CAI 4104/6108 – Machine Learning Engineering:

Convolutional Neural Networks

Prof. Vincent Bindschaedler

Spring 2024

Administrivia: Project

- Project Proposals are due **3/27** on Canvas (by 11:59pm)
  - Create a group (4 or 5 students)
    - Canvas: Go to People → Project Groups
  - Identify a **task**, **dataset**, and some **baselines** (& metrics)
  - Fill in the project proposal template and submit the PDF on Canvas

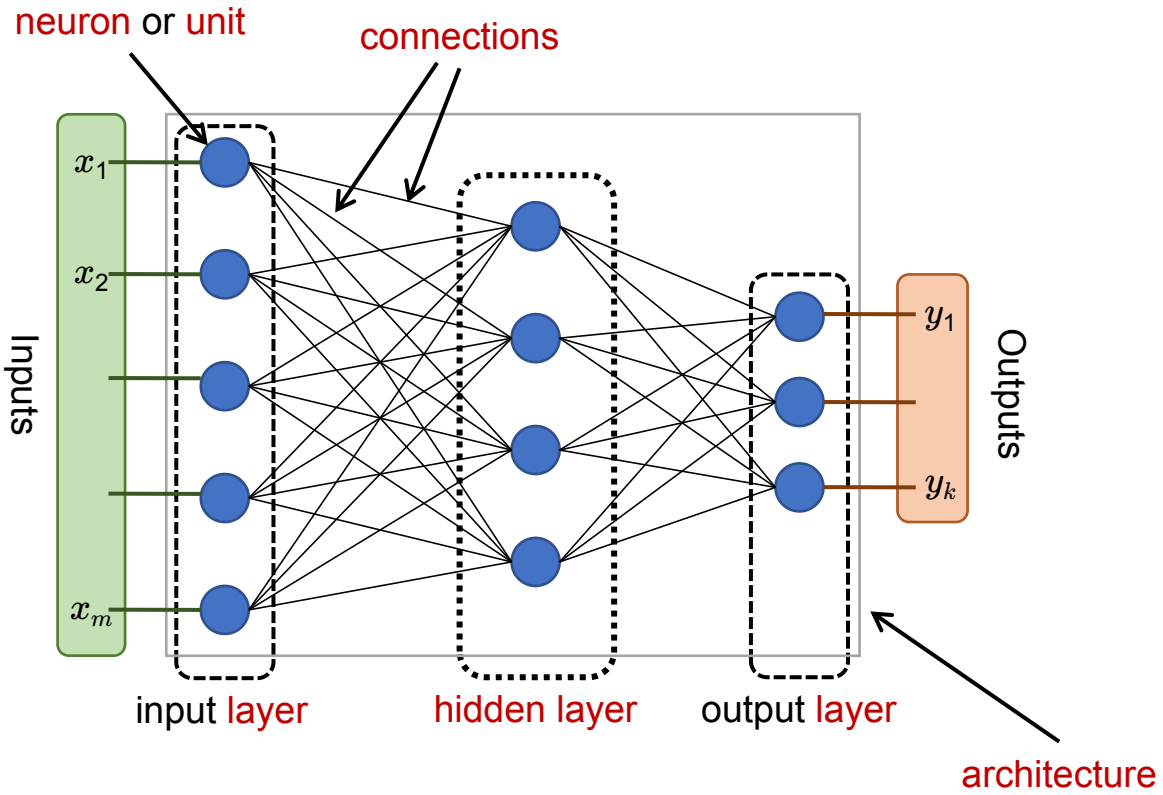
- Task & Dataset: your choice
  - There **cannot** be overlap with other groups
  - Baselines and metrics have to make sense
  - **No** trivial projects please!
    - No toys datasets, MNIST classification has been done to death, etc.

