AI

MDPI

Article

The Combinatorial Fusion Cascade as a Neural Network

Alexander Nesterov-Mueller

Institute of Microstructure Technology, Karlsruhe Institute for Technology, Hermann-von-Helmholtz-Platz 1, 76344 Eggenstein-Leopoldshafen, Germany; alexander.nesterov-mueller@kit.edu

Abstract: The combinatorial fusion cascade provides a surprisingly simple and complete explanation for the origin of the genetic code based on competing protocodes. Although its molecular basis is only beginning to be uncovered, it represents a natural pattern of information generation from initial signals and has potential applications in designing more-efficient neural networks. By utilizing the properties of the combinatorial fusion cascade, we demonstrate its embedding into deep neural networks with sequential fully connected layers using the dynamic matrix method and compare the resulting modifications. We observe that the Fiedler Laplacian eigenvector of a combinatorial cascade neural network does not reflect the cascade architecture. Instead, eigenvectors associated with the cascade structure exhibit higher Laplacian eigenvalues and are distributed widely across the network. We analyze a text classification model consisting of two sequential transformer layers with an embedded cascade architecture. The cascade shows a significant influence on the classifier's performance, particularly when trained on a reduced dataset (approximately 3% of the original). The properties of the combinatorial fusion cascade are further examined for their application in training neural networks without relying on traditional error backpropagation.

Keywords: combinatorial fusion cascade; neural networks; spectral analysis; transformer; training without backpropagation

Academic Editor: Demos T. Tsahalis

Received: 11 December 2024
Revised: 20 January 2025
Accepted: 22 January 2025
Published: 24 January 2025

Citation: Nesterov-Mueller, A. The Combinatorial Fusion Cascade as a Neural Network. *AI* **2025**, *6*, 23. https://doi.org/10.3390/ai6020023

Correction Statement: This article has been republished with a minor change. The change does not affect the scientific content of the article and further details are available within the backmatter of the website version of this article.

1. Introduction

A hypothetical chemical reaction called the combinatorial fusion cascade (CFC) has recently been proposed to explain the origin of the standard genetic code (SGC) [1]. Figure 1a shows half of the CFC to demonstrate how homogenous complementary coding triplets fused combinatorially to the protocodes at the first stage of the CFC and, accordingly, the combinatorial fusion of protocodes results in the formation of the SGC. With unexpected simplicity, the CFC explains the distribution of amino acids among codons in the SGC, the appearance of stop codons, and some deviations from the standard genetic code in mitochondria that were not possible with any other approaches [2].

This cascade–combinatorial fusion principle has inspired several innovative studies to explore the molecular basis for the different stages of the CFC closely connected with the origin of life. For example, peptide synthesis on short complementary RNA fragments has been demonstrated [3]. M. Yarus, a renowned expert in the field of genetic code evolution, considered fusion as a process yielding hybrid routes to the SGC and summarized that efficient evolution can occur when code fusion is less prone to failure [4]. The interactions between short RNA fragments and elongating peptide chains have been studied with the assumption of competing protocodes [5]. The fact that the genetic code originated not from chaotic combinations of bases but from homogenous triplets was attempted to be explained

by reducing strand entropy during non-enzymatic replication [6] or via the self-assembly of pure adenosine and pure uridine monophosphate strands on Montmorillonite clay [7].

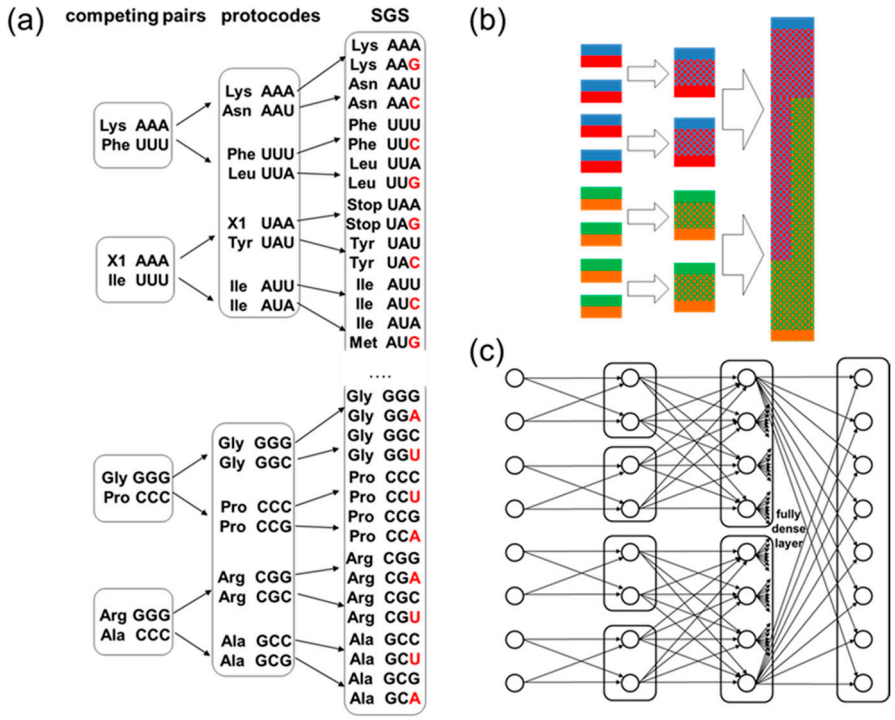

Figure 1. Combinatorial fusion cascade. (**a**) Half of the CFC is presented as a cascade of coding codons and the corresponding amino acids. The removal of the amino acid X1 from the cascade resulted in the emergence of two stop codons, UAA and UAG, in the SGC. The entities are highlighted with frames. The Crick–Watson substitutions that occurred after the fusion of the protocodes to the SGC are highlighted in red: A ↔ G, C ↔ U. (**b**) The mixing of bases in the CFC is represented as mixing of colors at different stages of the cascade. Red stands for A, blue for U, green for G, orange for C. At each stage of the CFC, the mixing of bases in codons occurs. Unmixed codons from the previous stage of the cascade make up ¼ of all codons in the new entity. (**c**) The schematic representation of the CFC for its realization as a neural network.

