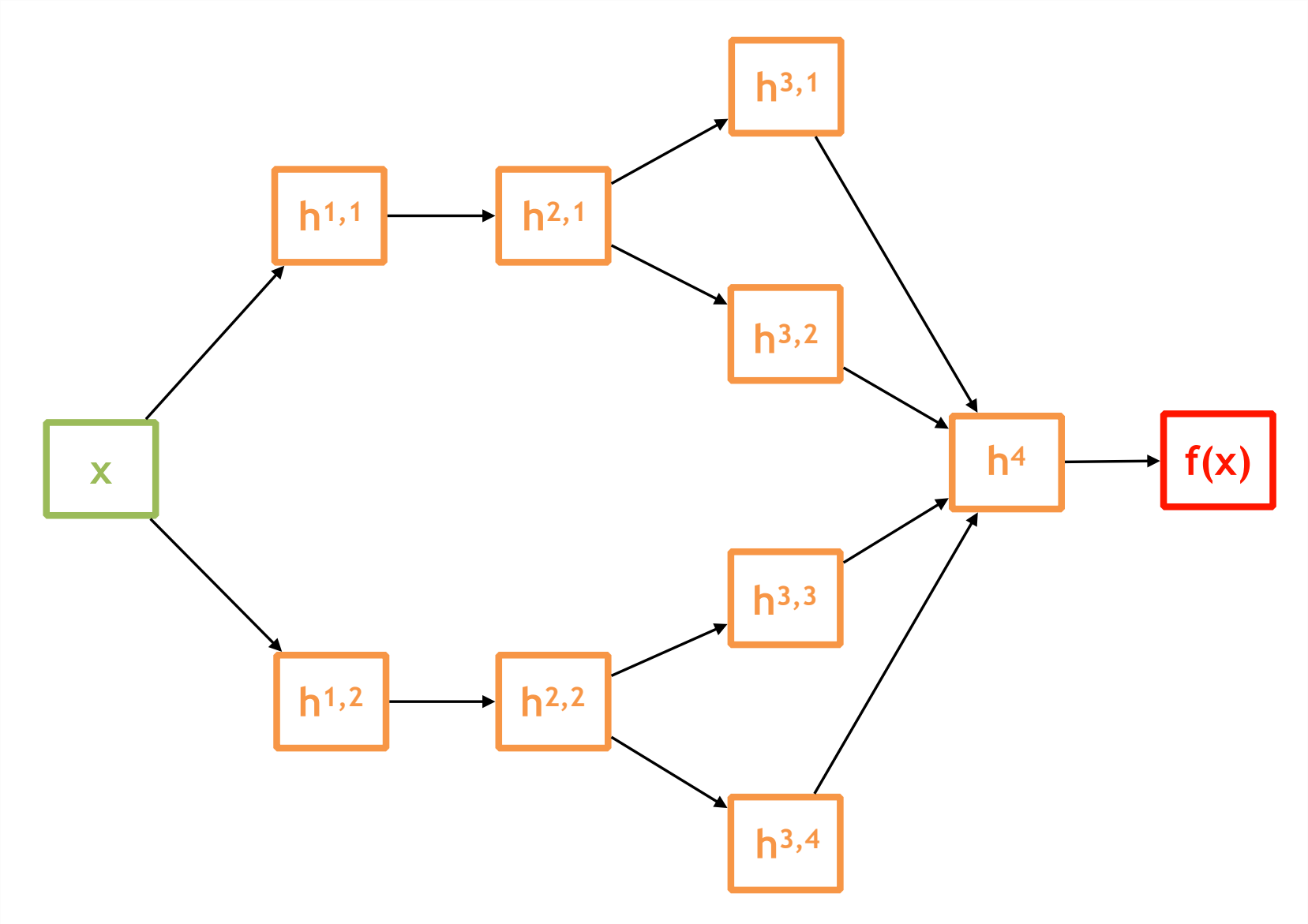
Distinct Blocks



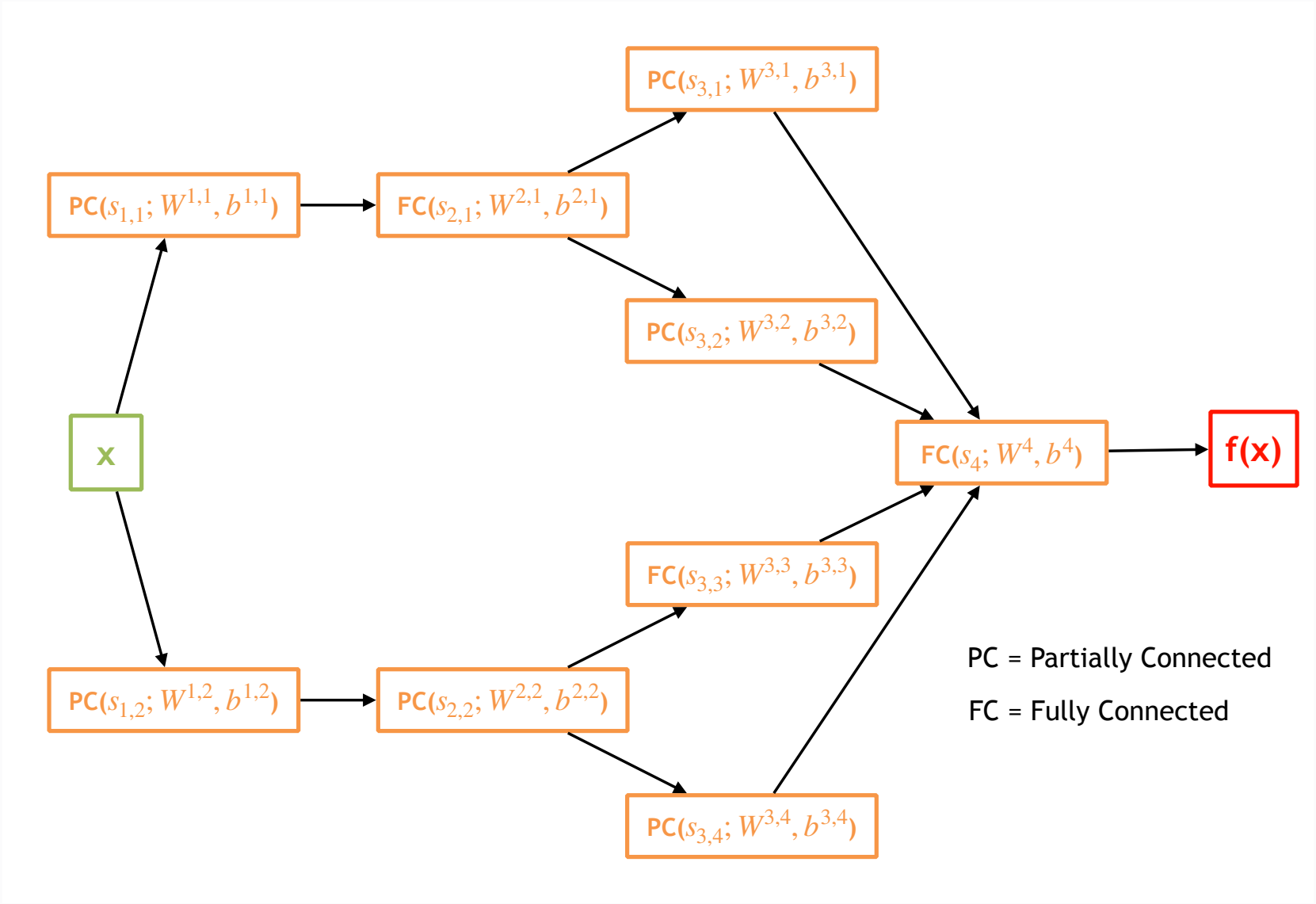
Distinct Blocks