Administrivia: Project

- Proposal
  - Length: 1 - 2 pages (use template)
  - Structure:
    - Follow instructions in template
  - **Writing counts! (try to be clear and concise)**

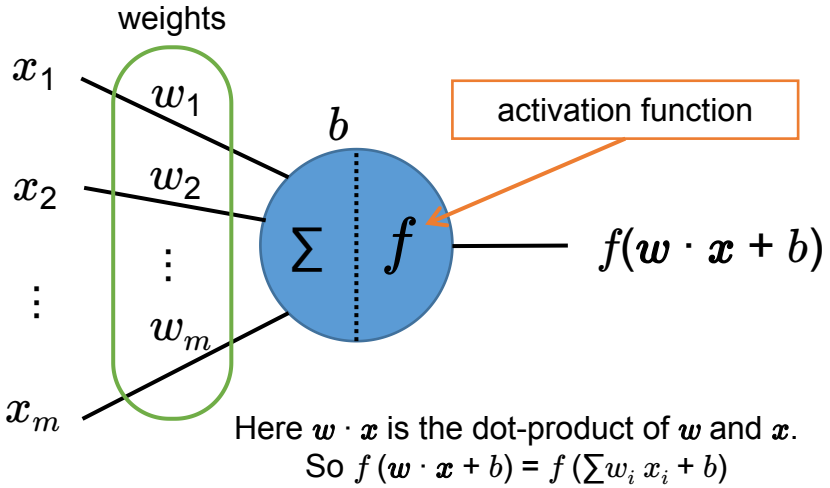
- Submission on Canvas
  - **Use provided Latex template!**
  - One per group; write all group members names
  - Only one person (point of contact) submits on Canvas
    - **Use the group functionality of Canvas**
    - You and **all** of your team mates must be in the same project group on Canvas

Reminder: Neural Network Terminology



Reminder: A Simple Neural Network

- Consider a single neuron / unit
  - The model is $h_{\boldsymbol{w},b}(\boldsymbol{x}) = f(\boldsymbol{w} \cdot \boldsymbol{x} + b)$
    - What if we take f to be the identity function?
      - That is: $f(z) = z$
    - What if we take f to be the sigmoid / logistic function?
      - That is: $f(z) = 1/(1+e^{-z})$

