Final Project: Tensors and Applications

MATH 5110: Applied Linear Algebra and Matrix Analysis

Group members: Ansa Brew-Smith, Neil Patel

Contents

1 Introduction

As we have seen in class, linear algebra is behind many applications of mathematics that have revolutionized modern technology. Multilinear algebra, which is the study of tensors or multilinear maps, is an extension and generalization of linear algebra that has also found applications in diverse fields such as physics, engineering, and computer science. In this paper, we give a basic theoretical introduction to this field and present some of its practical applications.

2 Theoretical Treatment

There are several equivalent ways to define tensors. We begin with a semi-abstract definition in terms of concepts we have studied in this class, examine the computational aspects which arise from this such as basis and coordinate expressions, then follow up with a more abstract definition of tensor product and finally present the universal property which is used in the modern definition. We start with a basic definition:

Definition 2.1: Let V be a vector space over $\mathbb{R}$ of dimension n (most definitions should work for any field but this is the most used in applications). The dual space of V, denoted V^* or $\text{Hom}(V, \mathbb{R})$ [1] is the space of all linear functions $f : V \to \mathbb{R}$.

This is a vector space, which can be verified by checking the vector space axioms. Moreover, we can construct a basis by viewing the coordinate functions as elements of V^*. Here is a concrete statement of this:

We know that given a basis $\{e_1, e_2, \ldots, e_n\}$ for V, any vector $v \in V$ can be represented as

[1] This notation stands for "homomorphism" since this is the space of all homomorphisms (in the category of vector spaces) from V to $\mathbb{R}$.

$$v = \sum_{i=1}^{n} c_i e_i = c_1 e_1 + c_2 e_2 + \cdots + c_n e_n$$

Define the coordinate functions e^i on a vector v as the functions that return the coefficient of e_i in the expansion of v above:

$$e^i(v) = c_i$$

Clearly, these functions belong to V^* since they go from V to $\mathbb{R}$. Also, from our knowledge of linear transformations, all elements of V^* can be expressed as

$$f(v) = a_1 c_1 + a_2 c_2 + \cdots + a_n c_n$$

Notice that this is a linear combination of the coordinate functions defined above, and since every element of V^* can be expressed this way, these functions form a basis for the dual space. Thus, the dual space V^* has the same dimension as the original vector space V.

Since V^* has the same dimension as V, we know they must be isomorphic, but it turns out that there is no obvious natural isomorphism between them. However, we can easily identify V with $(V^*)^*$ by viewing vectors as functions on elements of the dual space, i.e. for some fixed vector $v \in V$ and we define its function in the double dual space by

$$f_v(w) := w(v)$$

where $w \in V^*$ is a linear function from V to $\mathbb{R}$. This provides a "canonical" isomorphism between V and $(V^*)^*$, since there is a very natural bijection between vectors in V and linear functions on the dual space as given above. Now, we give a basic definition of tensors that we will later generalize:

Definition 2.2: A multilinear function or form (or tensor) over a vector space V is a function $f : V \times V \times \cdots \times V \to \mathbb{R}$ that is linear in each argument; i.e. if all but one of the input vectors are fixed, then it satisfies the definition of linearity for the remaining argument.

Note: we can construct tensors that have inputs in different vector spaces using the tensor product, which is what we will do later in this text. However, many applications including the ones given below rely on tensors with arguments in either a vector space V, its dual space V^*, or both. These are often said to be of type or degree (k, l) where k is the number of arguments in the dual space and l the number of arguments in V. The space of all tensors of a particular degree over a vector field form a vector space, which can be verified by checking the vector space axioms. We give several examples:

Example 2.3: The space of tensors of type $(0, 1)$ over a vector space V can be identified with the dual space V^* since they are maps from V to $\mathbb{R}$ that are linear in their single argument.

Example 2.4: The space of tensors of type $(0, 2)$ over a vector space V can be identified with bilinear forms on V, which we have seen can be represented by a matrix. For example, take $V = \mathbb{R}^2$ with the standard basis and define the multilinear map $f(v, w) = 1v_1w_1 + 2v_1w_2 + 3v_2w_1 + 4v_2w_2$ in terms of the vector components in this basis. We can represent this map by a matrix

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

where the evaluation on two vectors v and w is given by

$$v^T A w = \begin{bmatrix} v_1 & v_2 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = 1v_1w_1 + 2v_1w_2 + 3v_2w_1 + 4v_2w_2$$

Example 2.5: The space of tensors of type $(1, 1)$ over a vector space V can be identified with linear operators from V to V through the following isomorphism.

Take any linear operator $T : V \to V$. Then, we can construct a tensor A of type $(1, 1)$ by requiring that

$$A(v, v') = v'(T(v))$$

for all $v \in V$ and $v' \in V^*$. Although we will not do so here, it is simple to show that this construction defines a unique tensor and that the same is true in reverse, making this relationship an isomorphism.

