CS/ECE 5381/7381 Computer Architecture Spring 2023

Dr. Manikas
Computer Science
Lecture 23: Apr. 21, 2023
(EXTRA LECTURE – RECORDED ONLY

Assignments

- Quiz 10 – due **Mon, Apr. 24** (11:59 pm)
  - Covers concepts from Module 11
    - Including extra lecture of Friday

Quiz 10 Details

- The quiz is open book and open notes.
- You are allowed 90 minutes to take this quiz.
- You are allowed 2 attempts to take this quiz - your highest score will be kept.
  - Note that some questions (e.g., fill in the blank) will need to be graded manually
- Quiz answers will be made available 24 hours after the quiz due date.

Domain-Specific Architectures

(Chapter 7, Hennessy and Patterson)

Note: some course slides adopted from publisher-provided material

Outline

- <u>7.1 Introduction</u>
- 7.2 Guidelines for DSAs
- 7.3 Example Domain: Deep Neural Networks
- 7.4 Google's Tensor Processing Unit

Introduction

- The computer architecture concepts described in previous lectures took advantage of "Moore's Law"
  - Gordon Moore (Fairchild, later cofounder of Intel)
  - 1960's: prediction
    - Number of transistors per chip will grow *exponentially* over time

What did Moore's Law enable?

- Deep memory hierarchy
- Pipelines
- Branch prediction
- Out-of-order execution
- Multithreading
- Multiprocessing

Limitations of Modern Designs

- Physical limitations of current designs ending Moore’s Law
- In order to continue improvements in computer performance, need to develop DSAs (Domain-Specific Architectures)
- As opposed to general core machines (which we studied previously), these are machines that are developed for specific **applications (domains)**

Outline

- 7.1 Introduction
- 7.2 Guidelines for DSAs
- 7.3 Example Domain: Deep Neural Networks
- 7.4 Google's Tensor Processing Unit

Guidelines for DSA Design

- Use dedicated memories to minimize data movement
- Invest resources into more arithmetic units or larger memories
- Use the easiest form of parallelism that matches the domain
- Reduce data size and type to the simplest needed for the domain
- Use a domain-specific programming language

Outline

- 7.1 Introduction
- 7.2 Guidelines for DSAs
- 7.3 Example Domain: Deep Neural Networks
- 7.4 Google's Tensor Processing Unit

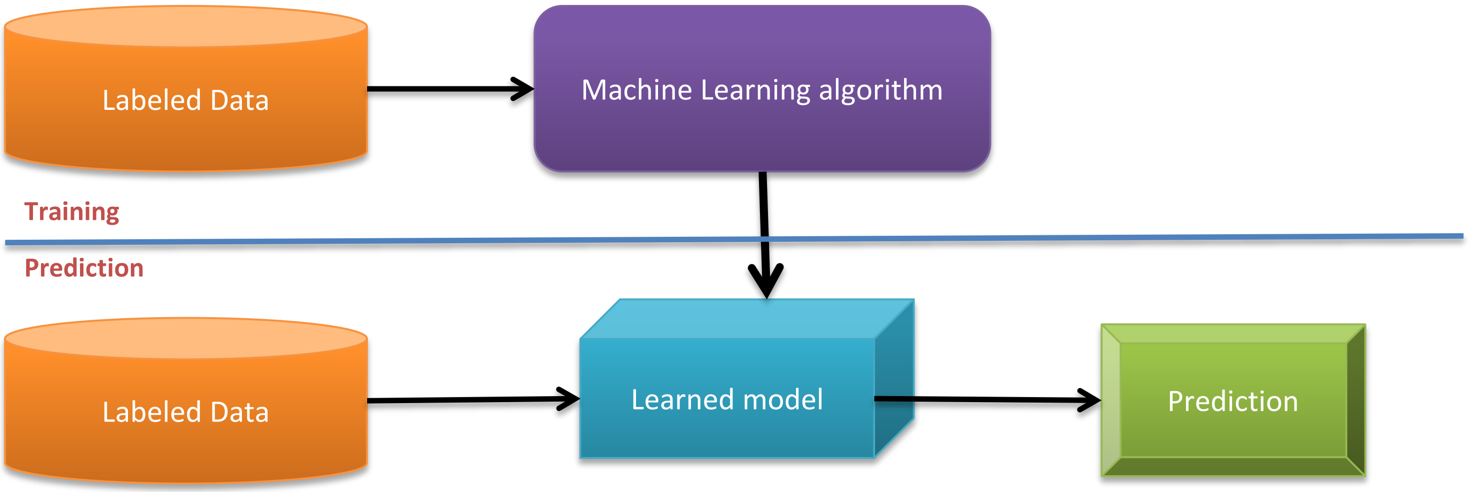
Deep Neural Networks

Background

Some materials provided by https://www.webpages.uidaho.edu/vakanski/Courses/Adversarial_Machine_Learning/Fall_2021/Lecture_2_Deep_Learning_Overview.pptx