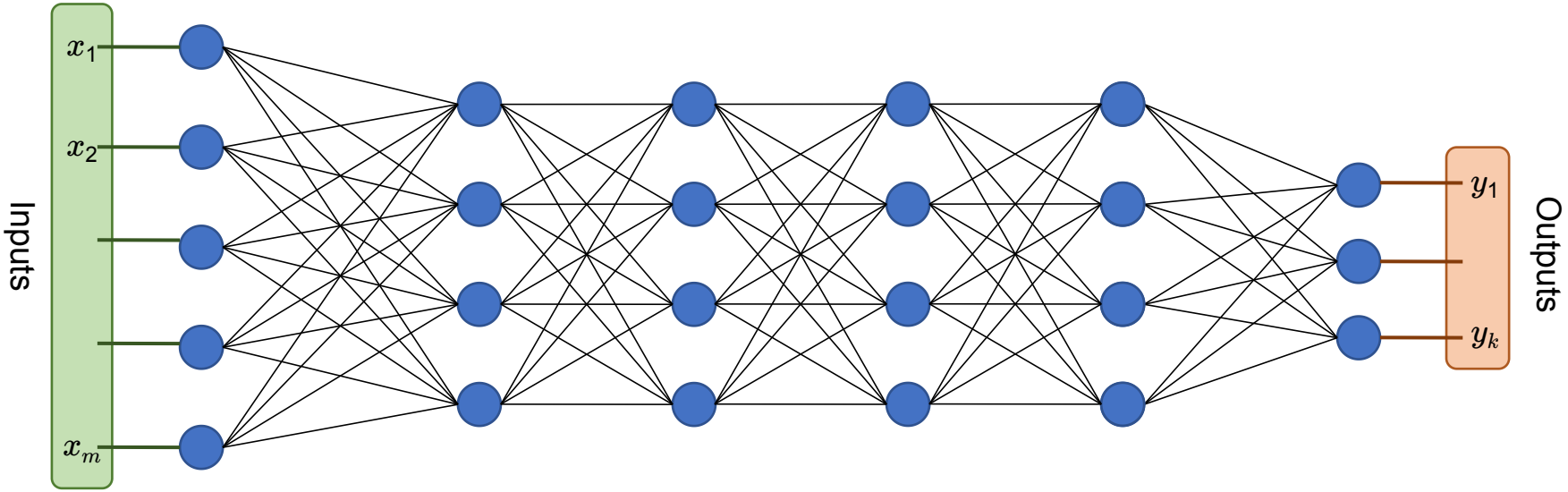


Here $\boldsymbol{w} \cdot \boldsymbol{x}$ is the dot-product of $\boldsymbol{w}$ and $\boldsymbol{x}$.
So $f(\boldsymbol{w} \cdot \boldsymbol{x} + b) = f(\sum w_i x_i + b)$

- The Perceptron
  - Invented by Frank Rosenblatt in 1957
    - "The Perceptron—a perceiving and recognizing automaton". Report 85-460-1. Cornell Aeronautical Laboratory
  - A different neuronal architecture called a threshold linear unit (TLU)
    - No bias term
    - With a step activation function. For example:
      - heaviside(z) = 0 if z ≤ 0, 1 otherwise (z ≥ 1) ; or sign(z)

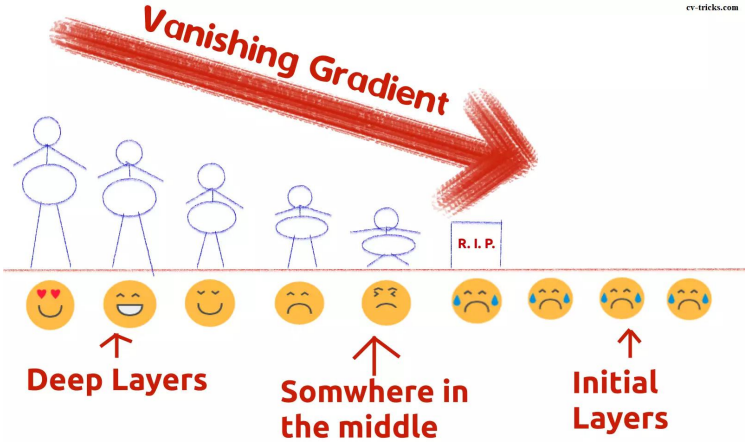
Reminder: Deep Neural Networks

- What is a deep neural network?
  - Any neural network with two or more hidden layers
  - Nowadays, the best neural networks architectures for many applications & problems are deep
    - E.g.: AlexNet (2012) has 8 layers, ResNet18 has 18 layers, GPT-2 has 48 layers

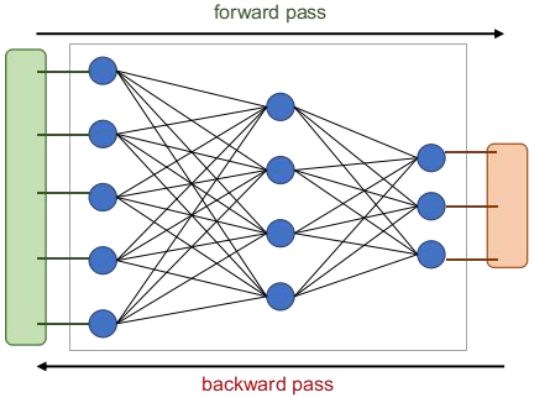


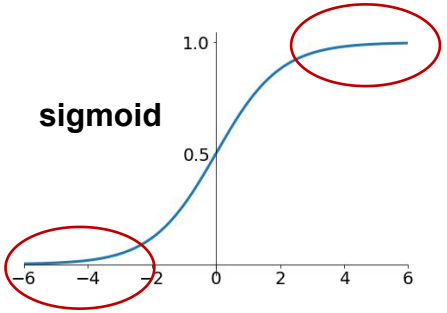
Reminder: Unstable Gradients

- Vanishing/Exploding Problems:
  - Gradient vector becomes very small (vanishing gradient) or very large (exploding gradient) during backpropogation
    - Difficult to update weights of lower/earlier layers => Training does not converge
  - Instance of a more general problem: unstable gradients
    - Layers (of a deep neural network) learn at very different rates



