Applied and Computational Harmonic Analysis 68 (2024) 101601

Contents lists available at ScienceDirect

Applied and Computational Harmonic Analysis

journal homepage: www.elsevier.com/locate/acha

# LU decomposition and Toeplitz decomposition of a neural network

Yucong Liu [a], Simiao Jiao [a], Lek-Heng Lim [b,*]

[a] Department of Statistics, University of Chicago, Chicago, IL 60637, United States of America
[b] Computational and Applied Mathematics Initiative, University of Chicago, Chicago, IL 60637, United States of America

ARTICLE INFO

Communicated by Gerlind Plonka

*Keywords:*
Neural networks
Toeplitz matrices
Hankel matrices
Triangular matrices
Convolutional neural networks
Universal approximation

ABSTRACT

Any matrix $A$ has an LU decomposition up to a row or column permutation. Less well-known is the fact that it has a 'Toeplitz decomposition' $A = T_1 T_2 \cdots T_r$ where $T_i$'s are Toeplitz matrices. We will prove that any continuous function $f : \mathbb{R}^n \to \mathbb{R}^m$ has an approximation to arbitrary accuracy by a neural network that maps $x \in \mathbb{R}^n$ to $L_1\sigma_1 U_1 \sigma_2 L_2 \sigma_3 U_2 \cdots L_r \sigma_{2r-1} U_r x \in \mathbb{R}^m$, i.e., where the weight matrices alternate between lower and upper triangular matrices, $\sigma_i(x) := \sigma(x - b_i)$ for some bias vector $b_i$, and the activation $\sigma$ may be chosen to be essentially any uniformly continuous nonpolynomial function. The same result also holds with Toeplitz matrices, i.e., $f \approx T_1 \sigma_1 T_2 \sigma_2 \cdots \sigma_{r-1} T_r$ to arbitrary accuracy, and likewise for Hankel matrices. A consequence of our Toeplitz result is a fixed-width universal approximation theorem for convolutional neural networks, which so far have only arbitrary width versions. Since our results apply in particular to the case when $f$ is a general neural network, we may regard them as LU and Toeplitz decompositions of a neural network. The practical implication of our results is that one may vastly reduce the number of weight parameters in a neural network without sacrificing its power of universal approximation. We will present several experiments on real data sets to show that imposing such structures on the weight matrices dramatically reduces the number of training parameters with almost no noticeable effect on test accuracy.

## 1. Introduction

Among the numerous results used to justify and explain the efficacy of feed-forward neural networks, possibly the best known are the universal approximation theorems of various types. These theorems explain the expressive power of neural networks by showing that they can approximate various classes of functions to arbitrary accuracy under various measures of accuracy. The universal approximation theorems in the literature may be divided into two categories, applying respectively to:

(i) *shallow wide networks:* neural networks of fixed depth and arbitrary width;
(ii) *deep narrow networks:* neural networks with fixed width and arbitrary depth.

In the first category, we have the celebrated results of Cybenko [1], Hornik [9], Pinkus [26], et al. We state the last of these for easy reference:

* Corresponding author.
*E-mail addresses:* yucongliu@uchicago.edu (Y. Liu), smjiao@uchicago.edu (S. Jiao), lekheng@uchicago.edu (L.-H. Lim).

https://doi.org/10.1016/j.acha.2023.101601
Received 28 November 2022; Received in revised form 29 July 2023; Accepted 27 September 2023

Available online 6 October 2023

**Theorem 1.1** *(Pinkus [26]). Let $\sigma \in C(\mathbb{R})$ and $\Omega \subseteq \mathbb{R}^n$ be compact. The set of $\sigma$-activated neural networks with one hidden layer and arbitrary width is dense in $C(\Omega, \mathbb{R}^m)$ with respect to the uniform norm if and only if $\sigma$ is not a polynomial.*

In the second category, an example is provided by Kidger and Lyons [14], again quoted below for easy reference:

**Theorem 1.2** *(Kidger and Lyons [14]). Let $\sigma \in C(\mathbb{R})$ be a nonpolynomial function, continuously differentiable at at least one point, with nonzero derivative at that point. Let $\Omega \subseteq \mathbb{R}^n$ be compact. Then the set of $\sigma$-activated neural networks with fixed width $m+n+1$ and arbitrary depth is dense in $C(\Omega, \mathbb{R}^m)$ with respect to the uniform norm.*

In all these results, the weight matrices used in each layer are assumed to be dense general matrices; in particular, these neural networks are fully connected. The goal of our article is to show that even when we impose special structures on the weight matrices — upper and lower triangular, Toeplitz or Hankel — we will still have the same type of universal approximation results, for both shallow wide and deep narrow networks alike. In addition, our numerical experiments will show that when kept at the same depth and width, a neural network with these structured weight matrices suffers almost no loss in expressive powers, but requires only a fraction of the parameters — note that an $m \times n$ triangular matrix with $p = \max(m, n)$ has at most $p(p+1)/2$ parameters whereas an $m \times n$ Toeplitz or Hankel matrix has exactly $m + n - 1$ parameters.

The saving in training cost goes beyond a mere reduction in the number of weight parameters. The forward and backward propagations in the training process ultimately reduce to matrix-vector products. For Toeplitz or Hankel matrices, these come at a cost of $O(n \log n)$ operations as opposed to the usual $O(n^2)$.

An alternative way to view our results is that these are "LU decomposition" and "Toeplitz decomposition" of a nonlinear function in the context of neural networks. A departure from the case of linear functions is that an LU decomposition of a nonlinear function requires not just one lower triangular matrix and one upper triangular matrix but several of these alternating between lower triangular and upper triangular, and sandwiching an activation. The Toeplitz (or Hankel) decomposition of a linear function is a consequence of the following result, which can be readily extended to $m \times n$ matrices, as we will see in Section 2.2.

**Theorem 1.3** *(Ye and Lim [30]). Every $n \times n$ matrix can be expressed as a product of $2n+5$ Toeplitz matrices or $2n+5$ Hankel matrices.*

Again we will see that this also applies to a nonlinear continuous function as long as we introduce an activation function between every Toeplitz or Hankel factor. Another caveat in these results is that the exact equality used in linear algebra is replaced by the most common notion of equality in approximation theory, namely, equality up to an arbitrarily small error. As in Theorems 1.1 and 1.2, our results will apply with essentially any nonpolynomial continuous activations, including but not limited to common ones like ReLU, leaky ReLU, sigmoidal, hyperbolic tangent, etc.

We will prove these results in Section 2, with shallow wide neural networks in Section 2.1 and deep narrow neural networks Section 2.2, after discussing prior works in Section 1.1 and setting up notations in Section 1.2. The experiments showing the practical side of these results are in Section 4 with a cost analysis in Section 3.

### 1.1. Prior works

We present a more careful discussion of existing works in the literature, in rough chronological order. To the best of our knowledge, there are six main lines of works related to ours. While none replicates our results in Section 2, they show a progression towards our work in spirit — with the increase in width and depth of neural networks, it has become an important endeavor to reduce the number of redundant training parameters through other means.