Machine Learning Basics

- ***Artificial Intelligence (AI)*** is a scientific field concerned with the development of algorithms that allow computers to learn without being explicitly programmed
- ***Machine Learning (ML)*** is a branch of Artificial Intelligence, which focuses on methods that learn from data and make predictions on unseen data



Machine Learning Strategies

- **Supervised Learning:** training set contains data and the correct output of a given task with that data
- **Unsupervised Learning:** the training set contains data, but no solutions

Examples of Supervised Learning

- **Classification Algorithms:** training set is dataset and class of each piece of data
  - computer learns how to classify new data
- **Regression Algorithms:** predict a value of an entity's attribute

Examples of Unsupervised Learning

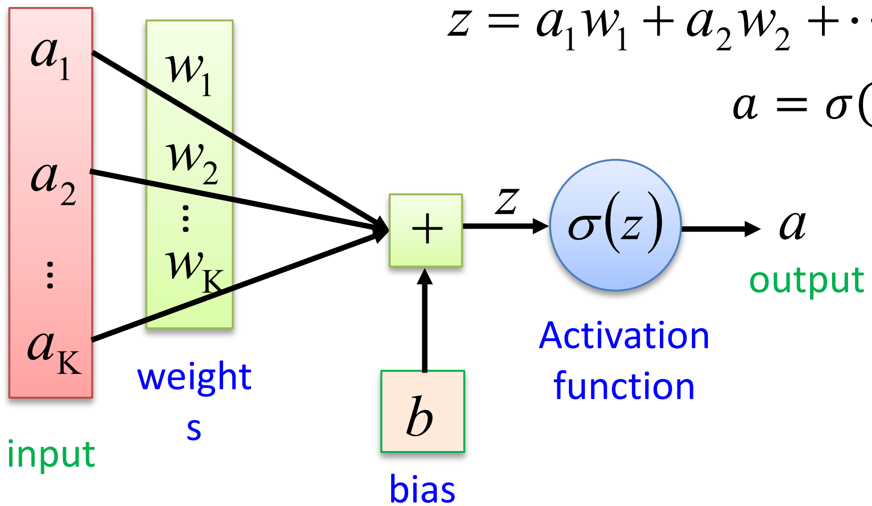
- **Clustering Algorithms:** training set is dataset covering various dimensions
  - data are partitioned into clusters based on specified criteria.
- **Dimensionality Reduction Algorithms:** training set is also dataset covering various dimensions
  - algorithm projects the data to fewer dimensions
  - Goal: attempt to better capture the fundamental aspects of the original data

Neural Networks

- **Neural Networks** (NNs) can be applied to all the basic machine learning algorithms
- NNs consist of basic computational units (**neurons**) that are organized in layers
  - Synaptic weights connect neurons of adjacent layers
  - Each neuron computes a transfer function, which is a weighted sum of outputs generated by the previous layer, then applies an activation function

Elements of Neural Networks (NNs)

- NNs consist of hidden layers with neurons (i.e., computational units)
- A single neuron maps a set of inputs into an output number, or $f: R^K \rightarrow R$