Remark 2.6: Notice that both bilinear forms and linear operators over a vector space can be represented by matrices, but there is a crucial difference between their representations. Their matrices transform differently under a change of basis: for instance, a certain bilinear form and linear operator may have the same matrix in one basis but not another. This is one of the primary reasons that tensors are so important in applications, since they represent maps between vector spaces that are independent of basis but can change differently depending on what they represent. These transformation rules are so important that they are often used in the definition of a tensor in the context of physics, where tensors are used to represent universal quantities such as the electromagnetic field (see below).

We now define the tensor product of vectors and vector spaces, which generalizes the ideas presented above.

Definition 2.7: We start with the free vector space Z defined by two vector spaces V and W (the field does not matter but we use $\mathbb{R}$ for simplicity and applicability). This is the vector space that has as its basis elements of $V \times W$, in other words ordered pairs of the form (v, w) for $v \in V$ and $w \in W$. Then, any element of Z can be expressed as a formal finite sum

$$\sum_i c_i(v_i, w_i)$$

for coefficients $c_i \in \mathbb{R}$ and $v_i \in V$, $w_i \in W$. This is a purely formal definition and is not particularly meaningful yet, as there are no relations between the elements of this vector space and the basis has infinitely many elements.

To remedy this, we can impose relations on this vector space to produce a quotient space that has desirable properties. As the name suggests, we want the tensor product of two vector spaces to act like ordinary multiplication in some sense. The tensor product will be represented by the $\otimes$ symbol, and we differentiate between $v \otimes w$, which represents the tensor formed by multiplying tensors from V and W, and $V \otimes W$, the vector space where these tensors live. To act like multiplication, the elements of our tensor product space should satisfy these identities:

$$\begin{gathered}(v_1 + v_2) \otimes w = v_1 \otimes w + v_2 \otimes w \\ v \otimes (w_1 + w_2) = v \otimes w_1 + v \otimes w_2 \\ c(v \otimes w) = (cv) \otimes w = v \otimes (cw)\end{gathered}$$

By using the basis element (v, w) in Z to represent what will become $v \otimes w$ in the quotient space, we can write the following relations that represent the identities above:

$$\begin{gathered}((v_1 + v_2), w) - (v_1, w) - (v_2, w) = 0 \\ (v, (w_1 + w_2)) - (v, w_1) - (v, w_2) = 0 \\ c(v, w) - (cv, w) = 0 \\ c(v, w) - (v, cw) = 0\end{gathered}$$

Then, we denote the span of vectors in Z of the forms given by the left side of the equations above as R, which is a linear subspace since it is a span. To make these identities true in the quotient space, we can simply define it as Z/R since the coset R will become the identity (zero vector) in the new space. We then define $V \otimes W = Z/R$ where the ordered pairs (v, w) are identified with the tensor products $v \otimes w$.

Example 2.8: We can now link back to the less abstract definition above. If we view V as the space of linear functions of V^* into $\mathbb{R}$ and W as the space of linear functions of W^* into $\mathbb{R}$, then $V \otimes W$ is the space of multilinear functions $f : V^* \times W^* \to \mathbb{R}$.

Example 2.9: In our new notation, we can identify what we have called the space of tensors of type (k, l) above with a tensor product

$$\underbrace{V \otimes V \otimes \cdots \otimes V}_{k \text{ times}} \otimes \underbrace{V^* \otimes V^* \otimes \cdots \otimes V^*}_{l \text{ times}}$$

Thus, we see that the new definition is more general because a special case of it reproduces the first definition we presented above.

Definition 2.10: The universal property which is satisfied by the tensor product is given as follows. Let V, W, X be vector spaces and $h : V \times W \to X$ a bilinear map. For any vector space X, any map h factors uniquely through a linear map. What this means is that there exists a unique linear map (up to isomorphism) $\tilde{h} : V \otimes W \to X$ such that the diagram below commutes, i.e. $\tilde{h} \circ \varphi = h$.

$$\begin{array}{ccc} V \times W & \xrightarrow{\varphi} & V \otimes W \\ & \searrow^{h} & \downarrow{\scriptstyle \tilde{h}} \\ & & X \end{array}$$

Here, φ is the inclusion map of $V \times W$ in $V \otimes W$ when ordered pairs (v, w) are identified with products $v \otimes w$. [2] This presents the modern way of defining tensor products, which is to require that there is a space that has this universal property for any V, W and then try to find it. Note that this is non-constructive since this universal property does not even prove that such a space exists, only that if it does there is only one (up to isomorphism). However, the definition given above can be proven to satisfy this property, which proves existence. This definition is preferred in modern mathematics since a similar universal property can be used to construct many important spaces from more basic ones.

