GALALA UNIVERSITY
Powered by
Arizona State University

# Neural Network Interpretability: Bilinear MLPs & Transcoders


| Name | ID |
|---|---|
| Mostafa Mohamed Saleh | 223101188 |
| Youssef Ahmed Ibrahim | 223101109 |
| Karim Mohamed Mostafa | 223102240 |
| Youssef Bassem Atef | 223102042 |

## Prof. Shaker Elsappagh


## Overview


This report explores two complementary approaches to neural network interpretability: Bilinear MLPs and Transcoders. Both methods successfully extract interpretable, digit-specific features that explain model behavior, revealing that neural networks exhibit low-rank structure where a small number of features capture most of the computation.


1. Introduction
Understanding how neural networks make decisions is crucial for building trustworthy AI systems. This report examines two powerful techniques for interpreting neural network behavior through feature extraction and visualization.

### 1.1 Key Questions Addressed

- What features do neural networks learn?
- How can we visualize these features?
- Why do models make certain predictions?
- Can we understand model errors?

### 1.2 Two Approaches

**Bilinear MLPs** replace standard activation functions with bilinear operations, enabling direct mathematical analysis through eigendecomposition.

**Transcoders** use sparse autoencoders to learn interpretable features from pre-trained models without architectural modifications.

# 2. Bilinear MLPs

## 2.1 Overview

Bilinear MLPs replace standard ReLU activation functions with bilinear operations. This architectural change enables direct mathematical interpretation of learned features through eigendecomposition, without requiring additional training.

## 2.2 Architecture

A bilinear MLP consists of:

- **Input layer**: 784 dimensions (28×28 MNIST images)
- **Bilinear layer**: Implements x^T W x operation
- **Output layer**: 10 classes (digits 0-9)

## 2.3 Mathematical Foundation

The core operation is:

$$f(x) = x^T W x$$

Where W is a learnable weight matrix that can be decomposed as:

$$W = Q \Lambda Q^T$$

Through eigendecomposition, where:

- Q contains eigenvectors (the features)
- Λ contains eigenvalues (feature importance)

## 2.4 Training Configuration

| Parameter | Value |
|---|---|
| Learning Rate | 0.01 |
| Batch Size | 128 |
| Epochs | 10 |
| Test Accuracy | ~98% |

## 2.5 Feature Extraction via Eigendecomposition

Features are extracted by:

- Computing the weight matrix W for each output class

- Performing eigendecomposition: W = Q Λ Q^T
- Visualizing eigenvectors as 28×28 images
- Using eigenvalues to rank feature importance

### 2.6 Low-Rank Structure

Analysis reveals that a small number of features capture most model behavior:

| Top Eigenvectors | Accuracy |
|---|---|
| 10 | ~90% |
| 20 | ~95% |
| 50 | ~97% |

## 3. Transcoders

### 3.1 Overview

Transcoders use sparse autoencoders to learn interpretable features from pre-trained neural networks. Unlike bilinear MLPs, transcoders work with any existing architecture without requiring changes to the original model.

### 3.2 Architecture

A transcoder consists of three components:

- **Encoder**: Maps hidden states to sparse feature space
- **Decoder**: Reconstructs hidden states from features
- **ReLU activation**: Enforces sparsity in feature activations

### 3.3 Mathematical Foundation

The encoding process:

$$features = ReLU(W\_enc \cdot h + b\_enc)$$

The reconstruction:

$$h\_reconstructed = W\_dec \cdot features + b\_dec$$

### 3.4 Training Configuration

| Parameter | Value |
|---|---|
| Hidden Dimension | 128 |
| Feature Dimension | 64 |

| Parameter | Value |
|---|---|
| L1 Penalty | 1e-4 |
| Test Accuracy | ~98% |

### 3.5 Feature Analysis

Transcoders enable multiple types of feature analysis:

- **Class Selectivity**: Which features are specific to each digit
- **Activation Frequency**: How often each feature fires
- **Feature Importance**: Which features contribute most to predictions

