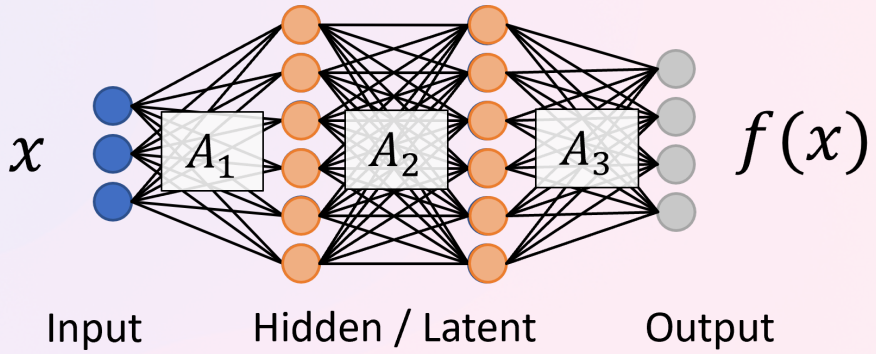
# Basics of Deep Learning

David I. Inouye

# What Is Deep Learning? Sequential Transformations Learned From Data

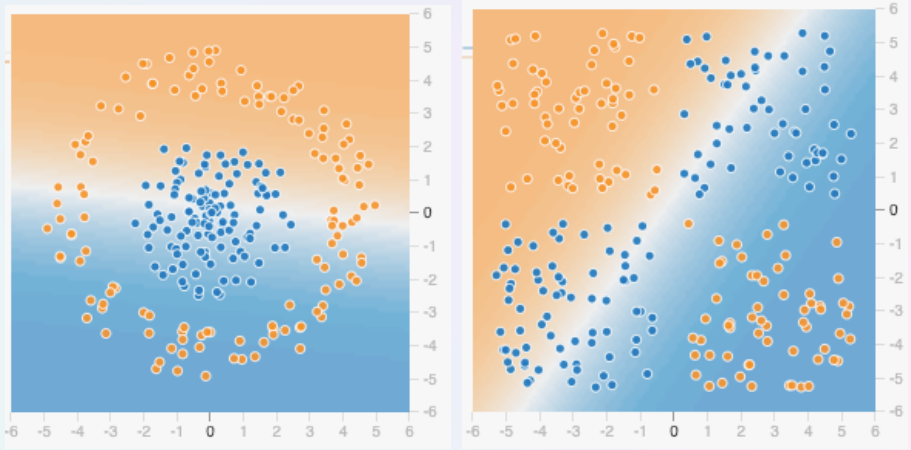
- Classical deep neural networks: $f(x) = \sigma(A_3\sigma(A_2\sigma(A_1x)))$



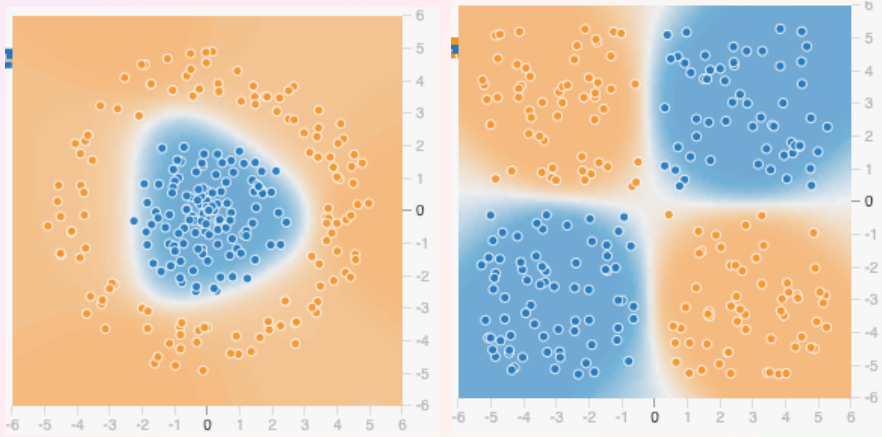

- More generally, **deep models** are sequential transformations: $f(x) = f_3(f_2(f_1(x)))$
  - $z^{(1)} = f_1(x)$ (Layer 1)
  - $z^{(2)} = f_2(z^{(1)})$ (Layer 2)
  - $z^{(3)} = f_3(z^{(2)})$ (Layer 3)
- **Deep learning** estimates these transformations from data