$$z = a_1 w_1 + a_2 w_2 + \cdots + a_K w_K + b$$

$$a = \sigma(z)$$

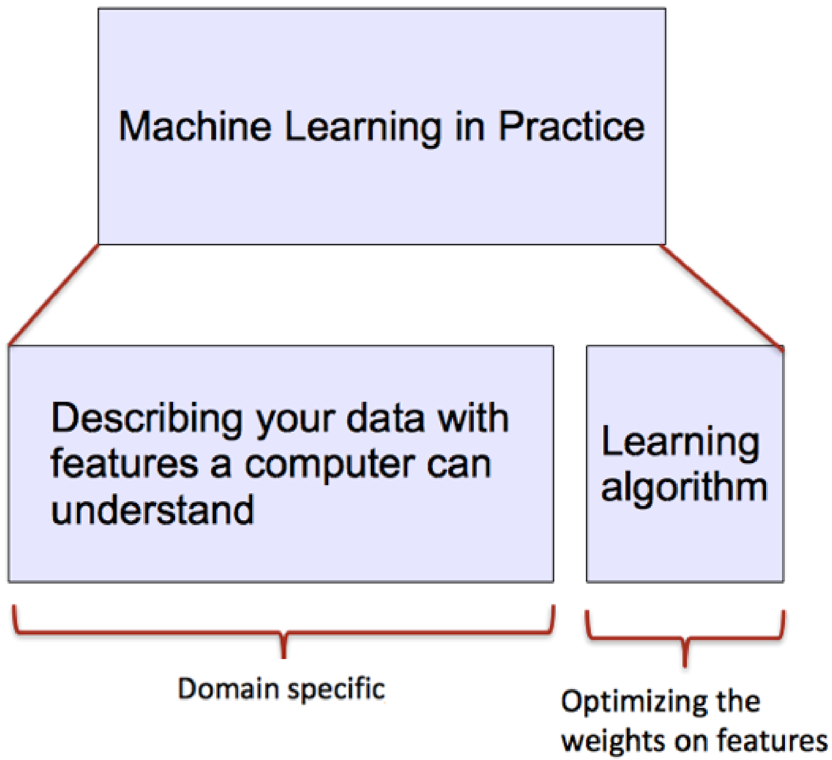
Slide credit: Hung-yi Lee – Deep Learning Tutorial

Training Neural Networks

- Common training method: **backpropagation**
  1. **Forward Propagation:** feed forward patterns using connection weights, then compare the output of the last layer with the expected value to compute the training error.
  2. **Backward Propagation:** feed the training error back to the network to adjust the synaptic weights of the neurons.
  3. Repeat Steps 1 and 2 until a specified stopping criterion is satisfied.
- One iteration cycle (Steps 1 and 2) is called an **epoch**.

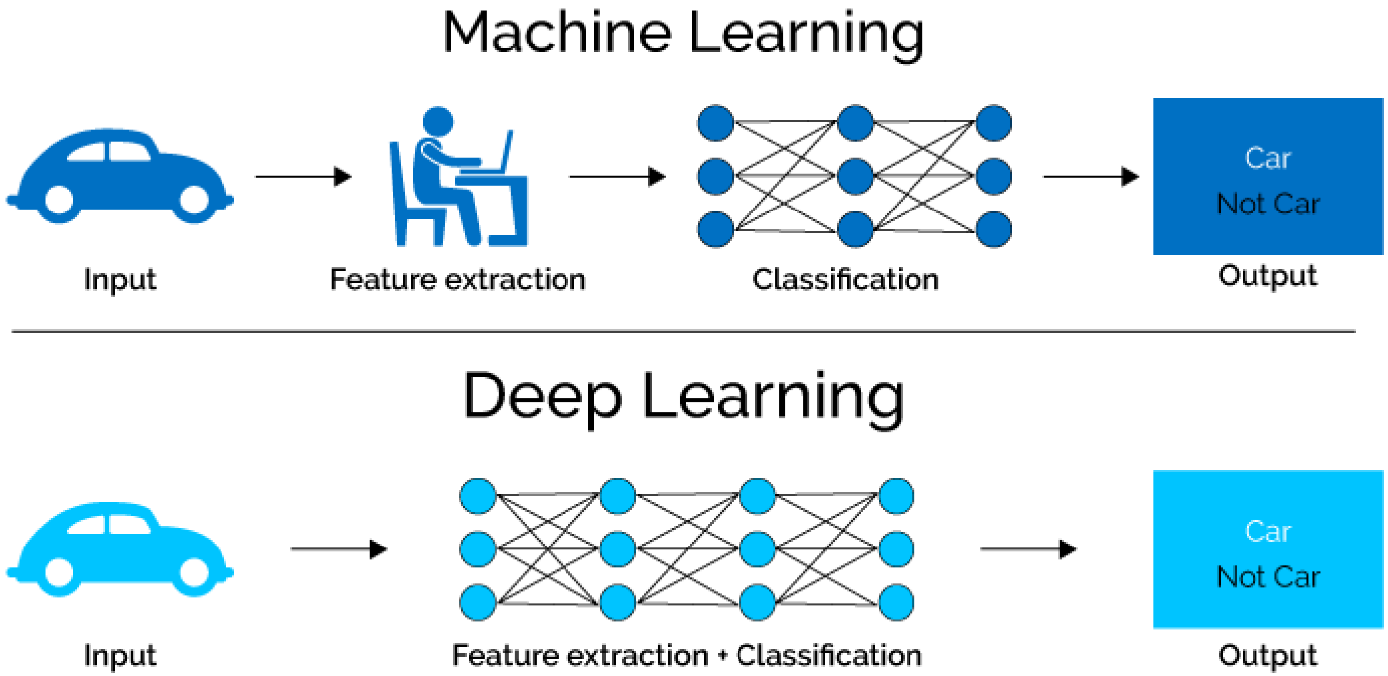
ML vs. Deep Learning

- Conventional machine learning methods rely on human-designed feature representations
  - ML becomes just optimizing weights to best make a final prediction



