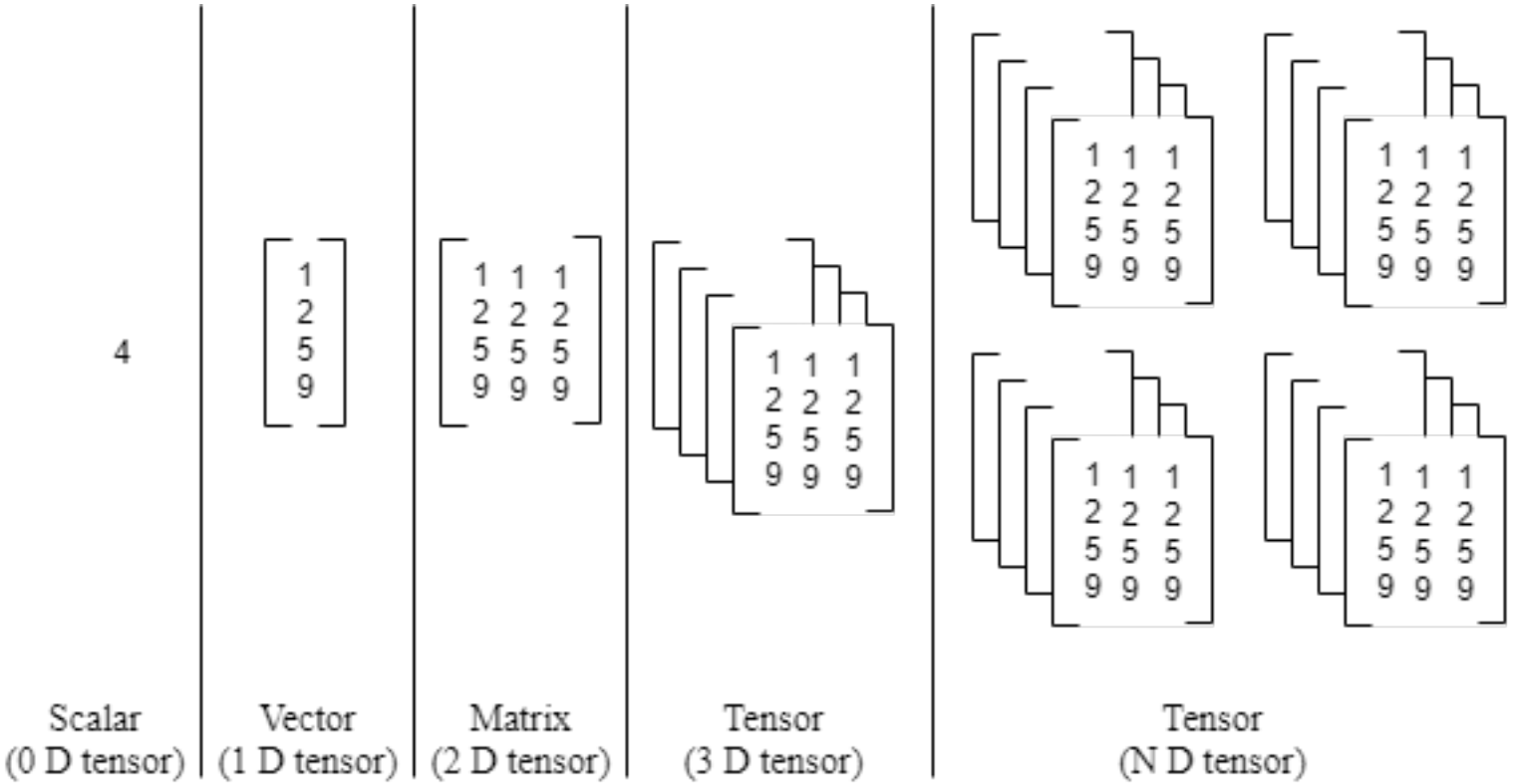
Tensors and Torch Operations

SYDE 599 Deep Learning F23

September 21, 2023

Tensors

- A tensor is a homogenous, n-dimensional array of numbers
  - 0-D: Scalar
  - 1-D: Vector
  - 2-D: Matrix
  - 3-D+: Tensor