# Motivation 1: Linear Models Cannot Model Complex Classification Boundaries

## Linear models cannot capture complex patterns



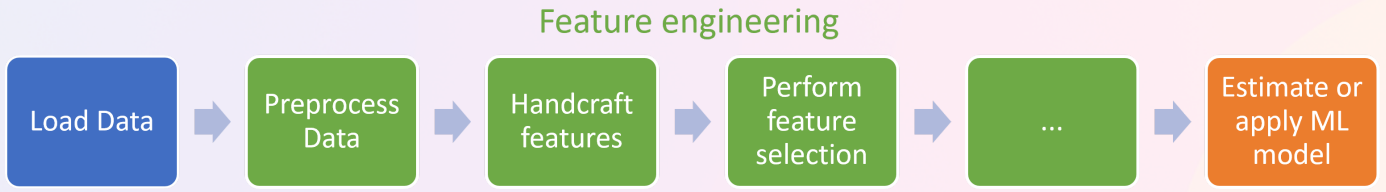
## Non-linear models can capture complex patterns



https://playground.tensorflow.org/

# Motivation 2: Hand Crafting Features Can Increase Performance but Is Expensive

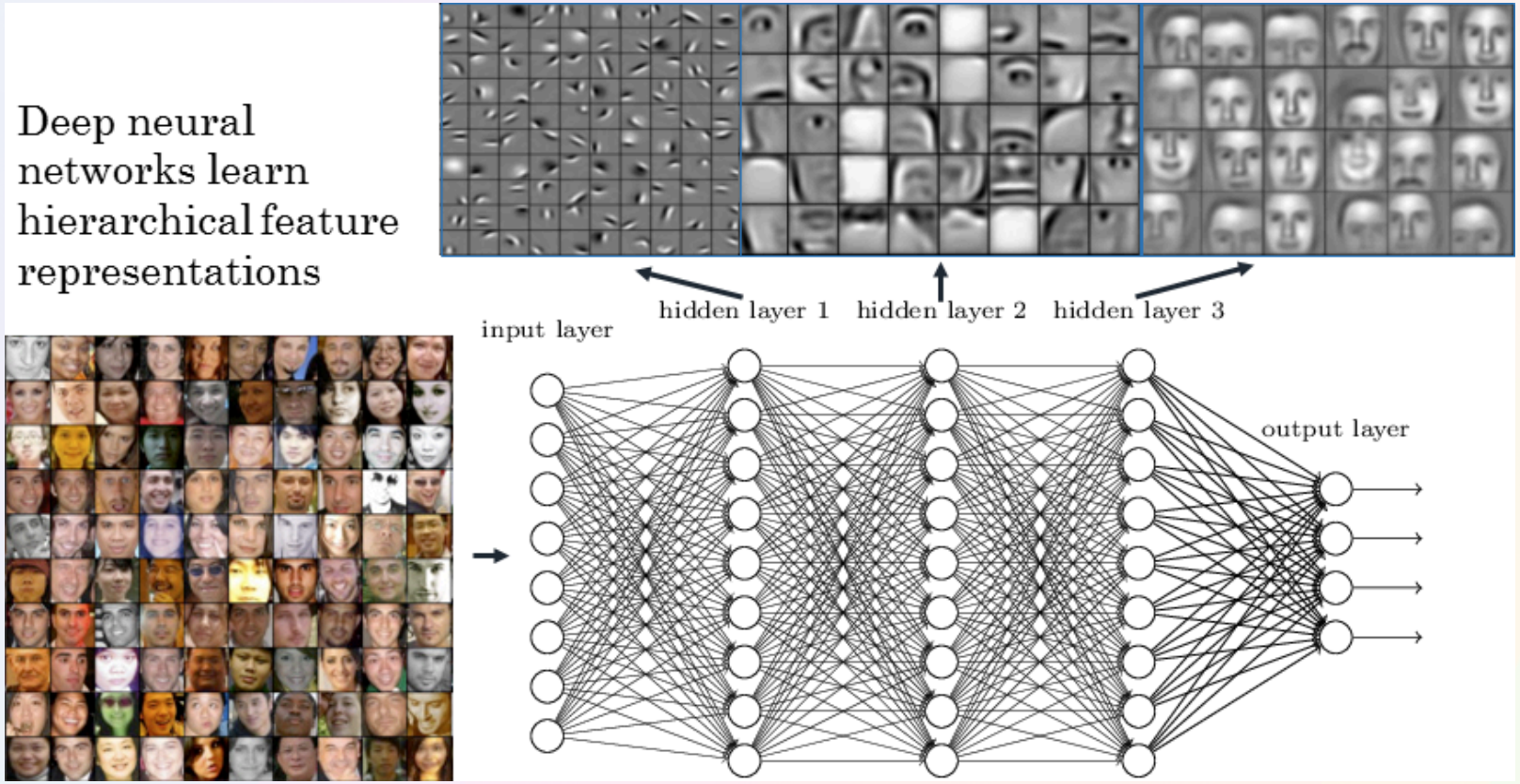
## Classical Machine Learning




## Deep Learning


- Caveat: But now you have to select the model architecture (a little like feature engineering).

# Motivation 3: Deep Learning Can Automatically Learn a Hierarchy of Representations



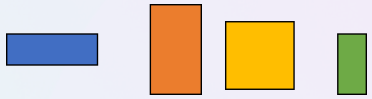

https://towardsdatascience.com/a-road-map-for-deep-learning-b9aee0b2919f

# The Key Design Choices of Deep Learning Are Architecture, Algorithm, and Objective Function

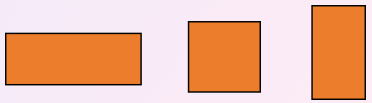
1. Deep model **architecture**
2. Deep learning optimization **algorithm**
3. Deep learning **objective function** design