*Shallow wide neural networks:* The earliest universal approximation theorems are for one-hidden-layer neural networks with arbitrary width, beginning with the eponymous theorem of Cybenko [1], which shows that a fully-connected sigmoid-activated network with one hidden layer and an arbitrary number of neurons can approximate any continuous function on the unit cube in $\mathbb{R}^n$ up to arbitrary accuracy. Cybenko's argument also works for ReLU activation and could be extended to a fixed number of hidden layers simply by requiring that the additional hidden layers approximate an identity map. Hornik et al. [10] obtained the next major generalization to nondecreasing activations with $\lim_{x\to-\infty} \sigma(x) = 0$ and $\lim_{x\to+\infty} \sigma(x) = 1$. The most general universal approximation theorem along this line is that of Pinkus [26] stated earlier in Theorem 1.1. The striking aspect is that it is a necessary and sufficient condition, showing that such universal approximation property characterizes the "nonpolynomialness" of the activation function.

*Deep narrow networks:* With the advent of deep neural networks, the focus has changed to keeping the width fixed and allowing the depth to increase. Lu et al. [21] showed that ReLU-activated neural networks of width $n+4$ and arbitrary depth are dense in $L^1(\mathbb{R}^n)$. Hanin and Sellke [7] showed that such neural networks of width $m+n$ are dense in $C(\Omega, \mathbb{R}^m)$ for any compact $\Omega \subseteq \mathbb{R}^n$. The aforementioned Theorem 1.2 of Kidger and Lyons [14] is another alternative with more general continuous activations and with width $m+n+1$. An extreme case is provided by Lin and Jegelka [20] for ResNet with a single neuron per hidden layer but with depth going to infinity.

*Width-depth tradeoff:* The tradeoff between width and depth of a neural network is now well studied. The results of Eldan and Shamir [2], Telgarsky [28] explain the benefits of having more layers — a deep neural network cannot be well approximated by shallow neural networks unless they are exponentially large. On the other hand, the results of Johnson [12], Park et al. [25] revealed the limitations of deep neural networks — they require a minimum width for universal approximation; although these results do not cover exotic structures like ResNet. There are also studies on the memory capacity of wide and deep neural networks [31].

*Neural network pruning:* Pruning refers to techniques for eliminating redundant weights from neural networks and it has a long history [17,8,6,19]. A recent highlight is the lottery ticket hypothesis proposed in Frankle and Carbin [3] that led to extensive follow-up work [23,4,22]. Our results in Section 2 may be viewed as a particularly aggressive type of pruning whereby we either set half the weight parameters to zero, as in the LU case, or even reduce the number of weight parameters by an order of magnitude, from $O(n^2)$ to $O(n)$, as in the Toeplitz/Hankel case.

*Convolutional neural networks:* The result closest to ours is likely the universal approximation theorem for deep convolutional neural network of Zhou [32]. However his result provides the necessary width and depth in terms of the approximating accuracy $\varepsilon$, and as such requires arbitrary width and depth at the same time. We will deduce an alternative version with fixed width in Corollary 2.5.

*Hardware acceleration:* In the context of accelerating training of neural networks via GPUs, FPGAs, ASICs, and other specialized hardware (e.g., Google's TPU, Nvidia's H100 AI processor), there have been prior works on exploiting structured matrix algorithms for matrix-vector multiply, notably for triangular matrices in [11] and Toeplitz matrices in [13].

### 1.2. Notations and conventions

We write $\|\cdot\|$ for both the Euclidean norm on $\mathbb{R}^n$ and the Frobenius norm on $\mathbb{R}^{m\times n}$. The zero matrix in $\mathbb{R}^{m\times n}$ is denoted $\mathbb{0}_{m\times n}$. The zero vector and the vector of all ones in $\mathbb{R}^n$ will be denoted $\mathbb{0}_n$ and $\mathbb{1}_n$ respectively.

Let $A = (a_{ij}) \in \mathbb{R}^{m\times n}$. If $a_{ij} = 0$ whenever $i > j$, then $A$ is *upper triangular*; if $a_{ij} = 0$ whenever $i < j$, then $A$ is *lower triangular*. A matrix is *Toeplitz* (resp. *Hankel*) if has equal entries along its diagonals (resp. reverse diagonals). More precisely, $A$ is Toeplitz if $a_{i,i+r} = a_{j,j+r}$ whenever $-m+1 \le r \le n-1$, $1 \le i, j \le m$, $1 \le i+r, j+r \le n$. Similarly $A$ is Hankel if $a_{i,r-i} = a_{j,r-j}$ whenever $2 \le r \le m+n$, $1 \le i, j \le m$, $1 \le r-i, r-j, \le n$. Note that the definitions of these structured matrices do not require that $m = n$.

An $m \times n$ Toeplitz or Hankel matrix requires only $m+n-1$ parameters to specify — standard convention is to just store the first row and first column of a Toeplitz matrix and the first row and last column of a Hankel matrix. For example, when $m = n$, we have

$$T = \begin{bmatrix} a_0 & a_{-1} & & a_{1-n} \\ a_1 & a_0 & \ddots & \\ & \ddots & \ddots & a_{-1} \\ a_{n-1} & & a_1 & a_0 \end{bmatrix}, \qquad H = \begin{bmatrix} a_0 & a_1 & \cdots & a_{n-1} \\ a_1 & a_2 & \iddots & a_n \\ \vdots & \iddots & \iddots & \vdots \\ a_{n-1} & a_n & \cdots & a_{2n-2} \end{bmatrix}.$$

We write $C(\Omega, \mathbb{R}^m)$ for the set of continuous functions on $\Omega$ taking values in $\mathbb{R}^m$, with $C(\Omega)$ for the special case when $m = 1$. Throughout this article we will use the uniform norm for all function approximations; there will be no confusion with the norms introduced above as we will always specify our uniform norm explicitly as $\sup_{x\in\Omega}$.

Any univariate function $\sigma : \mathbb{R} \to \mathbb{R}$ defines a *pointwise activation* $\sigma : \mathbb{R}^n \to \mathbb{R}^n$ for any $n \in \mathbb{N}$ through applying $\sigma$ coordinatewise to vectors in $\mathbb{R}^n$. We will sometimes drop the parentheses, writing $\sigma x$ to mean $\sigma(x)$, to reduce notational clutter.

A $k$-layer neural network $\nu : \mathbb{R}^n \to \mathbb{R}^m$ has the following structure:

$$\nu(x) = A_k \sigma_{k-1} A_{k-1} \sigma_{k-1} \cdots \sigma_2 A_2 \sigma_1 A_1 x + b_k$$

for any input $x \in \mathbb{R}^n$, *weight* matrix $A_i \in \mathbb{R}^{n_i \times n_{i-1}}$,

$$\sigma_i(x) := \sigma(x + b_i)$$

with $b_i \in \mathbb{R}^{n_i}$ the *bias* vector, and $\sigma$ the *activation* function. The output size of the $i$th *layer* is $n_i$ and always equals the input size of $(i+1)$th layer, with $n_0 = n$ and $n_k = m$. If there is no special structure on the weight matrix $A_i$, then the $i$th layer is called a *fully-connected layer*.

The class of *convolutional neural networks* deserves special mention, not least because they launched the deep learning revolution [16]. If the weight matrix $A_i \in \mathbb{R}^{s\times t}$ arises from a convolution with some *kernel* vector $\kappa = (a_{1-t}, \dots, a_0, \dots, a_{s-1}) \in \mathbb{R}^{s+t-1}$, then the $i$th layer is called a *convolutional layer*. In other words, the $i$th layer may be expressed as $\sigma(\kappa * x + b_i)$, where $*$ denotes the *convolution* operation. A convolutional neural network is one where the initial layers are all convolutional and the subsequent layers are all fully-connected.