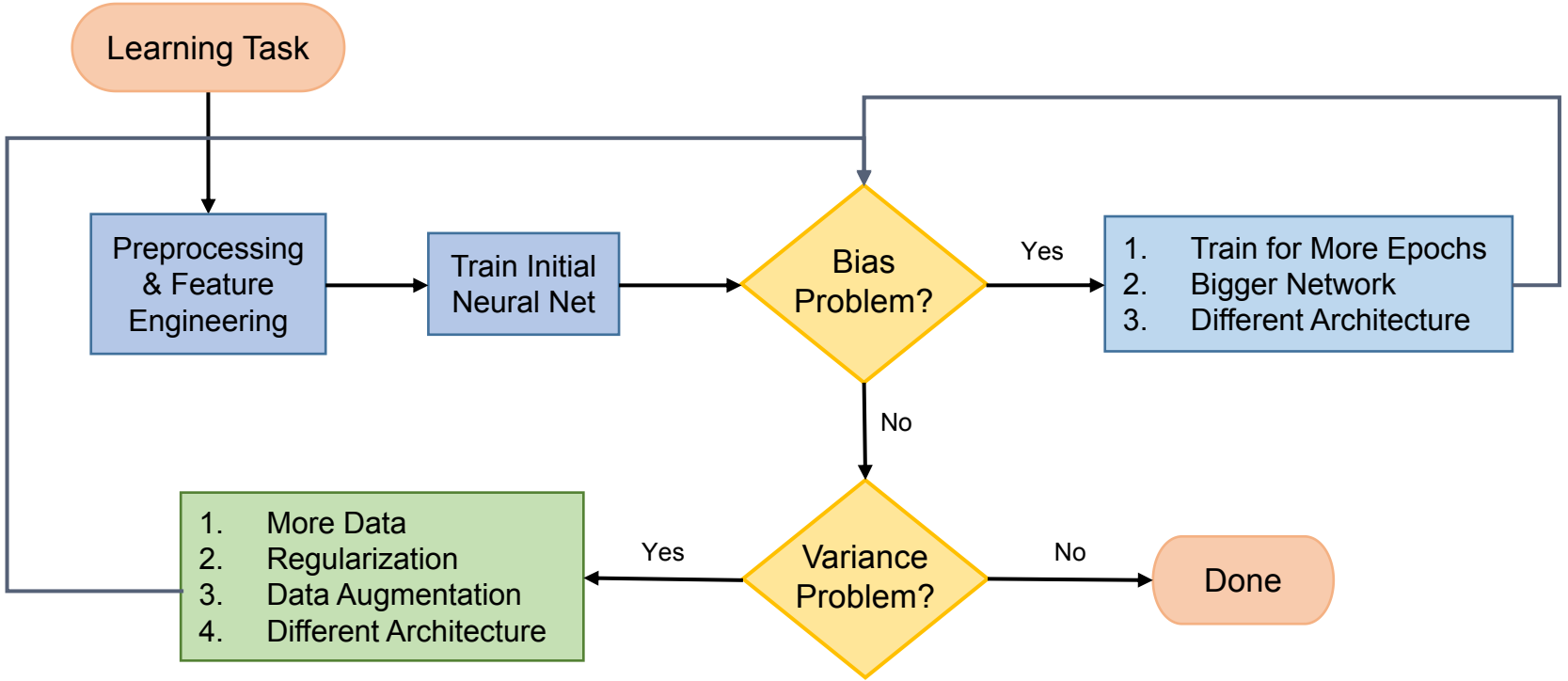
Source: https://cv-tricks.com/keras/understand-implement-resnets/





