COMP 433 Lecture 3
Intro to Neural Networks

Practical Matters

- Lab 2 posted (due next week)

- Lab 3 will be posted this night

- October 5 (to be confirmed) - First quiz - 40 mins

Quiz Schedule

- October 5 - 40 mins
- November TBD (30-45 min)
- December TBD (30-45 min)

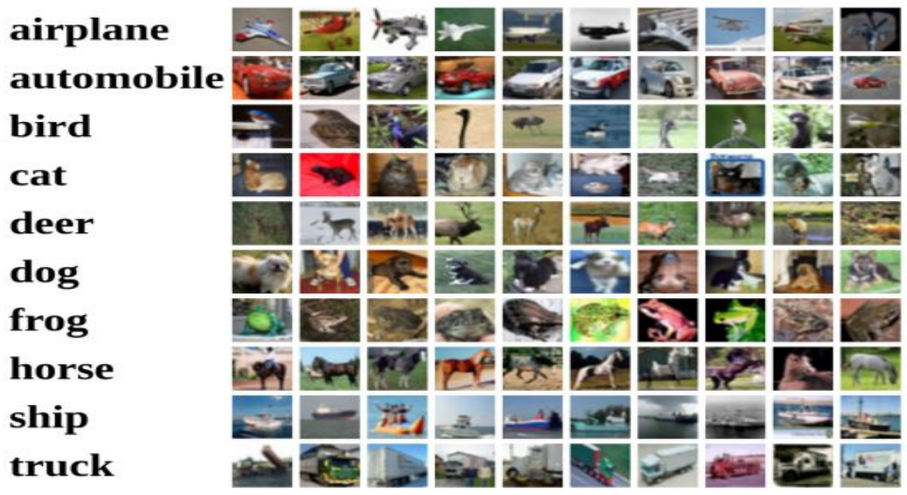
Deep Learning

Key aspects of deep learning models to understand

- Neural Networks - universal function approximator
- Representation Learning - avoid hand-crafted features
- Modular but powerful/expressive model framework

Neural Networks

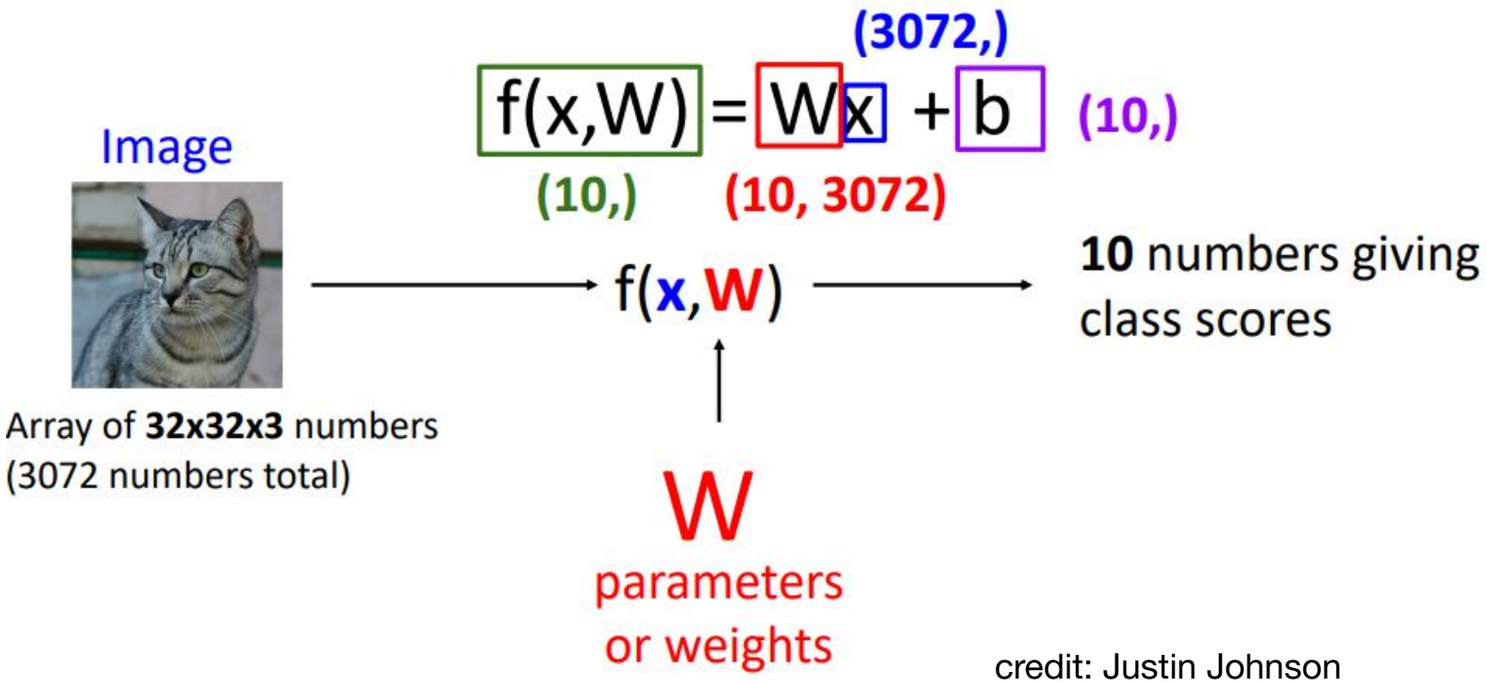
- CIFAR 10 dataset



50,000 training images
each image is **32x32x3**

10,000 test images.

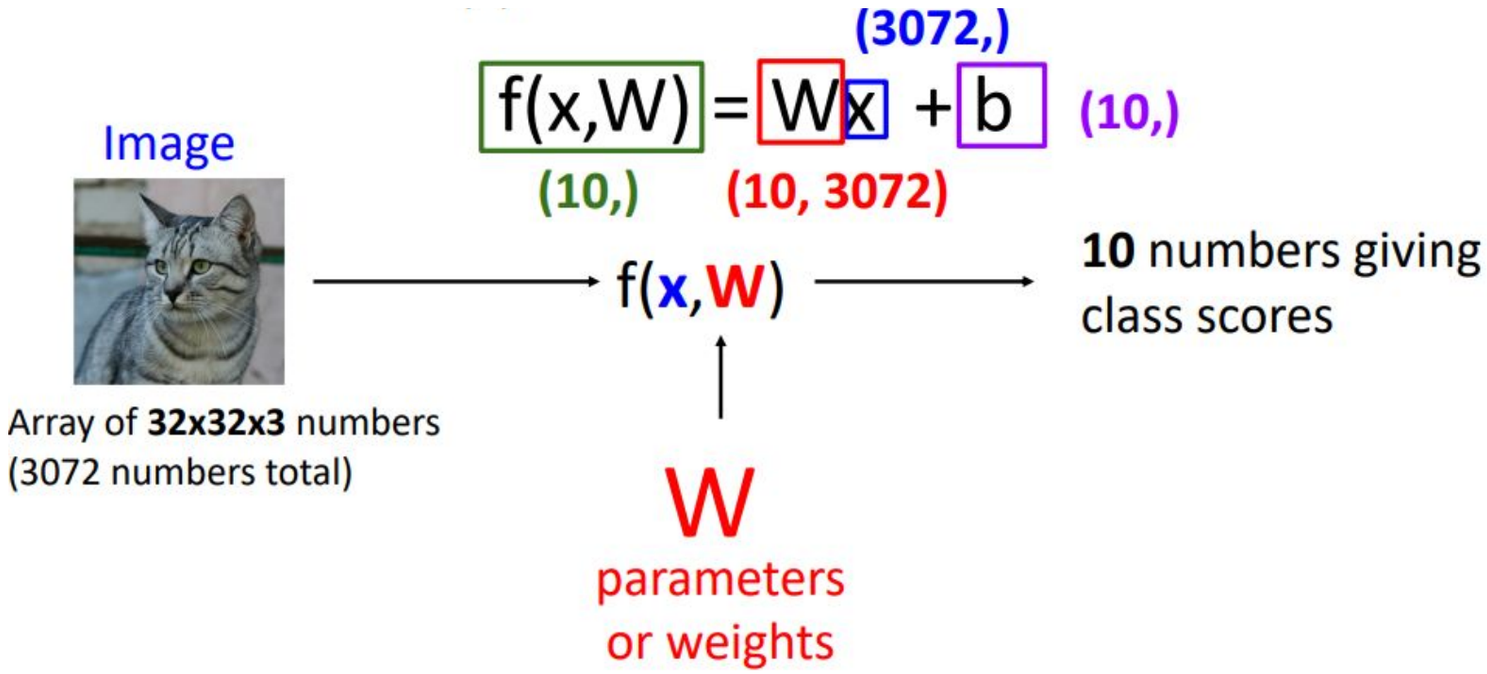
- Image classification: linear classifiers



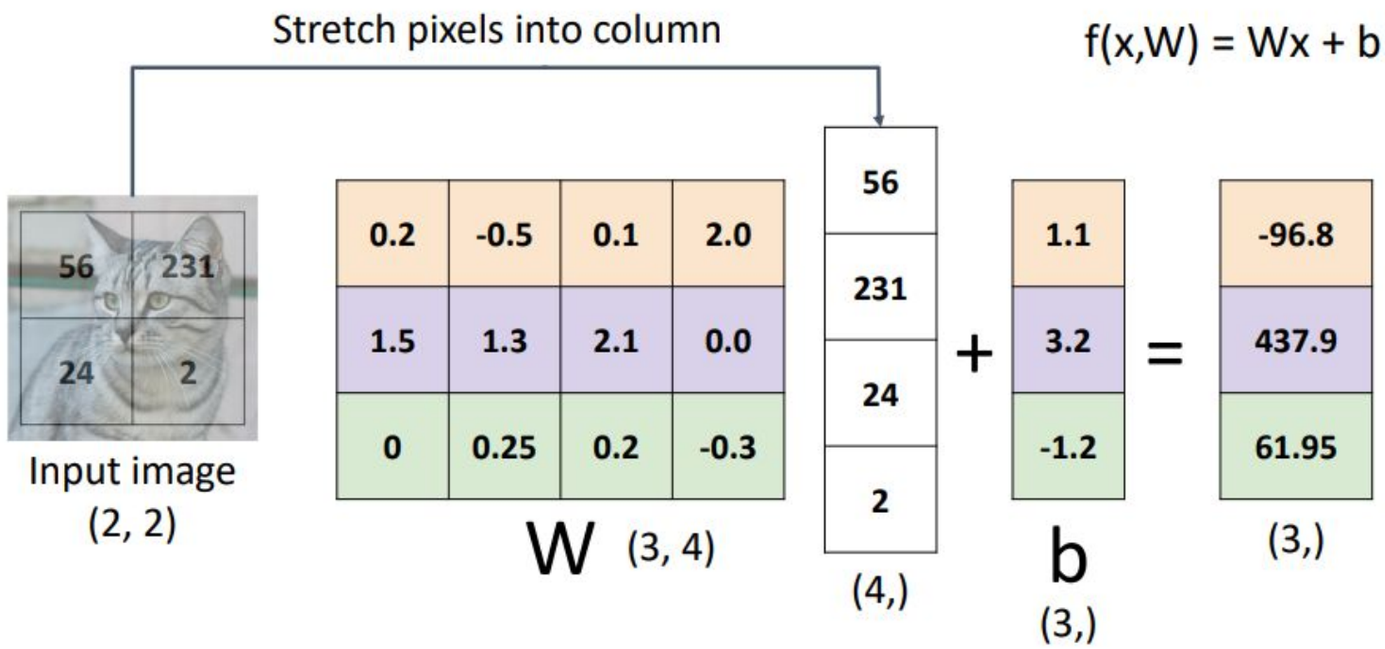
credit: Justin Johnson

Neural Networks

- Image classification: linear classifiers (algebraic point)

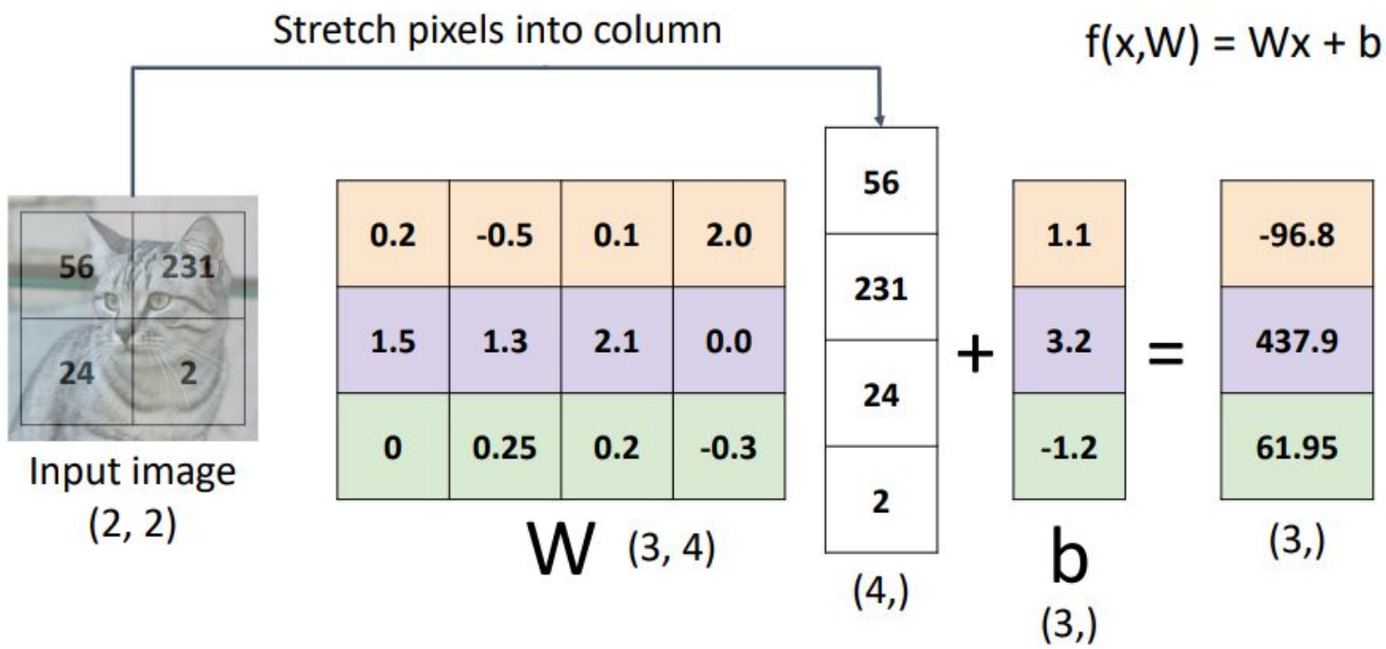


- Image classification: linear classifiers (algebraic point)
  - Example for **3 classes and considering only images of size 2x2 (unrealistic)**