The CFC exhibits two key properties. First, it involves the cascading expansion of triplets due to the incorporation of new bases into the initially low-entropy triplets during transitions between stages. Second, during these transitions, competing codes interact to determine which code becomes dominant and acquires new triplets, while the other becomes recessive. These cascading and self-organizing behaviors inspired the design of the combinatorial fusion cascade architecture in multilayer neural networks. Viewing the CFC as a neural network reveals it as a cascade of interconnected multimodal sub-architectures, merging into progressively complex structures based on the same multimodal principle. This process culminates in a global layer that unifies all modalities into a single encoding layer.

The principles of fusion explored in the CFC align with recent advancements in data-driven fusion frameworks in neural network research. For example, the iPSNeXt architecture [8], designed for classifying hiPSC-derived endothelial cells in photomicrographs, employs fusion to integrate features from edge detection and texture operators into a unified representation. This feature fusion enhances performance metrics such as precision and sensitivity while reducing computational complexity. Similarly, in multi-class skin lesion classification, the CSLNet framework [9] demonstrates the effectiveness of combining complementary hierarchical features through convolutional block fusion. These studies underscore the value of fusion mechanisms in improving robustness, addressing

class imbalances, and enhancing generalization. These innovations offer insights into how fusion principles can be applied in neural networks to achieve superior performance with fewer trainable parameters and greater computational efficiency.

Previously, the combinatorial cascade architecture did not attract practical attention, since in modern neural networks such as in image recognition tasks; layer convergence from a large number of nodes to a small number of outputs responsible for specific labels is required. This is necessary for the efficient adjustment of neural network weights through the error backpropagation method [10]. Therefore, we deliberately do not address image recognition tasks, which are dominated by well-established and highly effective convolutional neural networks (CNNs). These networks operate on architectures that are conceptually opposite to the cascading idea in CFC. CNNs aim to reduce connections by focusing on local feature integration. In contrast, CFC maintains or even increases the number of nodes as it advances to higher stages in the cascade.

Modular neural networks are recognized for their precise handling of subtasks, robustness, and fault tolerance [11,12]. As a distributed modular architecture, the CFC may therefore be particularly relevant to the development and application of artificial intelligence. Modularity also underlies human consciousness when visual, auditory, and other sensory signals are integrated by the neocortex into a single perception of the surrounding world.

This work explores the possibility of embedding the CFC into deep neural networks instead of sequential fully dense layers. Fully dense layers play a significant role in attention-based architectures, such as transformers [13]. However, their training requires substantial computational resources and energy. Therefore, the search for more efficient neural architectures has become a pressing challenge. Additionally, the CFC, as a natural pattern underlying the origin of life, may be of interest for developing training methods that do not rely on error backpropagation.

2. Unique Characteristics of the CFC

The combinatorial cascade resembles growing m-ary trees ($m \in \mathbb{N}$), where branches form increasingly broad connections, ultimately resulting in a densely connected "crown". The CFC consists of 16 intertwined quaternary trees. Table 1 shows the SGC elements at different stages of the cascade.

Table 1. Cascade features of the natural CFC at different stages. Entities are sets of codons with different degrees of entropy: complementary pairs, protocodes, or the SGC.

	Competing Pairs	Protocodes	SGC
Number of codons	4	16	64
Number of complementary pairs	2	8	32
Number of competing entities	8	4	1

An important feature of the cascade is the complementarity of codons in entities. In particular, complementarity between hydrophobic and hydrophilic amino acids is built into SGC [14].

In the SGC, fewer entropic codons are preserved and passed on to subsequent stages of the cascade (Figure 1b; the coding of different databases is presented in different colors): the inherited codons of the initial complementary pairs constitute ¼ of the codons at the protocode stage. Similarly, all codons from the protocode stage are inherited in the SGC and constitute ¼ of it. From the point of view of neural network connectivity, some signal patterns are transmitted in the cascade without change.

Another feature of the CFC is that the competition between amino acids (features) for codons at the initial stage of the cascade turns into the degenerative coding of amino acids.

If no new amino acids enter the cascade tree, then four different codons code for one amino acid in the SGC.

3. Methods

3.1. Implementation of the CFC in Neural Networks

The implementation of a combinatorial cascade in neural networks is shown schematically in Figure 1c. This cascade ends with a fully dense matrix.

The dynamic mask method is used to simulate a combinatorial cascade. The entire network can be represented as follows:

$$\boldsymbol{x}^{(L)} = f\left(\left(\boldsymbol{x}^{(L-1)}\boldsymbol{W}^{L} \odot \boldsymbol{M}^{(L)}\right) + \boldsymbol{b}^{(L)}\right). \tag{1}$$

Here, $\boldsymbol{x}^{(L-1)}\boldsymbol{W}^{L}$ is the standard matrix multiplication, f is the activation function, $\odot\boldsymbol{M}^{(L)}$ is the element-wise product with the dynamic mask $\boldsymbol{M}^{(L)}$, and $\boldsymbol{b}^{(L)}$ the bias vector for layer L ($L = 1, 2, \ldots, L_{max}$).

A general form of $\boldsymbol{M}^{(L)}$ is expressed as a sum of diagonal matrices:

$$\boldsymbol{M}^{(L)} = \sum_{k=0}^{d} \boldsymbol{D}_k, \tag{2}$$

where $\boldsymbol{D}_0$ is the main diagonal of $\boldsymbol{M}$ filled with ones, and $\boldsymbol{D}_k$ (for $k = 1, 2, \ldots, d$) is the k-th diagonal with ones on the k-th off-diagonal both above and below the main diagonal. The all-ones matrix $\boldsymbol{M}^{(L)}$ corresponds to the full dense connections.