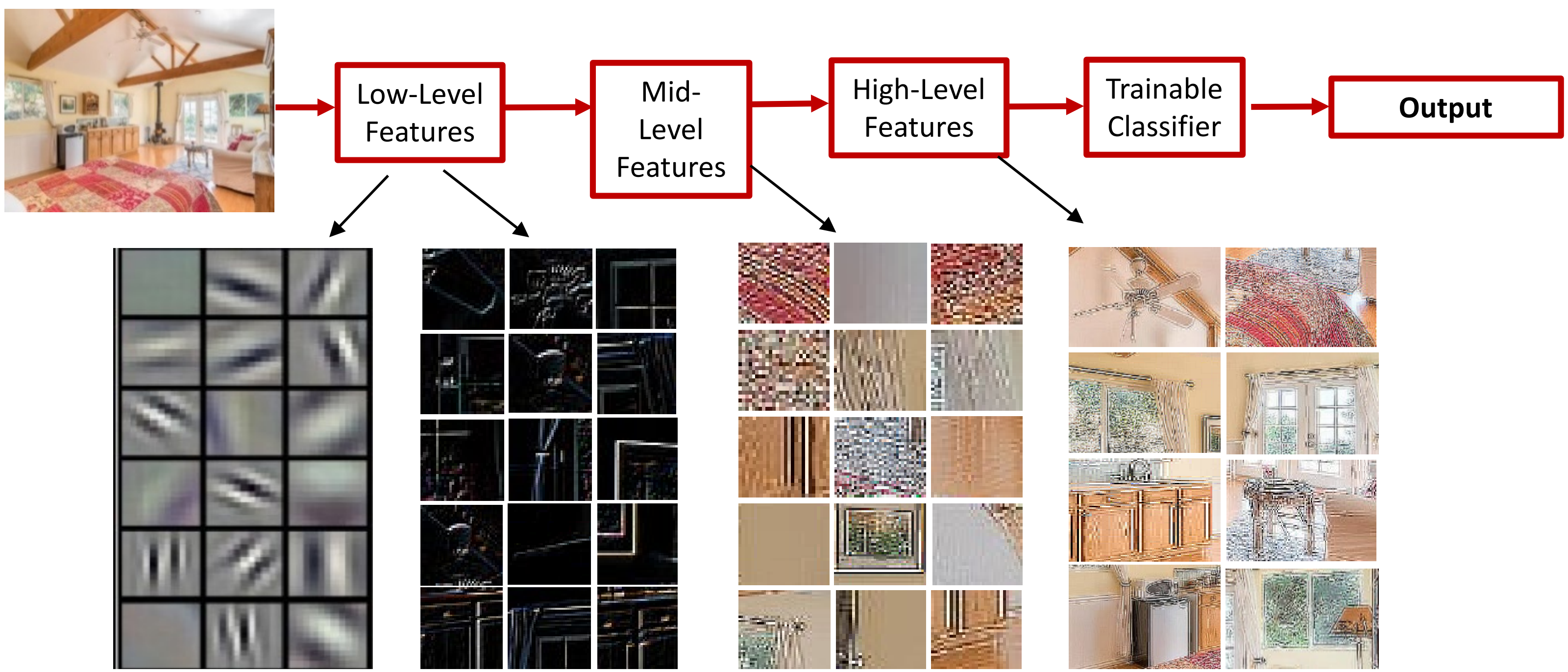
ML vs. Deep Learning

- *Deep learning* (DL) is a machine learning subfield that uses multiple layers for learning data representations
  - DL is exceptionally effective at learning patterns



ML vs. Deep Learning

- DL applies a multi-layer process for learning rich hierarchical features (i.e., data representations)
  - Input image pixels → Edges → Textures → Parts → Objects



