Ariel University

Deep learning methods solving complex problems

A thesis submitted in partial fulfillment of the requirements for the degree Doctor of Philosophy

By

Avigail Stekel

Type text here

This work was prepared under the supervision of Professor Amos Azaria

Submitted to the Senate of Ariel University

03/07/2023

Summary

Deep learning has emerged as a powerful approach for solving complex problems in diverse domains. Its ability to learn intricate patterns and representations directly from raw data has transformed the field of artificial intelligence. In this work, we explore the application of deep learning to address several distinct problems across different domains.

First, we investigate the Goldbach conjecture, a famous open problem in mathematics. The conjecture states that every even number greater than two can be expressed as the sum of two prime numbers. By developing a deep learning model, we aim to predict the number of Goldbach partitions for a given even number. Surprisingly, our model outperforms existing analytical estimations, without requiring prime factorization of the number. This advancement brings us closer to solving one of the most prominent open problems in the world of mathematics.

Next, we delve into the realm of privacy preservation. Our focus is on concealing personal information from images while retaining other relevant features. To achieve this, we propose a variational autoencoder (VAE) model trained on a dataset that includes labels of the information to be concealed, such as gender or age. By directly adding these labels to the VAE's sampled latent vector, we ensure that the model does not learn or access the concealed information. Our method successfully conceals private information while maintaining other image properties, as demonstrated through user studies. This approach holds promise for privacy preservation and can mitigate bias in systems that rely on image analysis.

Lastly, we explore the evolution of the Hebrew language, specifically the transition from biconsonantal (2C) etymons to triconsonantal (3C) roots. Using the BHSA corpus, a manually annotated dataset of the Hebrew Bible, and the Word2Vec method for semantic meaning representation, we investigate the hypothesis of the evolution from 2C etymons to 3C roots in biblical Hebrew. Our analysis reveals that words with different roots, likely originating from the same 2C etymons, form denser clusters compared to random word sets. These statistically significant findings strongly support the hypothesis and shed light on the historical development of Semitic morphology.

By applying deep learning techniques to these distinct problems, we demonstrate the versatility and effectiveness of this approach in tackling complex challenges. From mathematical conjectures to privacy preservation and linguistic evolution, deep learning offers a powerful framework for problem-solving and advancing our understanding across a wide range of domains.

Extensive review

The papers provided cover several research fields, including mathematics, machine learning, image processing, privacy preservation, and linguistics. Here is an extensive and up-to-date overview of each field:

Deep learning

Deep learning is a subfield of machine learning that focuses on training deep neural networks with multiple layers to learn complex patterns and representations directly from raw data. Unlike traditional machine learning approaches that rely on manual feature engineering, deep learning models automatically learn hierarchical representations of data through a series of interconnected layers.

Deep neural networks, also known as artificial neural networks, are composed of input layers, hidden layers, and an output layer. Each layer consists of multiple interconnected nodes, called neurons, which perform computations and pass information to the next layer. Deep learning models use a process called backpropagation to iteratively adjust the weights and biases of the neurons, optimizing them to minimize the difference between predicted outputs and the ground truth.

The key advantages of deep learning include:

Learning Complex Patterns: Deep learning models excel at learning intricate patterns and representations from large and unstructured datasets. They can capture both low-level features, such as edges and textures, and high-level abstract concepts, enabling them to understand and interpret complex data.

End-to-End Learning: Deep learning models can learn end-to-end mappings directly from raw input to output without requiring manual feature extraction. This reduces the need for human expertise and domain knowledge, making deep learning more accessible and applicable to a wide range of tasks.

Scalability: Deep learning models can scale effectively with large amounts of data and compute resources. They can handle massive datasets and take advantage of parallel computing on GPUs or specialized hardware like TPUs, allowing for efficient training and inference on complex models.

Transfer Learning: Deep learning models can leverage knowledge learned from one task or domain and apply it to another related task or domain. Transfer learning enables faster and more effective training by transferring learned representations or weights, reducing the need for extensive labelled data.

Wide Range of Applications: Deep learning has shown exceptional performance across various domains, including computer vision, natural language processing, speech

recognition, recommendation systems, and autonomous driving. It has achieved groundbreaking results in tasks such as image classification, object detection, machine translation, and voice synthesis.

Despite its successes, deep learning also presents challenges, including the need for large labelled datasets, computational resources, and potential overfitting. Researchers are continuously exploring techniques to address these challenges, such as data augmentation, regularization, and model architecture improvements.

Deep learning has revolutionized many fields by pushing the boundaries of what is possible with machine learning. It has led to significant advancements in technology and has the potential to drive innovation in areas like healthcare, finance, autonomous systems, and scientific research. As research in deep learning continues, it promises to unlock new capabilities, improve decision-making processes, and shape the future of artificial intelligence.

Mathematics:

The field of mathematics encompasses a wide range of topics, including number theory, which deals with properties and relationships of numbers. The Goldbach conjecture, mentioned in the first two texts, is a famous unsolved problem in number theory. It states that every even number greater than two can be expressed as the sum of two prime numbers. Despite being proposed over two centuries ago, the conjecture remains unproven.

Predicting on a new region:

When predicting using a Deep Neural Network (DNN) on a new region, there are several challenges that can arise. One significant issue is the problem of generalization. DNNs are trained on a specific dataset, and their performance may degrade when applied to data from a different region with distinct characteristics. This is known as the problem of domain adaptation or transfer learning. The differences in language, culture, or context between regions can lead to variations in the data distribution, making it difficult for the DNN to generalize well.

Another challenge is the availability of labeled data for the new region. DNNs typically require large amounts of labeled data to train effectively. If labeled data for the new region is scarce or unavailable, it becomes challenging to adapt the DNN to make accurate predictions. In such cases, techniques like unsupervised or semi-supervised learning, where the model leverages unlabeled data or limited labeled data, respectively, may be explored.

Furthermore, the linguistic nuances and variations across different regions can impact the performance of DNNs. The model may struggle with understanding regional dialects, slang, or specific cultural references, leading to inaccurate predictions. Addressing this challenge may require region-specific data augmentation techniques, incorporating regional lexicons or leveraging pre-training models that are specifically trained on diverse regional data.

In summary, when using DNNs for prediction in a new region, challenges related to generalization, availability of labeled data, and regional linguistic variations need to be carefully considered and addressed to ensure the model performs effectively in the new context.

Image Processing and Privacy Preservation:

Privacy preservation refers to the protection of sensitive and personally identifiable information (PII) from unauthorized access, use, or disclosure. With the increasing collection, storage, and analysis of vast amounts of data in various domains, privacy preservation has become a critical concern to ensure the confidentiality and security of individuals' personal information. There are several important aspects and techniques related to privacy preservation:

Anonymization: Anonymization techniques aim to remove or obfuscate personally identifiable information from datasets. This can be achieved through methods such as data aggregation, generalization, or suppression of identifying attributes. Anonymized datasets protect the privacy of individuals by preventing the direct identification of specific individuals from the data.

Differential Privacy: Differential privacy is a mathematical framework that provides a rigorous privacy guarantee for data analysis. It ensures that the presence or absence of an individual's data does not significantly affect the output of a computation or analysis, thereby preserving privacy. Differential privacy techniques add noise or perturbation to the data to achieve privacy guarantees while maintaining data utility.

Encryption: Encryption is a widely used technique to protect data privacy. It involves encoding data in a way that can only be decrypted by authorized parties with the appropriate decryption key. Encryption ensures that even if the data is intercepted or accessed without authorization, it remains unreadable and confidential.

Secure Multi-Party Computation (MPC): Secure MPC allows multiple parties to jointly compute a function or perform analysis on their combined datasets without disclosing their individual data to each other. It ensures that each party's data remains private throughout the computation process, enabling collaboration while preserving privacy.

Federated Learning: Federated learning is a distributed machine learning approach that trains models on decentralized data sources while keeping the data itself on local devices or servers. This approach allows for collaborative model training without the need to share the raw data, thus preserving data privacy.

Homomorphic Encryption: Homomorphic encryption is a cryptographic technique that enables computations to be performed directly on encrypted data without decrypting it. This allows for secure data processing and analysis while maintaining the confidentiality of the underlying data.

Privacy-Preserving Data Publishing: Privacy-preserving data publishing methods aim to release data for public use while minimizing the risk of re-identification. Techniques like k-anonymity, l-diversity, and t-closeness provide privacy guarantees by modifying or adding noise to the published data to protect individuals' privacy.

Privacy preservation is crucial in various domains, including healthcare, finance, telecommunications, and e-commerce. It ensures compliance with privacy regulations, builds trust among individuals, and mitigates the risks of data breaches and misuse. As technology advances, research and development in privacy-preserving techniques continue to evolve to address emerging privacy challenges and strike a balance between data utility and individual privacy rights.

It is important for organizations and individuals to prioritize privacy preservation by implementing appropriate technical and organizational measures, adopting privacy-by-design principles, and staying informed about evolving privacy regulations and best practices.

Image processing involves manipulating and analyzing images to enhance or extract useful information. Privacy preservation in image processing refers to techniques that aim to conceal or remove sensitive or personal information from images while retaining other relevant information. The third text describes a method for concealing personal information, such as gender, age, and ethnicity, from images using a variational autoencoder (VAE). This approach allows for privacy-preserving image analysis and reduces bias by removing sensitive information while preserving other properties like smile, hairstyle, and brightness. Such techniques can be valuable for privacy-conscious applications and mitigating biases in machine learning systems.

Linguistics and Historical Evolution:

Linguistics is the scientific study of language and its structure, including how languages change over time. The final text discusses the hypothesis of historical evolution in Semitic morphology from biconsonantal (2C) etymons to triconsonantal (3C) roots. Semitic languages, including Hebrew and Arabic, exhibit patterns where three-consonant roots form the basis of word formation. The study utilizes a manually annotated corpus of the Hebrew Bible and employs techniques like Word2Vec, a word representation method, to explore the hypothesis. The findings support the idea that words with different roots may have originated from the same two-consonant etymons, providing insights into the historical development of Semitic languages.

Overall, these research fields showcase the interdisciplinary nature of scientific inquiry, with mathematics, machine learning, image processing, privacy preservation, and linguistics intersecting to address important problems and advance our understanding in various domains. Ongoing research in these fields continues to push the boundaries of knowledge and drive innovation in numerous applications.

The provided papers discuss various topics in mathematics and machine learning. The first paper focuses on the Goldbach conjecture, which states that every even number greater than two can be expressed as the sum of two prime numbers. We present a deep learning model that predicts the number of Goldbach partitions for a given even number. The model outperforms existing estimations and does not require prime factorization. We believe that this approach brings us closer to solving this famous open problem.

The second paper discusses the development of a model to predict Goldbach's function, which determines the number of ways an even number can be expressed as an unordered sum of two prime numbers. We initially use a simple multilayer perceptron but find that its performance deteriorates when tested on larger numbers. To overcome this, we introduce two novel deep learning architectures that significantly outperform the initial model. We also incorporate an analytically derived estimation to improve the models' performance.

The third paper introduces a learning model that can conceal personal information, such as gender, age, and ethnicity, from an image while preserving other information like smile, hair-style, and brightness. The model, a variational autoencoder (VAE), is trained on a dataset that includes labels of the information to be concealed. The VAE avoids learning the relation between the images and labels, resulting in encoded images that lack the concealed information. The method successfully conceals private information, as demonstrated by a convolutional neural network's inability to restore the original information. However, other properties of the original image remain intact in the concealed image. This architecture can be used for privacy preservation and reduce bias in systems.

The final paper explores the hypothesis of the historical evolution of Semitic morphology from biconsonantal (2C) etymons to triconsonantal (3C) roots. We use a manually annotated corpus of the Hebrew Bible and Word2Vec, a method for semantic word representation, to study this hypothesis in biblical Hebrew. We find that words with different roots that may have originated from the same 2C etymons form denser clusters than random sets of words of the same size. These statistically significant differences strongly support the hypothesis of evolution from 2C etymons to 3C roots in biblical Hebrew and other Semitic languages.

2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI)

Goldbach's Function Approximation Using Deep Learning

1st Avigail Stekel
Department of Computer Science
Ariel University
Ariel, Israel
avigail.st@gmail.com

2nd Merav Chkroun
Department of Computer Science
Ariel University
Ariel, Israel
meravgu@gmail.com

3rd Amos Azaria
Department of Computer Science
Ariel University
Ariel, Israel
Carnegie Mellon University, Pittsburgh, PA
amos.azaria@ariel.ac.il

***Abstract*—Goldbach conjecture is one of the most famous open mathematical problems. It states that every even number, bigger than two, can be presented as a sum of 2 prime numbers. In this work we present a deep learning based model that predicts the number of Goldbach partitions for a given even number. Surprisingly, our model outperforms all state-of-the-art analytically derived estimations for the number of couples, while not requiring prime factorization of the given number. We believe that building a model that can accurately predict the number of couples brings us one step closer to solving one of the world most famous open problems. To the best of our knowledge, this is the first attempt to consider machine learning based data-driven methods to approximate open mathematical problems in the field of number theory, and hope that this work will encourage such attempts.**

I. Introduction

On June 1742, the mathematician Christian Goldbach wrote a letter to his friend, Leonard Euler, describing his conjecture that states that every even integer is a sum of two prime numbers [Goldbach1742]. Since then expert mathematicians, students and many others have tried to prove this conjecture or disprove it. Even though more than two hundred and fifty years have passed, the conjecture remains open. The conjecture can be checked directly for limited sets of numbers. To this date, Goldbach's conjecture has been verified up-to 4×10^{18} [Oliveira e Silva *et al.*2014]. During the past centuries, despite no actual proof being found, there has been some important and significant progress related to this conjecture. In this paper, we focus on approximation of the Goldbach's function, denoted by $G(n)$. This function returns the number of Goldbach partitions that a given number has [Fliegel and Robertson1989]. Rephrasing Goldbach's conjecture in terms of Goldbach's function would state that the value of Goldbach's function (for all even numbers greater than 4) is greater than or equal to 1. For example, $G(100) = 6$, because $100 = 3+97 = 11+89 = 17+83 = 29+71 = 41+59 = 47+53$. See Figure 2 for an illustration of Goldbach's function on the first 100 even numbers. The plot of the Goldbach function has a form of a comet and is consequently called "Goldbach's comet" [Fliegel and Robertson1989] (See Figure 1 for Goldbach's function values for all even numbers between 4 and 4×10^6.

Several works have suggested different approximations for Goldbach's function which they have derived analytically. Unfortunately, some of these approximations are very far from the actual values taken by Goldbach's function, while others require prime factorization (prime decomposition) which is believed to be an intractable operation on large numbers.

In this paper we suggest a different approach to approximating Goldbach's function, we propose using a deep learning approach.

It may seem that deep learning is not a suitable approach for this type of problems, as the input to the approximation function is only a single number, and deep learning has shown success in situations in which the input is composed of a large vector or a matrix. We therefore propose a simple, yet powerful concept of translating the number into different bases. Surprisingly our approach outperforms current state-of-the-art approximations of Goldbach's function, resulting in an error rate of only 3.0%. Furthermore, our method does not require prime factorization of the number, which is intractable for large numbers. We believe that our model may shed light on the behavior of Goldbach's function and may bring us one step closer to proving or disproving Goldbach's conjecture. Furthermore, introducing deep learning to the field, may assist in proving or disproving other similar open mathematical problems.

II. Background

Prime and natural numbers have always aroused mathematicians' interest. In 1900 Hillbert made his famous speech at the 2nd International Congress of Mathematics held in Paris, saying there are 23 unsolved problems for mathematicians of the 20th century [Wang2002]. One of those math problems was Goldbach's conjecture.

