Edge Computing

Lecture 05: Basics of ML

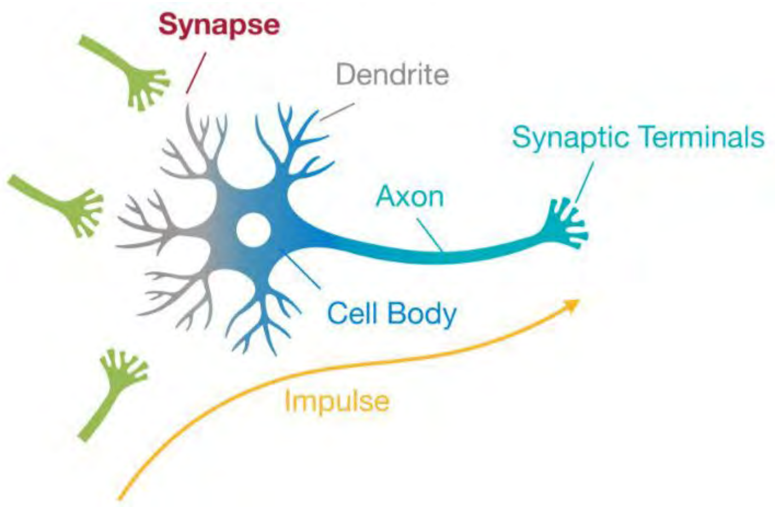
Recap

- Edge System Evaluation
  - System Metrics vs Performance Metrics
- Edge System Optimization
  - Architecture design
  - Device optimization
- Concurrency
  - SIMD
  - Threading & Multiprocessing

Agenda

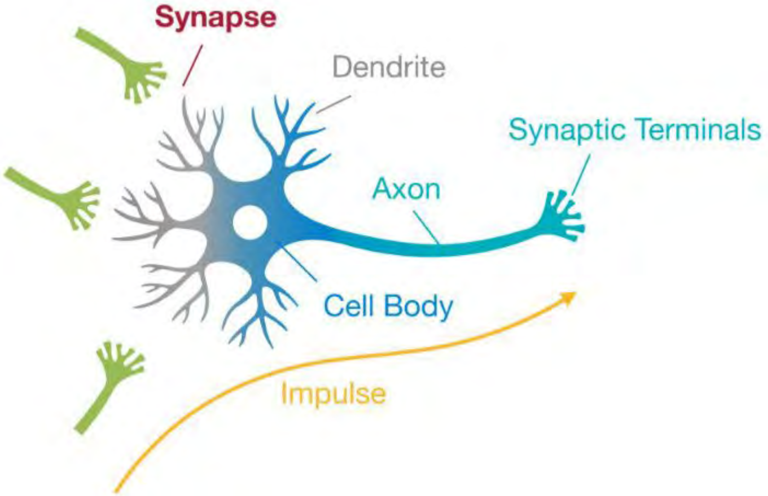
- Neural network: terminology
- Common building block: layer
- Convolution neural network
- Lab 2: object detection

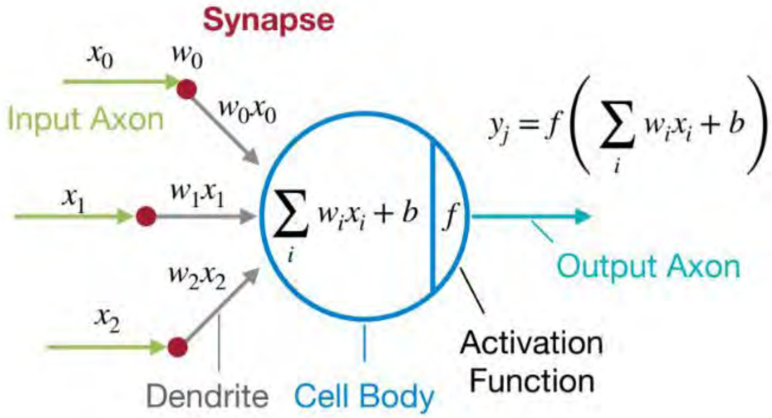
Neuron



- ***Synapse***: a small gap at the end of a neuron that allows a signal to pass from one neuron to the next
- ***Dendrite***: a short branched extension of a nerve cell, along which impulses received from other cells at synapses are transmitted to the cell body.
- ***Axon (Nerve fiber)***: The long portion of a neuron that conducts impulses away from the body of the cell.

