The Final Project

Yanwei Fu

May 22, 2023

Abstract

(1) This is the final project of our course. The deadline is 5:00 pm, June 20, 2023. Please upload the report via elearning. The final project include three tasks listed in this document. In principle, we donot suggest using other topics/tasks in the final project, which will incur the discounted scores of this project.

(2) The paper must be in NIPS format (downloadable from [1]). We do not need the double-blind review.

(3) The goal of your write-up is to document the experiments you've done and your main findings. So be sure to explain the results. Generate a single pdf file of your projects and turned in along with your code. Package your code and a copy of the write-up pdf document into a zip or tar.gz file and named as Final-Project-student-id1-student-id2.[zip|tar.gz]

(4) You are open to using anything to help you finish this task.

(5) About the deadline and penalty. In general, you should submit the paper according to the deadline of each mini-project. The late submission is also acceptable; however, you will be penalized 10% of scores for each week's delay. **The submission can only be delayed for up to three days, as I have to submit the final score by June 24th 2023.**

(6) For the tasks below, we DO care about the performance on each dataset with the correct evaluation settings.

Note that:

(a) If the size of training instances is too large, you may want to apply some sampling techniques to extract a small portion of training data.

(b) If you think the dimension of features is too high, you may also want to use some techniques to do feature dimension reduction.

(c) The referring papers are listed as an introduction to the context of problems. It's not necessarily to exactly implement these papers, which is not an easy task.

1 Introduction

1.1 Collaboration Policy

You are allowed to work in a group with at most one collaborator. You will be graded on the creativity of your solutions, and the clarity with which you can explain them. If your solution does not live up to your expectations, then you should explain why and provide some ideas on how to improve it. You are free to use any third-party ideas or codes that you wish as long as it is publicly available. You must provide references to any work that is not your own in the write-up.

[1] `https://nips.cc/Conferences/2016/PaperInformation/StyleFiles`

1.2 Writing Policy

The final project (20%) is finished by one team. Each team should have up to 2 students. You will solve a real-world Big-Data problem. The final report should be written in English. The main components of the report will cover

1. Introduction to the background and potential applications (2%);
2. Review of the state-of-the-arts (3%);
3. Algorithms and critical codes in a nutshell (10%);
4. Experimental analysis and discussion of proposed methodology (5%).

Please refer to our latex example[2].

1.3 Submitting Policy

The paper must be in NIPS format. Package your code and a copy of the write-up pdf document into a zip or tar.gz file called Final-Project-student-id1-student-id2.[zip|tar.gz]. Include functions and scripts that you had used. Upload them to eLearning. In the submitted documents, you should include a readme.txt file to well explain the authors and co-workers of this project. The TAs can know the names of your works.

1.4 Evaluation of Final Projects

We will review your work on the following NIPS criteria:

Overview: You should briefly summarize the main content of this paper, as well as the Pros and Cons (advantages and disadvantages) in general. This part aims at showing that you had read and at least understand this paper.

Quality: Is the paper technically sound? Are claims well-supported by theoretical analysis or experimental results? Is this a complete piece of work, or merely a position paper? Are the authors careful (and honest) about evaluating both the strengths and weaknesses of the work?

Clarity: Is the paper clearly written? Is it well-organized? (If not, feel free to make suggestions to improve the manuscript.) Does it adequately inform the reader? (A superbly written paper provides enough information for the expert reader to reproduce its results.)

Originality: Are the problems or approaches new? Is this a novel combination of familiar techniques? Is it clear how this work differs from previous contributions? Is related work adequately referenced?

Significance: Are the results important? Are other people (practitioners or researchers) likely to use these ideas or build on them? Does the paper address a difficult problem in a better way than previous research? Does it advance the state of the art in a demonstrable way? Does it provide unique data, unique conclusions on existing data, or a unique theoretical or pragmatic approach?

[2] `http://yanweifu.github.io/courses/SLML/chap5/IEEE_TAC_2016.zip`

1.4.1 Minimum Requirements

For all the projects listed below, in general you should devise your own deep learning models which target each specific problem of each project. You should compare with the machine learning algorithms taught in this course/projects. Thus, the minimum requirements, as you can imagine, just apply and compare with these methods; and explain the advantages and disadvantages of using these methods for the project problem. Note that, your algorithms can be derived from one of these existing deep learning algorithms; and feel free to use any machine learning packages you like.

