What comes after transformers? – A selective survey connecting ideas in deep learning*

Johannes Schneider[1]

University of Liechtenstein, Vaduz, Liechtenstein
johannes.schneider@uni.li

Abstract. Transformers have become the de-facto standard model in artificial intelligence since 2017 despite numerous shortcomings ranging from energy inefficiency to hallucinations. Research has made a lot of progress in improving elements of transformers, and, more generally, deep learning manifesting in many proposals for architectures, layers, optimization objectives, and optimization techniques. For researchers it is difficult to keep track of such developments on a broader level. We provide a comprehensive overview of the many important, recent works in these areas to those who already have a basic understanding of deep learning. Our focus differs from other works, as we target specifically novel, alternative potentially disruptive approaches to transformers as well as successful ideas of recent deep learning. We hope that such a holistic and unified treatment of influential, recent works and novel ideas helps researchers to form new connections between diverse areas of deep learning. We identify and discuss multiple patterns that summarize the key strategies for successful innovations over the last decade as well as works that can be seen as rising stars. Especially, we discuss attempts on how to improve on transformers covering (partially) proven methods such as state space models but also including far-out ideas in deep learning that seem promising despite not achieving state-of-the-art results. We also cover a discussion on recent state-of-the-art models such as OpenAI's GPT series and Meta's LLama models and, Google's Gemini model family.

Keywords: transformers, attention, state-space models, capsule networks, survey, review, deep learning, architectures, layers

1 Introduction

Transformers are widely regarded as the driving force behind artificial intelligence, e.g., of so-called foundation models[101] such as OpenAI's ChatGPT or early models such as BERT. Transformers are at the top of most machine learning benchmark leaderboard including computer vision, speech, and natural language processing. A major advantage of deep learning is its layered, modular structure. It enables construction of models from individual components in a flexible manner. Researchers have created a large selection

* This is an extended version of the published paper by Johannes Schneider and Michalis Vlachos titled "A survey of deep learning: From activations to transformers" which appeared at the International Conference on Agents and Artificial Intelligence(ICAART) in 2024. It was selected for post-publication.

of layers, architectures, and objectives. Keeping up with the rapid developments in AI models is a difficult task.

There exist multiple reviews with a narrow focus such as large language models (e.g. [77]) and convolutional neural networks (e.g. [56]). Previous studies [2,109,19,3] with a wider focus have become dated and miss new developments such as transformers and self-supervised learning. Furthermore, no survey or novel text book such as [11] has focused on alternatives towards transformers. Multiple works have discussed particular shortcomings of transformers such as computational inefficiency, e.g., due to quadratic time complexity of (naive) self-attention [99,37,12]. Thus, there is no work that looks at recent deep learning holistically integrating general ideas in deep learning and transformers, emphasizing alternatives or ways ahead. However, taking a broader and more holistic look at different (sub-)disciplines can be highly beneficial: For instance, NLP and computer vision have reciprocally shaped one another; CNNs first emerged in computer vision and subsequently found applications in NLP, whereas transformers originated in NLP and were later incorporated into computer vision. Consequently, breaking down the barriers between disciplines proves to be highly advantageous.

This paper is driven by the goal of examining the recent advancements in deep learning from an more encompassing perspective, rather than concentrating on a specific specialized field. We contend that such an approach is crucial, especially as significant new developments have decelerated; currently, the vast majority of models employs the "old" but still mostly state-of-the art transformer architecture presented already in 2017[123].

Providing a comprehensive overview of the field is challenging, if not impossible, given the vast number of articles published annually and the constant expansion of pertinent topics. Our strategy is to choose influential works through (i) usage statistics, (ii) specialized surveys, and (iii) for transformer alternatives, we rely on public debates (as well as our own literature search). We also offer an invigorating discussion of shared design patterns across areas that have been successful.[1]

2 Overview

Figure 1 outlines the topics addressed in this survey. In Part 1, We start our discussion with the key topic of this survey, i.e., transformers. We refer readers with lack of knowledge of basic architectural components of transformers such as attention to Part 2 of this work or text books[11,29]. However, Part 2, is also recommended for more experienced readers to reflect on recent ideas in deep learning. In Part 1, we elaborate on current architectures including commercial and open-source models but also go beyond by discussing more far-out ideas in deep learning. The implementations of these ideas have often not yet shown state-of-the-art results or even near state-of-the-art results, but nevertheless provide interesting ideas that might just need further thoughts.

In Part 2, our exploration focuses on deep learning design, encompassing both objectives and training methods. Particular emphasis has been placed on work validated by

[1] This paper extends the conference paper [103] by engaging more deeply in discussing transfromer alternatives among other aspects

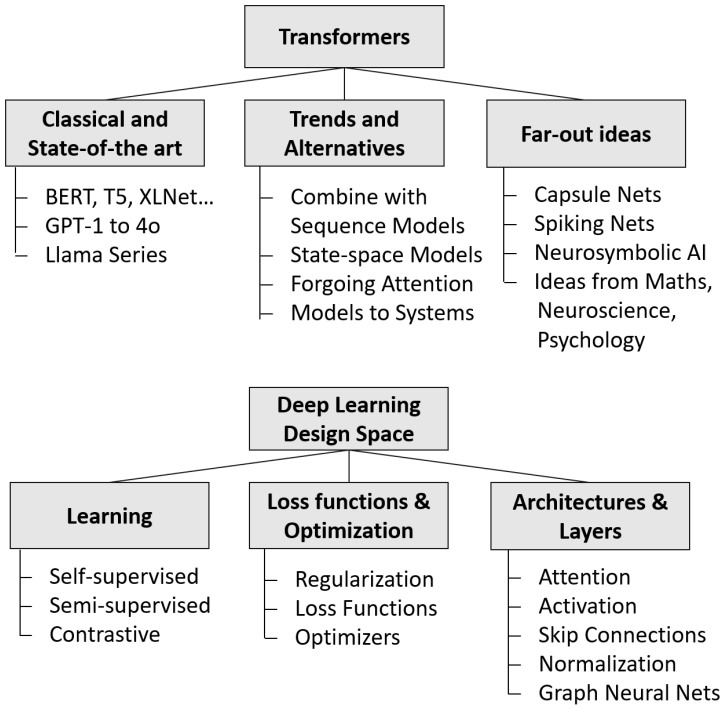

Fig. 1: Overview: Part 1 deals with transformers, Part 2 with deep learning (Lower graph based on [103])

usage metrics from the well-known platform "Paperswithcode.com." Our inclusion criteria feature impactful research published since 2016 and emerging contributions from 2020 onward that have quickly risen to prominence.

The coverage of each subject area is influenced by the volume of recent research and its fundamental importance. We omit discussions on data or computational strategies, such as data augmentation, model compression, and distributed machine learning, due to space constraints. This necessitated a discerning selection of model types, excluding significant categories like multi-modal models and autoencoders.

3 Transformers

Transformers have emerged as the leading architecture in deep learning. When paired with self-supervised training on extensive datasets, they have achieved top performance across numerous benchmarks in NLP (refer to [67] for a detailed survey) and computer vision (as outlined in [39,57]). Introduced in 2017 [123], numerous adaptations have been developed to address challenges such as computational demands and data efficiency in the original transformer model. We discuss classical and current models in Section 3.1, emerging alternatives to transformers and trends in Section 3.2 and more far-out ideas going beyond transformers in Section 3.3.

It is often stated that transformers possess lower inductive bias (e.g., compared to CNNs and RNNs), making them more adaptable. Consequently, they need a larger volume of training data to offset the reduced inductive bias. Transformers are commonly pre-trained using self-supervised learning, and oftentimes fine-tuned towards specific tasks on labeled data. Classical deep learning sequence models such as recurrent neural networks (RNNs) often exhibit a time-dependency upon inference and training, e.g., after processing of one token a hidden state must be updated before the next token can be processed. This limits parallelization and leads to other issues such as vanishing gradients for long sequences. Transformers, in contrast, allow to process the entire sequence at once using self-attention to predict the next token, i.e., they are much more parallelizable than sequential models such as RNNs. Unfortunately, naive implementation of self-attention require quadratic run-time, which makes them difficult to use for long sequences. Ideas like mixture of experts (MoE) transformer-based model have been employed to reduce computational burdens (especially during inference). MoE uses a learned routing function to select only a subset of the model for processing, which saves on computation. The idea of MoE dates back well before the year 2000 [134]. It has gained traction more recently as a means to reduce computational effort, in particular, during inference.

3.1 Classical and state-of-the-art transformer architectures

The original transformer[123], designed for NLP, utilizes an encoder and decoder similar to earlier recurrent neural networks. The architecture features multiple layers of transformer blocks, as depicted in Figure 2. Essential components include multi-head attention, layer normalization, and skip connections. For enhanced efficiency, several commercial and open-source models adopt the mixture of experts technique, which selectively computes outputs from a subset of models within an ensemble. Additionally, positional encodings and input embeddings are crucial. The absolute positional encodings PE for position pos in [123] are calculated using sinusoidal functions with varying frequencies:

$$\mathrm{PE}(pos, 2i) = \sin(pos/10000^{2i/d}) \tag{1}$$

$$\mathrm{PE}(pos, 2i+1) = \cos(pos/10000^{(2i)/d}) \tag{2}$$

where i represents the encoding dimension and d the total number of dimensions. This method was chosen because it allows relative positions, potentially as significant as absolute positions, to be a linear function of the absolute position encodings.