The final output $\boldsymbol{y}$ after L_{max} layers is calculated as follows:

$$\boldsymbol{y} = \boldsymbol{x}^{(L_{max})}. \tag{3}$$

In a combinatorial cascade dynamic matrix of size $n \times n$, the sparsity increases progressively layer by layer, with the number of non-zero elements k growing at each combinatorial cascade stage. It corresponds to the number of diagonals d_L filled with non-zero elements: $k = d_L \times n$. If the number of non-zero diagonals increases progressively, for example, like a quaternary tree: 2, 8, 32, 128, etc., the computational complexity of matrix multiplication C_L is estimated as follows:

$$C_L = O(d_L \times n) = O\left(2^{2L-1} \times n\right). \tag{4}$$

The total computational complexity C_{CFC} across all L_{max} layers for the CFC network is as follows:

$$C_{CFC} = n\sum_{L=1}^{L_{max}} O\left(2^{2L-1}\right) \tag{5}$$

3.2. CFC Laplace Matrix

The Laplacian matrix $\boldsymbol{L} \in \mathbb{R}^{n\times n}$ is defined based on the graph's adjacency matrix $\boldsymbol{A}$ and degree matrix $\boldsymbol{D}$:

$$\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}. \tag{6}$$

Here, degree matrix $\boldsymbol{D}$ is a diagonal matrix with elements D_{ii} that represent the number of edges connected to node i, and adjacency matrix $\boldsymbol{A}$ is a symmetric matrix with elements A_{ij}:

$$A_{ij} = \begin{cases} 1, \text{ if there is an edge between nodes } i \text{ and } j, \\ 0, \text{ otherwise.} \end{cases} \tag{7}$$

The Laplacian eigenvalues λ_i and eigenvectors $\boldsymbol{v_i}$ of the Laplacian matrix are calculated using the equation:

$$(\boldsymbol{L} - \lambda_i \boldsymbol{I})\boldsymbol{v_i} = 0, \quad (8)$$

where $\boldsymbol{I}$ is the identity matrix.

3.3. Python Codes

Python (version 3.12.4) codes were used to implement the main equations and generate the figures based on these equations (see provided codes using the link in the Data Availability Statement). The Transformer-based text classifier was compiled with the Adam optimizer, a binary cross-entropy loss function, and accuracy as the evaluation metric. The Adam optimizer was chosen for its efficiency in handling sparse gradients and its proven effectiveness across a wide range of deep learning tasks. The optimizer was initialized using TensorFlow's default configuration.

4. Results

4.1. Computational Complexity of the CFC

The total computational complexity C_{CFC} (5) grows exponentially with the increasing number of diagonals with layer depth. However, it is still much smaller than the total matrix calculation complexity $C_{FD} = O\left(L \times n^2\right)$ for fully dense matrices in earlier layers. Figure 2 estimates the number of stages of a quaternary combinatorial cascade, in which the total complexity of matrix calculation reaches the total complexity of matrix calculation of a fully dense neural network. Since the number of layers is an integer, the function has a step-like character. It is characteristic that the number of CFC layers, at which the computational complexity of the CFC reaches the computational complexity of the corresponding fully dense neural network, can remain constant in a large range of the number of neurons. In particular, the quaternary combinatorial cascade, consisting of 7 layers, reaches the complexity of a fully connected neural network in a wide range from 500 to 1000 neurons in a layer.

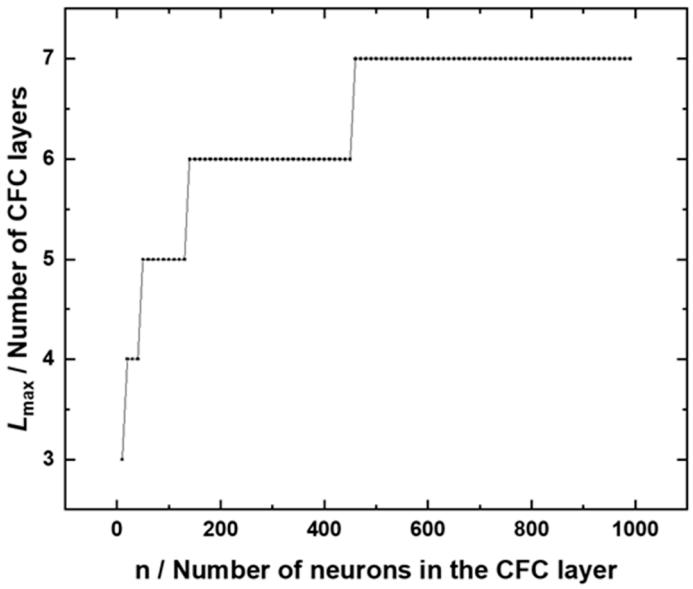

Figure 2. The maximum number of layers in a quaternary CFC necessary to achieve a total computational complexity equal to the total computational complexity of a fully dense neural network with the same number of layers, depending on the number of neurons in the layers. In this example, the weight matrices are square with dimensions $n \times n$.

4.2. Spectral Properties of the CFC Laplace Matrix

To identify the spectral properties of the CFC, we examine a neural network with four layers, similar to the one shown in Figure 1c, containing 32 neurons in each layer. The connectivity of the CFC network is described in Table 2.

Table 2. Connectivity of the CFC network.

Layers	Algorithm of Connection Design
1 and 2	The first two neurons in layer 1 connected with two neurons in layer 2 with identical indexes via full dense connections.The next two neurons in layer 1 connected with two neurons in layer 2 with identical indexes via full dense connections. etc.
2 and 3	The first 8 neurons in layer 2 connected with 8 neurons in layer 3 with identical indexes via full dense connections.The next 8 neurons in layer 2 connected with 8 neurons in layer 3 with identical indexes via full dense connections. etc.
3 and 4	All neurons in layer 3 connected with all neurons in layer 4 via full dense connections.

The eigenvectors and eigenvalues of the Laplace matrix for this neural network are compared with those of a fully connected four-layer neural network of the same size.

Figure 3 shows the Laplacian spectrum and some Laplacian eigenvectors for a fully dense neural network with four layers. Fiedler eigenvector [15] (the first nontrivial eigenvector with index i = 1) shows the global structure of the neural network by identifying four clusters corresponding to the four layers. Using the analogy of connected vibrators, the behavior of the neural network's Laplacian eigenvectors as their eigenvalue increases can be interpreted as the emergence of more energetic oscillatory vibration modes. The entire spectrum of Laplacian eigenvalues of a fully connected network is divided into two qualitatively different groups. At low vibration energies, the eigenvectors are localized in the region with smaller edges: this is the input and output layer. As the vibration energy of the network increases, the localization of the Laplacian eigenvectors moves to the middle of the neural network with a higher concentration of connections between nodes.

In the case of the CFC (Figure 4), most of the eigenvectors have a similar complex beat shape as in the case of the fully connected neural network in Figure 3. The localization of these eigenvectors shifts as the energy of the network oscillations increases from regions with lower to higher connectivity: input layer nodes (e.g., i = 20 and 31), first hidden layer nodes (e.g., i = 36), output layer nodes (e.g., i = 95) and second hidden layer nodes with the densest connectivity (e.g., i = 96).

