Neural Networks and their utility as Universal Function Approximators

Erik Connerty

July 22nd, 2023

Code

Abstract

In this paper, we provide an in-depth study of various architectures of neural networks, elaborating their components, functionality, and their underlying processes. We further delve into the phenomenon of neural networks as 'universal function approximators' and provide a perspective on the future of the field of Artificial Intelligence.

Contents

1 Introduction

In the ever-evolving field of computer science, one area that has consistently commanded attention and seen rapid progression is that of Neural Networks. These mathematical models, inspired by the human brain's functioning, have provided remarkable breakthroughs in various domains, including computer vision [1], natural language processing, and even drug discovery, to name a few. As we continue to innovate and refine these architectures, one property that persistently underlies their efficacy is their capacity as Universal Function Approximators - their ability to represent a wide range of complex functions given a sufficient number of parameters [2].

The study of Neural Networks and their capabilities as Universal Function Approximators is not merely an academic endeavor; it holds significant implications for the development of machine learning and artificial intelligence technologies. As we continue to rely on these technologies in an increasing number of sectors - from healthcare to entertainment - understanding their potential and limitations becomes all the more crucial.

In this vein, we speculate on the idea of "Universal Model Approximation," a future development where a model could be automatically determined as "optimum" for a given input data set and a desired output [3, 4]. Both the model architecture and its parameters could be found through an iterative process akin to backpropagation. This process, known as "Neural Architecture Search", offers intriguing prospects for the future of automated machine learning [5] and artificial intelligence.

In this paper, we aim to provide a comprehensive examination of various Neural Network architectures, delving into their components and their underlying processes. We will then focus on the property of Universal Function Approximation in these networks, discussing its implications and its future prospects. Through this exploration, we hope to shed light on the intricacies of these powerful tools, and speculate on the potential developments, setting the stage for future research and development in the field.

2 History of Neural Networks

Neural networks, although commonly associated with recent advancements in technology, have roots that reach back to the mid-twentieth century. Inspired by the workings of the human brain, researchers and scientists sought ways to emulate the brain's capacity for learning and pattern recognition.

3 History

The evolution of neural networks, often equated with the broader field of artificial intelligence, can be seen as unfolding in distinct stages or "waves". Each wave has played a critical role in advancing our current understanding and capabilities.

The idea of multiple waves of AI was first detailed by John Launchbury [6], but we will try to segment them in our own way.

3.1 First Wave: Birth of the Concept

The first wave, beginning in the late 1940s and extending into the 1980s, marked the origination of the "neural network" concept [7]. This phase is notable for its initial explorations into models inspired by the brain's structure and function. The groundbreaking Perceptron, developed by Frank Rosenblatt in 1957 [8], serves as a prime example of this era. Nevertheless, these initial models were significantly simplified due to constraints in computational power and theoretical understanding, and they lacked the ability to execute complex tasks [9].

3.2 Second Wave: Recognition and Backpropagation

The period from the 1980s to the late 2010s, recognized as the second wave, marked a time of significant advancement in the theoretical understanding and practical applications of neural networks. A pivotal development during this period was the backpropagation algorithm, which facilitated the training of multi-layered networks and addressed some of the limitations of the Perceptron model [10]. The recognition of the transformative potential of neural networks spurred a proliferation of research and application during this wave. However, despite these strides, advancement during this wave was still constrained by limitations in computational power and a scarcity of extensive datasets [11].

3.3 Third Wave: Contextual Adaptation and Modern Era

Starting with the introduction of the Transformer model [12] and extending to the present day, the third wave is marked by the availability of powerful GPUs, large-scale datasets, and the refinement of algorithms . This wave, often synonymous with the deep learning era, has seen neural networks become a dominant force in machine learning. Novel architectures like the Transformer and Diffusion models have thrust AI into mainstream applications, ranging from image and speech recognition to natural language processing and autonomous driving, thus influencing various sectors.

However, the defining attribute of the third wave AI systems is their capability for contextual adaptation. These systems are designed to understand context and meaning, adjusting their responses appropriately. They can not only identify an object but also elucidate why it is identified as such and the reasoning behind this conclusion. This capability marks a significant departure from the "black box" systems of the second wave. Third wave AI systems are capable of learning in a manner more reminiscent of human learning, using descriptive, contextual models rather than extensive sets of labeled training data.

3.4 Towards the Fourth Wave

As we look towards the future, we anticipate the advent of the fourth wave of AI: human-level intelligence and beyond. We envision personal AI assistants capable of conversing in natural language, AI research scientists that learn independently, intelligent tutoring systems, and domestic robots with sophisticated capabilities. While these may appear as science fiction today, numerous entities worldwide are striving to materialize the next generation of artificial intelligence solutions.