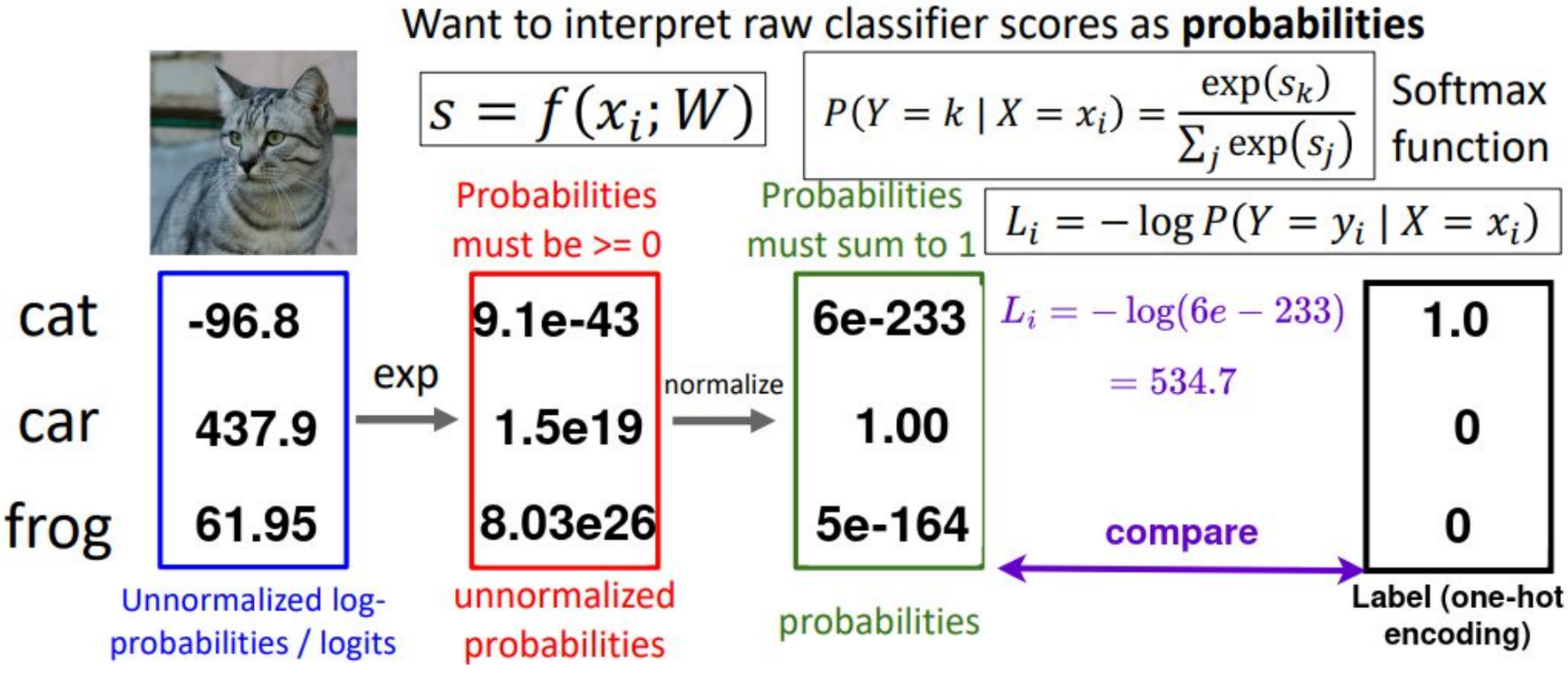


Neural Networks

- Image classification: linear classifiers (algebraic point)
  - Example for **3 classes and considering only images of size 2x2 (unrealistic)**

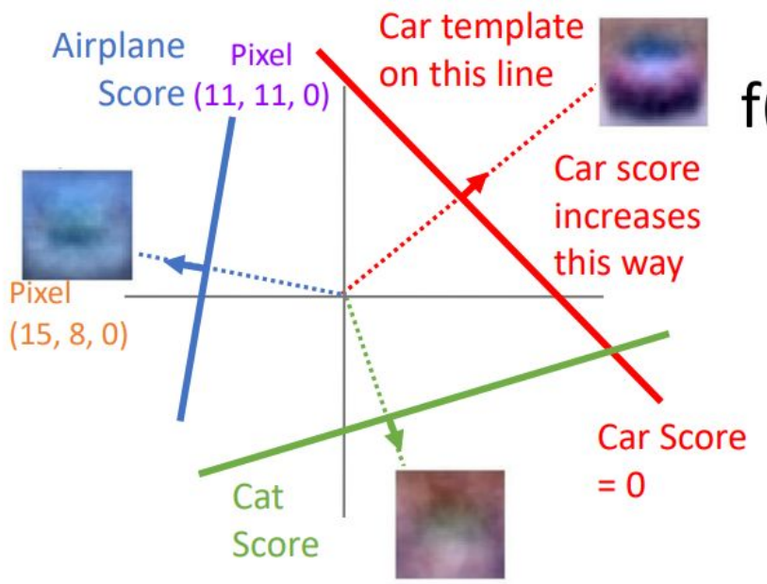


- Linear classifier (multinomial logistic regression) with loss function



Neural Networks

- Image classification: linear classifiers (geometric point)



$f(x,W) = Wx + b$

Array of **32x32x3** numbers
(3072 numbers total)

credit: Justin Johnson

- Linear classifier (multinomial logistic regression) with loss function

Want to interpret raw classifier scores as **probabilities**

$s = f(x_i; W)$

$P(Y = k \mid X = x_i) = \frac{\exp(s_k)}{\sum_j \exp(s_j)}$ Softmax function

$L_i = -\log P(Y = y_i \mid X = x_i)$

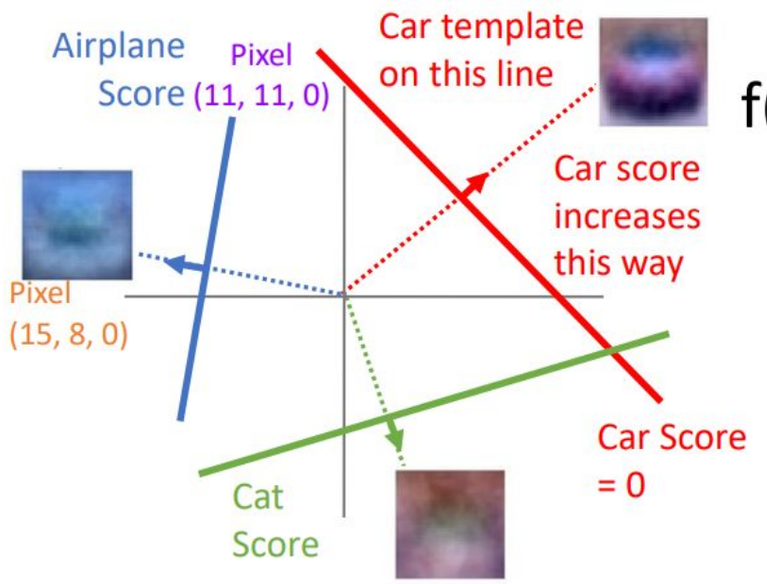
	Unnormalized log-probabilities / logits	Probabilities must be >= 0 (unnormalized probabilities)	Probabilities must sum to 1 (probabilities)	Label (one-hot encoding)
cat	-96.8	9.1e-43	6e-233	1.0
car	437.9	1.5e19	1.00	0
frog	61.95	8.03e26	5e-164	0

exp → normalize → compare

$L_i = -\log(6e - 233) = 534.7$

Neural Networks

- Image classification: linear classifiers (geometric point)



$f(x,W) = Wx + b$

Array of **32x32x3** numbers
(3072 numbers total)

credit: Justin Johnson

- Linear classifier (multinomial logistic regression) with loss function

Want to interpret raw classifier scores as **probabilities**

$s = f(x_i; W)$

$P(Y = k \mid X = x_i) = \frac{\exp(s_k)}{\sum_j \exp(s_j)}$ Softmax function

$L_i = -\log P(Y = y_i \mid X = x_i)$

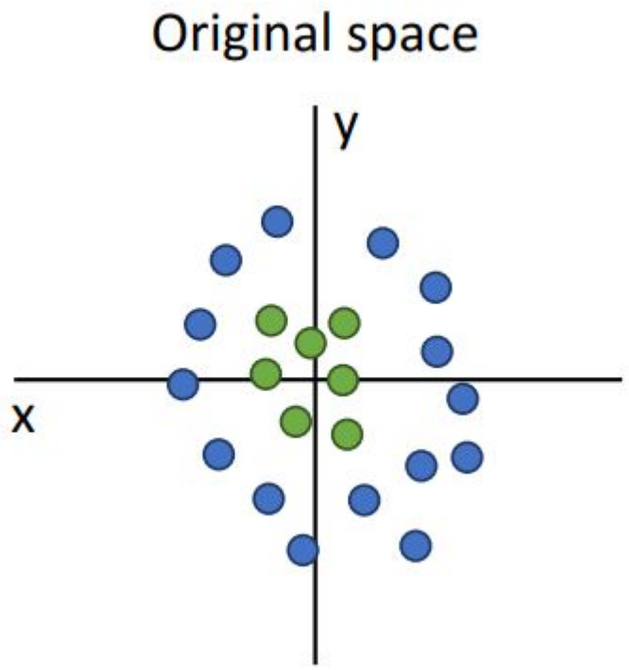
	Unnormalized log-probabilities / logits	Probabilities must be >= 0 (unnormalized probabilities)	Probabilities must sum to 1 (probabilities)	Label (one-hot encoding)
cat	-96.8	9.1e-43	6e-233	1.0
car	437.9	1.5e19	1.00	0
frog	61.95	8.03e26	5e-164	0

exp → normalize → compare

$L_i = -\log(6e - 233) = 534.7$

Neural Networks

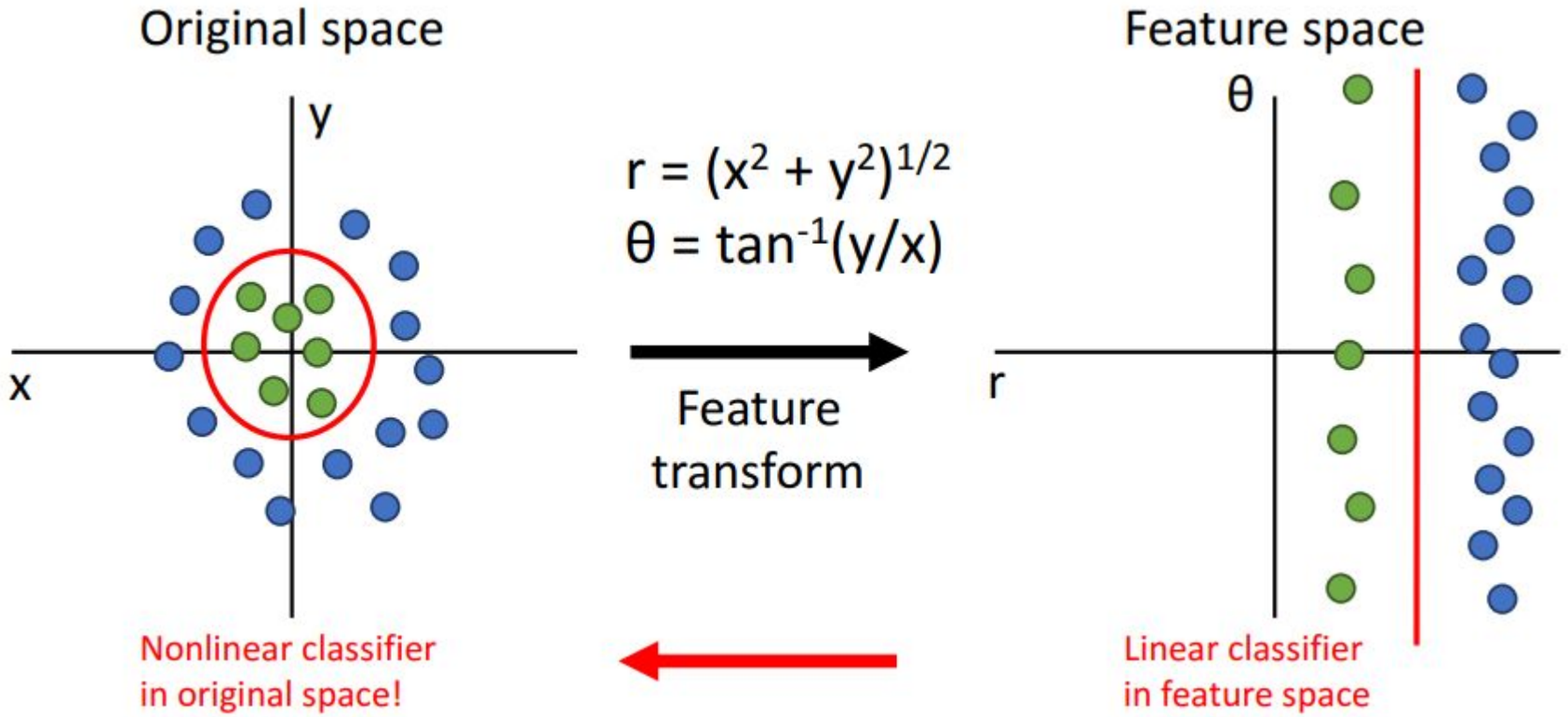
- Image classification: linear classifiers
- Linear classifiers aren’t powerful in for some tasks!



credit: Justin Johnson

Neural Networks

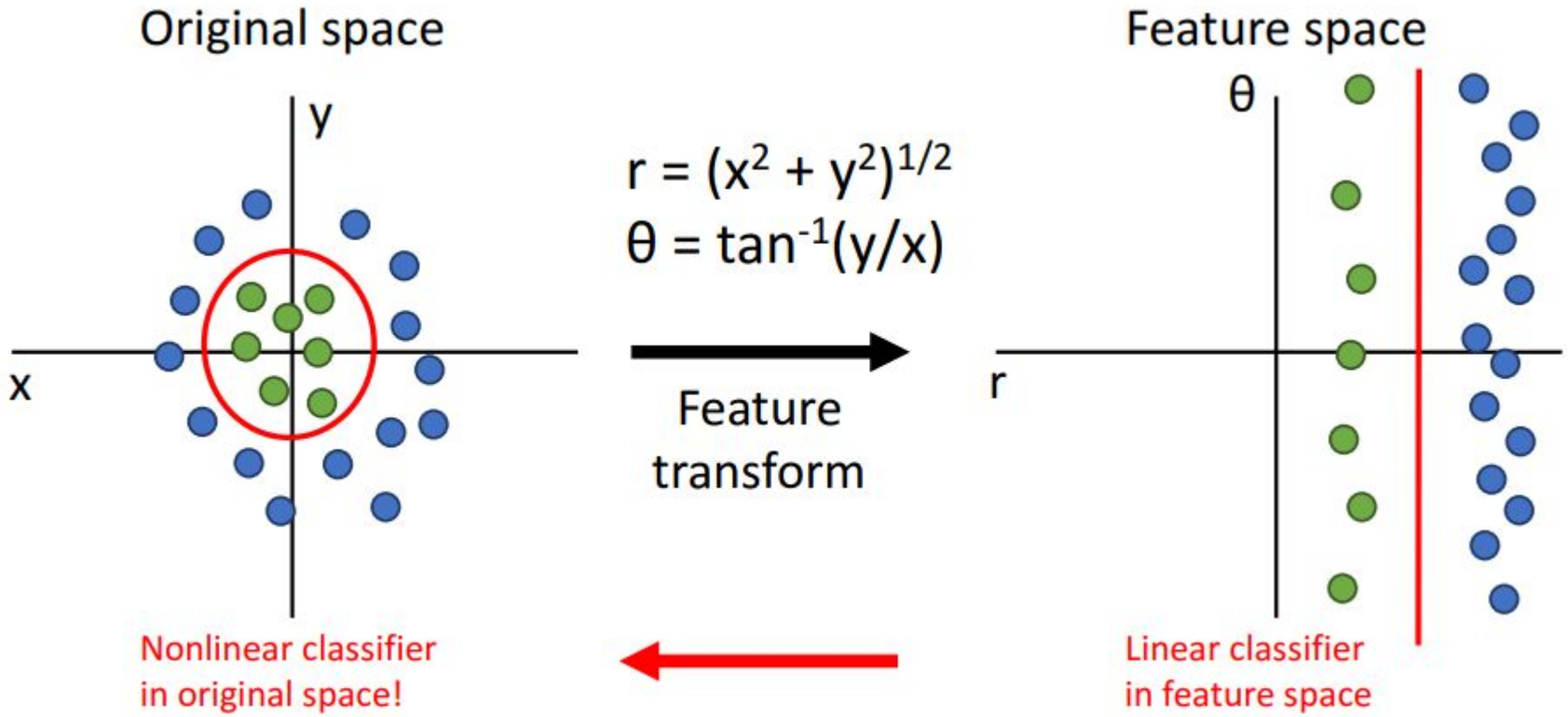
- Linear classifiers aren't powerful in for some tasks!
- Solution: Feature Transformations!