The CFC network also has fundamental differences. The Fiedler vector of the CFC does not distinguish the cascade structure but divides the network into 16 clusters (4 clusters per layer). Analogs of Fiedler eigenvectors appear at the junction of cascade layers (eigenvectors with i = 33, 35 in Figure 4). It is interesting to note that Fiedler analogs also appear at maximum vibration energies of the neural network, e.g., i = 126. Clustering according to the stages of the cascade (four clusters) occurs with an energy significantly exceeding the energy of Fiedler mode. Only cascade eigenvectors, not Fiedler eigenvector, and its analogs, contrary to expectations, are distributed throughout the entire neural network.

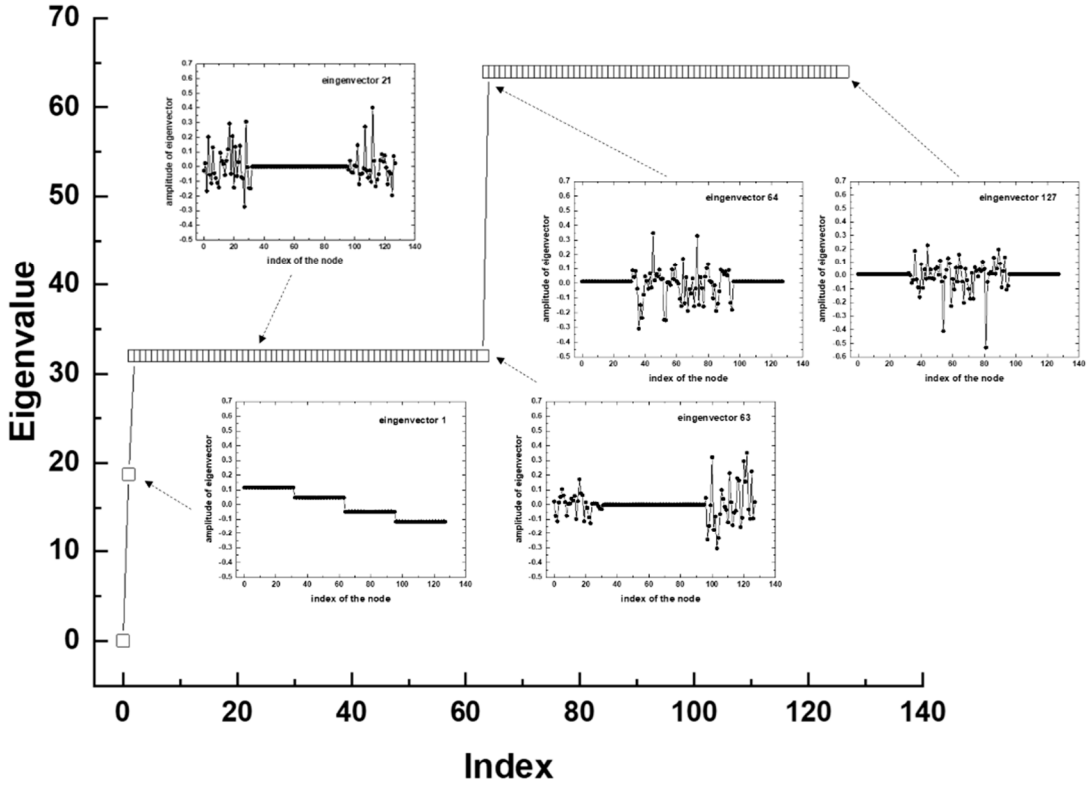

Figure 3. Laplacian eigenvalue spectrum and die eigenfunctions for indexes 1, 21, 63, 64, and 127 of a fully dense four-layer neuronal network with 32 nodes in each layer.