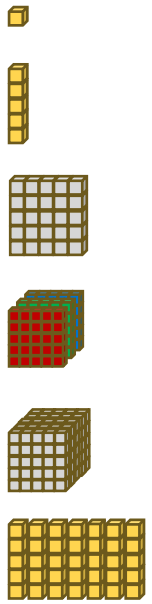
Tensor shape conventions

- In deep learning, we will use the following letters to represent certain dimensions
  - N for Number of examples, B for Batch size
  - L for Length of a sequence
  - W for Width, horizontal spatial dimension of an image
  - H for Height, vertical spatial dimension of an image
  - C for Channels (images or signals), D for feature Dimension (most other data)
- The batch dimension is always first / on the left
- The feature dimension is always last / on the right

Tensor shape conventions

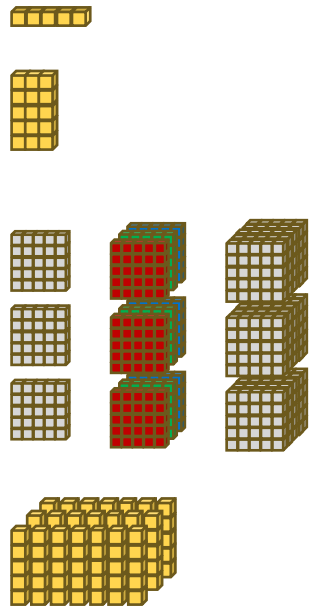
Single data elements

- Scalar: (1,)
- Column vector: (D,) or (1, D)
- Greyscale image: (W, H, 1)
- RGB image: (W, H, 3)
- Arbitrary image: (W, H, C)
- Sequence of vectors: (L, D)



Batches of data elements

- Row vector: (B, 1)
- Feature matrix: (B, D)
- Greyscale images: (B, W, H, 1)
- RGB images: (B, W, H, 3)
- Arbitrary images: (B, W, H, C)
- Sequences of vectors: (B, L, D)



Torch dtypes

- We almost always want our tensor dtype to be torch.float32
  - Float operations default to torch.float32, it's also optimized for efficient compute on GPUs
  - Exception is arrays that are used for indexing, which use torch.long / torch.int64
  - By default, integers are torch.long, so conversion to torch.float32 may be required

Tensor indexing

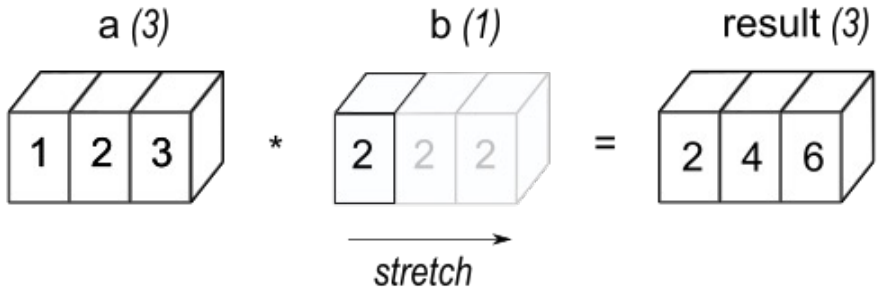
- Tensors can be indexed in the following ways to select elements:
  - Single integers (0, 1, 2, …, D-1) or negative integers (-1, -2, …) to select from end
  - Lists of integers (repeats allowed)
  - Slices, using colon notation (or slice objects)
  - Boolean masks the same size as the dimension(s)
  - Ellipsis to infer dimensions
- Size of resulting tensor after selection depends on method
  - Single integer: Dimension dropped
  - Multiple elements: Dimension kept, size is length of selector
  - Colon or ellipsis: Original dimension size(s) kept