credit: Justin Johnson

Neural Networks

- Image classification: linear classifiers
  - Linear classifiers aren't powerful!
  - Solution: Feature Transformations!



credit: Justin Johnson

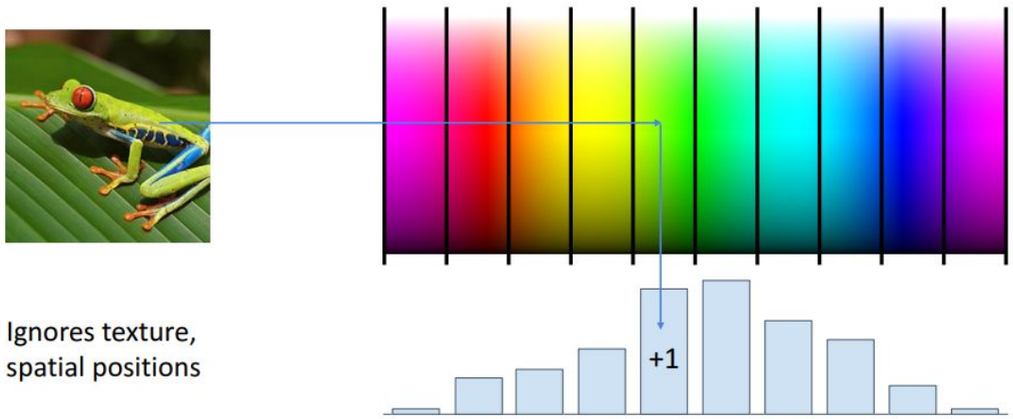
Neural Networks

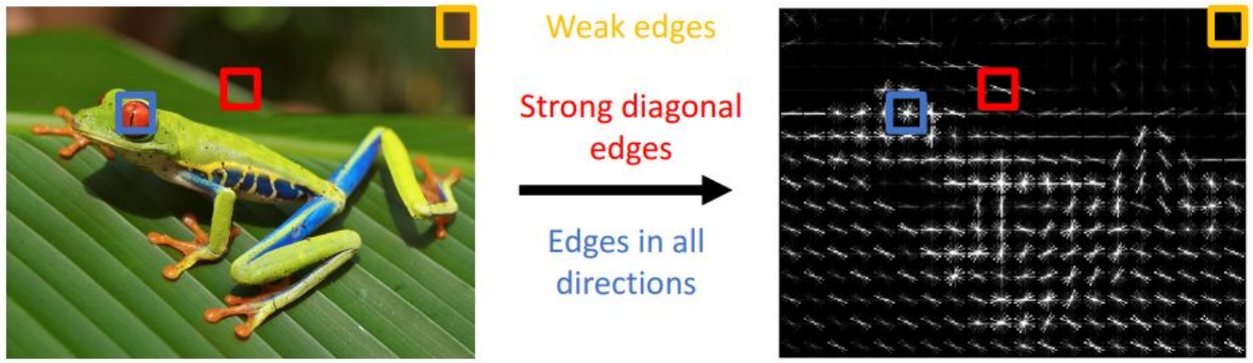
- Image classification:
  - Linear classifiers aren’t powerful!
  - How to extract powerful features from (image) data?

credit: Justin Johnson

Neural Networks

- Image classification:
  - Linear classifiers aren't powerful in for some tasks!
  - How to extract powerful features from (image) data?
- Examples of hand-crafted image features
  - Color histogram, Histogram of Oriented Gradient (HoG)

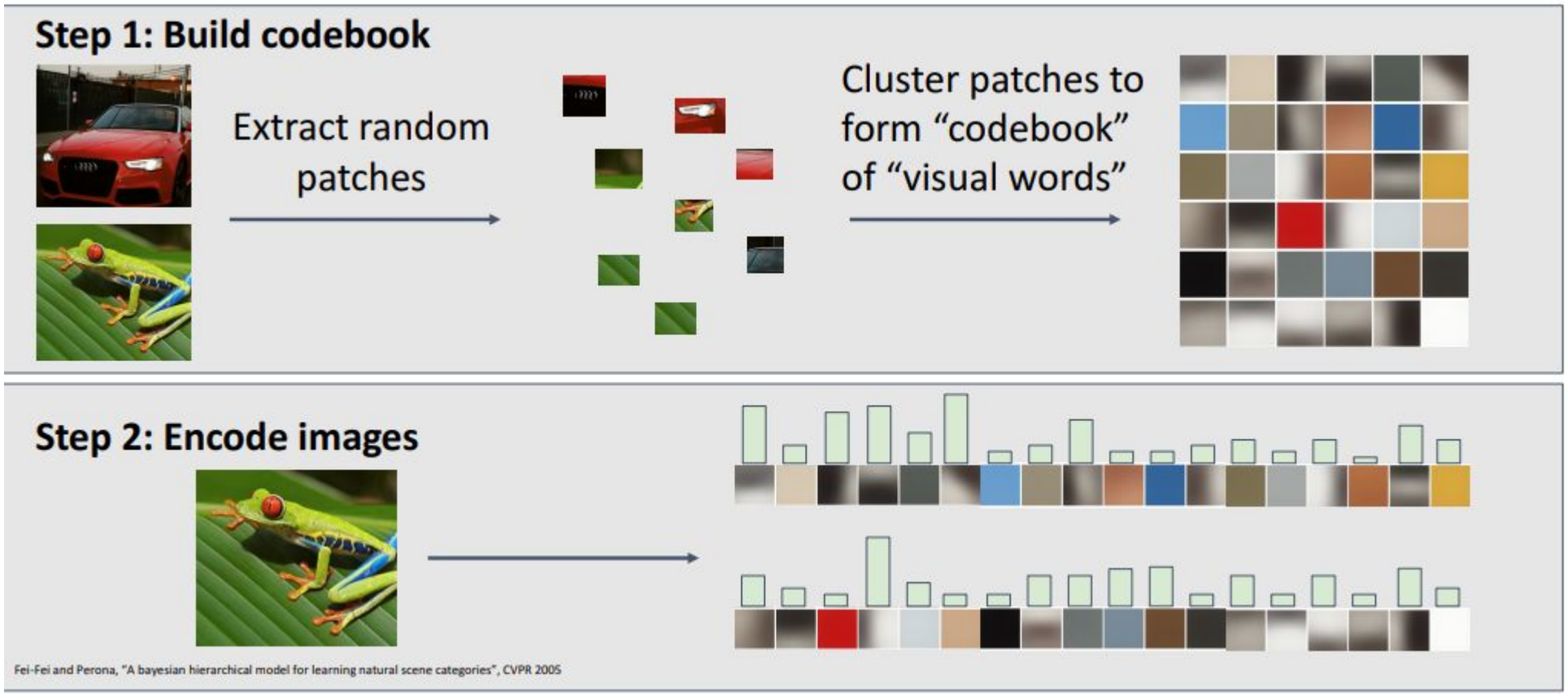




credit: Justin Johnson

Neural Networks

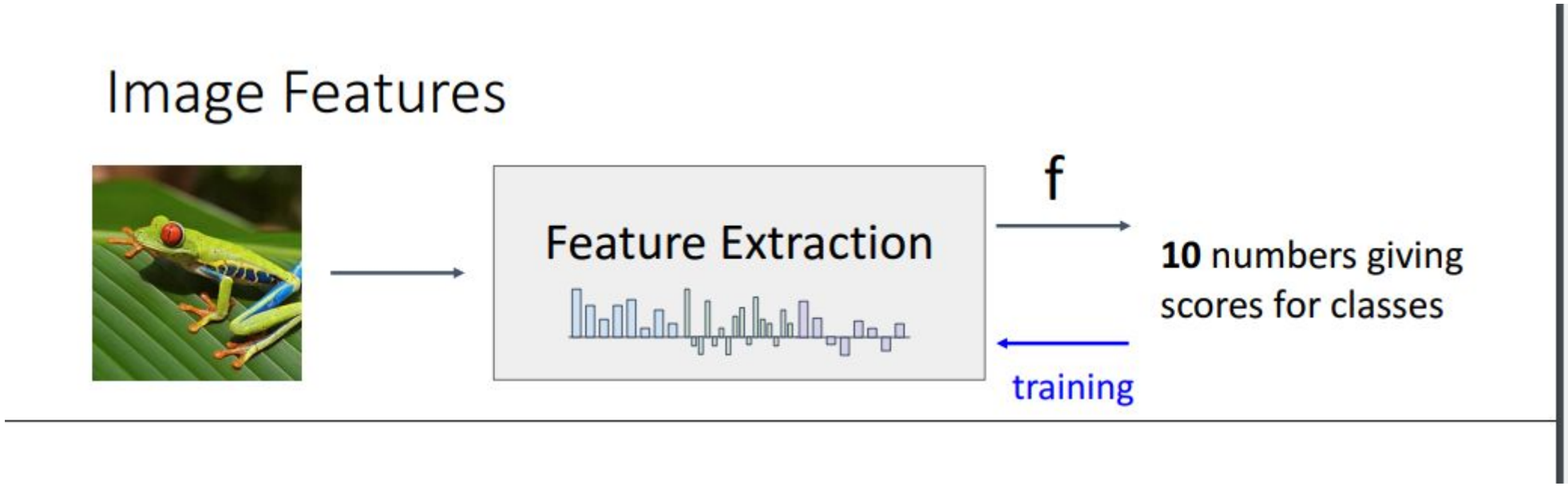
- Image classification:
  - Linear classifiers aren't powerful in for some tasks!
  - How to extract powerful features from (image) data?
- Examples of hand-crafted image features
  - Bag of Words (data-driven)



credit: Justin Johnson

Neural Networks

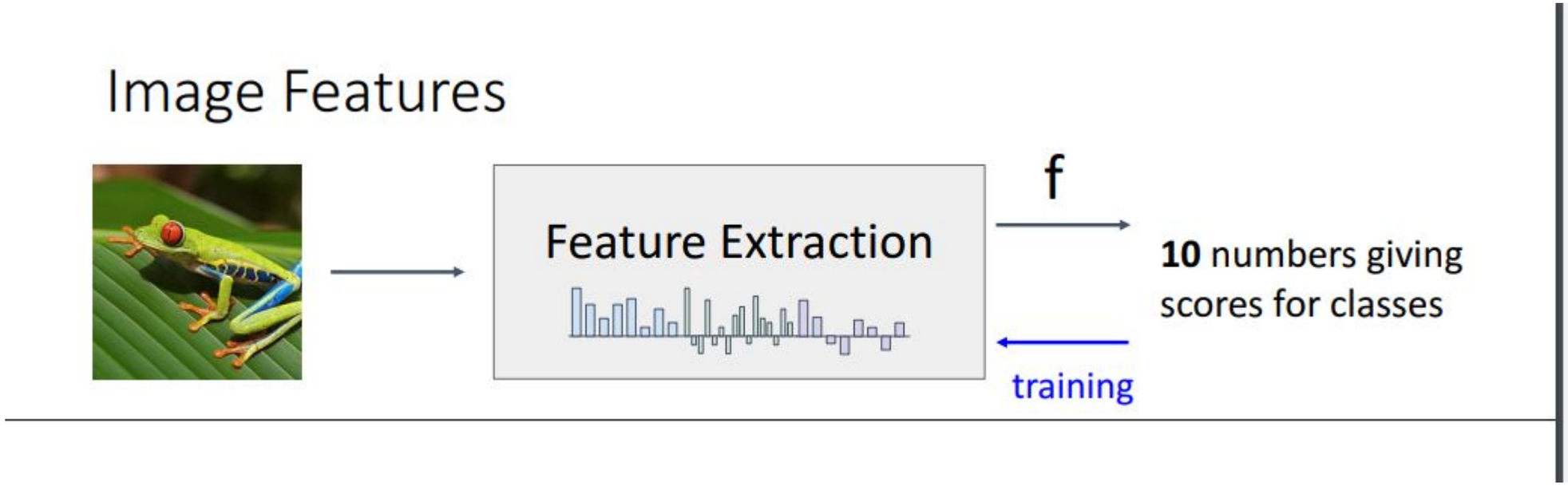
- Image classification:
  - With feature extraction, build a classifier on top of extracted features



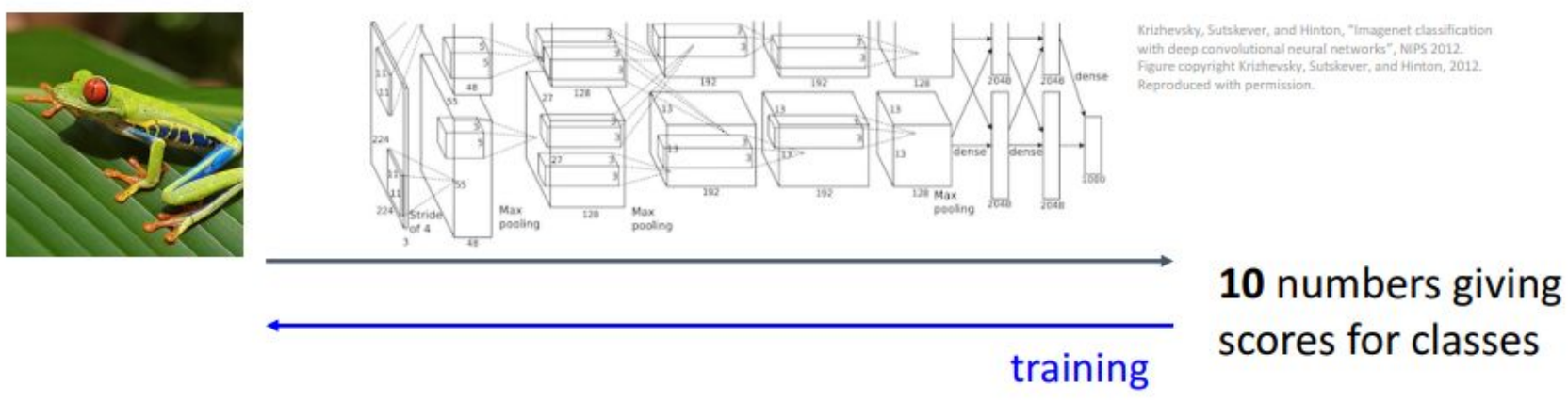
credit: Justin Johnson

Neural Networks

- Image classification:
  - With feature extraction, build a classifier on top of extracted features



- Image classification:
  - The neural network idea: learn end-to-end the features!