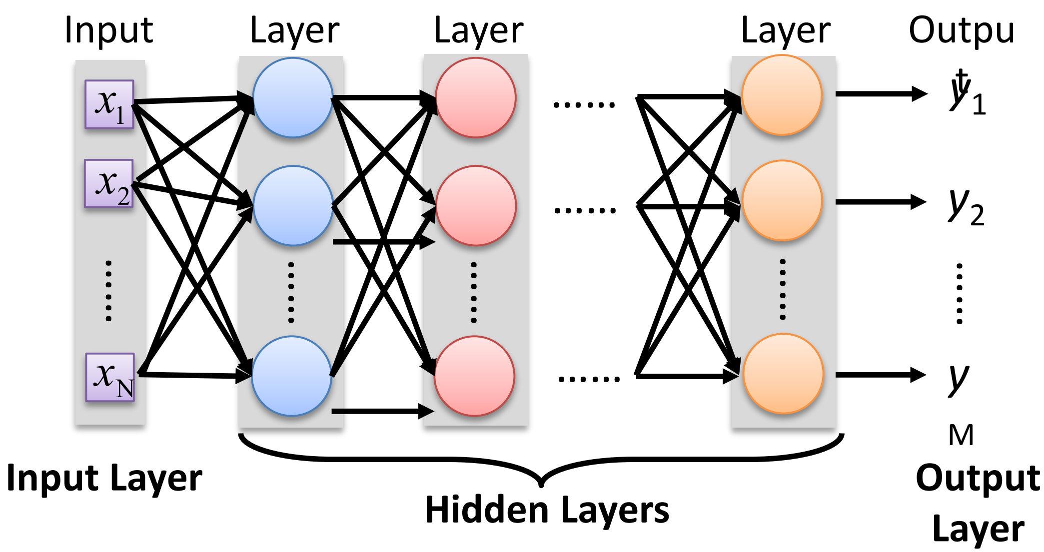
Why is DL Useful?

- DL provides a flexible, learnable framework for representing visual, text, linguistic information
  - Can learn in supervised and unsupervised manner
- DL represents an effective end-to-end learning system
- Requires large amounts of training data
- Since about 2010, DL has outperformed other ML techniques
  - First in vision and speech, then NLP (Natural Language Processing), and other applications

Elements of Deep Neural Networks (DNN)

Introduction to Neural Networks

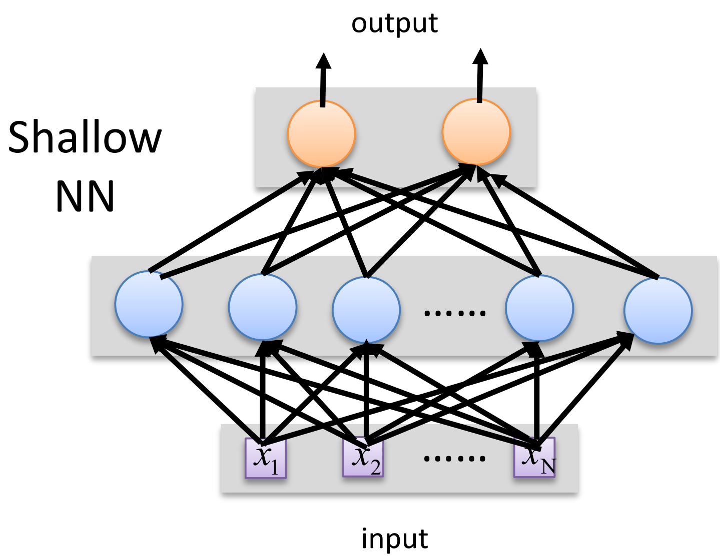
- Deep NNs have many hidden layers
  - Fully-connected (dense) layers (a.k.a. Multi-Layer Perceptron or MLP)
  - Each neuron is connected to all neurons in the succeeding layer

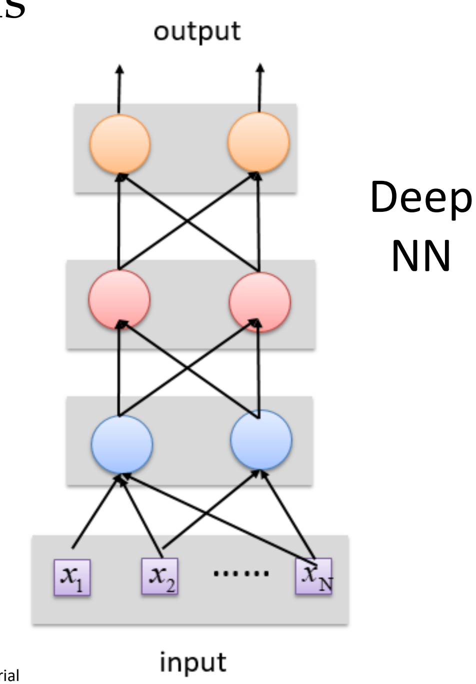


Slide credit: Hung-yi Lee – Deep Learning Tutorial

Deep vs Shallow Networks

- Deeper networks perform better than shallow networks
  - But only up to some limit: after a certain number of layers, the performance of deeper networks plateaus