Bidirectional Encoder Representations from Transformers(BERT) [18]:The encoder of the transformer architecture generates contextual word embeddings through a masked language model pre-training objective and self-supervised learning. In this approach, the model is tasked with predicting randomly selected, masked input tokens based on their surrounding context, thus providing it with bidirectional information—that is, access to tokens both before and after the masked words. Unlike traditional next-word prediction methods, where no tokens following the predicted word are

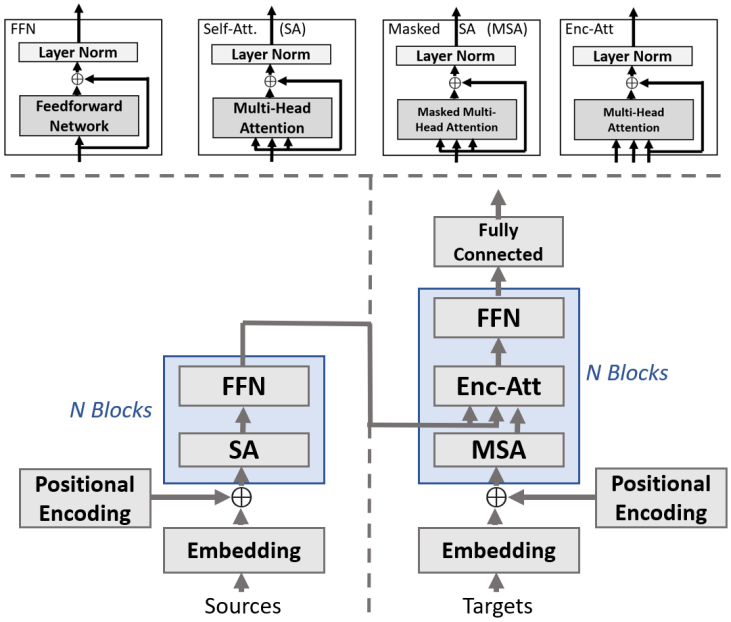

Fig. 2: Transformer with the four basic blocks on top and the encoder and decoder at the bottom Figure from [103]

presented. Additionally, the model is required to determine whether a pair of sentences (A, B) are consecutive sentences from the same document or two unrelated sentences. Once pre-trained via self-supervised methods, the model can be fine-tuned for specific tasks using labeled data.

The original BERT model has undergone various improvements, for instance, [98] minimized BERT's computational demands, and [69] extended training durations, utilized longer sequences, larger batches, and more data. These enhancements have led to more robust and generalizable representations.

Text-to-Text Transfer Transformer (T5)[92] considers all text-based language models as systems that produce output text from an input text. This differs from BERT[18] through the use of causal masking in training, which blocks the network from seeing "future" tokens of the target. Additionally, T5 diverges in its pre-training tasks.

BART[63] is a denoising autoencoder used for pretraining sequence-to-sequence models with a conventional transformer-based machine translation architecture. It has proven effective in tasks such as language generation, translation, and comprehension. The training involves corrupting text using a variety of noising functions, including token deletion, masking, sentence permutation, and document rotation. The learning process involves reconstructing the original text from its corrupted form. The variety of noising techniques utilized can be attributed to BART's adaptation and generalization of concepts from previous models like BERT and GPT[89], combining a bi-directional encoder (similar to BERT) with an autoregressive decoder (similar to GPT).

XLNet [132] This approach merges the benefits of autoregressive modeling, as seen in GPT, where it predicts the next token, with the denoising auto-encoding capabilities of BERT[18], which involves reconstructing x from a noisy input $\hat{x}$ created by masking words in x. It achieves this by using a permutation language model that samples a permutation of $Z = z_0, z_1, ..., z_{T-1}$ of the sequence $(0, 1, 2, ..., T-1)$ aiming for the objective:

$$\max p(u_{z_T} | u_{z_0}, ..., u_{z_{T-1}}) \tag{3}$$

Importantly, the inputs themselves are not actually permuted, which would be impractical and inconsistent with later fine-tuning tasks. Instead, the permutation influences the attention mask to ensure that the factorization order determined by Z is preserved.

The **Vision Transformer** [21] splits an images into small patches, each of which is flattened and embedded linearly along with position embeddings. These embedded vectors for each patch are then processed by a standard transformer encoder.

The **Swin Transformer** [70] constructs hierarchical feature maps in computer vision applications instead of just a single resolution feature map. Additionally, it limits the computation of self-attention to within a local window, which helps reduce the overall computation time.

PaLM 2: The original PaLM model[17], containing 540 billion parameters, parallels other prominent models like GPT-3 in scale. Its technical advancements primarily focus on the scalability of model training, enabling a single model to be trained efficiently across tens of thousands of accelerator chips. Slight modifications were made to the original transformer architecture[123], such as incorporating SwiGLU activations, represented by

$$Swish(xW) \cdot xV \tag{4}$$

, where Swish is defined in Eq. 22. Additional changes include alternative positional embeddings optimized for longer sequences, multi-query attention for faster computation, the elimination of biases for improved training stability, and shared input-output embeddings.

PaLM 2[30], the more advanced successor of PaLM, distinguishes itself by utilizing a different mixture of datasets that incorporate a broader range of languages and domains, such as programming languages and mathematics. Although it uses the traditional transformer architecture, it operates with a smaller model size yet employs more computational resources for training. Moreover, it broadens its pre-training objectives beyond simple next-word or masked-word prediction, incorporating a variety of tasks to enhance its performance.

OpenAI's GPT to GPT-3 on to ChatGPT 3.5 and GPT-4o: The first version of Generative Pre-trained Transformer (GPT) is based only on the transformer's decoder to predict tokens one after the other. GPT[89] initially undergoes unsupervised pre-training, typically followed by supervised fine-tuning. The pre-training process involves a large corpus of tokens $U = (u_0, u_1, ..., u_{n-1})$ and focuses on maximizing the probability of predicting subsequent tokens based on prior ones:

$$L(U) = \sum_i \log p(u_i | u_{i-k}, ..., u_{i-1}) \tag{5}$$

where k represents the context window size, and the conditional probability is computed using a neural network, specifically a multi-layer transformer decoder[68] that omits the encoder as per [123]. Additionally, this method reduces the memory usage of the attention mechanism.
Building on the foundational GPT model, GPT-2 [90] introduces a few key enhancements, such as relocating layer normalization to the beginning of each sub-block and adding an additional normalization after the last self-attention block. Moreover, the initialization of residual weights has been adjusted, and there has been an increase in the vocabulary, context window, and batch sizes.
GPT-3[13] maintains a structure nearly identical to GPT-2, but it expands dramatically in scale, with more than 100 times the number of parameters and variations in the volume of training data.
(Chat)GPT-3.5 [82] is closely related to InstructGPT[85], designed with a focus on adhering to user directives. InstructGPT refines GPT-3's capabilities through a dual-phase fine-tuning approach: (i) guided by demonstrations from labelers using supervised learning and (ii) leveraging human evaluations of model responses in a reinforcement learning framework. ChatGPT adheres to this methodology, where (i) supervised learning involves human AI trainers engaging in scripted dialogues, where humans act as both the user and the AI agent. These dialogues are subsequently merged with the InstructGPT conversations, which have been reformatted into dialogues. In Phase (ii), AI trainers assess the quality of responses generated by ChatGPT, choosing from multiple possibilities for a randomly chosen model-generated text. This ranking informs the reinforcement learning phase to refine the model's performance.
The technical specifications of GPT-4, the follow-up to ChatGPT-3.5, have not been officially released as per the technical report[83], although some unofficial sources [73] claim to have insights. The report suggests that GPT-4 retains many characteristics of ChatGPT 3.5 but extends its capabilities to multi-modal functions, allowing it to process images as well. It highlights significant enhancements in training efficiency and a notable achievement in predicting the performance of large-scale models based on the outcomes of smaller models, which may have been trained on less data. This aspect is particularly crucial given the significant impact of computational costs and time on the development of extensive deep learning models.
Later models by OpenAI such as GPT-4 Turbo and GPT-4o [84] focused on improving efficiency and response times and multi-modality, including audio, with comparable performance or modest gains on benchmarks. While there are no technical details available, models were likely down-sized and the amount of training was increased, possibly with new or improved data.

Meta's Open source LLama series: LLama is Meta's open-source model family of LLMs, ranging from LLama[120], LLama-2[121] on to LLama-3.1[76]. The LLama-3 paper also discussed multi-modal experiments. The largest model LLama 3.1 405B achieves similar performance as commercial models such as GPT4o and Claude Sonnet 3.5[76]. All models are dense transformers rather than MoE to improve training stability. Improvements, in particular for LLama-3 originated from scaling data, models and training and changes in post-processing, e.g, using direct preference optimization (DPO)[91] rather than reinforcement learning. That is, LLama-3 included no major ar-

chitectural changes compared to prior LLama models and, in turn, to the original transformer architecture or self-supervised pre-training proceducre. Minor updates were the usage of grouped query attention [1] and preventing self-attention between different documents (using an attention mask). Architectural details including hyperparameters are provided in [76].

Gemini, Claude and other models: The Gemini family has been briefly described in a technical report [117] and the version 1.5 family in [95]. However, technical details are mostly missing. It is also based on the transformer architecture by Vaswani et al. with improvements in architecture such as multi-query attention[107] and optimization to help training at scale and improve efficiency. It is multi-modal including video. The version 1.5 improved long-context understanding of inputs and it introduced a sparse mixture of expert (MoE) transformer-based model. The company 'Mistral.AI' also launched Mixtral, an opensource MoE model[50]. GPT-4 is also alleged to be based on MoE with eight experts [73]. Though Anthropic's Claude models are claimed to perform well, there was no official information on its architecture or training but only a model card [4].

3.2 Trends and Alternatives to Transformers

The original transformer architecture is still the key model for LLMs and a key driver for generative AI – as elaborated in the prior section 3.1. The prior section also highlighted a few trends that emerged in the last few years to improve transformers. Many focus on computational aspects by altering existing transformers, e.g., using a MoE to reduce computation or altering attention mechanisms, while others are not driven by computation in particular and suggest more radical ideas, e.g, by combining older models like RNNs with new elements of transformers.