Neural Networks Review

- Can be written as composition of simple linear operator + pointwise nonlinearity
  - The pointwise nonlinearity or the activation function here is
    σ(x) = ReLU(x) = max (0,x)
  - Bias terms are omitted for simplicity

(**Before**) Linear score function: $f = Wx$

(**Now**) 2-layer Neural Network $f = W_2 \max(0, W_1 x)$

or 3-layer Neural Network

$$f = W_3 \max(0, W_2 \max(0, W_1 x))$$

$W_3 \in \mathbb{R}^{C \times H_2}$ $W_2 \in \mathbb{R}^{H_2 \times H_1}$ $W_1 \in \mathbb{R}^{H_1 \times D}$ $x \in \mathbb{R}^D$

credit: Justin Johnson

Neural Networks Review

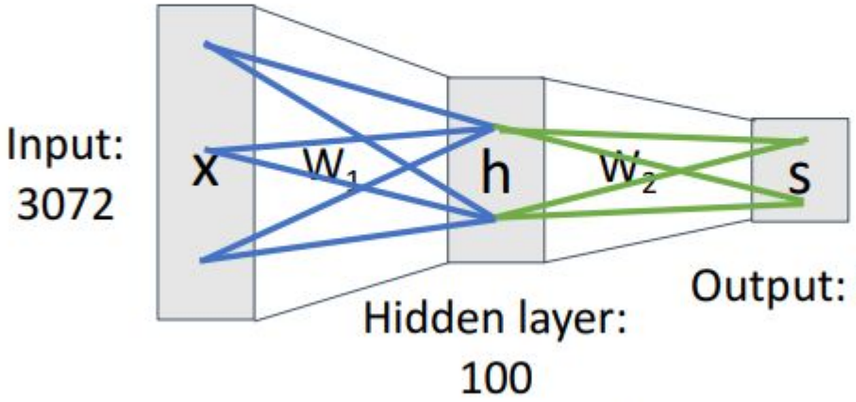
- Can be written as composition of simple linear operator + pointwise nonlinearity
  - The pointwise nonlinearity or the activation function here is
    σ(x) = ReLU(x) = max (0,x)
  - Bias terms are omitted for simplicity

(**Before**) Linear score function: $f = Wx$

(**Now**) 2-layer Neural Network $f = W_2 \max(0, W_1 x)$

Element (i, j) of W_1 gives the effect on h_i from x_j

All elements of x affect all elements of h



Element (i, j) of W_2 gives the effect on s_i from h_j

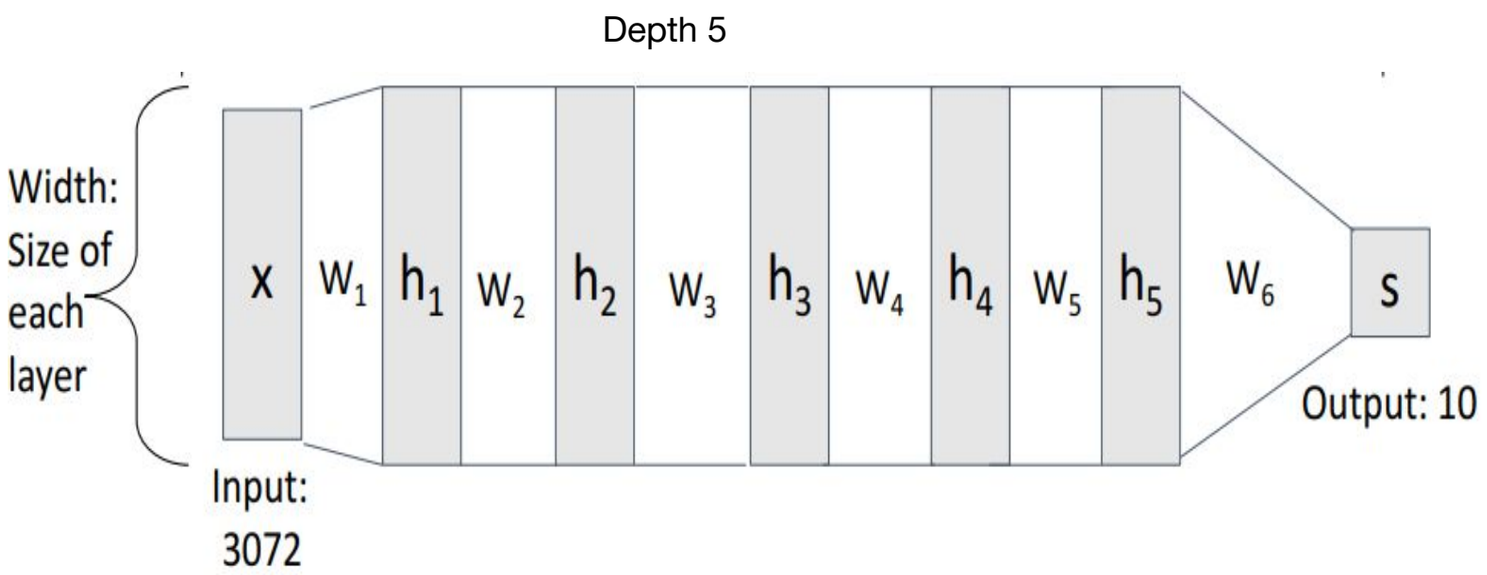
All elements of h affect all elements of s

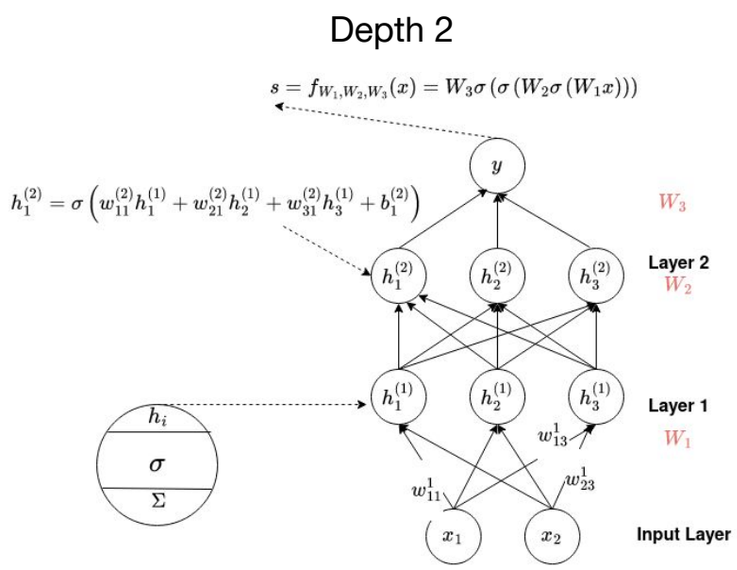
Fully-connected neural network
Also "Multi-Layer Perceptron" (MLP)

credit: Justin Johnson

Neural Networks Review

- Can be written as composition of simple linear operator + pointwise nonlinearity
  - Layers can be stacked together with depth
    - Depth: number of hidden layers (in this course)
    - Width: size of each layer (number of neurons)





$$s = W_6 \max(0, W_6 \max(0, W_5 \max(0, W_4 \max(0, W_3 \max(0, W_2 \max(0, W_1 x))))))$$

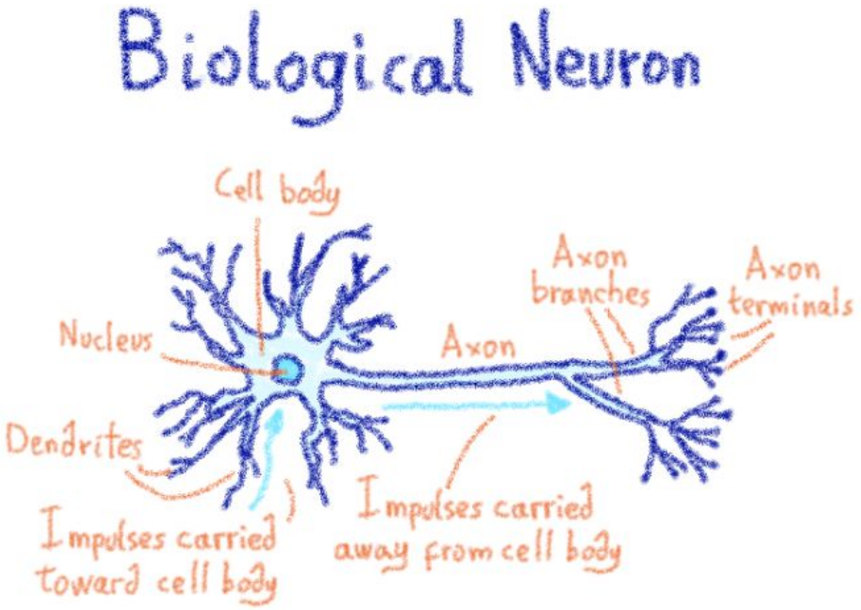
W_i Matrix of parameters at layer i

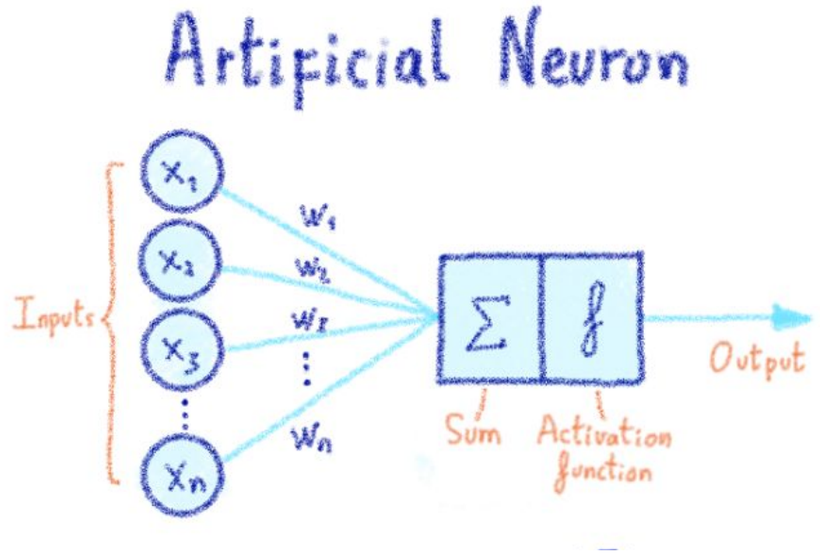
σ Pointwise non-linearity

credit: Justin Johnson

Neural Networks and Bio-Inspiration

- Aspects of neural networks have been historically inspired by biological systems



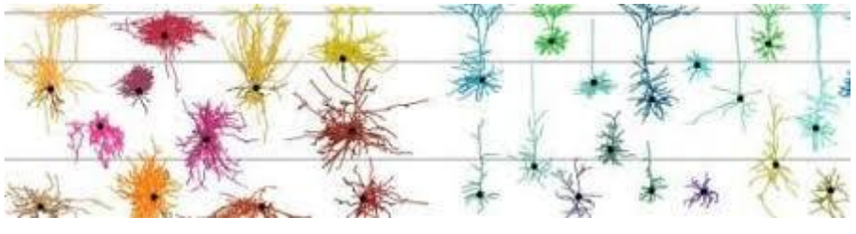


Images from Jean-Louis Queguiner

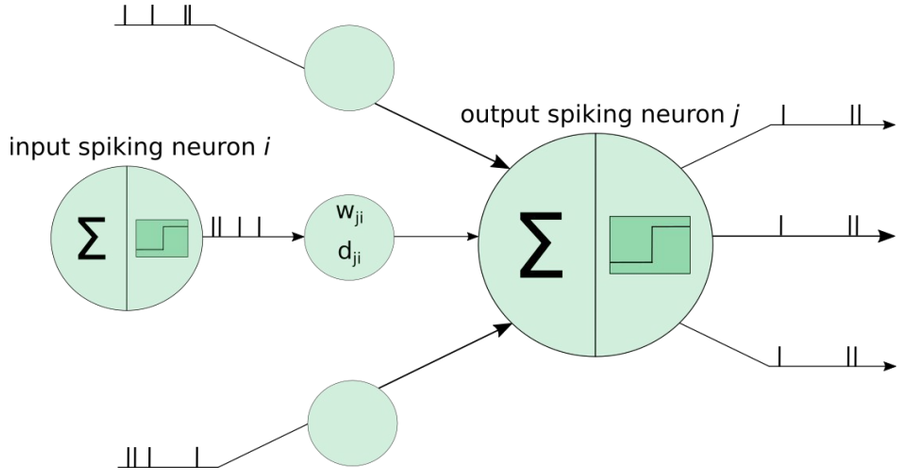
Biologically Plausible

Careful to not bring the biological analogy too far

- Biological neurons much more complex and have large variety



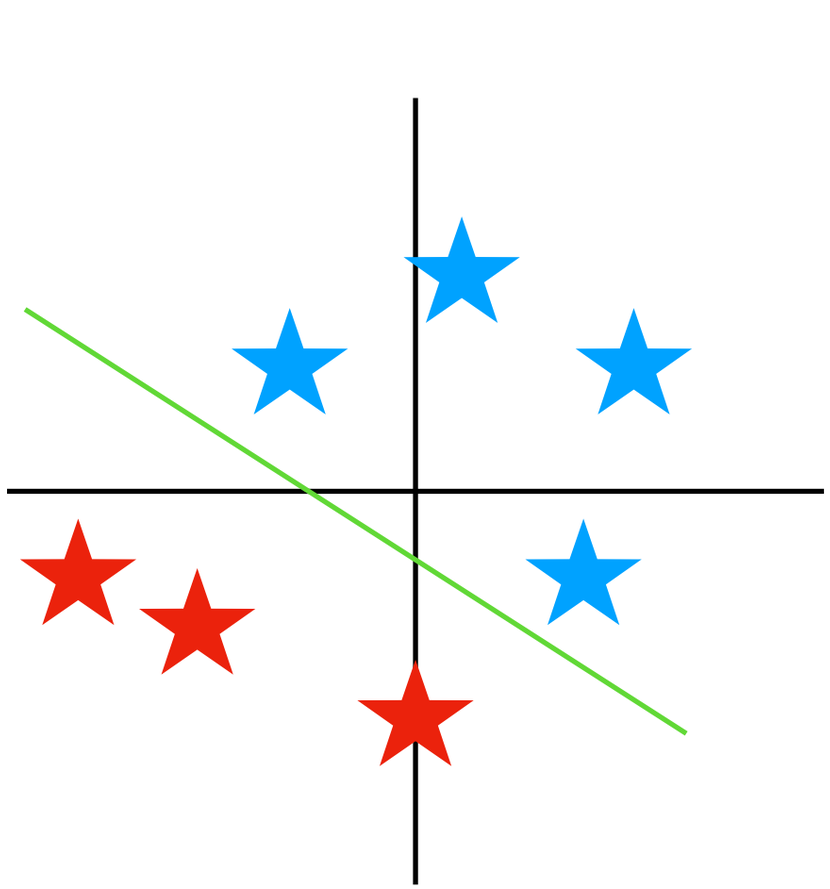
- Timing is (potentially) important in real neuronal activity (spike timing)