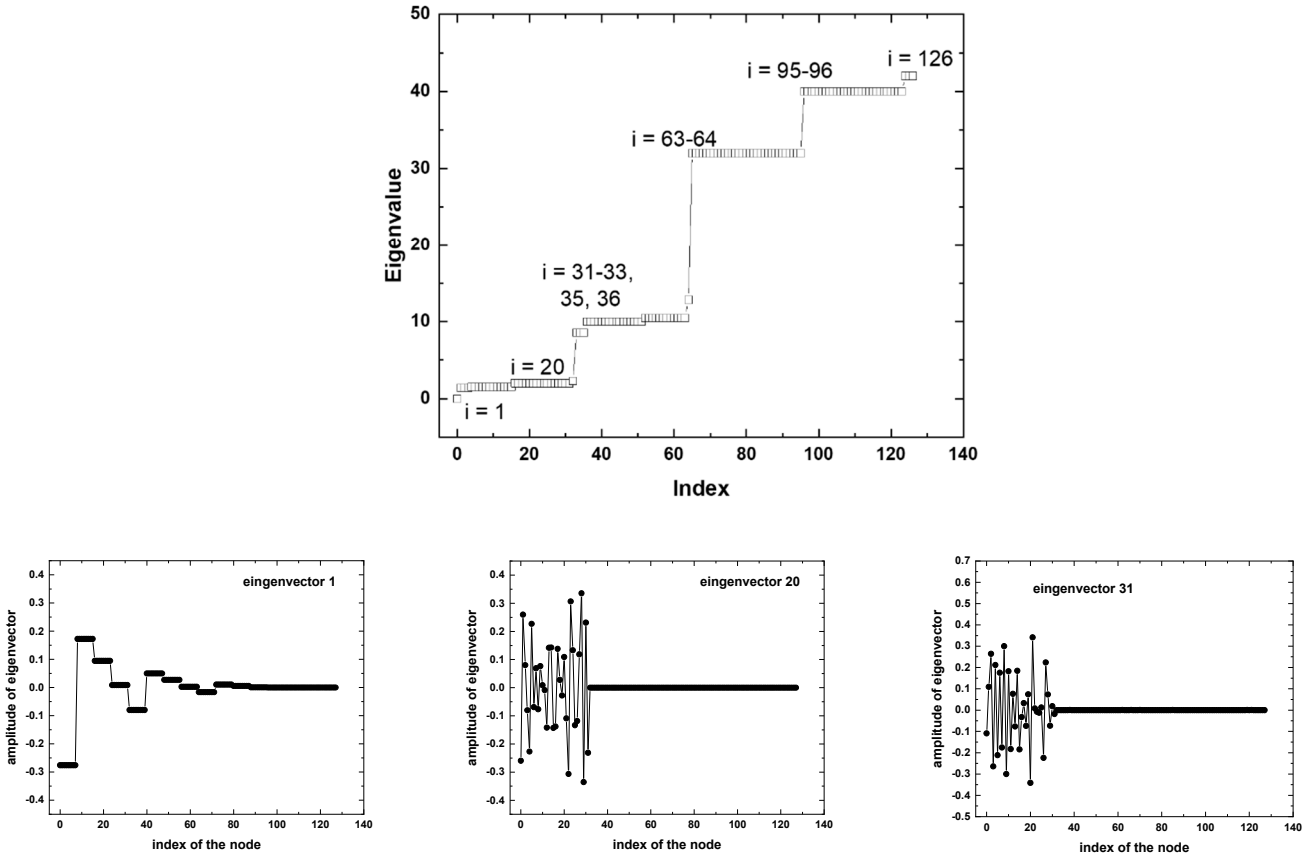

Figure 4. *Cont.*

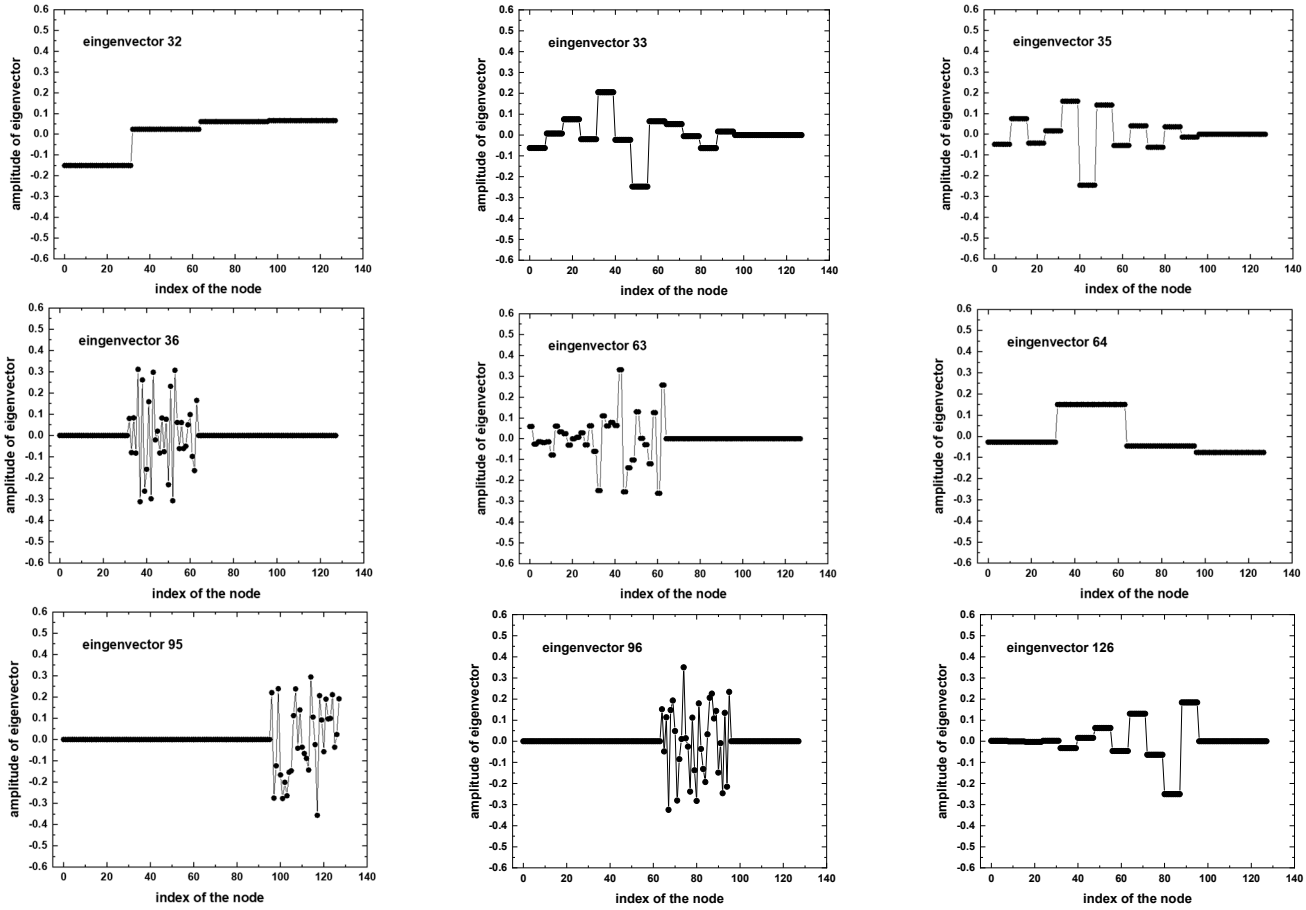

Figure 4. Laplacian eigenvalue spectrum and Laplacian eigenfunctions of four-layer CFC neuronal network with 32 nodes in each layer, with connectivity defined in Table 2.

4.3. Combinatorial Cascade Architecture in Transformers

Fully connected layers are used extensively in Transformers, both in the multi-head attention (MHA) layers and in the feed-forward network (FFN) layers. Transformer-based large-scale language models, like GPT-3, which have billions of parameters, face a significant number of operations and increase in memory requirements. This is recognized as a serious bottleneck for the further development of large language models [16]. Here, we explore how a cascade architecture could influence the operation of Transformers by integrating it into a Transformer-based text classifier.

Figure 5a illustrates the architecture of a text classifier designed to evaluate the impact of the CFC architecture on its performance. The classifier uses the standard IMDB dataset [17], an embedding layer, two sequential transformers, and an output layer for binary sentiment classification. In the embedding, 25,000 reviews are tokenized into numerical sequences, where each word is replaced by its corresponding index in a predefined vocabulary. The dimension of the embedding matrix was 10,000 $\times$ 128 (number of tokens x dimension of the model). To ensure consistent input length, sequences are padded or truncated to a fixed size. The transformer architecture incorporates key elements such as dropout layers, normalization, and residual connections (see Data Availability Statement for the Python code for the text classifier).