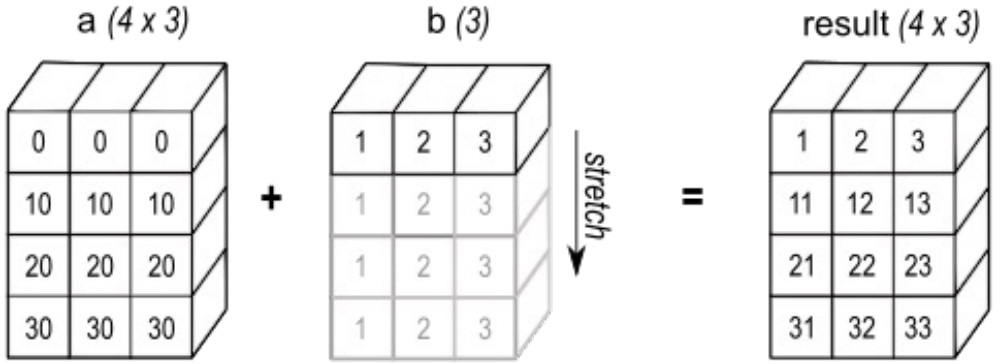
Elementwise operations

- Easiest tensor operations to apply, they modify each element independently
- Examples: torch.exp(), torch.relu(), torch.sigmoid()

Broadcasting

- Typically, we would expect binary operators to work with two arrays when they have the exact same shape, so that the operation can be applied elementwise
- Broadcasting rules assume the shape of the right operand can be transformed to be compatible with the left operand





