Most papers about neural networks use the notation of vectors and matrices from applied linear algebra. This notation works well with vector spaces, but is ill suited for describing neural networks that operate on tensors with many different axes. Consider the following equation (Vaswani et al., 2017):

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V. \tag{1}$$

where Q, K, and V are sequences of query, key, and value vectors packed into matrices. Does the product $QK^\top$ sum over the sequence, or over the query/key features? We would need to know the sizes of Q, K, and V to know that it's taken over the query/key features. Is the softmax taken over the query sequence or the key sequence? The usual notation doesn't even offer a way to answer this question. With multiple attention heads, the notation becomes more complicated and leaves more questions unanswered. With multiple sentences in a minibatch, the notation becomes more complicated still, and most papers wisely leave this detail out. As a preview, in our notation the above equation becomes

$$\text{Attention}(Q^{\mathsf{key}}, K^{\mathsf{seq},\mathsf{key}}, V^{\mathsf{seq}}) = \text{softmax}\left(\frac{Q \underset{\mathsf{key}}{\cdot} K}{\sqrt{|\mathsf{key}|}}\right) \underset{\mathsf{seq}}{\cdot} V \tag{2}$$

making it unambiguous the types of the tensors, and which axis each operation applies to. The same equation works *unchanged* when we let Q be a sequence of vectors in $\mathbb{R}^{\mathsf{key}}$ rather than a single vector and make the values in V be vectors rather than scalar. It also extends to multiple heads and minibatching.

We now explain our notation using a running example. Consider an image that is modeled of as an order three tensor I that maps a triple (x, y, c) where x is an X-coordinate, y is Y-coordinate, and c is a *channel* in $\{\text{`R'}, \text{`G'}, \text{`B'}\}$ into a real number corresponding to the intensity of this channel at this pixel. In our notation we write this as $I \in \mathbb{R}^{\mathsf{width},\mathsf{height},\mathsf{channel}}$. The *type* of I is denoted as $\{\mathsf{width}, \mathsf{height}, \mathsf{channel}\}$ and we will sometimes use superscripts such as $I^{\mathsf{width},\mathsf{height},\mathsf{channel}}$ to remind the reader of it. If x, y, c are known to be indices into the $\mathsf{width}, \mathsf{height}, \mathsf{channel}$ axes respectively, then we can simply write $I_{x,y,c}$ for corresponding coordinate of I. Order here does not matter and $I_{x,y,c} = I_{y,x,c} = I_{c,x,y}$ etc. If x, y, c do not have such types (e.g., if they are simply integers) then we need to *explicitly cast* them into these types by writing $I_{\mathsf{width}[x],\mathsf{height}[y],\mathsf{channel}[c]}$. If we provide fewer indices than the number of axes in I then the result is the restriction of the tensor to these values. For example, $I_{x,y,\mathsf{channel}[\text{`B'}]}$ is the $\{\mathsf{width}, \mathsf{height}\}$ tensor for which every (x, y) coordinate is the intensity of the blue channel.

By specifying axes, we can use notation such as $\underset{mean}{\mathsf{channel}}(I)$ to denote the $\{\mathsf{width}, \mathsf{height}\}$ type tensor where each pixel location (x, y) contains the mean pixel intensity across the $\mathsf{channel}$ axis. In contrast, $\underset{mean}{\mathsf{width},\mathsf{weight}}(I)$ denotes the $\{\mathsf{channel}\}$ type tensor that contains the average RGB values across all pixels. The same logic extends to any operation on tensors. Consider the "90 degree rotation" operation ROT that takes a tensor $A \in \mathbb{R}^{\mathsf{width},\mathsf{height}}$ and maps it to the tensor $B \in \mathbb{R}^{\mathsf{width},\mathsf{height}}$ such that for every (x, y) pair of coordinates, $B_{x,y} = A_{y,x}$. In our notation we write this as follows:

$$ROT(A^{\mathsf{width},\mathsf{height}}) = B\text{s.t.}$$

where

$$B_{\mathsf{height}[x],\mathsf{width}[y]} = A_{x,y} \quad \text{For every } (x, y) \in \mathsf{width}(A) \times \mathsf{height}(A) \tag{3}$$

That is, $\mathsf{width}(A)$ is the set of all indices that index into width axis of A (i.e., the elements of width on which the partial map A is defined), and we consider this as a disjoint set from the analog set $\mathsf{height}(A)$. Hence there is no ambiguity as to which point $A_{x,y}$ corresponds to, and the order does not matter (i.e., $A_{x,y}$ is equal to $A_{y,x}$ and in both cases correspond to the value of A at the point where the width axis is x and the

height axis is y). If we want x to index into the height axis of B instead of into the width axis then we need to explicitly *cast* it as is done in (3).