Combining Sequence Models and Transformers: That is, there is a stream of research that aims to take elements from sequence models such as RNNs and LSTMs to transformers. However, one alleged success factor for transformers in the first place is their parallelizability in contrast to classical sequence models.

[8] proposed xLSTM, which improves on the classical LSTM cell by using exponential gating with memory mixing and a new memory structure. It allows for improved parallel training. While the results are promising, they stem mostly from smaller models (1.3B) and, thus, need to be validated using larger models.

Receptance weighted key value (RWKV) [87] combines the well parallelizable training of transformers with the fast inference of RNNs. For inference they achieve linear time complexity in sequence length S and memory needs linear in the (embedding) dimension. In contrast, classical transformer require a factor S more for both time and space(memory). The architecture bares similarity with an ordinary transformer but they use a variant of linear attention to focus on channels rather than the classical dot-product token attention.

Retentive networks[115] replace multi-head attention with multi-scale retention. They derive it from a recurrent model by adding context awareness. In a recurrent setting, a state s is updated using a matrix Q. Context awareness means that the matrix Q

is also mulitplied by the context X derived from the current input. They also perform matrix diagonalization to obtain an easily parallizable formulation.

State-space models - Replacing attention with a merger of CNNs and RNNs:

State-space models (SSM) are well known in natural science dating back more 60 years[51]. They can be written as $x'(t) = Ax(t) + Bu(t)$, $y(t) = Cx(t) + Du(t)$ with matrixes A, B, C, and D. They have been adopted for sequence modeling[35], where they can be interpreted as "a combination of recurrent neural networks (RNNs) and convolutional neural networks (CNNs), with inspiration from classical state space models"[33]. In particular, [34] focuses on long range dependencies leading to the so-called structured state space sequence model (S4) [34]. Technically, they apply a low-rank correction to matrix A allowing the matrix to be diagonalized stably. Special SSM layers have also been proposed for LLMs achieving comparable performance on small models [27]. A major evolution is MAMBA[33], which neither requires attention nor MLPs. It introduces a selection mechanism that allows to select data in an input-dependent manner. They also create a hardware aware algorithm and simplify prior SSM architectures by merging with the MLP layer into a single block.

Forgoing attention based on MLPs only and gated convolutions: One of the simplest architectures relies on multi-layer perceptrons (MLP)s only. The MLP-mixer[118] has been shown to achieve state-of-the-art results in computer vision without convolutions and self-attention. It relies on MLPs that are applied on spatial locations or feature channels instead. It also maintaining other common architectural elements such as skip-connections and layer normalization. Recent work has also improved on MLPs for mixing, e.g., by using Monarch matrices [26].

A drop-in replacement for attention is Hyena [88]. It is based on long convolutions and data-controlled gating. The paper also discusses three factors that must be fulfilled by attention alternatives but are often missed, i.e., (i) data control: Attention is a linear data controlled operator, (ii) sublinear parameter scaling: parameters (of attention layers) does not depend on sequence length, and (iii) unrestricted context: interaction between any tokens is possible, e.g., there is no locality.

Models to systems - Integrating planning and external tools: While a single LLM is often seen or believed to be capable of doing any task, there are a number of approaches that aim at building systems rather than improving only the deep learning model in isolation. That is, LLMs are enhanced with other components that interact to improve its performance. For example, [52] claim that large language models (LLMs) cannot plan but can generate ideas that are critiqued by separate models, which may be domain-specific and not LLMs. Ideas like Retrieval Augmented Generation (RAG)[64] have altered the way how LLMs process prompts by first enhancing them with external knowledge, which helped to reduce hallucinations. The usage of external tools by the LLM such as calculators has also improved the performance of LLM-based systems for many tasks[100].

3.3 Far-out models and ideas

The prior section contained some interesting ideas that have arleady shown promising outcomes compared to transformers in empirical evaluations. However, one might also

seek to leverage more revolutionary ideas that have been proposed in the context of deep learning but have not yet yielded the desired success.

Capsule networks - Returning vectors instead of scalars: Capsule networks have been proposed in 2011[45]. The essential idea is that a neuron outputs a vector rather than a scalar. Hinton et al. motivate the usage of a vector output in the context of computer vision. They argue that a (visual) feature is often characterized by multiple instantation parameters such as position, scale, lighting, etc. that resemble a vector rather than just a scalar. A capsule learns an implicitly defined visual entity across some limited transformations such as lighting or scaling. It returns the probability that the entity is present and a set of instantation parameters (e.g., lighting or scaling). They argue that a key advantage of this approach is the recognition of whole-part relationships, e.g., to detect a face one might identify a mouth and nose and both must have appropriate scaling and orientation. The training of the network requires pairs of transformed images to learn capsules that can extract pose parameters from inputs. Capsules themselves might consist of classical elements such as dense layers and gates. Capsule networks have been further improved since their introcduction in 2011[96,46]. They managed to achieve good performance on simple daasets such as MNIST but failed to deliver close to state-of-the-art performance on challenging datasets such as ImageNet.

Spiking neural networks - Adding temporality through non-synchronous activating neurons: Spiking neural networks [72] exist for more than 25 years, achieving remarkable, but no state-of-the-art results. Spiking neural networks add a temporal dimension to neural networks. A neuron activates once its "potential" exceeds a threshold through transmission of spiking signals to its adjacent neurons. The spike increases their potential. Potentials decrease upon sending spikes but also spontaneously over time (without transmission). Thus, in contrast to non-spiking networks, transmission of signals in the network can occur in non-periodic, non-synchronized cycles among neurons. As such the time, when a signal occurs also resembles information in addition to the actual information in the spike. This effectively allows to perform time-coding as common distributed systems [104].

Neurosymbolic AI - Leveraging domain specific (symbolic) languages: There is also an (ongoing) debate on the value of symbolic approaches in conjunction with neural networks relying on continuous representations. For example, in 2017 a neuro symbolic approach for program synthesis was proposed[86] that can generate program code in a domain specific language from input-output examples. While the synthesis appears trivial compared to current program skills of LLMs, the paper still contains a number of interesting ideas, e.g., related to how to search the space of possible programs and how to incrementally refine programs. Other neuro-symbolic approaches have also performed concept extraction based on neural networks and used domain specific languages to perform quasi-symbolic program execution [74]. A comprehensive overview with a focus on NLP is given in [38].

Embrassing ideas from the mathematics community: Classical ideas from mathematics have also entered the domain of deep learning. For example, residual neural networks (ResNets) [42] essentially solve ordinary differential equations(ODE) [14]. A simple ODE is $\frac{dy}{dt} = f(y,t)$ and the inital condition $y(t_0) = y_0$. Its solution y_n at dis-

cretized time steps can be computed (recursively) using the Euler method:

$$y_{n+1} = y_n + f(y_n, t_n) \cdot (t_{n+1} - t_n)$$

A layer n in a residual network essentially performs the same computation: $y_{n+1} = y_n + f(y_n, t_n)$ Thus, a sequence of residual blocks can be interpreted as a solution of the ODE with the Euler method with the initial condition $y(0) = X$, where X is the input to the network. Viewing ResNets as ODEs allows to utilize mathematical tools such as ODE solvers and, in turn, might also have benefits with respect to memory and parameter efficiency. The idea of reformulating learning systems has also led to performance improvements [40]. That is, e.g., [40] introduce time-continuous RNNs described as a linear first order system.

Embrassing ideas from neuroscience and psychology: While AI does not seek to rebuild the brain, (articial) neural networks structurally still share similarity with the real neural networks in our heads. Thus, almost any finding in the neuroscience community might be used as an inspiration for improving artificial neural networks. The main hinderness to adopting neuroscience in deep learning research is "that we simply do not have enough information about the brain to use it as a guide"[29]. For instance, the bain has (neurological) feedback loops: A neuron receives and processes a signal, and can then send a response back to the original source. While numerous attempts have been made to leverage this idea, overall success has been limited, i.e., it is essentially a call for more research. For example, [53] argued that feedback (or recurrent circuits) are critical for fast object identification. In another idea from psychology, a reflective architecture was introduced that leverages explanations computed from gradients (GRADCAM) during learning but also during inference[102] drawing inspirations from Kahneman's fast thinking (System 1) and slow (reflective) thinking (System 2). The idea of ordering samples by difficulty to improve learning of deep learning models, i.e., curriculuum learning [10], can be traced back to the idea of shaping in animal training (e.g., [111]).

4 Loss functions and Optimization

We elaborate on fundamental loss functions and optimizers.

4.1 Loss Functions

Loss functions, as reviewed in [126], typically include several terms including a regularization term. While these functions are generally tailored to specific tasks, certain universal principles can be applied across various tasks. It is common to combine multiple loss terms using a weighted approach. Frequently, enhancements to previous studies are achieved merely by modifying the loss function.

The **Triplet Loss** [20] was introduced for Siamese networkss, though its roots trace back to earlier work[105]. The overarching principle involves comparing a given input with both a positive and a negative input, aiming to maximize the association with

positively related inputs and minimize it with negative ones. It operates on input pairs (x,y), each processed by a distinct but architecturally identical network. The goal is to maximize the joint probability $p(x,y)$ for all pairs (x,y):

$$L(\mathcal{V}_p,\mathcal{V}_n) = -\frac{1}{|\mathcal{V}_p|\cdot|\mathcal{V}_n|}\sum_{x\in\mathcal{V}_p}\sum_{y\in\mathcal{V}_n}\log p(x,y) \tag{6}$$

$$= -\frac{1}{|\mathcal{V}_p|\cdot|\mathcal{V}_n|}\sum_{x\in\mathcal{V}_p}\sum_{y\in\mathcal{V}_n}\log(1+e^{x-y}) \tag{7}$$

Here, $\mathcal{V}_p$ and $\mathcal{V}_n$ are the positive and negative score set respectively.

