Self-Attention Between Datapoints: Going Beyond Individual Input-Output Pairs in Deep Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

We challenge a common assumption underlying most supervised *deep learning*: that a model makes a prediction depending only on its parameters and the features of a *single input*. To this end, we introduce a general-purpose deep learning architecture that takes as input the *entire dataset* instead of processing one datapoint at a time. Our approach uses self-attention to reason about relationships between datapoints explicitly, which can be seen as realizing non-parametric models using parametric attention mechanisms. However, unlike conventional non-parametric models, we let the model learn end-to-end from the data how to make use of other datapoints for prediction. Empirically, our models solve cross-datapoint lookup and complex reasoning tasks unsolvable by traditional deep learning models. We show highly competitive results on tabular data, early results on CIFAR-10, and give insight into how the model makes use of the interactions between points.

1 Introduction

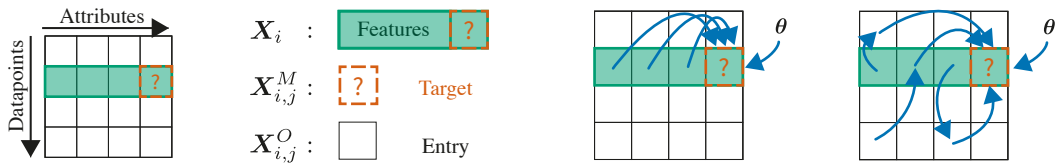
From CNNs [49] to Transformers [76], most of supervised deep learning relies on *parametric* modeling: models learn parameters $\boldsymbol{\theta}$ from a set of training data $\mathcal{D}_{\text{train}} = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_n, \boldsymbol{y}_n)\}$ to maximize training likelihoods $p(\boldsymbol{y} \mid \boldsymbol{x}; \boldsymbol{\theta})$ mapping from features $\boldsymbol{x} \in \mathcal{X}$ to target values $\boldsymbol{y} \in \mathcal{Y}$. At test time, they then make a prediction $p(\boldsymbol{y}^* \mid \boldsymbol{x}^*; \boldsymbol{\theta})$ that depends only on those parameters $\boldsymbol{\theta}$ and the test input $\boldsymbol{x}^*$. That is, parametric models do not consider direct dependencies between datapoints.

This paper challenges parametric modeling as the dominant paradigm in deep learning. Based on the same end-to-end learning motivations that underpin deep learning itself, we consider giving models the *additional flexibility* of using training data *directly* when making predictions $p(\boldsymbol{y}^* \mid \boldsymbol{x}^*, \mathcal{D}_{\text{train}}; \boldsymbol{\theta})$.

Concretely, we introduce **Non-Parametric Transformers (NPTs)**: a general deep learning architecture that takes the entire dataset as input and predicts by explicitly *learning* interactions between datapoints (Fig. 1). NPTs leverage both parametric and *non*-parametric predictive mechanisms, with the use of end-to-end training allowing the model to naturally learn from the data how to balance the two. Namely, instead of just learning predictive functions from the features to the targets of independent datapoints, NPTs can also learn to reason about general relationships *between* inputs. We show that these models *learn* to look up information from other datapoints and capture the causal mechanism generating the data in semi-synthetic settings. However, unlike conventional non-parametric models, NPTs are not forced to *only* make predictions in this manner: they can also use the power of conventional parametric deep learning. We use multi-head self-attention [4, 51, 76] to model relationships between datapoints and construct a training objective for NPTs with a stochastic masking mechanism inspired by recent work in natural language processing [23].

A key contribution of this paper is opening the door to more general treatment of how deep learning models can make use of dependencies between datapoints for predictions. Our results demonstrate that NPTs make use of interactions between datapoints in practice, and we show highly competitive

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.



(a) Input Data (b) Notation (c) Parametric Model (d) **NPT**

Figure 1: NPTs learn direct interactions between datapoints. (a) Input data: predict masked target entry **[?]** for datapoint $\boldsymbol{X}_i$. (b) Notation from §2. (c) Parametric models predict only from the features of the given input. (d) NPTs predict by modeling relationships between all points in the dataset.

performance on several established tabular datasets as well as early image classification results. Additionally, we show that NPTs can solve complex reasoning tasks by combining representation learning and cross-datapoint lookup; something that is impossible for conventional deep learning or non-parametric models due to their inability to *learn* relations *between* datapoints.