Distinct Blocks Computational Graph



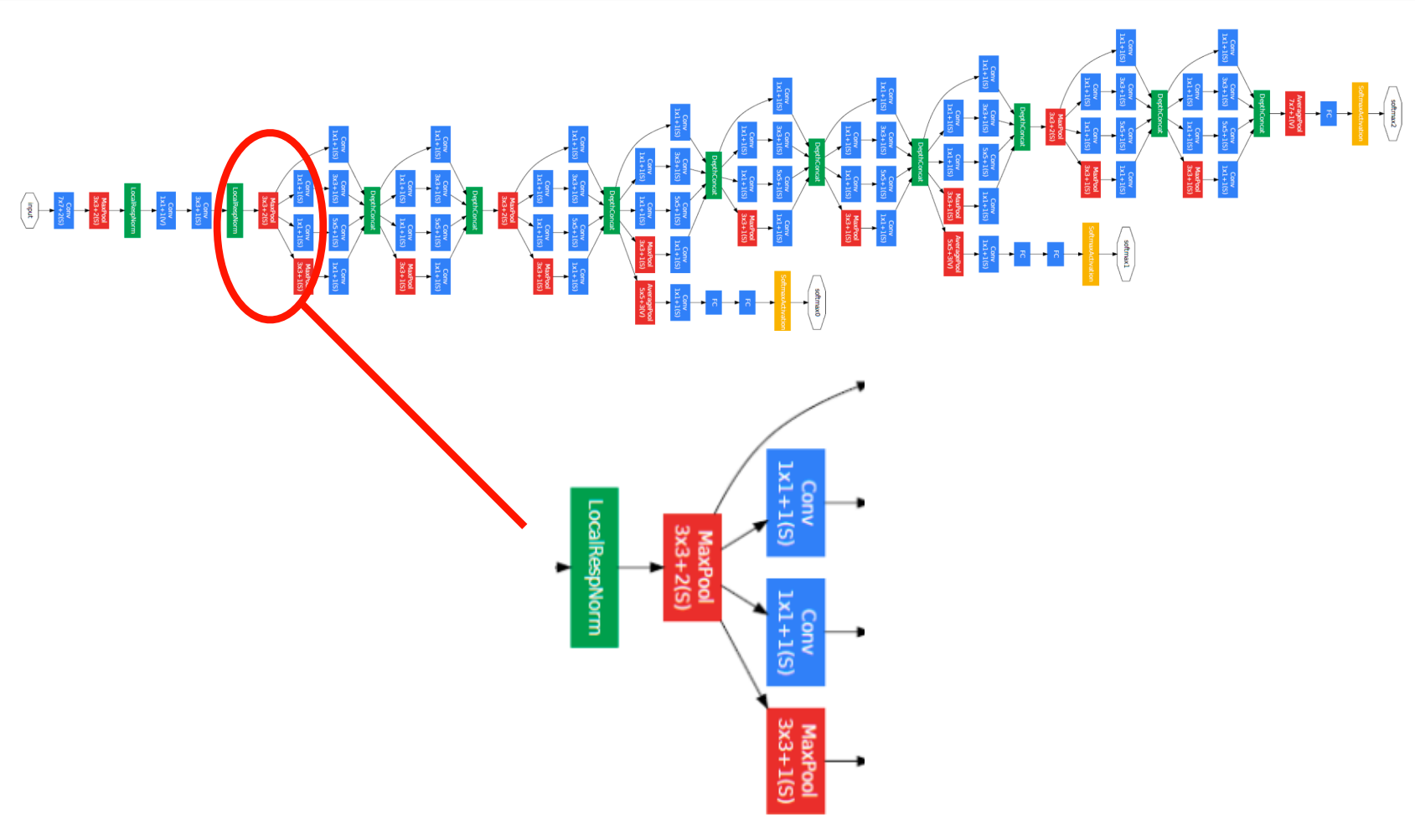
Distinct Blocks Computational Graph



Is this Enough?