Definition 2.11: We can also give a more concrete definiton of the tensor product which is useful for computations. Take two multilinear functions (tensors)

$$f : V_1 \times V_2 \times \cdots \times V_k \to \mathbb{R},\ g : V_{k+1} \times V_{k+2} \times \cdots \times V_{k+l} \to \mathbb{R}$$

where V_i are all vector spaces over $\mathbb{R}$ (or another field; the functions would map to this field in the definiton instead of $\mathbb{R}$). Then, the tensor product of f and g is defined as

$$\begin{gathered} f \otimes g : V_1 \times V_2 \times \cdots \times V_{k+l} \to \mathbb{R} \\ (f \otimes g)(v_1, v_2, \ldots, v_{k+l}) := f(v_1, v_2, \ldots, v_k) g(v_{k+1}, v_{k+2}, \ldots, v_{k+l}) \end{gathered}$$

This definition is equivalent to the one above and also satisfies the universal property.

Example 2.12: We have seen that bilinear forms on a vector space V are tensors of type $(0, 2)$. The standard inner product on $\mathbb{R}^n$ is a special case of this, since it is a bilinear form whose matrix representation corresponds to the identity matrix of size n. Denote the inner product tensor by I. Taking once again the coordinate functions e^i as a basis for $(\mathbb{R}^n)^*$, we see that

$$I = \sum_{i=1}^{n} e^i \otimes e^i$$

since the standard inner (or dot) product on $\mathbb{R}^n$ is the sum of the componentwise products of two vectors.

[2] Note that these are not the only elements of $V \otimes W$; these are simply the "pure" tensors which can be represented as the product of a single element of V and W.

3 Applications

3.1 Lorentz Transformation and Faraday Tensor

Classical electromagnetism primarily studies two vector fields, the electric field $\vec{E}$ and the magnetic field $\vec{B}$, yet in special relativity these two quantities combine into a single mathematical object, the antisymmetric rank-2 tensor $F^{\mu\nu}$. In this way the components of $\vec{E}$ and $\vec{B}$ are simply entries of $F^{\mu\nu}$, and a change of inertial observer corresponds to a linear transformation of that tensor. Let O and O' be two inertial reference frames (frames that move at constant velocity) in relative motion with 3-velocity $\vec{v}$ and define

$$\beta = \frac{\|\vec{v}\|}{c}, \quad n_i = \frac{v_i}{\|\vec{v}\|}, \quad \gamma = \frac{1}{\sqrt{1-\beta^2}}.$$

In coordinates $x^\mu = (ct, x^i)$, the Lorentz boost matrix $\Lambda^\mu{}_\nu$ is

$$\Lambda^\mu{}_\nu = \begin{bmatrix} \gamma & -\gamma\,\beta\,n_j \\ -\gamma\,\beta\,n_i & \delta^i{}_j + (\gamma-1)\,n^i n_j \end{bmatrix},$$

which satisfies $\Lambda^\mu{}_\alpha\,\Lambda^\nu{}_\beta\,\eta_{\mu\nu} = \eta_{\alpha\beta}$ with the Minkowski metric $\eta = \mathrm{diag}(+1,-1,-1,-1)$. In the original frame O, the tensor $F^{\mu\nu}$ encodes $\vec{E}$ and $\vec{B}$ by

$$F^{\mu\nu} = -F^{\nu\mu}, \quad F^{0i} = -E^i, \quad F^{ij} = -\varepsilon^{ijk}B_k,$$

so that

$$F^{\mu\nu} = \begin{bmatrix} 0 & -E_x & -E_y & -E_z \\ E_x & 0 & -B_z & B_y \\ E_y & B_z & 0 & -B_x \\ E_z & -B_y & B_x & 0 \end{bmatrix}.$$

Under a boost Λ, one applies the tensor transformation law

$$F'^{\mu\nu} = \Lambda^\mu{}_\alpha\,\Lambda^\nu{}_\beta\,F^{\alpha\beta},$$

which in matrix notation reads $F' = \Lambda\,F\,\Lambda^T$ and preserves the antisymmetry. Reading off the transformed electric and magnetic fields from $F'^{0i} = -E'_i$ and $F'^{ij} = -\varepsilon^{ijk}B'_k$ yields

$$E'_i = \gamma\big(E_i + \varepsilon_{ijk}\,\beta\,n^j B^k\big) - (\gamma-1)\,n_i(n^j E_j), \quad B'_i = \gamma\big(B_i - \varepsilon_{ijk}\,\beta\,n^j E^k\big) - (\gamma-1)\,n_i(n^j B_j).$$

All steps reduce to pure tensor algebra—index contraction, raising and lowering with $\eta_{\mu\nu}$, and matrix multiplication—thus making manifest the linear action of the Lorentz group on the field tensor and the invariance of scalar combinations such as $F_{\mu\nu}F^{\mu\nu}$.