### 3.6 Low-Rank Structure

| Active Features | Accuracy |
|---|---|
| 8 | ~85% |
| 16 | ~92% |
| 32 | ~96% |
| 64 (full) | ~98% |

## 4. Feature Visualization Techniques

### 4.1 Bilinear MLP Visualization

Eigenvectors can be reshaped and visualized as 28×28 images. Color coding reveals:

- **Blue regions**: Positive contribution - pixels here activate the feature
- **Red regions**: Negative contribution - pixels here also activate (XOR behavior)
- **White regions**: Near-zero contribution - pixels here don't matter

### 4.2 Eigenvalue Interpretation

- **Positive eigenvalues**: Features that support classification as this digit
- **Negative eigenvalues**: Features that indicate it's NOT this digit

### 4.3 Transcoder Visualization

Encoder weights represent templates that features look for in the input. Additional analysis includes:

- **Activation Frequency**: How often each feature fires (some features are 'dead')
- **Class Selectivity**: How specific each feature is to certain digit classes
- **Co-activation**: Which features tend to fire together

## 5. Comparative Analysis

### 5.1 Side-by-Side Comparison

| Aspect | Bilinear MLP | Transcoder |
|---|---|---|
| Method | Eigendecomposition | Sparse Autoencoder |
| Feature Source | Eigenvectors | Encoder weights |
| Extra Training | No | Yes |
| Works with ReLU | No | Yes |
| Test Accuracy | ~98% | ~98% |

### 5.2 Advantages and Limitations

#### Bilinear MLPs

**Advantages:**

- No extra training required
- Exact mathematical interpretation
- Features derived directly from weights
- Can construct adversarial examples analytically

**Limitations:**

- Requires architectural change
- May have performance gap vs standard MLPs at scale
- Only validated on smaller models

#### Transcoders

**Advantages:**

- Works with any existing architecture
- No changes to original model
- Flexible sparsity control
- Rich activation-based analysis

**Limitations:**

- Requires additional training
- Features are approximate, not exact
- May miss some model behaviors

## 6. Key Findings

### 6.1 Low-Rank Structure Discovery

Both methods reveal that neural networks exhibit low-rank structure. A small number of interpretable features (10-32) capture most of the model's behavior. This is encouraging for interpretability research, as it suggests neural networks may be more tractable to understand than previously thought.

### 6.2 Feature Interpretability

Extracted features are highly interpretable and can be visualized as meaningful patterns. Features correspond to recognizable aspects of digits such as curves, loops, vertical lines, and horizontal strokes.

### 6.3 Error Analysis Capabilities

Both methods enable understanding why models make mistakes by identifying:

- **Misleading features**: Features that fired unexpectedly
- **Missing features**: Features that should have fired but didn't

## 7. Applications

These interpretability techniques enable several practical applications:

- **Model Debugging**: Understand why models make specific predictions
- **Overfitting Detection**: Identify when models learn spurious patterns
- **Adversarial Robustness**: Construct and defend against adversarial examples
- **Knowledge Extraction**: Understand what concepts models have learned
- **Model Comparison**: Compare what different architectures learn

## 8. Conclusions

This report has examined two complementary approaches to neural network interpretability. Bilinear MLPs offer exact mathematical interpretation through eigendecomposition, while transcoders provide flexible post-hoc analysis of pre-trained models.

Both methods successfully reveal that neural networks learn low-rank representations, where a small number of interpretable features capture most model behavior. This finding is encouraging for the broader field of AI interpretability and suggests that understanding complex neural networks may be more tractable than previously believed.

The techniques demonstrated here on MNIST provide a foundation for scaling to more complex models and datasets, enabling safer and more trustworthy AI systems through improved transparency and understanding.

## 9.References

1.Transcoders Find Interpretable LLM Feature Circuits

https://arxiv.org/abs/2406.11944

2.Bilinear MLPs Enable Weight-Based Mechanistic Interpretability

https://arxiv.org/abs/2410.08417