- We have gone from general differentiable functions last lecture to functions that are just compositions of linear sums and fixed element–wise nonlinearities
- We do not need to make this restriction; we may want to use other building block computations instead
- Almost all alternatives used in practice are still parameterized only by weights and biases: it is rare to have **differentiable** parameters that are not either scaling or addition factors
    - Products, maximums, logarithms, inverses etc do not require additional tunable parameters
- The parameters of virtually every deep learning model is just the collection of its **weights and biases**
- Deep learning is thus simply neural networks with fancy **architectures** (i.e. high–level computational graphs)

Example Architecture: GoogleNet



Going Deeper With Convolutions. Szegedy et al. CVPR 2015

The Need For Depth

- A network with a single, infinitely wide, hidden layer and shared nonlinearity is a universal predictor for many choices of activation function (examples sheet)
- . . . But in practice such methods are limited
  - May need huge number of hidden units
  - Overly strong inductive biases
- Multiple layers allow us to construct more flexible hypothesis classes with much weaker inductive biases
  - Dependency on choice of activation functions is diminished
- Huge body of empirical evidence for the effectiveness of such deep networks, particularly for large quantities of high–dimensional data
- Can think of multiple layers a hierarchy of increasingly detailed features, customized to our dataset through training

Deeper Layers can Detect More Complex Features

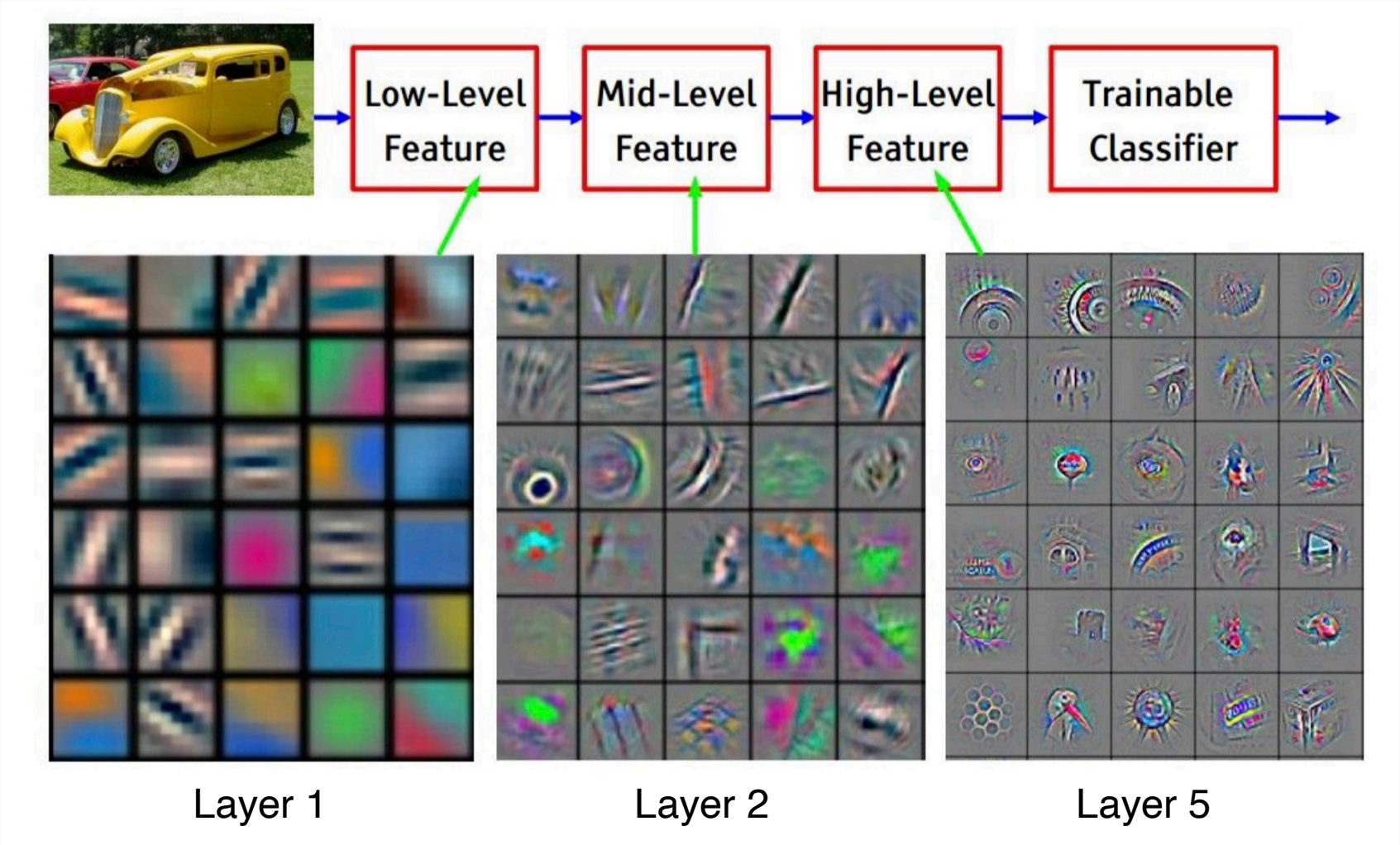


Figure 1: Visualization of features represented by different layers. Individual images in the grid correspond to example dimensions of a h^{ℓ}

Deeper Layers can Detect More Complex Features

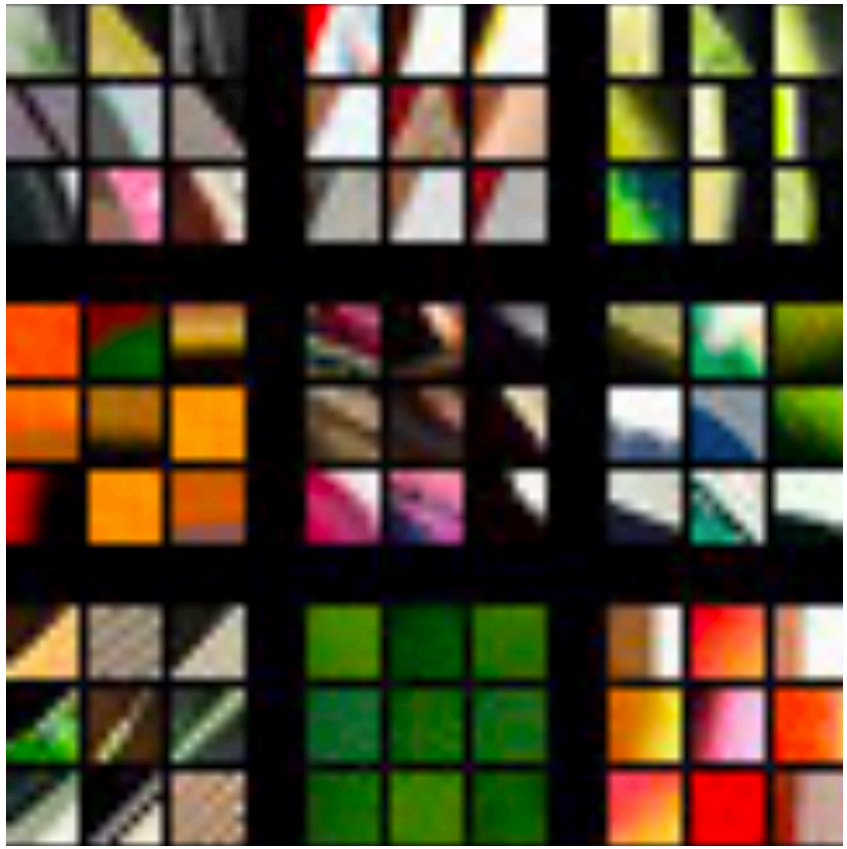


Figure 2: Inputs that produce highest node output for different nodes. Taken from, Visualizing and understanding CNNs, Zeiler and Fergus, ECCV 2013