3.2 Tucker Image Decomposition

Tucker Decomposition serves a similar purpose to singular value decomposition, while taking the three color channels (red, green, and blue) into account. Let $\mathcal{X} \in \mathbb{R}^{H\times W\times 3}$ be the third-order tensor whose (i,j,k) entry is the intensity of the kth color channel at pixel (i,j). The Tucker decomposition approximates $\mathcal{X}$ by

$$\mathcal{X} \;\approx\; \mathcal{G} \;\times_1 A^{(1)} \;\times_2 A^{(2)} \;\times_3 A^{(3)},$$

where $\mathcal{G} \in \mathbb{R}^{r_1\times r_2\times r_3}$ is the *core tensor* and each $A^{(n)} \in \mathbb{R}^{I_n\times r_n}$ is a factor matrix. Here $I_1 = H$, $I_2 = W$, $I_3 = 3$, and the triple (r_1, r_2, r_3) are the chosen multilinear ranks controlling the compression. In practice, one obtains $\mathcal{G}$ and $\{A^{(n)}\}$ by solving a sequence of truncated singular value decompositions (HOSVD) or via higher-order orthogonal iteration. The reconstructed tensor is then

$$\widehat{\mathcal{X}} = \mathcal{G} \times_1 A^{(1)} \times_2 A^{(2)} \times_3 A^{(3)},$$

which in array form yields the compressed image. By choosing $r_1 \ll H$, $r_2 \ll W$, and $r_3 = 3$, the storage cost falls from $HW\cdot 3$ entries to

$$r_1 r_2 r_3 \;+\; H\,r_1 \;+\; W\,r_2 \;+\; 3\,r_3,$$

achieving significant data reduction when r_1 and r_2 are small. In our implementation, we load the image into NumPy and TensorLy, form the tensor $\mathcal{X}$, and compute

$$(\mathcal{G},\, \{A^{(n)}\}) = \text{Tucker}(\mathcal{X},\, (r_1, r_2, 3)).$$

Reconstruction follows by $\widehat{\mathcal{X}} = \text{Tucker\_to\_tensor}(\mathcal{G}, \{A^{(n)}\})$, clipping values to $[0, 255]$ and converting back to 8-bit integers. We then compare reconstructions for ranks

$$(r_1, r_2) \in \{(5,5), (10,10), (20,20), (50,50), (100,100), (H, W)\},$$

observing that as (r_1, r_2) increase, the approximation error decreases and visual fidelity improves, while storage grows linearly in r_1 and r_2. This demonstrates how Tucker decomposition provides a principled multilinear extension of low-rank matrix approximation to tensor data.

3.3 Machine Learning (CNN)

Using the Fashion-MNIST dataset, we developed a Convolutional Neural Network model that builds on the Tucker Decomposition and earlier tensor definitions. Each Fashion-MNIST image is a 28×28 grayscale image and can be represented as a 2D tensor in $\mathbb{R}^{28\times 28}$. We trained the model by feeding multiple images into the network at once in a batch. This gives us a 4D tensor of shape:

$$\text{Batch of images: } \mathbb{R}^{N\times C\times H\times W}$$

where N is the number of images in the batch, $C = 1$ is the number of channels, and H, W are the image height and width. Here we use tensors as functions that take in multiple vectors and return a number. In our case, these tensors store grayscale values and are used as inputs to tensor operations like convolution, pooling, and activation in the network.

As noted in Remark 2.6, tensors can transform differently under a change of basis. In CNNs, this is reflected in the translation-equivariant property of convolutional layers allowing us to gather features and make use of spatial identities while preserving the structure of the input under shifts.

Having discuss Tucker decomposition as a method to factorize high-dimensional tensors. This technique returns in CNN compression, where weight tensors are approximated with Tucker formats to reduce model size while retaining key geometric structure. This allows us to save computational space and while building a more efficient models.

We will now compare the the traditional CNN vs a tucker CNN. Both have form where the convo layers are either torch convo layer vs tucker convo layer.

$$\hat{y} = f_{\text{fc2}} \circ \text{ReLU} \circ f_{\text{fc1}} \circ \text{Flatten} \circ \text{MaxPool} \circ \text{ReLU} \circ f_{\text{conv2}} \circ \text{MaxPool} \circ \text{ReLU} \circ f_{\text{conv1}}(x)$$

	A	B	C	D	E	F	G	H	I	J
1	train	Model 1	Tucker					test	Model 1	Tucker
2	accuracy	0.9808	0.9457						0.9143	0.9039
3	speed	11.905688047409058	11.51899242401123						2.0522408485412598	1.8600788116455078
4										
5										
6										
7										
8										
9										
10	model size	421,642	410,699							
11	training time	221.14s	209.80s							
12	iterations	9,380	9,380							