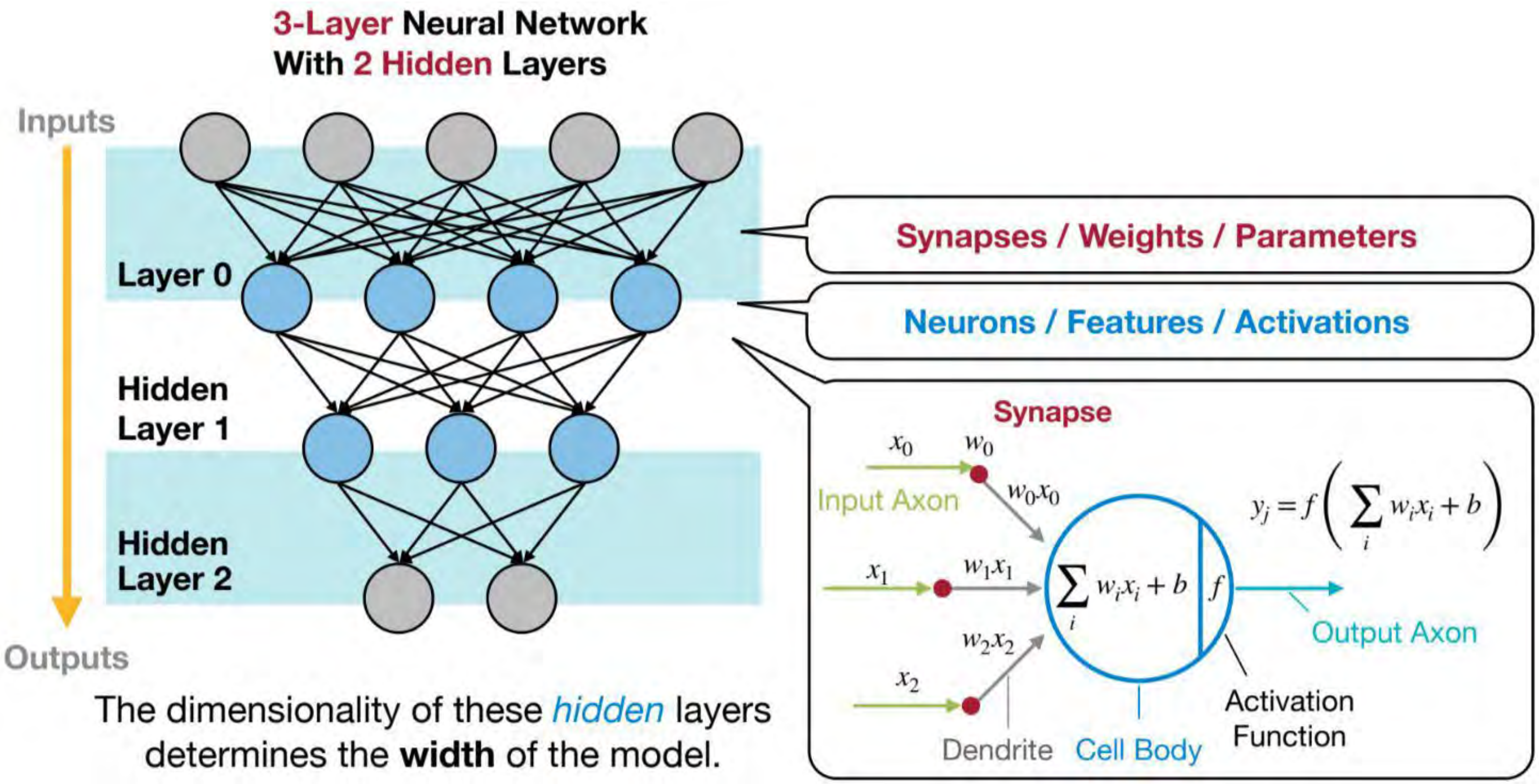
Modeling Neuron





https://hanlab.mit.edu/courses/2023-fall-65940

Neural Network

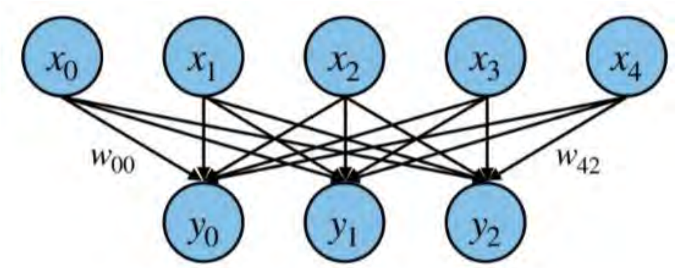


https://hanlab.mit.edu/courses/2023-fall-65940

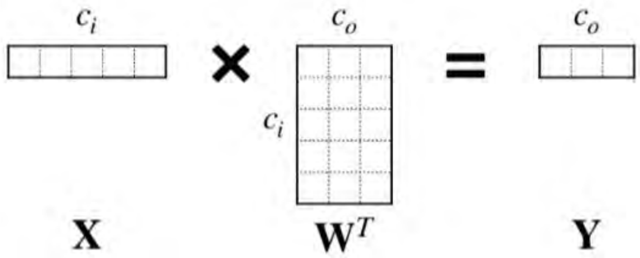
Agenda

- Neural network: terminology
- Common building block: layer
- Convolution neural network
- Lab 2: object detection

Fully-Connected (FC) Layer



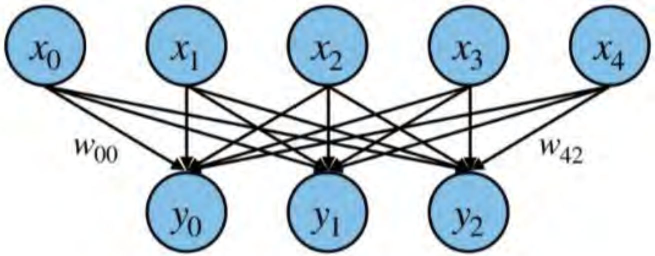
$$y_i = \sum_j w_{ij} x_j + b_i$$



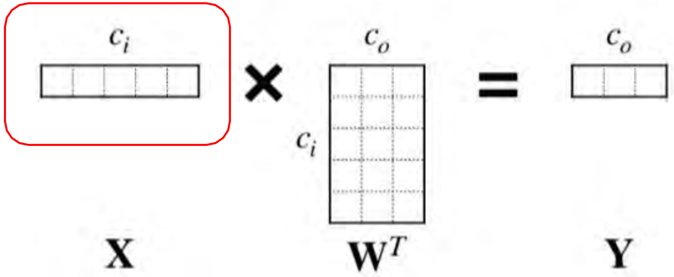
Tensor as in Tensorflow

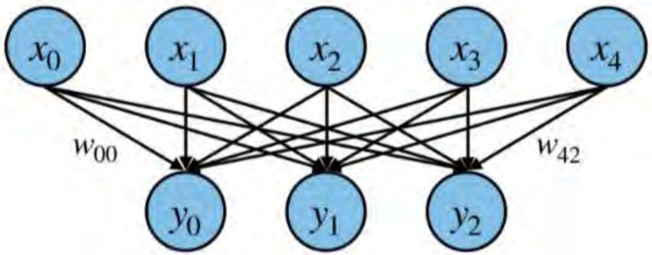
- In ML context, ***Data Tensor*** is simply a multidimensional array
- Strict math, Scalar, Vector, Tensor
    - Multi-linear mapping between set of domain vector spaces to a range vector space

Fully-Connected (FC) Layer

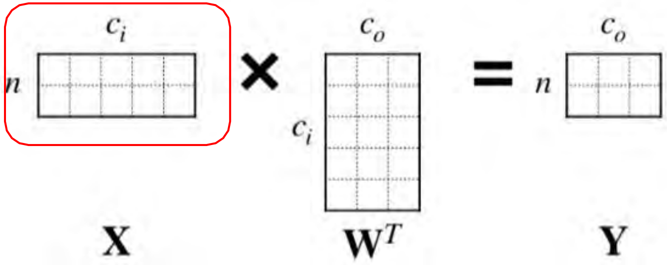


$$y_i = \sum_j w_{ij} x_j + b_i$$





$$y_i = \sum_j w_{ij} x_j + b_i$$



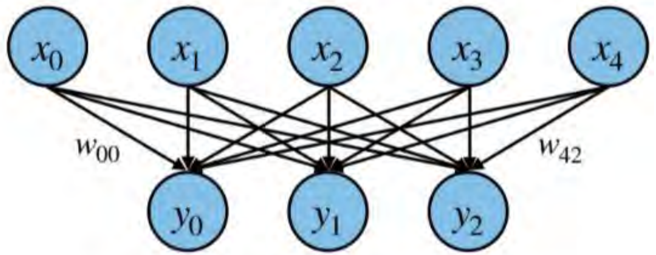
https://hanlab.mit.edu/courses/2023-fall-65940

Fully-Connected (FC) Layer

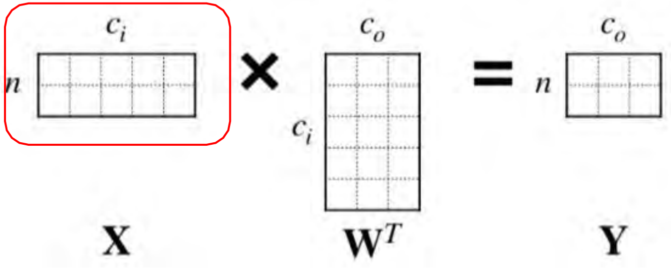
Everything is a tensor!

- **Shape of Tensors:**
  - Input Features $\mathbf{X}$: (n, c_i)
  - Output Features $\mathbf{Y}$: (n, c_o)
  - Weights $\mathbf{W}$: (c_o, c_i)
  - Bias $\mathbf{b}$: $(c_o,)$