Background. While questioning parametric modeling assumptions is unconventional in deep learning, in statistics so-called *non-parametric* models are a well-known and long-established field of study. Non-parametric models make predictions in explicit dependence of the training data $p(\boldsymbol{y}^* \mid \boldsymbol{x}^*, \mathcal{D}_{\text{train}})$. The most popular example of such models in the machine learning community are perhaps Gaussian Processes [64]. Non-parametric models typically do not require any training of parameters, and instead often directly interpolate between training points according to a fixed procedure, e.g., [64, p.17]. The interactions between inputs are fully defined by architectural choices and a small set of hyperparameters that must be carefully chosen. Conventional non-parametric models cannot *learn* – in the sense familiar to deep learning practitioners – interactions from the data, limiting the flexibility these models have in adapting to the data at hand. Approaches such as Deep Gaussian Processes [21], Deep Kernel Learning [78], and Neural Processes [32, 33, 42] have all sought to apply ideas from deep neural networks to non-parametrics. Compared to NPTs, these approaches rely heavily on motivations from stochastic processes. This leads to them being either less flexible than NPTs or requiring strong assumptions on the data, making them *inapplicable* to the practical scenarios considered in this paper (cf. §3). Unlike previous work, NPTs explicitly learn interactions between datapoints and can be applied to general supervised machine learning tasks. We refer to §3 for an overview of these and other related approaches.

We next discuss the specifics of our model (§2), before moving on to related work (§3), empirical results (§4), and finally, limitations, future work, and conclusions (§5).

2 Non-Parametric Transformers

Non-Parametric Transformers (NPTs) explicitly *learn* relationships between datapoints to improve predictions. To accomplish this, they rely on three main ingredients: **(1)** We provide the model with the **entire dataset – all datapoints – as input**. At test time, both training and test data are input to the model; during training, the model learns to predict targets from the training data only. We approximate this where necessary for large data (§2.6). **(2)** We use **self-attention between datapoints** to explicitly model relationships between them. For example, at test time, the attention mechanism models relationships amongst training points, amongst test points, and between the two. **(3)** NPT's training objective is to reconstruct a corrupted version of the input dataset. Similar to BERT [23], we apply **stochastic masking** to both features and targets and minimize a loss on NPT's predictions at entries masked out in the input. Next, we introduce the three components in detail.

2.1 Datasets as Inputs

NPTs take as input the entire dataset $\boldsymbol{X} \in \mathbb{R}^{n \times d}$. The datapoints are stacked as the rows of this matrix $\{\boldsymbol{X}_{i,:} \in \mathbb{R}^d \mid i \in 1 \ldots n\}$, and we refer to the columns as attributes $\{\boldsymbol{X}_{:,j} \in \mathbb{R}^n \mid j \in 1 \ldots d\}$. Each attribute is assumed to share a semantic meaning among all datapoints. In single-target classification and regression, we assume that the targets (labels) are the final attribute $\boldsymbol{X}_{:,d}$, and the other attributes $\{\boldsymbol{X}_{:,j} \mid j \neq d\}$ are input features, e.g., the pixels of an image. Each $\boldsymbol{X}_{i,j}$ is an entry or value. In addition to tabular data, many modalities such as images, graphs, or timeseries can be reshaped to fit this format. Note that this is a departure from common notation for supervised learning as introduced in §1, as the input $\boldsymbol{X}$ now includes both features and targets (collectively, attributes).

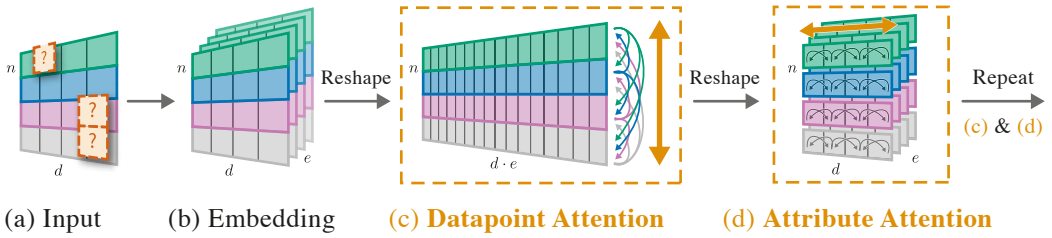


Figure 2: Overview of the Non-Parametric Transformer. (a) The input dataset and mask matrix are stacked and (b) linearly embedded for all datapoints independently. NPT then applies (c) Attention Between Datapoints (ABD, §2.4) across all n samples of hidden dimension $h = d \cdot e$. (d) Attention Between Attributes (ABA, §2.5) then attends between the attributes for each datapoint independently. We repeat steps (c) and (d) and obtain a final prediction from a separate linear projection (not shown).