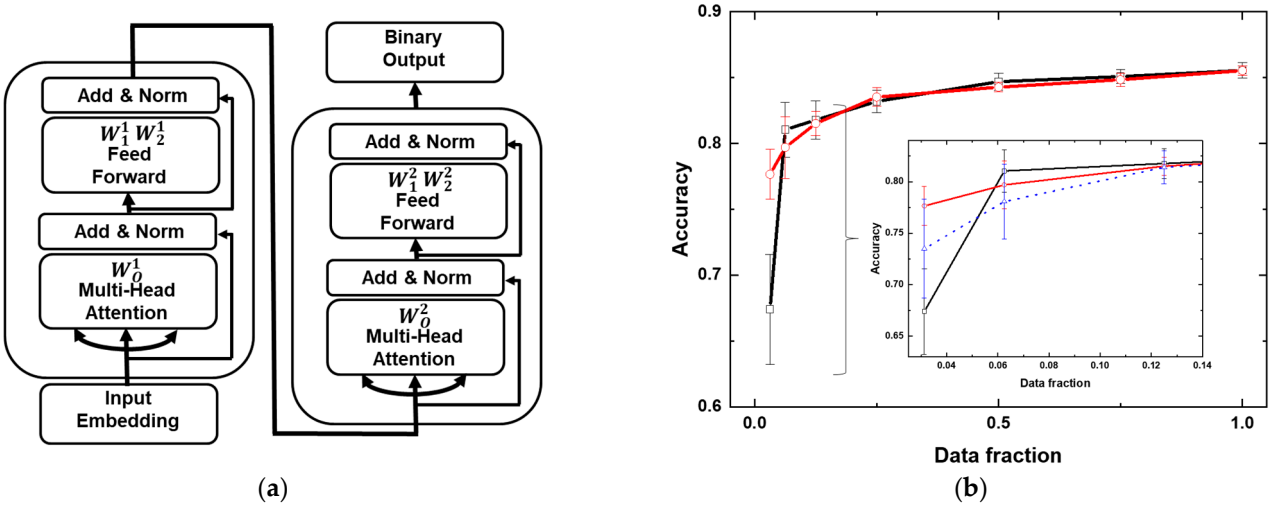

Figure 5. Text classifier with an integrated combinatorial cascade. (**a**) The basic concept of the text classifier consists of two transformers. A cascade change of parameters is realized sequentially on six fully dense matrices, starting with the output projection matrix W_O^1 of the first transformer and up to the second FFN matrix of the second transformer W_2^2 (**b**) Accuracy of the text generator with the dynamic CFC mask and cascaded initial weights (read lines with circles) and fully dense matrices (black lines with squares) after training on data of different volumes. Date fractions: 1/32, 1/16, 1/8, 1/4, 1/2, 1. Inbox includes the accuracy for the text generator with fully dense matrices and cascaded initial weights (blue dash line with triangles).

We selected the IMDB dataset for two key reasons. First, it serves as a well-known benchmark task in sentiment analysis. Second, it has only binary labels, meaning the output layer consists of a single neuron connected to a system of fully connected layers without any dimensionality reduction in the model. This ensures that the neural network is almost entirely composed of fully connected layers of the same dimensions, with no intermediate reductions until the output layer. To implement the combinatorial cascade architecture, six matrices were selected: the fully dense projection matrix $\boldsymbol{W}_O^1$ from the multi-head attention part of the first transformer, two fully dense matrices $\boldsymbol{W}_1^1$ and $\boldsymbol{W}_2^1$ from the feed-forward part of the first transformer, as well as analogous matrices from the second transformer: $\boldsymbol{W}_O^2$ from the multi-head attention part $\boldsymbol{W}_1^2$ and $\boldsymbol{W}_2^2$ from the feed-forward block. The projection matrix plays a crucial role in aggregating information from multiple attention heads into a unified representation for each token. The feed-forward block matrices are essential for transforming the input data and extracting higher-level features. To these six matrices were applied six dynamic matrices, the number of diagonals of which increased according to the binary cascade: 2, 4, 8, . . ., 128. For simplicity of implementation of the cascade architecture, the dimension of these matrices was 128 × 128. Thus, the last matrix of the cascade $\boldsymbol{W}_2^2$ remained completely dense. The application of a binary cascade resulted in a reduction of two-thirds of the weights of all selected matrices.

Based on the fact that CFC has a cascade multimodality, by analogy, we multiplied the initial weights for the entire cascade of selected matrices by a factor decreasing in a binary geometric progression: 1, 1/2, 1/4, . . ., 1/32.

Figure 5b compares the accuracy of the text classifier with full-weight matrices and that with the built-in cascade and with cascaded weight reduction as a function of the number of reviews used for training. Overall, introducing the cascade did not significantly affect accuracy. However, a notable difference emerged when the number of training samples was reduced by a factor of 1/32 (781 reviews); while the standard text classifier experienced a substantial drop in accuracy, the classifier with the built-in cascade exhibited a more gradual decline. The average accuracy difference between the two cases was nearly

10%. It is also interesting to note that the standard deviation of accuracy in the case of a cascade text classifier was reduced more than twice. The inlet of Figure 5b shows another dependence for accuracy (blue dashed line), for the case of a standard text generator with a cascade factor for initial weights. Of the three cases shown in Figure 5b, the text classifier with a built-in connectivity cascade and a cascade factor for initial weights gives the greatest accuracy in the area of significant reduction of the number of samples for training.

Reducing the embedding matrix dimension by limiting the number of tokens to 1000 (down from 10,000) during training on the full dataset did not result in a significant difference in accuracy for either the cascade or standard text classifiers. When the number of tokens was further reduced to 100, the accuracy dropped to 0.72, with no noticeable difference between the two text classifier types. This reduction in tokens can be interpreted as an experiment involving adversarial perturbations. The IMDB dataset is structured such that reducing the token count retains only the most frequently used words. Tokens outside the reduced vocabulary are replaced with the [UNK] token, which likely influences the models' ability to generalize.

In experiments with the cascade and standard text classifiers, we observed no differences between them in terms of training time or memory usage. Notably, the total number of weights in the matrices where the CFC was implemented accounted for only 4.5% of the model's total weights. The majority of the weights were located in the embedding layer and were not modified by the dynamic matrix.