The **Supervised Contrastive Loss**[59] groups clusters of points from the same class together in the embedding space while pushing apart samples from different classes. It is designed to utilize label information more effectively than cross-entropy loss, optimizing the separation and aggregation based on class identity.

$$\mathcal{L}_i^{sup} = \frac{-1}{2N_{\tilde{y}_i}-1}\cdot \tag{8}$$

$$\sum_{j=1}^{2N}\mathbf{1}_{i\neq j}\cdot\mathbf{1}_{\tilde{y}_i=\tilde{y}_j}\cdot\log\frac{\exp(z_i\cdot z_j/\tau)}{\sum_{k=1}^{2N}\mathbf{1}_{i\neq k}\cdot\exp(z_i\cdot z_k/\tau)} \tag{9}$$

where $N_{\tilde{y}_i}$ denotes the total number of images in the minibatch with the same label $\tilde{y}_i$ as the anchor i. The overall loss is calculated as the sum of the losses for all anchor points i, i.e., $\mathcal{L} = \sum_i \mathcal{L}_i^{sup}$. This loss formulation is particularly advantageous for supervised learning due to its:

- Ability to generalize across an arbitrary number of positive examples.
- Increased contrastive effectiveness as the number of negative examples grows.

The **Cycle Consistency Loss**[136] is specifically designed for unpaired image-to-image translation using generative adversarial networks. It facilitates the learning of mappings between two distinct image domains X and Y. Optimizing the loss supports the learning of mappings $G: X \to Y$ and $F: Y \to X$ so that one reverses the other, i.e., $F(G(x)) \approx x$ and $G(F(y)) \approx y$.

$$L(G,F) = \mathbb{E}_{x\sim p_{data}(x)}[||F(G(x))-x||_1] \tag{10}$$

$$+\mathbb{E}_{y\sim p_{data}(y)}[||G(F(y))-y||_1] \tag{11}$$

Focal Loss[65] modifies the standard cross-entropy loss to concentrate learning efforts on hard-to-classify samples. It incorporates a factor $(1-p)^\gamma$, where p is the probability of a sample from the cross entropy loss and γ is a freely adjustable parameter.

$$L(p) = (1-p)^\gamma \log(p) \tag{12}$$

4.2 Regularization

Regularization techniques, as reviewed in [80], are critically beneficial for deep learning applications. Explicit regularization incorporates an additional loss term $R(f)$ for the network f into the loss function $L(x)$ for data (x_i, y_i) with a balancing parameter λ.

$$\min_f \sum_i L(x_i, y_i) + \lambda R(f) \tag{13}$$

Implicit regularization encompasses all other forms of regularization, such as early stopping or employing a robust loss function. Common methods like *L2*-regularization and dropout[113], which involves setting the activations of a random subset of neurons to zero, are among the most extensively used techniques.

Entropy Regularization [79] is designed to promote diversity, particularly through asynchronous methods in deep reinforcement learning [127,79]. This approach enhances the diversity of actions in reinforcement learning by avoiding excessive optimization towards only a small portion of the environment. Entropy is calculated based on the probability distribution of actions as determined by the policy as:

$$H(x) = \sum_x \pi(x) \cdot \log(\pi(x)) \tag{14}$$

Path Length Regularization [54] [54] focuses on enhancing the image quality for generative adversarial networks by ensuring consistent step lengths in the latent space and changes in the generated images. That is, changes in the latent space representation lead to (proportional) changes in the generated image (and vice versa). The principle behind this is to facilitate a fixed-size step in the latent space $\mathcal{W}$ to yield a consistent, non-zero change in image magnitude. This approach aims to improve the conditioning of GANs, which can simplify the process of architecture search and generator inversion. Gradients, with respect to $\mathbf{w} \in \mathcal{W}$, originating from random directions in the image space should be almost equal in length independent of $\mathbf{w}$ or the image space direction. The local metric scaling characteristics of the generator $g : \mathcal{W} \to \mathcal{Y}$ are stated by the Jacobian matrix $\mathbf{J_w} = \delta g(\mathbf{w})/\delta \mathbf{w}$. The formulation for the regularizer is:

$$\mathbb{E}_{\mathbf{w},\mathbf{y}\sim\mathcal{N}(0,\mathbf{I})}(||\mathbf{J_w^T y}||_2 - a)^2 \tag{15}$$

where y corresponds to random images with pixel values that follow a normal distribution, and $w \sim f(z)$, where z is also normally distributed. The constant a is defined as the exponential moving average of $||\mathbf{J_w^T y}||_2$. The paper also avoids the computationally expensive, explicit computation of the Jacobian.

DropBlock[28] selectively drops correlated regions within feature maps instead of dropping features independently. This method is particularly effective for convolutional neural networks, where feature maps tend to show spatial correlations and a real-world feature typically corresponds to a contiguous spatial area within these maps.

R_1 **Regularization** [75] penalizes the discriminator in generative adversarial networks based on the gradient to stabilize training:

$$R_1(\psi) = \frac{\gamma}{2} E_{p_D(x)}[||\nabla D_\psi(x)||^2] \tag{16}$$

Technically, the regularization term imposes penalties on gradients that are orthogonal to the data manifold, aiming to refine the learning process.

4.3 Optimization

Optimization, as explored in [114], involves determining the best set of network parameters to minimize the loss function. The two most widely recognized techniques are stochastic gradient descent (SGD) and Adam. Neither technique consistently surpasses the other across all scenarios in terms of generalization performance. SGD has historical roots extending back to the 1950s [60], whereas Adam was developed more recently, in 2014 [61].

Adafactor [108] enhances the efficiency of the Adam optimization algorithm by reducing its memory requirements. This is achieved by keeping only row- and column-wise statistics of parameter matrices, instead of storing detailed per-element information.

Layerwise adaptive large batch optimization (LAMB)[133] builds on Adam and accelerates training by utilizing large mini-batches. It applies per-dimension and layer-wise normalization to further enhance the optimization process.

Decoupled Weight Decay Regularization for ADAM: AdamW[71] leverage a seemingly trivial observation: The original Adam optimizer updates weights with (L2-)regularization after calculating the gradients for Adam. However, it intuitively makes more sense that moving averages of gradients should exclude regularization.

RAdam and AMSGrad: Both techniques address the convergence issues associated with Adam. Rectified Adam (RAdam)[66] adjusts the variance of the adaptive learning rate, which is initially large. This approach is akin to the warm-up heuristic, where small initial learning rates are beneficial. AMSGrad[94] uses the maximum of past squared gradients instead of the exponential average to enhance stability and convergence.

Stochastic Weight Averaging: Naive averaging of weights from different epochs during stochastic gradient descent, using either a constant or cycling learning rate, has been shown to improve performance[49].

Two Time-scale Update Rule(TTUR): TTUR[44] enhances the training of generative adversarial networks by employing separate learning rates for the discriminator and the generator. When the generator is fixed, the discriminator can reach a local minimum. This approach remains effective even if the generator converges slowly, for instance, by using a smaller learning rate. By allowing the generator to more deeply assimilate feedback from the discriminator before exploring new regions, this method aids in the convergence of GANs and can lead to improved performance. Besides ensuring convergence, the performance may also improve since the discriminator must first learn new patterns before they are transferred to the generator. In contrast, a generator which is overly fast, drives the discriminator steadily into new regions without capturing its gathered information.

Sharpness-Aware Minimization[25] minimizes both the loss value and sharpness, which enhances generalization. It seeks parameters within neighborhoods of low loss values, rather than focusing solely on parameters that individually exhibit low loss. The

loss function is defined as:

$$\min_{w} \max_{||\varepsilon||_p \leq \rho} L(w+\varepsilon) \tag{17}$$

5 Self, Semi-supervised and Contrastive learning

Semi-supervised learning utilizes a large volume of unlabeled data alongside a small amount of labeled data, as reviewed in [131]. Self-supervised learning, on the other hand, generates (pseudo) labels from artificial tasks, reducing the need for manually labeled data. Both approaches alleviate the burden of collecting human-annotated data. Combining self-supervised (pre-)training with fine-tuning on a small human-labeled dataset can achieve state-of-the-art results. This paradigm has significantly expanded in recent years (surveyed in [24]). It is often integrated with contrastive learning, which aims to learn the distinction between similar and dissimilar data. By automatically distorting data to varying degrees, creating "pseudo-labeled" data for self-supervised learning becomes straightforward.

The **simple framework for contrastive learning** (SimCLR)[15] maximizes the agreement between two inputs derived from different augmentations of the same data sample. Augmentations can include random cropping, color distortions, and Gaussian blur. A standard ResNet[42] is used to obtain representation vectors, which are then further processed through a simple MLP before applying the contrastive loss.

Bootstrap Your Own Latent (BYOL) [31] employs both an online network and a target network, each with identical architectures comprising an encoder, a projector, and a predictor, but with separate weights. The parameters of the target network are updated as an exponential moving average of the online network's parameters. The task of the online network is to predict the representation of the target network given an augmentation of the same input.