A. Approximations for Goldbach's Function

Goldbach's Conjecture is divided into two conjectures:

1) The 'weak' Goldbach's conjecture states that 'Every odd number greater than 5 can be expressed as a sum of three primes'. For example, 11 is the sum of 3, 3 and 5. 21 is the sum of 2, 2 and 17. The weak conjecture was finally proved in 2013 [Helfgott2013].

978-1-5386-7325-6/18/$31.00 ©2018 IEEE
DOI 10.1109/WI.2018.00-46

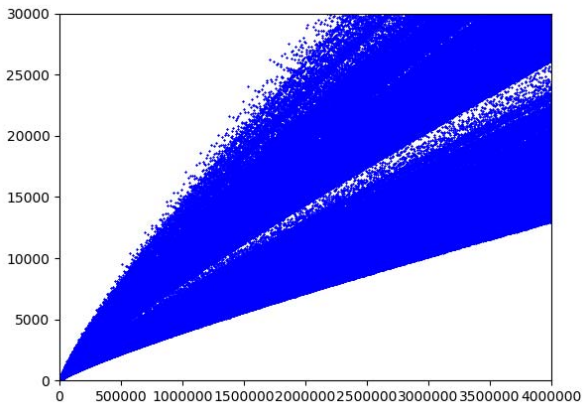

Fig. 1: Goldbach function values for all even numbers between 4 and 4×10^6. This function is sometimes referred to as Goldbach's comet, due to its shape.

2) The 'strong' Goldbach's conjectures which states that 'Every *even* integer greater than 2 is a sum of two primes'. The number 6 for example, can be presented with only one pair of prime numbers , $3+3$. However, when examining even numbers greater than 12 , there are apparently, at least two pairs of prime numbers that sum to each even number, for example, $14 = 7 + 7$ and $14 = 3 + 11$. One might assume that the greater the even number, the more different pairs it has, yet by observing different even numbers this assumption turns out to not always hold. For example, while 34 and 36 have 4 Goldbach partitions each, 38 has only 2 Goldbach partitions as is shown in Figure 2. This conjecture remains open until this date.

In this paper we focus on Goldbach's function which provides the number of Godlbach's partitions an even number has. More formally, Let $n \in \mathbb{N}$, the Goldbach's function is given by:

$$G(n) = \sum_{\{p,q\} \in \mathbb{P} \times \mathbb{P} \wedge p \leq q} \mathbb{1}\{p + q = n\} \tag{1}$$

where, $\mathbb{P}$ is the set of all prime numbers, and $\mathbb{1}$ is the indicator function that returns 1 if the expression is *true* and 0 otherwise.

Over the years there have been several attempts to find an analytic approximation to Goldbach's function. Hardy and Littlewood [Hardy and Littlewood1922] proposed the following approximation:

$$G_1(n) = 2 \cdot C_2 \frac{n}{(ln(n))^2} \prod_{p|n} (\frac{p-1}{p-2}) \tag{2}$$

where C_2 is their twin prime constant:

$$C_2 = \prod_{p \geq 3} \left(1 - \frac{1}{(p-1)^2}\right) \cong 0.6601618158 \tag{3}$$

n denotes an even number, and p denotes all the prime factors of n. While this function was originally proposed as an upper-bound, it is widely used as an approximation. Baker, suggested multiplying $G_1(n)$ by $\frac{3}{5}$ to yield a better approximation [Baker2007] (we will refer to Baker's approximation as $G_2(n)$).

Note that this approximation requires factorizing n, which is assumed to be a hard problem. Currently, best known prime factorization algorithm (GNFS) [Buhler *et al.*1993] runs in time complexity of:

$$O\left(exp\left(\left(\sqrt[3]{\frac{64}{9}} + o(1)\right) \sqrt[3]{log(n)} \sqrt[3]{(loglog(n))^2}\right)\right)$$

where n is the number being factored. Note that the input size is considered $log(n)$, since the number of digits required to represent n is $log(n)$.

To overcome this prime factorization requirement, the following approximation was proposed [Provatidis *et al.*2013]:

$$G_3(n) = \frac{n}{(ln(n))^2} \tag{4}$$

This approximation is derived from Gauss' approximation provided in 1793 for the probability of a number being prime. According to Gauss, this probability is given by:

$$f(m) = \frac{m}{ln(m)}$$

Therefore, for an even number n the following may be used as an approximation for its number of Goldbach partitions:

$$\sum_{m=3}^{n/2} \frac{m}{ln(m)} \cdot \frac{n-m}{ln(n-m)} \approx \frac{n}{2ln(n)^2}$$

Note that G_3 is monotone, and thus cannot capture the phenomenon that larger numbers may sometimes have less Goldbach partitions than smaller numbers. The following approximation, which is also monotone, was proposed by Markakis et al. in [Markakis *et al.*2012]:

$$G_4(n) = \frac{n}{(ln(n/2))^2} \tag{5}$$

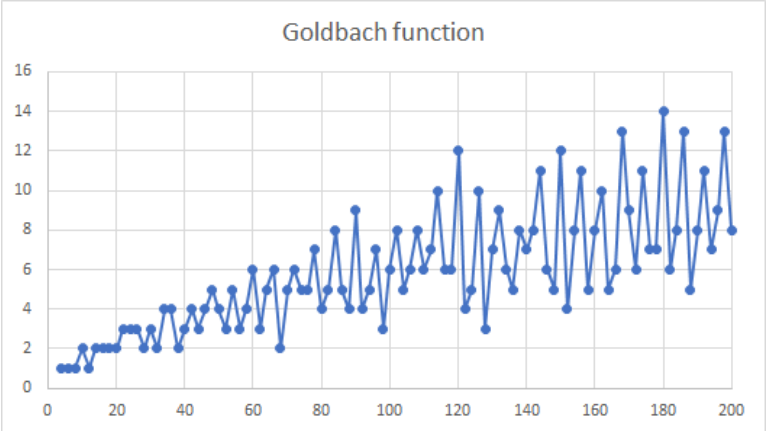

Fig. 2: The number of Goldbach partitions of a couple of even numbers

B. Related Work

In addition to attempts for finding an approximation to Goldbach's function, there have been several attempts to finding upper and lower-bound to it, that is, a function that limits the number of Goldbach partitions from above or below. The $G_1(n)$ function proposed by Hardy and Littlewood [Hardy and Littlewood1922] was originally suggested as an upper bound. One proposed lower-bound function provided by Provatidis et al. [Provatidis *et al.*2013] is:

$$2/3 * G_1(n) \tag{6}$$

This lower-bound was derived analytically, and it is shown that as n grows, the probability of it having less Goldbach partitions than the lower bound approaches 0. However, proving this lower-bound as a strict lower-bound, would imply the proof of Goldbach's conjecture, since this lower-bound is at least 1 for every even number.

Montgomery and Vaughan define another function related to Goldbach's conjecture, capturing non-Goldbach numbers, that is, numbers that cannot be written as a sum of two prime numbers [Montgomery and Vaughan1975]. Montgomery and Vaughan's function, $E(n)$, denotes all even numbers smaller than n that are not a Goldbach number. Montgomery and Vaughan prove that there exists an absolute constant $\delta > 0$ such that

$$E(N) \ll N^{1-\delta}. \tag{7}$$

There are several fields lying in the intersect of artificial intelligence and mathematical problems. Automated theorem proving [Bibel2013] is a field in which machines use various artificial intelligence based methods, such as heuristic search, in an attempt to find a proof for a given conjecture. In 1956, Newell and Simon developed the "Logic Theorist" [Newell and Simon1956]. The Logic Theorist was based on heuristic search and successfully proved 38 of the 52 theorems that appear in the second Chapter of Principia Mathematica [Whitehead and Russell1912].

The 'Automated Mathematician' (AM for short) was created by Douglas Lenat in Lisp. [Lenat1977]. AM used heuristic search to find interesting properties in mathematics. AM defined 250 various heuristics and tried to infer different mathematical properties by applying these heuristics. AM discovered the concept of natural numbers, prime numbers, it conjectured (without proof) the unique prime factorization theory and defined the concept of Goldbach partitions. Unfortunately, AM was not able to discover any "new to mankind" mathematics, and it turned out to be very hard for it to discover new heuristics. One of the statements Lenat's AM produced was the Goldbach conjecture [Larson2005]. AM was more about finding interesting problems than solving them. An improved system named EURISKO was later developed by Lenat, with an attempt to learn these heuristics by its own [Lenat1983], [Lenat and Brown1984].

Colton et al. [Colton *et al.*2000] developed an artificial intelligent system for identifying mathematical concepts, such as, types of graphs, types of groups and types of numbers. For example, their method can identify that a sequence such as 1, 4, 9, 16 etc. is a sequence of squared numbers. They state that the state-of-the-art at their time for identifying these concepts was just a data-base.

III. Deep Learning Based Goldbach's Function Approximation

In this section we present a deep learning based model to approximate Goldbach's function values.

A. Data Composition

In order to train and evaluate the different methods, we composed a data-set consisting of the number of Goldbach partitions that all even numbers from 4 to 4×10^6 have. To that end, we first computed all prime numbers at that range, and stored them as a list and as a hashmap. For each even number, n, we iterated on all prime numbers (using the list of all primes) that are smaller than or equal to $\frac{n}{2}$. For each of these prime numbers, p we test (using the hashmap) whether $n-p$ is a prime number itself. If so, we increment n's counter by one.

We shuffled the data and split it into a train set, (80% of the data; 16×10^5 numbers), a validation-set, (10% of the data; 2×10^5 numbers), and the remaining 10% was reserved for the test-set (2×10^5 numbers).

B. Model Features

From each number we extracted 42 features. We converted every number to its binary representation, ternary representation (base 3), quinary representation (base 5) and septenary representation (base 7). The time complexity of computing these base representations for a number n is $O(log(n))$. In practice we computed those representations when composing the data, so we simply incriminated the representation of the previous number by 2 for all bases. We truncated these representations and used the 10 least significant digits for each representation. The intuition behind using these different representations lies in the fact that these transformation are computationally cheap to extract and that they might allow the model to retrieve underlying information on the number. The first 4 prime numbers $(2, 3, 5, 7)$ were selected as the bases. In addition to the representations in the different bases, we added the number itself (normalized), and the logarithm of that number.

C. Model Architecture

We used a fully connected neural network as our model. We set the number of neurons to 200 on each hidden layer. We used Adam optimizer [Kingma and Ba2014], with a learning rate of 0.001. We used a mini-batch size of 1024 and trained the model for approximately 200 epochs on the data. We used early stopping [Prechelt1998], that is, we evaluated the validation set every epoch and saved the variables which obtained the lowest validation error. We varied the number of hidden layers, starting at a simple linear regression model (with no hidden layers), a model with 3 hidden layers, 5 hidden layers, and 7 hidden layers. Each of these models was trained on the training data and their performance was evaluated on the validation set. See Table I for a summary of the validation results. As can be seen in the table, the model with 5 hidden layers performed best on the validation set, and was therefore chosen as the model for our further analysis. For a given number n, the time complexity of generating the features and evaluating our model is $O(log(n))$, which is the best time complexity one could expect from an algorithm that reads the entire input (which requires $O(log(n))$ digits to represent).

	Train MSE	Validation MSE
Linear regression	960,400	1,016,064
3 hidden layers	92,933	107,223
5 hidden layers	89,764	**103,457**
7 hidden layers	88,446	105,903

TABLE I: Train and validation mean squared error (MSE) according to the number of hidden layers. We select the model with the lowest validation error (5-hidden layers).

D. Results

Table II presents the performance of our model in comparison to the formulas that appear in the literature, in terms of mean squared error (MSE), root mean squared error (RMSE) and the error rate in comparison to the number of actual pairs (that is, the RMSE divided by the mean of the number of Goldbach partitions each number in the test-set has). In addition to the analytically derived estimations, we also considered K-Nearest Neighbors (KNN) as another baseline (with the K set to 105 neighbors, as that value performed best). As can be seen in the table, our model outperformed all previous approximation attempts, achieving a new state-of-the-art approximation model. Furthermore, our model does not require factorizing the given number. Figure 3 compares the approximation of the different methods on 20 randomly sampled numbers from the test-set. As illustrated in the figure, our approach achieves the best fit to the actual points. While G_3 and G_4 follow the average value of Goldbach's function, they do not follow the ups and downs of it. G_1 does follow the ups and downs of Goldbach's function but keeps a gap all long the plot. While this gap is corrected nicely by G_2, G_2 (as well as G_1) requires prime factorization to be computed.

Using our trained model, we tried to articulate what a number violating Goldbach's conjecture may look like. We used a hill climbing search method on the base representations of the input features to the model. We set the number itself to 10^6 and its log value accordingly. Iteratively, we traversed each of the digits of each of the base representations, searching for the digit value that minimizes our model's prediction. We repeated this process until no digit was changed. Table III presents the base representations of a hypothetical number found by the search method. According to our model, this number violates Goldbach's conjecture, with a prediction of -192,886 Goldbach partitions (note the negative value). This number is a factor of 14, has a remainder of 2 when divided by 3 and a remainder of 4 when divided by 5. We performed a search on numbers satisfying base 7 representation, that is, numbers of the form $m \times 7^{10} + 6 \times 7^9 + 7^8 + 6 \times 7^2 + 4 \times 7^1, m \in \mathbb{N}$, and tested whether these numbers satisfy also the other bases representations. While such numbers are likely to exist, our attempts for finding such a number have failed, and we conclude that no such number exists that is smaller than 10^{19}. Furthermore, even if we found such a number, once we plug-in the number to the model, it might predict a value larger than 0, and even if our model predicts a value less than 0, it is very well likely that our model does not perform that well when considering numbers so much larger than those it was trained on.

E. Feature Analysis

In this section we analyze the contribution each of the features has on the performance of the model. Table IV presents the performance of the model (in mean squared error) when trained without each of the following sets of features: base 2, 3, 4, 5 and 7 representation, without the number itself and without its log. As can be seen in the table, the base-3 features seem to have the greatest impact on the model,

	MSE on test	RMSE on test	Error rate
G_1 [Hardy and Littlewood1922]*	89,059,989	9437.1	87.6%
G_2 [Baker2007]*	221,437	470.6	4.4%
G_3 [Provatidis *et al.*2013]	24,902,559	4990.3	46.3%
G_4 [Markakis *et al.*2012]	22,517,117	4745.2	44.0%
KNN	22,155,849	4707.0	43.7%
Deep-learning based method	**105,100**	**324.0**	**3.0%**

TABLE II: error of the deep learning based method in comparison to the state-of-the-art approximations. Asterisk (*) denotes models that require prime factorization of the given number.

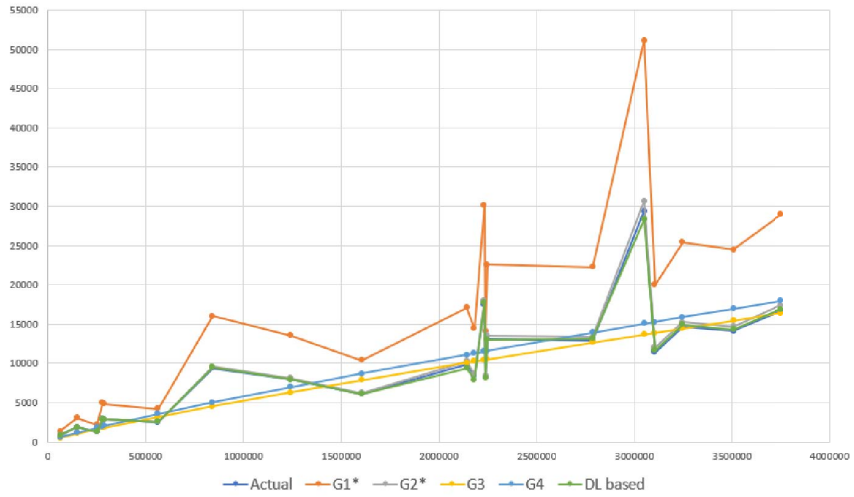

Fig. 3: Prediction of the different methods on 20 randomly picked numbers from the test-set. The asterisk (*) denotes models that require prime factorization of the given number.