Saturation

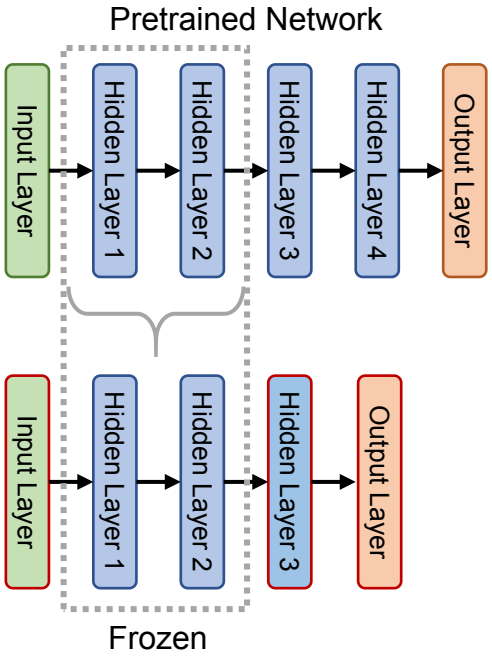
Reminder: Problem Solving



Reminder: Transfer Learning

- Should you train a deep neural network from scratch?
  - Not always. When possible you should use transfer learning:
    - Pick a pre-trained deep neural network in the same or related domain
    - Then fine-tune on the task you care about

- Reusing an existing deep neural network
  1. Pick some layers to reuse (typically the earlier layers)
  2. Freeze these layers
     - This will set the corresponding parameters as non-trainable
       - Optimization: you can actually ***cache*** the outputs of frozen layers for every input
  3. Add your own layers hidden layer(s)
  4. Replace or discard upper layers
     - You should always discard the existing output layer and use your own