## 2. Universal approximation by structured neural networks

We present our main results and proofs, beginning with shallow wide networks and followed by deep narrow networks.

### 2.1. Fixed depth, arbitrary width

This is an easy case that we state for completeness. Our universal approximation result in this case only holds for scalar-valued functions. The more interesting case for arbitrary depth neural networks in Section 2.2 will hold for vector-valued functions.

We begin with an observation that, if width is not a limitation, then any general weight matrix may be transformed into a Toeplitz or Hankel matrix.

**Lemma 2.1** *(General matrices to Toeplitz/Hankel matrices). Any matrix $A \in \mathbb{R}^{m\times n}$ can be transformed into a Toeplitz or Hankel matrix by inserting additional rows.*

**Proof.** This is best illustrated by way of a simple example first. For a $2\times 2$ matrix

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix},$$

inserting a row vector in the middle makes it Toeplitz:

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{22} & a_{11} \\ a_{21} & a_{22} \end{bmatrix};$$

and similarly inserting a different row vector in the middle makes it Hankel:

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{12} & a_{21} \\ a_{21} & a_{22} \end{bmatrix}.$$

For an $m\times n$ matrix

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix},$$

inserting $n-1$ rows between the first and the second row

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \end{bmatrix}$$

turns it Toeplitz

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{2n} & a_{11} & \ddots & \vdots \\ \vdots & \ddots & \ddots & a_{12} \\ a_{22} & & a_{2n} & a_{11} \\ a_{21} & a_{22} & \cdots & a_{2n} \end{bmatrix}.$$

Now repeat this to the remaining pairs of adjacent rows of $A$, we see that after inserting a total of $(m-1)(n-1)$ rows, we obtain an $(mn-n+1)\times n$ Toeplitz matrix. The process for transforming a general $m\times n$ matrix into a Hankel matrix by inserting rows is similar. □

Evidently, the statement and proof of Lemma 2.1 remain true if 'row' is replaced by 'column' but we will only need the row version in our proofs.

**Theorem 2.2** *(Universal approximation by structured neural networks I). Let $\Omega \subseteq \mathbb{R}^n$ be compact and $\sigma : \mathbb{R} \to \mathbb{R}$ be nonpolynomial. For any $f \in C(\mathbb{R}^n)$ and any $\varepsilon > 0$, we have*

$$\sup_{x\in\Omega} |f(x) - \nu(x)| \le \varepsilon$$

*for some one-layer neural network $\nu : \mathbb{R}^n \to \mathbb{R}$,*

$$\nu(x) = a^{\mathsf{T}}\sigma(Ax + b),$$

*with $a, b \in \mathbb{R}^m$, $m \in \mathbb{N}$, and $A \in \mathbb{R}^{m\times n}$ that can be chosen to be*

(i) *a Toeplitz matrix,*

(ii) *a Hankel matrix,*
(iii) *or a lower triangular matrix.*

**Proof.** It follows from Theorem 1.1 that for a given $f \in C(\mathbb{R}^n)$, there exist $c, d \in \mathbb{R}^p$, $p \in \mathbb{N}$, $B \in \mathbb{R}^{p\times n}$ so that

$$\sup_{x\in\Omega} |f(x) - d^{\mathsf{T}}\sigma(Bx + c)| \le \varepsilon.$$

Here of course $B$ has no specific structure. We begin with the Toeplitz case. By Lemma 2.1, we first transform $B \in \mathbb{R}^{p\times n}$ into a Toeplitz matrix $A \in \mathbb{R}^{m\times n}$ for some $m \in \mathbb{N}$. Since $A$ is obtained from $B$ by inserting rows, let the rows $i_1, \dots, i_p$ of $A$ be rows $1, \dots, p$ of $B$. Now let $a \in \mathbb{R}^m$ be the vector whose $i_j$th entry is exactly the $j$th entry of $d$ and zeroes everywhere else. Likewise, let $b \in \mathbb{R}^m$ be the vector whose $i_j$th entry is exactly the $j$th entry of $c$ and zeroes everywhere else. Then we clearly have $d^{\mathsf{T}}\sigma(Bx + c) = a^{\mathsf{T}}\sigma(Ax + b)$ and the required result follows. The Hankel case is identical. For the remaining case, we set

$$a = \begin{bmatrix} \mathbb{0}_n \\ d \end{bmatrix}, \quad A = \begin{bmatrix} \mathbb{0}_{n\times n} \\ B \end{bmatrix}, \quad b = \begin{bmatrix} \mathbb{0}_n \\ c \end{bmatrix},$$

and observe that $d^{\mathsf{T}}\sigma(Bx + c) = a^{\mathsf{T}}\sigma(Ax + b)$. Hence the required result follows. □

Theorem 2.2 is false if $A$ is required to be upper triangular. To see this, take $n = 2$, and let $A \in \mathbb{R}^{m\times 2}$ be upper triangular. A one-layer neural network $\nu : \mathbb{R}^2 \to \mathbb{R}$ with $A$ as weight matrix takes the form $\nu(x_1, x_2) = s_1\sigma(a_{11}x_1 + a_{12}x_2 + b_1) + s_2\sigma(a_{22}x_2 + b_2) + c$. If we just set $\sigma$ to be the ReLU activation, then clearly $\nu$ will not be able to approximate a function like, say, $f(x_1, x_2) = x_1^2 + x_2^2$ arbitrarily well. The point is that an upper triangular $A \in \mathbb{R}^{m\times 2}$ has at most three parameters regardless of how large $m$ is, so the neural network $\nu$ defined by $A$ is a piecewise linear function with at most four linear pieces, and therefore cannot approximate $f$ to arbitrary accuracy. This is also why the counterexample does not apply to a lower triangular $A \in \mathbb{R}^{m\times 2}$, which has $2m - 1$ parameters and its associated neural network can approximate any continuous $f$ to arbitrary accuracy when $m$ is large enough (as shown in the proof above).

### 2.2. Fixed width, arbitrary depth

The one-layer arbitrary width case above is more of a curiosity. Modern neural networks are almost invariably multilayer and we now provide the result that applies to this case. We first show that the identity map on $\mathbb{R}^n$ may be approximated by essentially any continuous pointwise activation. This is a generalization of [14, Lemma 4.1].