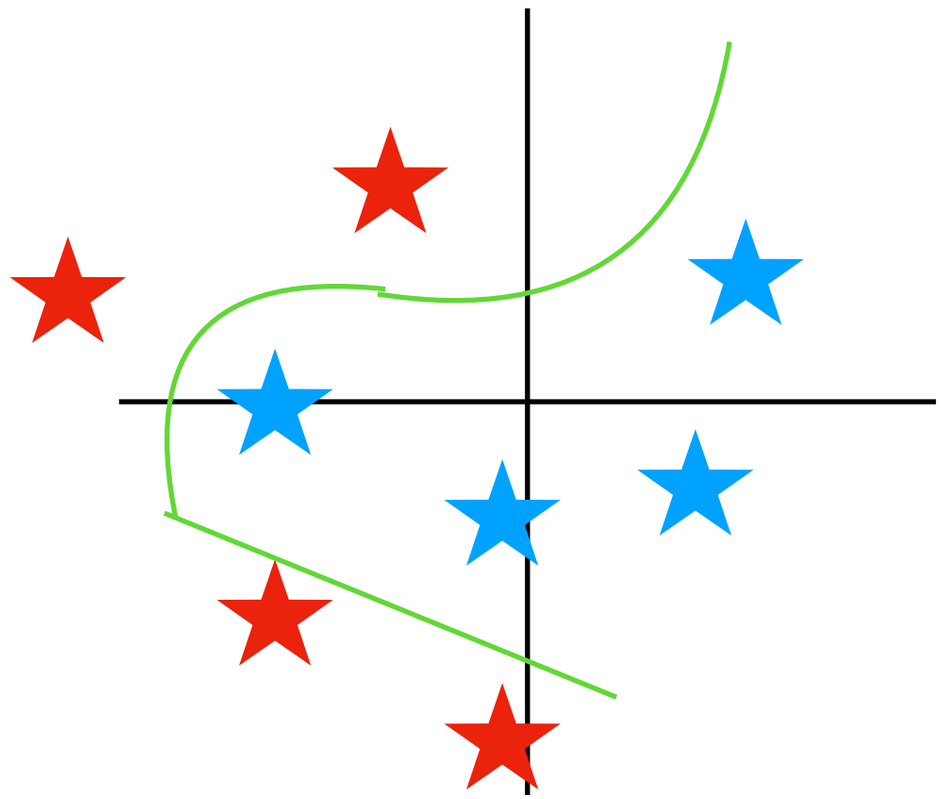


Lobo, J. Et al (2020). Spiking Neural Networks and online learning: An overview and perspectives. Neural Networks.

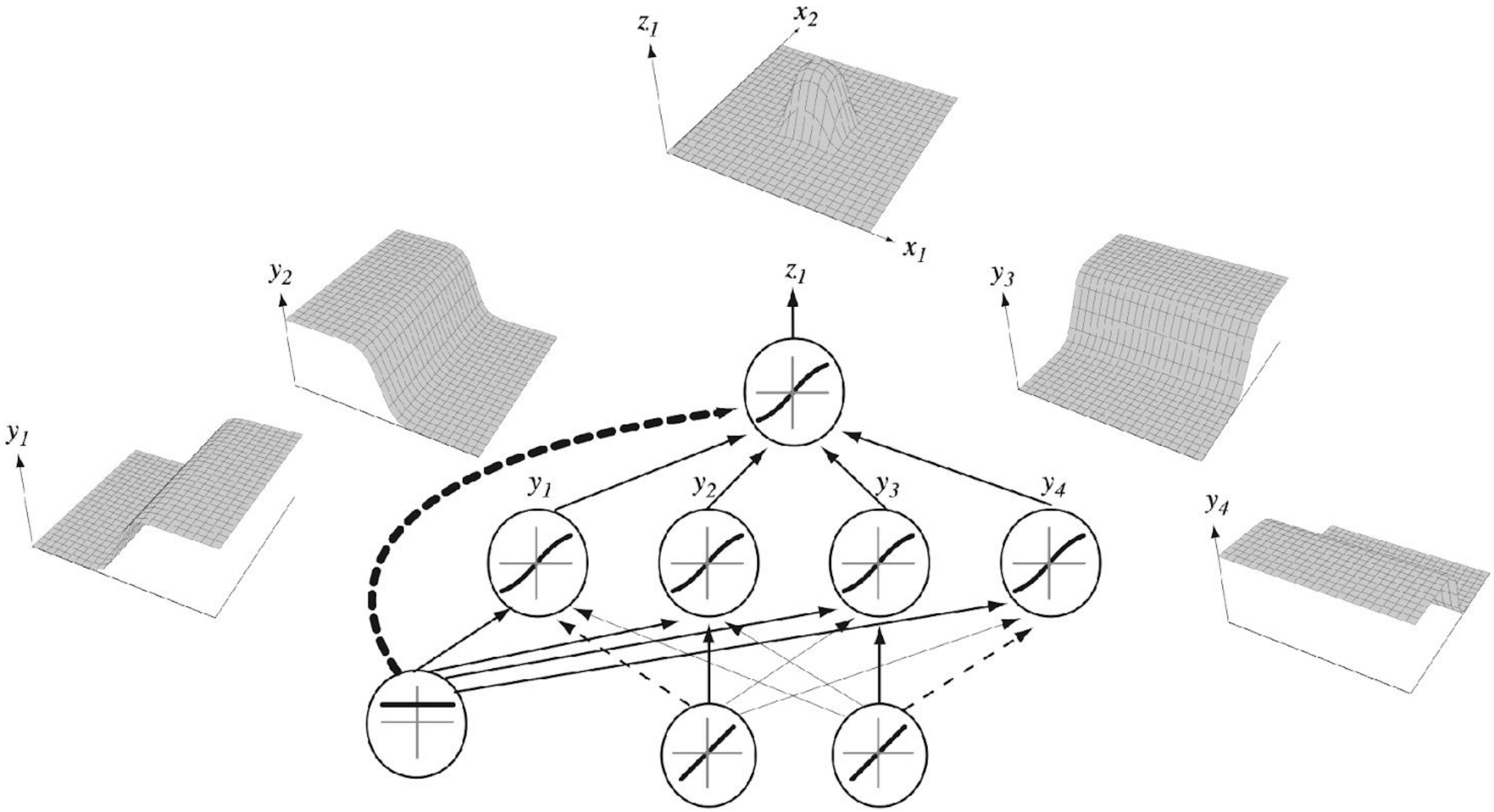
- Learning algorithm in brains is unknown! May not be anything like existing methods for NN
- Brains have feedback and recurrence
- Many other huge differences

Representational Power of Linear Models



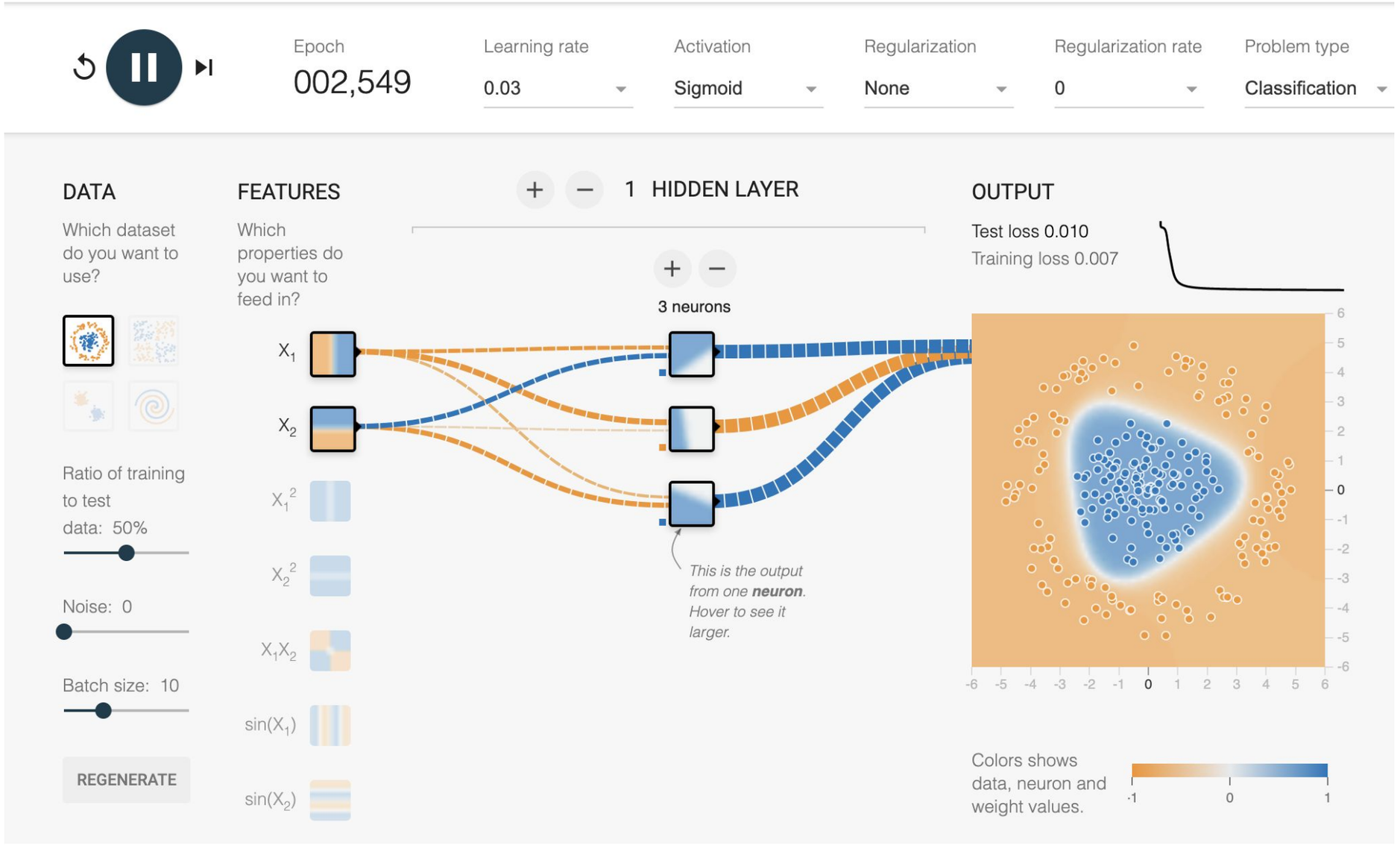


Non-linear 1-hidden layer network



From Pascal Vincent and Hugo Larochelle

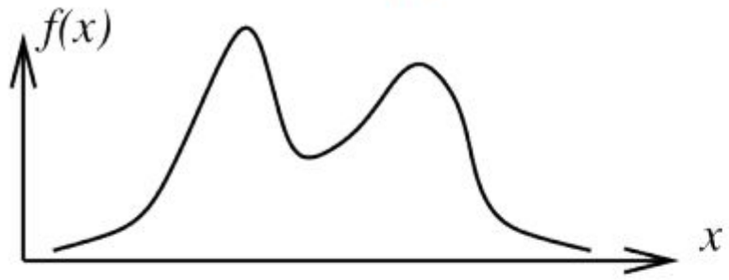
Playground Tensorflow



https://playground.tensorflow.org/

Universal Approximation Theorem

$f_{target} : \mathscr{R}^d \rightarrow \mathscr{R}$ **Function we want to approximate**



$f_W(x) = W_2\rho \circ W_1 x$ **Approximation parametrized by** W_1, W_2

Under conditions on ρ **(satisfied for e.g. for sigmoid), there exists** W_1, W_2

which obtains at most ϵ **error**

$$\sup_{x \in [0,1]^d} |f_W(x) - f_{target}(x)| < \epsilon$$

Cybenko 1989

Why Deep?

Most functions of interest can be approximated with a single hidden layer network:

Why do we need multiple layers?

- Deeper networks can *much more efficiently* represent many functions (number of parameters)

1-hidden layer runs on

N-hidden layer

- Deeper networks, for some data, may bias the learning process towards more general solutions

Images from Wikimedia Commons

Capacity of Deep NN

- Several results exist showing that Deep NN are more parameter efficient

- Existing results are for restricted settings (on the target or network) or make use of assumptions

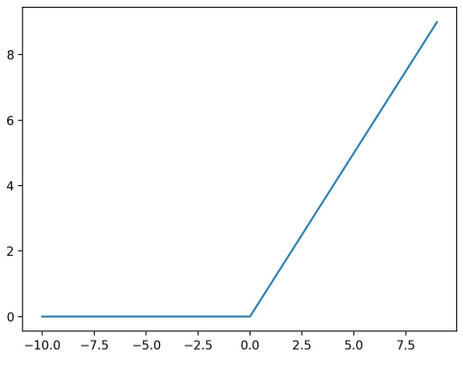
Eldan, Ronen, and Ohad Shamir. "The power of depth for feedforward neural networks." *Conference on learning theory*. 2016.

Montufar, Guido F., et al. "On the number of linear regions of deep neural networks." *Advances in neural information processing systems*. 2014.

Capacity of Deep NN

- (Montufar et al NIPS 2014) studies number of piecewise regions represented by ReLU networks and followed up by many works

Relu non-linearity

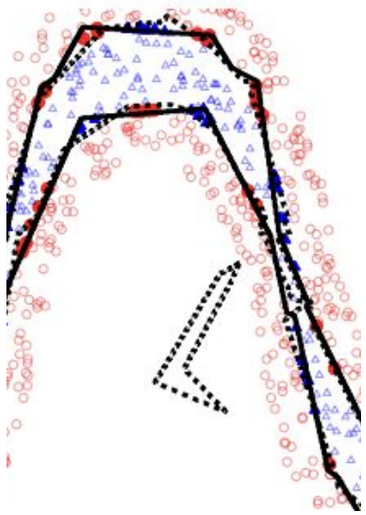


Relu network

$$\begin{aligned} \mathbf{h}^1 &= \max\{0, W^1\mathbf{x} + b^1\} \\ \mathbf{h}^l &= \max\{0, W^l\mathbf{h}^{l-1} + b^l\} \\ \mathbf{y} &= W^{L+1}\mathbf{h}^{\mathbf{L}} \end{aligned}$$

Example in 2d

Another example

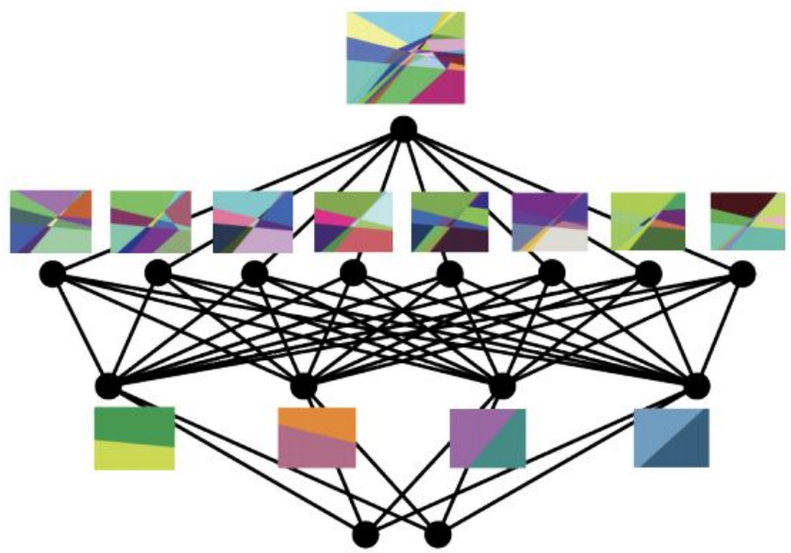


Montufar, Guido F., et al. "On the number of linear regions of deep neural networks." *Advances in neural information processing systems*. 2014.