Deeper Layers can Detect More Complex Features

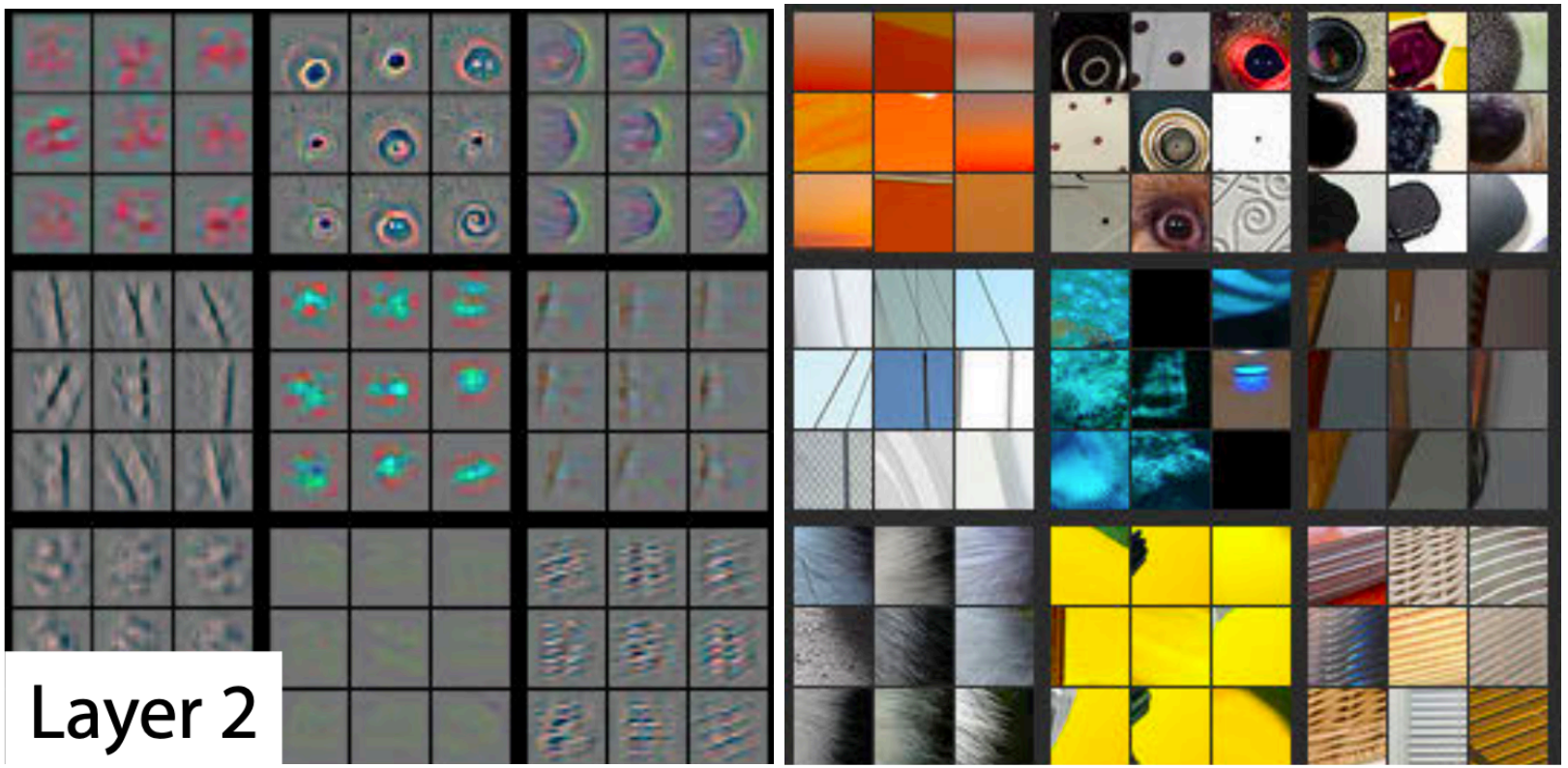


Figure 2: Right: Inputs that produce highest node output for different nodes. Left: Visualization of what causes the triggering. Taken from, Visualizing and understanding CNNs, Zeiler and Fergus, ECCV 2013