2 Task for Traning an Implicit Network: Functa

We are following the idea of "Functa", which is published by DeepMind:

- From data to functa: Your data point is a function and you should treat it like one[A]. 2022. `https://github.com/deepmind/functa`
- Spatial functa: Scaling functa to imagenet classification and generation[A]. 2023.

In deep learning, a common method to represent data in natural as binary digits in computer is to store data on a discrete grid. However, the original signals of those data are always continuous. For example, an image can be seen as a continuous mapping from 2 dimensional coordinate to spectral color. But when we store image as binary sequences, we sample the image in a discrete way and then get a point matrix sampling data of the original image. Most of neural network designs are based on such a discrete storage model, such as a convolutional network, which is essentially a discretized expression of the convolution operation of a continuous function. In recent years, many methods for implicitly representing continuous signals through neural networks have been proposed. These methods fits an implicit neural network representation by taking discrete-valued maps of individual data measurements as a dataset, and achieves decent accuracy. Based on such models, many methods for data reconstruction through implicit networks have been proposed. However, due to the generative properties of the model itself, it is easier to apply such implicit representation models to reconstruction and generation tasks than to other types of tasks. In this project, we will use images as the main model to explore the properties, capabilities, and implementability of the implicit representation of data, as well as downstream tasks.

2.1 Background Introduction

In the field of computer vision, deep learning has achieved great success. In 2012, Alex Krizhevsky, Ilya Sutskever and Geoffrey Hinton proposed the first "deep convolutional neural network", AlexNet. Since then, convolutional neural networks (CNN) have become one of the most popular mainstream models in the field of computer vision. In 2015, the fully convolutional neural network (FCN) was proposed, which proved that CNN under end-to-end and pixel-to-pixel training methods surpassed the most advanced technology at that time and paved the way for the development of subsequent semantic segmentation models.

However, this type of image deep learning model is based on the discrete representation of images. The discrete representation of images is very concise and has many advantages, but it also has many disadvantages. It is difficult for computers to accurately express a continuous spatial sequence. In such cases, many studies consider using implicit neural networks to implicitly express a continuous function by training a series of discrete parameters. In 2020, Sitzmann et al. proposed SIREN [17], a universal implicit neural network framework that can efficiently fit various continuous functions implicitly, and its accuracy in function value, gradient, and second-order moment is far higher than that of traditional methods. Based on SIREN, Dupont et al. proposed the concept of Functa [4] in 2022. They used SIREN to implicitly fit the discretized data and directly used model parameters as data for secondary training. Dupont et al. mainly focused on optimizing Functa itself, using modulation [16] method to share model parameters between data points, and using meta learning [5] method to accelerate large-scale sample fitting. In terms of training, Dupont et al. mainly focused on generation, reconstruction and other tasks, but the SIREN model itself is a fit to samples and is good at such generative tasks. Since the implicit expression of the model completely destroys the properties of natural data itself, such as local similarity in space and so on, it is difficult for convolutional methods to be used for classification and detection tasks of implicit models. Dupont et al. tested image classification performance on shapenet dataset [2], with good results. However, Dupont et al. also pointed out that in the case where implicit parameters completely destroy the data space feature structure, most models cannot be applied and can only be classified with small MLPs. To address this deficiency, Dupont et al. proposed the spacial functa model after spatial function improvement - spacial functa [1]. Spacial functa constructs a modulation vector group related to spatial coordinates and uses interpolation to map different modulation vectors to any coordinate. This method makes the implicit vector also have spatial features, greatly improving the performance of classification tasks.

Implicit neural representation has a wide range of application backgrounds and research value. It eliminates the limitations of traditional parameters such as resolution and mode, and has greater advantages in data transformation and enhancement. At the same time, implicit neural representation is more difficult to represent its characteristics than traditional representation methods, and there are still many problems to be solved.

2.2 Datasets

You can use any image dataset for training and evaluation. Since we only have 1080ti GPU, you may consider CiFAR-10 or MNIST dataset.

2.3 Implement the basic Functa

Please implement the functa, verify the fit effect for different model sizes and image sizes, and provide some visualization. In order to quantitatively measure the fitting ability of the trained model to the original image, you should restore the RGB value of the sample point corresponding to the original image from the implicit neural representation and calculate the **peak signal-to-noise ratio** between the original and restored images.