5. Discussion

The combinatorial diffusion cascade is a recently discovered natural pattern that has not been previously studied mathematically. Its effective descriptive power regarding the origin of the genetic code could indicate that CFC has several special properties that are interesting for applications in machine learning.

It turned out that the cascade partitioning of the neural network is not described by the Fiedler eigenvector but is described by other vectors corresponding to more energetic excitations of the network. The Fiedler vector itself, however, divides the network into smaller fragments, and a partition similar to the Fiedler eigenvector found on other vectors with higher eigenvalues of the Laplace matrix. In contrast to the Fiedler eigenvector, for its analogs (Figure 4, eigenvectors with i = 1, 33, 35, 126) and the eigenvectors of the neural network, consisting of a sequence of fully connected matrices (Figure 3), the cascade vectors are distributed throughout the entire network (see Figure 4). Thus, the CFC architecture exhibits hybrid properties, combining localized eigenvectors, characteristic of fully connected matrices, with Fiedler and cascading eigenvectors, which have a broader distribution. This balance enables specialization through localized eigenvectors while promoting generalization and robustness to noise through the broadly distributed eigenvectors [18,19]. This makes networks with the CFC potentially more adaptable to a wide range of tasks.

A significant influence of the cascade on the network behavior was observed during training on a reduced set of samples (about 3% of the initial one). The cascade architecture, despite its small share (ca. 4.5%) of the total number of network weights, turned out to be significantly more stable, yielding an average accuracy greater by 10% and a smaller standard deviation.

The decrease in the standard deviation in the case of a text classifier with a built-in cascade could be explained by a decrease in the Lipschitz constant that controls the sensitivity of the neural network to perturbations in its parameters [20]. Since the total Lipschitz constant of a neural network is equal to the product of the Lipschitz constants for individual matrices [21], the matrix with the highest sparsity is decisive for comparing the

average deviation. The spectral norm of sparse masks is generally lower, which ensures a smaller standard deviation of accuracy in the case of random sets of initial weights.

Reducing the embedding dimension did not lead to a difference in accuracy between the standard classifier and the classifier with the built-in cascade when trained on the full IMDB dataset. This is surprising, since with an embedding matrix dimension of 100×128, the ratio of the weights of the matrices modified by the dynamic cascade becomes significant compared to the embedding weights. Probably, the number of samples is of great importance for accuracy. This property can be used for more effective strategies for training large language models.

Technically, there can be different forms of implementing a cascade on neural networks. For example, instead of a geometric progression, an arithmetic progression can be used to increase connectivity along the cascade. This could stretch the cascade to a large number of fully dense matrices of a large neural network. Alternatively, CFC with a geometric progression can be applied sequentially by dividing the architecture into multiple cascades. For example, in a transformer with 100 layers, this approach would result in 50 cascades, each spanning two transformers. It is also not at all necessary for the cascade matrices to be square, as in the test classifier given in the article.

An alternative method for constructing a CFC involves separately creating matrices with the desired sparsity and subsequently applying the backpropagation algorithm without incorporating a dynamic matrix. In general, both approaches are equivalent in their functionality. The first method is simpler from a programming perspective but introduces additional parameters in the form of matrices with unit elements. The second method requires more complex programming but eliminates the need for a dynamic matrix, thereby reducing the total number of neural parameters. In language models, the number of matrix weights affected by the dynamic matrix is significantly smaller compared to other components, such as the embedding layer. As a result, the slight increase in neural system parameters due to the introduction of a dynamic matrix has a negligible impact on training time and memory requirements.

6. What Is Wrong with Backpropagation?

This question was asked by Geoffrey Hinton, exploring the possibilities of alternative methods of training neural networks [22]. He noted that backpropagation is very effective, but fundamentally different from the work of the brain: “The brain needs a learning procedure that can learn on fly”, “without taking frequent timeouts”.

The text classifier model with the integrated CFC demonstrated higher accuracy when trained on a reduced volume of data compared to a standard model without CFC (Figure 5). At the same time, it did not provide an advantage when trained on a large volume of distorted data. This result lends itself to an intuitive interpretation: it is easier to learn from a few convincing examples than from a multitude of poorly formulated ones. It is important to note that this result cannot be generalized to the application of CFC in language models. However, a natural CFC had no alternative but to learn on the fly, without taking frequent timeouts. Here, we highlight some properties of CFC that could be particularly intriguing from the perspective of developing brain-like neural networks.

The dynamic matrix method used in this paper allows us to consider the neural network as a cascaded hyper network when the parameters of the primary network are controlled by another matrix above them [23]. Brain hyperactive network analysis has been widely applied in neuroimaging studies [24,25]. Some recent studies suggest that the brain’s complex integrative actions are assembled into the brain hypernetwork that has as fundamental components the tetra-partite synapses, formed by neural, glial, and extracellular molecular networks [26].

We have not, in this work, used a couple of interesting properties of CFC, which may be relevant for training a neural network without backpropagation. A quarter of the codes after merging competency entities are preserved. New codes formed as a result of combinatorics are divided between those already existing or newly accepted into the genetic code amino acids. In essence, this means that two modal neural networks can be combined into one layer without additional training. Alternatively, the resulting new codons (connections) can be used to integrate new recognition functions into the network.

In the genetic code, a key element is codon complementarity, a property that is preserved at all stages of the cascade. A training without backpropagation based on two forward passes, working in the same way but with opposite objectives, was demonstrated by Hinton in the context of the forward-forward algorithm [22]. In this approach, the positive pass adjusted the weights, increasing the goodness at each hidden layer, and the negative pass decreased the goodness at the same layers. Adjusting the weights using novel complementarity principles may replace the calculation of goodness to reduce computations. Complementarity implies automatic knowledge of the entire code if half of it is known.

If an efficient training method could be found on a cascade architecture, then a cascade neural network, as shown in Figure 2, could achieve significant computational complexity in just a few neural layers due to the combinatorial explosion of cascade connections.

Funding: This Project is supported by the Federal Ministry for Economic Affairs and Climate Action (BMWK) on the basis of a decision by the German Bundestag.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Python codes corresponding to the equations and figures in the text, along with a detailed description of each programming step, are available in the Zenodo repository at https://zenodo.org/records/14382353 (access date 11 December 2024).