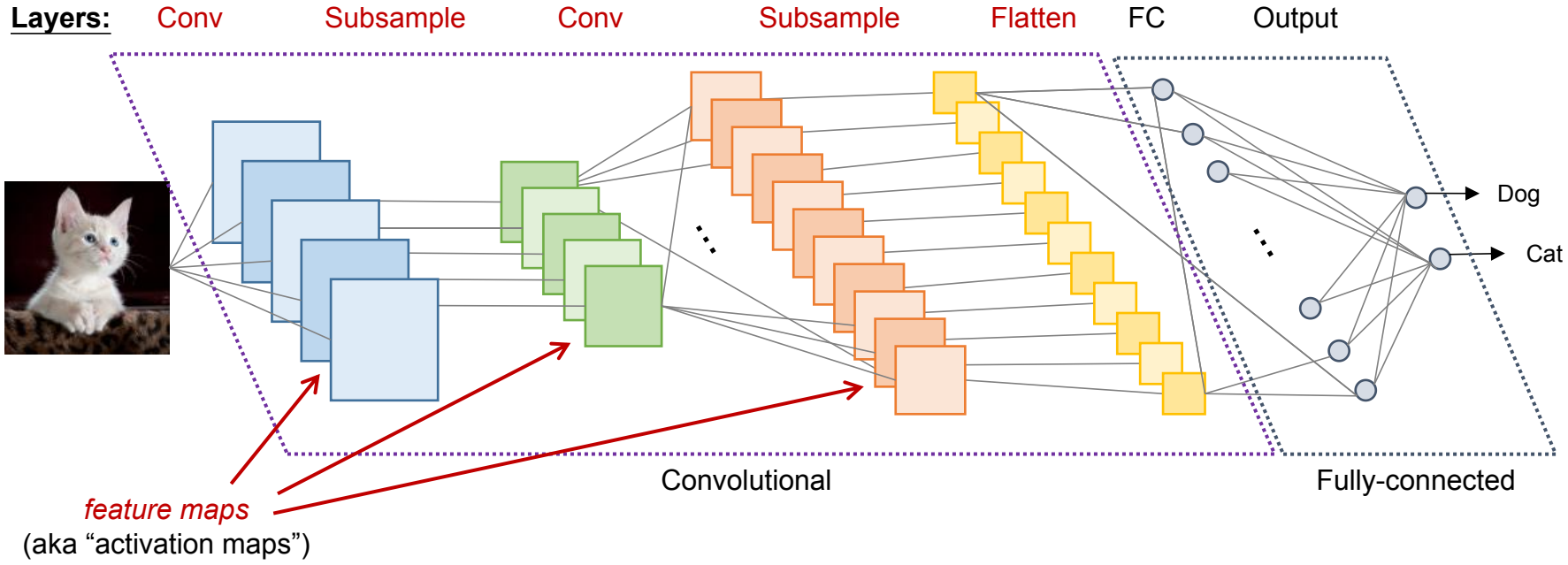


Convolutional Neural Networks

- History:
  - 1958: Hubel and Wiesel experiments on cats
    - Won the Nobel Prize in Physiology or Medicine (1981)
    - Insight: neurons in visual cortex have a small local receptive field
  - 1998: LeCun et al. propose the LeNet-5 architecture
- Convolutional Neural Networks:
  - Architecture for neural networks using convolutional layers
    - Convolutional layers: each neuron/unit is only connected to a small number of neurons/units in the previous layer
    - Fewer neurons/units than fully-connected layers
  - Well-suited to computer vision tasks or tasks on image data
    - Can also be applied to other tasks: for example some tasks in natural language processing
  - Preeminent neural network architectures for many state-of-the-art applications
    - E.g.: self-driving cars, video classification, image search systems, etc.
  - Remark: CNNs have high memory usage *during training*

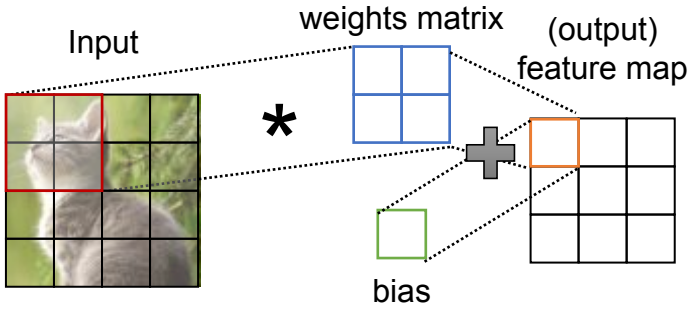
Convolutional Neural Network Architecture

- Example & Terminology:



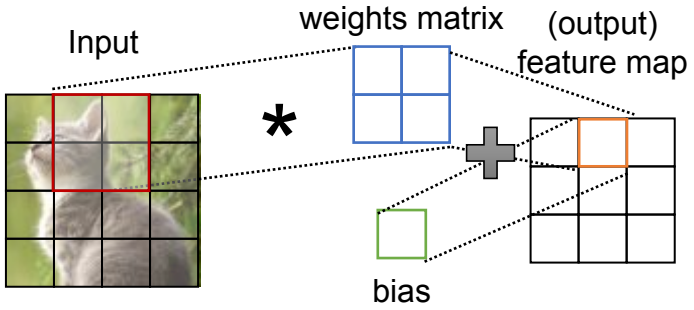
Convolutional Layer

- A convolutional layer has a set of filters (aka kernels)
  - Each filter slides (i.e., convolves) across the image (or previous layer's output) producing a feature map
  - The filter is represented by a $f_w \times f_h$ matrix of weights $\boldsymbol{F}$ and a bias b; there is also an activation function
    - Applying the filter produces a ***single*** *output value* (real number) for each sliding window
  - Parameters: weight matrix $\boldsymbol{F}$ and bias b
  - Hyperparameters: filter/kernel size (f_w, f_h), stride, padding strategy ('valid' or 'same'), and activation function