Base	10 least significant digits
Base 2	0, 0, 1, 0, 1, 0, 0, 0, 0, 0
Base 3	2, 0, 2, 0, 2, 1, 2, 0, 0, 2
Base 5	0, 0, 0, 0, 0, 0, 0, 0, 1, 4
Base 7	6, 1, 0, 0, 0, 0, 0, 6, 4, 0

TABLE III: Base representations of a hypothetical even number violating Goldbach's conjecture, with our model predicting a negative value of Goldbach's partitions for it.

Features used in model	MSE
Without base 2	138,369
Without base 3	419,002
Without base 5	112,653
Without base 7	252,696
Without log	135,153
Without number	99,463
Least significant digits	391,707
Full model (all features)	89,764

TABLE IV: Mean squared error (MSE) of model trained on a subset of the features.

as removing them results with the highest error. Next in importance are the base-7 features. While, base-2 and base-5 seem to have a positive impact on the model, removing each of them separately does not harm the model's performance that much. Interestingly, the number itself turned out to be the least important feature. We also trained and evaluated the model while using only the single least significant digit of each of the bases; not surprisingly, this model did not perform well.

IV. Discussion

As stated in the introduction, Goldbach's conjecture has been verified up-to 4×10^{18}. This verification was performed by using exhaustive search. Our approximation model may allow a selective search method in which Goldbach's conjecture can be verified only for suspicious numbers according to our model, that is, only numbers that our model predicts will have a very low number of pairs. This approach can also be used to find numbers that violate the lower-bound proposed by [Provatidis *et al.*2013]. However, such selective search may require retraining our model on data closer to the target distribution (i.e., larger numbers), and adding additional digits to the base representations.

The success of our method can be attributed, for the most part, to the base representations added as features. In our work we used based representations for the first 4 prime numbers (2, 3, 5, 7), though it is likely that adding few additional base representations with the following prime numbers (e.g. 11, 13, 17), would increase the model's accuracy. However, it is

impractical to add more than a few additional representations (adding all prime representations up to the given number would require prime factorization, which is the exact problem our method tries to avoid).

In this paper we do not attempt to use DNN for actually proving a theorem (which is a completely different field), but believe that DNN can be used to help solving open mathematical problems in different ways, the first is by finding counterexamples (e.g. using selective search, as mentioned in the paper), the second is by analyzing the features that the DNN considers when computing its values. In our example we show that using different base representations (which can be computed cheaply) is very useful for computing Goldbachs Conjecture, (as opposed to the expensive prime factorization computation which was previously known). Future advancements in explainable neural networks, might help shedding additional light on this problem when applied to our results.

One of the main requirements for machine learning based methods to perform well is that the train and test data both come from the same distribution. Therefore, if the test data comes from a different distribution (e.g. higher numbers), it is likely that our model will not perform that well. Nevertheless, while testing our model with higher numbers (between 4,000,000 and 5,000,000), our model (trained on numbers between 4 and 4,000,000) resulted in a RMSE of 655.8, which still significantly outperforms models G_3 and G_4 who obtained RMSEs of 9088.8 and 8666.3 respectively. We do not compare to G_1 and G_2 which both require prime factorization. Note that our additional baseline of KNN is impractical when the distributions of the train and test data do not match.

While deep learning has shown great success in many different fields [Lv *et al.*2015], [Cruz-Roa *et al.*2013], [Alipanahi *et al.*2015], we believe that the success shown in this paper related to an open mathematical problem in number theory, is a big step and should not be disregarded as being merely another deep learning application. Our work may lead to a new paradigm of using deep learning (or machine learning in general) to solve mathematical problems such as prime factorization, friendly numbers, finding prime twins and many similar problems, which may currently seem out of the scope of deep learning methods.

V. Conclusions

Goldbach's conjecture and Goldbach's function have remained open mathematical questions for over two and a half centuries. There have been several analytic attempts to approximate Goldbach's function, but unfortunately, these approximations either do not work well in practice or require prime factorization (prime decomposition) which is a hard problem. In this paper we present the first deep-learning based approach to approximating Goldbach's function. We show that our approach outperforms current state-of-the-art approximations while not requiring prime factorization. We believe that our results can bring us one step closer to solving one of the worlds most significant open mathematical question.

References

[Alipanahi *et al.*2015] Babak Alipanahi, Andrew Delong, Matthew T Weirauch, and Brendan J Frey. Predicting the sequence specificities of dna-and rna-binding proteins by deep learning. *Nature biotechnology*, 33(8):831, 2015.

[Baker2007] John Baker. Excel and the goldbach comet. *Spreadsheets in Education (eJSiE)*, 2(2):2, 2007.

[Bibel2013] Wolfgang Bibel. *Automated theorem proving*. Springer Science & Business Media, 2013.

[Buhler *et al.*1993] Joe P Buhler, Hendrik W Lenstra, and Carl Pomerance. Factoring integers with the number field sieve. In *The development of the number field sieve*, pages 50–94. Springer, 1993.

[Colton *et al.*2000] Simon Colton, Alan Bundy, and Toby Walsh. Automatic identification of mathematical concepts. In *ICML*, pages 183–190, 2000.

[Cruz-Roa *et al.*2013] Angel Alfonso Cruz-Roa, John Edison Arevalo Ovalle, Anant Madabhushi, and Fabio Augusto González Osorio. A deep learning architecture for image representation, visual interpretability and automated basal-cell carcinoma cancer detection. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 403–410. Springer, 2013.

[Fliegel and Robertson1989] Henry F Fliegel and Douglas S Robertson. Goldbach's comet: the numbers related to goldbach's conjecture. *Journal of Recreational Mathematics*, 21(1):1–7, 1989.

[Goldbach1742] Christian Goldbach. Letter to l. *Euler, June*, 7, 1742.

[Hardy and Littlewood1922] Godfrey H Hardy and John E Littlewood. Some problems of diophantine approximation: The lattice-points of a right-angled triangle. *Proceedings of the London Mathematical Society*, 2(1):15–36, 1922.

[Helfgott2013] Harald A Helfgott. The ternary goldbach conjecture is true. *arXiv preprint arXiv:1312.7748*, 2013.

[Kingma and Ba2014] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[Larson2005] Craig E Larson. A survey of research in automated mathematical conjecture-making. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 69:297, 2005.

[Lenat and Brown1984] Douglas B Lenat and John Seely Brown. Why am and eurisko appear to work. *Artificial intelligence*, 23(3):269–294, 1984.

[Lenat1977] Douglas B Lenat. Automated theory formation in mathematics. In *Proceedings of the 5th international joint conference on Artificial intelligence-Volume 2*, pages 833–842. Morgan Kaufmann Publishers Inc., 1977.

[Lenat1983] Douglas B Lenat. Eurisko: a program that learns new heuristics and domain concepts: the nature of heuristics iii: program design and results. *Artificial intelligence*, 21(1-2):61–98, 1983.

[Lv *et al.*2015] Yisheng Lv, Yanjie Duan, Wenwen Kang, Zhengxi Li, and Fei-Yue Wang. Traffic flow prediction with big data: a deep learning approach. *IEEE Transactions on Intelligent Transportation Systems*, 16(2):865–873, 2015.

[Markakis *et al.*2012] Emmanuel Markakis, Christopher Provatidis, and Nikiforos Markakis. Some issues on goldbach conjecture. *Number Theory*, 29, 2012.

[Montgomery and Vaughan1975] H Montgomery and R Vaughan. The exceptional set of goldbach's problem. *Acta Arithmetica*, 27(1):353–370, 1975.

[Newell and Simon1956] Allen Newell and Herbert Simon. The logic theory machine–a complex information processing system. *IRE Transactions on information theory*, 2(3):61–79, 1956.

[Oliveira e Silva *et al.*2014] Tomás Oliveira e Silva, Siegfried Herzog, and Silvio Pardi. Empirical verification of the even goldbach conjecture and computation of prime gaps up to 4 10^1. *Mathematics of Computation*, 83(288):2033–2060, 2014.

[Prechelt1998] Lutz Prechelt. Automatic early stopping using cross validation: quantifying the criteria. *Neural Networks*, 11(4):761–767, 1998.

[Provatidis *et al.*2013] Christopher Provatidis, Emmanuel Markakis, and Nikiforos Markakis. Rule of thumb bounds in goldbachs conjecture. *American Journal of Mathematical Analysis*, 1(1):8–13, 2013.

[Wang2002] Yuan Wang. *The Goldbach Conjecture*, volume 4. World scientific, 2002.

[Whitehead and Russell1912] Alfred North Whitehead and Bertrand Russell. *Principia mathematica*, volume 2. University Press, 1912.

Deep Learning Architectures for Approximating Goldbach's Function in New Regions

Avigail Stekel, Amos Azaria

Computer Science Dept., Ariel University, Israel

***Abstract*—Goldbach conjecture is one of the most famous open mathematical problems. He asserts that: Every even number greater than two is the sum of two prime numbers. The Goldbach function receives an even number and returns the number of different ways to write it as an unordered sum of two prime numbers. We developed a simple multi-layer perceptron that attempts to predict Goldbach's function. This simple model performs well when trained and tested on numbers up to 4 million. However, as expected, the model's performance significantly deteriorates when trained on smaller numbers (up to 4 million) but tested on larger numbers ($4-10$ million).**

To overcome this problem, we present two novel deep learning architectures. In these architectures we introduce two types of multiplication layers, which we believe are more appropriate for solving mathematical relations. We show that both architectures significantly outperform the simple multi-layer perceptron when trained on smaller numbers and tested on larger numbers. We further improve the performance of the deep learning architectures by using a known analytically derived estimation that is used in order to normalize the model's output.

***Index Terms*—Goldbach's function; Deep learning; Out-of-scope inference.**

I. Introduction

In June 1742, the mathematician Christian Goldbach wrote a letter to his friend, Leonard Euler, describing his conjecture that states that every even integer larger than two is a sum of two prime numbers [1]. Since then mathematicians have tried to prove this conjecture or disprove it. Even though more than two hundred and fifty years have passed, the conjecture remains open. To this date, Goldbach's conjecture has been verified manually up to 4×10^{18} [2]. During the past centuries, despite no actual proof being found, there has been some important and significant progress related to this conjecture.

A more general problem is to determine the number of different options there are for a given even number, n, to be written as a sum of two prime numbers. Each such option is called a Goldbach partition. That is, a Goldbach partition is composed of three numbers: two primes, which sum to a given even number, n. This problem is known as the *Goldbach's function*, denoted $G(n)$ [3]. Rephrasing Goldbach's conjecture in terms of Goldbach's function would state that the value of Goldbach's function (for all even numbers greater than 2) is greater than or equal to 1. For example, $G(100)=6$, because $100=3+97=11+89=17+83=29+71=41+59=47+53$ and there any other. The plot of the Goldbach's function has a form of a comet and is consequently called "Goldbach's comet" [3] (See Figure 1 for Goldbach's function values for all even numbers up to 4×10^6.

Several works ([4], [5]) have suggested different approximations of Goldbach's function, which they derived analytically. Unfortunately, some of these approximations are very far from the actual values taken by Goldbach's function, while others require prime factorization (prime decomposition), which is believed to be an intractable operation on large numbers. In this paper we suggest a different approach to approximate Goldbach's function, by utilizing unique architectures for neural networks (i.e., deep learning).

Deep learning is a hierarchical based approach to approximate complex functions, which most commonly uses neural network architectures that are composed of multiple layers. Training the network is performed by executing grading descent, which adjusts the weights of the network in order to minimize a loss function. Indeed, deep learning has shown success in many different fields [6]. However, it may seem that deep learning is not a suitable approach for this type of problems, as the input to the approximation function is only a single number, and deep learning has shown success in situations in which the input is composed of a large vector or a matrix. We therefore propose a simple, yet powerful concept of translating the number into different bases. Using our approach, a simple multi-layer perceptron performs well when trained on some numbers up to 4 million and tested on other numbers up to 4 million as we show in [7]. However, the simple multi-layer perceptron's performance significantly deteriorates when trained on smaller numbers (up-to 4 million) but tested on larger numbers (4-10 million).

To overcome the problem of the multi-layer perceptron model that does not perform well when tested on a new distribution, we present two novel deep learning architectures. In these architectures we introduce two types of multiplication layers, which we believe are more appropriate for solving mathematical relations. The first architecture includes a multiplication layer, in which some of the neurons are multiplied before serving as an input for the next layer. In the second architecture, an additional layer is included, in which all inputs are activated using a logarithmic function, and then all outputs are activated with an exponential function. This allows the multiplication of any input neurons. We show that both architectures significantly outperform the simple multi-layer perceptron when trained on smaller numbers and tested on larger numbers. We further improve the performance of the deep learning architectures by using a known analytically derived estimation (see Equation 6) that is used in order to normalize the model's output.

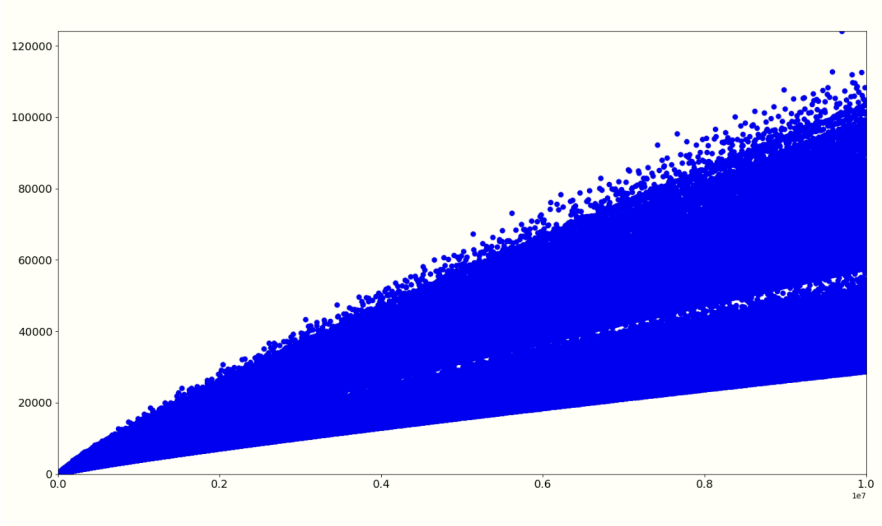

Fig. 1: Goldbach's function values for all even numbers between 4 and 10×10^6. This function is sometimes referred to as Goldbach's comet, due to its shape.

II. Background

Prime and natural numbers have always aroused mathematicians' interest. In 1900 Hillbert made his famous speech at the 2nd International Congress of Mathematics held in Paris, saying there are 23 unsolved problems for mathematicians of the 20th century [8]. One of those math problems was Goldbach's conjecture.