Figure 1: Performance Comparison of CNN vs Tucker

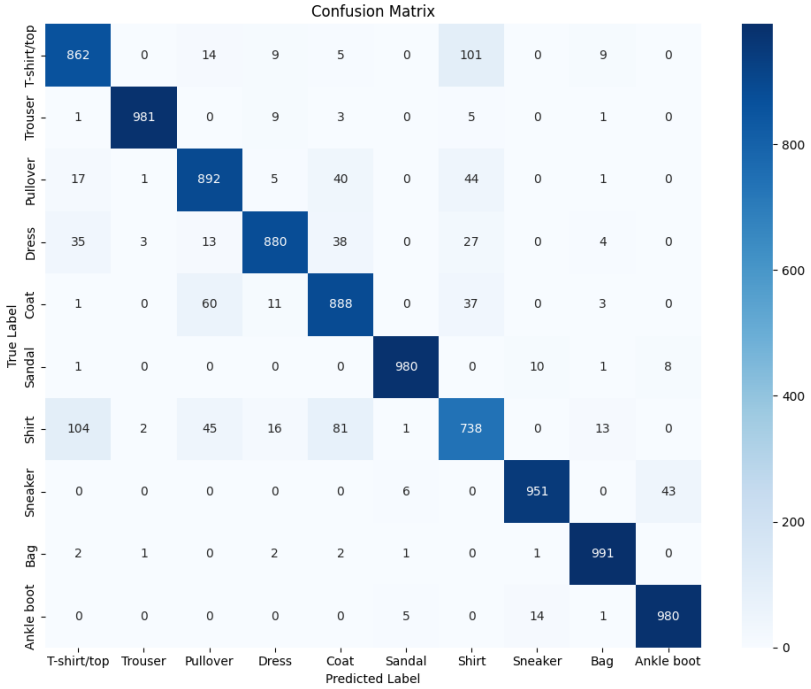

Figure 2: confusion matrix for standard model

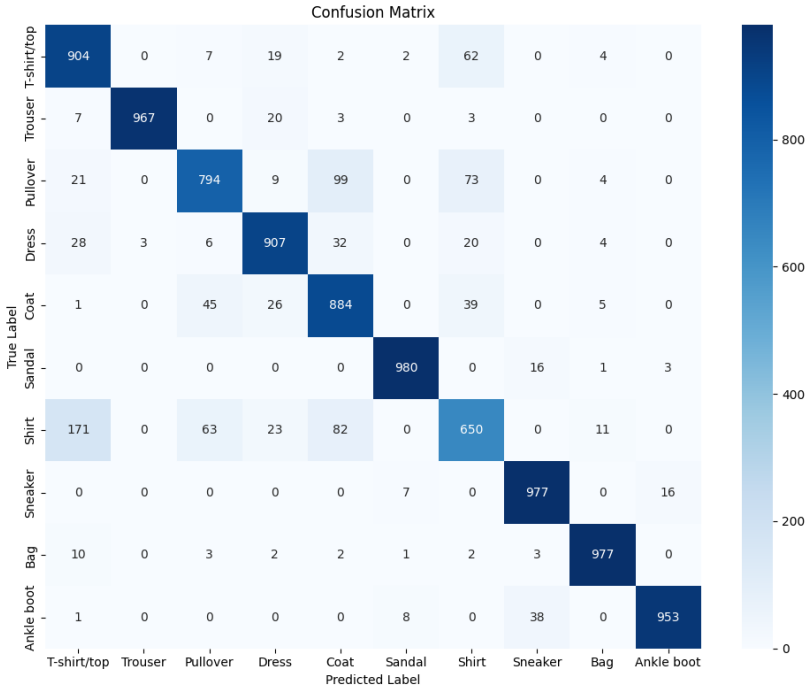

Figure 3: confusion matrix for tucker model

From our initial training, both models performed relatively well, achieving accuracies of 91% and 90%, respectively. Similarly, when examining the confusion matrices, we observed that the Tucker model incorrectly classified shirts and T-shirts significantly more often than the CNN. This is reasonable, as the Tucker model reduces dimensionality while attempting to preserve geometric features.

However, when analyzing the training results, the Tucker model did not perform as expected. The results showed almost no improvement, with only a 2% reduction in parameters and a 0.7× speedup in training. Rather than achieving the substantial compression we anticipated, the Tucker decomposition had only a minimal impact on the model.

To understand this more deeply, we can use theoretical analysis tools—SVD and PCA—and apply them to our problem. SVD will show us where our model's parameters actually reside and whether our layers are compressible. PCA will show us how compression affects the learned features and why certain classes (like Shirts) lost accuracy.