**Lemma 2.3** *(Approximation of identity). Let $\sigma : \mathbb{R} \to \mathbb{R}$ be any continuous function that is continuously differentiable at some point $a \in \mathbb{R}$ with $\sigma'(a) \neq 0$. Let $I : \mathbb{R}^n \to \mathbb{R}^n$ be the identity map. Then for any compact $\Omega \subseteq \mathbb{R}^n$ and any $\varepsilon > 0$, there exists a $\delta > 0$ such that whenever $0 < |h| < \delta$, the function $\rho_h : \mathbb{R}^n \to \mathbb{R}^n$,*

$$\rho_h(x) := \frac{1}{h\sigma'(a)}[\sigma(hx + a\mathbb{1}_n) - \sigma(a\mathbb{1}_n)], \tag{1}$$

*satisfies*

$$\sup_{x\in\Omega} \|\rho_h(x) - I(x)\| \le \varepsilon.$$

**Proof.** Subscript $i$ in this proof refers to the $i$th coordinate. As $\Omega$ is compact, $|x_i| \le L$ for some $L > 0$ and for all $i = 1, \dots, n$. Since the derivative $\sigma'$ is continuous, there exists $\eta > 0$ such that

$$|\sigma'(b) - \sigma'(a)| < \frac{|\sigma'(a)|\varepsilon}{L\sqrt{n}}$$

whenever $|b - a| \le \eta$. Let $\delta = \eta/L$. Then for $0 < |h| < \delta$, we have

$$|\rho_h(x)_i - x_i| = \left|\frac{\sigma(a + hx_i) - \sigma(a)}{h\sigma'(a)} - x_i\right| = \left|\frac{x_i\sigma'(\xi)}{\sigma'(\alpha)} - x_i\right| \le L\left|\frac{\sigma'(\xi) - \sigma'(a)}{\sigma'(a)}\right| \le \frac{\varepsilon}{\sqrt{n}}$$

for some $\xi$ between $a + hx_i$ and $a$ by the mean value theorem. The last inequality follows from $|\xi - a| \le |hx_1| \le |h|L \le \eta$. Note that $\delta$ is independent of all $x_i$'s and therefore $x$. Hence we may take $\sup_{x\in\Omega}\|\cdot\|$ to get the required result. □

The proof of our main result below relies on two facts: that we may use $\rho_h$ to approximate the identity map; and that if we scale the input of our activation by $h$ or scale the output by $1/h\sigma'(a)$, it does not affect the structure of our weight matrices — Toeplitz, Hankel, and triangular structures are preserved under scalar multiplication.

**Theorem 2.4** *(Universal approximation by structured neural networks II). Let $\Omega \subseteq \mathbb{R}^n$ be compact and $\sigma : \mathbb{R} \to \mathbb{R}$ be any uniformly continuous nonpolynomial function that is continuously differentiable at at least one point in $\Omega$, and has nonzero derivative at that point. For any $f \in C(\mathbb{R}^n, \mathbb{R}^m)$ and any $\varepsilon > 0$, there exists a neural network $\nu : \mathbb{R}^n \to \mathbb{R}^m$,*

$$\nu(x) = A_k\sigma_{k-1}A_{k-1}\sigma_{k-1}\cdots\sigma_2 A_2\sigma_1 A_1 x + b_k,$$

*such that*

$$\sup_{x\in\Omega}\|f(x)-\nu(x)\| < \varepsilon,$$

*where the weight matrices* $A_1 \in \mathbb{R}^{(m+n+1)\times n}$,

$$A_2,\dots,A_{k-1} \in \mathbb{R}^{(m+n+1)\times(m+n+1)},$$

*and* $A_k \in \mathbb{R}^{m\times(m+n+1)}$ *may be chosen to be*

(i) *all Toeplitz,*
(ii) *all Hankel,*
(iii) *upper triangular for odd-indexed* $A_i$ *and lower triangular for even-indexed* $A_i$*;*

*with bias vectors* $b_1,\dots,b_{k-1} \in \mathbb{R}^{m+n+1}$, $b_k \in \mathbb{R}^m$, *and* $\sigma_i(x) := \sigma(x+b_i)$.

**Proof.** By Theorem 1.2, there is a neural network $\varphi$ of width $m+n+1$ such that $\sup_{x\in\Omega}\|f(x)-\varphi(x)\| < \varepsilon/2$. We will write $\varphi$ recursively as

$$\varphi(x) = B_k\varphi_k(x) + c_k$$

with $\varphi_0(x) = x$ and $\varphi_{j+1}(x) = \sigma(B_j\varphi_j(x)+c_j)$, $j = 1,\dots,k-1$. Here $B_1 \in \mathbb{R}^{(m+n+1)\times n}$, $B_k \in \mathbb{R}^{m\times(m+n+1)}$, and $B_2,\dots,B_{k-1} \in \mathbb{R}^{(m+n+1)\times(m+n+1)}$.

By Theorem 1.3, the square matrices $B_2,\dots,B_{k-1}$ may each be decomposed into a product of Toeplitz matrices:

$$B_j = T_1^{(j)}T_2^{(j)}\cdots T_{r_j}^{(j)}. \tag{2}$$

As for $B_1$, we have

$$B_1 = [B_1, \mathbb{0}_{(m+n+1)\times(m+1)}]\begin{bmatrix} I_n \\ \mathbb{0}_{(m+1)\times n}\end{bmatrix}$$

and as $[I_n, \mathbb{0}_{n\times(m+1)}]^{\mathsf{T}} \in \mathbb{R}^{(m+n+1)\times n}$ is a rectangular Toeplitz matrix and Theorem 1.3 applies to the square matrix $[B_1, \mathbb{0}_{(n+m+1)\times(m+1)}] \in \mathbb{R}^{(m+n+1)\times(m+n+1)}$, we also have a Toeplitz decomposition for $B_1$. The argument applied to $B_1$ also applies to $B_k^{\mathsf{T}}$. Hence we have

$$B_1 = T_1^{(1)}\cdots T_{r_1}^{(1)}, \quad B_k = T_1^{(k)}\cdots T_{r_k}^{(k)}$$

as well. We thus obtain

$$\varphi(x) = T_1^{(k)}\cdots T_{r_k}^{(k)}\varphi_k(x) + c_k$$

with $\varphi_0(x) = x$ and

$$\varphi_{j+1}(x) = \sigma\big(T_1^{(j)}\cdots T_{r_j}^{(j)}\varphi_j(x) + c_j\big) \tag{3}$$

for $j = 1,\dots,k-1$.

Let us fix $j$ and drop the superscripts to avoid notational clutter. Between each adjacent pair of Toeplitz matrices $T_i$ and $T_{i+1}$, we may insert an identity map $I : \mathbb{R}^{n+m+1} \to \mathbb{R}^{n+m+1}$ and apply Lemma 2.3 to approximate $I$ by $\rho_{h_i}$ for some $h_i$ depending on $T_i$ and $T_{i+1}$ to be chosen later. Since

$$\begin{aligned} T_i\rho_{h_i}T_{i+1}x &= \frac{1}{h_i\sigma'(a)}T_i\sigma(h_iT_{i+1}x + a\mathbb{1}_n) - \frac{\sigma(a)}{h_i\sigma'(a)}T_i\mathbb{1}_n \\ &=: T_i'\sigma(T_{i+1}'x + b_{i+1}) + b_i \end{aligned} \tag{4}$$

each of these terms has the form we need, and the bias vectors are given by

$$b_{i+1} := a\mathbb{1}_n \qquad \text{and} \qquad b_i := -\frac{\sigma(a)}{h_i\sigma'(a)}T_i\mathbb{1}_n.$$

Observe that the matrices $T_i' := (1/h_i\sigma'(a))\cdot T_i$ and $T_{i+1}' := h_iT_{i+1}$ remain Toeplitz matrices as the Toeplitz structure is invariant under scaling. We will replace each identity map between adjacent Toeplitz matrices in (3) for each $i = 1,\dots,r_j-1$; and then do this for each $j = 1,\dots,k-1$. By (4), the resulting map is a $\sigma$-activated neural network with all weight matrices Toeplitz. We will denote this neural network by $\nu$.