A. Approximations of Goldbach's Function

Goldbach's Conjecture is divided into two conjectures:

1) The 'weak' Goldbach's conjecture states that 'every odd number greater than 5 can be expressed as a sum of three primes'. For example, 11 is the sum of 3, 3 and 5. 21 is the sum of 2, 2 and 17. The weak conjecture was finally proved in 2013 [9].
2) The 'strong' Goldbach's conjecture, which states that 'every *even* integer greater than 2 is a sum of two primes'. The number 6 for example, can be presented with only one pair of prime numbers, $3+3$. However, when examining even numbers greater than 12 , there are apparently, at least two pairs of prime numbers that sum to each number. One might assume that the greater the even number, the more different pairs it has, yet by observing different even numbers this assumption does not always hold. For example, while 34 and 36 have 4 Goldbach partitions each, 38 has only 2 Goldbach partitions. There have been multiple attempts to make progress with respect to the strong Goldbach's conjecture, some of which were very recent [10], [11], [12]. However, this conjecture remains open to date.

In this paper we focus on Goldbach's function, which provides the number of Godlbach's partitions an even number has. More formally, let $n \in \mathbb{N}$. Then, Goldbach's function is given by:

$$G(n) = \sum_{\{p,q\} \in \mathbb{P} \times \mathbb{P} \wedge p \leq q} \mathbb{1}\{p+q=n\} \tag{1}$$

where, $\mathbb{P}$ is the set of all prime numbers, and $\mathbb{1}$ is the indicator function that returns 1 if the expression is *true* and otherwise 0.

Over the years there have been several attempts to find an analytic approximation of Goldbach's function. Hardy and Littlewood [4] proposed the following approximation:

$$G_1(n) = 2 \cdot C_2 \frac{n}{(ln(n))^2} \prod_{p|n} \left(\frac{p-1}{p-2}\right) \tag{2}$$

where n is an even number, p denotes all the prime factors of n. C_2 is a number that they refer to as the twin prime constant, which equals:

$$C_2 = \prod_{p \geq 3} \left(1 - \frac{1}{(p-1)^2}\right) \cong 0.6601618158 \tag{3}$$

where p denotes all prime numbers. We note that C_2 is called the twin prime constant because it was previously used in formula developed to estimate the number of twin primes (i.e., two primes, p_1 and p_2 such that $|p_1 - p_2| = 2$) that are smaller than a given number.

While this function was originally proposed as an upper-bound, it is widely used as an approximation. Baker suggested multiplying $G_1(n)$ by $\frac{3}{5}$ to yield a better approximation [5] (we will refer to Baker's approximation as $G_{1'}(n)$). As stated by Hardy & Littlewood [4] the $G_1(n)$ function can only be used as a good approximation when approaching the limit (i.e., for very large numbers). Therefore, Granville [13] provides the following function, which achieves a better approximation for smaller numbers:

$$G_2(n) = C_2 \int_2^{n-2} \frac{dt}{ln(t) \cdot ln(n-t)} \prod_{p|n} \left(\frac{p-1}{p-2}\right). \tag{4}$$

Granville [13] shows that $G_2'(n) = G_2(n) \cdot \left(1 - \frac{4}{\sqrt{n}} \prod_{p \geq 3} (1 - \frac{n/p}{p-2})\right)$ yields a better approximation.

Note that $G_1(n)$, $G_{1'}(n)$, $G_2(n)$ and $G_2'(n)$ require factoring n, which is assumed to be a hard problem. Currently, the

best known prime factorization algorithm (GNFS) [14] runs in a time complexity of:

$$O\left(exp\left(\left(\sqrt[3]{\frac{64}{9}}+o(1)\right)\sqrt[3]{log(n)}\sqrt[3]{(loglog(n))^2}\right)\right)$$

where n is the number factored. Note that the input size is considered $log(n)$, since the number of digits required to represent n is $log(n)$. We therefore do not consider those approximations in this paper (see [7] for a comparison of a deep learning based method with these methods).

To overcome the factorization requirement, the following approximation of Goldbach's function was proposed [15]:

$$G_3(n)=\frac{n}{(ln(n))^2} \tag{5}$$

This approximation is derived from Gauss' approximation provided in 1793 for the probability of a number being prime. According to Gauss, this probability is given by:

$$f(m)=\frac{m}{ln(m)}$$

Therefore, for an even number n the following may be used as an approximation of its number of Goldbach partitions:

$$\sum_{m=3}^{n/2}\frac{m}{ln(m)}\cdot\frac{n-m}{ln(n-m)}\approx\frac{n}{2ln(n)^2}$$

Note that G_3 is monotone, and thus cannot capture the phenomenon that larger numbers may sometimes have less Goldbach partitions than smaller numbers. The following approximation, which is also monotone, was proposed by Markakis et al. in [16]:

$$G_4(n)=\frac{n}{(ln(n/2))^2}. \tag{6}$$

We note that G_4 was shown to outperform G_3 (see [7]).

III. Deep Learning Based Architectures for Goldbach's Function Approximation

In this section we present several deep learning based models to approximate Goldbach's function values. In previous work we compared the simple deep learning based model (the Multilayer perceptron Basic Model) to other baselines when trained and tested on numbers up to $4M$ [7]. In this paper we focus on the performance of our models when trained on numbers up to $4M$, but tested on larger numbers $(4-10M)$.

A. Data Composition

In order to train and evaluate the different methods, we composed a dataset consisting of the number of Goldbach partitions that all even numbers from 4 to 4×10^6 have. To that end, we first computed all prime numbers at that range, and stored them as a list and as a hashmap. For each even number, n, we iterated on all prime numbers (using the list of all primes) that are smaller than or equal to $\frac{n}{2}$. For each of these prime numbers, p, we tested (using the hashmap) whether $n-p$ is a prime number itself. If so, we incremented n's counter by one.

We shuffled the data and split it into a training-set, (90% of the data; 18×10^5 numbers), and the remaining 10% was reserved for the $0-4M$ test-set (2×10^5 numbers).

B. Model Features

From each number we extracted the following 26 features. We converted every number to its binary representation, ternary representation (base 3), quinary representation (base 5) and septenary representation (base 7). The time complexity of computing these base representations for a number n is $O(log(n))$. In practice we computed those representations while composing the data. In order to compute the base-representations, we iteratively divided the number by the required base. We truncated the base representations and used the 6 least significant digits for each representation. The intuition behind using these different representations lies in the fact that these transformations are computationally cheap to extract and that they might allow the model to retrieve underlying information on the number. The first 4 prime numbers $(2,3,5,7)$ were selected as the bases. We used the 6 least significant digits because it is the largest number of digits that allows the biggest base (base 7) to complete at least two cycles with the training data. That is, $2\times7^7<4\times10^6$, but $7^8>4\times10^6$ (and therefore, we cannot use 7 digits). In addition to the representations in the different bases, we added the number itself (divided by 2,000,000, which is the average of the training-set), and the logarithm of the number.

IV. Model Architectures

We used 3 different model architectures:

A. Basic-MLP

This simple model uses a fully connected neural network. The number of neurons was set to 200 on each hidden layer, and the Adam optimizer [17] was used, with a learning rate of 0.001. We used a mini-batch size of 1024 and trained the model for approximately 200 epochs on the data. These values are similar to those we used in previous work [7]. In previous work, [7], we tested several options for finding the optimal number of hidden layers; the model with 5 hidden layers outperformed all other options. We therefore used 5 hidden layers in this paper as well.

For a given number n, the time complexity of generating the features and evaluating our model is $O(log(n))$, which is the best time complexity one could expect from an algorithm that reads the entire input (which requires $O(log(n))$ digits to represent).

We now introduce our novel architectures that were developed for the Goldbach's function approximation in new regions.

B. Multiplication-Layer Model

The multiplication-layer model includes a layer that multiplies pairs of neurons (see Figure 2). Namely, after several fully connected layers (in our case 2) a fraction of the following layer is separated from the rest (in our case 40% of the neurons were separated). The architecture includes random connections between some of the separated neurons; these neurons were multiplied by each other. In practice the model shuffles the rows of an identity matrix, which we denote J.

The group of neurons that were separated, denoted V, is then multiplied, on the right side, by J. The result, $V \times J$, is component-wise multiplied by the original group of separated neurons, V . This process implies that each of the neurons is paired exactly twice (each time with a different neuron). It is possible however, that a neuron will multiply itself, or multiply the same neuron twice.

The motivation behind this architecture was inspired by the analytically derived approximations (G_1 and G_2 functions), which perform relatively well but require prime factorization and are therefore not practical for large numbers. As can be seen, G_1 and G_2 are based on the multiplication of primary factors. We believe that the real solution of the function must include multiplication components. However, a standard neural network cannot multiply two features by each other, but can only approximate such an action. Unfortunately, while such an approximation might perform well when tested on the training-set scope, its performance significantly deteriorates when tested on larger numbers. Therefore, by using the multiplication-layer model there will be actual multiplications between some neurons; this may improve the model performance, especially when predicting numbers larger than the scope of the training-set.

C. Ln-Layer Model

Another architecture introduced in this paper is the ln-layer model (see Figure 3). Similar to the multiplication-layer model, in the ln-Layer model after several fully connected layers, the neurons are divided into two groups. The first group of neurons is fully connected to the next layer (using ReLU activation). The second group is activated first by ReLU and then by the natural logarithm, and its output serves as an input to a separate fully connected layer. That layer is then activated by an exponential function. The two groups are then concatenated and continue with a simple fully connected architecture. Converting the neuron data to natural logarithm and then back by an exponential function enables the network to multiply multiple neurons by each-other.

In addition to the previously introduced architectures, we consider an additional method that make use of Goldbach's function approximation formula developed by Markakis et al. [16], G_4 (see equation 6). This formula was selected because it is the most accurate among the analytically derived formulas that do not require prime factorization.

D. Normalization of the Result

We normalize the result by G_4. To this end, each of the previously presented models (basic model, multiplication-layer model and ln-layer model), is trained to predict the value of Goldbach's function divided by G_4 (rather than simply predicting the value of the Goldbach's function). In addition to the obvious benefit from normalizing the prediction value, dividing by G_4 will likely improve the performance of all models when tested on values that are not in the same distribution area. This is because the model learns the relation between Goldbach's function and G_4; we believe that this relation is less prone to changes (than the actual Goldbach's function) as the number grows. We denote this method by adding '+n' to the original model (that is, the Basic-MLP model with normalized result is denoted Basic-MLP+n, and the Multiplication-layer and Ln-layer models are denoted Multiplication-layer+n and Ln-layer+n, respectively).

E. Results

Test range	Basic-MLP	Multiplication-layer	Ln-layer
4-5	0.4	**0.2**	0.29
5-6	1.58	**1.19**	1.58
6-7	7.65	**4.67**	6.43
7-8	32.53	**12.6**	18.49
8-9	96.42	**23.8**	46.3
9-10	178.88	**38.71**	107.75

TABLE I: MSE of the basic-MLP model, Multiplication-layer model and the Ln-layer model. Note that all models were trained on numbers up to 4 million and were tested on numbers between 4 and 10 million. **All the numbers are in millions**

Table I presents the performance of our models in comparison to G_4 (equation 6), in terms of mean squared error (MSE). All models were trained on numbers up to 4 million, and were tested on numbers up to 10 million. The numbers in the overlapping part (i.e. up to 4 million) were carefully split into a training-set (90% of the data) and a test-set (10% of the data). As expected, the MSE grows as the numbers grow. As we hypothesised, both the Multiplication-layer and the Ln-layer models outperformed the basic-MLP. Somewhat surprisingly, the Multiplication-layer outperformed the Ln-layer model.

Table II presents the MSE of the models normalized by G_4. As expected, the performance of all the methods significantly improved (see Figure 4). Interestingly, the Ln-layer+n model, gained the lead, and outperformed both basic-MLP+n and multiplication-layer+n. However, we would like to note that the derivation of the G_4 formula is not trivial, and therefore, the Multiplication-layer architecture may be used in different domains, in which such an analytically derived function (such as G_4) is not available.

We compare the performance of the Multiplication-layer and the Ln-layer+n models with the performance of the basic MLP model and the G_4 function (which does not require factorization). As depicted by Table III, the Multiplcation and the Ln-layer+n methods outperform the basic MLP and G_4 at all ranges. The Multiplciation model, which does not use any analytically derived formula, outperforms the Ln-layer+n on the 0-4M range (on the test-set), and achieves very close performance in the 4-5M range.

In addition, we test the performance of the Ln-Layer+n model when trained on a subset of the training set. Recall that the training set is composed of 90% of the even numbers between 0 to four million. Table IV presents the performance of the Ln-Layer+n when trained on 100%, 50%, 10% and 1% of the training set. As expected, the performance of the model is much better when trained on a larger data-set; however, even with only 1% of the training set, the Ln-Layer+n performs

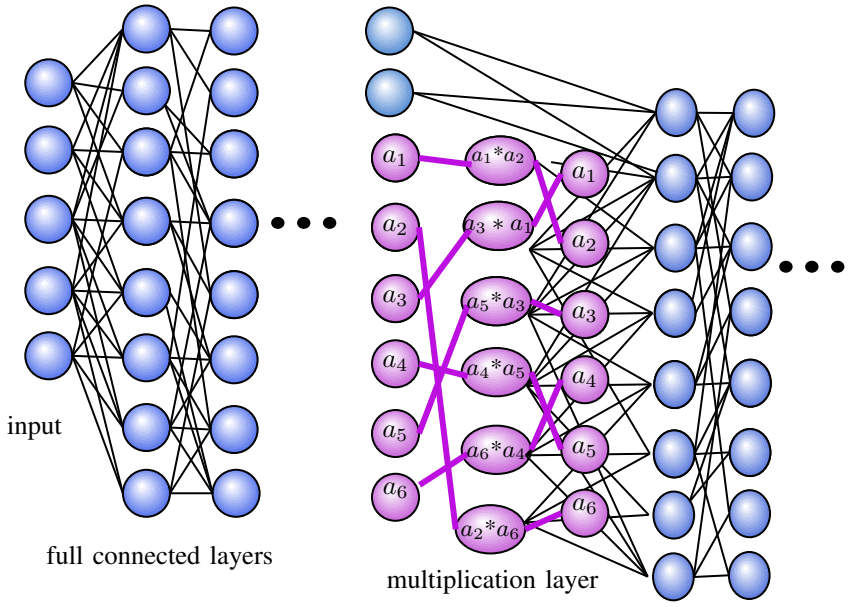

Fig. 2: An illustration of the multiplication architecture. The arcs between the purple neurons ($a_1 - a_6$) demonstrate the multiplication. Note that each of the neurons is paired exactly twice with another neuron. The other neurons (which appear in blue) follow a standard fully connected architecture.

Test range	Basic-MLP + n	Multiplication-layer + n	Ln-layer + n
4-5	**0.17**	0.21	0.19
5-6	0.46	0.47	**0.37**
6-7	1.47	2.08	**1.41**
7-8	4.15	8.5	**3.72**
8-9	10.54	20.95	**6.54**
9-10	23	35.16	**10.79**

TABLE II: MSE of models normalized by G_4. **All numbers are in millions**

Test range	Multiplication	Ln-layer + n	basic MLP	G_4
0-4 (test-set)	**0.034**	0.052	0.1	22.5
4-5	0.2	**0.19**	0.4	73
5-6	1.19	**0.37**	1.58	105.6
6-7	4.67	**1.41**	7.65	140.69
7-8	12.6	**3.72**	32.53	180.14
8-9	23.8	**6.54**	96.42	223.55
9-10	38.71	**10.79**	178.88	270.89

TABLE III: MSE of the Multiplication-layer model and the Ln-layer+n model compared to the MSE of G_4. **All the numbers are in millions**.

