

# ECE382M.20: System-on-Chip (SoC) Design

---

## Lecture 1 – Project Overview

Andreas Gerstlauer  
Electrical and Computer Engineering  
University of Texas at Austin  
gerstl@ece.utexas.edu



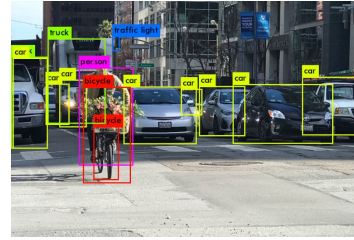
## Lecture 1: Outline

---

- **Marketing requirements**
  - Market focus, product description
  - Cost metrics, product features
- **Product requirements**
  - Deep learning
  - Hardware acceleration
- **Project description**
  - Deep/Convolutional Neural Networks (DNNs/CNNs)
  - Object recognition
  - You Only Look Once (YOLO) CNN
  - Hardware and software development tasks

## Market Focus

- **Visual object recognition**
  - Computer vision for drones, self-driving cars, home automation, ...
  - Camera-based automotive driver assistance systems (ADAS)



Source: Jonathan Hui

### ➤ What problem are we trying to solve?

- Standard camera for
  - Collision avoidance
  - Lane tracking/keeping
  - Traffic sign recognition
  - ...
- Detect, locate and classify objects in video stream
  - Bounding boxes
  - Types of objects



## Competition

- **MobilEye (an Intel company)**
  - <http://mobileye.com>
  - Custom ASIC/SoC solution
- **Movidius (an Intel company)**
  - <https://www.movidius.com/>
  - Custom ASIC/SoC solution
- **NVIDIA DRIVE PX**
  - <https://www.nvidia.com/en-us/self-driving-cars>
  - ARM+GPU (Tegra) based solution
  - Used by Tesla

## Product Description

---

- **Visual object recognition SoC**
  - Deliver hardware + software intellectual property (IP)
- **Cost metrics**
  - Real-time: frames per second (FPS), reaction time
  - Detection accuracy: mean average precision (mAP)
  - Power/thermal: W and operating temperature (°C)
  - Cost: \$ or die area (mm<sup>2</sup>)
- **Product features**
  - Supported image resolutions
  - Supported detection classes
  - Flexibility: dynamic, over-the-air reprogramming/updating

## Product Requirements

---

- **High detection accuracy → deep learning**
  - Convolutional Neural Network (CNN)
  - Trained on large image data set
  - Very computationally intensive
- **High frame rate, low power → hardware acceleration**
  - Key/dominating computational kernels
  - Convolutions and matrix operations
  - General matrix-matrix multiplication (GEMM)
- **Flexibility → software support**
  - Standard embedded Linux environment
  - Software optimizations for performance and power

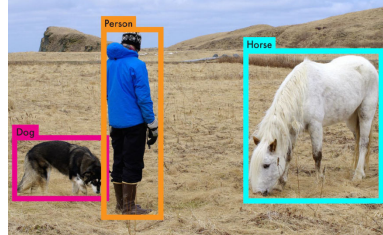
## Objection Detection using Deep Learning

- **Classification vs. detection**

Image classification, global feature



Object detection, classification + localization

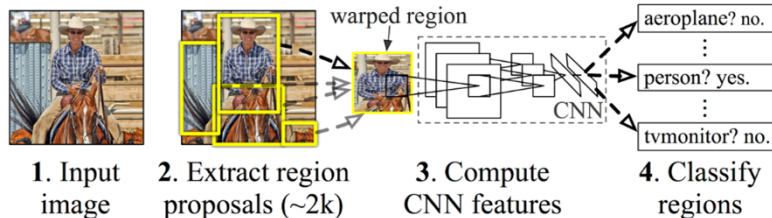


- Convolutional neural networks (CNNs) widely used for image classification
- Sliding windows of different size/shape + CNN-based classification for brute-force, naïve object detection

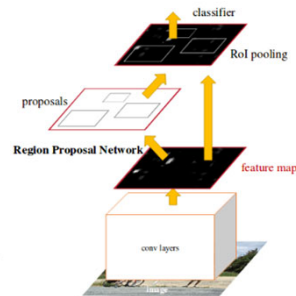
## Object Recognition (1)

- **Region based (Fast/Faster/Mask R-CNN)**

### R-CNN: Regions with CNN features

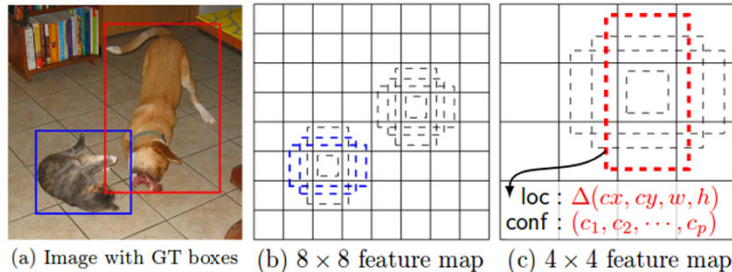


- Fast versions by sharing convolutional layers
- Common feature extraction for region proposal and classification