Convolutional Layer

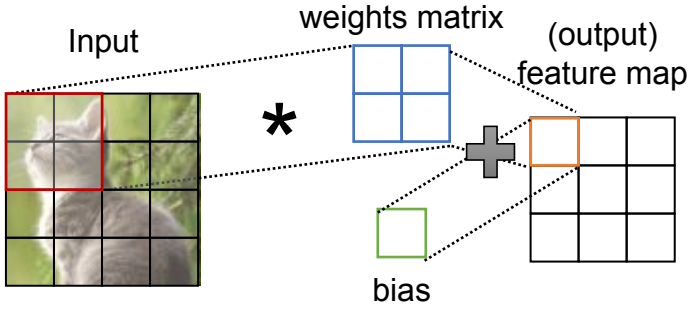
- A convolutional layer has a set of filters (aka kernels)
  - Each filter slides (i.e., convolves) across the image (or previous layer's output) producing a feature map
  - The filter is represented by a $f_w \times f_h$ matrix of weights $\boldsymbol{F}$ and a bias b; there is also an activation function
    - Applying the filter produces a ***single*** *output value* (real number) for each sliding window
  - Parameters: weight matrix $\boldsymbol{F}$ and bias b
  - Hyperparameters: filter/kernel size (f_w, f_h), stride, padding strategy ('valid' or 'same'), and activation function



- Remarks:
  - The weights matrix remains the same throughout the convolution
    - There are only $f_w\ f_h$ +1 parameters for the filter (and it does not depend on the size of the input)
  - Typically we have multiple filters per layer, so we get one feature map as output for each filter
  - Output size of feature map depends on the size of the filter, stride, and padding strategy

Convolutional Layer

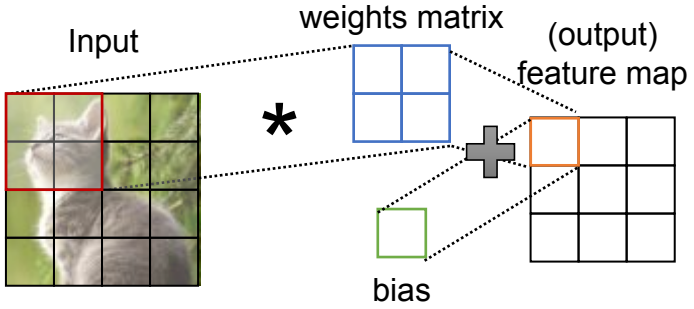
- A convolutional layer has a set of filters (aka kernels)
  - Each filter slides (i.e., convolves) across the image (or previous layer's output) producing a feature map
  - The filter is represented by a $f_w \times f_h$ matrix of weights $\boldsymbol{F}$ and a bias b; there is also an activation function
    - Applying the filter produces a ***single*** *output value* (real number) for each sliding window
  - Parameters: weight matrix $\boldsymbol{F}$ and bias b
  - Hyperparameters: filter/kernel size (f_w, f_h), stride, padding strategy ('valid' or 'same'), and activation function



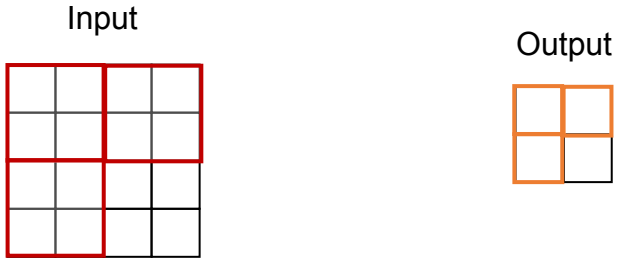
(2,2) filter, stride=1,
padding "valid" (no padding)

Convolutional Layer

- A convolutional layer has a set of filters (aka kernels)
  - Each filter slides (i.e., convolves) across the image (or previous layer's output) producing a feature map
  - The filter is represented by a $f_w \times f_h$ matrix of weights $\boldsymbol{F}$ and a bias b; there is also an activation function
    - Applying the filter produces a ***single*** *output value* (real number) for each sliding window
  - Parameters: weight matrix $\boldsymbol{F}$ and bias b
  - Hyperparameters: filter/kernel size (f_w, f_h), stride, padding strategy ('valid' or 'same'), and activation function