Serra T. et al Bounding and Counting Linear Regions of Deep Neural Networks. Neurips 2017

Capacity of Deep NN

- (Montufar et al NIPS 2014)



- Number of linear regions in the output increase exponentially with depth and polynomial in width

$$\Omega\left(\left(\frac{n}{d}\right)^{(L-1)d} n^{d}\right)$$

L hidden layers
d input dimension
n width -- number of hidden units per layer

Hanin, Boris, and David Rolnick. "Complexity of linear regions in deep networks." *arXiv preprint arXiv:1901.09021* (2019).

Montufar, Guido F., et al. "On the number of linear regions of deep neural networks." *Advances in neural information processing systems*. 2014.

Depth Helps Generalization (Optional)

- Common conjecture based on empirical observations with some existing, but limited theoretical backing

- Some relevant work:

Urban, Gregor, et al. "Do deep convolutional nets really need to be deep and convolutional?." *arXiv preprint arXiv:1603.05691* (2016).

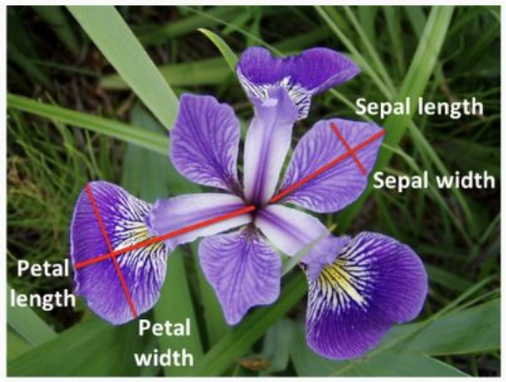
Arora, Sanjeev, et al. "Implicit regularization in deep matrix factorization." *Advances in Neural Information Processing Systems*. 2019.

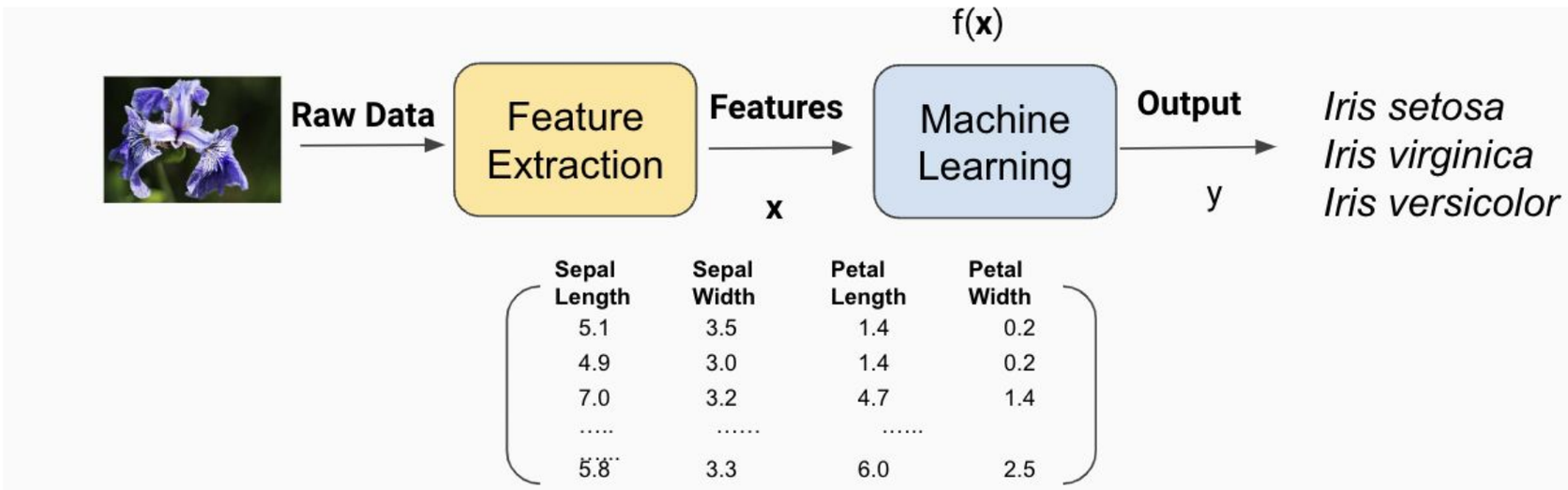
Pérez, Guillermo Valle, Chico Q. Camargo, and Ard A. Louis. "Deep learning generalizes because the parameter-function map is biased towards simple functions." ICLR 2019

Deep Learning

- Rebranding of (deep) Neural Networks - universal function approximator
- Representation learning - avoid hand-crafted features
- Modular but powerful model framework

Classic ML Pipeline





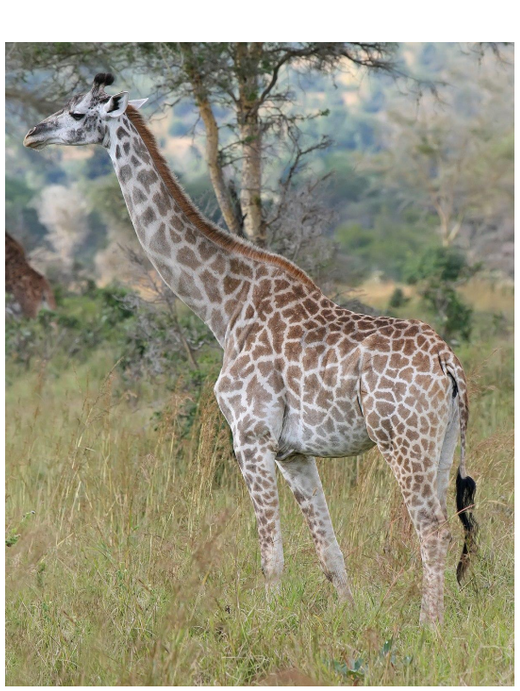
Traditional ML literature assumes useful “features” about the data are extracted before applying ML algorithms

Supervised Representation Learning

- Linear models are fast and easy to use
- Designing hand crafted feature is thus tempting
- Difficult for humans to fully describe models of perception and relevant features they use — representation learning can fill this gap

Representation Learning

- We can interpret intermediate hidden layer outputs as learned features



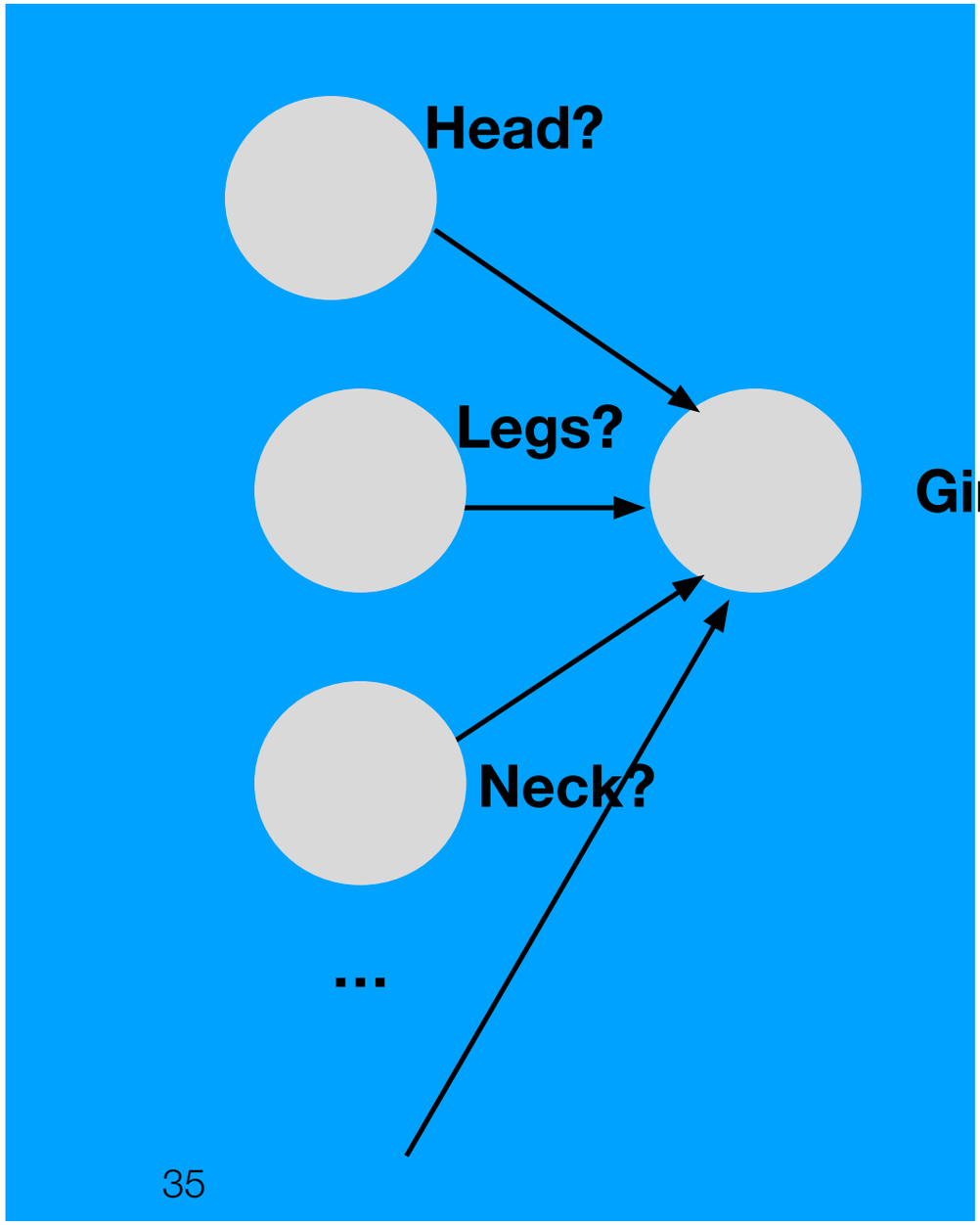
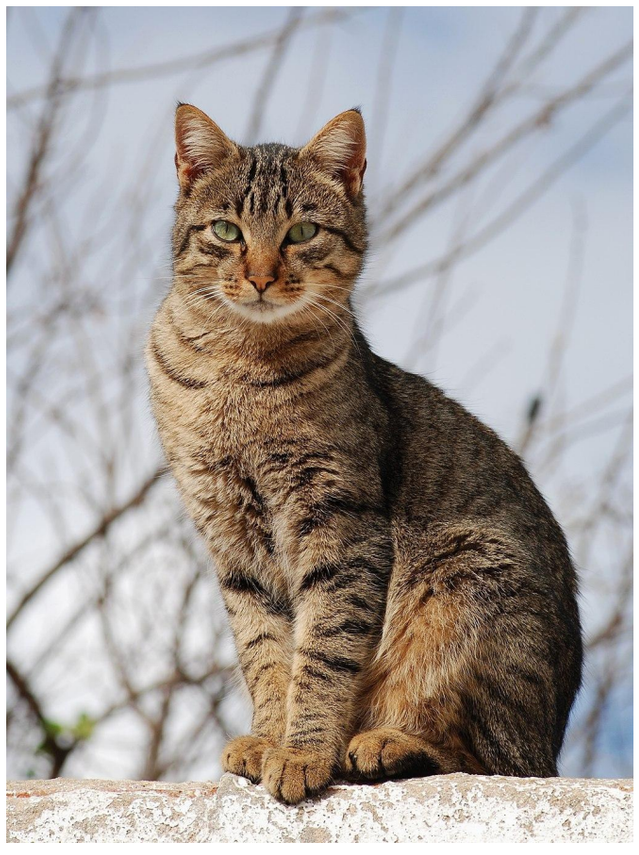


Photo from Wikimedia Commons

Representation Learning

- A key feature of representation learning is reusability, features from one category can be relevant in another suggesting generality