2.4 Share the model parameters across images

For datasets with similar characteristics between samples, we can train a Modulated SIREN model that shares parameters across samples on the entire dataset. The model parameters are divided into two parts, one is based on the shared parameters of the original siren model, and the other is the sample-specific parameters based on model modulation. Given a set of model shared parameters, we can train a small number of sample modulation parameters on any sample in a short period of time. Therefore, before training the modulation parameters, we need to reasonably train the shared parameters of the model. According to the method proposed in Functa, meta-learning [5] can be used to learn the model shared parameters. Meta-learning is a model-agnostic meta-learning algorithm that is compatible with any model trained using gradient descent and is suitable for a variety of different types of problems. The goal of meta-learning is to train the basic model for various learning tasks, so that it can solve new learning tasks with only a small number of training samples, with good generalization performance.In meta-learning training with SIREN shared parameters, model training is divided into inner and outer layers. Specifically, for a given multi-sample batch, the inner layer loops through each sample, freezing the SIREN shared parameters of the model and fitting the modulation parameters corresponding to the samples; After completing the inner layer training, the outer loop freezes the fitting results of the modulation parameters in the inner loop for each sample, calculates the sample loss, and performs gradient descent on the mean loss of the entire batch to fit the shared parameters

Please implement this method and provide some quantitative and qualitative comparison with single image functa.

2.5 Functa for downstream tasks

After completing the batch training of the sample SIREN model, the parameters are used as input for downstream tasks. This method of secondary training is called Functa. In the SIREN model, all the parameters contained in each hidden layer are expanded into a parameter vector, which is the hidden vector. The hidden vector can be used as a substitute for raw sample data for training and contains abstract features of the dataset itself.

You can use the Cifar-10 dataset (or a downsampled version of it) or other image dataset for classification, and test the classification performance based on the hidden vectors of images. You should split training, validation, and test set to properly validate the performance. Besides, please provide some insight into the performance compared to the model you trained in Project 2, *i.e.*, why the performance is better or worse than the model trained based on original images. You can try any model architecture to boost the performance.

Try other more downstream tasks you like to. We will give high socres to the projects of successfully utilizing functa on many downstream tasks with good performance.

3 The Task for Network Sparsity: Pruning Transformer via Sparsity

In recent years, Transformer has shown remarkable results in various applications such as Natural Language Processing (NLP) [19] and computer vision [8]. However, such results rely on millions or even billions of parameters, which limits its deployment in limited storage platforms. The goal of this project aims to effectively prune the Transformer model while preserving its performance on the CIFAR10 dataset, which can be downloaded from `https://www.cs.toronto.edu/~kriz/cifar.html`. Then you can prune the transformer model via sparse regularization.

3.1 Background Introduction

Transformer. As illustrated in Fig. 1, the transformer is composed of an encoder and a decoder. The encoder consists of encoding layers that process the input iteratively one layer after another, while the decoder consists of decoding layers that do the same thing to the encoder's output. Each encoder layer is encoding information about the relevance among parts of the inputs.

To effectively learn multiple types of relevance, the transformer incorporates multi-head attention, which is defined as:

$$\text{MultiheadedAttention}(Q, K, V) = \text{Concat}\left(\text{Attention}\left(QW_i^Q, KW_i^K, VW_i^V\right)\right) W^O,$$

where $\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^{\mathrm{T}}}{\sqrt{d_k}}\right) V$ and the matrices W_i^Q, W_i^K, W_i^V are "projection matrices" owned by individual attention head i, and W^O is a final projection matrix owned by the whole multi-headed attention head. Similar to the encoder, the decoder consists of a self-attention mechanism, an attention mechanism over the encodings, and a feed-forward neural network.

Sparse Regularization. The most common type of sparse regularization is ℓ_1 sparsity [18], which has been incorporated into neural networks [12] and applied to network pruning [9, 14]. Besides, the [7] considers the