It remains to choose the $h_i$, or more accurately the $h_{ij}$ since we have earlier dropped the index $j$ to simplify notation, in a way that

$$\sup_{x\in\Omega}\|f(x)-\nu(x)\| \le \varepsilon.$$

Given that $\sup_{x\in\Omega}\|f(x)-\varphi(x)\|<\varepsilon/2$, it suffices to show

$$\sup_{x\in\Omega}\|\varphi(x)-\nu(x)\|\le\frac{\varepsilon}{2}. \tag{5}$$

There is no loss of generality but a great gain in notational simplicity in assuming that $r_j=2$ for $j=1,\dots,k$ and $k=2$, i.e.,

$$\varphi(x)=T_1^{(2)}T_2^{(2)}\sigma(T_1^{(1)}T_2^{(1)}x+c_1)+c_2,$$
$$\nu(x)=T_1^{(2)}\rho_{h_2}T_2^{(2)}\sigma(T_1^{(1)}\rho_{h_1}T_2^{(1)}x+c_1)+c_2.$$

The reasoning is identical for the general case by repeating the argument for the $k=2=r_1=r_2$ case. Now set

$$\psi(x):=T_1^{(2)}T_2^{(2)}\sigma(T_1^{(1)}\rho_{h_1}T_2^{(1)}x+c_1)+c_2.$$

We will first show that there exists $h_1\neq 0$, such that

$$\sup_{x\in\Omega}\|\varphi(x)-\psi(x)\|\le\frac{\varepsilon}{4}. \tag{6}$$

Then we will prove that for the given $h_1$, there exists $h_2\neq 0$ such that

$$\sup_{x\in\Omega}\|\nu(x)-\psi(x)\|\le\frac{\varepsilon}{4}.$$

By our assumption, $\sigma$ is uniformly continuous on $\mathbb{R}$. So there exists $\eta>0$ such that

$$|\sigma(a)-\sigma(b)|\le\frac{\varepsilon}{4\sqrt{n}\|T_1^{(1)}\|\|T_2^{(1)}\|}$$

for any $a,b\in\mathbb{R}$ with $|a-b|\le\eta$. If we could choose $h_1\neq 0$ so that

$$\sup_{x\in\Omega}\|T_1^{(1)}T_2^{(1)}x-T_1^{(1)}\rho_{h_1}T_2^{(1)}x\|\le\eta, \tag{7}$$

then (6) would follow. Note that the $\sqrt{n}$ factor is necessary as $\sigma$ is applied coordinatewise to an $n$-dimensional vector.

Since $\Omega$ is compact, so is $\Omega_1:=\{T_2^{(1)}x:x\in\Omega\}$. Applying Lemma 2.3 to $\Omega_1$ with $\eta/\|T_1^{(1)}\|$, we obtain $h_1\neq 0$ with

$$\sup_{y\in\Omega_1}\|\rho_{h_1}(y)-y\|\le\frac{\eta}{\|T_1^{(1)}\|}$$

and thus

$$\sup_{x\in\Omega}\|T_1^{(1)}T_2^{(1)}x-T_1^{(1)}\rho_{h_1}T_2^{(1)}x\|\le\|T_1^{(1)}\|\sup_{y\in\Omega_1}\|\rho_{h_1}(y)-y\|\le\eta.$$

Next set $\Omega_2:=\{T_2^{(2)}\sigma(T_1^{(1)}\rho_{h_1}T_2^{(2)}x+c_1):x\in\Omega\}$, which is again compact. Applying Lemma 2.3 to $\Omega_2$ with $\varepsilon/4\|T_1^{(2)}\|$, we obtain $h_2\neq 0$ with

$$\sup_{y\in\Omega_2}\|\rho_{h_2}(y)-y\|\le\frac{\varepsilon}{4\|T_1^{(2)}\|}.$$

Hence

$$\sup_{x\in\Omega}\|\nu(x)-\psi(x)\|\le\|T_1^{(1)}\|\sup_{y\in\Omega_2}\|\rho_{h_2}(y)-y\|\le\frac{\varepsilon}{4},$$

which together with (6) gives us (5) as required.

To summarize the argument, if

$$\varphi(x)=T_1^{(2)}T_2^{(2)}\sigma(T_1^{(1)}T_2^{(1)}x+c_1)+c_2$$

approximates $f$ to arbitrary accuracy, then we may choose $h_1$ and $h_2$ so that

$$\nu(x)=T_1^{(2)}\rho_{h_2}T_2^{(2)}\sigma(T_1^{(1)}\rho_{h_1}T_2^{(1)}x+c_1)+c_2$$

approximates $f$ to arbitrary accuracy and $\nu$ has all weight matrices Toeplitz. For general $k$ and $r_1,\dots,r_k$, we may similarly determine a finite sequence of $h_1,h_2,h_3,\dots$ successively and insert a copy of $\rho_{h_i}$ between each pair of Toeplitz matrices while maintaining the approximation error within $\varepsilon$. As a reminder, the inserted copy of $\rho_{h_i}$ results in a $\sigma$-activation with a bias as in (1).

Furthermore, in the above proof, the only property of Toeplitz matrix we have used is that the Toeplitz structure is preserved under multiplication by any scalar. This scaling invariance also holds true for Hankel matrices and triangular matrices. Consequently, the same arguments apply verbatim if we had used a Hankel decomposition [30, Equation 2]

$$B_j = H_1^{(j)} H_2^{(j)} \cdots H_{r_j}^{(j)}$$

in place of the Toeplitz decomposition in (2). Indeed our argument above extends to any decomposition of the weight matrices into a product of structured matrices whose structures are preserved under scaling.

Now there is a slight complication for the case of triangular matrices — not every matrix will have a decomposition of the form

$$B_j = L^{(j)} U^{(j)} \tag{8}$$

where $L^{(j)}$ is lower triangular and $U^{(j)}$ is upper triangular. Note that the standard LU decomposition of a matrix requires an additional permutation matrix multiplied either to the left or right [5]. Nevertheless, we could use the fact that any square matrix all of whose principal minors are invertible has a decomposition of the form (8), and since such matrices are dense in $\mathbb{R}^{n\times n}$, any matrix has an LU *approximation* to arbitrary accuracy.

For the rectangular weight matrices in the first and last layers, we note that they can be treated much in the same way as we did in the Toeplitz case. If $B$ is an $m \times n$ matrix and $m > n$, then write

$$B = [B, \mathbb{0}_{m\times(m-n)}] \begin{bmatrix} I_n \\ \mathbb{0}_{(m-n)\times n} \end{bmatrix}.$$

Since $[B, \mathbb{0}_{m\times(m-n)}]$ is an $m \times m$ square matrix, it has an approximation $[B, \mathbb{0}_{m\times(m-n)}] \approx LU$ to arbitrary accuracy and therefore $B \approx LU'$ to arbitrary accuracy with $U' = U \begin{bmatrix} I_n \\ \mathbb{0}_{n\times(m-n)} \end{bmatrix}$. The argument for $m > n$ is similar. In short, LU-decomposable matrices are also dense in $\mathbb{R}^{m\times n}$.