Notations	
n	Batch Size
c_i	Input Channels
c_o	Output Channels

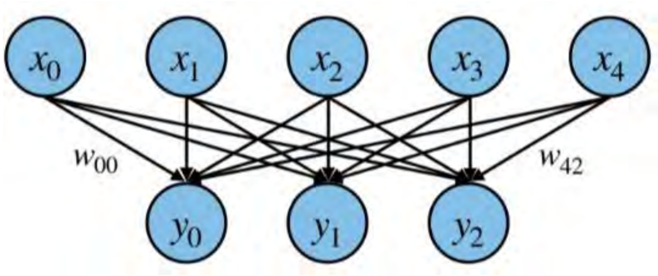


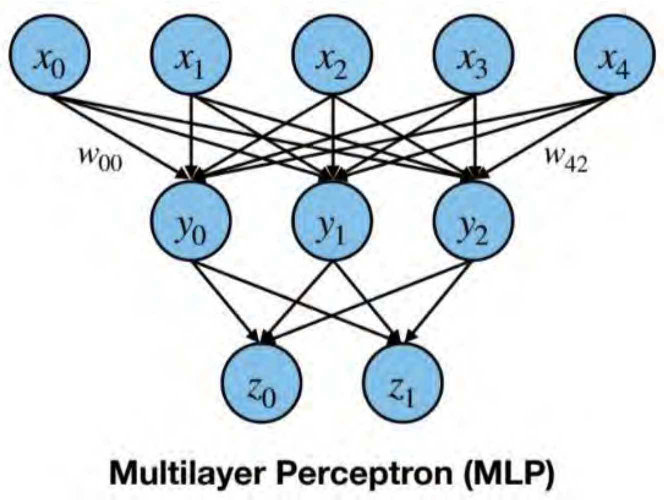
$$y_i = \sum_j w_{ij} x_j + b_i$$



https://hanlab.mit.edu/courses/2023-fall-65940

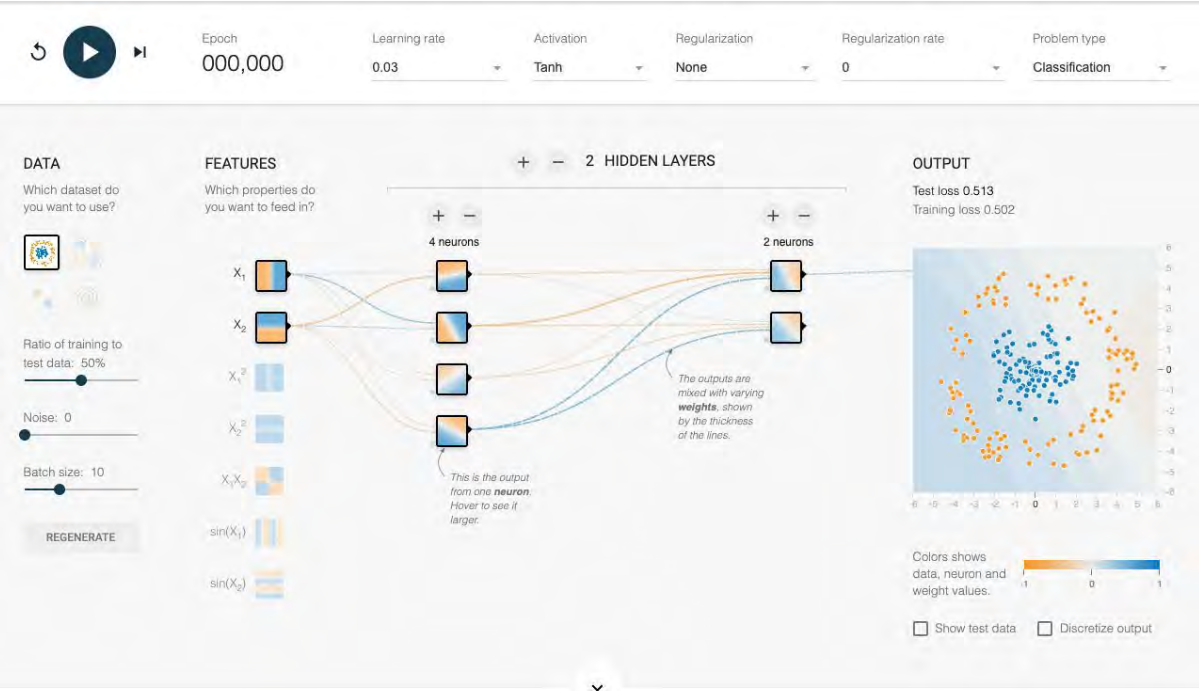
Multilayer Perceptron (MLP)





Demo: Tensorflow Playground

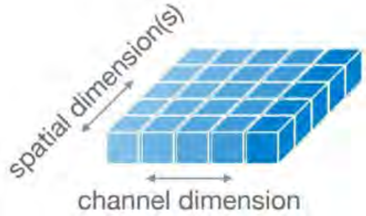
- [A Neural Network Playground]
    - Neuron is one kind of simple ***classifier / filter***



Convolution Layer

- **Shape of Tensors:** **1D Conv**
  - Input Features $\mathbf{X}$: $\cancel{(n, c_i)}$ (n, c_i, w_i)
  - Output Features $\mathbf{Y}$: $\cancel{(n, c_o)}$ (n, c_o, w_o)
  - Weights $\mathbf{W}$: $\cancel{(c_o, c_i)}$
  - Bias $\mathbf{b}$: $(c_o,)$

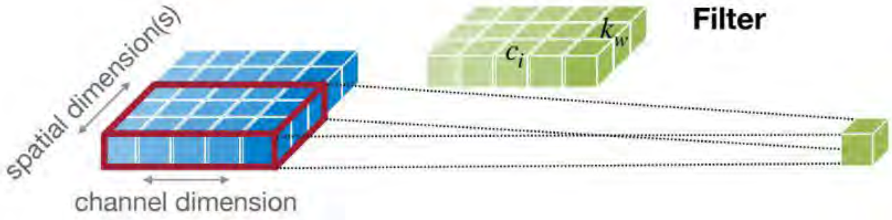
Notations	
n	Batch Size
c_i	Input Channels
c_o	Output Channels
w_i, w_o	Input/Output Width



Convolution Layer

- **Shape of Tensors:** **1D Conv**
  - Input Features $\mathbf{X}$: $\cancel{(n, c_i)}$ (n, c_i, w_i)
  - Output Features $\mathbf{Y}$: $\cancel{(n, c_o)}$ (n, c_o, w_o)
  - Weights $\mathbf{W}$: $\cancel{(c_o, c_i)}$
  - Bias $\mathbf{b}$: $(c_o,)$

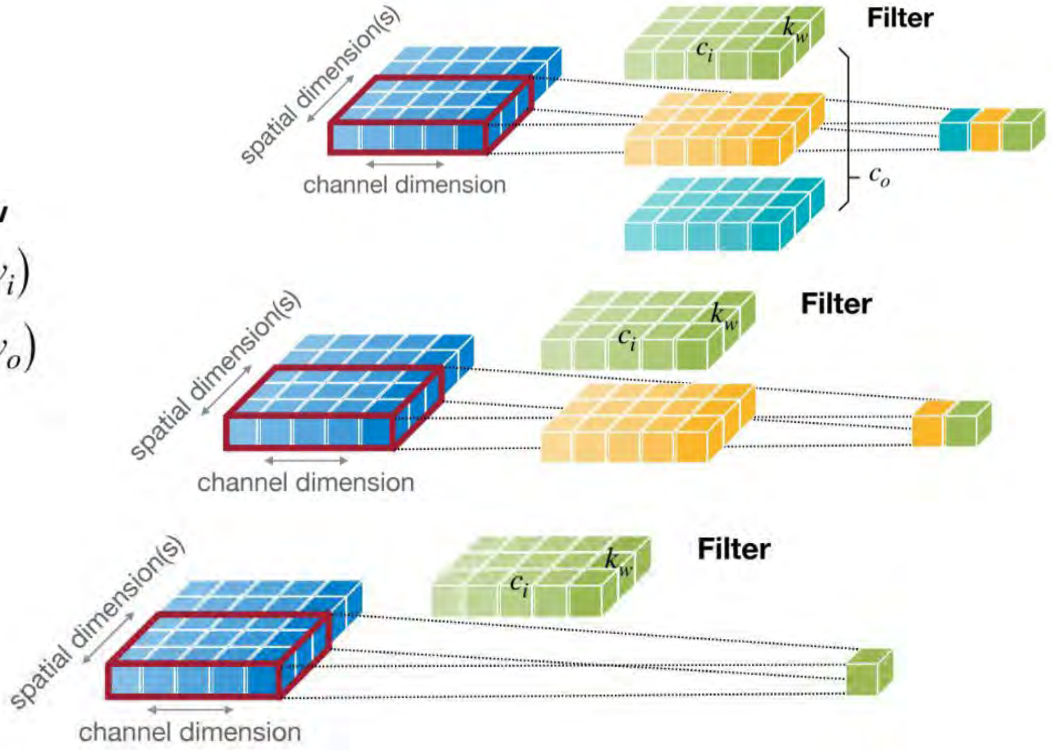
Notations	
n	Batch Size
c_i	Input Channels
c_o	Output Channels
w_i, w_o	Input/Output Width
k_w	Kernel Width



Convolution Layer

- **Shape of Tensors:** **1D Conv**
  - Input Features $\mathbf{X}$: $\cancel{(n, c_i)}$ (n, c_i, w_i)
  - Output Features $\mathbf{Y}$: $\cancel{(n, c_o)}$ (n, c_o, w_o)
  - Weights $\mathbf{W}$: $\cancel{(c_o, c_i)}$
  - Bias $\mathbf{b}$: $(c_o,)$

Notations	
n	Batch Size
c_i	Input Channels
c_o	Output Channels
w_i, w_o	Input/Output Width
k_w	Kernel Width