Barlow Twins[135] rely on an objective function that aims to reduce cross-correlation C between outputs for a set of image Y^A and their distorted versions Y^B as close to the identity as possible, i.e., the loss (including λ as a tuning parameter) is:

use an objective function aiming to minimize the cross-correlation C between outputs for a set of images Y^A and their distorted versions Y^B, bringing it as close to the identity matrix as possible. The loss function, incorporating λ as a tuning parameter, is given by:

$$L = \sum_i (1 - C_{i,i})^2 + \lambda \cdot \sum_i \sum_{j \neq i} C_{i,j}^2 \tag{18}$$

Momentum Contrast (MoCo) [41] constructs a dynamic dictionary via an encoder using unsupervised contrastive learning. During training, it performs look-ups, ensuring that an encoded query closely matches its corresponding encoded key while being dissimilar to other keys. The dictionary functions as a queue of data samples: for each mini-batch, new encoded samples are added, and the oldest mini-batch is dequeued.

The key encoder's parameters are updated using a momentum-based moving average of the query encoder, promoting consistency over time.

Noisy Student: [129] outlines a method where an CNN-based EfficientNet model is initially trained on labeled data. This trained model then acts as a teacher, generating pseudo labels for unlabeled images. Subsequently, a larger model is trained on the combined labeled and pseudo-labeled data. This process is iteratively repeated, with each student model becoming the teacher for the next iteration. During the training of the student model, techniques like dropout and data augmentation are employed to introduce noise, making the learning process more challenging and allowing the student to surpass the performance of the teacher.

FixMatch [112] predicts the label of a weakly-augmented image. If the confidence in this prediction exceeds a certain threshold, the model is then trained to produce the same label for a strongly-augmented version of the image.

6 Architectures and Layers

We elaborate on four key types of layers: activation layers, skip connections, normalization layers, and attention layers. This is followed by a discussion of various contemporary architectures based on transformers and graph neural networks.

6.1 Activation

Activation functions are typically non-linear and significantly influence gradient flow and learning. Early activation functions, such as sigmoid and tanh, were commonly used from the 1960s through the early 2000s. However, these functions can cause training difficulties in deep networks due to the vanishing gradient problem when they saturate. The introduction of the rectified linear unit (ReLU) in 2010[81] was a breakthrough. Although the original ReLU remains widely used, transformer architectures have introduced other activation functions and ReLU variants. Most of these alternatives share the qualitative behavior of ReLU, where outputs for negative inputs are of small magnitude and outputs for positive inputs are unbounded (see [5] for a survey).

Gaussian Error Linear Units (GELU)[43] These functions weigh inputs by their percentile (ReLUs only use the sign). The activation function is the product of the input and the standard Gaussian cumulative distribution function $\Phi(x)$, i.e.,

$$GELU(x) = x \cdot \Phi(x) \tag{19}$$

The **Mish** activation[78] originates from systematic search inspired by Swish and ReLU:

$$f(x) = x \cdot \tanh(soft^+(x)) \tag{20}$$

$$\text{with } soft^+(x) := \ln(1 + e^x) \tag{21}$$

In contrast, the Swish activation[93] is:

$$f(x) = x \cdot sigmoid(\beta x) \tag{22}$$

Here β is a learnable parameter.

6.2 Skip connections

Skip connections were introduced for residual networks[42]. In their simplest form, the output y for an input x of a block L of layers with a skip connection is $y(x) = L(x) + x$. The term "residual" was used in the original paper because the layer L is tasked with learning a residua $L(x) = H(x) - x$ rather than the desired mapping H itself. Since then, skip connections have been utilizied in a number of variations.

ResNeXt Block[130]: This split-transform-merge approach for residual blocks involves evaluating a set of residual blocks concurrently and then aggregating their outputs back into one output. A **Dense Block**[47] receives inputs from all preceding layers with matching feature map sizes and connects to all such subsequent layers.

Inverted Residual Block[97]: By reversing the channel width sequence to a narrow-wide-narrow order from the original wide-narrow-wide configuration[42], and using depthwise convolutions for the wide layer, parameters are reduced, and residual blocks execute more quickly. Furthermore, the activation function of the last layer within a block is skipped.

6.3 Normalization

With the advent of batch normalization[48], the concept of normalization has significantly enhanced training speed, stability, and generalization in neural networks including almost all architectures. However, its necessity is debated[106]; for certain applications, careful initialization and learning rate adjustments may render normalization partially redundant.
The principle behind normalization is to transform a value x to a normalized value $\tilde{x}$, by subtracting the mean μ and scaling by the standard deviation σ, i.e., $\tilde{x} = \frac{x-\mu}{\sigma}$. Normalization approaches differ in the computation of μ and σ, e.g., μ and σ can be computed across different channels.

Layer Normalization: Normalization statistics are calculated using summed inputs[6] for a layer L with $|L|$ neurons as:

$$\mu = \frac{1}{|L|}\sum_{i=0}^{|L|-1} a_i \qquad \sigma = \sqrt{\frac{1}{|L|}\sum_{i=0}^{|L|-1}(a_i-\mu)^2} \tag{23}$$

Unlike batch normalization, this method does not limit the batch size and eliminates inter-batch dependencies, making it suitable even for batch sizes as small as one.

LayerScale[119] has been established for transformers as a per-channel multiplication of outputs of a residual block with a diagonal matrix:

$$x_{l'} = x_l + diag(\lambda_1, ..., \lambda_d) \cdot SA(\eta(x)) \tag{24}$$
$$x_{l+1} = x_{l'} + diag(\lambda_1, ..., \lambda_d) \cdot FFN(\eta(x)) \tag{25}$$

SA is the self-attention layer, FFN is the feed forward network, and η the layer-normalisation (see Figure 2).

Instance Normalization[122] calculates for a four-dimensional input, such as an image of height H, width W, channels C, and batch size T:

$$\mu_{t,c} = \frac{1}{HWT} \sum_{t<T,w<W,h<H} x_{t,c,w,h} \tag{26}$$

$$\sigma_{t,c} = \sqrt{\frac{1}{HWT} \sum_{t<T,w<W,h<H} (x_{t,c,w,h} - \mu_{t,c})^2} \tag{27}$$

This method can be applied, for example, to normalize image contrast. Various adaptations exist, including a version that scales according to weight norms[55].

6.4 Attention

Attention mechanisms, explored in [12,37], enable the learning of relevance scores for inputs, mimicking cognitive attention processes. This allows certain parts of the inputs to be highlighted as highly important, while others may be ignored as irrelevant. Often, the importance of a specific input is influenced by its context; for example, the significance of a word in a text document usually relies on the words surrounding it.

Scaled Dot-Product Multi-Head Attention [123]:

The use of dot products, coupled with down-scaling, has been highly effective in calculating attention scores. Attention takes a query Q, a key K and a value V as inputs and outputs an attention score:

$$\text{Att}(Q,K,V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}}) \cdot V \tag{28}$$

Employing multiple, independent attention mechanisms in parallel enables the system to focus on different aspects of the input simultaneously. In the case of multi-head attention, matrices $\mathbf{W}$ are learned to facilitate this process. Formally:

$$\text{MultiHead}(\mathbf{Q},\mathbf{K},\mathbf{V}) = [\text{h}_0,\ldots,\text{h}_{n-1}]\mathbf{W}_0 \tag{29}$$

$$\text{where head h}_i = \text{Att}(\mathbf{QW}_i^Q, \mathbf{KW}_i^K, \mathbf{VW}_i^V) \tag{30}$$

Factorized (Self-)Attention [16] decreases both the computational and memory demands compared to the original "full" self-attention[123], which allows every element to attend to every other prior input. In factorized self-attention, attention is limited to a subset of input elements. Formally, an output matrix is generated using a matrix of input embeddings X and the connectivity pattern $S = \{S_1,...,S_n\}$, where S_i is the set of indices of input vectors attended to by the ith output vector.

$$\text{FacAtt}(X,S) = (A(\mathbf{x}_i, S_i))_{i\in[1,n]} \tag{31}$$

$$a(\mathbf{x}_i, S_i) = \text{softmax}(\frac{(W_q\mathbf{x}_i)K_{S_i}^T}{\sqrt{d}}) \cdot V_{S_i} \tag{32}$$

$$K_{Si} = (W_k\mathbf{x}_j)_{j\in S_i} \qquad V_{S_i} = (W_v\mathbf{x}_j)_{j\in S_i} \tag{33}$$

For full self-attention $S_i^F := \{j | j \neq i\}$, i.e., all indexes to inputs prior to the ith input. In contrast, factorized self-attention has p separate attention heads, where the mth head is given by a subset $A_i^{(m)} \subset S_i^F$ with $S_i = A_i^{(m)}$. For strided self-attention:

$$A_i^{(1)} = \{t, t+1, ...i\} \text{ for } t = \max(0, i-l) \tag{34}$$

$$A_i^{(2)} = \{j : (i-j) \mod l = 0\} \tag{35}$$

This pattern is adequate, when structure is similar to the stride-like images. For data without a periodic structure like text, fixed attention can be preferable:

$$A_i^{(1)} = \{j : \lfloor j/l \rfloor = \lfloor i/l \rfloor\} \tag{36}$$

$$A_i^{(2)} = \{j : j \mod l \in \{t, t+1, ...l\}\} \tag{37}$$

where $t = l - c$ with hyperparameter c. For instance, with a stride of 128 and $c = 8$, all future positions greater than 128 can attend to positions 120-128, all greater 256 to 248-256, and so on.

Multi- and Grouped Query Attention[107,1] Multi-query attention(MQA)[107] uses multiple query but only one key head and one value head, which differs from standard muli-head attention. This helps in lowering memory bandwidth for loading keys and values but it reduce quality and training stability. MQA can be added to trained models without MQA with (relatively) little compute [1]. Grouped query attention aims at striking a balance between a single and many heads. Per head they use multiple queries rather than just one. This allows to lower quality gaps.

A **Residual Attention Network (RAN)**[125] capitalizes on the concept of skip connections and is composed of two branches: a mask branch and a trunk branch. The trunk branch is responsible for processing features and can be any type of network, while the mask branch determines the weights for these features. The output of the attention module is

$$H_{i,c}(x) = (1 + M_{i,c}(x)) \cdot F_{i,c}(X) \tag{38}$$

where i is a spatial position and c is a channel. $M(x)$ should be approximatedly 0, $H(x)$ approximates original features $F(x)$.