There is also an alternative approach by way of a little-known result of Nagarajan et al. [24]: Any matrix in $\mathbb{R}^{n\times n}$ can always be decomposed into a product of *three* triangular matrices

$$B_j = L_1^{(j)} U^{(j)} L_2^{(j)}.$$

Note that this result may also be applied to the transpose of a matrix. So the conclusion is that any square matrix has an LUL decomposition and a ULU decomposition. The required result then follows from applying ULU decompositions to weight matrices in the odd layers and LUL decompositions to weight matrices in the even layers, adjusting for rectangular weight matrices with the argument in the previous paragraph. For example, for a neural network of the form

$$B_2 \sigma(B_1 x + c),$$

we decompose it into

$$L_1^{(2)} U^{(2)} L_2^{(2)} \sigma(U_1^{(1)} L^{(1)} U_2^{(1)} x + c)$$

and insert an appropriate activation between every successive factor as in the Toeplitz case to obtain an arbitrary accuracy approximation. □

Note that the neural network $\nu$ constructed in the proof of Theorem 2.4 has fixed width $m+n+1$ as in Theorem 1.2 but a departure from Theorem 1.2 is that $\sigma$ has to be uniformly continuous and not just continuous. Nevertheless, almost all common activations like ReLU, sigmoid, hyperbolic tangent, leaky ReLU, etc, meet this requirement.

It is perhaps also worth highlighting that the proof of Theorem 2.4 would extend to any decomposition of weight matrices into a product of structured matrices whose structures are preserved under multiplication by scalars.

A particularly interesting implication of the proof of Theorem 2.4 is that *fixed width* convolutional neural networks has the universal approximation property. While Zhou [32] has also obtained a universal approximation theorem for convolutional neural networks, it requires arbitrary width. Our version below requires a width of at most $m + n + 1$ and, as will be evident from the proof, holds regardless of how the convolutional layers and fully connected layers in the network are ordered.

**Corollary 2.5** *(Universal approximation theory for convolutional neural network). Let $\Omega \subseteq \mathbb{R}^n$ be compact, $\sigma : \mathbb{R} \to \mathbb{R}$ be any uniformly continuous nonpolynomial function which is continuously differentiable at at least one point, with nonzero derivative at that point. Then for any function $f \in C(\mathbb{R}^n, \mathbb{R}^m)$ and any $\varepsilon > 0$, there exists a deep convolutional neural network $\nu : \mathbb{R}^n \to \mathbb{R}^m$ with width $m + n + 1$ such that*

$$\sup_{x\in\Omega} \|f(x) - \nu(x)\| < \varepsilon.$$

**Proof.** Recall that a convolutional neural network is one that consists of several convolutional layers at the beginning and fully-connected layers consequently, as defined at the end of Section 1.2. Observe that in the proof of Theorem 2.4, there is no need to make every layer Toeplitz — we could replace any layer with a few Toeplitz layers or choose to keep it as is with general weight matrices while preserving the $\varepsilon$-approximation. So there is a $k$-layer neural network $g$ with first $k'$ layers Toeplitz and remaining $k - k'$ layers general such that $\sup_{x\in\Omega} \|f(x) - g(x)\| < \varepsilon$. Let $T \in \mathbb{R}^{s\times t}$ be Toeplitz, i.e.,

$$T = \begin{bmatrix} a_0 & a_{-1} & \cdots & a_{1-t} \\ a_1 & a_0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & a_{-1} \\ a_{s-t-1} & \ddots & \ddots & a_0 \\ a_{s-t} & \ddots & \ddots & a_1 \\ a_{s-t+1} & \ddots & \ddots & \vdots \\ \vdots & \ddots & a_{s-t} & a_{s-t-1} \\ a_{s-1} & \cdots & a_{s-t+1} & a_{s-t} \end{bmatrix}$$

when $s \geq t$ and

$$T = \begin{bmatrix} a_0 & a_{-1} & \cdots & a_{s-t+1} & a_{s-t} & a_{s-t-1} & \cdots & a_{1-t} \\ a_1 & a_0 & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & a_{s-t-1} \\ a_{s-1} & \cdots & a_1 & a_0 & a_{-1} & \cdots & a_{s-t+1} & a_{s-t} \end{bmatrix}$$

when $t \geq s$. Define the kernel $\kappa = (a_{1-t}, \dots, a_0, \dots, a_{s-1}) \in \mathbb{R}^{s+t-1}$. Since its dimension $s+t-1$ is larger than the dimension of an input $x \in \mathbb{R}^t$, the convolution of $\kappa$ and $x$ is taken in the sense of [15], i.e., with zero-padding that adds $s-1$ zeros on each side of $x$ to obtain an input $\widetilde{x} = (0, \dots, 0, x_1, \dots, x_t, 0, \dots, 0) \in \mathbb{R}^{t+2s-2}$ of the appropriate dimension. Regardless of whether $s \geq t$ or $t \geq s$, we have

$$\kappa * \widetilde{x} = \begin{bmatrix} a_{s-1} & \cdots & a_1 & a_0 & \cdots & a_{1-t} & 0 & \cdots & 0 \\ 0 & a_{s-1} & \cdots & a_1 & a_0 & \cdots & a_{1-t} & \cdots & 0 \\ & \ddots & \ddots & & \ddots & \ddots & & \ddots & \vdots \\ 0 & \cdots & 0 & a_{s-1} & \cdots & a_1 & a_0 & \cdots & a_{1-t} \end{bmatrix} \begin{bmatrix} 0 \\ \vdots \\ 0 \\ x_1 \\ \vdots \\ x_t \\ 0 \\ \vdots \\ 0 \end{bmatrix} = Tx.$$

Note in particular that the output $\kappa * \widetilde{x} \in \mathbb{R}^s$. A layer with $T$ as weight matrix is therefore equivalent to a convolutional layer with kernel $\kappa$ and zero-padding. By repeating this in every Toeplitz layer in $g$, we transform it into a convolutional neural network $\nu$ with $k'$ convolutional layers and $k - k'$ fully connected layer. □

The requirement that the activation function $\sigma$ be nonpolynomial in Theorem 2.4 and Corollary 2.5 may be replaced instead by a polynomial of degree at least two. The caveat is that the width of $\nu$ would have to be increased to $m+n+2$. The proof remains the same, but instead of Theorem 1.2, the proofs will rely on Proposition 4.11 in [14].

## 3. Training cost analysis

Here we perform a basic estimate of how much savings one may expect from imposing an LU or Toeplitz/Hankel structure on a neural network. The reduction in weight parameters is the most obvious advantage: an $m \times n$ upper triangular matrix requires $(n+1)n/2$ parameters if $m \geq n$ and $(2n-m+1)m/2$ if $m < n$; an $m \times n$ lower triangular matrix requires $(2m-n+1)n/2$ parameters if $m \geq n$ and $(m+1)m/2$ if $m < n$; an $m \times n$ Toeplitz or Hankel matrix requires just $m+n-1$ parameters. However, there is also a slightly less obvious advantage that we will discuss next.