Deeper Layers can Detect More Complex Features

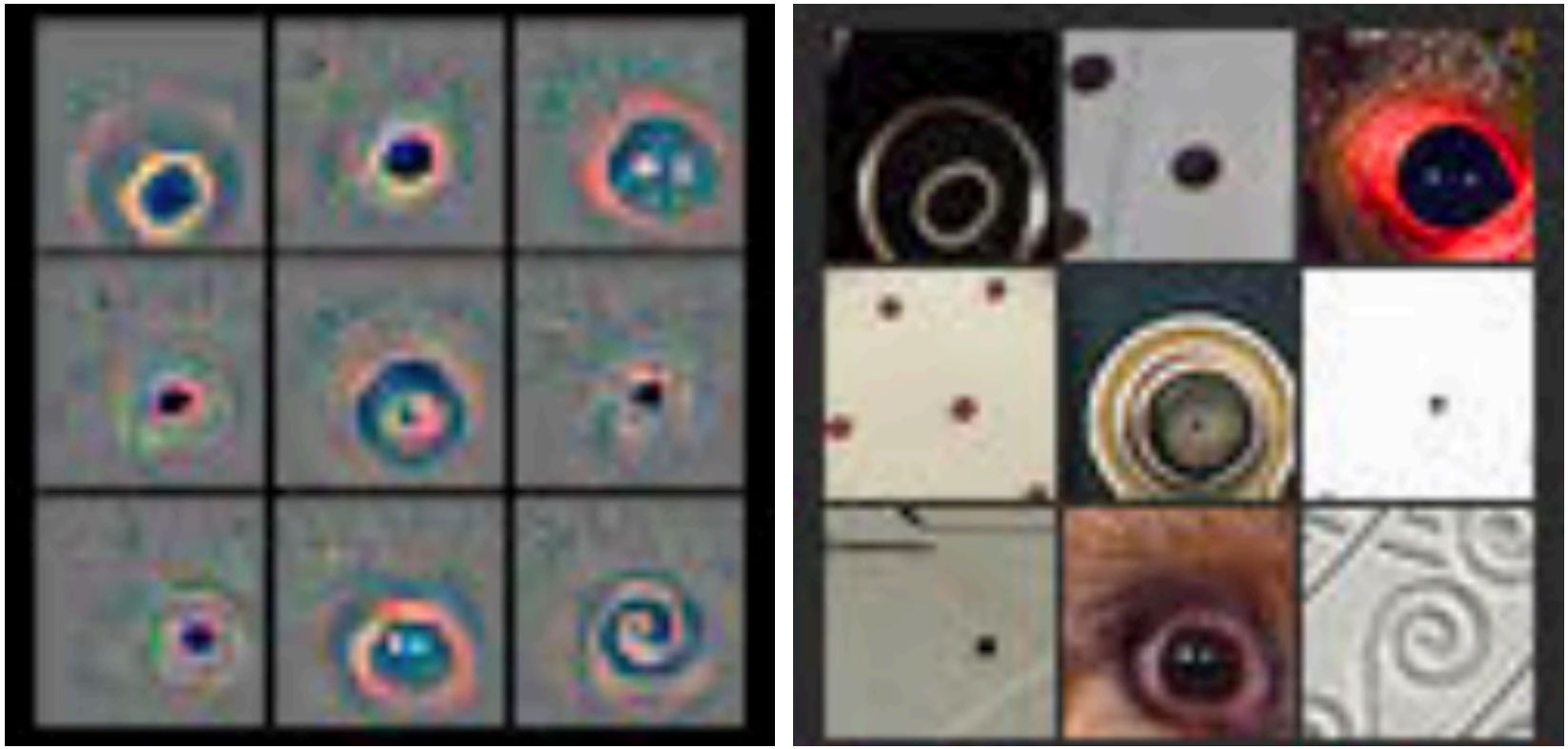


Figure 2: Right: Inputs that produce highest node output for different nodes. Left: Visualization of what causes the triggering. Taken from, Visualizing and understanding CNNs, Zeiler and Fergus, ECCV 2013

Deeper Layers can Detect More Complex Features

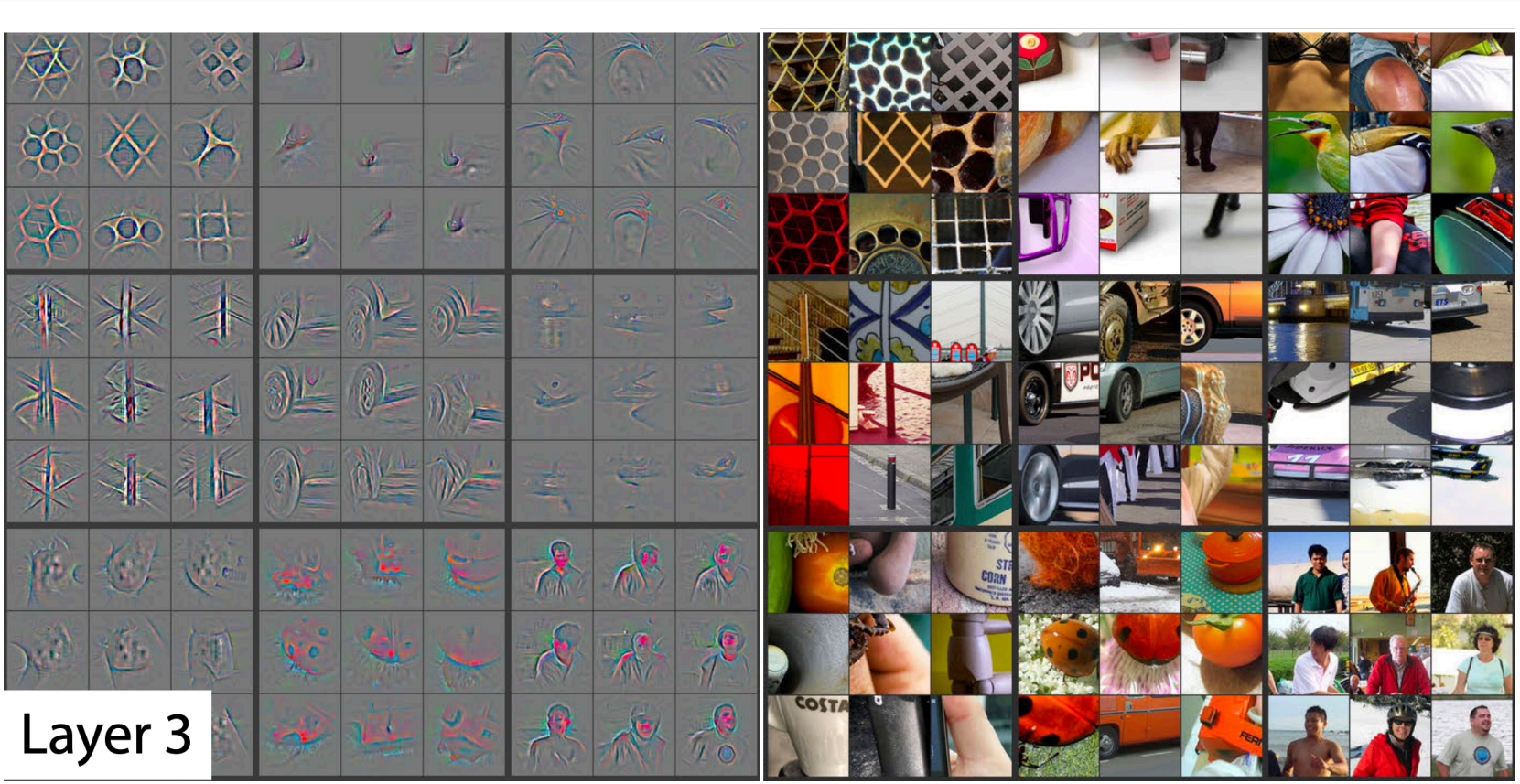


Figure 2: Right: Inputs that produce highest node output for different nodes. Left: Visualization of what causes the triggering. Taken from, Visualizing and understanding CNNs, Zeiler and Fergus, ECCV 2013