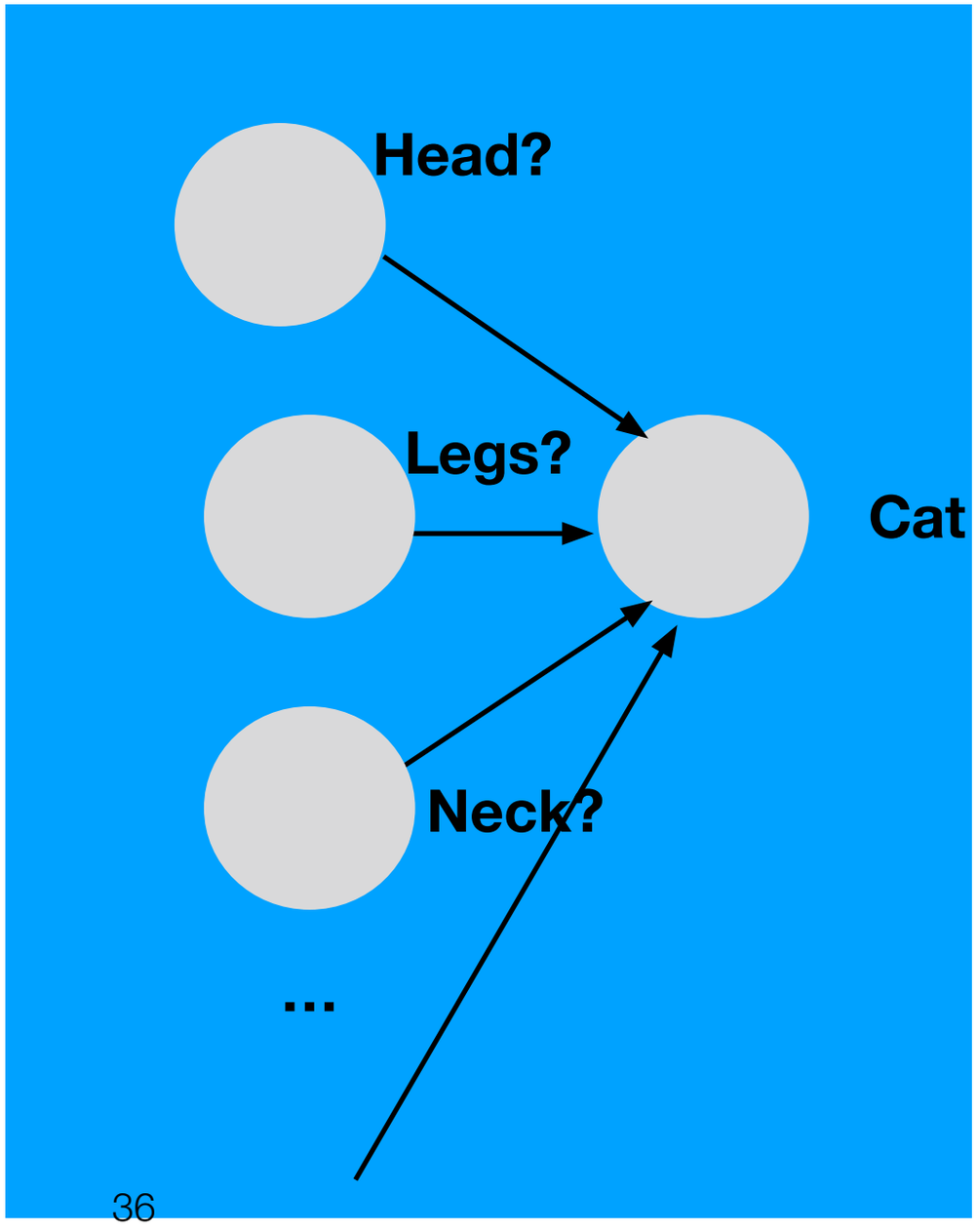
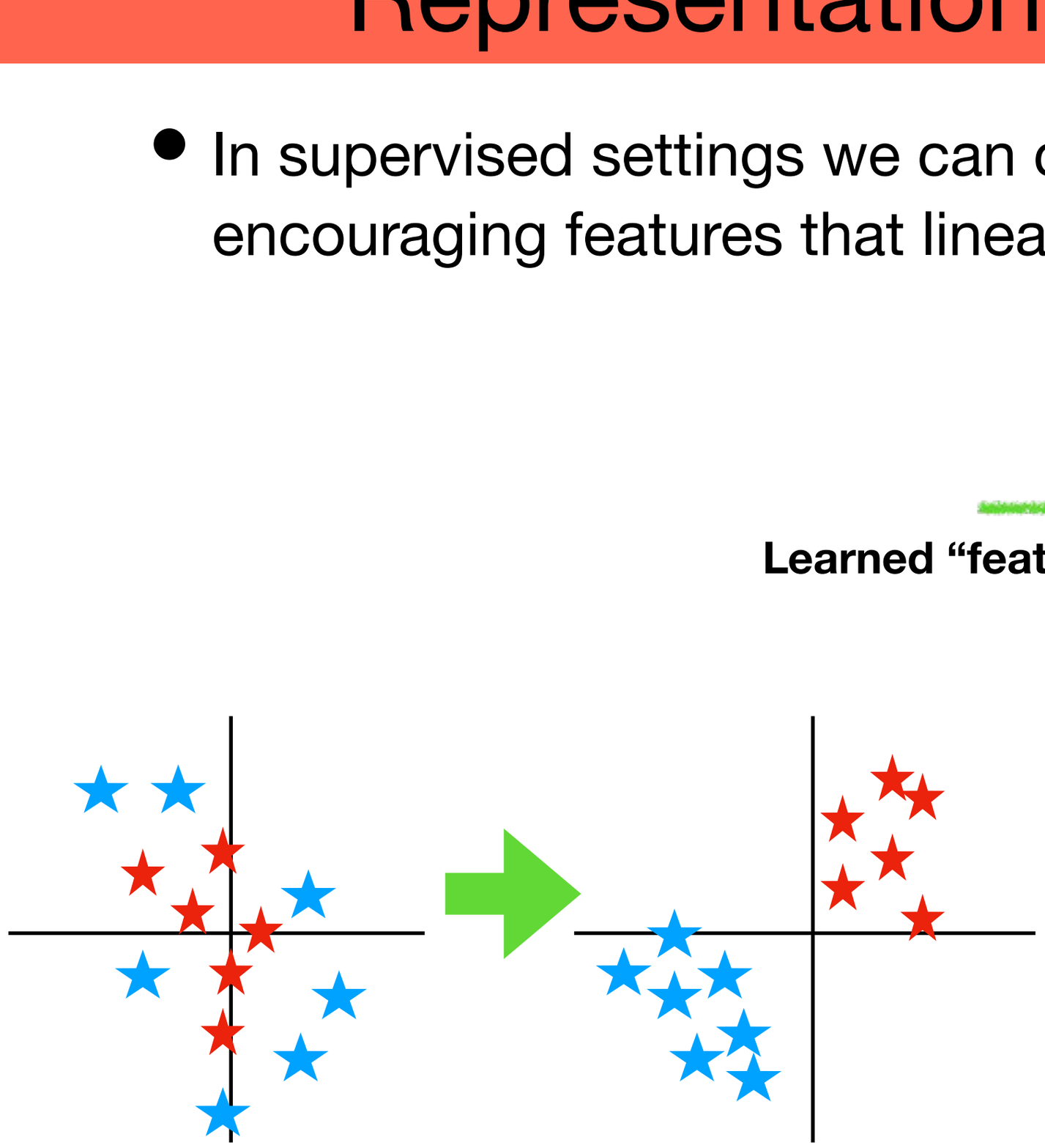
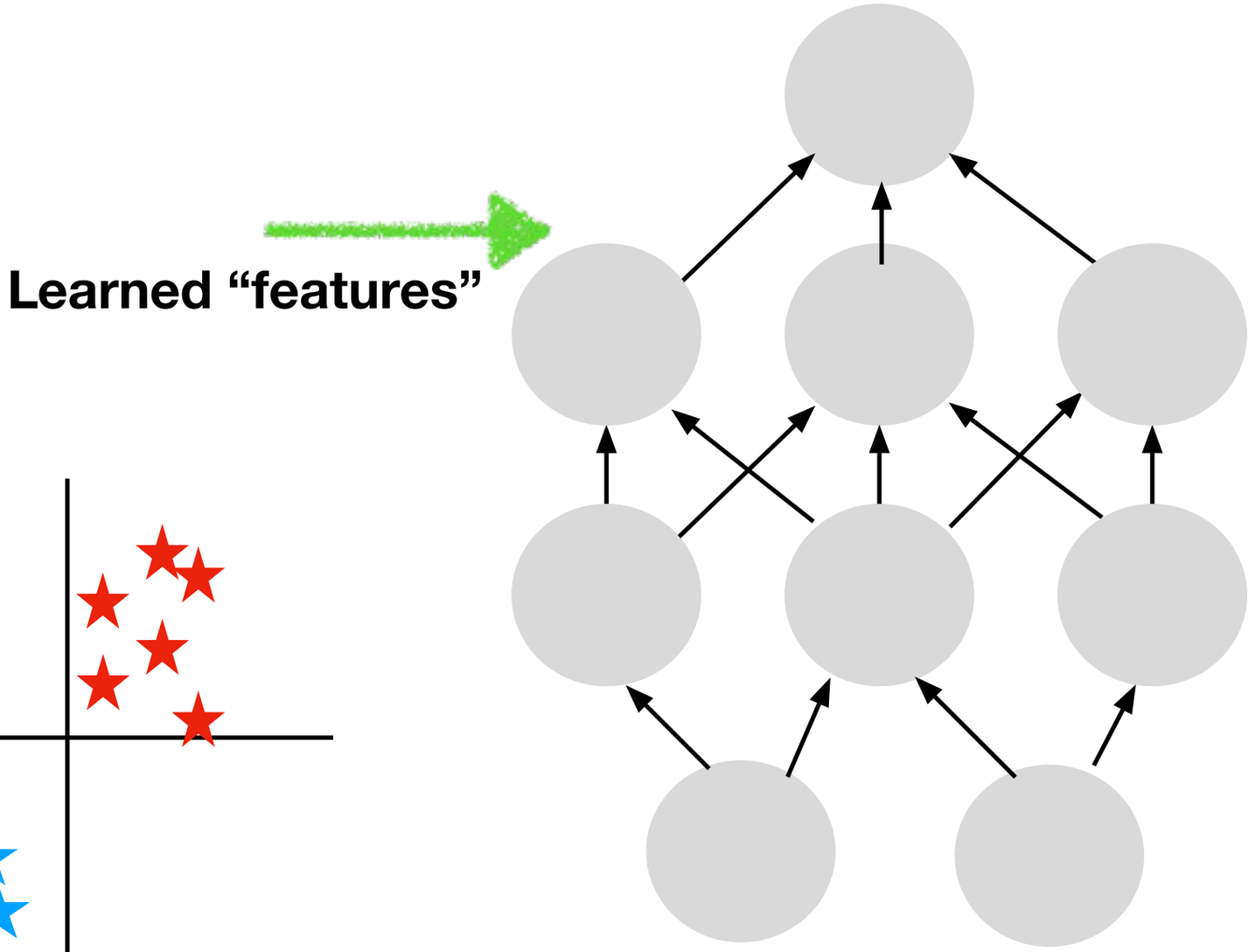


Photo from Wikimedia Commons

Representation Learning

- In supervised settings we can consider final layer as encouraging features that linearly separate the data



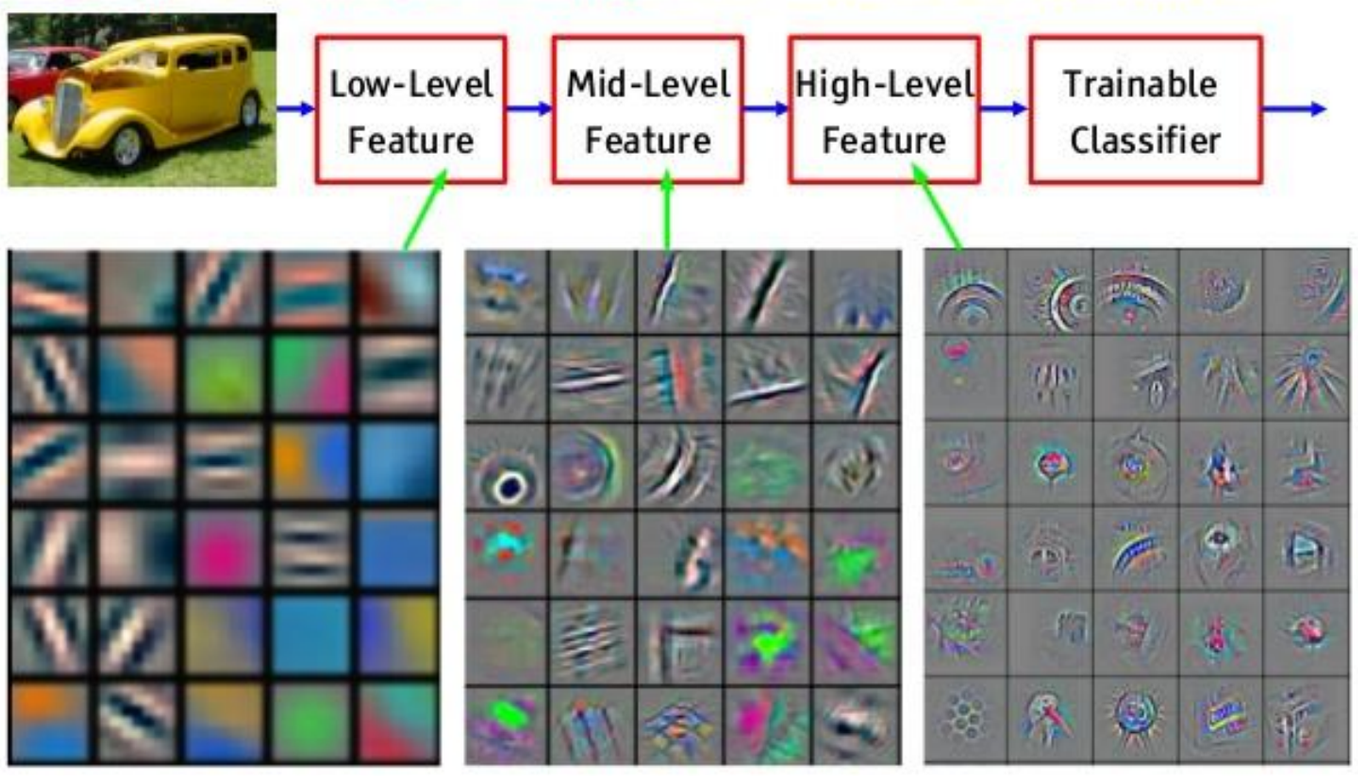


Representation Learning

- In order to optimize the learning objectives with a given deep architecture the learning process can discover relevant features of the data

- Intermediate representations in deep networks can find features similar to those designed by humans

- However they can also learn ones we wouldn't think of



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013] From Y Lecun slides

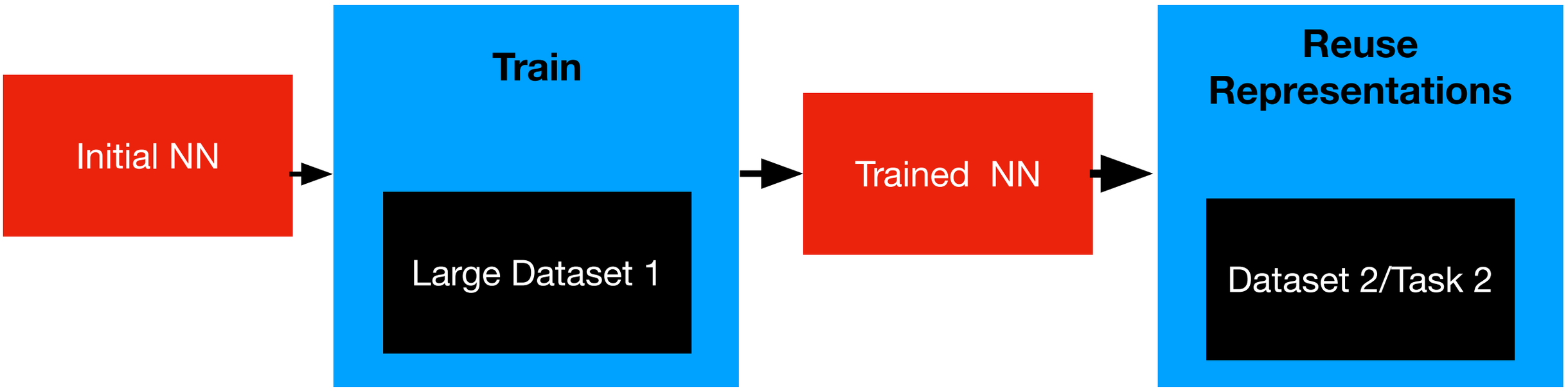
Representation Learning Re-usability

- A goal of representation learning is to be useful on the task at hand but also on new tasks and datasets

- With enough varied data representation learning can often outperform hand crafted feature

    - Imagenet learned features more useful in most computer vision tasks than hand-crafted features

- Can reuse representations in multiple ways



Unsupervised Rep. Learning

- Unsupervised representation learning simple example, auto encoders

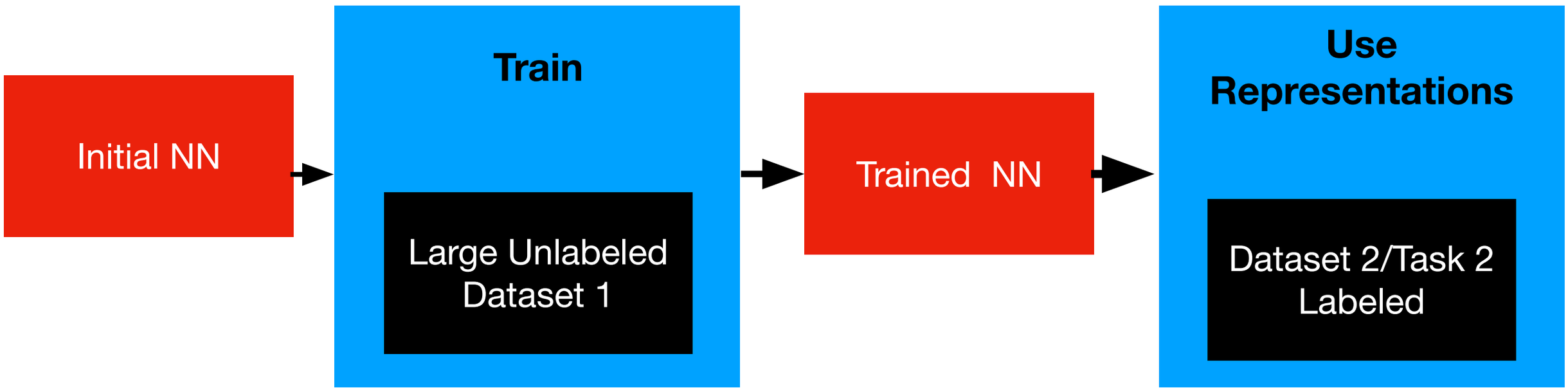
Encoder Network

Decoder Network

Photo from Wikimedia Commons

Unsupervised Rep. Learning

- Unsupervised representation learning has been a major driver of deep learning research since 2006