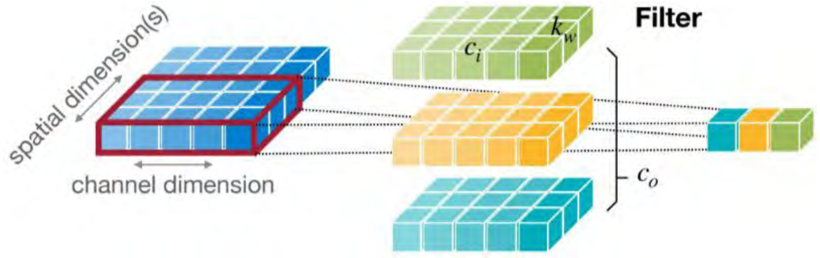


Weight sharing

https://hanlab.mit.edu/courses/2023-fall-65940

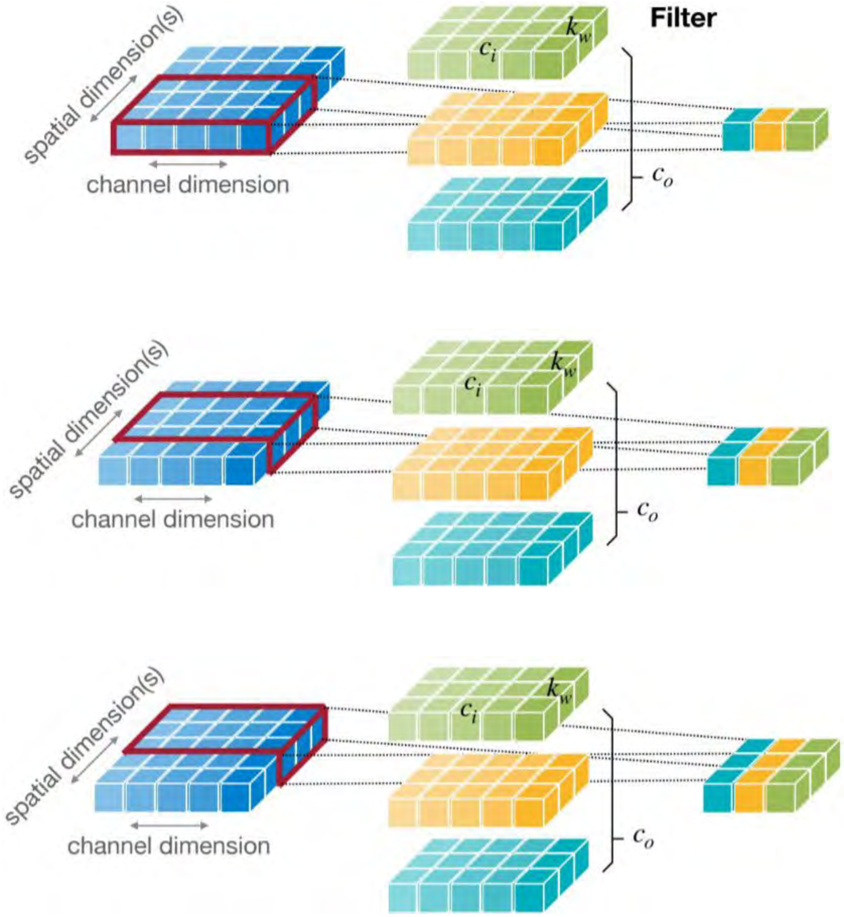
Convolution Layer

- Convolve the ***same weight*** (filter) over ***1D*** spatial dimension



Convolution Layer

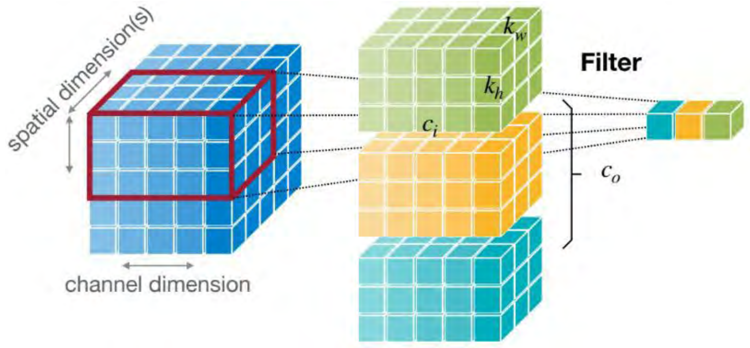
- Convolute the ***same weight*** (filter) over ***1D*** spatial dimension



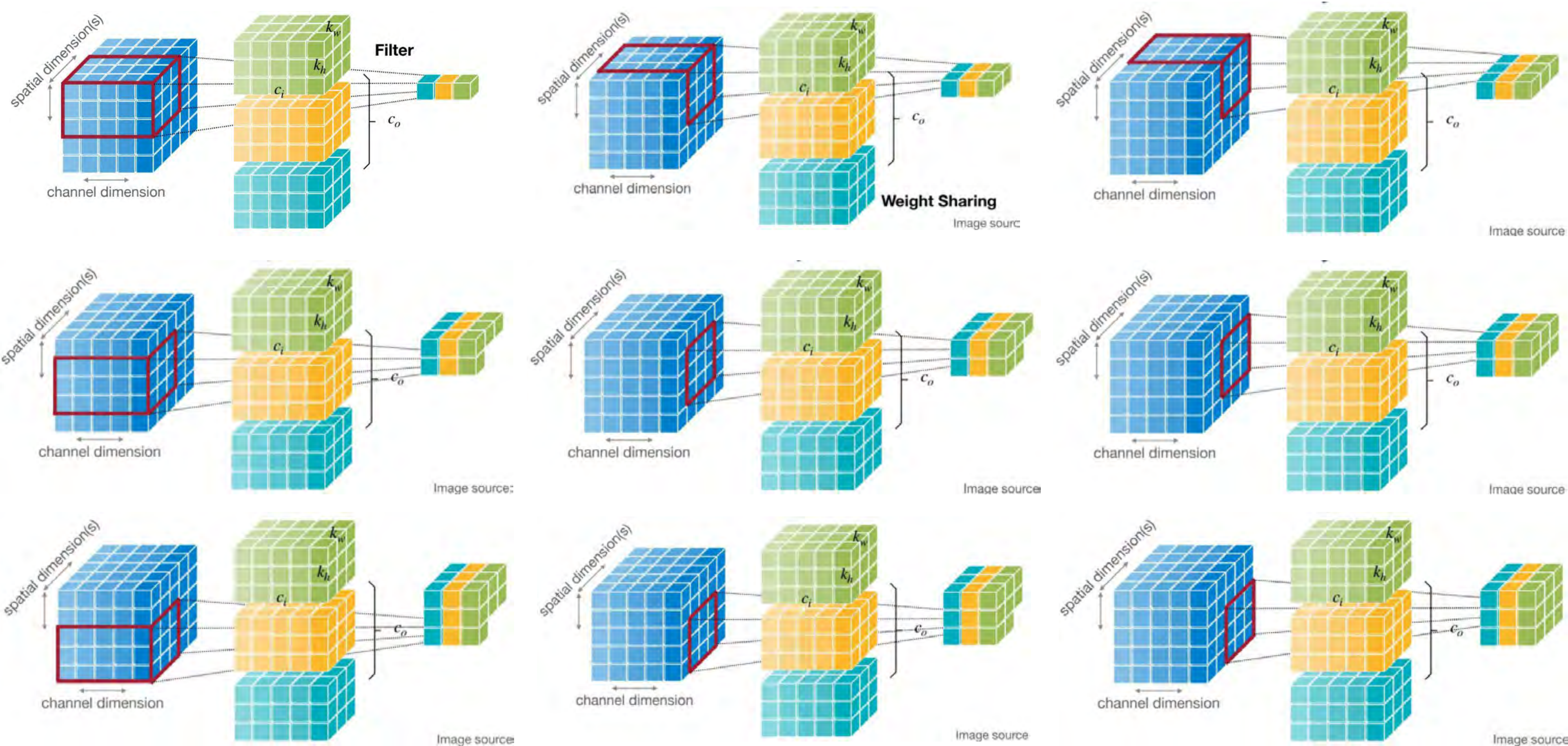
https://hanlab.mit.edu/courses/2023-fall-65940