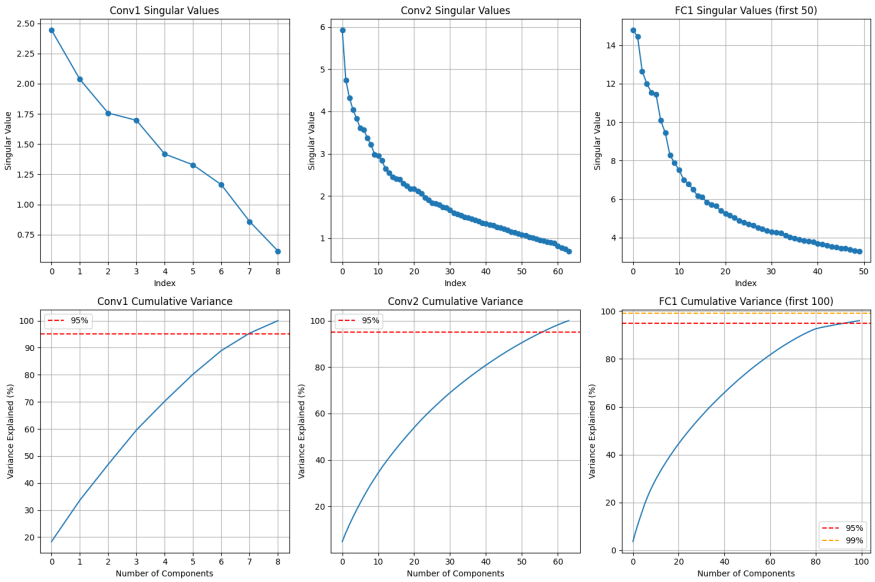

Figure 4: SVD CNN

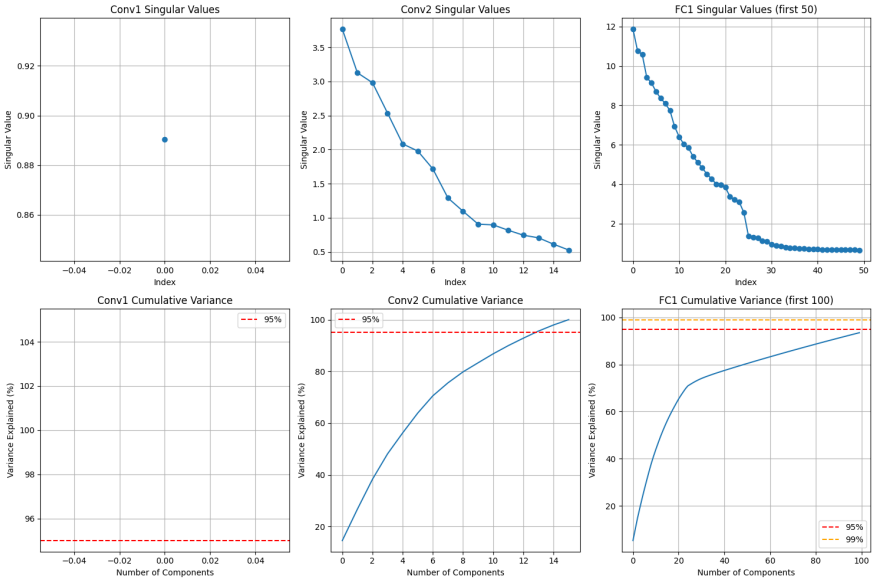

Figure 5: SVD Tucker

Table 1: Component Compression Summary

Layer	Standard	Tucker%	Difference
Conv1	9 components, need 7 for 95%	1 component	85%reduction
Conv2	64 components, need 40 for 95%	16 components, need 13 for 95%	75% reduction
FC1	128 components, need 65 for 95%	same	No compression

The SVD plots show that convolutional layers have sharp singular value dropoffs, meaning most information concentrates in the first few components—this proves they're highly compressible. Tucker exploits this by reducing Conv2 from 64 to 16 components while still capturing 95% of variance. However, FC1 shows a gradual decline requiring 65 of 128 components for 95% variance, and Tucker doesn't compress it at all. Since FC1 contains 95% of model parameters, compressing only the conv layers—despite their extreme compressibility—barely impacts the overall model size.