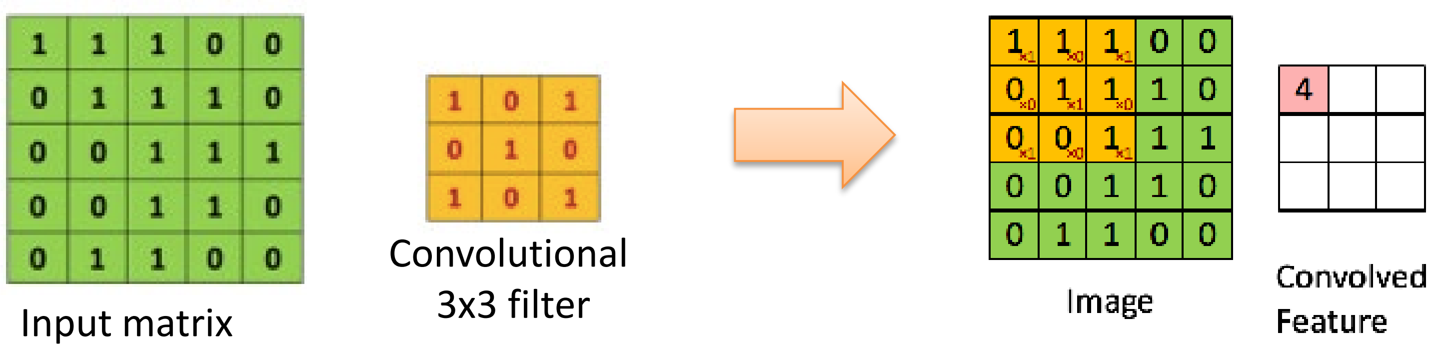
Slide credit: Hung-yi Lee – Deep Learning Tutorial

Example: Convolutional Neural Networks

- Convolutional Neural Network (CNN) is an NN with the following layers:
  1. **Convolution Layer**: allows local properties in the data to be discovered
  2. **Local Contrast Normalization (LCN) Layer**: limits the effects of intensity variations over various feature maps
  3. **Pooling/Subsampling Layer**: adds robustness to small shifts of input data by looking at groups of input neurons
  4. **Fully Connected Layer**: last layer in CNN. This is the common NN layer where every input neuron is connected to each layer neuron. Acts as a classifier by separating the input data space.

Convolutional Neural Networks (CNNs)

- *Convolutional neural networks* (CNNs) were primarily designed for image data
- CNNs use a convolutional operator for extracting data features
  - Allows parameter sharing
  - Efficient to train
  - Have less parameters than NNs with fully-connected layers
- CNNs are robust to spatial translations of objects in images
- A convolutional filter slides (i.e., convolves) across the image



CNN Applications

- CNN’s are often applied to pattern-recognition problems, where vectors are used to represent real signals
  - For images, the vector becomes a matrix
  - CNN’s are able to capture local properties of pixel groups within the matrix