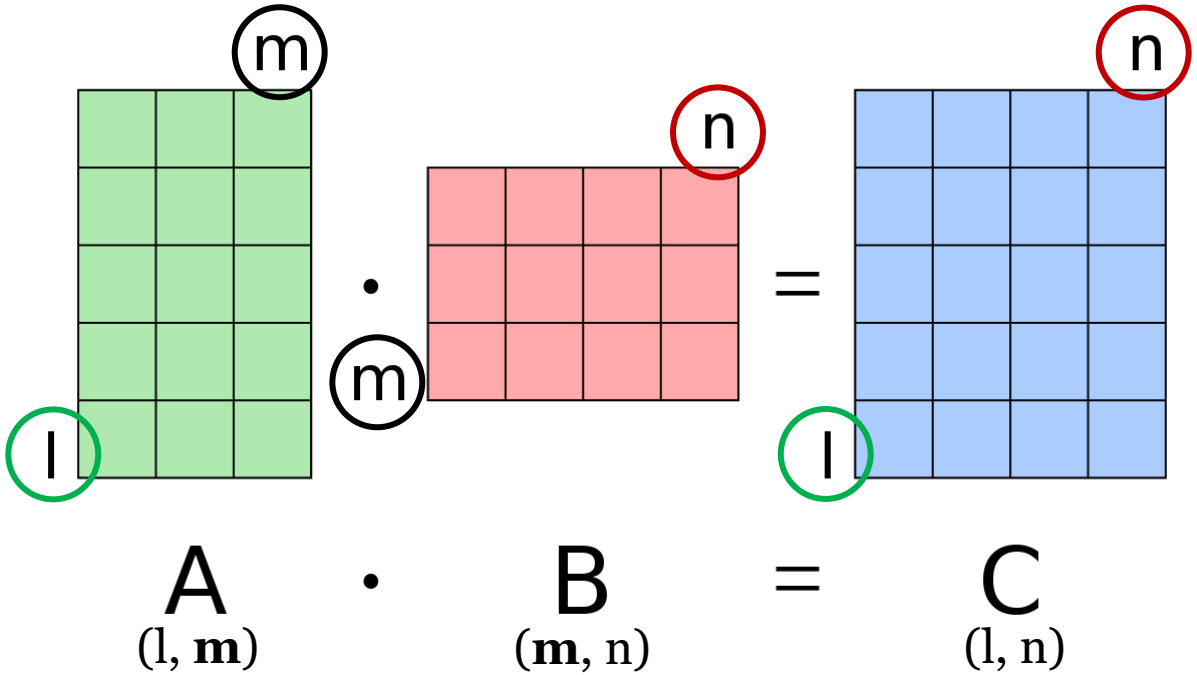
UNIVERSITY OF WATERLOO

Broadcasting

- Broadcasting is compatible when:
  - Trailing (right-most) dimensions match left operand, or
  - A dimension has size 1
- Typical operations:
  - Addition: Adding a scalar to a tensor, adding a bias vector to each row
  - Multiplication: Multiplication by a scalar, scaling each column by its standard deviation
  - Exponentiation: Raise a tensor to power of a scalar, e.g. squaring
  - Comparison operations: Evaluate if all elements are less than a scalar, evaluate if elements equal zero

Matrix multiplication

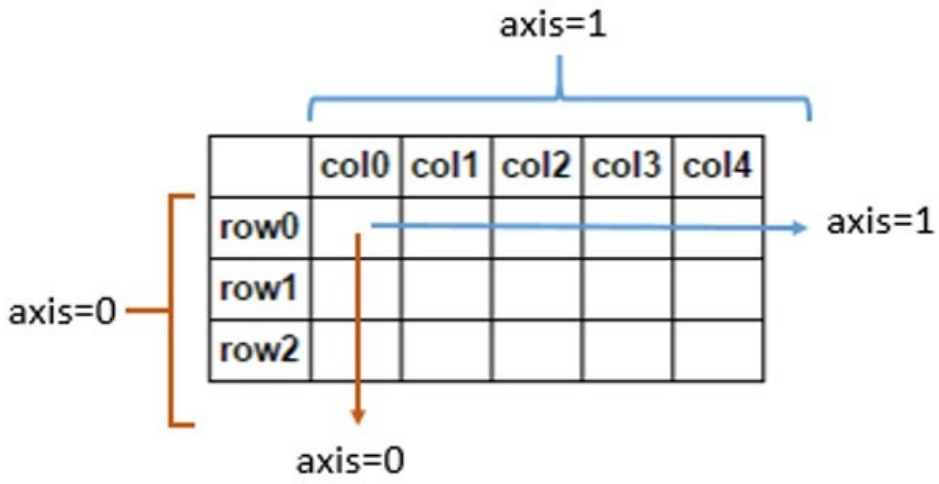
- Also known as dot product or inner product
- Left operand (A) is at least 2-D tensor, right operand (B) is a 2-D tensor (matrix)
- Matmul is broadcastable if ndim(A) > 2
  - (B, H, W, C) @ (C, D) = (B, H, W, D)
- Matmul is a linear transformation on A from m-dimensional vector space to n-dimensional vector space



- Last dimension of A must be same size as second-last dimension of B
- C maintains same shape of A, but with last dimension the same as last dimension of B

Reduction operations

- Apply a function that reduces along one or more dimensions
- By default, reduces across entire tensor, but can specify dimension with keyword "dim" (note Numpy uses keyword "axis" instead)
- Reduction along a dim will drop that dim from the shape
- Most reduction operators use sum internally
- Examples:
  - torch.sum(), torch.mean(), torch.std()
  - torch.softmax(), loss functions like MSE, BCE, CE
  - torch.min(), torch.max()



Reshape operations

- Alter the shape / layout of the tensor without adding or modifying the elements
- Useful for making shapes compatible with broadcasting, especially if dims are lost after a reduction
- Examples:
  - Tensor.reshape(): General reshape, specify a tuple of the new shape
  - Tensor.squeeze(): Remove singleton dims
  - Tensor.unsqueeze(): Add a singleton dim
  - Tensor.flatten(): Unravel into a vector
  - Tensor.permute(): Permute order of dims
  - Tensor.T: Transpose, swap dims 0 and 1 (matrix only)