The *signature* of ROT is that it maps a $\{\mathsf{width}, \mathsf{height}\}$-type tensor into a $\{\mathsf{width}, \mathsf{height}\}$-type tensor. If we feed into ROT a tensor A with an extra axis such as $\mathsf{channel}$ then it is considered a *dangling axis* which means that it is carried through to the output, which will be a tensor B of type $\{\mathsf{width}, \mathsf{height}, \mathsf{batch}\}$, such that for every $c \in \mathsf{channel}(A)$, $B_c = ROT(A_c)$. That is, the operation ROT is applied independently to every restriction of A to the coordinates where the $\mathsf{channel}$ axis equals c. Similarly, if we feed into a tensor A with type $\{\mathsf{width}, \mathsf{height}, \mathsf{channel}, \mathsf{batch}\}$ corresponding to a *batch* of images then the output type will be $\{\mathsf{width}, \mathsf{height}, \mathsf{channel}, \mathsf{batch}\}$, with ROT applied to every $\{\mathsf{width}, \mathsf{height}\}$ restriction independently.

1 Formal definitions

We now describe the formal definitions. An *axis* is a set of the form $\{(\text{name}, x) | x \in X\}$ where name is a string and X is an ordered set. If we don't specify the ordered set X then it is assumed to be the set of natural numbers $\mathbb{N}$. Our convention is that that axes' names are of the form type or $\text{type}^{\text{annotation}}$, with the assumption that axes with names such as layer^{in} and $\text{layer}^{\text{out}}$ correspond to the same type of indices. All our axes will have a unique names of the form $\text{type}^{\text{annotation}}$ (where annotation might be empty). We use $\mathsf{type}^{\text{annotation}}$ to denote the unique axis with name $\text{type}^{\text{annotation}}$. For $x \in X$, we use $\mathsf{type}^{\text{annotation}}[x]$ to denote the corresponding element $(\text{type}^{\text{annotation}}, x)$ in the set $\mathsf{type}^{\text{annotation}}$. we use notation such as ax or $\mathsf{ax}^1, \mathsf{ax}^2, \ldots$ to denote generic axes.

A *tensor type* is a finite set $\mathcal{T} = \{\mathsf{ax}^1, \ldots, \mathsf{ax}^d\}$ of axes. A (real-valued) *tensor* of type $\mathcal{T}$ is a partial map $A : \mathsf{ax}^1 \times \mathsf{ax}^2 \times \cdots \times \mathsf{ax}^d \to \mathbb{R}$, where the set of coordinates $(i_1, \ldots, i_d)$ on which A is defined is a product set $I_1 \times \cdots I_d$ where $I_j \subseteq \mathsf{ax}^j$ for $j = 1 \ldots d$. We write $A(i_1, \ldots, i_d)$ or $A_{i_1,\ldots,i_d}$ for the corresponding coordinate. Since $i_1, \ldots, i_d$ are elements of disjoint sets, we can write the indices or inputs in any order without ambiguity. If we supply a partial assignment $I = \{i_{i_1}, \ldots, i_{j_\ell}\}$ to the axes $\mathsf{ax}^{j_1}, \ldots, \mathsf{ax}^{j_\ell}$ with $j_1 \, ldots, j_\ell$ being distinct indices in $[d]$, then A_I is the tensor of shape $\mathcal{T}' = \mathcal{T} \setminus \{\mathsf{ax}^{j_1}, \ldots, \mathsf{ax}^{j_\ell}\}$ that maps every $d - \ell$ set of indices $\{k_1, \ldots, k_{d-\ell}\}$ which are elements of distinct axes in $\mathcal{T}'$ into $A_{k_1,\ldots,k_{d-\ell},i_{i_1},\ldots,i_{j_\ell}}$. If $\mathcal{T} = \{\mathsf{ax}^1, \ldots, \mathsf{ax}^d\}$ is a type then we let $\mathbb{R}^{\mathsf{ax}^1,\ldots,\mathsf{ax}^d}$ denote the set of all tensors of type $\mathcal{T}$. We denote the set I_j of coordinates of ax^j on which A is defined by $\mathsf{ax}^j(A)$.