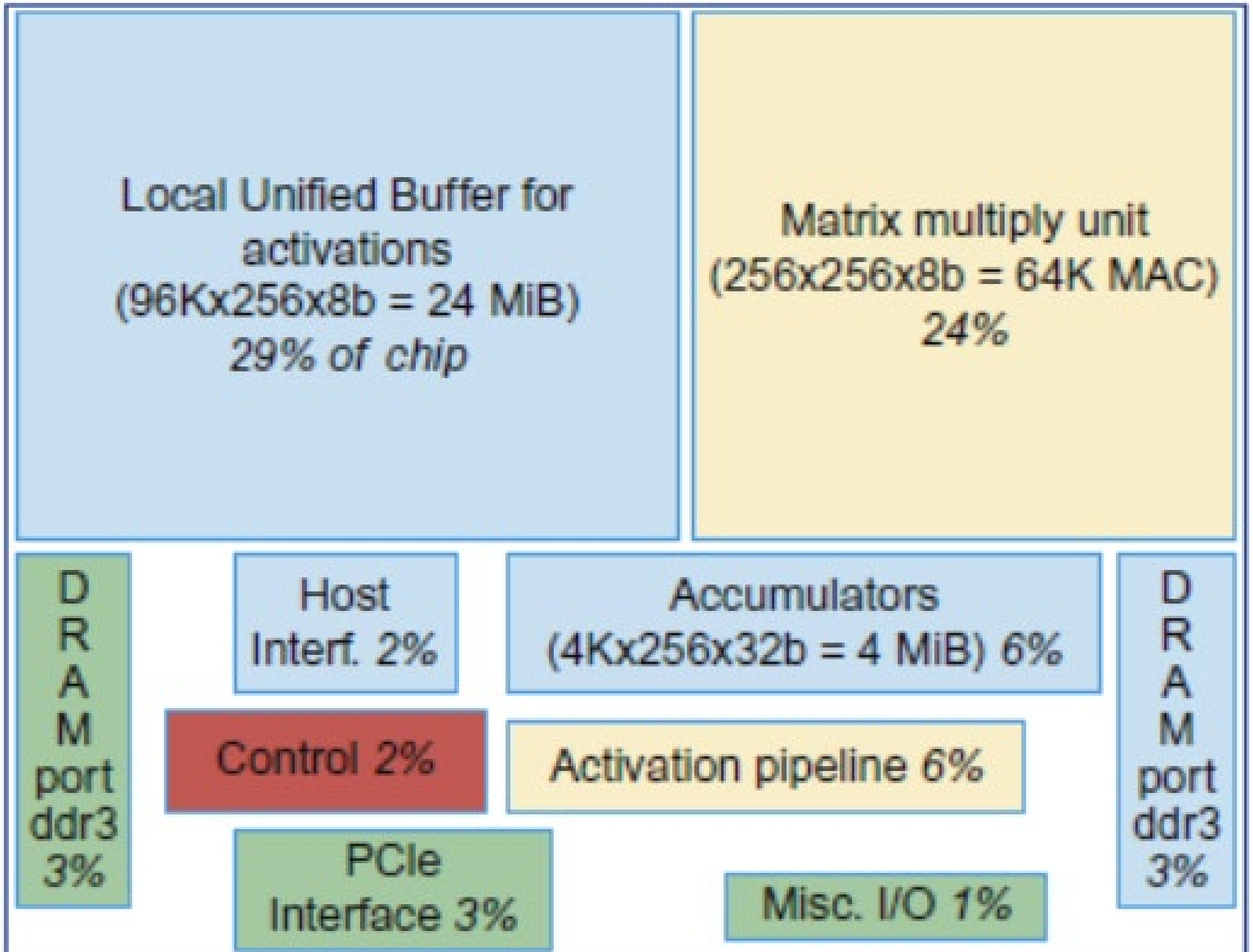
Outline

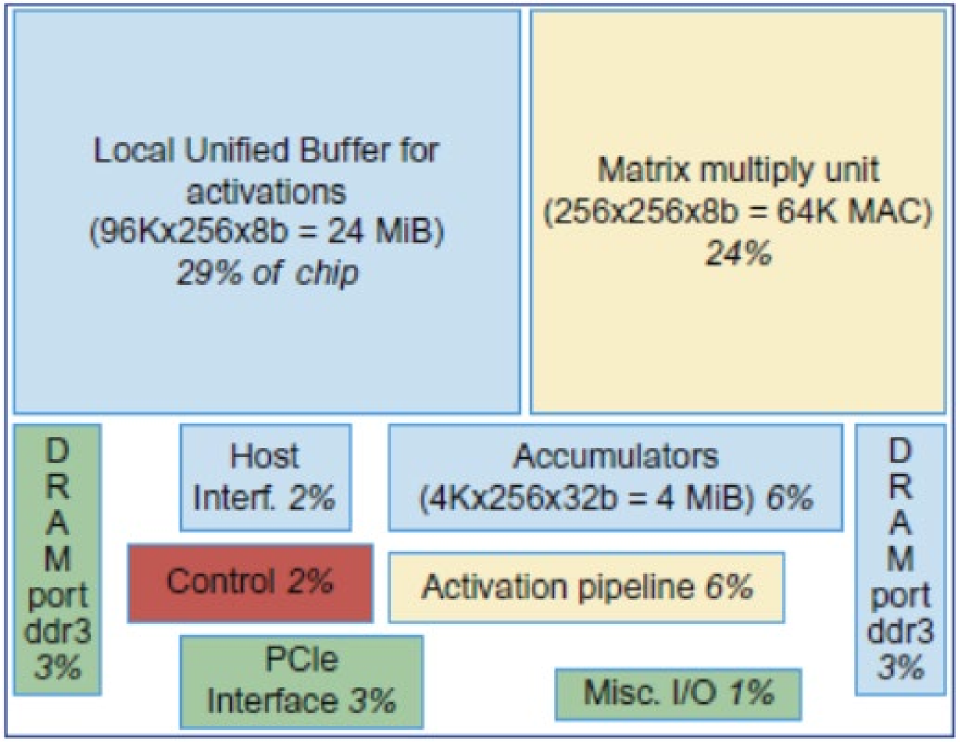
- 7.1 Introduction
- 7.2 Guidelines for DSAs
- 7.3 Example Domain: Deep Neural Networks
- 7.4 Google's Tensor Processing Unit