Convolution Layer

- 2D Convolution
  - Over 2D spatial



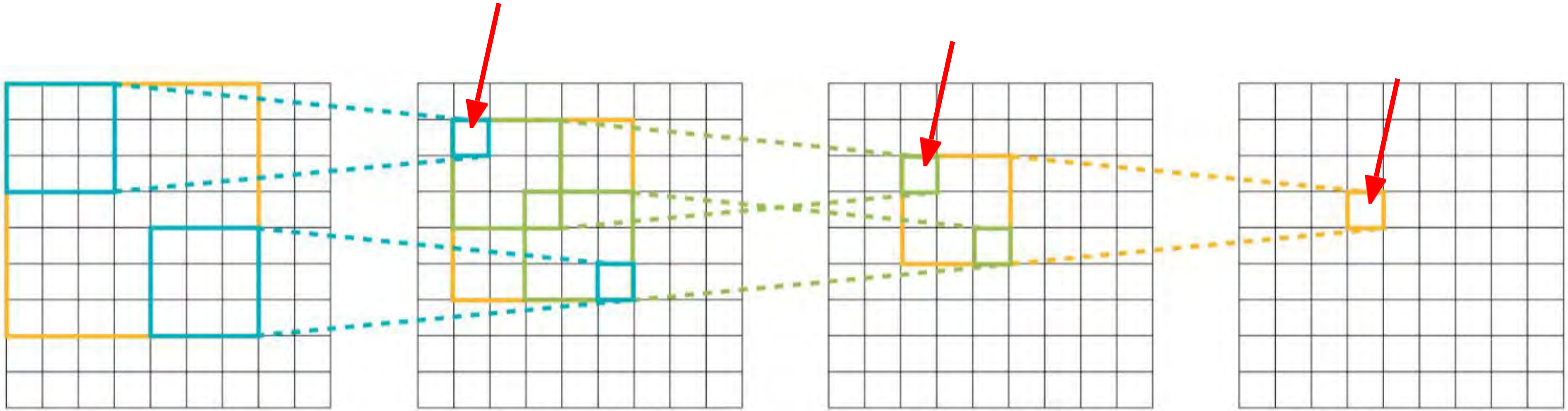
Conv2D



spatial dimension(s)
channel dimension
Filter
Weight Sharing
k_w
k_h
c_i
c_o
Image source

Receptive Field

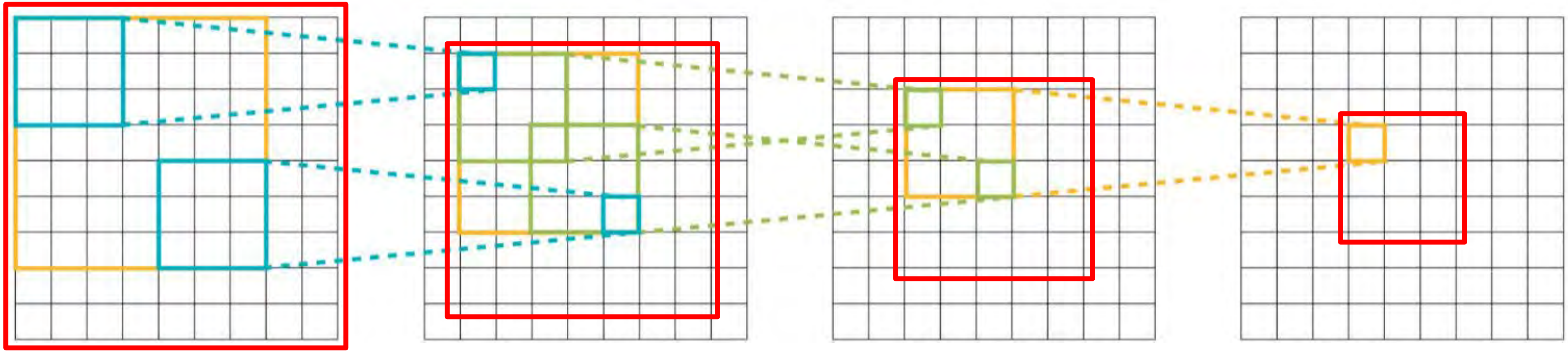
- The dimension of the ***original input*** that impact on one output value



https://hanlab.mit.edu/courses/2023-fall-65940

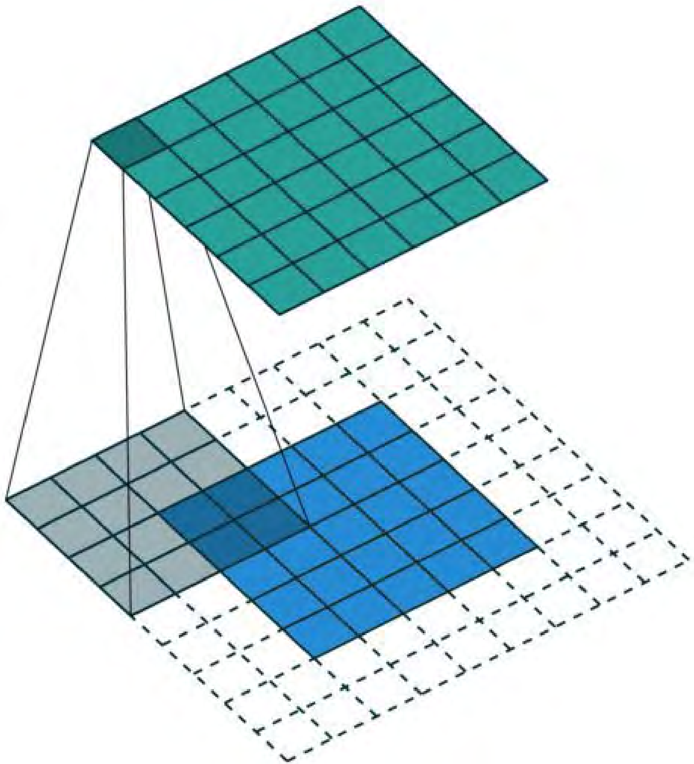
Receptive Field

- The dimension of the ***original input*** that impact on one output value



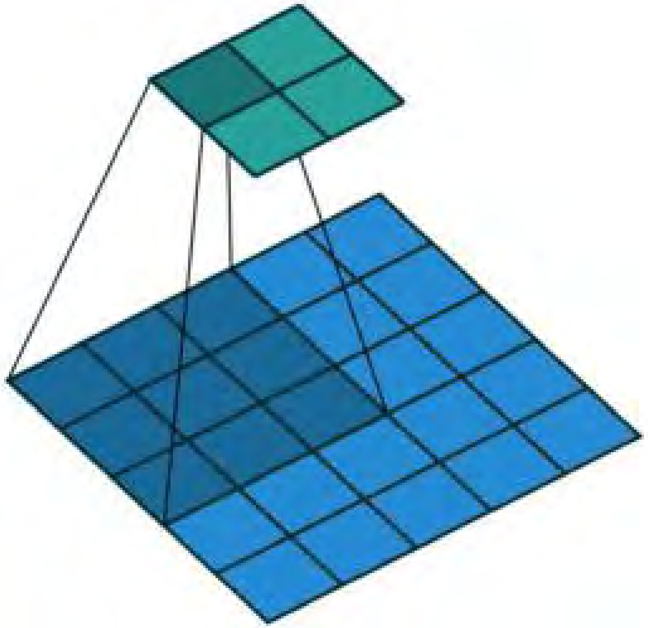
https://hanlab.mit.edu/courses/2023-fall-65940

Padding



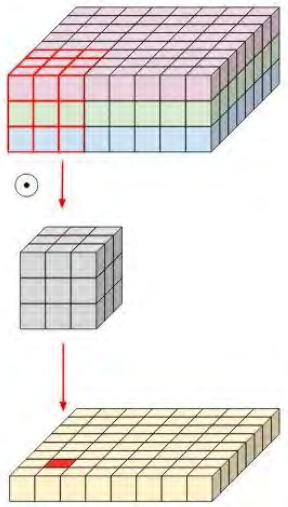
https://github.com/vdumoulin/conv_arithmetic