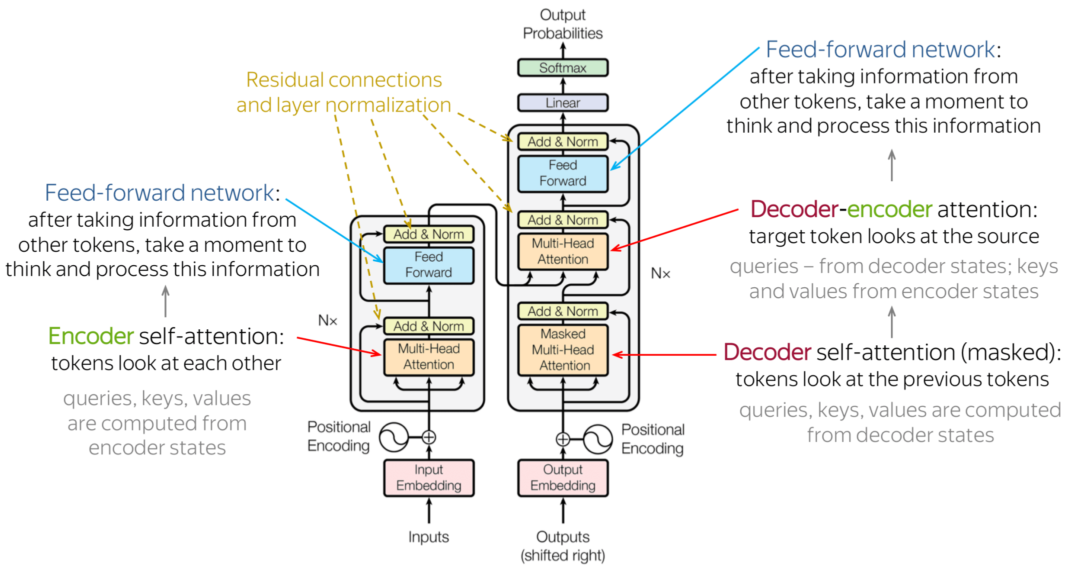

Output Probabilities
Softmax
Linear
Residual connections and layer normalization
Feed-forward network: after taking information from other tokens, take a moment to think and process this information
Add & Norm
Feed Forward
Multi-Head Attention
Masked Multi-Head Attention
Nx
Decoder-encoder attention: target token looks at the source
queries – from decoder states; keys and values from encoder states
Encoder self-attention: tokens look at each other
queries, keys, values are computed from encoder states
Decoder self-attention (masked): tokens look at the previous tokens
queries, keys, values are computed from decoder states
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)

Figure 1: Transformer architecture.

structural sparsity (including group sparsity, and layer sparsity). Recently, it has been applied to prune the transformer, by exploiting different candidates for pruning. For example, the [15] proposed to prune multiple heads; while [13] proposed to prune the tokens. In this project, you are encouraged to explore the effectiveness of these candidates and other possibilities.

3.2 Dataset and the Model

You should use CIFAR10 as the dataset, and follow `https://www.cs.toronto.edu/~kriz/cifar.html` for the train/test splitting protocol. The base model for pruning is the Vision Transformer (ViT) model [3], which is pre-trained ImageNet-21k (14 million images, 21,843 classes) at a resolution of 224×224. Then you can finetune/train from the start the transformer model on CIFAR10 with sparse regularization.

3.3 Evaluation Protocol

Please report the top 1 and top 5 accuracies on the test data, with respect to the sparsity level from 0.1 to 0.9, with 0.1 as the space interval. The sparsity level is defined as the proportion of nonzero parameters among all parameters. In general, we will like the project that can make the sparser transformer with better performance. Note that the evaluation protocol is referring to Sec 7.2 "Experiments on Network Sparsification" of [7].

4 The Task for Optimization: questions following Project 2.

Following Sec. 2.3 of our Project 2, there are some extra point.

4.1 Gradient Predictiveness

Similarly, to illustrate the increase in the stability and predictiveness of the gradients, you will make analogous measurements for the ℓ_2 distance between the loss gradient at a given point of the training and the gradients corresponding to different points along the original gradient direction. At this time you still need to choose different learning rates to train different models, but the difference is that this time the gradient value of the loss (back-propagated back to the output layer) is saved instead of the loss value, of course you can use the same model as Sec. 2.3.1. Then you should calculate the L2 distance between these gradients and take the maximum a nd the minimum (you can try other step size and show your results and comments).

Similarly, please write a function VGG_Grad_Pred() to get clear comparison, and as before, you need to report the parameters you selected and the experimental results. If you can add your comments and thoughts, it will make the report more complete.

4.2 "Effective" β-Smoothness

To further demonstrate the effect of BN on the stability/Lipschitzness of the gradients of the loss, we should plot in the "effective" β-smoothness of the standard and BN networks throughout the training. The "effective" β-smoothness refers to the maximum difference (in ℓ_2-norm) in gradient over distance moved in that direction.

Similarly, please write a function VGG_Beta_Smooth() to get clear comparison, and as before, you need to report the parameters you selected and the experimental results. If you can add your comments and thoughts, it will make the report more complete.

4.3 Try to make a new optimizing solver

Nowadays the great success of deep learning relies on the over-parameterization. Though the over-parameterization can benefit the training process, it brings difficulties when using deep neural networks in our daily life due to a large quantity of parameters. As mentioned in our lecture, instead of using backward selection methods to get a compact model, we can use DessiLBI [6] to conduct a forward selection that can get the trained over-parameterized model and sparse structure simultaneously.