The standard basic procedure in training a neural network involves a loss function $\ell$ on the output of the network. Common examples include cross-entropy loss, mean squared error loss, mean absolute error loss, negative log-likelihood loss, etc. We calculate the gradient of $\ell$ under each weight parameter, and then update each parameter with the corresponding gradient scaled by a learning rate. The training process comprises two parts, forward propagation and backward propagation. In forward propagation, the neural network is evaluated to produce the output from the input. The computational cost is dominated by the matrix-vector multiplication in each layer:

$$y_i = A_i z_i + b_i, \qquad z_{i+1} = \sigma(y_i)$$

In backward propagation, we calculate the gradient of each parameter with chain rule. In the $i$th layer, the gradient is calculated from

$$\nabla_{z_i} \ell = A_i^{\mathsf{T}} \nabla_{y_i} \ell, \qquad \nabla_{A_i} \ell = (\nabla_{y_i} \ell) \otimes z_i,$$

where $\otimes$ denotes outer product. Again, the computational cost is dominated by matrix-vector multiplication in each layer.

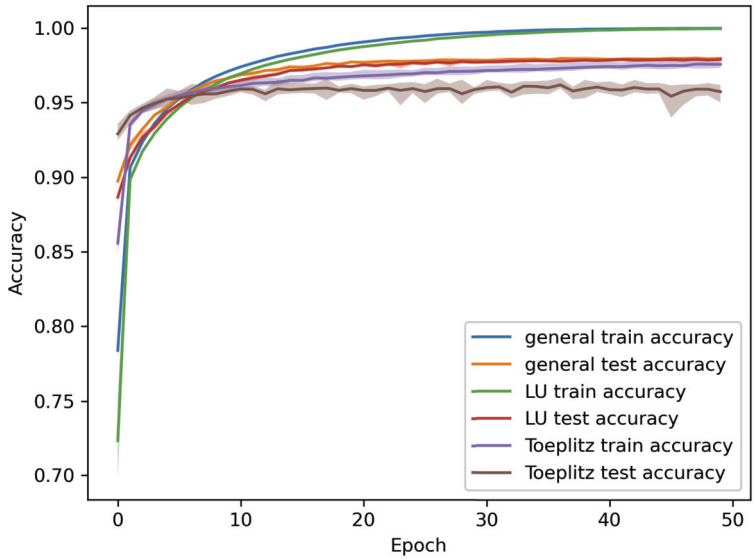


**Fig. 1.** Accuracy on MNIST. (For interpretation of the colors in the figures, the reader is referred to the web version of this article.)

Given that training cost ultimately boils down to matrix-vector multiplications, we expect massive savings by exploiting such algorithms for structured matrices, particularly in the Toeplitz or Hankel cases, as these matrix-vector products can be computed in $O(n \log n)$ complexity, compared to the usual $O(n^2)$ for general matrices. But even triangular matrices would immediately halve the cost of training.

## 4. Experiments

We have conducted extensive experiments to demonstrate that neural networks with structured weight matrices such as those discussed in this article are almost as accurate as general ones. For a fair comparison, in each experiment we fixed the width and depth of the neural networks, changing only the type of weight matrices used, whether general (i.e., no structure), triangular, Toeplitz, or Hankel. In particular, all weight matrices have same dimensions, differing only in their structures or lack therefore. We have also taken care to avoid over-fitting in all our experiments, to ensure that we are not comparing one overfitted neural network with another. One telling sign of over-fitting is poor test accuracy, but in all our experiments, test accuracy is reasonably high.

We performed our experiments with three common data sets: MNIST comprises a training set of 60,000 and a test set of 10,000 handwritten digits. CIFAR-10 comprises 60,000 $32 \times 32$ color images in 10 classes, with 6,000 images per class, divided into a training set of 50,000 and a test set of 10,000. WikiText-2 is a collection of over 100 million tokens extracted from verified 'Good' and 'Featured' articles on Wikipedia.

We used our neural networks in three different contexts: as *multilayer perceptrons*, i.e., the classic feed-forward neural network with fully connected layers; as *convolutional neural networks* that have convolutional, pooling, and fully-connected layers as in LeCun et al. [18]; and as *transformers*, a widely-used architecture based on attention mechanisms [29].

### 4.1. MNIST and multilayer perceptron:

For an image classification task with MNIST, we compare a three-layer multilayer perceptron with three general weight matrices against one where the three weight matrices are upper, lower, and upper triangular respectively; and another where all three weight matrices are Toeplitz. We use a cross entropy loss, set learning rate to 0.01, batch size to 20, and trained for 50 epochs. The mean, minimum, and maximum accuracy of each epoch over five runs are reported in Fig. 1. Our results show that the LU neural network has similar performance to the general neural network on both training accuracy and test accuracy. While the Toeplitz neural network sees poorer performance, its test accuracy, at greater than 95%, is within acceptable standards.

### 4.2. CIFAR-10 and convolutional neural networks:

For another image classification task with CIFAR-10, we compared a three-fully-connected-layer AlexNet [16] with three general weight matrices to one with three triangular weight matrices and another with three Toeplitz weight matrices. We set the learning rate at 0.01, batch size at 32, and trained for 100 epochs. The results are in Fig. 2. In this case, we see no significant difference in the performance — LU AlexNet and Toeplitz AlexNet do just as well as the usual AlexNet.