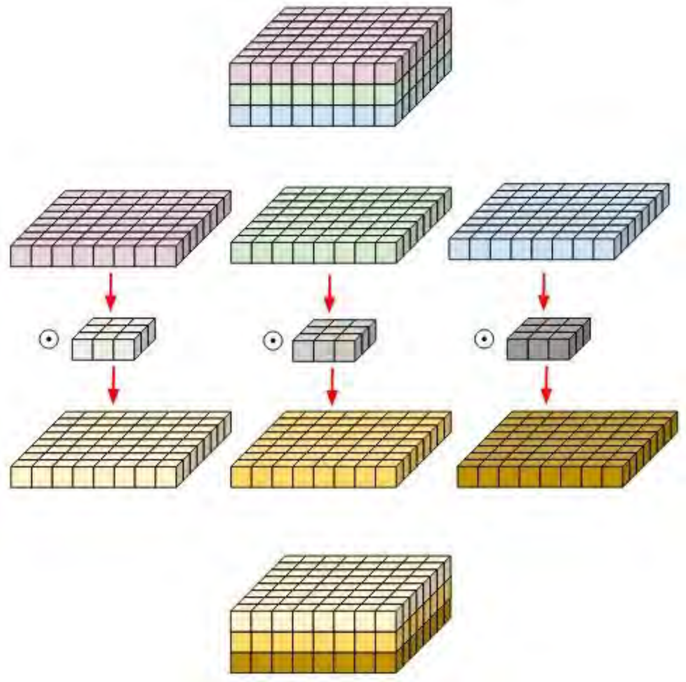
Strides



https://github.com/vdumoulin/conv_arithmetic

Depthwise Convolution Layer

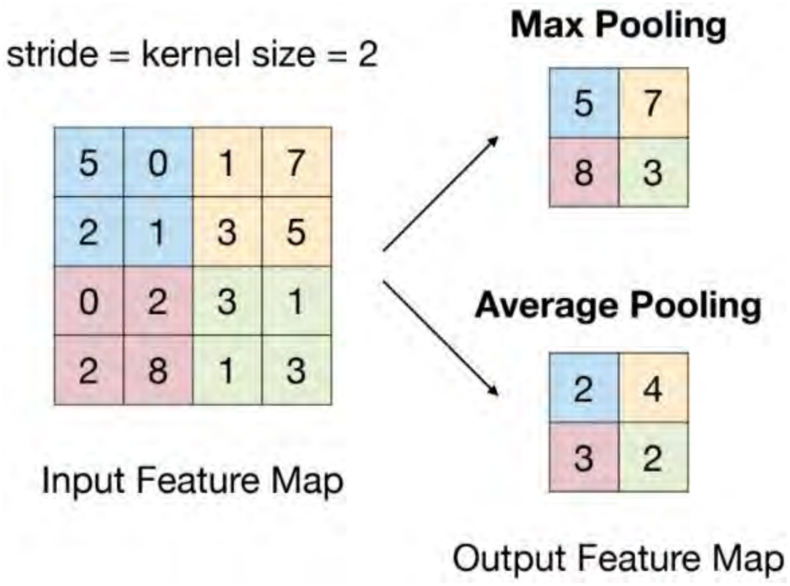




https://medium.com/@zurister/depth-wise-convolution-and-depth-wise-separable-convolution-37346565d4ec

Pooling Layer

- Pool the feature in the receptive field
- Simple math operation
  - No learnable parameters

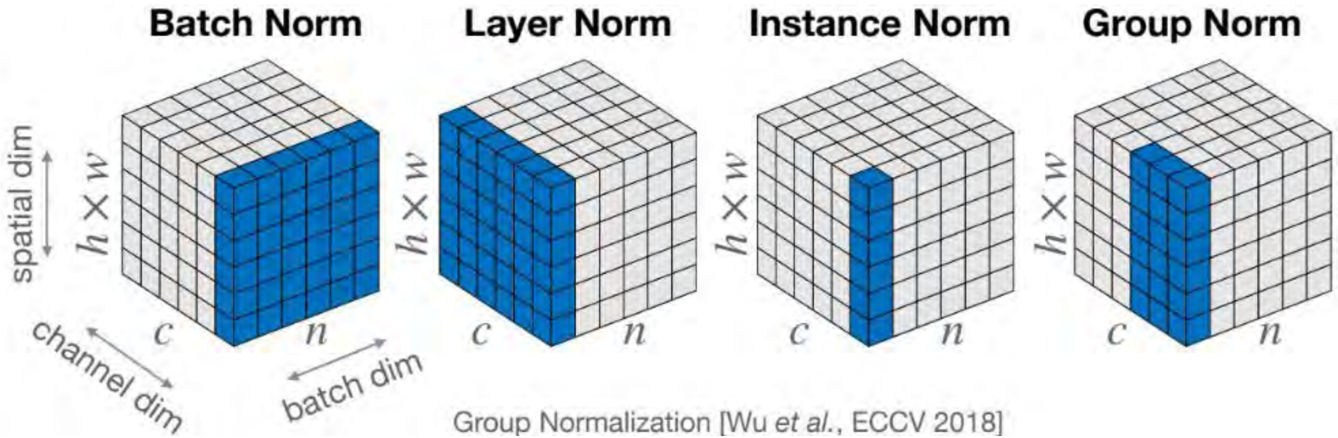


https://hanlab.mit.edu/courses/2023-fall-65940

Normalization Layer

- Normalize a group of features to a **mean** and **standard deviation**

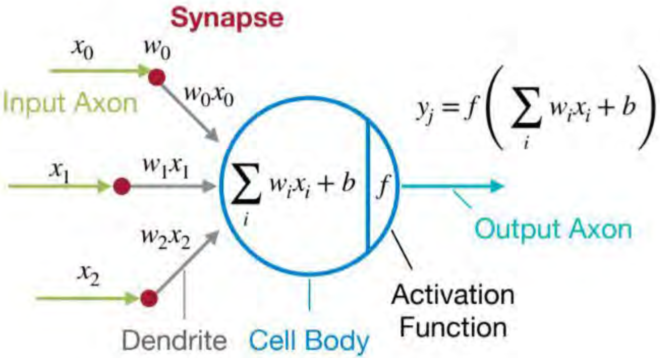
$$\hat{x}_i = \frac{1}{\sigma}\left(x_i - \mu_i\right)$$

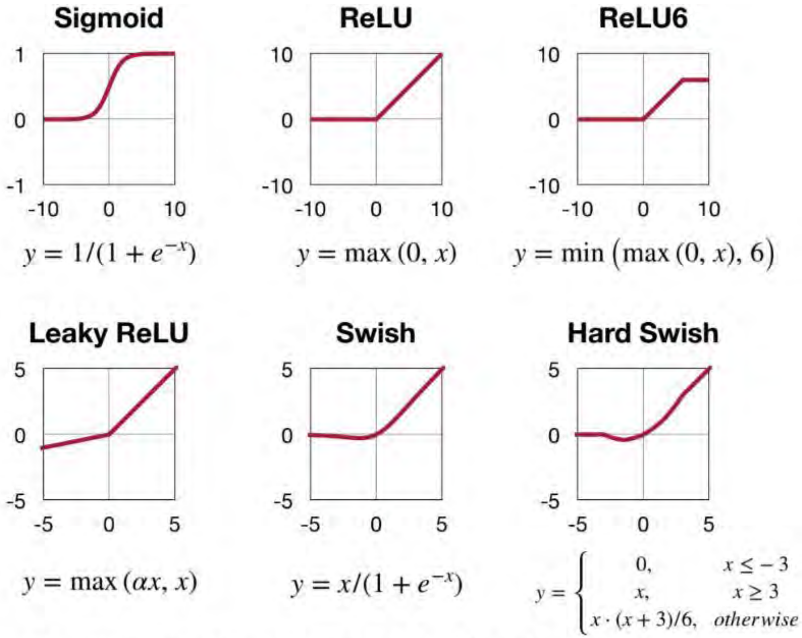


Group Normalization [Wu *et al.*, ECCV 2018]

Activation Function

- Introduce ***non-linearity*** to the neural network





Other Activation Functions: Tanh, GELU, ELU, Mish...

https://hanlab.mit.edu/courses/2023-fall-65940

Agenda

- Neural network: terminology
- Common building block: layer
- Convolution neural network
- Lab 2: object detection