Deeper Layers can Detect More Complex Features

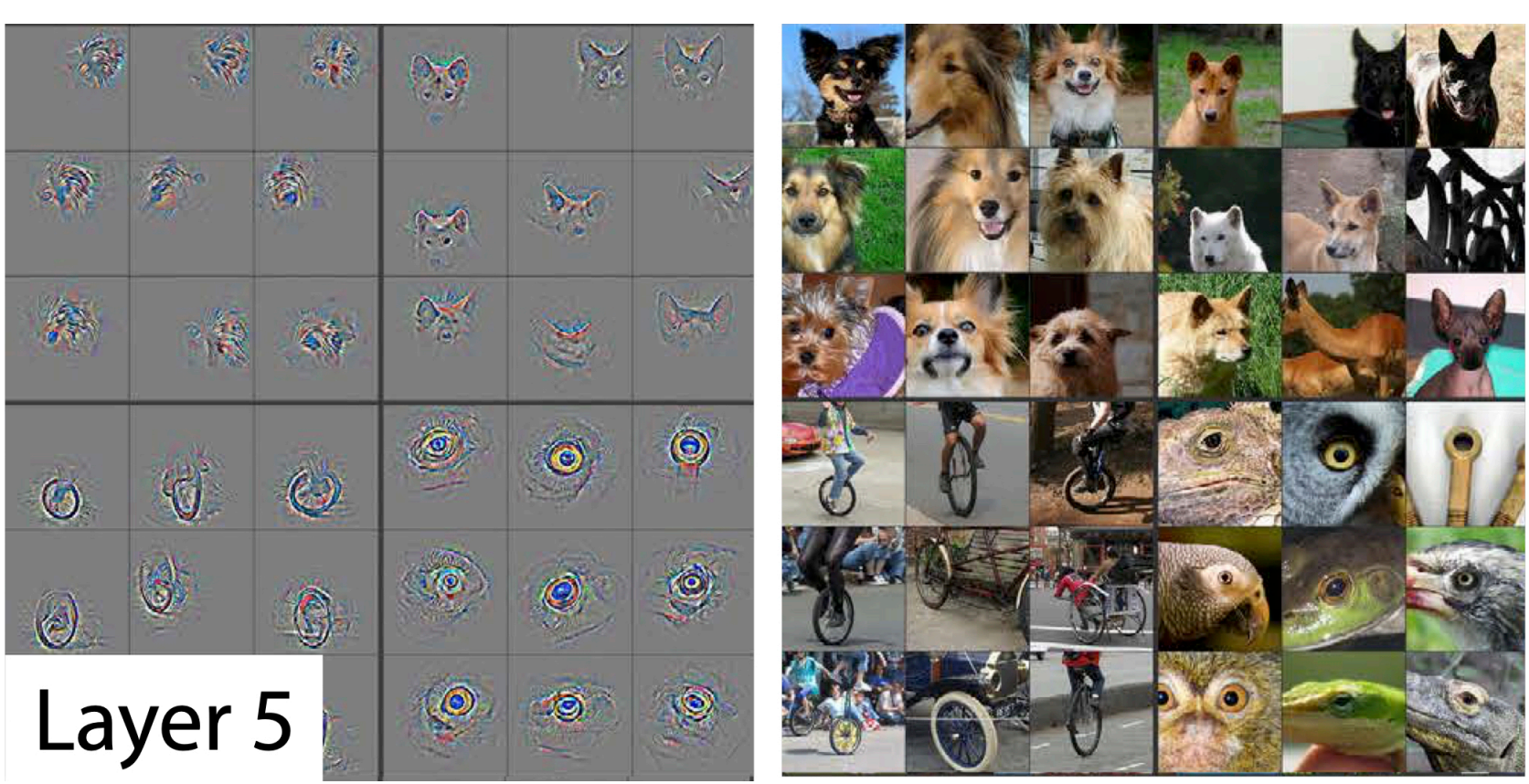


Figure 2: Right: Inputs that produce highest node output for different nodes. Left: Visualization of what causes the triggering. Taken from, Visualizing and understanding CNNs, Zeiler and Fergus, ECCV 2013