In the context of this paper, we also speculate on AutoML [5] and "Universal Model Approximation".

4 A Brief Mathematical Overview of Neural Networks

Neural Networks are mathematical models comprising layers of neurons. Each neuron performs a mathematical operation that takes an input, performs a weighted sum, and follows it with a non-linear transformation.

4.1 Linear and Non-Linear Transformations

At the core of a neuron's operation is the concept of linear and non-linear transformations. Given an input vector $\boldsymbol{x}$ of dimension d, a weight matrix $\boldsymbol{W}$ of dimension $d \times k$, and a bias vector $\boldsymbol{b}$ of dimension

k, a layer of neurons first performs a linear transformation on $\boldsymbol{x}$ as follows:

$$\boldsymbol{z} = \boldsymbol{W}^T \boldsymbol{x} + \boldsymbol{b}$$

Here, $\boldsymbol{z}$ is a k-dimensional vector known as the pre-activation vector, and each element z_i is the linear transformation performed by the i-th neuron in the layer. The dot product $\boldsymbol{W}^T \boldsymbol{x}$ allows the neurons to learn patterns in the input space.

Next, a non-linear transformation is applied element-wise to $\boldsymbol{z}$ via an activation function f, resulting in the output vector $\boldsymbol{a}$:

$$\boldsymbol{a} = f(\boldsymbol{z})$$

The activation function f introduces non-linearity into the model, allowing it to learn complex, hierarchical patterns. Commonly used activation functions include the sigmoid function $\sigma(z) = \frac{1}{1+e^{-z}}$, ReLU (Rectified Linear Unit) $f(z) = \max(0, z)$, and tanh (hyperbolic tangent) function $f(z) = \tanh(z)$.

4.2 Gradient Descent

Training a neural network involves tuning its weights and biases based on the error it produces. This is done using an algorithm called gradient descent, which iteratively adjusts the parameters of the model in the direction that minimizes the cost function.

Mathematically, in each iteration, the parameters are updated as follows:

$$\theta = \theta - \alpha \nabla J(\theta)$$

where:

- θ represents the vector of parameters (weights and biases) of the model,
- α is the learning rate, determining the size of steps taken towards the minimum,
- $J(\theta)$ is the cost function,
- $\nabla J(\theta)$ is the gradient of the cost function with respect to the parameters, also a vector whose elements correspond to the partial derivatives of the cost function with respect to each parameter in θ.

In essence, gradient descent moves in the direction of steepest descent in the cost function. This process is visualized as a ball rolling down the hill in a 3D graph representing the cost function.

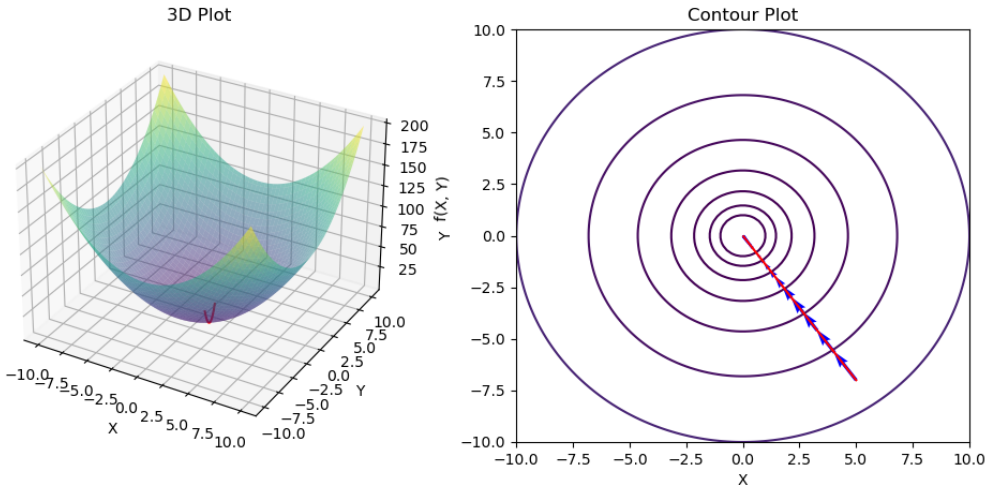

Figure 1: 50 iterations of gradient descent locating the global minimum of this function.

NOTE: Gradient Descent will not always find a **global minimum**. It is only guaranteed to find a **local minimum** for a function.