Large Kernel Attention[36] break down a (large-scale) convolution into three more manageable components: a depth-wise dilated convolution, a non-dilated depth-wise convolution, and a 1x1 convolution across channels. Subsequently, an attention map is derived from the outputs of these convolutions.

Sliding Window Attention[9] This approach focuses on enhancing the efficiency of attention mechanisms in terms of both time and memory by reducing the number of input pairs considered. Specifically, for a designated window size w, each token attends to $\frac{w}{2}$ tokens on either side.

6.5 Graph Neural Networks

Graph neural networks, reviwed in [128,58], can be viewed as an extension of CNNs and transformers to accommodate graph-structured data. These networks process information represented as nodes interconnected by edges. We delve into various graph

models, focusing on how to derive node embeddings that are applicable to downstream tasks.

Graph Convolutional Networks (GCN)[62] use CNNs for semi-supervised learning. They approximate spectral graph convolutions using polynomials of order k, which a CNN can compute with k linear layers. This also implies that dependencies are localized, i.e., only up to nodes of distance k from a node.

Scalable Feature Learning for Networks(Node2Vec)[32] is designed to learn feature vectors that effectively preserve the neighborhood characteristics of nodes within a graph. This method employs random walks to generate samples of neighborhoods, allowing it to capture and represent nodes based on their roles or the communities to which they belong.

Graph Attention Networks[124] utilizes masked self-attention layers that enable nodes to adaptively focus on the features of their neighboring nodes. Specifically, node j assigns importance scores to the features of node i based on their connectivity. The use of masking ensures that only connected node pairs are considered. Unlike GCNs, this method allows for different levels of importance to be assigned to nodes within the same neighborhood. Additionally, it avoids the expensive matrix operations associated with eigendecompositions.

TuckER[7] focuses on using factorization for link prediction within a knowledge graph, where knowledge is structured as (subject, relation, object) triplets. The objective is to predict the existence of a relationship between two entities. This approach models the graph as a binary tensor, with the dimensions representing subjects, relations, and objects. Tucker decompositions are then employed to break down this binary tensor into a core tensor and separate embedding matrices for subjects, relations, and objects.

Embedding by Relational Rotation (RotatE)[116] is used for predicting missing links in knowledge graphs, similar to the previously described TuckER[7], but with a focus on modeling additional relational properties like composition and inversion. In this method, entities are embedded in a complex space, and relations are treated as element-wise rotations. These rotations are optimized to effectively transition from one entity to another within the complex embedding space.

Graph Transformer[23] extends the original transformer architecture to handle graph structures by implementing attention mechanisms that consider the connectivity of each node's neighborhood. This adaptation generalizes the concept of position encoding to suit graph data. The model also replaces layer normalization with batch normalization and introduces the capability to learn representations for edges, in addition to node representations.

7 Discussion

Our findings suggest that despite many small and creative innovations since the original transformer architecture, there have not been any significant "breakthrough" discoveries that have led to much better leaderboard results. The improvements on models such as large language models within the last few years have been characterized by the enlargement of existing networks such as GPT, the increase of data volume (and quality),

focus on computational efficiency, and a shift towards self-supervised learning. This could indicate a need for more daring approaches to research rather than incremental improvements of existing works and it raises the question: *Are transformers all we can do?*

Combining different elements or pursuing more radical ideas as outlined in this work could be one way to go beyond transformers. But advancing novel ideas such as state-space models and ideas that have existed for a while such as capsule networks, but have not lived up to their hopes, is not an easy task. We have identified a few general patterns that have been proven effective in many areas and might help in accomplishing this task:

- **Multi-X** refers to the strategy of deploying the same component repeatedly in parallel configurations, such as employing several residual blocks (ResNeXt) or utilizing multi-head attention mechanisms or multiple models as for mixture of experts. This concept is closely aligned with the principles of "ensemble learning."
- **Higher order layers** involve more complex operations than the linear layers and simple ReLU functions typically used in classical CNNs and MLPs. Examples of these more advanced layers include Mish or attention layers, which facilitate deeper and more nuanced processing.
- **Data controlled gating** is an instance of a higher order layer. The idea of data controlled gating, e.g., depending on some (possibly transformed) input X, another input X' is deemed relevant or ignored, seems powerful. This idea can be found in attention but also in LSTM layers and GELU, where we have “self-gating”. On a larger scale also mixture of experts perform data controlled gating and, more generally, in transistors. Data controlled gating can also often be interpreted as **Weighing functions**, which involves the use of parameterized functions to weigh inputs. Instead of simply aggregating inputs, these functions assign weights to them, often derived from learned parameters. These functions act as "gates," allowing the flow of information only within a specific range of input parameters.
- **Moving average** refers to the technique of averaging weights, as seen in methods like SGD and BYOL.
- **Decompose** refers to breaking down matrices into simpler components, as exemplified by methods like TuckER and large kernel attention.

Furthermore, the transformer seems to violate the “no free lunch theorem” by Wolpert saying that no model can be best for all (generative) tasks. While fine-tuning models on specific data can yield better (transformer) models for specific tasks, it is not clear, whether specialized architectures might substantially outperform the general transformer in certain sets of tasks, where it seems to outperform. This is analogous to activation functions, where no activation outperforms on all datasets and within all models [22].

Our survey focused on key design elements in building deep learning models. Taking a practical approach, we chose to ignore theoretical works, which should be further explored in future studies. For example, [110] derived a theoretical framework for dynamical systems allowing to better compare model classes such as linear attention and state space models.

Our survey intentionally focused on more recent, yet well-established works, which could be seen as either a strength or a limitation. The selection of papers for the deep

learning part was guided by a prominent platform that offers leaderboards. The rise of such platforms, which allow the upload of papers and models and provide information on citations and rankings, offers a fresh perspective compared to traditional survey methods that often choose papers more arbitrarily. Although this approach benefits readers looking for "what works well and what is very promising," it may overlook innovative ideas that need more research to fully demonstrate their potential. This could contribute to the "winner-takes-all" dynamic, reinforcing already successful ideas. Furthermore, such platforms might undergo a hype-cycle, e.g., become very popular for a while before vanishing. Nevertheless, given the vast number of papers, some form of selection is essential for conducting a comprehensive survey of deep learning.

We recognize that online platforms offering leaderboards and related features are highly valuable to the research community and should continue to be developed. However, we found that manual verification was necessary, such as double-checking relevance with Google Scholar citations and reviewing surveys and papers, to identify works and methods that were not accurately listed on the platform.

8 Conclusions

We have provided a concise yet thorough overview of the deep learning design landscape, with a focus on transformers and their potential successors. Key works from various significant areas that have emerged in recent years have been summarized. We believe that our holistic overview in a single paper can establish connections that may inspire novel ideas. Additionally, we have identified four patterns that characterize many improvements in this field. To further advance the development of artificial intelligence, it is crucial to generate fundamentally new and successful approaches, as recent improvements, though numerous and often very creative, have primarily been incremental.

References

1. Ainslie, J., Lee-Thorp, J., de Jong, M., Zemlyanskiy, Y., Lebrón, F., Sanghai, S.: Gqa: Training generalized multi-query transformer models from multi-head checkpoints. arXiv preprint arXiv:2305.13245 (2023)
2. Alom, M.Z., Taha, T.M., Yakopcic, C., Westberg, S., Sidike, P., Nasrin, M.S., Hasan, M., Van Essen, B.C., Awwal, A.A., Asari, V.K.: A state-of-the-art survey on deep learning theory and architectures. electronics (2019)
3. Alzubaidi, L., Zhang, J., Humaidi, A.J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M.A., Al-Amidie, M., Farhan, L.: Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions. Journal of big Data (2021)
4. Anthropic: The claude 3 model family: Opus, sonnet, haiku. Online (2023), https://www-cdn.anthropic.com/de8ba9b01c9ab7cbabf5c33b80b7bbc618857627/Model_Card_Claude_3.pdf, accessed: 2024-07-20
5. Apicella, A., Donnarumma, F., Isgrò, F., Prevete, R.: A survey on modern trainable activation functions. Neural Networks (2021)
6. Ba, J.L., Kiros, J.R., Hinton, G.E.: Layer normalization. arXiv:1607.06450 (2016)