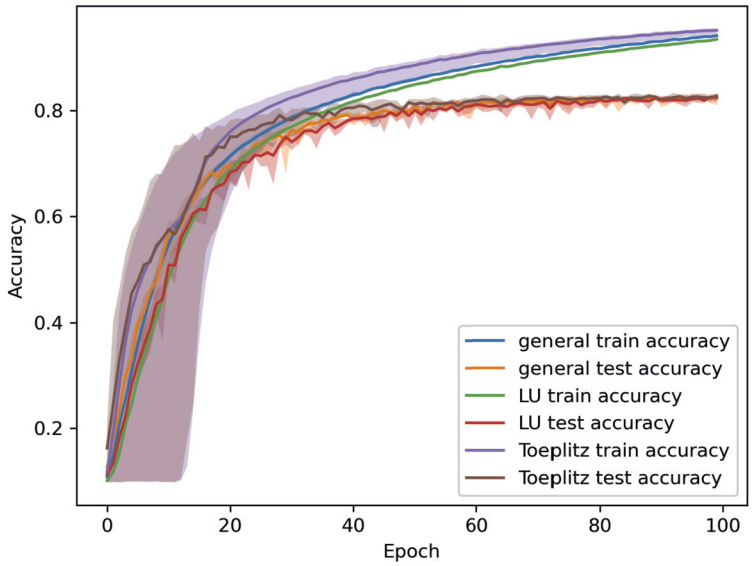


**Fig. 2.** Accuracy on CIFAR-10.

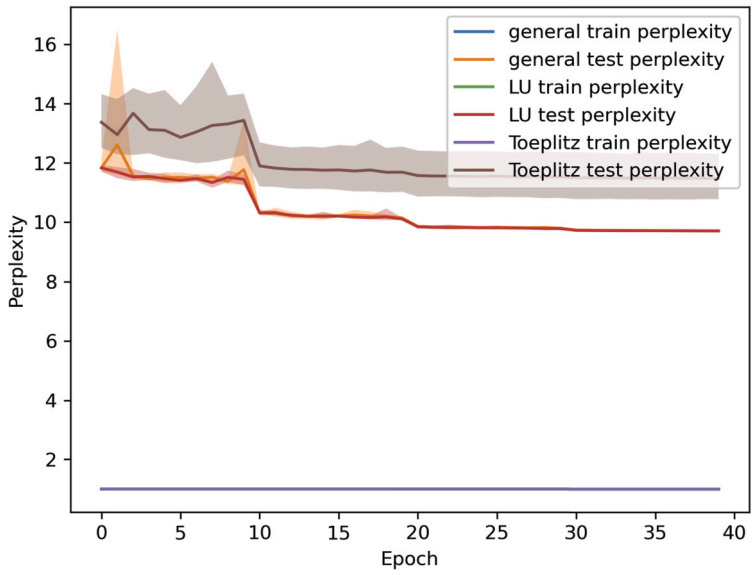


**Fig. 3.** Perplexity on Wiki-Text 2.

### *4.3. WikiText and transformer:*

We use a transformer with a two-head attention structure for a language modeling task with WikiText-2. As before, we compare three versions of the transformer where the fully-connected layers are either general, LU, or Toeplitz neural networks. We use a batch size of 20, a learning rate of 5, decaying by 0.2 for every 10 epochs. The mean, minimum, and maximum perplexity of each epoch over five runs are reported in Fig. 3. Recall that perplexity is the exponential of cross entropy loss, and thus a lower value represents a better result. Here the LU transformer performs as well as the standard transformer; the Toeplitz transformer, while slightly less accurate, is nevertheless within acceptable standards.

## 5. Conclusion

Our results here may be viewed as a first step towards extending the standard matrix decompositions — widely regarded as one of the top ten algorithms of the 20th century [27] — from linear maps to continuous maps. Viewed in this light, there are many open questions: Is there a reasonable way to extend QR decomposition or singular value decomposition in a manner similar to what we did for LU and Toeplitz decompositions? Could one compute such decompositions in a principled way like their linear counterpart

as opposed to fitting them with data? Can one design neuromorphic chips with lower energy cost or with lower gate complexity by exploiting such decompositions?

## Data availability

Data will be made available on request.

## Acknowledgments

This work is partially supported by the DARPA grant HR00112190040, the NSF grants DMS 1854831 and ECCS 2216912, and a Vannevar Bush Faculty Fellowship ONR N000142312863.

## References

[1] G. Cybenko, Approximation by superpositions of a sigmoidal function, Math. Control Signals Syst. 2 (4) (1989) 303–314.
[2] R. Eldan, O. Shamir, The power of depth for feedforward neural networks, in: Conference on Learning Theory, PMLR, 2016, pp. 907–940.
[3] J. Frankle, M. Carbin, The lottery ticket hypothesis: finding sparse, trainable neural networks, in: International Conference on Learning Representations, 2019.
[4] J. Frankle, G.K. Dziugaite, D. Roy, M. Carbin, Linear mode connectivity and the lottery ticket hypothesis, in: International Conference on Machine Learning, PMLR, 2020, pp. 3259–3269.
[5] G.H. Golub, C.F. Van Loan, Matrix Computations, third edition, Johns Hopkins Studies in the Mathematical Sciences, Johns Hopkins University Press, Baltimore, MD, 1996.
[6] S. Han, J. Pool, J. Tran, W. Dally, Learning both weights and connections for efficient neural network, Adv. Neural Inf. Process. Syst. 28 (2015).
[7] B. Hanin, M. Sellke, Approximating continuous functions by relu nets of minimal width, arXiv:1710.11278, 2017.
[8] B. Hassibi, D. Stork, Second order derivatives for network pruning: optimal brain surgeon, Adv. Neural Inf. Process. Syst. 5 (1992).
[9] K. Hornik, Approximation capabilities of multilayer feedforward networks, Neural Netw. 4 (2) (1991) 251–257.
[10] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural Netw. 2 (5) (1989) 359–366.
[11] T. Inoue, H. Tokura, K. Nakano, Y. Ito, Efficient triangular matrix vector multiplication on the gpu, in: International Conference on Parallel Processing and Applied Mathematics, Springer, 2019, pp. 493–504.
[12] J. Johnson, Deep, skinny neural networks are not universal approximators, in: International Conference on Learning Representations, 2019.
[13] V.I. Kelefouras, A.S. Kritikakou, K. Siourounis, C.E. Goutis, A methodology for speeding up mvm for regular, Toeplitz and bisymmetric Toeplitz matrices, J. Signal Process. Syst. 77 (3) (2014) 241–255.
[14] P. Kidger, T. Lyons, Universal approximation with deep narrow networks, in: Conference on Learning Theory, PMLR, 2020, pp. 2306–2327.
[15] S. Kiranyaz, O. Avci, O. Abdeljaber, T. Ince, M. Gabbouj, D.J. Inman, 1D convolutional neural networks and applications: a survey, Mech. Syst. Signal Process. 151 (2021) 107398.
[16] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, Commun. ACM 60 (6) (2017) 84–90.
[17] Y. LeCun, J. Denker, S. Solla, Optimal brain damage, Adv. Neural Inf. Process. Syst. 2 (1989).
[18] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.
[19] H. Li, A. Kadav, I. Durdanovic, H. Samet, H.P. Graf, Pruning filters for efficient convnets, arXiv:1608.08710, 2016.
[20] H. Lin, S. Jegelka, Resnet with one-neuron hidden layers is a universal approximator, Adv. Neural Inf. Process. Syst. 31 (2018).
[21] Z. Lu, H. Pu, F. Wang, Z. Hu, L. Wang, The expressive power of neural networks: a view from the width, Adv. Neural Inf. Process. Syst. 30 (2017).
[22] E. Malach, G. Yehudai, S. Shalev-Schwartz, O. Shamir, Proving the lottery ticket hypothesis: pruning is all you need, in: International Conference on Machine Learning, PMLR, 2020, pp. 6682–6691.
[23] A. Morcos, H. Yu, M. Paganini, Y. Tian, One ticket to win them all: generalizing lottery ticket initializations across datasets and optimizers, Adv. Neural Inf. Process. Syst. 32 (2019).
[24] K.R. Nagarajan, M.P. Devasahayam, T. Soundararajan, Products of three triangular matrices, Linear Algebra Appl. 292 (1–3) (1999) 61–71.
[25] S. Park, C. Yun, J. Lee, J. Shin, Minimum width for universal approximation, in: International Conference on Learning Representations, 2021.
[26] A. Pinkus, Approximation theory of the mlp model in neural networks, Acta Numer. 8 (1999) 143–195.
[27] G.W. Stewart, The decompositional approach to matrix computation, Comput. Sci. Eng. 2 (1) (2000) 50–59.
[28] M. Telgarsky, Benefits of depth in neural networks, in: Conference on Learning Theory, PMLR, 2016, pp. 1517–1539.
[29] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, Adv. Neural Inf. Process. Syst. 30 (2017).
[30] K. Ye, L.-H. Lim, Every matrix is a product of Toeplitz matrices, Found. Comput. Math. 16 (3) (2016) 577–598.
[31] C. Yun, S. Sra, A. Jadbabaie, Small relu networks are powerful memorizers: a tight analysis of memorization capacity, Adv. Neural Inf. Process. Syst. 32 (2019).
[32] D.-X. Zhou, Universality of deep convolutional neural networks, Appl. Comput. Harmon. Anal. 48 (2) (2020) 787–794.