4.3 Backpropagation

The computation of the gradient of the cost function is achieved through backpropagation, an algorithm that calculates the gradient of the loss with respect to the weights and biases in the network. This is done by applying the chain rule to propagate gradients of the cost backwards through the network.

Starting from the final layer, the partial derivatives of the loss function with respect to the weights and biases are computed and stored. These values are then used to compute the gradients in the preceding layer, and this process continues until gradients for all parameters have been calculated. These gradients are then used to update the weights and biases during the gradient descent step.

5 Current Neural Network Architectures

5.1 Feed-Forward Neural Networks

Feed-Forward Neural Networks, often simply referred to as Neural Networks, provide the cornerstone for most neural network models. In this architecture, information moves in one direction from the input layer, through one or more hidden layers, to the output layer. Each layer can be seen as performing a transformation on the input, refining the data progressively with each layer's weights and biases to produce a final output.

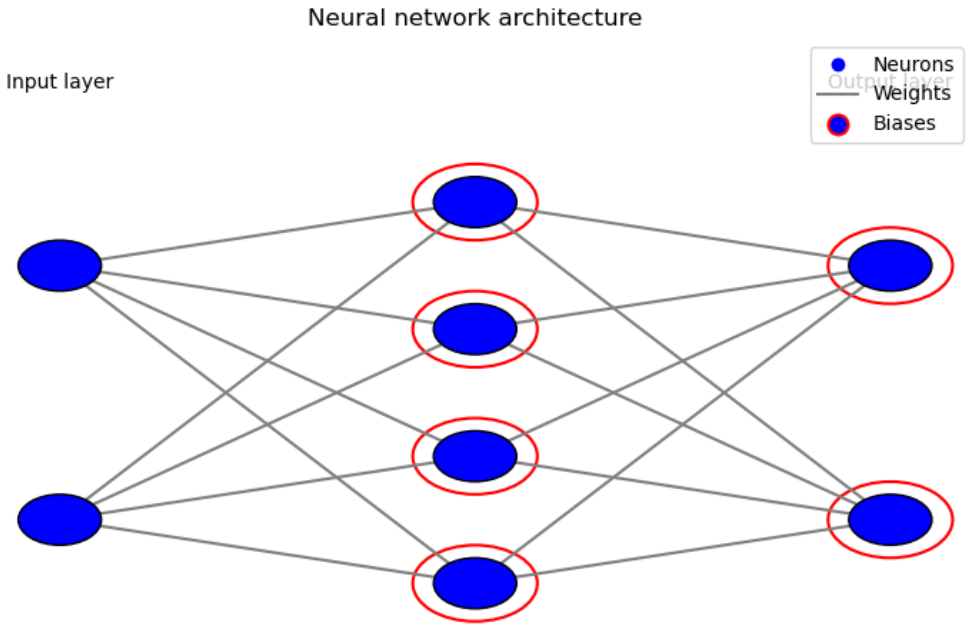

Figure 2: A simple 3-layer neural network with 2 input neurons, 4 hidden neurons, and 2 output neurons.

5.2 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) are a form of feed-forward neural networks that have proven exceptionally useful for processing multi-dimensional data, notably images. CNNs trace their roots back to early attempts at machine vision, which relied on simple linear classifiers. However, these classifiers were incapable of capturing the hierarchical nature of visual data, prompting the need for more sophisticated methods.

The development of CNNs can be seen as a leap forward in this regard. Central to their design is the convolutional layer, a unique construct that performs a convolution operation to extract high-level features from input data. This operation enables CNNs to automatically and adaptively learn spatial hierarchies of features, a significant advance over manual feature extraction methods prevalent in the era of linear classifiers.

A major proponent of this novel architecture was Yann LeCun, who, in collaboration with his colleagues, proposed LeNet-5 [13], one of the earliest and most influential CNN models. LeNet-5 was specifically designed for handwriting and character recognition, and it paved the way for more complex and effective architectures such as AlexNet [1], VGG [14], and ResNet [15].

CNNs have since found immense success in numerous applications, particularly in computer vision tasks such as image classification, object detection, and semantic segmentation. Their ability to deal with the complex structure of real-world visual data has had a profound impact on the capabilities of artificial intelligence systems.

5.3 Autoencoders

Autoencoders are a type of neural network used for unsupervised learning tasks such as dimensionality reduction and noise reduction. They are comprised of an encoder, which compresses the input into a lower-dimensional code, and a decoder, which reconstructs the input data from this code. Their

ability to learn data-driven, highly compressed representations make them valuable for tasks such as anomaly detection and generative modeling.