Test range	100% of training set	50% of training set	10% of training set	1% of training set	G_4
0-4 (test-set)	**0.052**	0.13	0.2	1.8	22.5
4-5	**0.19**	0.28	0.4	7.1	73
5-6	**0.37**	0.76	1.2	11.3	105.6
6-7	**1.41**	2.1	3.7	20.1	140.6
7-8	**3.72**	5.13	8.18	29.6	180.1
8-9	**6.54**	10.4	15.2	48.8	223.5
9-10	**10.79**	17.96	23.6	67.3	270.8

TABLE IV: MSE of the Ln-layer+n model when trained on 100%, 10%, and 1% of the training set, which is composed of 90% of the even numbers between 0-4 million, compared to the MSE of G_4.

significantly better than G_4 at all ranges. Finally, Table V presents the average number of Goldbach partitions for each of the ranges, the mean absolute error (MAE) values of our selected model (ln-layer+n) along with the MAE of G_4, and the error rate of our selected model. We note that the error rate is less than 5% also in the higher ranges.

V. Discussion

As stated in the introduction, Goldbach's conjecture was verified up to 4×10^{18}. This verification was performed by

Test range	Average no. of partitions	Ln-layer + n MAE	Ln-layer + n error rate	G_4
0-4 (test-set)	10.7	0.13	1.2%	3.4
4-5	22.0	0.25	1.1%	7.1
5-6	26.2	0.4	1.5%	8.5
6-7	30.3	0.7	2.4%	9.8
7-8	34.3	1.1	3.3%	11.1
8-9	38.2	1.4	3.8%	12.4
9-10	42.1	1.9	4.5%	13.6

TABLE V: The average number of Goldbach partitions for each of the ranges in thousands, the mean absolute error (MAE) values of our selected model (ln-layer+n) along with the MAE of G_4 in thousands, and the error rate of our selected model.

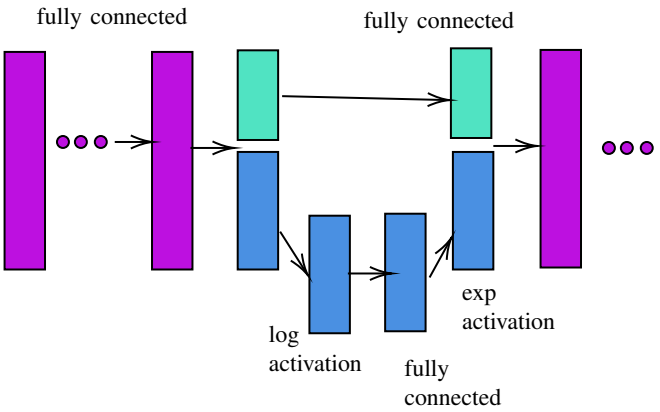

Fig. 3: An illustration of the ln-layer architecture. One of the layers is divided into two sets of neurons. The green set is fully connected to the next layer, while the blue set is first activated by the natural logarithm, then fully connected and activated by an exponent.

using an exhaustive search (see also [18] for an implementation requiring minimal space). Our approximation model may allow a selective search method in which Goldbach's conjecture can be verified only for suspicious numbers according to our model. In other words, only numbers that our model predicts will have a very low number of partitions. This approach can also be used to find numbers that violate the lower-bound proposed by [15]. The methods presented in this paper may allow training on smaller numbers while performing the selective search on larger numbers.

In previous work [7], we used the 10 least significant digits of each base presentation. This model performed well when trained and tested on numbers up to 4 million. However, when this model was tested on large numbers we observed a large gap between its error on $5-6$ million and $6-7$ million. Namely, when tested on numbers between 4 and 5 million the error was 0.35 million, when tested on $5-6$ million it was 0.77 million, but when tested on $6-7$ million, it rose to 128.68 million. This gap is attributed to the fact that in the $0-4$ million region (the training data), the 9th and 10th digits in septenary (base-7) were constantly 0, and the 8th digit in septenary increased monotonically. This is because the number $4,000,000$ is written as $45,666,544$ in septenary (with 0 in the 9th and 10th digits). This caused the weights associated with the 9th and 10th septenary digits to remain with their initial assigned values (noise). Furthermore, since the 8th digit of base-7 increased monotonically, the trained model usually assigned higher values to numbers that had a higher value in the 8th septenary digit. That is, the model did not see any large numbers with a low value, or a 0, in the 8th septenary digit. Therefore, when tested on numbers with 0 in the 8th septenary digit, the model predicted a small value (despite the given number being large). For example, 6.5 million is written as $106,151,303$ in septenary, and therefore has a 0 in the 8th digit location. To overcome this problem and avoid this type of over-fitting, in this paper, we used only the 6 least significant bits. Therefore, in all models presented in this paper, the error rose gradually. For example, in the basic-MLP model presented in this paper, which has an architecture similar to the model presented in [7], the MSE was 0.4 for $4-5$ million, 1.58 for $5-6$ million, and 7.65 for $6-7$ million.

In this paper though we do not attempt to use a deep neural network (DNN) to actually prove a theorem (which is a completely different field), we believe that DNN can be used to help solving open mathematical problems in different ways. The first method is by finding counterexamples (e.g. using selective search) and the second by analyzing the features that the DNN considers when computing its values. In our example we show that using different base representations (which can be computed inexpensively) is very useful for approximating Goldbach's function, (as opposed to the expensive prime factorization computation which was previously known). Future advancements in explainable neural networks, might help shed additional light on this problem when applied to our results. While deep learning has shown great success in many different fields [19], [20], [21], we believe that the success shown in this paper related to an open mathematical problem in number theory, is a big step and should not be disregarded as being merely another deep learning application. Our work may lead to a new paradigm of using deep learning (or machine learning in general) to solve mathematical problems such as prime factorization, friendly numbers, finding prime twins and many similar problems, which may currently seem out of the scope of deep learning methods.

VI. Conclusions

Goldbach's conjecture and Goldbach's function have remained open mathematical questions for over two and a half centuries. There have been several analytic attempts to approximate Goldbach's function, but unfortunately, these approximations either do not work well in practice or require prime factorization (prime decomposition) which is a hard

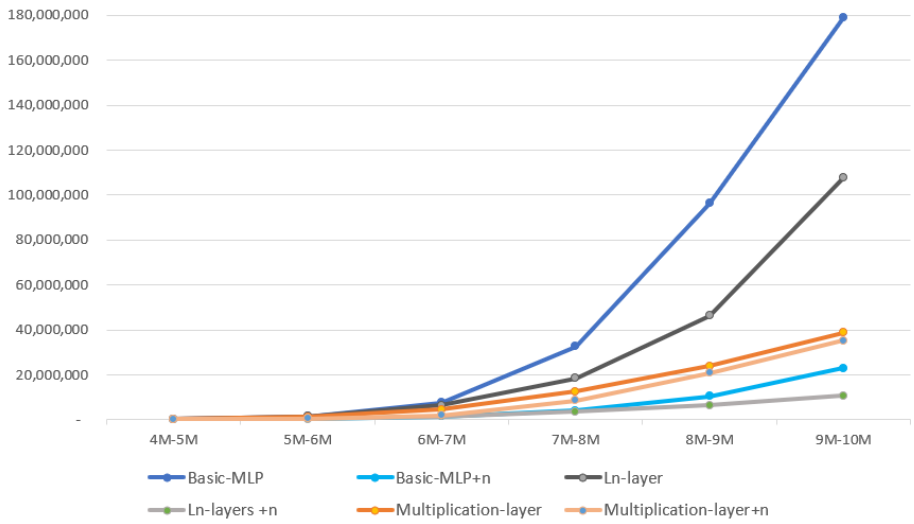

Fig. 4: This plot compares the MSE of the three models (Basic-MLP, Multiplication-layer and Ln-layer) with the MSE of their normalized versions (lower is better). As depicted in the figure, all models gain from this normalization.

problem. In previous work, we developed a basic multi-layer perceptron and show that this simple model performs well when trained and tested on numbers up to 4 million. However, the model's performance significantly deteriorates when trained on smaller numbers (up-to 4 million) but tested on larger numbers ($4 - 10$ million).

To overcome this problem, in this work we presented two novel deep learning architectures. In these architectures we introduced two types of multiplication layers; in the first architecture some of the neurons of a specific layer are multiplied by each-other (the Multiplication-layer model). In the second architecture a log activation is performed at the output of a set of neurons that is followed by a fully connected layer with an exponential activation (the Ln-layer model). We showed that both architectures significantly outperform the basic multi-layer perceptron when trained on smaller numbers and tested on larger numbers. We further improved the performance of the deep learning architectures by normalizing the model's output by a known analytically derived estimation (G_4).

REFERENCES

[1] C. Goldbach, Letter to L, Euler 7 (1742) 1.

[2] T. Oliveira e Silva, S. Herzog, S. Pardi, Empirical verification of the even Goldbach conjecture and computation of prime gaps up to 4×10^{18}, Mathematics of Computation 83 (288) (2014) 2033–2060.

[3] H. F. Fliegel, D. S. Robertson, Goldbach's comet: the numbers related to Goldbach's conjecture, Journal of Recreational Mathematics 21 (1) (1989) 1–7.

[4] G. H. Hardy, J. E. Littlewood, Some problems of diophantine approximation: The lattice-points of a right-angled triangle, Proceedings of the London Mathematical Society 2 (1) (1922) 15–36.

[5] J. Baker, Excel and the Goldbach comet, Spreadsheets in Education (eJSiE) 2 (2) (2007) 2.

[6] I. Goodfellow, Y. Bengio, A. Courville, Y. Bengio, Deep learning, Vol. 1, MIT press Cambridge, 2016.

[7] A. Stekel, M. Shukrun, A. Azaria, Goldbach's function approximation using deep learning, in: 2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI), IEEE, 2018, pp. 502–507.

[8] Y. Wang, The Goldbach Conjecture, Vol. 4, World scientific, 2002.

[9] H. A. Helfgott, The ternary Goldbach conjecture is true, arXiv preprint arXiv:1312.7748 (2013) 1–79.

[10] I. O. Bado, New discovery on goldbach, International Journal of Progressive Sciences and Technologies 13 (2) (2019) 216–221.

[11] C. Liu, A study of relationship among goldbach conjecture, twin prime and fibonacci number., IJ Network Security 19 (3) (2017) 406–412.

[12] A. Berdondini, The importance of finding the upper bounds for prime gaps in order to solve the twin primes conjecture and the goldbach conjecture, arXiv preprint arXiv:2002.07174 (2020) 1–10.

[13] A. Granville, Refinements of goldbach's conjecture, and the generalized riemann hypothesis, Functiones et Approximatio Commentarii Mathematici 37 (1) (2007) 159–173.

[14] J. P. Buhler, H. W. Lenstra, C. Pomerance, Factoring integers with the number field sieve, in: The development of the number field sieve, Springer, 1993, pp. 50–94.

[15] C. Provatidis, E. Markakis, N. Markakis, Rule of thumb bounds in Goldbach's conjecture, American Journal of Mathematical Analysis 1 (1) (2013) 8–13.

[16] E. Markakis, C. Provatidis, N. Markakis, Some issues on Goldbach conjecture, Number Theory 29 (2012) 1–30.

[17] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, Proceedings of the 3rd International Conference on Learning Representations (ICLR) (2014) 1–15.

[18] J. Richstein, Computing the number of goldbach partitions up to 5 10 8, in: International Algorithmic Number Theory Symposium, Springer, 2000, pp. 475–490.

[19] Y. Lv, Y. Duan, W. Kang, Z. Li, F.-Y. Wang, Traffic flow prediction with big data: a deep learning approach, IEEE Transactions on Intelligent Transportation Systems 16 (2) (2015) 865–873.

[20] A. A. Cruz-Roa, J. E. A. Ovalle, A. Madabhushi, F. A. G. Osorio, A deep learning architecture for image representation, visual interpretability and automated basal-cell carcinoma cancer detection, in: International Conference on Medical Image Computing and Computer-Assisted Intervention, Springer, 2013, pp. 403–410.

[21] B. Alipanahi, A. Delong, M. T. Weirauch, B. J. Frey, Predicting the sequence specificities of dna-and rna-binding proteins by deep learning, Nature biotechnology 33 (8) (2015) 831.

Learning to Conceal: A Method for Preserving Privacy and Avoiding Prejudice in Images

Avigail Stekel, Moshe Hanukoglu, Aviv Rovshitz, Nissan Goldberg, and Amos Azaria
Computer Science Department and Data Science Center
Ariel University, Israel

***Abstract*—We introduce a learning model able to conceal personal information (e.g. gender, age, ethnicity, etc.) from an image while maintaining any additional information present in the image (e.g. smile, hair-style, brightness). Our trained model is not provided the information that it is concealing, and does not try learning it either. Namely, we created a variational autoencoder (VAE) model that is trained on a dataset including labels of the information one would like to conceal (e.g. gender, ethnicity, age). These labels are directly added to the VAE's sampled latent vector. Due to the limited number of neurons in the latent vector and its appended noise, the VAE avoids learning any relation between the given images and the given labels, as those are given directly. Therefore, the encoded image lacks any of the information one wishes to conceal. The encoding may be decoded back into an image according to any provided properties (e.g. a 40-year old woman).**

Our method successfully conceals the private information; a convolutional neural network trained on the concealed images cannot restore the original private information. In contrast to the private information, a user study shows that the remaining properties of the original image carry-on to the concealed image. The proposed architecture can be used as a mean for privacy preserving and can serve as an input to systems, which will become unbiased and not suffer from prejudice.

I. Introduction

There are many implications of user privacy with respect to user data; foremost is the fear of exposing personal information over a social network. As indicated by Almadhoun et al. [2] 75.8% of the respondents do not believe that they would feel totally safe when providing sensitive information about themselves over the social networks. Indeed, several network attacks exist that allow strangers to extract personal information from a victim [14]. Therefore, any personal data uploaded to the internet might be exposed by a third party who should not be permitted to view it. Surveillance cameras, which record public locations 24/7, may be perceived as highly privacy invaders. Furthermore, with the rise of miniature particle accelerators, and the use of terahertz waves, which allow to "see through" clothes, surveillance cameras may become much more invasive.

In addition, user-privacy also relates to research communities. Rothstein [10] states that the current regulatory frameworks of the Common Rule and Privacy Rule emphasize privacy interests, but they overlook the privacy interests of individuals whose health information and biological specimens are used in research without their knowledge, consent, or authorization. Methods for data anonymization would allow faster and more effective research, including life-saving and medical research, as people are more likely to be willing to share truly anonymized data.

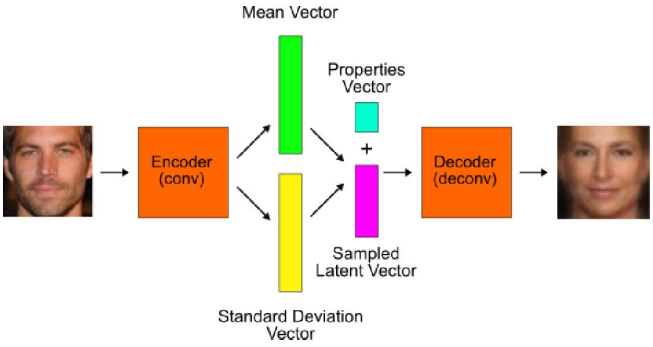

Figure 1. General architecture of The Blind Autonencoder for Fairness and Objectiveness (BAFO).

To overcome the issues of privacy and prejudice in images, we introduce the Blind Autonencoder For Fairness and Objectiveness (BAFO), a novel deep learning architecture that is based on a variation autoencoder (VAE) [9]. Namely, BAFO is trained on a dataset including labels of the information one would like to conceal (e.g. gender, ethnicity, age). These labels are directly added to the VAE's sampled latent vector (see Figure 1). Due to the limited number of neurons in the latent vector and its appended noise, the VAE avoids learning any relation between the given images and the given labels, as those are given directly. Therefore, the encoded image lacks any of the information one wishes to conceal; BAFO is practically blind to all this information. The encoding may be decoded back into an image according to any provided properties (e.g. a 40-year old woman).