## Object Recognition (2)

- **Single Shot Multibox Detector (SSD)**

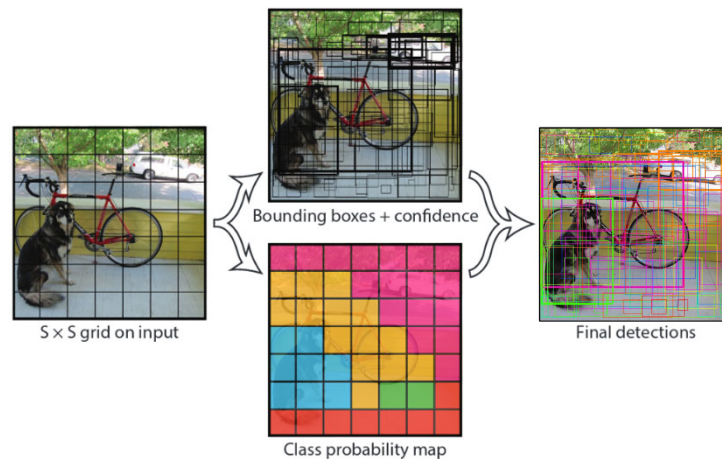


- Apply windows of fixed size and shape at multiple scales
- Detect both bounding box and class within window
- Predict likelihood of different box/class combinations

## Object Detection (3)

- **You Only Look Once (YOLO)**

- Single grid/scale, predict arbitrary bounding box (and class)



## You Only Look Once (YOLO)



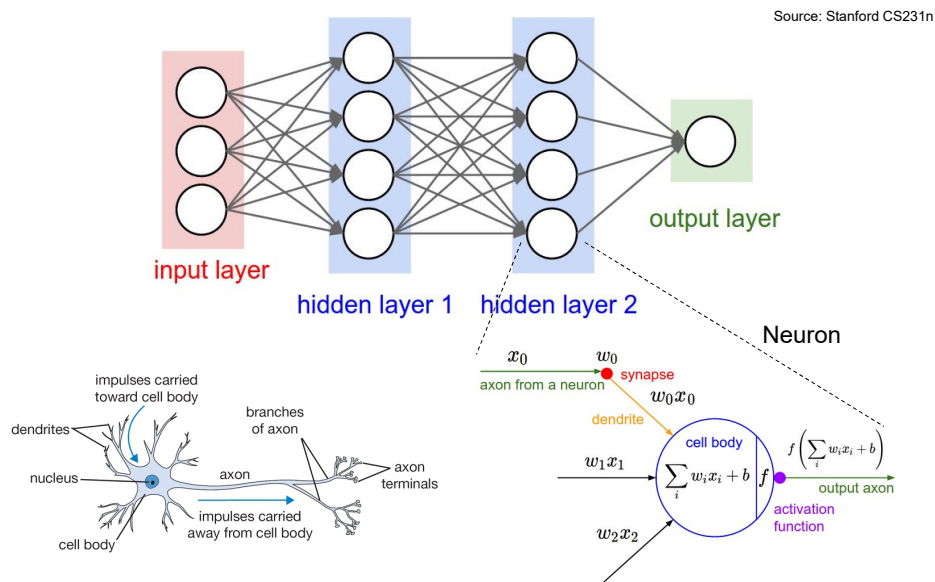
- **Default implementation on top of Darknet**
  - General open-source CNN framework/library in C
- **Also available for other deep learning frameworks**
  - PyTorch, Caffe2 [Facebook], TensorFlow [Google]