5.4 Generative Adversarial Networks (GANs)

Generative Adversarial Networks (GANs) consist of two distinct models: a Generator, which creates new data instances, and a Discriminator, which attempts to distinguish the generated data from real, true instances. The two models are trained together, with the Generator learning to produce more realistic output to fool the Discriminator, thereby leading to the generation of high-quality synthetic data.

5.5 Recurrent Neural Networks (RNNs)

Recurrent Neural Networks (RNNs) are designed to process sequential data, making them particularly useful for tasks such as time-series prediction and natural language processing. They achieve this by incorporating a 'hidden state' from previous inputs into the current input's processing, thereby maintaining a form of 'memory' about past inputs.

5.6 Graph Neural Networks (GNNs)

Graph Neural Networks (GNNs) are a novel type of neural network designed to process graph-structured data. They can aggregate information from a node's neighbors and capture the complex relationships in the graph data. This makes GNNs powerful tools for tasks in social network analysis, recommendation systems, and biological network analysis.

5.7 Transformers

Transformers, introduced by Vaswani et al. in "Attention is All You Need" [12], are a type of neural network architecture that has shown exceptional effectiveness in handling sequence data, especially text. At the heart of the Transformer model is the self-attention mechanism, a method that allows the model to weigh different parts of a sequence differentially. This mechanism enables the model to focus on the most pertinent portions of the input sequence for any given output.

Unlike Recurrent Neural Networks, Transformers avoid the sequential computation inherent in RNNs, allowing for parallel processing of sequences, which can significantly speed up training times. They also effectively handle long-range dependencies in the input data, overcoming the issue of vanishing gradients, which can hinder the training of deep RNNs.

Transformers have ushered in a revolution in the field of natural language processing, spawning a plethora of models based on their architecture, such as BERT [16] for understanding the context of words in sentences and GPT-3 [17] for generating human-like text. These models have achieved state-of-the-art results on numerous benchmarks and have been integral in applications ranging from machine translation to content creation and beyond.

The advent of Transformer-based models underscores the potential of this architecture, which continues to be at the forefront of advancements in machine learning and artificial intelligence.

5.8 Diffusion Models

Diffusion models are a type of generative model that transform a simple initial distribution, such as Gaussian noise, into a complex data distribution through a learned iterative process [18]. They utilize an underlying stochastic differential equation (SDE) to guide this transformation, incrementally adding 'information' at each step to morph the initial noise into a sample resembling the target distribution.

A compelling application of diffusion models is in the field of image generation, where they have been combined with text encoders like OpenAI's CLIP [19] to generate images from textual descriptions. CLIP is designed to comprehend images and texts in a unified embedding space, effectively performing translations between visual and textual data.

In this cooperative setup, the diffusion model generates an image, while CLIP evaluates the image against the text description and provides feedback. This feedback is then used to fine-tune the generation process, encouraging the diffusion model to produce images more in line with the supplied text.

The ability of diffusion models to work synergistically with encoders like CLIP presents exciting possibilities for controlled and high-quality data generation, attesting to the adaptability and versatility of neural networks.

6 Components of Neural Networks

6.1 Layers

Neural network layers house the neurons that are fine-tuned during the training process to obtain desired outcomes. Notable types of layers include Linear Layers (also known as Fully Connected Layers), Convolutional Layers, Pooling Layers, and Dropout Layers. Each has a unique function, such as feature detection, dimension reduction, or mitigation of overfitting.

6.2 Activation Functions

Activation functions introduce non-linearity into the network, enhancing its ability to model complex relationships. These functions are applied to the output of each neuron, dictating their activation level based on the input they receive. Examples include the Sigmoid, Tanh, and ReLU functions, each having unique characteristics and use cases.

6.3 Loss Functions

Loss functions quantify the discrepancy between the expected and actual outputs of the network. These functions form the metric that the training process seeks to minimize. Selection of a loss function depends on the task at hand, with examples including Mean Squared Error for regression tasks and Cross-Entropy Loss for classification tasks.

6.4 Optimizers

Optimizers are algorithms that adjust the parameters of the network (weights and biases) based on the output of the loss function. They implement the backpropagation algorithm, exploiting automatic differentiation to compute gradients efficiently. Examples include Stochastic Gradient Descent (SGD), RMSprop, and Adam.

7 Process of Training a Neural Network

7.1 Data Pre-processing

Pre-processing involves preparing the data for the network. This can involve manual labeling, noise reduction, normalization, and other preparation methods. For example, in a transformer model, a tokenizer is used to convert text into a form that the network can process, assigning unique IDs to each word.

7.2 Model Initialization

At this stage, the model architecture is defined, and its parameters are initialized. Initialization strategies can influence the model's learning speed and final performance. The initialized model is typically transferred to a GPU for efficient computation.