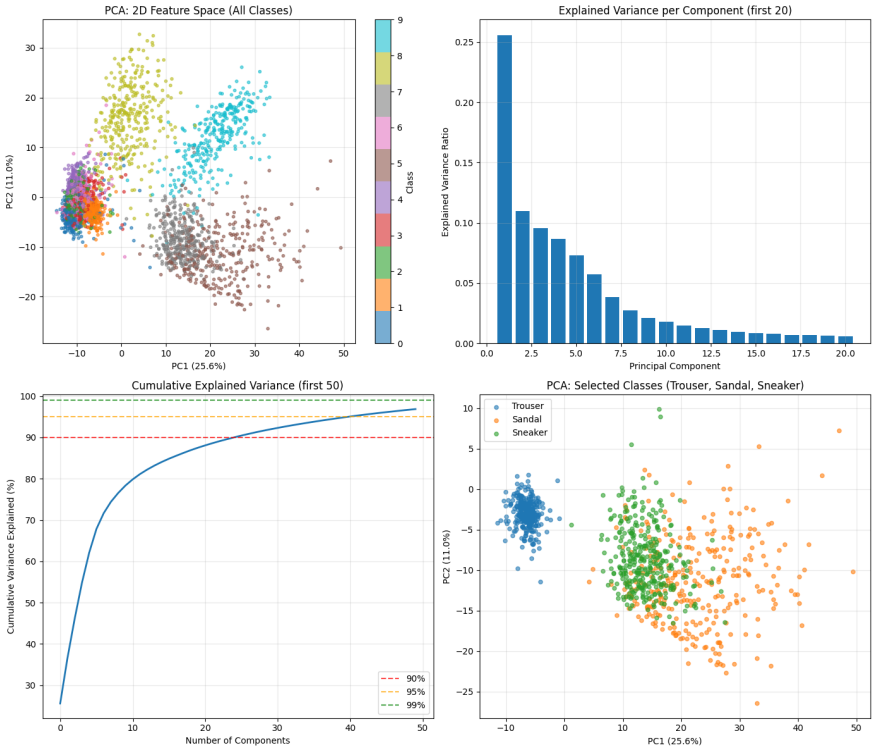

Figure 6: PCA CNN

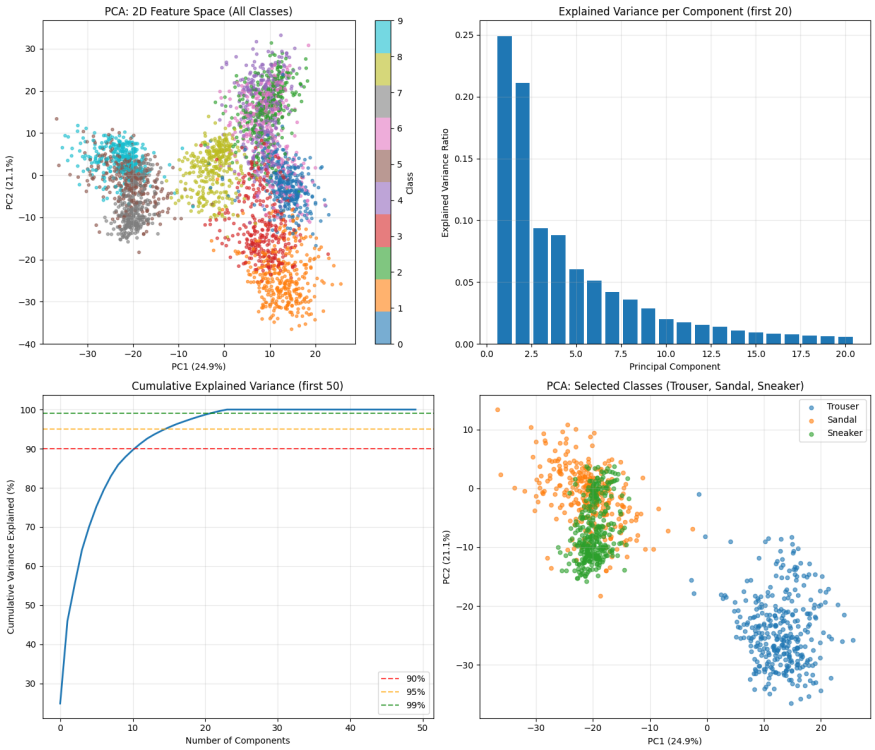

Figure 7: PCA Tucker

PCA shows both models learn nearly identical feature representations—PC1 captures ∼25% variance in both, and ∼10 components explain 90% of information. The 2D plots reveal good class separation overall, with expected overlap in similar categories (Sandal/Sneaker, which matches confusion matrix errors). Tucker's PCA is virtually identical to Standard's, proving compression preserved feature quality. The issue isn't Tucker degrading features—it's that we compressed only 5% of the model (conv layers), leaving the 95%-dominant FC architecture untouched.

Seeing that is overlap in the features, we computed grassmann distances between class feature subspaces to quantify geometric separation. However all class pairs, including confused pairs (Shirt-T-shirt: 0.0009, Sandal-Sneaker: 0.0011) and separated pairs (Trouser-Shirt: 0.0009), showed near-zero distances with principal angles ≈ 0. This indicates all classes occupy the same 128-dimensional subspace rather than distinct geometric subspaces, confirming that classification depends on cluster positions within a shared feature manifold, not subspace separation.

Having identified that FC1 dominates the parameter count (95% of the model), we turn to classical matrix analysis techniques to address this bottleneck. Rather than applying Tucker decomposition—which is designed for higher-order tensors—we apply low-rank matrix factorization.

We decompose the 3136×128 weight matrix W into a product of two smaller matrices:

$$W \approx W_1 W_2$$

where $W_1 \in \mathbb{R}^{64\times 3136}$ (compression matrix) and $W_2 \in \mathbb{R}^{128\times 64}$ (expansion matrix). This creates a 64-dimensional bottleneck, forcing the network to learn its singular value sized rank approximation of the original transformation.