7. Balažević, I., Allen, C., Hospedales, T.M.: Tucker: Tensor factorization for knowledge graph completion. arXiv:1901.09590 (2019)
8. Beck, M., Pöppel, K., Spanring, M., Auer, A., Prudnikova, O., Kopp, M., Klambauer, G., Brandstetter, J., Hochreiter, S.: xlstm: Extended long short-term memory. arXiv preprint arXiv:2405.04517 (2024)
9. Beltagy, I., Peters, M.E., Cohan, A.: Longformer: The long-document transformer. arXiv:2004.05150 (2020)
10. Bengio, Y., Louradour, J., Collobert, R., Weston, J.: Curriculum learning. In: Proceedings of the 26th annual international conference on machine learning. pp. 41–48 (2009)
11. Bishop, C.M., Bishop, H.: Deep learning: Foundations and concepts. Springer Nature (2023)
12. Brauwers, G., Frasincar, F.: A general survey on attention mechanisms in deep learning. Transactions on Knowledge and Data Engineering (2021)
13. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al.: Language models are few-shot learners. Advances in neural information processing systems (2020)
14. Chen, R.T., Rubanova, Y., Bettencourt, J., Duvenaud, D.K.: Neural ordinary differential equations. Advances in neural information processing systems **31** (2018)
15. Chen, T., Kornblith, S., Norouzi, M., Hinton, G.: A simple framework for contrastive learning of visual representations. In: Int. Conf. on machine learning (2020)
16. Child, R., Gray, S., Radford, A., Sutskever, I.: Generating long sequences with sparse transformers. arXiv:1904.10509 (2019)
17. Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H.W., Sutton, C., Gehrmann, S., et al.: Palm: Scaling language modeling with pathways. arXiv preprint arXiv:2204.02311 (2022)
18. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv:1810.04805 (2018)
19. Dong, S., Wang, P., Abbas, K.: A survey on deep learning and its applications. Computer Science Review (2021)
20. Dong, X., Shen, J.: Triplet loss in siamese network for object tracking. In: European Conf. on computer vision (ECCV) (2018)
21. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. arXiv:2010.11929 (2020)
22. Dubey, S.R., Singh, S.K., Chaudhuri, B.B.: Activation functions in deep learning: A comprehensive survey and benchmark. Neurocomputing **503**, 92–108 (2022)
23. Dwivedi, V.P., Bresson, X.: A generalization of transformer networks to graphs. arXiv:2012.09699 (2020)
24. Ericsson, L., Gouk, H., Loy, C.C., Hospedales, T.M.: Self-supervised representation learning: Introduction, advances, and challenges. Signal Processing Magazine (2022)
25. Foret, P., Kleiner, A., Mobahi, H., Neyshabur, B.: Sharpness-aware minimization for efficiently improving generalization. arXiv:2010.01412 (2020)
26. Fu, D., Arora, S., Grogan, J., Johnson, I., Eyuboglu, E.S., Thomas, A., Spector, B., Poli, M., Rudra, A., Ré, C.: Monarch mixer: A simple sub-quadratic gemm-based architecture. Advances in Neural Information Processing Systems **36** (2024)
27. Fu, D.Y., Dao, T., Saab, K.K., Thomas, A.W., Rudra, A., Ré, C.: Hungry hungry hippos: Towards language modeling with state space models. arXiv preprint arXiv:2212.14052 (2022)
28. Ghiasi, G., Lin, T.Y., Le, Q.V.: Dropblock: A regularization method for convolutional networks. Advances in neural information processing systems (2018)
29. Goodfellow, I., Bengio, Y., Courville, A.: Deep learning (2016)

30. Google: Palm 2 technical report. https://ai.google/static/documents/palm2techreport.pdf (2023)
31. Grill, J.B., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., Doersch, C., Avila Pires, B., Guo, Z., Gheshlaghi Azar, M., et al.: Bootstrap your own latent-a new approach to self-supervised learning. Adv. in neural information processing systems (2020)
32. Grover, A., Leskovec, J.: node2vec: Scalable feature learning for networks. In: ACM SIGKDD Int. Conf. on Knowledge discovery and data mining (2016)
33. Gu, A., Dao, T.: Mamba: Linear-time sequence modeling with selective state spaces. arXiv preprint arXiv:2312.00752 (2023)
34. Gu, A., Goel, K., Ré, C.: Efficiently modeling long sequences with structured state spaces. arXiv preprint arXiv:2111.00396 (2021)
35. Gu, A., Johnson, I., Goel, K., Saab, K., Dao, T., Rudra, A., Ré, C.: Combining recurrent, convolutional, and continuous-time models with linear state space layers. Advances in neural information processing systems **34**, 572–585 (2021)
36. Guo, M.H., Lu, C.Z., Liu, Z.N., Cheng, M.M., Hu, S.M.: Visual attention network. arXiv:2202.09741 (2022)
37. Guo, M.H., Xu, T.X., Liu, J.J., Liu, Z.N., Jiang, P.T., Mu, T.J., Zhang, S.H., Martin, R.R., Cheng, M.M., Hu, S.M.: Attention mechanisms in computer vision: A survey. Computational Visual Media (2022)
38. Hamilton, K., Nayak, A., Božić, B., Longo, L.: Is neuro-symbolic ai meeting its promises in natural language processing? a structured review. Semantic Web (Preprint), 1–42 (2022)
39. Han, K., Wang, Y., Chen, H., Chen, X., Guo, J., Liu, Z., Tang, Y., Xiao, A., Xu, C., Xu, Y., et al.: A survey on vision transformer. transactions on pattern analysis and machine intelligence (2022)
40. Hasani, R., Lechner, M., Amini, A., Rus, D., Grosu, R.: Liquid time-constant networks. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 35, pp. 7657–7666 (2021)
41. He, K., Fan, H., Wu, Y., Xie, S., Girshick, R.: Momentum contrast for unsupervised visual representation learning. In: Conf. on computer vision and pattern recognition (2020)
42. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Conf. on computer vision and pattern recognition (2016)
43. Hendrycks, D., Gimpel, K.: Gaussian error linear units (gelus). arXiv:1606.08415 (2016)
44. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: Gans trained by a two time-scale update rule converge to a local nash equilibrium. Advances in neural information processing systems (2017)
45. Hinton, G.E., Krizhevsky, A., Wang, S.D.: Transforming auto-encoders. In: Artificial Neural Networks and Machine Learning–ICANN 2011: 21st International Conference on Artificial Neural Networks, Espoo, Finland, June 14-17, 2011, Proceedings, Part I 21. pp. 44–51 (2011)
46. Hinton, G.E., Sabour, S., Frosst, N.: Matrix capsules with em routing. In: International conference on learning representations (2018)
47. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Conf. on computer vision and pattern recognition (2017)
48. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: Int. Conf. on machine learning (2015)
49. Izmailov, P., Podoprikhin, D., Garipov, T., Vetrov, D., Wilson, A.G.: Averaging weights leads to wider optima and better generalization. arXiv:1803.05407 (2018)
50. Jiang, A.Q., Sablayrolles, A., Roux, A., Mensch, A., Savary, B., Bamford, C., Chaplot, D.S., Casas, D.d.l., Hanna, E.B., Bressand, F., et al.: Mixtral of experts. arXiv preprint arXiv:2401.04088 (2024)

51. Kalman, R.E.: A new approach to linear filtering and prediction problems (1960)
52. Kambhampati, S., Valmeekam, K., Guan, L., Stechly, K., Verma, M., Bhambri, S., Saldyt, L., Murthy, A.: Llms can't plan, but can help planning in llm-modulo frameworks. arXiv preprint arXiv:2402.01817 (2024)
53. Kar, K., Kubilius, J., Schmidt, K., Issa, E.B., DiCarlo, J.J.: Evidence that recurrent circuits are critical to the ventral stream's execution of core object recognition behavior. Nature neuroscience **22**(6), 974–983 (2019)
54. Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., Aila, T.: Analyzing and improving the image quality of stylegan. In: Conf. on computer vision and pattern recognition (2020)
55. Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., Aila, T.: Analyzing and improving the image quality of stylegan. In: Conf. on computer vision and pattern recognition (2020)
56. Khan, A., Sohail, A., Zahoora, U., Qureshi, A.S.: A survey of the recent architectures of deep convolutional neural networks. Artificial intelligence review (2020)
57. Khan, S., Naseer, M., Hayat, M., Zamir, S.W., Khan, F.S., Shah, M.: Transformers in vision: A survey. ACM computing surveys (CSUR) (2022)
58. Khoshraftar, S., An, A.: A survey on graph representation learning methods. ACM Transactions on Intelligent Systems and Technology **15**(1), 1–55 (2024)
59. Khosla, P., Teterwak, P., Wang, C., Sarna, A., Tian, Y., Isola, P., Maschinot, A., Liu, C., Krishnan, D.: Supervised contrastive learning. Advances in neural information processing systems (2020)
60. Kiefer, J., Wolfowitz, J.: Stochastic estimation of the maximum of a regression function. The Annals of Mathematical Statistics (1952)
61. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv:1412.6980 (2014)
62. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv:1609.02907 (2016)
63. Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., Zettlemoyer, L.: BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. pp. 7871–7880 (2020)
64. Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.t., Rocktäschel, T., et al.: Retrieval-augmented generation for knowledge-intensive nlp tasks. Advances in Neural Information Processing Systems **33**, 9459–9474 (2020)
65. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: Int. Conf. on computer vision (2017)
66. Liu, L., Jiang, H., He, P., Chen, W., Liu, X., Gao, J., Han, J.: On the variance of the adaptive learning rate and beyond. arXiv:1908.03265 (2019)
67. Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., Neubig, G.: Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. ACM Computing Surveys (2023)
68. Liu, P.J., Saleh, M., Pot, E., Goodrich, B., Sepassi, R., Kaiser, L., Shazeer, N.: Generating wikipedia by summarizing long sequences. arXiv:1801.10198 (2018)
69. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V.: Roberta: A robustly optimized bert pretraining approach. arXiv:1907.11692 (2019)
70. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. In: Int. Conf. on computer vision (2021)
71. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. arXiv:1711.05101 (2017)