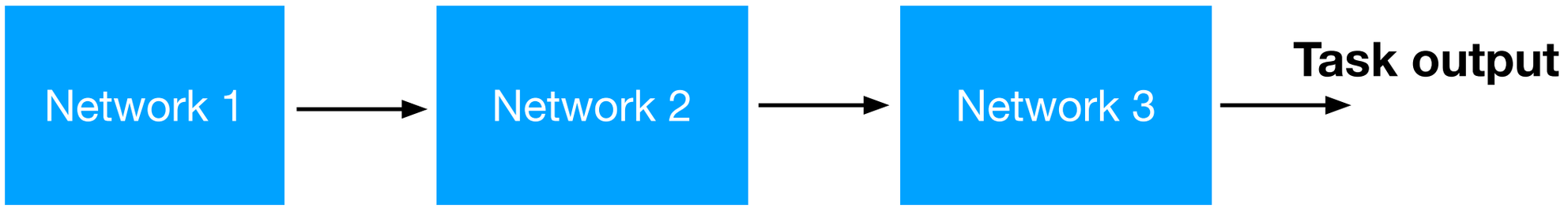
- More and more unsupervised learning methods are becoming useful

Deep Learning

- Rebranding of (deep) Neural Networks - universal function approximator
- Representation learning - avoid hand-crafted features
- Modular framework for high capacity models

A Modular Framework

- Gradient descent and back-propagation algorithm leads to set of tools for jointly adapting modular system for a single objective



A Modular Framework

Toolbox

Component 1

Component 2

Component 3

Component 4

Assemble model components from toolbox

Photo from Wikimedia Commons

Block 1 → Block 2 → Block 3

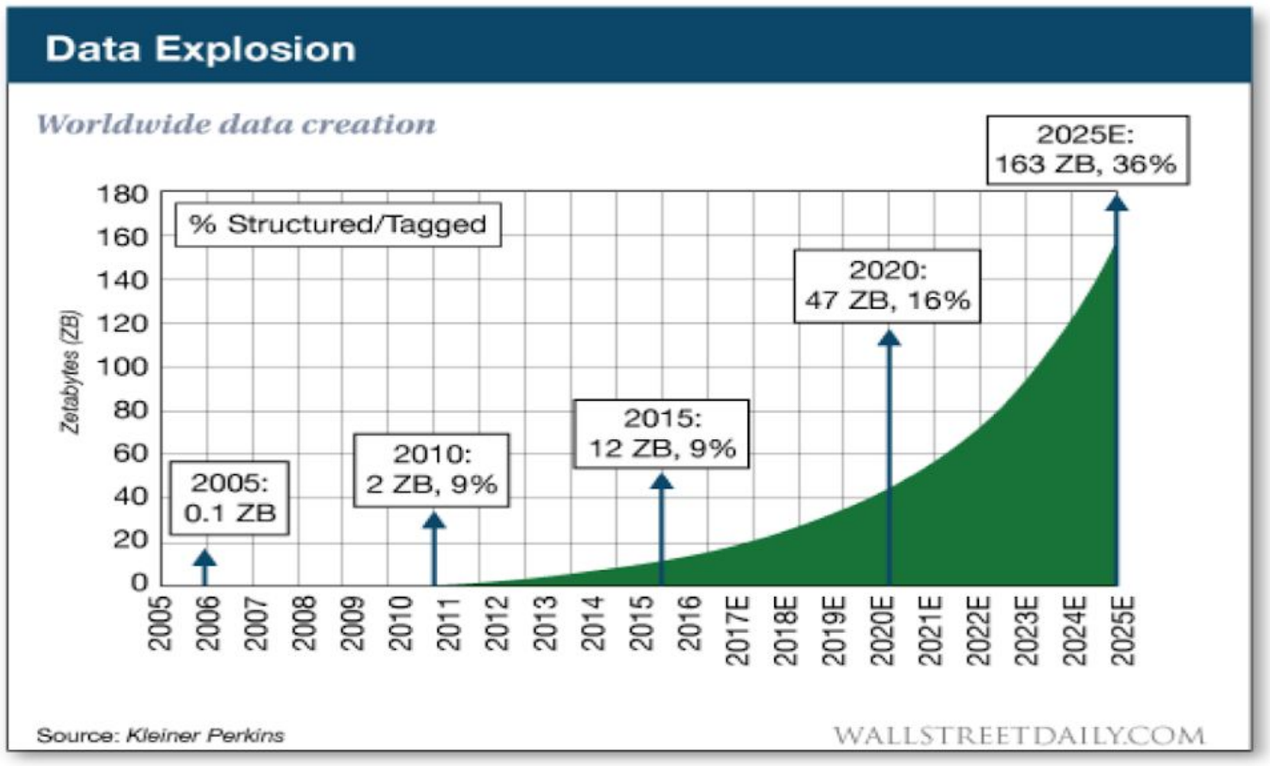
Learn parameters

A Modular Framework

- Software frameworks for adjusting all model components is set
- Adding priors is easy and learning is formulaic
- Flexible framework to build models with strong approximation ability

Large Data and Compute

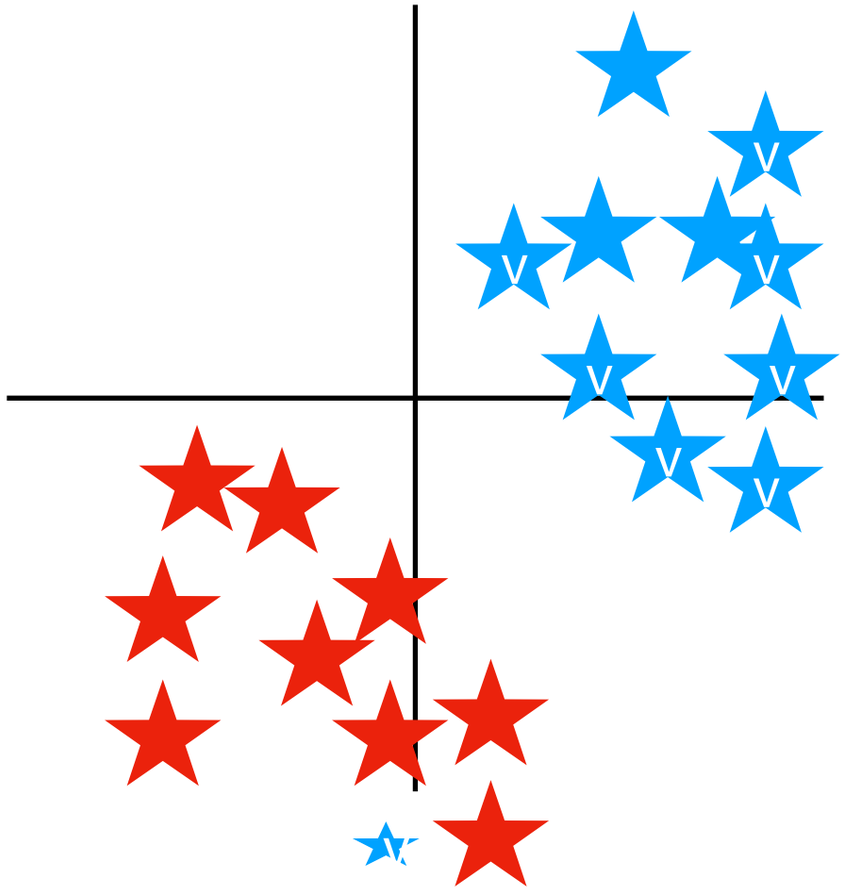
- Deep learning models and frameworks are ripe to take advantage of larger datasets and increasing compute



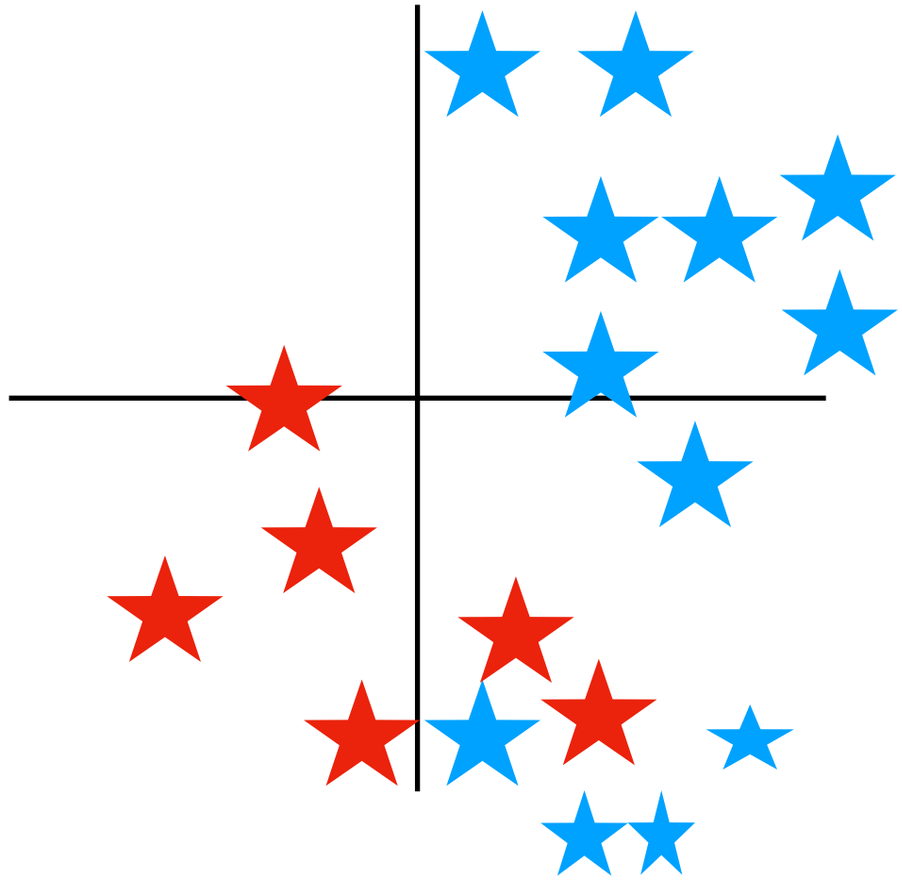
Images from Nvidia Inc

More data + high capacity model

10,000 samples

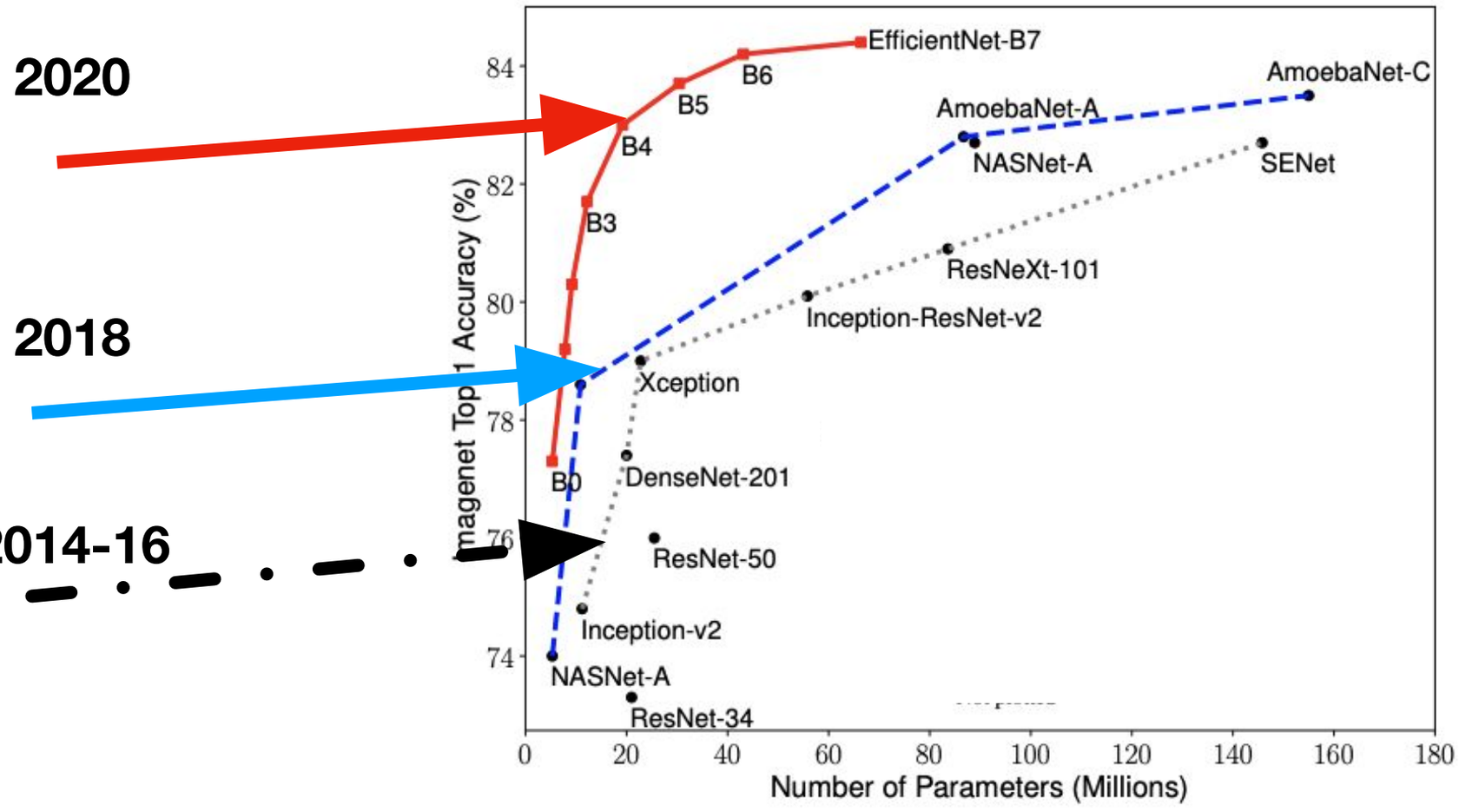


1 Million samples



Bigger Models Keep Being Better

- Original Imagenet dataset of ~1.2 million images released in 2010



- For architecture class bigger model increases performance — power law
- More efficient architectures shift the curve over time

Figure from:

EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks

Alexnet

2012

When do we not use Deep Learning

- Many machine learning problems do not require deep learning or huge models
- Training speed — deep learning is slow
  - Logistic regression, Random Forests, Gradient boosting, and related can often be fit on a laptop for mid-sized problems, including hyperparameter search
- Linearly separable problems

When do we not use Deep Learning

- Smaller datasets, without external related data, sometimes more easily solved with hand-crafted features + simpler classifiers

- Good feature extractors exist

    - Deep Learning methods particularly excel on perceptual data (images, speech, language)

- Interpretability is critical

- Tabular data can often be solved equally well by other methods

- This is a non-exhaustive list, there are many other situations as well!