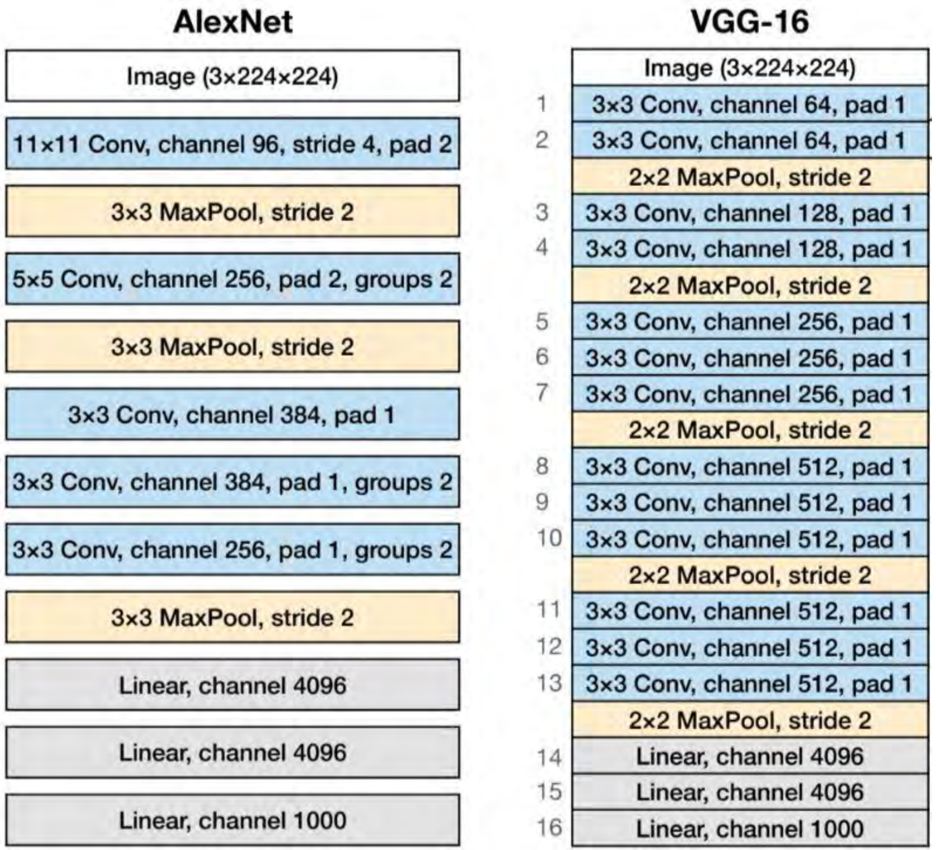
AlexNet

AlexNet	C × H × W	H, W
Image (3×224×224)	3×224×224	
11×11 Conv, channel 96, stride 4, pad 2	96×55×55	$\frac{224+2\times2-11}{4}+1=55$
3×3 MaxPool, stride 2	96×27×27	$\frac{55+0-3}{2}+1=27$
5×5 Conv, channel 256, pad 2, groups 2	256×27×27	$\frac{27+2\times2-5}{1}+1=27$
3×3 MaxPool, stride 2	256×13×13	$\frac{27+0-3}{2}+1=13$
3×3 Conv, channel 384, pad 1	384×13×13	$\frac{13+2\times1-3}{1}+1=13$
3×3 Conv, channel 384, pad 1, groups 2	384×13×13	$\frac{13+2\times1-3}{1}+1=13$
3×3 Conv, channel 256, pad 1, groups 2	256×13×13	$\frac{13+2\times1-3}{1}+1=13$
3×3 MaxPool, stride 2	256×6×6	$\frac{13+0-3}{2}+1=6$
Linear, channel 4096	4096	
Linear, channel 4096	4096	
Linear, channel 1000	1000	

- Trained on ImageNet
- Classify RGB images of 224X224 into 1000 classes

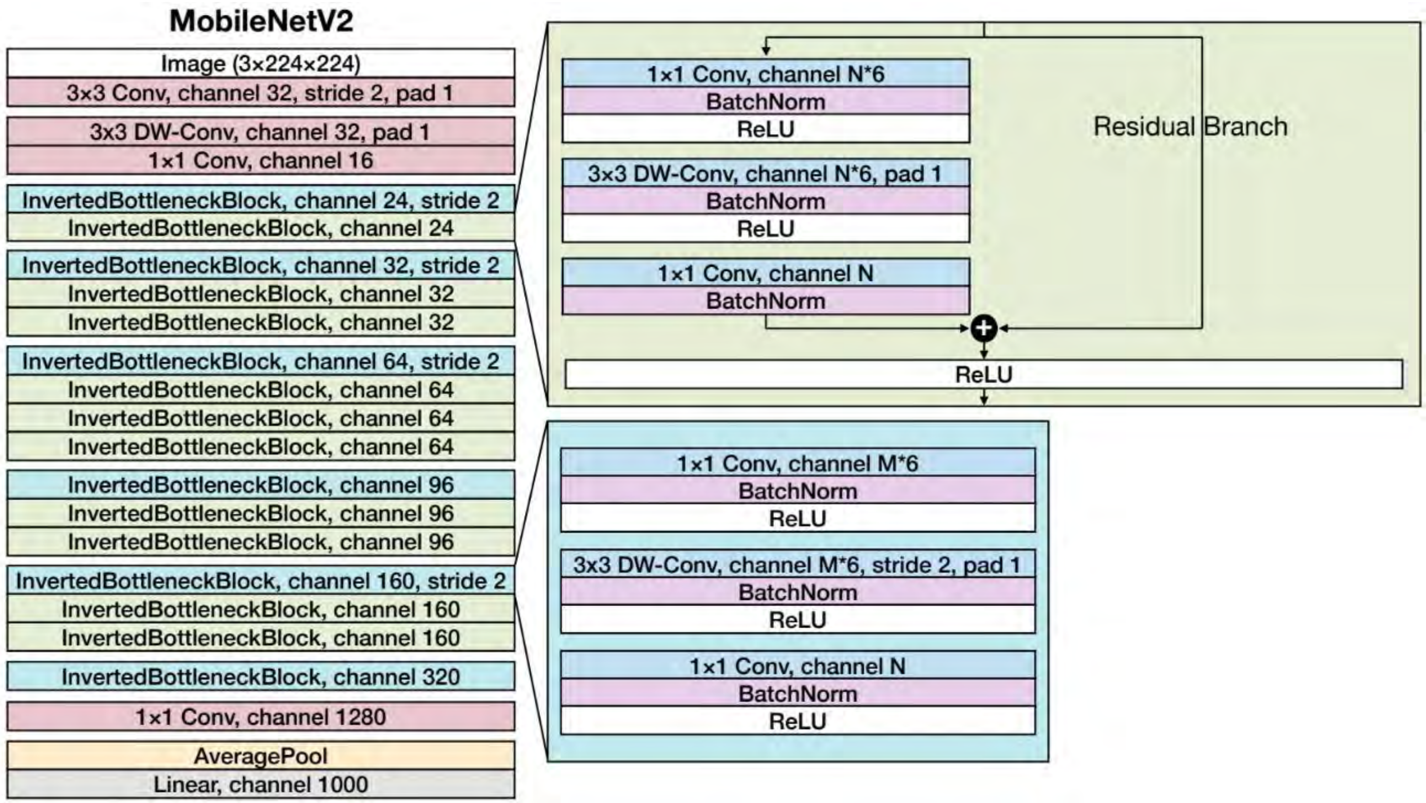
ImageNet Classification with Deep Convolutional Neural Networks [Krizhevskyet al., NeurIPS 2012]

VGG-16



Very Deep Convolutional Networks for Large-Scale Image Recognition
[Simonyanet al., ICLR 2015]