ECE382M.20: SoC Design, Lecture 1

© 2021 A. Gerstlauer

11

## Traditional Neural Networks



➤ **Deep Neural Networks (DNNs) with many hidden layers**

ECE382M.20: SoC Design, Lecture 1

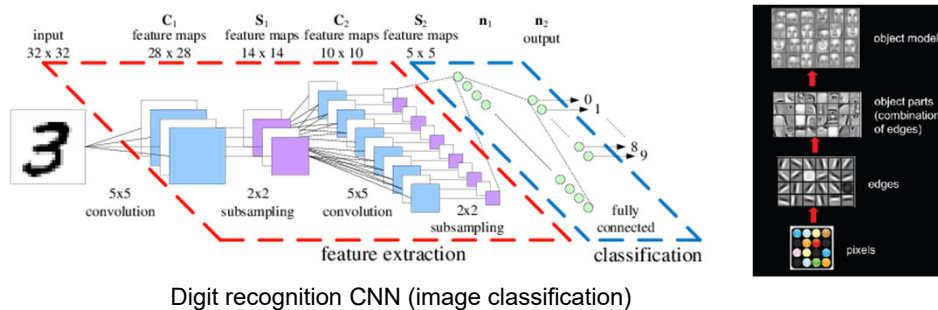
© 2021 A. Gerstlauer

12

## Convolutional Neural Networks (CNNs)

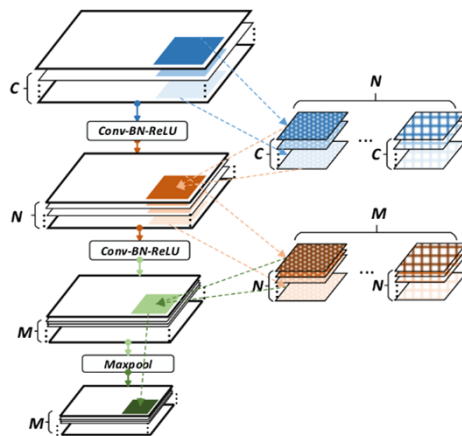
- **Different types of layers**

- Convolutional: convolutions with trainable filters
  - Rectified linear units (ReLU): elementwise function
  - Pooling: non-linear down-sampling
  - Fully connected: traditional neural networks
- } usually combined