7.3 Optimizer and Loss Function Selection

An optimizer and loss function are chosen for the specific task. Their selection is crucial as they directly influence how the network learns. Hyperparameters such as the learning rate for the optimizer are also initialized at this stage.

7.4 Training Loop

The model is trained on the input data in iterations known as epochs. Each epoch involves a forward pass where predictions are made, a calculation of loss, and a backward pass where gradients are calculated using automatic differentiation and the model parameters are updated.

7.5 Model Evaluation

After training, the model is evaluated against a separate test dataset. Evaluation metrics, which depend on the task, are used to assess the model's performance. These metrics provide an estimate of how the model is likely to perform on unseen real-world data.

8 Universal Function Approximation in Neural Networks

One of the most critical aspects to understand about neural networks is their ability to serve as Universal Function Approximators. This concept, foundational to the field of machine learning, illuminates the profound capability of neural networks to approximate any function over a given range to a high degree of accuracy.

In essence, a neural network with at least one hidden layer containing a finite number of neurons can approximate continuous functions on compact subsets of $\mathbb{R}^n$, given that the activation function for the hidden layer neurons satisfies certain properties. This theorem was first proved by George Cybenko for sigmoid activation functions and later extended by Kurt Hornik to apply to a broader class of activation functions.

This property means that neural networks, given enough neurons and suitable activation functions, have the potential to model any input-output relationship, no matter how complex. This isn't limited to simple mathematical functions, but extends to diverse domains such as image classification, natural language processing, and even gameplay strategy in complex board games.

Moreover, neural networks are not merely sets of rote instructions; they are highly flexible and modular computing architectures that can adapt their internal structures to fit a wide range of input data. They accomplish this through the iterative learning process, adjusting their internal weights and biases to minimize the discrepancy between their current output and the desired output.

However, it's crucial to note that while the universal approximation theorem guarantees the existence of such a network, it does not provide any guidance on how to construct it or guarantee that the training process will necessarily find the optimal weights and biases. These remain open challenges in the field, and the quest for efficient network architectures and training methods that can fully leverage the power of universal function approximation is an active area of research.

9 Future Prospects

As we appreciate the power of Neural Networks as "Universal Function Approximators", a tantalizing prospect emerges - that of "Universal Model Approximation". Although this concept remains a speculative direction, it lights a promising path for future research. The vision is to automatically and optimally construct a neural network model, given a specific set of inputs, a pre-determined computational budget, and the desired outputs.

Such development would revolutionize how we approach machine learning and artificial intelligence, possibly relegating manual architecture selection and labor-intensive hyperparameter tuning as relics of the past. The model architecture could potentially be another variable in the optimization equation,

discovered through an iterative process similar to backpropagation which currently fine-tunes weights and biases in a network.

10 Conclusion

Neural networks stand as an indispensable cornerstone in computer science, offering a dynamic approach to function approximation when such functions are not well-defined a priori. The journey of their development, from the dawn of artificial neural networks to the contemporary era of advanced, computationally-efficient models, is a testament to continuous innovation.

These computational models allow us to transcend traditional algorithm design and tap into the immense potential of our computing resources. Neural networks offer an adaptive lens to interpret complex data, extract meaningful insights, and predict future trends, fueling progress across numerous domains and industries. As we look towards the future, the idea of Universal Model Approximation, and its relation to AutoML, offers a thrilling trajectory for the next evolution in machine learning and artificial intelligence.

Acknowledgements

Special thanks to Dr. Vignesh Narayanan, Dr. Michael Huhns, Dr. Forest Agostinelli, and Dr. Biplav Srivastava for guiding me through this initial forray into Artificial Intelligence and its potential uses.

References

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25, 2012.

[2] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, 1991.

[3] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. *International Conference on Learning Representations*, 2017.

[4] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search, 2019.

[5] Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren. Automated machine learning: Methods, systems, challenges. 2019.

[6] John Launchbury. A darpa perspective on artificial intelligence, 2017. Accessed on July 22, 2023.

[7] Warren S. McCulloch and Walter Pitts. *A Logical Calculus of the Ideas Immanent in Nervous Activity*. The University of Chicago Press, 1943.

[8] Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

[9] Marvin Minsky and Seymour Papert. *Perceptrons: An Introduction to Computational Geometry*. MIT Press, 1969.

[10] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[11] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[12] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc., 2017.

[13] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[14] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[16] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[17] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

[18] Elad Hoffer, Itay Hubara, and Nir Ailon. Deep unsupervised learning through spatial contrasting, 2018.

[19] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021.