72. Maass, W.: Networks of spiking neurons: the third generation of neural network models. Neural networks **10**(9), 1659–1671 (1997)
73. Mandar Kahade: Gpt-4: 8 models in one. https://www.kdnuggets.com/2023/08/gpt4-8-models-one-secret.html (2023), accessed: 2024-07-20
74. Mao, J., Gan, C., Kohli, P., Tenenbaum, J.B., Wu, J.: The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. arXiv preprint arXiv:1904.12584 (2019)
75. Mescheder, L., Geiger, A., Nowozin, S.: Which training methods for GANs do actually converge? In: Int. Conf. on machine learning (2018)
76. Meta: The llama 3 herd of models. Online (2024), https://ai.meta.com/research/publications/the-llama-3-herd-of-models/, accessed: 2024-07-19
77. Min, B., Ross, H., Sulem, E., Veyseh, A.P.B., Nguyen, T.H., Sainz, O., Agirre, E., Heinz, I., Roth, D.: Recent advances in natural language processing via large pre-trained language models: A survey. arXiv preprint arXiv:2111.01243 (2021)
78. Misra, D.: Mish: A self regularized non-monotonic activation function. arXiv:1908.08681 (2019)
79. Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., Kavukcuoglu, K.: Asynchronous methods for deep reinforcement learning. In: Int. Conf. on machine learning (2016)
80. Moradi, R., Berangi, R., Minaei, B.: A survey of regularization strategies for deep models. Artificial Intelligence Review (2020)
81. Nair, V., Hinton, G.E.: Rectified linear units improve restricted boltzmann machines. In: Int. Conf. on machine learning (ICML-) (2010)
82. OpenAI: Chatgpt: Optimizing language models for dialogue. https://openai.com/blog/chatgpt/ (2022)
83. OpenAI: Gpt-4 technical report (2023)
84. OpenAI: Hello gpt-4o! https://openai.com/index/hello-gpt-4o/ (2024), accessed: 2024-07-19
85. Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C.L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al.: Training language models to follow instructions with human feedback. arXiv:2203.02155 (2022)
86. Parisotto, E., Mohamed, A.r., Singh, R., Li, L., Zhou, D., Kohli, P.: Neuro-symbolic program synthesis. arXiv preprint arXiv:1611.01855 (2016)
87. Peng, B., Alcaide, E., Anthony, Q., Albalak, A., Arcadinho, S., Biderman, S., Cao, H., Cheng, X., Chung, M., Grella, M., et al.: Rwkv: Reinventing rnns for the transformer era. arXiv preprint arXiv:2305.13048 (2023)
88. Poli, M., Massaroli, S., Nguyen, E., Fu, D.Y., Dao, T., Baccus, S., Bengio, Y., Ermon, S., Ré, C.: Hyena hierarchy: Towards larger convolutional language models. In: International Conference on Machine Learning. pp. 28043–28078 (2023)
89. Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al.: Improving language understanding by generative pre-training (2018)
90. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al.: Language models are unsupervised multitask learners. OpenAI blog (2019)
91. Rafailov, R., Sharma, A., Mitchell, E., Manning, C.D., Ermon, S., Finn, C.: Direct preference optimization: Your language model is secretly a reward model. Advances in Neural Information Processing Systems **36** (2024)
92. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P.J.: Exploring the limits of transfer learning with a unified text-to-text transformer. The Journal of Machine Learning Research (2020)
93. Ramachandran, P., Zoph, B., Le, Q.V.: Searching for activation functions. arXiv preprint arXiv:1710.05941 (2017)

94. Reddi, S.J., Kale, S., Kumar, S.: On the convergence of adam and beyond. arXiv:1904.09237 (2019)
95. Reid, M., Savinov, N., Teplyashin, D., Lepikhin, D., Lillicrap, T., Alayrac, J.b., Soricut, R., Lazaridou, A., Firat, O., Schrittwieser, J., et al.: Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. arXiv preprint arXiv:2403.05530 (2024)
96. Sabour, S., Frosst, N., Hinton, G.E.: Dynamic routing between capsules. Advances in neural information processing systems **30** (2017)
97. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: Mobilenetv2: Inverted residuals and linear bottlenecks. In: Conf. on computer vision and pattern recognition (2018)
98. Sanh, V., Debut, L., Chaumond, J., Wolf, T.: DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. arXiv:1910.01108 (2019)
99. de Santana Correia, A., Colombini, E.L.: Attention, please! a survey of neural attention models in deep learning. Artificial Intelligence Review **55**(8), 6037–6124 (2022)
100. Schick, T., Dwivedi-Yu, J., Dessì, R., Raileanu, R., Lomeli, M., Hambro, E., Zettlemoyer, L., Cancedda, N., Scialom, T.: Toolformer: Language models can teach themselves to use tools. Advances in Neural Information Processing Systems **36** (2024)
101. Schneider, J., Meske, C., Kuss, P.: Foundation models: a new paradigm for artificial intelligence. Business & Information Systems Engineering pp. 1–11 (2024)
102. Schneider, J., Vlachos, M.: Reflective-net: Learning from explanations. Data Mining and Knowledge Discovery pp. 1–22 (2023)
103. Schneider, J., Vlachos, M.: A survey of deep learning: From activations to transformers. In: Proceedings of the International Conference on Agents and Artificial Intelligence (ICAART) (2024)
104. Schneider, J., Wattenhofer, R.: Trading bit, message, and time complexity of distributed algorithms. In: International Symposium on Distributed Computing. pp. 51–65. Springer (2011)
105. Schultz, M., Joachims, T.: Learning a distance metric from relative comparisons. Advances in neural information processing systems **16** (2003)
106. Shao, J., Hu, K., Wang, C., Xue, X., Raj, B.: Is normalization indispensable for training deep neural network? Advances in Neural Information Processing Systems (2020)
107. Shazeer, N.: Fast transformer decoding: One write-head is all you need. arXiv preprint arXiv:1911.02150 (2019)
108. Shazeer, N., Stern, M.: Adafactor: Adaptive learning rates with sublinear memory cost. In: Int. Conf. on Machine Learning (2018)
109. Shrestha, A., Mahmood, A.: Review of deep learning algorithms and architectures. IEEE access (2019)
110. Sieber, J., Alonso, C.A., Didier, A., Zeilinger, M.N., Orvieto, A.: Understanding the differences in foundation models: Attention, state space models, and recurrent neural networks. arXiv preprint arXiv:2405.15731 (2024)
111. Skinner, B.F.: Reinforcement today. American Psychologist **13**(3), 94 (1958)
112. Sohn, K., Berthelot, D., Carlini, N., Zhang, Z., Zhang, H., Raffel, C.A., Cubuk, E.D., Kurakin, A., Li, C.L.: Fixmatch: Simplifying semi-supervised learning with consistency and confidence. Advances in neural information processing systems (2020)
113. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research (2014)
114. Sun, R.Y.: Optimization for deep learning: An overview. Operations Research Society of China (2020)
115. Sun, Y., Dong, L., Huang, S., Ma, S., Xia, Y., Xue, J., Wang, J., Wei, F.: Retentive network: A successor to transformer for large language models. arXiv preprint arXiv:2307.08621 (2023)

116. Sun, Z., Deng, Z.H., Nie, J.Y., Tang, J.: Rotate: Knowledge graph embedding by relational rotation in complex space. arXiv:1902.10197 (2019)
117. Team, G., Anil, R., Borgeaud, S., Wu, Y., Alayrac, J.B., Yu, J., Soricut, R., Schalkwyk, J., Dai, A.M., Hauth, A., et al.: Gemini: a family of highly capable multimodal models. arXiv preprint arXiv:2312.11805 (2023)
118. Tolstikhin, I.O., Houlsby, N., Kolesnikov, A., Beyer, L., Zhai, X., Unterthiner, T., Yung, J., Steiner, A., Keysers, D., Uszkoreit, J., et al.: Mlp-mixer: An all-mlp architecture for vision. Advances in neural information processing systems **34**, 24261–24272 (2021)
119. Touvron, H., Cord, M., Sablayrolles, A., Synnaeve, G., Jégou, H.: Going deeper with image transformers. In: Int. Conf. on Computer Vision (2021)
120. Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al.: Llama: Open and efficient foundation language models. arXiv preprint arXiv:2302.13971 (2023)
121. Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al.: Llama 2: Open foundation and fine-tuned chat models. arXiv preprint arXiv:2307.09288 (2023)
122. Ulyanov, D., Vedaldi, A., Lempitsky, V.: Instance normalization: The missing ingredient for fast stylization. arXiv:1607.08022 (2016)
123. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems (2017)
124. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. arXiv:1710.10903 (2017)
125. Wang, F., Jiang, M., Qian, C., Yang, S., Li, C., Zhang, H., Wang, X., Tang, X.: Residual attention network for image classification. In: Conf. on computer vision and pattern recognition (2017)
126. Wang, Q., Ma, Y., Zhao, K., Tian, Y.: A comprehensive survey of loss functions in machine learning. Annals of Data Science (2020)
127. Williams, R.J., Peng, J.: Function optimization using connectionist reinforcement learning algorithms. Connection Science (1991)
128. Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., Philip, S.Y.: A comprehensive survey on graph neural networks. Transactions on neural networks and learning systems (2020)
129. Xie, Q., Luong, M.T., Hovy, E., Le, Q.V.: Self-training with noisy student improves imagenet classification. In: Conf. on computer vision and pattern recognition (2020)
130. Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. In: Conf. on computer vision and pattern recognition (2017)
131. Yang, X., Song, Z., King, I., Xu, Z.: A survey on deep semi-supervised learning. Transactions on Knowledge and Data Engineering (2022)
132. Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R.R., Le, Q.V.: Xlnet: Generalized autoregressive pretraining for language understanding. Advances in neural information processing systems (2019)
133. You, Y., Li, J., Reddi, S., Hseu, J., Kumar, S., Bhojanapalli, S., Song, X., Demmel, J., Keutzer, K., Hsieh, C.J.: Large batch optimization for deep learning: Training bert in 76 minutes. arXiv:1904.00962 (2019)
134. Yuksel, S.E., Wilson, J.N., Gader, P.D.: Twenty years of mixture of experts. IEEE transactions on neural networks and learning systems **23**(8), 1177–1193 (2012)
135. Zbontar, J., Jing, L., Misra, I., LeCun, Y., Deny, S.: Barlow twins: Self-supervised learning via redundancy reduction. In: Int. Conf. on Machine Learning (2021)
136. Zhu, J.Y., Park, T., Isola, P., Efros, A.A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. In: Int. Conf. on computer vision (2017)