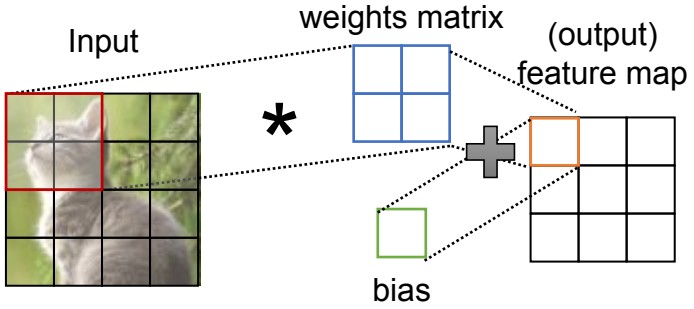


(2,2) filter, stride=**2**,
padding "valid" (no padding)

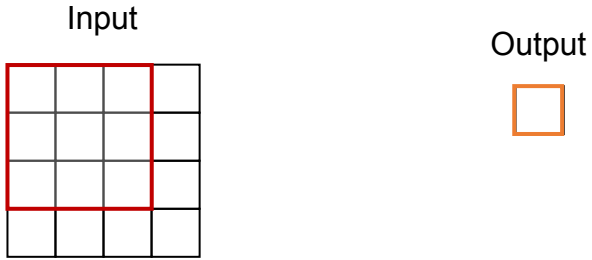


Convolutional Layer

- A convolutional layer has a set of filters (aka kernels)
  - Each filter slides (i.e., convolves) across the image (or previous layer's output) producing a feature map
  - The filter is represented by a $f_w \times f_h$ matrix of weights $\boldsymbol{F}$ and a bias b; there is also an activation function
    - Applying the filter produces a ***single*** *output value* (real number) for each sliding window
  - Parameters: weight matrix $\boldsymbol{F}$ and bias b
  - Hyperparameters: filter/kernel size (f_w, f_h), stride, padding strategy ('valid' or 'same'), and activation function

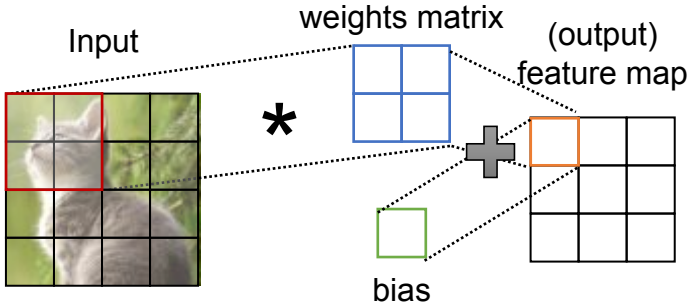


(3,3) filter, stride=2,
padding "valid" (no padding)

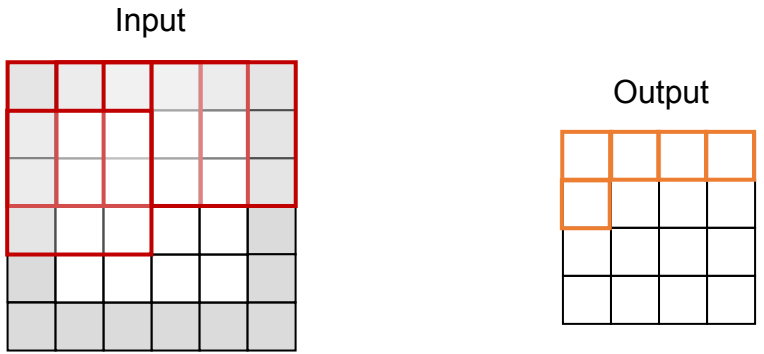


Convolutional Layer

- A convolutional layer has a set of filters (aka kernels)
  - Each filter slides (i.e., convolves) across the image (or previous layer's output) producing a feature map
  - The filter is represented by a $f_w \times f_h$ matrix of weights $\boldsymbol{F}$ and a bias b; there is also an activation function
    - Applying the filter produces a ***single*** *output value* (real number) for each sliding window
  - Parameters: weight matrix $\boldsymbol{F}$ and bias b
  - Hyperparameters: filter/kernel size (f_w, f_h), stride, padding strategy ('valid' or 'same'), and activation function



(3,3) filter, stride=1,
padding "**same**" (*zero* padding)



Next Time

- Wednesday (3/20): Lecture

- Upcoming:
  - Homework 3 is due 3/20
  - Project