An *operator* is a (possibly partial) map that takes as input one or more tensors of a specified type, and outputs a tensor of a specified type. The *signature* of an operator is the types of tensors it takes as input and the type of tensor is output. If we invoke an operator F with k inputs on tensors $A^1, \ldots, A^k$ that contain axes not appearing in the signatures then the result is defined as follows. For every i, we let $\mathcal{T}^i$ be the set of axes that the i-th tensor contains and are missing from the corresponding part of F's signature. For $F(A^1, \ldots, A^k)$ to be defined we need that **(1)** none of those axes appear in the signature of the output, and **(2)** if an axis ax appears in both $\mathcal{T}^i$ and $\mathcal{T}^j$ then it must hold that $\mathsf{ax}(A^i) = \mathsf{ax}(A^j)$. In this case, the output Y of $F(A^1, \ldots, A^k)$ will have, in addition to the axes in its signature, also all the axes in $\mathcal{T}^1 \cup \cdots \cup \mathcal{T}^k$. For every assignment I to these axes, Y_I equals $F(A^1_{\mathcal{T}^1=I}, \ldots, A^k_{\mathcal{T}^k=I})$ where by $A_{\mathcal{T}=I}$ we mean the restriction of A obtained by assigning to every ax in $\mathcal{T}$ the corresponding index in I.

Example: broadcasting. Consider the operator $ADD(x, y) = x + y$ whose signature is simply $ADD : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$. If A is a $\{\mathsf{row}, \mathsf{col}\}$ tensor and B is a col tensor, then $ADD(A, B)$ will be defined if $\mathsf{col}(A) = \mathsf{col}(B)$ and in this case have type $\{\mathsf{row}, \mathsf{col}\}$. For every $i \in \mathsf{row}(A)$ and $j \in \mathsf{col}(A) = \mathsf{col}(B)$, $ADD(A, B)_{i,j} = A_{i,j} + B_j$.

Preconditions and postconditions. An operator does not have to be defined on all tensors of a given type, and can have restrictions on the inputs, such as requiring certain inputs to have the same support, requiring the support to be a multiple of a certain number, etc. Since these restrictions can vary greatly in applications, these are not part of the signature of the operator, but rather can be stated as pre-conditions. Similarly, it is useful to state post-conditions on the various axes of tensors.

Dropping trivial dimensions convention. If a tensor A contains an axis ax with $|\mathsf{ax}(A)| = 1$, we say that ax is *trivial* in A. We make the convention that we do not distinguish between tensors that contain an axis trivially and ones that do not contain it at all. Hence we can add a trivial axis to make a tensor fit the signature of an operation, and we can drop one when we wish to align tensors. This means that for example we do not distinguish between a tensor A of type $\{\mathsf{row}, \mathsf{col}\}$ where $|\mathsf{row}(A)| = 1$ and a tensor of type $\{\mathsf{col}\}$: both are column vectors.

Casting. If A is a tensor containing axis ax then $A_{\mathsf{ax}\to\mathsf{ax}'}$ is the tensor replacing ax with ax'. This transformation also changes the corresponding indices according to the semantic interpretation of the indices. So for example, $\mathsf{channel} \to \mathsf{layer}$ might map (‘R’, ‘G’, ‘B’) to $(1, 2, 3)$. We can also cast multiple axes into one or vice versa, such as the flattening operations $A_{\{\mathsf{width},\mathsf{height},\mathsf{channel}\}\to\mathsf{layer}}$.

Slicing. If ax is an ax of the form $\{(ax, x) | x \in X\}$ and $Y \subseteq X$ then $\mathsf{ax}[Y]$ is the subset of ax defined as $\{(ax, x) | x \in Y\}$. Since X is well ordered, for every a, b we can use $\mathsf{ax}[a..b]$, $\mathsf{ax}[a..]$ or $\mathsf{ax}[..b]$ to denote the restrictions of ax to the sets Y of inputs between a and b, at least a, or at most b respectively.[1]

References

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008. Curran Associates, Inc.

[1]For concreteness, we use “math style” one-based indexing and inclusive intervals here, and assume the natural numbers $\mathbb{N}$ start with 1, and so for an axis ax supported on $\mathbb{N}$, $\mathsf{ax}[..n] = \mathsf{ax}[\{1, \ldots, n\}]$.