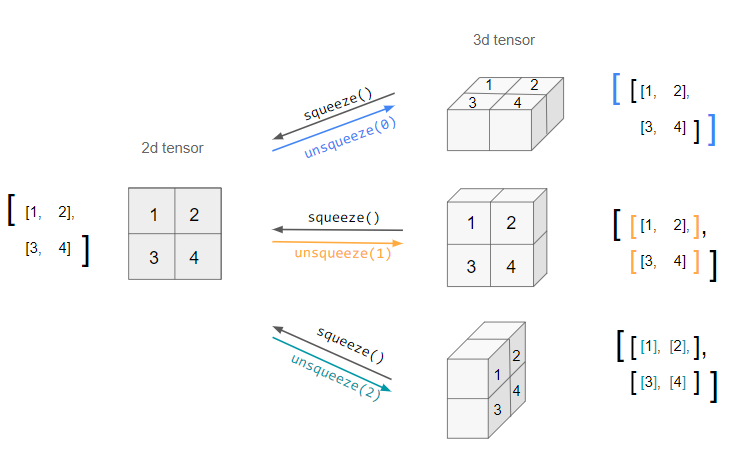
How does unsqueeze help us to solve certain broadcasting problems?

unsqueeze turns an n-dimensionsal tensor into an n+1-dimensional one, by adding an extra dimension of zero depth. However, since it is ambiguous which axis the new dimension should lie across (i.e. in which direction it should be "unsqueezed"), this needs to be specified by the [dim](https://pytorch.org/docs/stable/generated/torch.unsqueeze.html) argument.

Hence the resulting unsqueezed tensors have the same information, but the indices used to access them are different.

Here is a visual representation of what squeeze/unsqueeze do for an effectively 2d matrix, where it is going from a 2d tensor to a 3d one, and hence there are 3 choices for the new dimension's position:

[](https://i.stack.imgur.com/cFE6k.png)

How can we use indexing to do the same operation as unsqueeze?

It squeezes (removes) the size 1 and returns a tensor with all other dimensions of the input tensor. Compute torch. unsqueeze(input, dim). It inserts a new dimension of size 1 at the given dim and returns the tensor.

How do we show the actual contents of the memory used for a tensor?

The easiest[A] way to evaluate the actual value of a Tensor object is to pass it to the Session. run() method, or call Tensor. eval() when you have a default session (i.e. in a with tf.

When adding a vector of size 3 to a matrix of size 3×3, are the elements of the vector added to each row or each column of the matrix? (Be sure to check your answer by running this code in a notebook.)

For matrices or vectors to be added, they must have the same dimensions. Matrices and vectors are added or subtraced element by corresponding element.

Do broadcasting and expand\_as result in increased memory use? Why or why not?

The memory usage should not increase over time, if you are using the same shapes.

If you see an increase in memory usage while your model is training, you might accidentally store some tensors with the attached computation graph, e.g. by doing losses.append(loss) during training.If that’s the case, you should either .detach() the tensors or call .item() on them to get a Python literal.

Implement matmul using Einstein summation.

What does a repeated index letter represent on the lefthand side of einsum?

Evaluates the Einstein summation convention on the operands. Using the Einstein summation convention, many common multi-dimensional, linear algebraic array operations can be represented in a simple fashion.

What are the three rules of Einstein summation notation? Why?

There are essentially three rules of Einstein summation notation, namely:

Repeated indices are implicitly summed over.

Each index can appear at most twice in any term.

Each term must contain identical non-repeated indices.

What are the forward pass and backward pass of a neural network?

A loss function is calculated from the output values. And then "backward pass" refers to process of counting changes in weights (de facto learning), using gradient descent algorithm (or similar). Computation is made from last layer, backward to the first layer. Backward and forward pass makes together one "iteration".

Why do we need to store some of the activations calculated for intermediate layers in the forward pass?

The backprop equations require the activations, and we need to store them be- cause we go backwards through the computation graph.

What is the downside of having activations with a standard deviation too far away from 1?

As a rule of thumb, a CV >= 1 indicates a relatively high variation, while a CV < 1 can be considered low. This means that distributions with a coefficient of variation higher than 1 are considered to be high variance whereas those with a CV lower than 1 are considered to be low-variance. The higher the standard deviation, the riskier the investment. When using standard deviation to measure risk in the stock market, the underlying assumption is that the majority of price activity follows the pattern of a normal distribution.

How can weight initialization help avoid this problem?

The aim of weight initialization is to prevent layer activation outputs from exploding or vanishing during the course of a forward pass through a deep neural network.