The original FC1 layer performs $y = \sigma(Wx + b)$ where $x \in \mathbb{R}^{3136}$ is the flattened input, $W \in \mathbb{R}^{128\times 3136}$ is the weight matrix, and σ is the ReLU activation. We approximate $W \approx W_2 W_1$ using rank-r factorization, giving:

$$y = \sigma(W_2\, \sigma(W_1 x + b_1) + b_2) \tag{1}$$

The parameter count reduces from $128 \times 3136 + 128 = 401{,}536$ to $(64 \times 3136 + 64) + (128 \times 64 + 128) = 208{,}960$, achieving 48% compression.

Now we can create 4 models as combinations on compression on convolution layers and compressions on fully connected layer.

Table 2: Model Comparison

Model	**Params**	**Reduction**	**Accuracy**
Baseline	421,642	–	91.43%
Tucker Conv	410,699	2.6%	90.39%
Compressed FC	229,194	45.7%	90.92%
Tucker + FC	218,251	48.2%	90.45%

Results: Compressed FC achieved a 45.7% overall parameter reduction (421K $\rightarrow$ 229K) while maintaining 90.92% accuracy (only a 0.51% drop from the baseline). This clearly outperformed Tucker-only compression, which achieved a 2.6% reduction with a 1.04% accuracy loss. Combining both techniques (Tucker + FC) resulted in a 48.2% reduction with 90.45% accuracy—an acceptable trade-off for deployment scenarios that prioritize smaller model size.

Conclusion: This demonstrates the critical importance of analysis-driven compression: SVD identified where parameters reside, PCA revealed that compression works but was misapplied, and targeted matrix factorization solved the actual bottleneck. Tucker decomposition is effective—but only when applied to conv-heavy architectures. For FC-dominant networks, classical matrix factorization (low-rank approximation) is more appropriate.

Future Improvements: We chose our architecture based on speed and simplicity. We started with a single convolution layer for feature extraction followed by one fully connected layer for classification. Similarly, we chose to split the data into mini-batches of 64 samples so that any laptop could quickly train the model, and we implemented Adam as it is the standard optimization algorithm. Future work could include systematic hyperparameter tuning, learning rate scheduling, monitoring validation loss to determine the optimal number of training epochs, testing different architectures, and implementing regularization in the loss function.

4 Conclusion

In this project we explored the theory and applications of tensors as objects in both pure and applied mathematics. We examined how tensors generalize vectors and matrices as linear transformations, and how the tensor product provides a natural way of combining vector spaces. We saw that tensors are not only higher-dimensional arrays, but structured functions whose behavior under coordinate transformation encodes geometric and physical logic.

We also demonstrated how the abstract theory connects directly to real-world systems through various projects and demonstrations. In special relativity, tensors like the Faraday tensor $F_{\mu\nu}$ describe electric and magnetic fields in a unified way, transforming linearly under Lorentz boosts. We implemented Tucker decomposition to approximate color image data with reduced storage, showing how tensor factorizations generalize matrix SVD to higher dimensions for image and signal compression. Finally, we saw how a Convolutional Neural Network trained on the Fashion-MNIST dataset also uses tensors.

Across all these examples, we saw that tensors are a natural language for representing and manipulating multidimensional linear (or in other words, multilinear) relationships. Their universality and transformation properties make them indispensable tools in physics, engineering, and mathematics. This project not only deepened our understanding of the structure of tensors, but also revealed the power of multilinear algebra across different scientific and applied fields.

5 Sources

- "Tensor product," Wikipedia, Sep. 08, 2021. https://en.wikipedia.org/wiki/Tensor_product
- Y. Eliashberg, "Multilinear algebra, differential forms and Stokes' theorem," 2018. Accessed: Jun. 25, 2025. [Online]. Available: https://math.stanford.edu/ eliash/Public/math177/177-diff-forms.pdf
- S. Janson, "TENSORS AND DIFFERENTIAL FORMS." Accessed: Jun. 25, 2025. [Online]. Available: https://www2.math.uu.se/ svantejs/papers/sjN2.pdf
- D. J. Griffiths, Introduction to electrodynamics. Cambridge: Cambridge University Press, 2017.
- "Fashion MNIST," www.kaggle.com. https://www.kaggle.com/datasets/zalando-research/fashionmnist
- Wikipedia Contributors, "Fashion MNIST," Wikipedia, Oct. 10, 2024. https://en.wikipedia.org/wiki/Fashion_MNIST

Note: the code for the plots for Fashion CNN where assisted by the northeastern Claude account. We consider this to be an acceptable use of AI since it is tangentially related to our primary topic, and the explanation was written by a human with LaTeX typesetting assistance from ChatGPT.