    - (Application specific so we will discuss later with Transformers, VAEs, Diffusion etc.)

# The **Model Architecture** Defines the Structure of the Model (Though Not Parameter Values)

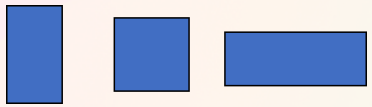
- **Which layers or modules?**
  - Fully connected
  - Convolutional
  - Residual blocks
  - ...



- **How big?**
  - What is the dimensions of the input and output?

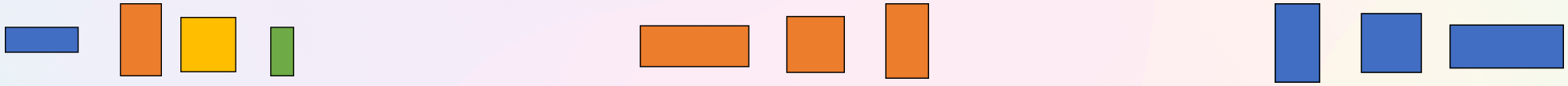


- **How many and in what order?**



# The Architecture Defines the **Inductive Bias** of the Model

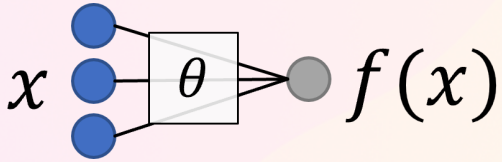
- **Inductive bias** is the bias of the model to perform better on certain problems
- A modern view of the “No Free Lunch Theorem”
- Example: Convolutional networks perform very well on image data
- Example: Attention-based “Transformer” networks have proven particularly successful for sequence data



# **Fully Connected Layers** Are Linear Functions Followed by Elementwise Non-Linear Activation Functions

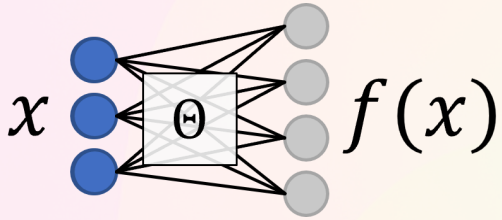
- Remember logistic regression:

$$f(x) = \sigma(\theta^T x)$$



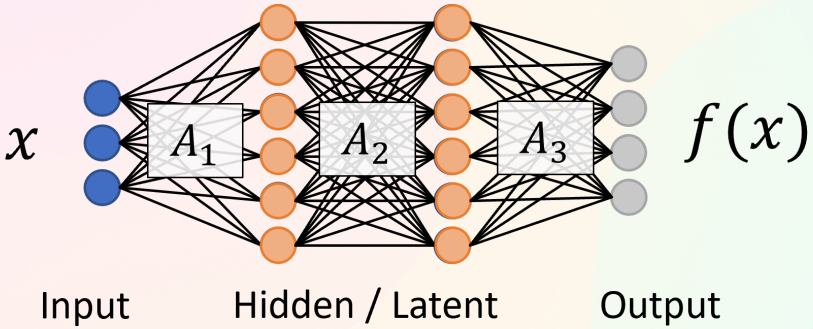

- A fully connected layer can be seen as multiple logistic regressions:

$$f_{FC}(x) = [\sigma(\theta_1^T x), \cdots, \sigma(\theta_k^T x)]$$



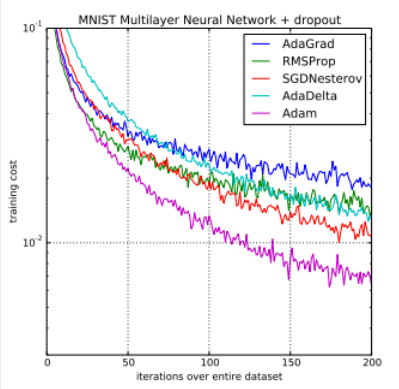

- A deep fully connected network is multiple fully connected layers:

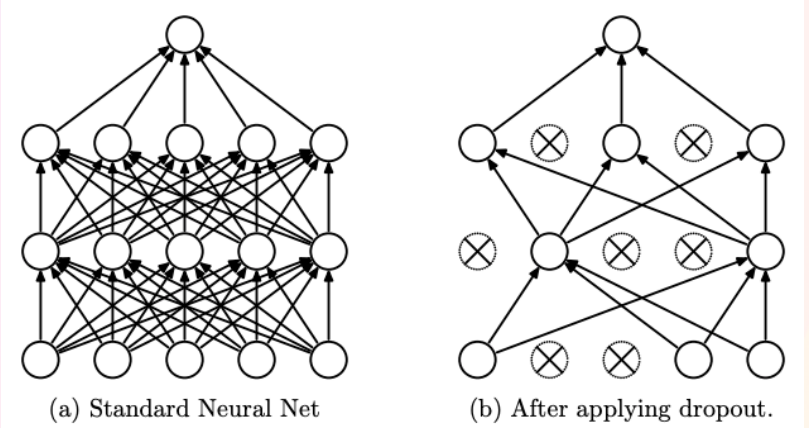
$$f(x) = \sigma(A_3\sigma(A_2\sigma(A_1 x)))$$



# The **Optimization Algorithm** Defines How the Parameters Will Be Updated

- **Optimizer**
  - SGD, ADAM, etc.
  - Step size
- **Special "optimization" layers**
  - BatchNorm
  - Dropout
- **Order of optimization updates**
  - Example: Multiple inner optimization problems (e.g., adversarial optimization, GAN)



(a) Standard Neural Net (b) After applying dropout.

# **Automatic Differentiation** Enables Decoupling Between Architecture Design and Algorithm

- All computation can be broken into simple components
  - Examples: sum, multiply, exponential, convolution
- Derivatives can be derived mathematically
- Derivatives for **any composition** can be derived via chain rule! 🙂
- (Prof. Jeffrey Siskind was a pioneer in automatic differentiation, see https://www.jmlr.org/papers/volume18/17-468/17-468.pdf)

# **Reverse-Mode** Automatic Differentiation Can Be Computed in Almost the Same Time as the Original Computation Itself!

- **Forward pass:** Original objective computation

$$\mathcal{L}(X, y; \theta) = \frac{1}{n} \sum_i \ell(y_i, f_k(\cdots f_2(f_1(x_i))))$$

- **Backward pass:** Compute gradient by stepping backwards through computation

$$\nabla_\theta \mathcal{L}(X, y; \theta)$$

  - Also called "backpropagation" algorithm since it backpropagates the derivative
- Amazingly, the cost of the forward and backward passes are equal up to a constant
- How many forward passes to approximate derivative via small finite differences?
- $O(M)$ where $M$ is the number of parameters!

# PyTorch and TensorFlow Implement Automatic Differentiation Directly

- Demo doing automatic differentiation