Further Visualization

https://youtu.be/AgkfIQ4IGaM

Adding Features to Images

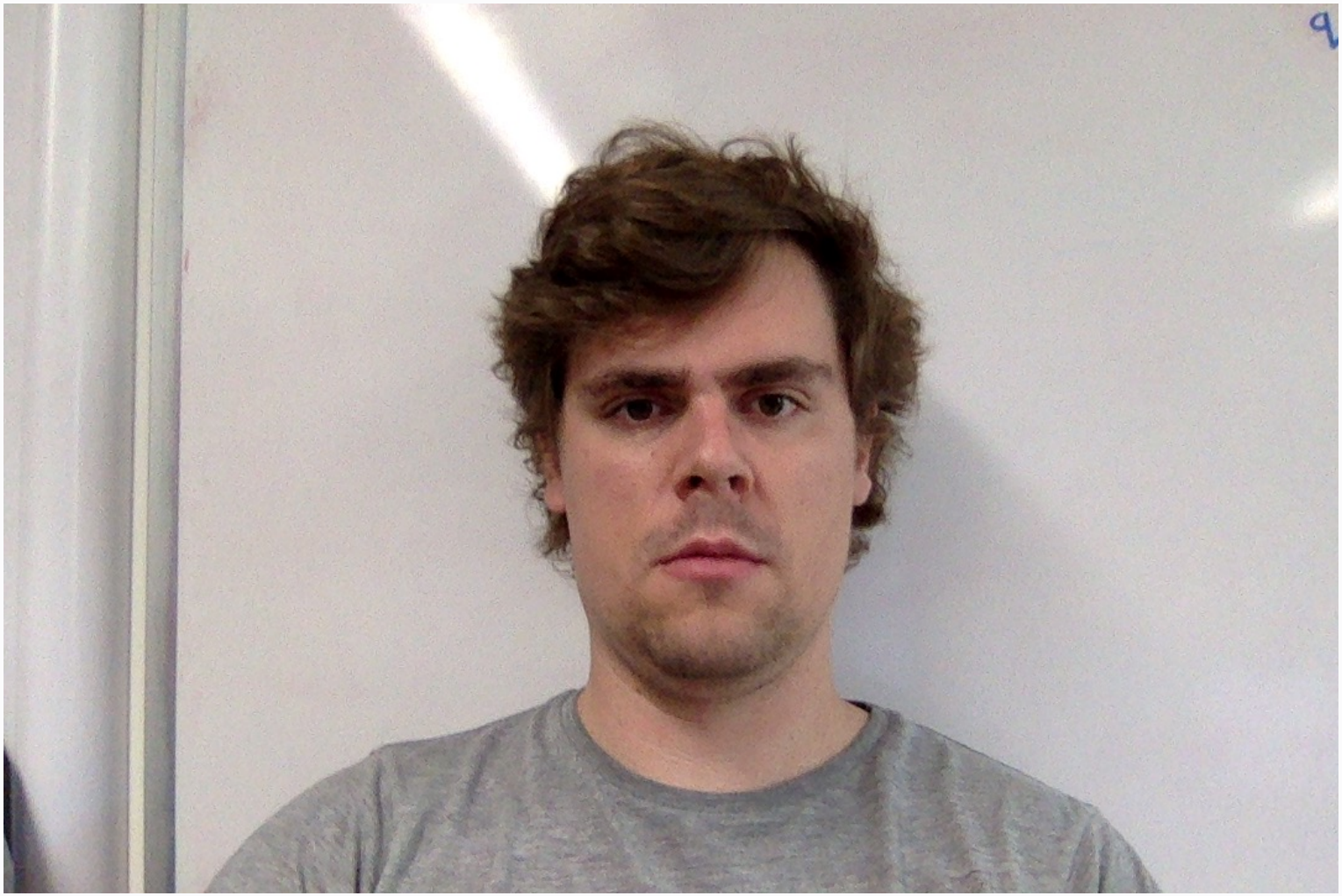


Figure 3: Original input image. We will take regularized gradient steps in the image itself to maximize the output of a particular network node.

Adding Features to Images

Figure 3: Applying the "DeepDream" approach at Layer 1 in a 25 layer network. Generated using `https://deepdreamgenerator.com/`.

Adding Features to Images

Figure 3: Applying the "DeepDream" approach at Layer 2 in a 25 layer network. Generated using `https://deepdreamgenerator.com/`.

Adding Features to Images

Figure 3: Applying the "DeepDream" approach at Layer 6 in a 25 layer network. Generated using `https://deepdreamgenerator.com/`.

Adding Features to Images

Figure 3: Applying the "DeepDream" approach at Layer 12 in a 25 layer network. Generated using `https://deepdreamgenerator.com/`.