Tensor Processing Unit (TPU)

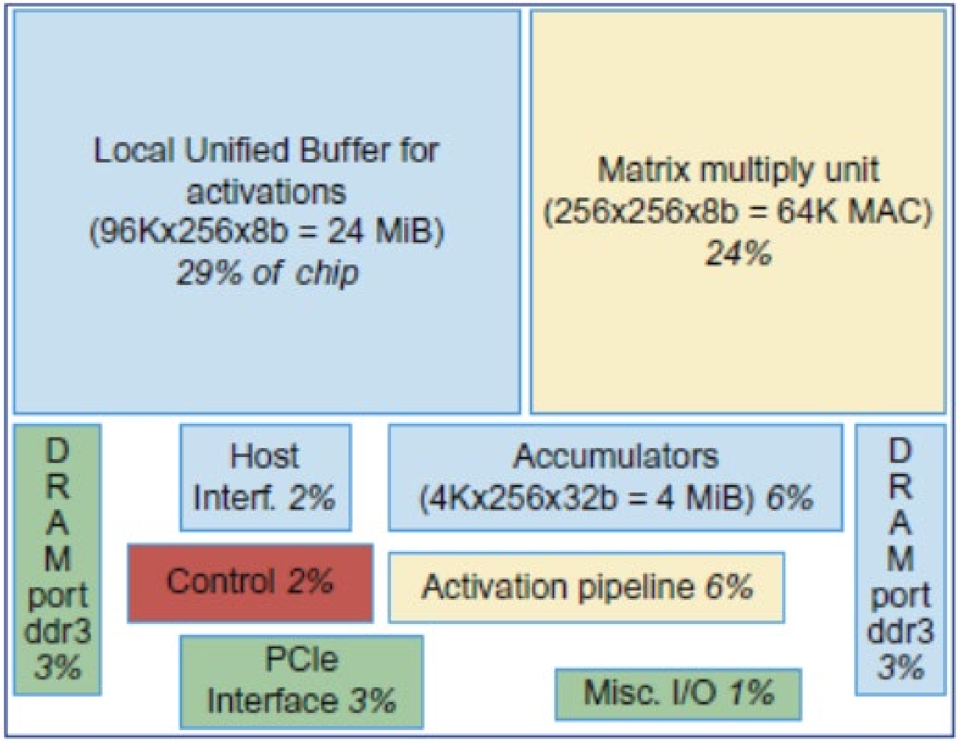
- Google’s DNN ASIC
  - Chip to implement Deep Neural Networks
- 256 x 256 8-bit matrix multiply unit
  - Recall: CNNs use vectors and matrices to represent their data

TPU ISA


Local Unified Buffer for activations
(96Kx256x8b = 24 MiB)
29% of chip
Matrix multiply unit
(256x256x8b = 64K MAC)
24%
D R A M port ddr3 3%
Host Interf. 2%
Accumulators
(4Kx256x32b = 4 MiB) 6%
D R A M port ddr3 3%
Control 2%
Activation pipeline 6%
PCIe Interface 3%
Misc. I/O 1%



Local Unified Buffer: data storage



Matrix multiply unit: Performs a matrix-matrix or vector-matrix multiply from the Unified Buffer into the accumulators

- takes a variable-sized B*256 input, multiplies it by a 256x256 constant input, and produces a B*256 output, taking B pipelined cycles to complete

TPU ISA Operation

- Read_Host_Memory
  - Reads memory from the CPU memory into the unified buffer
- Read_Weights
  - Reads weights from the Weight Memory into the Weight FIFO as input to the Matrix Unit
- MatrixMatrixMultiply/Convolve
  - takes a variable-sized B*256 input, multiplies it by a 256x256 constant input, and produces a B*256 output, taking B pipelined cycles to complete
- Activate
  - Computes activation function
- Write_Host_Memory
  - Writes data from unified buffer into host memory