For DessiLBI, the important sparse structures are discovered by the augmented variables Γ. For shallow networks and simple datasets such as LeNet on MNIST, the vanilla DessiLBI can get a good structure as shown in Figure. 2. (Please refer to Figure 1 in [6].) The first step of this part is to find a good structure of LeNet on MNIST using DessiLBI. The example code is contained in `https://github.com/DessiLBI2020/DessiLBI/tree/master/DessiLBI/`. The setting of hyperparameters can be found in [6]. Example code for training LeNet on MNIST is also contained in this repo.

After finishing the first step, you can try to combine DessiLBI with other optimization algorithm such as Adam [11] and use the combined one to find sparse structure on MNIST and CIFAR-10. (Both MNIST and CIFAR-10 can be obtained by using torchvision, please refer to `https://pytorch.org/vision/stable/datasets.html`). You are required to report both the accuracy and the structure for these experiments.

For instance, we can combine Adam [11] with DessiLBI. You can view the implementation of Adam in Pytorch (`https://pytorch.org/docs/1.2.0/_modules/torch/optim/adam.html#Adam`), then you can write a similar update for DessiLBI. Here you are required to alter the update of W in DessiLBI and keep the update of Γ the same as the original one. Figure. 3 shows the core of Adam implementation in Pytorch. In this experiment, you can consider using Adam or other adaptive gradient methods to update the W of DessiLBI in the process of optimizing the deep networks. The possible choices of deep networks could be ResNet-56 [10], VGG-19, and other more advanced deep structures. You can use MNIST and CIFAR10 dataset.

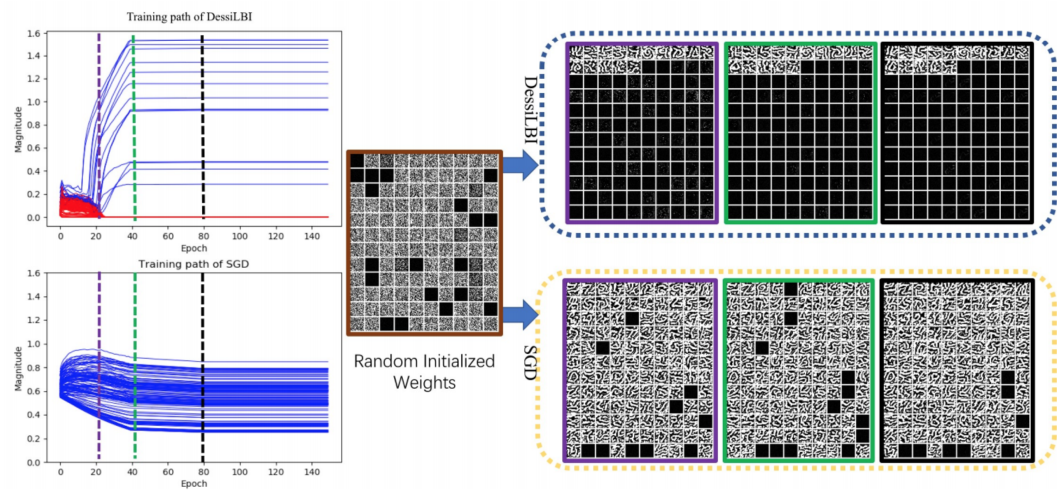
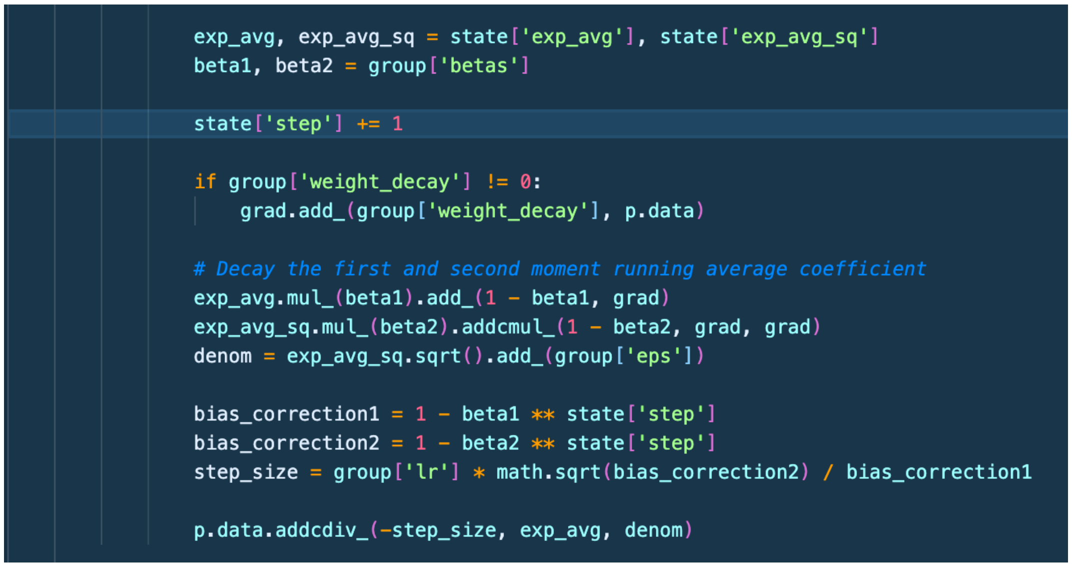