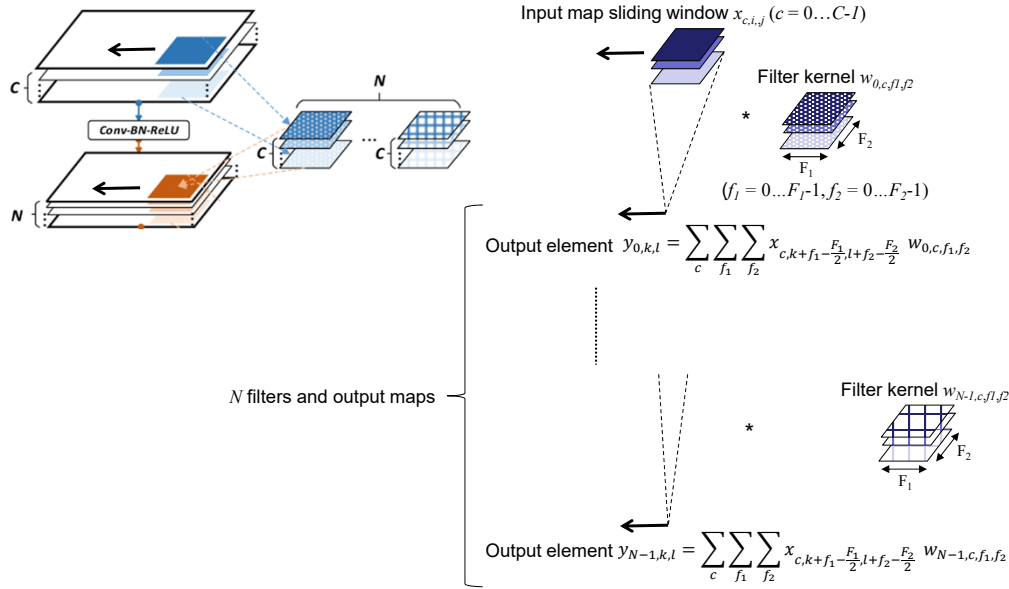


- **Fully convolutional network (FCN): no fully connected layer**

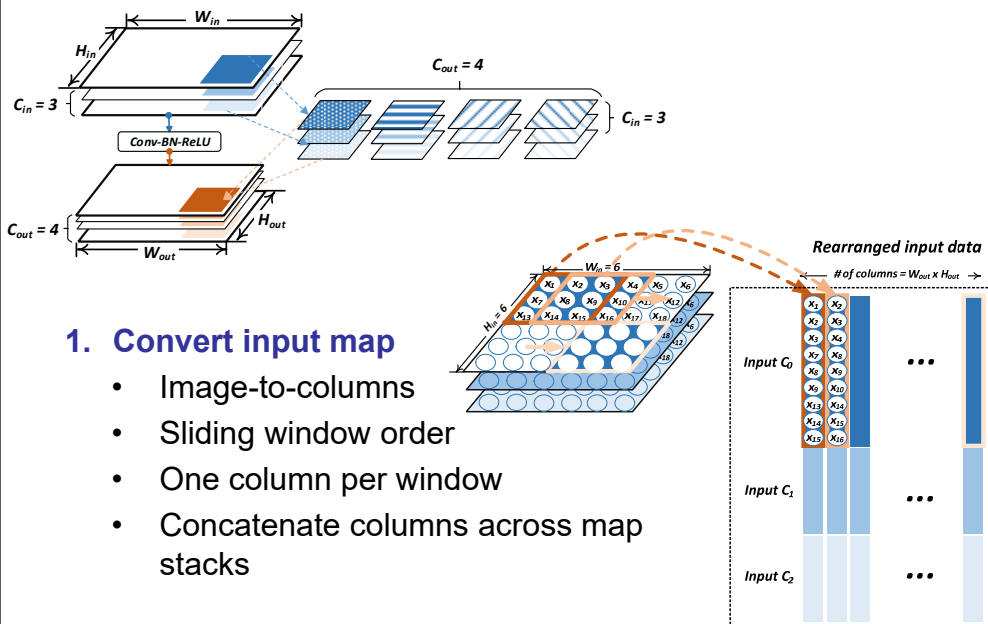
## Convolution Operations



## Convolution Operations

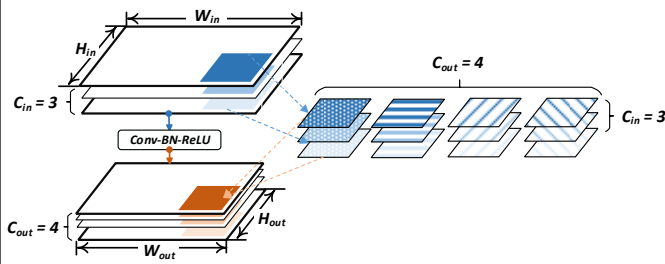


## Casting Convolutions as GEMM



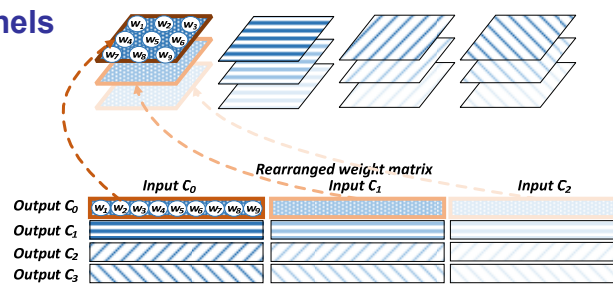


## Casting Convolutions as GEMM

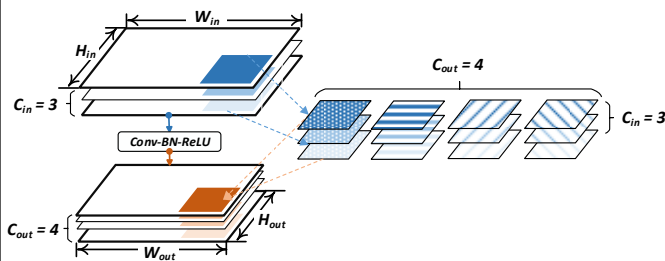


### 2. Convert filter kernels

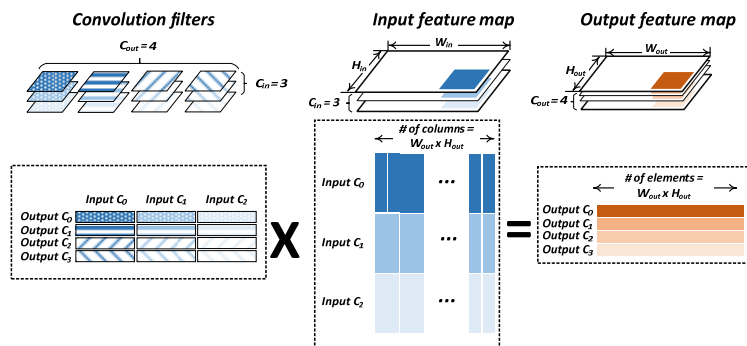
- One row per filter stack



## Casting Convolutions as GEMM



### 3. Perform GEMM



## Project Description

---

- **HW/SW co-design of an embedded SoC**
  - Low-power YOLO/Darknet implementation
  - ARM-based target platform
    - ARM Cortex-A9 processor, memory components, I/O devices
    - Custom hardware accelerators (GEMM)
    - Interconnected via standard system busses or memory/cache interfaces
  - Virtual and physical prototyping
    - SystemC TLM-based virtual platform model (QEMU ARM simulator)
    - ARM- and Xilinx FPGA-based prototyping board (Zynq-7000)
- Lab and project in teams
  - 2 per team, 42 teams for 21 boards

## Project Objectives and Activities

---

- **Project objective:**
  - Implement the YOLO/Darknet code on a ARM based SoC while meeting the performance, area and power metrics.
- **Project activities:**
  - Profile the YOLO/Darknet software implementation to determine performance bottlenecks
  - Optimize the YOLO/Darknet software (fixed point operation)
  - Partition the software into components which will run on the ARM processor and on hardware accelerators
  - Synthesize accelerators into Verilog for gate level implementation
  - Co-simulate and prototype the HW/SW implementation
  - Estimate timing, area and power metrics and validate against product requirements