It might seem unintuitive that during training BAFO is given the information that it should later conceal. However, BAFO's behavior might be similar to a child that is asked to use a calculator from the very beginning of preschool. The child might focus on acquiring complex mathematical skills but is not likely to know how to add or multiply numbers by herself, because she learns to trust the calculator instead. Furthermore, if that child will later use a different calculator that uses different functions, the computation results will become different. Another similar example is the autopilot, which may cause pilots to not be able to fly an airplane themselves, because they learn to trust the autopilot [4].

2375-0197/20/$31.00 ©2020 IEEE
DOI 10.1109/ICTAI50040.2020.00121

Using BAFO, security offices may monitor concealed surveillance footage, that is, surveillance footage in which all information required to be concealed (e.g. gender, race) is not present. It is important to note that this information is not only removed, but explicitly not learned by BAFO.

The general idea behind BAFO is to explicitly provide the model, during its learning process, with the information that it should conceal during the inference phase, in which this information will not be provided. We believe that this idea is not limited to images and videos, but is more general and can easily adapted to be used for concealing one's voice, and text. Concealing voice may be used by the press or by court when the identity of the speaker needs to remain unknown. Concealed text may be used when applying for education or job openings. A candidate may fear to fall for prejudice, and may therefore wish to hide her gender or race from the curriculum vitae (CV), in a way that this information cannot be deduced. Text concealing can be used also for robustly anonymizing data. This is because BAFO would remove not only any explicit identifiers (e.g. name and country of birth), but also from any implicit ones (e.g. specific expressions used by some group of people). In addition, users knowing that their data is totally anonymizing are more likely to share their data. Data sharing is especially important in any medical related system, which may literally save people's life.

It is very important to note that the idea behind BAFO is very different than other tools that may be used to convert one type of image to another (e.g. showing an older version of one-self) [1]. Such technologies apply a smart filter that converts some type of image to another, this filter is applied regardless of the original image. Therefore, in such systems if a feminine filter is applied on a woman, she would seem even more feminine. Unlike with BAFO, if trained to remove gender, in which all gender related aspects of any image are totally removed, and are later explicitly added in order to produce a new image.

We show that BAFO successfully conceals the private information; a convolutional neural network trained on the concealed images cannot restore the original private information. We further show that the concealed image does match the additional information provided. That is, for example, a male image concealed as a female, is usually classified as a female. A user study shows that the remaining properties of the original image carry-on to the concealed image; in 90% of the images, the participants could identify the original image that was concealed by BAFO to a given concealed image.

II. Related work

In recent years there have been a number of popular approaches for creating artificial images: Generative Adversarial Network (GAN) [6] is a Generative model for creating new information such as creating fake high quality images. GANs are trained to output images that look real, and are therefore sharp and have high contrast. GANs include a discriminator and a generator; both components compete with each-other. The discriminator identifies whether each image is original or has been created by the generator, while the generator's goal is to create images that will seem real to the discriminator. That is, the generator tries to deceive the discriminator into thinking that the the images created by the generator are original. Baek et al. [3] created a face editing tool that is based on GANs. However, GANs are not appropriate for our goal, since we do not intend to produce an image that looks as real as possible, but to preserve the original image while concealing the intended properties.

Another deep learning based approach for generating images is the Variational AutoEncoder (VAE) [5]. Similar to a standard AutoEncoder, a VAE architecture includes a bottleneck and, during training, it attempts to restore a given image. The representation in the bottleneck is, in fact, a compressed representation of the given image. However, a VAE learns two vectors, a mean vector and a standard deviation vector; the latent vector is sampled using the mean and standard deviation vectors. VAEs have been used for denoising and generating images that are similar to the training-set [7]. In this paper we propose a novel concept in which the VAE that is trained on the given data, but the labels are appended directly to the latent vector (see Figre 1). The VAE, therefore, does not attempt to learn any information that is directly provided, and becomes blind to these properties. While not being the primary intention of our VAE architecture, we note that it can also be used to generate images with specific attributes (e.g. only images of a 40-year-old female).

Several fair machine learning methods have been developed recently including some based on deep learning. Louizo et al. [8] develop a classifier that is based on a variational autoencoder, VFAE. This classifier obtains a sensitive variable along with additional insensitive data and outputs a prediction of a class for each input. The authors apply VFAE to four datasets. For example, one of the datasets the authors apply VFAE to is the Adult income dataset. In this domain, VFAE obtains the age of an individual along with 14 additional attributes, and outputs a prediction as to whether the individual has an annual income of over $50,000$ or not. This work differs from ours in several aspects. First, VFAE does not output any intermediate value that is human interpretable. Secondly, VFAE is trained for a specific classification task. Finally, VFAE requires the sensitive information at inference, while BAFO uses the sensitive information only at training, and can therefore conceal this sensitive information without obtaining it. We also note that VFAE is limited to a binary classification task, that is, a task with only two classes. This limitation was later relaxed by [11] who improved their method and extended it to a classifier with multiple classes.

Zemel et al. [12] introduced a measure, $yDiscrim$, of discrimination for a classification problem. $yDiscrim$ is the difference between the ratio of the positive predictions in a specific set and the ratio of the positive predictions in the rest of the examples in the data. Formally:

$$yDiscrim = \left| \frac{\sum_{n:s_n=1} \hat{y}_n}{\sum_{n:s_n=1} 1} - \frac{\sum_{n:s_n=0} \hat{y}_n}{\sum_{n:s_n=0} 1} \right| \qquad (1)$$

where S is a binary variable representing whether a given individual is a member of a specific set, and $\hat{y}$ is the prediction for y. We note that this measure is only applicable to a scenario in which the overall goal of a system is to classify individuals to different classes. Furthermore, this measure is not relevant for cases in which different groups must have different ratios. For example, a medical system predicting whether a subject is suffering from a specific illness. While in such cases we might wish to conceal private information, the system should still provide the correct ratio for each group. However, in our work, the overall goal of the system is to produce an output that conceals some information while remaining interpretable to humans.

III. Dataset

We use the UTKFace dataset [13], which contains approximately 23,000 headshot images. The images in the dataset are labeled with the age (ages range from 0 to 116), gender (male and female) and ethnic origin which is divided into five types. The dataset was split into 85% training-set and 15% test-set. All our software is available at: https://github.com/avigailst/Learning_to_Conceal.

IV. Method

In this section we introduce the Blind Autonencoder For Fairness and Objectiveness (BAFO).

The general architecture of the model is depicted in Figure 2. In the training phase, a labeled image is inserted as an input to BAFO. In the UTKFace dataset the images size is 56x56x3 (RGB), and the images are tagged by age, gender, and ethnicity. We note that the labels include the information that we intend to later conceal. The image is then compressed by the encoder into two vectors representing the parameters that describe the image, one vector for the mean and the other for the standard deviation. After sampling the latent vector from the mean and the standard deviation, it is concatenated to the the information BAFO is concealing, the age, the gender and the ethnicity. The latent vector is then decoded back to an image with a size similar to the input image.

The motivation behind this architecture is that, since the information to be concealed is provided without noise, the system will be able to devote all of its efforts to learning only the additional parameters that affect the image, and not the information that is explicitly provided.

1) Latent Vector Size: We consider two different sizes for the latent vector, 48 and 100. In order to evaluate the performance of BAFO and determine which vector size to use, we concealed the test data and decoded it into females and males in 5 age groups: 1-year-old, 20-year-old, 40-year-old, 60-year-old and 80-year-old. In order to evaluate the decoded concealed images with respect to the age, we developed an age classifier. We note that the age classifier itself had a root mean squared error (RMSE) of 6.75. The root mean squared error (RMSE) and mean absolute error (MAE) of the decoded concealed images appear in Table I. As depicted by the table, BAFO with a latent vector of size 48 seems to slightly outperform BAFO with a latent vector of size 100.

Target age	RMSE 48-cells	RMSE 100-cells	MAE 48-cells	MAE 100-cells
1	**12.62**	13.34	10.60	11.42
20	9.37	**9.00**	7.94	**7.70**
40	**7.89**	7.97	**6.56**	6.65
60	**18.03**	18.71	**16.45**	17.45
80	**27.41**	28.30	**26.03**	27.26
Average	**15.06**	15.47	**13.52**	14.10

Table I
THE ROOT MEAN SQUARED ERROR (RMSE) AND MEAN ABSOLUTE ERROR (MAE) OF CONCEALED IMAGES DECODED AS 1 YEAR OLDS, 20, 40 60 AND 80 YEAR OLDS (LOWER IS BETTER).

V. Results

Figure 3 presents 7 randomly picked images from the test-set, and the output of BAFO when concealed as a 40-year old woman, using 48 and 100 length latent vectors. The input images appear in the first row; in the second row are the images concealed by BAFO when using a 48 length latent vector; and in the third row are the images concealed by BAFO when using a 100 length latent vector. Note that the facial features are smoother and delicate in the cheek and nose areas, as well as the thinner eyebrows in pictures 1,4 and 6, as shown in Figure 3. Note the way the system conceals only the age and gender of the given image, but preserves the smile, including the presence of teach, hair-style, light conditions, brightness of the images and the background.

Figure 4 presents the same images when concealed as a 40-year-old male. The male's facial features became more coarse in the center of the face, accompanied by wrinkles at the sides of her eyes and mouth, as can be seen in the second, fifth and seventh images. Note the smile, the length of the eyebrows and the structure of the nose are preserved in the second row after the process is completed.

In Figure 5, we used a single image as the input and concealed it as a male and a female with ages varying from a one-year-old to an 80-year-old person. As seen in the figure, BAFO can conceal any image at any range of ages and both as a male and a female. This result may resemble common image editor tools (especially when making an image look older or younger) [1], which are mainly used for entertainment. However it is very important to note that BAFO is very different than those tools as it is not trained to convert images from one demographic group to another (or making an image older or younger), but is totally blind to the demographic group. Other image editor tools are either directly trained to convert from one demographic group to another, or to add (and remove) specific features (such as aging features, feminine features, smile features etc.). Very importantly, BAFO is not provided the age, gender or ethnicity of the original image, nor does it learn it itself, as BAFO is not required to know whether the image should be converted to a younger image

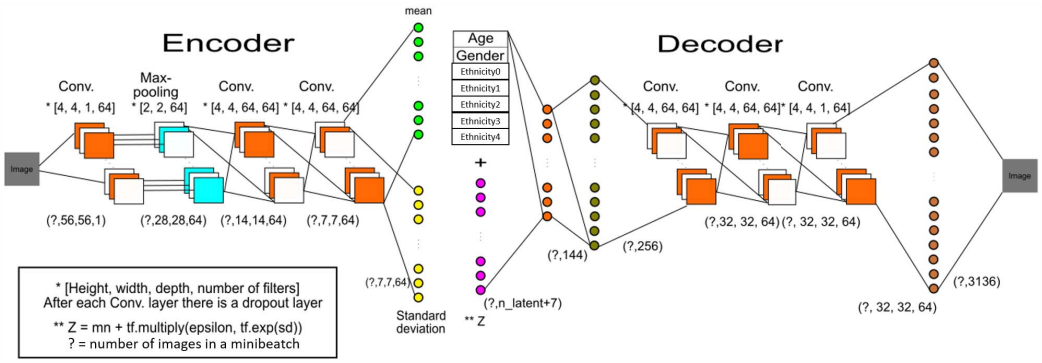

Figure 2. The Architecture of the VAE.

Figure 3. Images concealed by BAFO as a woman 40-year-old. In the first row are the input images; in the second row are the images concealed by BAFO when using a 48 length latent vector; and in the third row are the images concealed by BAFO when using a 100 length latent vector. Note the way the system conceals only the age and gender of the given image, but preserves the smile, including the presence of teach, hair-style, light conditions, brightness of the images and the background.

Figure 4. Concealing images as a 40-year-old male. In the first row are the input images and in the second row are the images concealed by BAFO when using a 48 length latent vector.

or an older image. Therefore, BAFO is truly unbiased and the demographic group is explicitly *added* to the image encoding only in order to produce a meaningful image.

VI. Evaluation

There are three dimensions for the evaluation of BAFO. The first is an evaluation of the quality of output with respect to the demographic information provided. That is, if a male image was concealed as a female, the output image should be classified as a female. The second is the evaluation of BAFO's ability to conceal information. That is, determining to what extent the concealed information can be retrieved from the generated image. The third dimension for evaluating

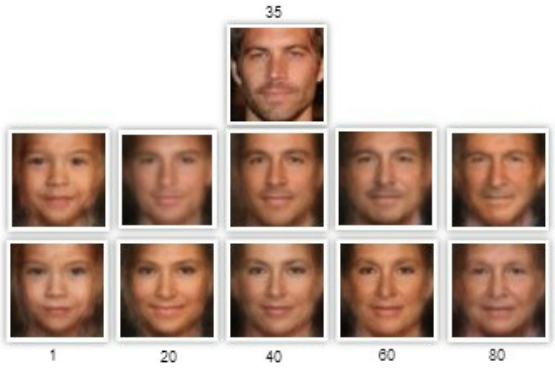

Figure 5. Concealing a single image as a man and a woman at different ages. This may resemble common image editor tools, however BAFO is very different than those tools as it is not trained to convert images from one demographic group to another, but is totally blind to the demographic group. The demographic group is explicitly *added* to the image encoding in order to produce a meaningful image.

BAFO is its ability to construct an image that is similar to all properties of the original image, excluding the concealed information. The former two dimensions were evaluated by using a convolutional neural network, while the later dimension was evaluated by human judges.

A. Evaluation with a Convolutional Neural Network

To evaluate the first and the second dimensions, we developed a classifier with two convolution layers and two fully connected neural network layers. The classifier was trained and tested on different datasets according to its evaluation goal. We focus on the evaluation of the gender property of the image. For that end we identified four different data groups (as shown in Figure 6):

- The images of *males* from the original dataset (group A).
- The images of *females* from the original dataset (group B).
- The images from groups A and B that were concealed as *males* by BAFO (group C).
- The images from groups A and B that concealed as *females* by BAFO (group D).

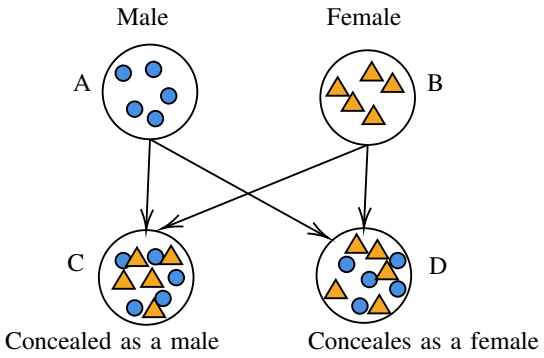

Figure 6. This figure illustrates the datasets that where used to evaluate BAFO using a convolutional neural network. The blue circles represent the male images and orange triangles represent female images. Group A and group B represent the original images, A for the male images and B for the female images. Groups C and D represent the images of A and B that were concealed as males in group C and as females in group D.