Figure 2: Illustration of training LeNet on MNIST.

Of course, it would be very nice to change the way of updating Γ by Adam or other adaptive gradient methods. This would be very interesting, but not the required in this project.



```
exp_avg, exp_avg_sq = state['exp_avg'], state['exp_avg_sq']
beta1, beta2 = group['betas']

state['step'] += 1

if group['weight_decay'] != 0:
    grad.add_(group['weight_decay'], p.data)

# Decay the first and second moment running average coefficient
exp_avg.mul_(beta1).add_(1 - beta1, grad)
exp_avg_sq.mul_(beta2).addcmul_(1 - beta2, grad, grad)
denom = exp_avg_sq.sqrt().add_(group['eps'])

bias_correction1 = 1 - beta1 ** state['step']
bias_correction2 = 1 - beta2 ** state['step']
step_size = group['lr'] * math.sqrt(bias_correction2) / bias_correction1

p.data.addcdiv_(-step_size, exp_avg, denom)
```

Figure 3: Adam Implementation.

References

[1] Matthias Bauer, Emilien Dupont, Andy Brock, Dan Rosenbaum, Jonathan Schwarz, and Hyunjik Kim. Spatial functa: Scaling functa to imagenet classification and generation. *arXiv preprint arXiv:2302.03130*, 2023.

[2] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[3] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[4] Emilien Dupont, Hyunjik Kim, SM Eslami, Danilo Rezende, and Dan Rosenbaum. From data to functa: Your data point is a function and you can treat it like one. *arXiv preprint arXiv:2201.12204*, 2022.

[5] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

[6] Yanwei Fu, Chen Liu, Donghao Li, Xinwei Sun, Jinshan Zeng, and Yuan Yao. Dessilbi: Exploring structural sparsity of deep networks via differential inclusion paths. In *International Conference on Machine Learning*, pages 3315–3326. PMLR, 2020.

[7] Yanwei Fu, Chen Liu, Donghao Li, Zuyuan Zhong, Xinwei Sun, Jinshan Zeng, and Yuan Yao. Exploring structural sparsity of deep networks via inverse scale spaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(2):1749–1765, 2022.

[8] Kai Han, Yunhe Wang, Hanting Chen, Xinghao Chen, Jianyuan Guo, Zhenhua Liu, Yehui Tang, An Xiao, Chunjing Xu, Yixing Xu, et al. A survey on vision transformer. *IEEE transactions on pattern analysis and machine intelligence*, 45(1):87–110, 2022.

[9] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28, 2015.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

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

[12] Ismael Lemhadri, Feng Ruan, and Rob Tibshirani. Lassonet: Neural networks with feature sparsity. In *International Conference on Artificial Intelligence and Statistics*, pages 10–18. PMLR, 2021.

[13] Ling Li, David Thorsley, and Joseph Hassoun. Sait: Sparse vision transformers through adaptive token pruning. *arXiv preprint arXiv:2210.05832*, 2022.

[14] Huizi Mao, Song Han, Jeff Pool, Wenshuo Li, Xingyu Liu, Yu Wang, and William J Dally. Exploring the granularity of sparsity in convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 13–20, 2017.

[15] Archit Parnami, Rahul Singh, and Tarun Joshi. Pruning attention heads of transformer models using a* search: A novel approach to compress big nlp architectures. *arXiv preprint arXiv:2110.15225*, 2021.

[16] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[17] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33:7462–7473, 2020.

[18] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[19] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.