MobileNetV2

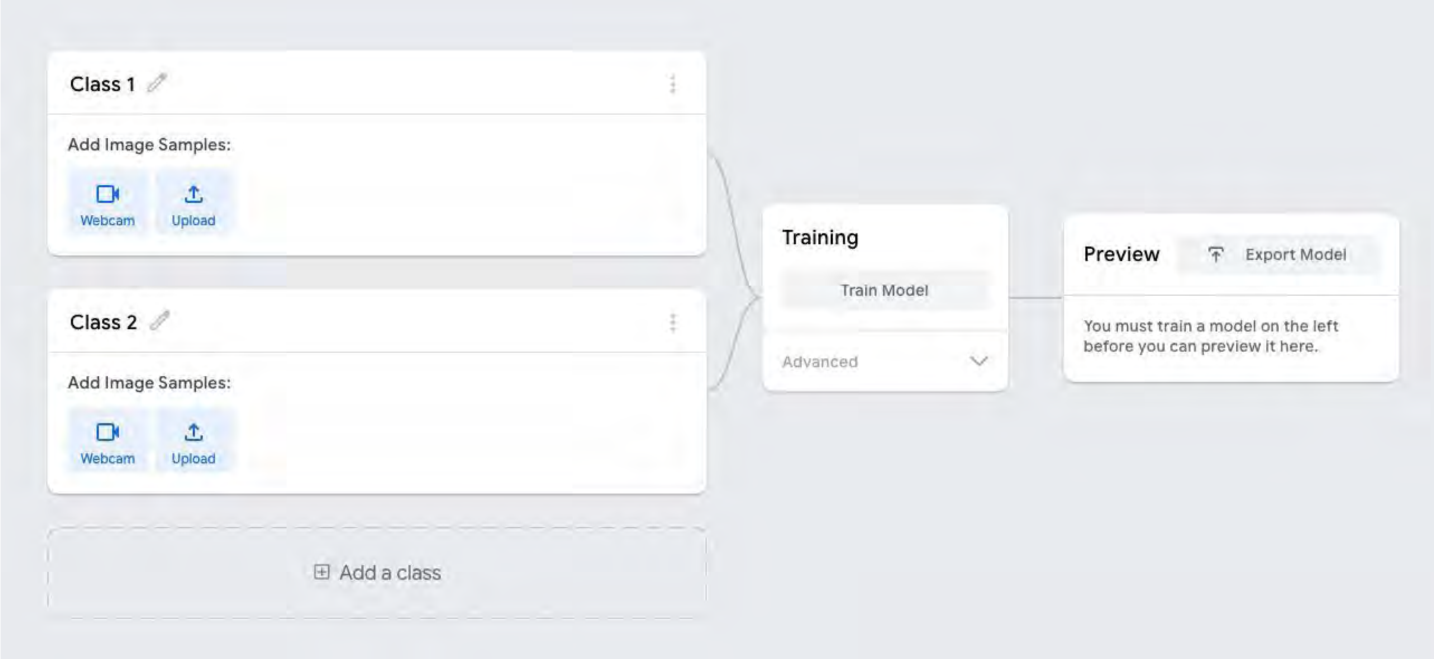


Model Zoo

- Classification
  - models/research/object_detection/g3doc/tf2_classification_zoo.md at master · tensorflow/models · GitHub
- Detection
  - models/research/object_detection/g3doc/tf2_detection_zoo.md at master · tensorflow/models · GitHub

Demo: Teachable Machine

- [Teachable Machine]
  - Auto-optimization, deployment ready:
    - Tensorflow.js, Tensorflow (Keras, SavedModel), Tensorflow lite (quantized, edge TPU)



Demo: Netron.app

- Netron App

Model we trained on Teachable Machine

MobileNet v1 (Downloaded)

Coding your first DNN!

- Example Implementations:
    - [MobileNet v2]()
- Optional:
    - Code a very simple neural net
    - Package it into a docker
    - Run on Jetson Nano
    - Lab 2 bonus point!

```python
def mobilenet_v2(input_shape=None,
  img_input = Input(shape=input_shape)

  first_block_filters = _make_divisible(32 * alpha, 8)
  x = Conv2D(first_block_filters,
             kernel_size=3,
             strides=(2, 2), padding='same',
             use_bias=False, name='Conv1')(img_input)
  x = BatchNormalization(epsilon=1e-3, momentum=0.999, name='bn_Conv1')(x)
  x = Activation(relu6, name='Conv1_relu')(x)

  x = _first_inverted_res_block(x,
                                filters=16,
                                alpha=alpha,
                                stride=1,
                                block_id=0)

  x = _inverted_res_block(x, filters=24, alpha=alpha, stride=2,
                          expansion=6, block_id=1)
  x = _inverted_res_block(x, filters=24, alpha=alpha, stride=1,
                          expansion=6, block_id=2)

  x = _inverted_res_block(x, filters=32, alpha=alpha, stride=2,
                          expansion=6, block_id=3)
  x = _inverted_res_block(x, filters=32, alpha=alpha, stride=1,
                          expansion=6, block_id=4)
  x = _inverted_res_block(x, filters=32, alpha=alpha, stride=1,
                          expansion=6, block_id=5)

  x = _inverted_res_block(x, filters=64, alpha=alpha, stride=2,
                          expansion=6, block_id=6)
  x = _inverted_res_block(x, filters=64, alpha=alpha, stride=1,
                          expansion=6, block_id=7)
  x = _inverted_res_block(x, filters=64, alpha=alpha, stride=1,
                          expansion=6, block_id=8)
  x = _inverted_res_block(x, filters=64, alpha=alpha, stride=1,
                          expansion=6, block_id=9)

  x = _inverted_res_block(x, filters=96, alpha=alpha, stride=1,
                          expansion=6, block_id=10)
  x = _inverted_res_block(x, filters=96, alpha=alpha, stride=1,
                          expansion=6, block_id=11)
  x = _inverted_res_block(x, filters=96, alpha=alpha, stride=1,
                          expansion=6, block_id=12)

  x = _inverted_res_block(x, filters=160, alpha=alpha, stride=2,
                          expansion=6, block_id=13)
  x = _inverted_res_block(x, filters=160, alpha=alpha, stride=1,
                          expansion=6, block_id=14)
  x = _inverted_res_block(x, filters=160, alpha=alpha, stride=1,
                          expansion=6, block_id=15)

  x = _inverted_res_block(x, filters=320, alpha=alpha, stride=1,
                          expansion=6, block_id=16)
```

```python
def _inverted_res_block(inputs, expansion, stride, alpha, filters, block_id):
  """Build an inverted res block."""
  in_channels = int(inputs.shape[-1])
  pointwise_conv_filters = int(filters * alpha)
  pointwise_filters = _make_divisible(pointwise_conv_filters, 8)
  # Expand

  x = Conv2D(expansion * in_channels, kernel_size=1, padding='same',
             use_bias=False, activation=None,
             name='mobl%d_conv_expand' % block_id)(inputs)
  x = BatchNormalization(epsilon=1e-3, momentum=0.999,
                         name='bn%d_conv_bn_expand' %
                         block_id)(x)
  x = Activation(relu6, name='conv_%d_relu' % block_id)(x)

  # Depthwise
  x = DepthwiseConv2D(kernel_size=3, strides=stride, activation=None,
                      use_bias=False, padding='same',
                      name='mobl%d_conv_depthwise' % block_id)(x)
  x = BatchNormalization(epsilon=1e-3, momentum=0.999,
                         name='bn%d_conv_depthwise' % block_id)(x)

  x = Activation(relu6, name='conv_dw_%d_relu' % block_id)(x)

  # Project
  x = Conv2D(pointwise_filters,
             kernel_size=1, padding='same', use_bias=False, activation=None,
             name='mobl%d_conv_project' % block_id)(x)
  x = BatchNormalization(epsilon=1e-3, momentum=0.999,
                         name='bn%d_conv_bn_project' % block_id)(x)

  if in_channels == pointwise_filters and stride == 1:
    return Add(name='res_connect_' + str(block_id))([inputs, x])

  return x
```

Summary

- Neural network: terminology
  - Neuron, Synapse, Activation, Weight, etc.
- Common building block: layer
  - FC, Conv, Conv2D, Depthwise Conv
  - Receptive field, padding, strides
  - Pooling, Normalization
- Convolution neural network
  - Alexnet, VGG16, MobileNetv2
  - Tensorflow, Tensorflow Lite

Week	Day	Date	Lecture	Lab Issue Date	Lab Due End of Day	Project Due (End of week)
1	Mon	01/06	Introduction			
	Wed	01/08	Edge Computing and Its Applications	Lab 0: Setup		
2	Mon	01/13	Edge Systems: Architecture			
	Wed	01/15	Edge Systems: Design and Optimization	Lab 1: Profiling Tools for Jetson	Lab 0	
3	Mon	01/20	Holiday	Final Project Description		Exam 1
	Wed	01/22	Edge ML: Basics of ML	Lab 2: Object Recognition	Lab 1	
4	Mon	01/27	Edge ML: Quantization and Pruning			Proposal
	Wed	01/29	Edge Computing Hardware: Architectures	Final Project Consultation		
5	Mon	02/03	Edge Computing Hardware: Special Accelerators	Lab 3: Client-Server Communica	Lab 2	Paper Pres. Slides & Quiz
	Wed	02/05	Edge & Cloud: Middleware			
6	Mon	02/10	Paper Presentation			
	Wed	02/12	Paper Presentation	Lab 4: Connecting to the Cloud	Lab 3	
7	Mon	02/17	Holiday			Deployment
	Wed	02/19	Paper Presentation			
8	Mon	02/24	Ethics, Privacy & Security		Lab 4	
	Wed	02/26	Edge Computing Research			
9	Mon	03/03	Project Presentation			
	Wed	03/05	Project Presentation			
10	Mon	03/10	Project Presentation			Report
	Wed	03/12	Project Presentation			

Next Lecture

- Making ML more efficient on the edge
  - Quantization & pruning