	Training	Test
$A+B \rightarrow A+B$	98%	88.3%
$A+B \rightarrow C+D$	$--$	86.5%
$C \rightarrow C$	86.5%	43.9%
$D \rightarrow D$	88.5%	47.7%

Table II
THE RESULTS OF THE CLASSIFIER THAT WAS DEVELOPED FOR EVALUATING BAFO BY THE FIRST AND THE SECOND DIMENSIONS. WHEN THE CLASSIFIER WAS TRAINED ON A+B AND TESTED ON THE TEST-SET OF A+B, ITS ACCURACY WAS 88.3%, WHICH IS VERY SIMILAR TO ITS ACCURACY WHEN TESTED ON GROUPS C+D. THIS IMPLIES THAT BAFO'S OUTPUTS MATCH THE REQUIRED GENDER. FURTHERMORE, THE CLASSIFIER DID NOT SUCCEED IN IDENTIFYING WHETHER AN IMAGE FROM GROUP C WAS CONCEALED FROM A MALE IMAGE OR A FEMALE IMAGE (ACCURACY 43.9%). A SIMILAR RESULT WAS OBTAINED IN GROUP D. THIS IMPLIES THAT BAFO CONCEALS THE GENDER PROPERTY VERY WELL.

All data groups were split into 90% training data and 10% test data. The classifier was first trained on the training data in groups A and B, with the label being the gender of the person in the images. When tested on the test data (of groups A and B) the classifier obtained an accuracy of 88.3%. A similar accuracy (86.5%) was obtained when tested on the images in groups C and D. Where in group C the classifier had to predict the value for male and in group D, female. This implies that BAFO's output matches the required gender. That is, BAFO succeeded in the evaluation of the first dimension.

In order to evaluate the second dimension, the classifier was then trained separately on groups C and D, with an attempt to recover the concealed information. That is, the images in groups C and D were labeled according to the gender in the original images. The classifier was tested on the test-set from groups C and D. In this evaluation lower accuracy means that the classifier could not recover the concealed information. That is, lower accuracy implies that BAFO had succeeded in concealing the private information. Indeed, the accuracy measured was very low (less than 50%, which could be obtained by a random classifier). All results are shown in II.

B. Evaluation with Human Judges

In order to evaluate the third dimension we conducted a survey. The participants were asked ten questions: In the first set of five questions the participants were shown an original image from the dataset and two images that were transformed, by BAFO, to a 40-year-old woman. The participants were asked to choose the image which they believed was transformed from the original given image. The structure of the second set of five questions was reversed; we provided one transformed image and two original images and the participants were asked to pick an original image. There were 127 participants aged between 17 and 72, 49 males and 80 females. The results are presented in the Table III. The participants answered correctly on 85.5% questions when the images that were in the questions were of the same ethnic group and 92.4% on questions with images not in the same ethnic group. The results show that BAFO's outputs preserve their properties, as humans successfully identify the original image even when the ethnicity property is the same.

	No. of queries	Success rate
From the same ethnic	315	85.8%
From a different ethnic	916	92.4%
Identifying the original image	670	88.8%
Identifying the concealed image	597	92.4%
Total	1267	90.5%

Table III
THE RESULTS FROM THE HUMAN SURVEY. THE RESULTS SHOW THAT BAFO'S OUTPUT PRESERVES THE IMAGE'S PROPERTIES AS HUMANS SUCCESSFULLY IDENTIFY THE ORIGINAL IMAGE (AND VICE-VERSA).

VII. DISCUSSION

As depicted by Table I and Figure 3, when modifying the latent vector size, there seems to be a trade-off between the image quality and the concealing performance. That is, with a large latent vector, the image decoded image quality is slightly better, but the image is slightly not concealed as well. This is expected, as the larger the latent vector, the more information BAFO can be stored in it, and the less does it need to rely on the additional information. On the other hand, the larger the latent vector, the more features may BAFO store in it and the higher the image quality. It is yet to be determine what the optimal latent vector size is, this may depend not only at the application, but also at the amount of labeled data available. Another approach is to increase the latent vector size, but also to modify the KL-divergence formula, so that the standard deviation will receive higher penalty rates, and therefore the data obtained from the mean-vector will be

very noisy. This will further encourage BAFO to rely on the provided information, as much as possible.

In order to decode the concealed images, so that it is understandable for humans, BAFO needs to be given some demographic information (or any other information related to the concealed information). However, it would be preferable if BAFO could present the information using a neutral representation. For example, by using an image that is not gender associated. As a first approach to achieve such an image, we simply provided a value of 0.5 (the mean of the values for female and for male) as the gender value to BAFO. However, this approach did not perform that well; this is not surprising, since no image with the value of 0.5 was given to BAFO during the training phase.

VIII. Conclusions & Future Work

In this paper we introduce BAFO, an image concealer that receives as input an image and conceals unwanted properties (such as gender, ethnic origin, and age). The concealed images may be viewed by humans in a way that would remove any prejudice related to the concealed properties. BAFO may be embedded in smart glasses for security officers or other law enforcers, such that some properties of people who they interact with are concealed. These images can serve as input to another machine learning system, which, due to the input it receives from BAFO, will be unbiased. BAFO may also be used as a mean for privacy preserving by social network users. A user may conceal user private information in images (e.g. age, gender, ethnic origin) before she uploads them to a social network. Users who are familiar with that user and know the private information, will be able to decode the image according to the private information, and will view an image that is very similar to the original image that was uploaded. Other people will see the images, but will not be able to extract the private information concealed in these images.

Extending the architecture described in this paper to concealing one's identity is straight-forward. Such concealed images should preserve privacy and therefore could be used by researchers in different fields. Future work will also include the extension of BAFO's architecture to video, voice and text. Such extensions may have major implications on privacy preserving and unbiased systems, such as an *unbiased* surveillance camera with automatic security threat detection.

IX. Acknowledgment

This research was supported in part by the Ministry of Science, Technology & Space, Israel.

References

[1] Zahid Akhtar, Dipankar Dasgupta, and Bonny Banerjee. Face authenticity: An overview of face manipulation generation, detection and recognition. In *Nutan College of Engineering & Research, International Conference on Communication and Information Processing (ICCIP)*, 2019.

[2] Nour Mohammed Almadhoun, P Dhanapal Durai Dominic, and Lai Fong Woon. Perceived security, privacy, and trust concerns within social networking sites: The role of information sharing and relationships development in the malaysian higher education institutions' marketing. In *2011 IEEE International Conference on Control System, Computing and Engineering*, pages 426–431. IEEE, 2011.

[3] Kyungjune Baek, Duhyeon Bang, and Hyunjung Shim. Editable generative adversarial networks: Generating and editing faces simultaneously. In *Asian Conference on Computer Vision*, pages 39–55. Springer, 2018.

[4] Nicholas Carr. *The glass cage: Automation and us*. WW Norton & Company, 2014.

[5] Carl Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.

[6] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[7] Huaibo Huang, Ran He, Zhenan Sun, Tieniu Tan, et al. Introvae: Introspective variational autoencoders for photographic image synthesis. In *Advances in Neural Information Processing Systems*, pages 52–63, 2018.

[8] Christos Louizos, Kevin Swersky, Yujia Li, Max Welling, and Richard Zemel. The variational fair autoencoder. *arXiv preprint arXiv:1511.00830*, 2015.

[9] Yunchen Pu, Zhe Gan, Ricardo Henao, Xin Yuan, Chunyuan Li, Andrew Stevens, and Lawrence Carin. Variational autoencoder for deep learning of images, labels and captions. In *Advances in neural information processing systems*, pages 2352–2360, 2016.

[10] Mark A Rothstein. Is deidentification sufficient to protect health privacy in research? *The American Journal of Bioethics*, 10(9):3–11, 2010.

[11] Jiaming Song, Pratyusha Kalluri, Aditya Grover, Shengjia Zhao, and Stefano Ermon. Learning controllable fair representations. *arXiv preprint arXiv:1812.04218*, 2018.

[12] Rich Zemel, Yu Wu, Kevin Swersky, Toni Pitassi, and Cynthia Dwork. Learning fair representations. In *International Conference on Machine Learning*, pages 325–333, 2013.

[13] Zhifei Zhang, Yang Song, and Hairong Qi. Age progression/regression by conditional adversarial autoencoder. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5810–5818, 2017.

[14] Bin Zhou and Jian Pei. Preserving privacy in social networks against neighborhood attacks. In *ICDE*, volume 8, pages 506–515. Citeseer, 2008.

Evidence From Computational Linguistics for the Concept of Gates in Hebrew

Anonymous Submission

Abstract

Keywords:

Introduction

Natural languages evolve both through various processes not supervised by people, growing from below, and through the efforts of correction and normalization (standardization) carried out by competent people and institutions.

An example of an uncontrolled process is the borrowing of words or the assimilation of phonemes due to geographic proximity to another language. For example, Akkadian lost its guttural consonants due to the influence of the non-Semitic Sumerians Huehnergard (2011), just as modern Hebrew, which is spoken today in Israel, lost its guttural consonants, being spoken by people born in Europe.

An example of s normalization process is the effort of Cyril and Methodius to canonize the Russian language as well as the recent effort of the Israel Academy of the Hebrew Language to revive this ancient language.

The evolutionary paths of human language are not always easy to trace back by reverse analysis. Many questions arise in connection with this evolution, and the earlier the observed stage of evolution is, the more questions stay without certain answers,

Data science, and machine learning in particular, can help language researchers through its ability to uncover traces of evolutionary events hidden from the eyes of researchers by the veil of history, statistically confirming their impact on the word structures, sentence structures, concept proximity, etc., of a given natural language or group of languages.

Semitic languages represent a vast area of research on the evolution of natural language. They are the dominant languages of some of the earliest known cultures. Many ancient documents are written in Semitic languages. The morphological structures of the roots of Semitic words are raw and primary, which indicates their relative proximity to the initial processes of natural language evolution. Semitic languages such as Akkadian, the language of the ancient Babylonian Empire, Hebrew, which was spoken by most of the biblical characters in the ancient kingdoms of Judah and Israel, and Aramaic, which was the dominant language (lingua franca) of the Persian Empire, were and continue to be of great interest to linguists.

There is a debate among researchers about whether the development of natural language occurred as an evolutionary leap or as a gradual evolutionary process. Another controversy, arising from the aforementioned debate, is whether speech skills are developed in the human brain by imitation (the Theory of Behaviorism) or universal structures already exist in the human brain even before birth, and they only find their channels of expression, which are specific to each particular language, making world grammars different from each other, but preserving the deep system of innate concepts/intents (the Theory of Nativism/Innatism).

We can reconcile Nativism and Behaviorism if we assume that evolutionary processes in the development of morphological structures took place either during the metaphysical formation of the human brain itself (implying that spiritual abstractions also evolve from simple to complex) or during the behavioral formation of natural languages. In any case, evolutionary patterns are evident in natural languages.

If the formation of language is indeed a gradual evolutionary process, then it is reasonable to assume that this evolution began among prehistoric people with the exchange of animal-like sounds - individual phonemes or syllables, which were enough to express few ideas and feelings. As people needed more concepts, as their environment and business became more diverse with different types of crops, farm animals, and tools, the need for a richer system of terms increased. Thus, people began to create new words by combining existing syllables.

This paper deals with a morphological phenomenon in Biblical Hebrew. Biblical Hebrew is a Semitic language, and as such its roots consist only of consonants where the vowels serve as the glue between the consonants and are not stable when listing the different syntactic forms of the root. An illustrative description of this situation is the metaphor of consonants as the pegs of a tent, and vowels as the tent sheets. A Semitic language first hammers the pegs and only then lays out the different sheets that may change depending on the weather (the context).

To illustrate this characteristic of the Semitic languages, we review several words derived from the root KTB denoting the English root "write": **K**o**T**e**V** = is writing, **K**a**T**a**B** = wrote, **K**a**T**u**B** = written, **KT**i**B**a = writing. Even the highly interconnected forms of the root show a high level of flexibility, if not total freedom, in selecting vowels along with high adherence to the three consonants of the root.

In this paper, we show the high feasibility of a gradual evolutionary process, and establish the evolution of Semitic syllables, or more correctly biconsonant (2C) roots [1], also known as "gates" [2] to triconsonant (3C) roots.

Most of the roots of Biblical Hebrew have three consonants. There are several indicators both in Biblical Hebrew and in its derivative, Modern Hebrew, that confirm the evolution of biconsonant gates into triconsonant roots. One indicator is the presence of a minority of biconsonant roots alongside the overwhelming majority of triconsonant roots in Bib-

[1] Vowels do not participate in the formation of Semitic roots

[2] The term "gate" comes from the fact the two consonants are seen as two jambs of a true gate

lical Hebrew. The next indicator is the presence of the "hollow" and the "disabled" subgroups of Hebrew roots in which the third consonant is simply missing, and the root is made up of two consonants and an auxiliary semi-consonant. Another indicator is the presence of biconsonants gates that clearly do not change their central meaning when a third consonant is added to them to form a triconsonant root. An example of such a gate is the Semitic gate PR whose primary meaning is "fruitfulness", a word that most succinctly describes a process of multiplication and expansion, and the addition of a third consonant does not harm this basic meaning and only adds a shade to it (it is possible to assume that in ancient Hebrew predecessor language, these additional consonants were added as suffixes). Figure 1 lists the various roots derived from the PR gate and their meanings, expressing the various shades of expansion: explosion, proliferation, exaggeration, separation, etc.

- PR$>$ = wild, savage (one who cannot control his instincts and they explode out).
- PRG = a flower with many seeds, spread out by the wind.
- PRD = separate (from one to many)
- PRH = proliferate
- PRZ = erase boundaries, exaggerate
- PRX = flower, flourish, prosper
- PRV = break down into pieces
- PRK = break gradually
- PRM = unstitch
- PRS = cut into pieces
- PR$<$ = ruffle
- PRY = burst
- PRQ = take to pieces, unpack
- PRR = crumble, granulate
- PRC = retire, secede
- PRT = detail

Figure 1: Various roots derived from the PR gate. Note that the BHSA Hebrew to Latin transliteration is used.

This is a well-known example, demonstrating a high level of internal statistical significance that can only confirm the phenomenon for this single gate. More gates with lower but still high levels of internal statistical significance are also widely known, making the phenomenon statistically significant throughout the whole language. For example, gate QC, which primarily means "edge", evolves into a group of roots whose hues semantically converge to "cut": QCB=allot, QCR=reap, QCC=chop, and more. Some gates hold proximity between them. For example, gate GZ is semantically close to QC and its derivative roots just mean more shades of "cut" (imaginative people may even note the phonetic closeness between the Semitic "QC" and the Latin "CUT", but this is not the right scope to delve into such controversial reasoning).

Another indication of the evolutionary development of Semitic roots is the presence of quadriconsonant(4C) roots in the Hebrew Bible. The most famous Biblical quadriconsonant roots are:

- $CM > L$ = left
- $T > T >$ = sweep with a broom
- $CXWH$ = bow.

To summarize, the main contribution of this work is ...

Related Work

An early source that expands on this "consonant-gate-root" evolution is the Kabbalistic "Book of Creation" (traditionally claimed to be authored by the patriarch Abraham). The book, which describes the formation of language in the Platonic concepts of celestial abstraction, examines in detail the formation of language in stages, starting from the stage of the formation of letters, moving on to the stage of combining pairs of letters into gates and ending with the stage of combining triplets of letters (or actually pairs of form $(gate, letter)$ = $((letter, letter), letter)$) into roots. Due to the mathematical nature of the Hebrew paradigms, known to anyone who has had the experience of learning this language, this combinatorial description just suggests itself.

A recent article closely related to our discussion is the work of A. Zeldin Zeldin (2021), in which he discusses the evolution of Hebrew and other Semitic roots and presents many interesting aspects of it.

The main idea of this article

The idea in a nutshell

The main idea of our experiment is to either prove or disprove (or at least not prove) the hypothesis of the gate-root evolution, which is the second part of the whole hypothesis of the phoneme-gate-root evolution, by using statistical means and, in particular, advanced Machine Learning techniques. The hypothesis of gate-to-root evolution would be confirmed if we achieved a high level of statistical significance in finding high intra-cluster semantic similarity when clustering the words of the target corpus by their gates. We now list the steps of the experiment.