Adding Features to Images

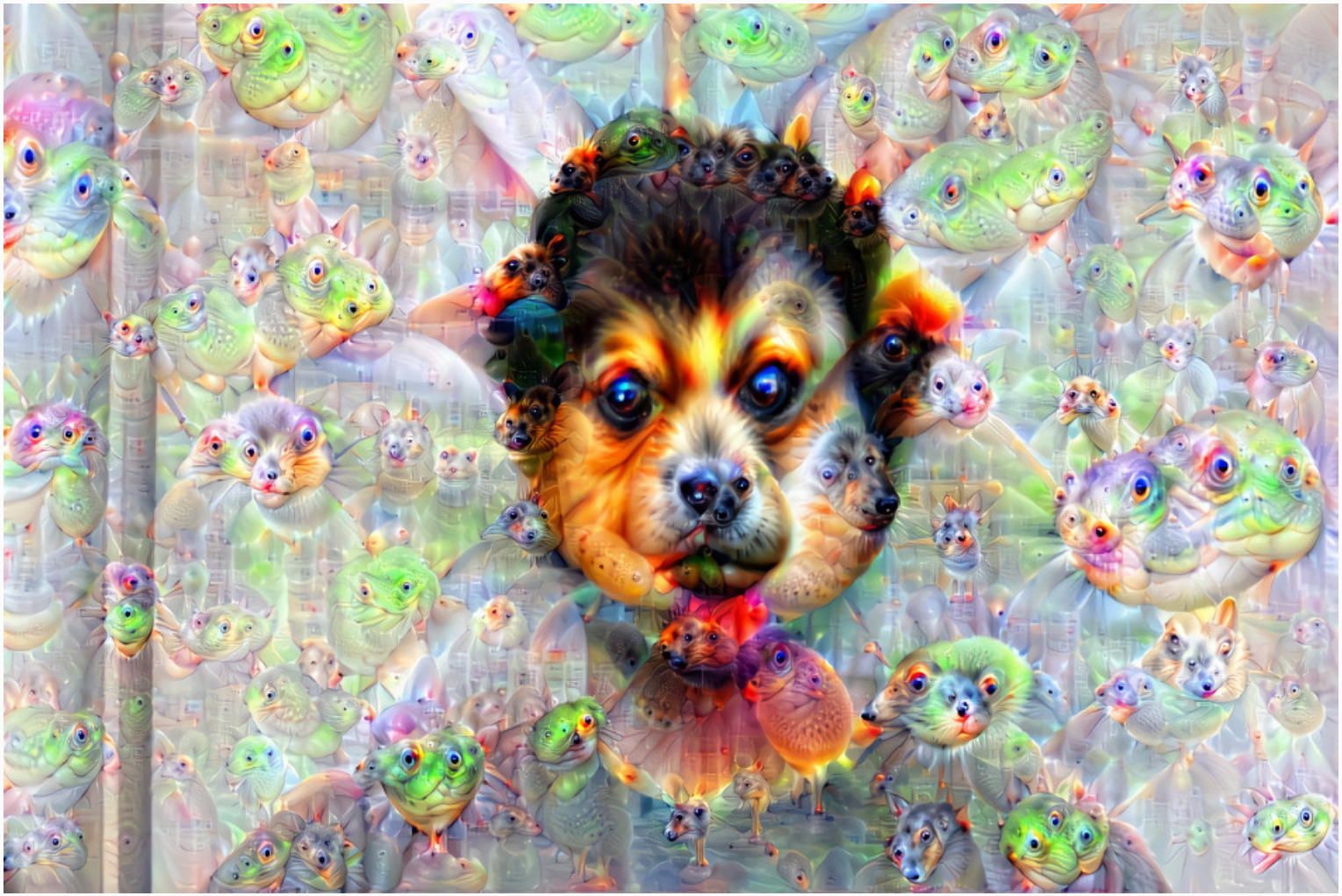


Figure 3: Applying the "DeepDream" approach at Layer 15 in a 25 layer network. Generated using `https://deepdreamgenerator.com/`.

Adding Features to Images

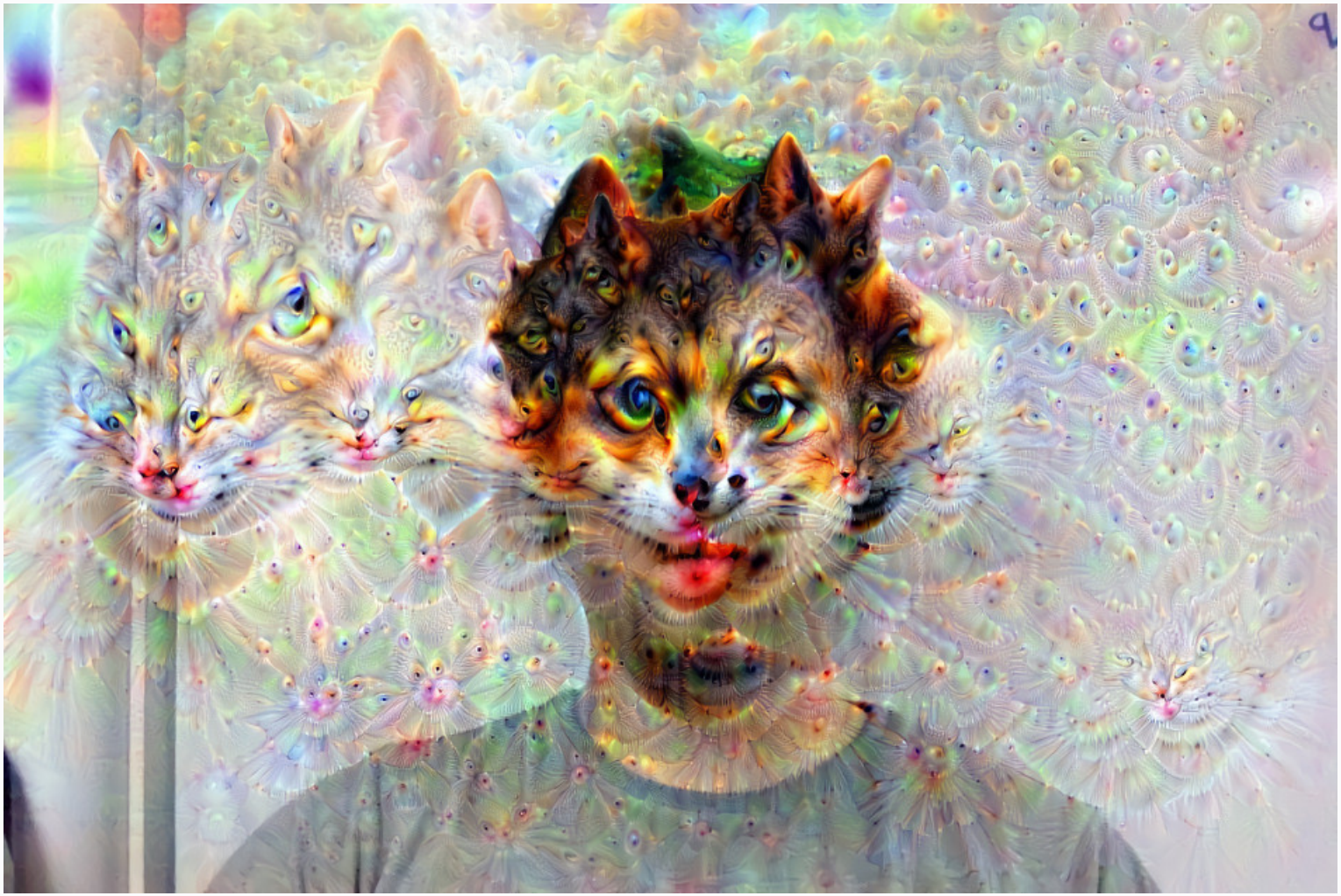


Figure 3: Applying the "DeepDream" approach at Layer 20 in a 25 layer network. Generated using `https://deepdreamgenerator.com/`.

Stylizing Images

Convert the above image to the style on the right

Stylizing Images

Stylizing Images

Or in the style of Renoir

A Starry Radcliffe

Stylized Portraits

Generated using

`deepdreamgenerator.com`

Recap

- Deep learning and neural networks are essentially the **same thing**, but the former conveys more modern motivations
- Neural networks are built up of **layers** of **nodes/units** with connections between nodes in consecutive layers
- Nonlinearities are usually fixed and applied **element–wise**, such that the model is formed by alternating linear mappings and scalar nonlinearities
- The vast majority of networks are parameterized entirely by their connection **weights** and node **biases**
- Having **deep** networks with many layers provides more flexible hypothesis classes that form hierarchies of features
- Very strong **empirical evidence** supporting their use

Further Reading

- Chapter 6 of Ian Goodfellow, Yoshua Bengio, and Aaron Courville. **Deep Learning**. `http://www.deeplearningbook.org`. MIT Press, 2016
- Nice intuitive introduction to neural networks `https://youtu.be/aircAruvnKk`
- Deep learning introductory lecture by Wojciech Czarnecki `https://youtu.be/FBggC-XVF4M`
- Massive dump of deep learning resources `https://awesomeopensource.com/project/ChristosChristofidis/awesome-deep-learning`