Acknowledgments: A.N.-M. acknowledges the Open Access Publishing Fund of the Karlsruhe Institute of Technology. A.N.-M. acknowledges the use of ChatGPT for enhancing Python code to implement the formulas and neural networks in this paper.

Conflicts of Interest: The author declares no conflicts of interest.

References

1. Nesterov-Mueller, A.; Popov, R. The Combinatorial Fusion Cascade to Generate the Standard Genetic Code. *Life* **2021**, *11*, 975. [CrossRef]
2. Koonin, E.V.; Novozhilov, A.S. Origin and Evolution of the Universal Genetic Code. *Annu. Rev. Genet.* **2017**, *51*, 45–62. [CrossRef] [PubMed]
3. Müller, F.; Escobar, L.; Xu, F.; Wegrzyn, E.; Nainyte, M.; Amatov, T.; Chan, C.Y.; Pichler, A.; Carell, T. A prebiotically plausible scenario of an RNA-peptide world. *Nature* **2022**, *605*, 279. [CrossRef] [PubMed]
4. Yarus, M. The Genetic Code Assembles via Division and Fusion, Basic Cellular Events. *Life* **2023**, *13*, 2069. [CrossRef]
5. Jenne, F.; Berezkin, I.; Tempel, F.; Schmidt, D.; Popov, R.; Nesterov-Mueller, A. Screening for Primordial RNA-Peptide Interactions Using High-Density Peptide Arrays. *Life* **2023**, *13*, 796. [CrossRef]
6. Kudella, P.W.; Tkachenko, A.V.; Salditt, A.; Maslov, S.; Braun, D. Structured sequences emerge from random pool when replicated by templated ligation. *Proc. Natl. Acad. Sci. USA* **2021**, *118*, e2018830118. [CrossRef]
7. Himbert, S.; Chapman, M.; Deamer, D.W.; Rheinstädter, M.C. Organization of Nucleotides in Different Environments and the Formation of Pre-Polymers. *Sci. Rep.* **2016**, *6*, 31285. [CrossRef] [PubMed]
8. Iqbal, I.; Ullah, I.; Peng, T.Y.; Wang, W.W.; Ma, N. An end-to-end deep convolutional neural network-based data-driven fusion framework for identification of human induced pluripotent stem cell-derived endothelial cells in photomicrographs. *Eng. Appl. Artif. Intell.* **2025**, *139*, 109573. [CrossRef]

9. Iqbal, I.; Younus, M.; Walayat, K.; Kakar, M.U.; Ma, J.W. Automated multi-class classification of skin lesions through deep convolutional neural network with dermoscopic images. *Comput. Med. Imag. Grap.* **2021**, *88*, 101843. [CrossRef] [PubMed]
10. Ackley, D.H.; Hinton, G.E.; Sejnowski, T.J. A Learning Algorithm for Boltzmann Machines. *Cogn. Sci.* **1985**, *9*, 147–169.
11. Amer, M.; Maul, T. A review of modularization techniques in artificial neural networks. *Artif. Intell. Rev.* **2019**, *52*, 527–561. [CrossRef]
12. Sabour, S.; Frosst, N.; Hinton, G.E. Dynamic routing between capsules. *Adv. Neural Inf. Process. Syst.* **2017**, *30*. [CrossRef]
13. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez A., N.; Kaiser, L.; Polosukhin, I. Attention Is All You Need. *arXiv* **2017**, arXiv:1706.03762.
14. Copley, S.D.; Smith, E.; Morowitz, H.J. A mechanism for the association of amino acids with their codons and the origin of the genetic code. *Proc. Natl. Acad. Sci. USA* **2005**, *102*, 4442–4447. [CrossRef] [PubMed]
15. Fiedler, M. Algebraic Connectivity of Graphs. *Czechoslov. Math. J.* **1973**, *23*, 298–305. [CrossRef]
16. Minaee, S.; Mikolov, T.; Nikzad, N.; Chenaghlu, M.; Socher, R.; Amatriain, X.; Gao, J. Large language models: A survey. *arXiv* **2024**, arXiv:2402.06196.
17. Maas, A.L.; Daly, R.E.; Pham, P.T.; Huang, D.; Ng, A.Y.; Potts, C. *Learning Word Vectors for Sentiment Analysis*; Association for Computational Linguistics: Portland, OR, USA, June 2011; pp. 142–150.
18. Bronstein, M.M.; Bruna, J.; LeCun, Y.; Szlam, A.; Vandergheynst, P. Geometric Deep Learning. *IEEE Signal Process. Mag.* **2017**, *34*, 18–42. [CrossRef]
19. Hata, S.; Nakao, H. Localization of Laplacian eigenvectors on random networks. *Sci. Rep.* **2017**, *7*, 1121. [CrossRef] [PubMed]
20. Leino, K.; Wang, Z.; Fredrikson, M. Globally-Robust Neural Networks. *arXiv* **2021**, arXiv:2102.08452.
21. Khromov, G.; Singh, S.P. Some intriguing aspects about lipschitz continuity of neural networks. *arXiv* **2023**, arXiv:2302.10886.
22. Hinton, G. The forward-forward algorithm: Some preliminary investigations. *arXiv* **2022**, arXiv:2212.13345.
23. Chauhan, V.K.; Zhou, J.D.; Lu, P.; Molaei, S.; Clifton, D.A. A brief review of hypernetworks in deep learning. *Artif. Intell. Rev.* **2024**, *57*, 250. [CrossRef]
24. Arbabshirani, M.R.; Plis, S.; Sui, J.; Calhoun, V.D. Single subject prediction of brain disorders in neuroimaging: Promises and pitfalls. *Neuroimage* **2017**, *145*, 137–165. [CrossRef] [PubMed]
25. Jie, B.; Wee, C.Y.; Shen, D.; Zhang, D.Q. Hyper-connectivity of functional networks for brain disease diagnosis. *Med. Image Anal.* **2016**, *32*, 84–100. [CrossRef] [PubMed]
26. Agnati, L.F.; Marcoli, M.; Maura, G.; Woods, A.; Guidolin, D. The brain as a "hyper-network": The key role of neural networks as main producers of the integrated brain actions especially via the "broadcasted" neuroconnectomics. *J. Neural Transm.* **2018**, *125*, 883–897. [CrossRef] [PubMed]