1. Find the appropriate target corpus. The corpus must be equipped with a word-to-root mapping (below **corpus**).
2. Find the appropriate word embedding algorithm for training a language model on the target corpus for measuring semantic similarity between words. The algorithm

must be effective for small corpora (the Hebrew Bible is a relatively small corpus, at least in terms of training advanced language models) and trainable in a reasonable time by using computational resources available to us (below **word_to_embedding**).

3. Find or develop a method for obtaining a gate given a root (below **root_to_gate**).

4. Now, run Algorithm that picks a representative word for every root, groups these representative words by their gates, and calculates the intra-group mean of pairwise semantic similarity for each group.

5. Compare the global "mean of means" of the pairwise semantic similarities from the previous bullet to those measured with random groups of words of the same size.

6. Calculate the T-test value between the gate-based groups and the random groups to ensure better distribution.

We now review every step in detail:

The corpus

We used the BHSA corpus Winther-Nielsen (2019), (Biblia Hebraica Stuttgartensia Amstelodamensis), which is an open Hebrew Bible Database using text-fabric. It contains the text of the Hebrew Bible, augmented with linguistic annotations, compiled by the Eep Talstra Centre for Bible and Computer, VU University Amsterdam.

Before passing the corpus to our pipeline, we ran the following preprocessing:

- First, we filtered out words other than verbs, nouns, adverbs, or adjectives because the other parts of speech carry less contextual information.

- We then normalized the words by converting them to their lexemes.

The word embedding algorithm

As a word embedding algorithm, we used **word2vec**. This algorithm works in a reasonable amount of time and provides good contextual relations between the resulting word embeddings, even when trained on small corpora (TODO: refer to Moshe's thesis).

The root_to_gate method

Inferring the gate from a 3-letter root works as follows:

1. Look for $'WJHN < MT'$ (the order matters). If found - delete the first occurrence and return the result.

2. Otherwise - delete the third letter and return the result.

The rationale behind this logic is being the $'H > MNTJW'$ letters unstable in Hebrew. These letters are often used as "reading servants" (vowel markers) or as prefixes or suffixes. The specified order $'WJHN < MT'$ determines the level of instability from higher to lower and therefore the most unstable found letter should be removed first. If no unstable letter was found - remove the third one, assuming the tendency of generating new words by adding suffixes. The last assumption will be discussed in Section .

The main algorithm

Below is the listing of the our main algorithm in a python-like pseudo-code style:

```
##
# First, train the language model on the
# target corpus

corpus = BHSA()
model = word2vec.train(corpus.sentences)

##
# This auxiliary method receives an array of
# words and returns the array of pairwise
# similarities between them

def get_pairwise_similarities(words, model):
    similarities = []
    for i in range(len(words)):
        for j in range(i + 1, range(len(words
            ))):
            similarities.append(model.
                get_similarity(words[i],
                words[j]))
    return similarities

##
# This is the main method of the algorithm

def main():
    gate_roots = {}

    # group the words by gate
    # then by root
    for w in corpus.words:
        gate = root_to_gate(w.root)
        if gate_roots [gate] is none:
            gate_roots[gate] = {}
        if gate_roots[gate][w.root] is none:
            gate_roots[gate][w.root] = []
        gate_roots[gate][w.root].append(w)
    groups = {}

    # for each gate, generate a list of
    # words, randomly selecting a single
    # word per root, and in parallel,
    # generate another list of random
    # words, of the same size
    for g in gate_roots:
        if groups[g] is none:
            groups[g] = {'found': [], 'random
                ': random.select(corpus.words
                , len(gate_roots[g]))}
        for r in gate_roots[g]:
                groups[g]['found'].append(
                    random.select(r, 1))

    # evaluate the pairwise similarities
    # for both the groups and then evaluate
    # their mean values
    found_similarities = [
        get_pairwise_similarities(groups[g]['
        found'], model) for g in groups]
    founds_means = [mean(s) in
        found_similarities]
    random_similarities = [
        get_pairwise_similarities(groups[g]['
```

```
    random'], model) for g in groups]
random_means = [mean(s) in
    random_similarities]

# finally return the global means and the
    T test value
return mean(found_means), mean(
    random_means),
    t_test(found_similarities,
        random_similarities)
```

Results

The average distance between the words sampled for the same gates is 1.019, while the average distance between randomly sampled words is 1.068. These differences are statistically significant ($p < 0.05$; using a student t-test). Therefore, we can conclude that the semantics of words with similar gates are closer to each other than random words.

Table 1: The number of root members of the different gates.

Roots of gate	Number of Gates	Average Distance
2	42	0.99
3	33	1.02
4	19	1.08
5	9	1.00
6	1	0.90
7	2	1.09
Total average:		1.019

Table 2: Example of two word with the same gate ($> C$) but different root. The distance between the two vectors that represent those words

Hebrew	English	The Root	The Gate
אֵשׁ	fire	$> C$	$> C$
אִשָּׁה	woman	$> NC$	$> C$
The distance compared to the average:			135%

Table 3: Example of two word with the same gate ($> C$) but different root. The distance between the two vectors that represent those words

Hebrew	English	The Root	The Gate
נָטָף	balsam	NVP	VP
טַף	babies	VNP	VP
The distance compared to the average:			137%

Table 4: Example of two word with the same gate (FB) but different root.

Hebrew	English	The Root	The Gate
שָׂבָע	satisfied	$FB <$	FB
שִׂיבָה	old age	FJB	FB
The distance compared to the average:			58%

Table 5: Example of two word with the same gate ($> C$) but different root. The distance between the two vectors that represent those words

Hebrew	English	The Root	The Gate
שַׁאֲנָן	tranquil	$C > H$	$C >$
שְׁאוֹן	rush	$C > N$	$C >$
The distance compared to the average:			40%

Conclusions

Future work

Optimize the algorithm by changing the underlying assumptions

In the course of the experiment, we had to proceed from the assumptions that seemed most likely. However, these assumptions are not absolute, and we can try to improve the results by questioning them and trying new directions, and examining alternative evolutionary patterns.

For example, the *root_to_gate* method can be modified to assume that if no unstable letters are found, the letter to be removed to find the gate is not necessarily the last one. If we refer back to the PR gate, we can assume that roots like P>R, PTR, or P<R evolved from this gate, although this evolutionary pattern adds a letter to the middle of the gate, not to the end. Moreover, sometimes the first letter of a root may be claimed to come from a prefix. At least one example of such a formation is widely known - the TRM root very clearly originates from the RM gate and the T prefix.

A second evolutionary pattern worth examining is anagram/metathesis - rearranging the letters of a given word to form a new word. It is obvious in many cases. For example, the connection between the roots PRZ and PZR or between KBF and KFB is obvious to every native Hebrew speaker.

More patterns to be examined are assimilation, voicing voiceless phonemes or unvoicing voiced phonemes. As we already mentioned, the close relationship between QY and GZ seems to be obvious. A comparison between QYY and GZZ or between QYR and GZR can be very compelling.

Finding larger corpora

Using a larger corpus may improve results. We may want to either add more root annotations to BHSA (currently only

about a quarter of words have root annotations) or try to use the Modern Hebrew corpus, even though it may be less efficient in finding the right contextual relationships. Using Tannaic Hebrew and Talmudic Hebrew is another way of expanding the corpus.

Testing more etymological hypotheses

An investigation of etymological phenomena like gate-to-root evolution may be our next task. For example, we may want to investigate the formation of square roots (roots made up of four consonants). A brief overview of square roots can highlight the following patterns of formation:

- Gate duplication, like in the GLGL, DRDR, or VMVM roots.
- A combination of two different gates. For example, the rare biblical root XCML, which denotes a shade of light and is used in modern Hebrew to denote electricity, is sometimes explained as a combination of two gates: XC (to be silent) + ML (to speak).
- Using paradigms from other Semitic languages to create Hebrew roots. For example, the quadriconsonant root CKLL developed from the triconsonant root KLL using an Akkadian paradigm of prepending C to the root (this pattern has become widespread in modern Hebrew, as you can see in the roots CXZR, CXPL, and others).
- Adoption of roots of foreign languages, for example, the root HNDS is of Persian origin.

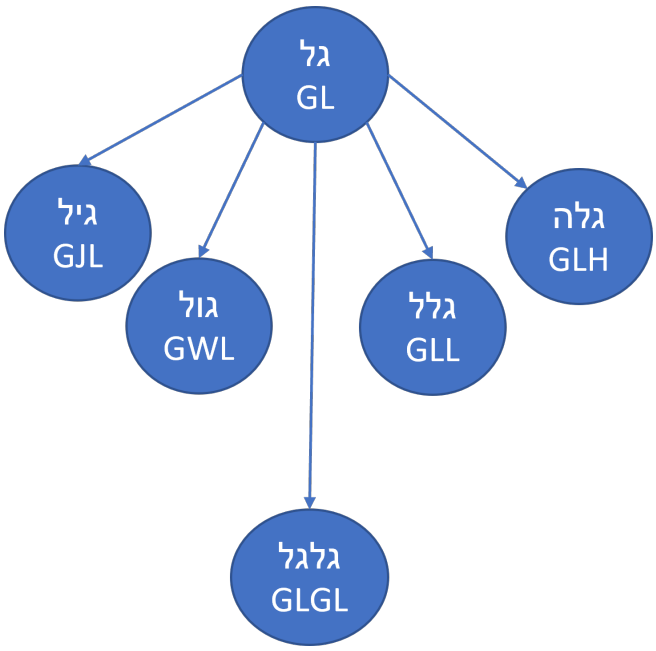

Figure 2

Acknowledgments

sdfasfsaf

References

Huehnergard, J. (2011). Appendix c: Historical akkadian phonology. In (p. 586-594). Leiden, The Netherlands: Brill. doi: https://doi.org/10.1163/9789004369160_046

Winther-Nielsen, N. (2019). Interfacing the hebrew bible: past, present and future applications for the bhsa. *HIPHIL Novum*, *5*(2), 143-152.

Zeldin, A. (2021). The semitic root evolution (cultural and historical aspect). *Uchenye Zapiski Kazanskogo Universiteta. Seriya Gumanitarnye Nauki*, *163*(4-5), 47–66.

Research methods and experimental system

In the paper on image processing, we introduce the Blind Autonencoder For Fairness and Objectiveness (BAFO). In the training phase, a labeled image is inserted as an input to BAFO. In the UTKFace dataset the images size is 56x56x3 (RGB), and the images are tagged by age, gender, and ethnicity. We note that the labels include the information that we intend to later conceal. The image is then compressed by the encoder into two vectors representing the parameters that describe the image, one vector for the mean and the other for the standard deviation. After sampling the latent vector from the mean and the standard deviation, it is concatenated to the information BAFO is concealing, the age, the gender and the ethnicity. The latent vector is then decoded back to an image with a size similar to the input image.

In the last paper, we utilized Word2Vec for word embedding.

Word2Vec is a popular technique in natural language processing (NLP) that aims to represent words as dense vectors in a high-dimensional space. There are two main architectures within Word2Vec: Continuous Bag-of-Words (CBOW) and Continuous Skip-Gram. CBOW predicts a target word based on its context, while Skip-Gram predicts the context words given a target word. In CBOW, the model learns to predict the target word by summing up the embeddings of the context words and passing them through a neural network. On the other hand, in the Skip-Gram architecture, the model predicts the context words based on the target word. Both CBOW and Skip-Gram use a shallow neural network with a hidden layer to learn the word embeddings. These word embeddings capture semantic and syntactic relationships between words, allowing for various NLP tasks such as word similarity, analogy detection, and even capturing complex linguistic patterns. Word2Vec has revolutionized NLP by providing efficient and effective word representations that capture the contextual and semantic information of words in a compact vector space.

We chose to utilize the Skip-Gram architecture as it yields superior results.

אוניברסיטת אריאל בשומרון

פתרון בעיות מורכבות באמצעות למידה עמוקה

חיבור זה הוגש כחלק מדרישות התואר
"דוקטור לפילוסופיה"

מאת

אביגיל שטקל

עבודה זו נכתבה בהנחיית פרופסור עמוס עזריה

מוגש לסנאט אוניברסיטת אריאל בשומרון

03/07/2023

תקציר

למידה עמוקה התגלתה כגישה רבת עוצמה לפתרון בעיות מורכבות בתחומים מגוונים. היכולת שלה ללמוד דפוסים וייצוגים מורכבים ישירות מנתונים גולמיים שינתה את תחום הבינה המלאכותית. בעבודה זו, אנו חוקרים את היישום של למידה עמוקה כדי לטפל במספר בעיות שונות בתחומים שונים.

ראשית, אנו חוקרים את השערת גולדבך, בעיה פתוחה מפורסמת במתמטיקה. ההשערה קובעת שכל מספר זוגי גדול משניים ניתן לבטא כסכום של שני מספרים ראשוניים. על ידי פיתוח מודל למידה עמוקה, אנו שואפים לחזות את מספר מחיצות גולדבך עבור מספר זוגי נתון. באופן מפתיע, המודל שלנו עולה על האומדנים האנליטיים הקיימים, מבלי לדרוש פירוק ראשוני של המספר. התקדמות זו מקרבת אותנו לפתרון אחת הבעיות הפתוחות הבולטות בעולם המתמטיקה.

לאחר מכן, אנו מתעמקים בתחום השמירה על הפרטיות. ההתמקדות שלנו היא בהסתרת מידע אישי מתמונות תוך שמירה על תכונות רלוונטיות אחרות. כדי להשיג זאת, אנו מציעים מודל מקודד אוטומטי (VAE) שאומן על מערך נתונים הכולל תוויות של המידע שיש להסתיר, כגון מין או גיל. על ידי הוספה ישירה של התוויות הללו לוקטור הסמוי שנדגם של VAE, אנו מבטיחים שהמודל לא לומד או מאחסן מידע הסמוי. השיטה שלנו מסתירה בהצלחה מידע פרטי תוך שמירה על מאפייני תמונה אחרים, כפי שהוכח במחקרי משתמשים. גישה זו טומנת בחובה הבטחה לשמירה על הפרטיות ויכולה למתן הטיה במערכות הנשענות על ניתוח תמונה.

לבסוף, אנו חוקרים את האבולוציה של השפה העברית, במיוחד את המעבר מאטימונים דו-עיצוריים (שערים) (C2) לשורשים תלת-עיצוריים (שורשים) (C3). באמצעות קורפוס BHSA, מערך נתונים ערוך ידנית של התנ"ך העברי, ושיטת Word2Vec לייצוג משמעות סמנטי, אנו חוקרים את השערת האבולוציה משערים C2 לשורשים C3 בעברית המקראית. הניתוח שלנו מגלה שמילים בעלות שורשים שונים, שמקורן ככל הנראה מאותם אטימונים C2, יוצרות אשכולות קרובים יותר בהשוואה לקבוצות מילים אקראיות. ממצאים מובהקים סטטיסטית אלה תומכים מאוד בהשערה ושופכים אור על ההתפתחות ההיסטורית של המורפולוגיה השמית.

על ידי יישום טכניקות למידה עמוקה לבעיות הנבדלות הללו, אנו מדגימים את הרבגוניות והיעילות של גישה זו בהתמודדות עם אתגרים מורכבים. מהשערות מתמטיות ועד לשמירה על פרטיות ואבולוציה לשונית, למידה עמוקה מציעה מסגרת רבת עוצמה לפתרון בעיות וקידום ההבנה שלנו במגוון רחב של תחומים