In masked language modeling [23], mask tokens denote which words in a sentence should be concealed and where model predictions will have a loss backpropagated at training time. Analogously, we use a binary matrix $\boldsymbol{M} \in \mathbb{R}^{n \times d}$ to specify which entries are *masked* in the input $\boldsymbol{X}$. This matrix is also passed to NPT as input. The task is to predict the masked values $\boldsymbol{X}^M = \{\boldsymbol{X}_{i,j} \mid \boldsymbol{M}_{i,j} = 1\}$ from the observed values $\boldsymbol{X}^O = \{\boldsymbol{X}_{i,j} \mid \boldsymbol{M}_{i,j} = 0\}$, i.e., to predict $p(\boldsymbol{X}^M \mid \boldsymbol{X}^O)$.

In summary, NPT takes as input the entire dataset and masking matrix $(\boldsymbol{X}, \boldsymbol{M})$, and makes predictions $\hat{\boldsymbol{X}} \in \mathbb{R}^{n \times d}$ for values masked at input. This general setup accommodates many machine learning settings simply by adjusting the placement of the binary masks in $\boldsymbol{M}$. We focus on single-target classification and regression – corresponding to a masking matrix $\boldsymbol{M}$ with 1s at all entries of the label column $\boldsymbol{X}_{:,d}$ – but outline multi-target settings, imputation, self-supervision using input features, and semi-supervision in Appendix C.6. Next, we describe the NPT architecture.

2.2 NPT Architecture

An overview of the Non-Parametric Transformer (NPT) is depicted in Fig. 2. NPT receives the dataset and masking matrix $(\boldsymbol{X}, \boldsymbol{M})$ as input (Fig. 2a). We stack these and apply an identical linear embedding to each of n datapoints, obtaining an input representation $\boldsymbol{H}^{(0)} \in \mathbb{R}^{n \times d \times e}$ (Fig. 2b). Next, we apply a sequence of multi-head self-attention layers [4, 23, 76]. Crucially, we alternatingly apply attention between *datapoints*, and attention between *attributes* of individual datapoints (Figs. 2c-d).

These operations allow our model to learn both relationships between datapoints as well as transformations of individual datapoints. Finally, an output embedding gives the prediction $\hat{\boldsymbol{X}} \in \mathbb{R}^{n \times d}$, which now has predicted values at entries that were masked at input. We refer to Appendix C.3 for details, such as treatment of categorical and continuous variables. Importantly:

Property 1. *NPTs are equivariant to a permutation of the datapoints. (cf. Appendix A for proof.)*

In other words, if the set of input datapoints are shuffled, NPTs produce the same predictions but shuffled in an analogous manner. This explicitly encodes the assumption that the learned relations between datapoints should not depend on their ordering. At a high level, permutation-equivariance (PE) holds because all components of NPT are PE, and the composition of PE functions is PE. We now briefly recap multi-head self-attention, an important operation in the NPT architecture.

2.3 Multi-Head Self-Attention

Multi-head self-attention (MHSA) is a powerful mechanism for learning complex interactions between elements in an input sequence. Popularized in natural language processing [4, 23, 76], MHSA-based models have since been successfully applied to many areas of machine learning (cf. §3).

Dot-product attention computes attention weights by comparing queries $\{\boldsymbol{Q}_i \in \mathbb{R}^{1 \times h_k} \mid i \in 1 \dots n\}$ with keys $\{\boldsymbol{K}_i \in \mathbb{R}^{1 \times h_k} \mid i \in 1 \dots m\}$, ultimately updating the representation of the queries by aggregating over values $\{\boldsymbol{V}_i \in \mathbb{R}^{1 \times h_v} \mid i \in 1 \dots m\}$ via the attention weights. We stack the queries, keys, and values into matrices $\boldsymbol{Q} \in \mathbb{R}^{n \times h_k}$, $\boldsymbol{K} \in \mathbb{R}^{m \times h_k}$, and $\boldsymbol{V} \in \mathbb{R}^{m \times h_v}$ and, as is commonly done for convenience, assume $h_k = h_v = h$. Then, we compute dot-product attention as

$$\text{Att}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}(\boldsymbol{Q}\boldsymbol{K}^T / \sqrt{h})\boldsymbol{V}. \tag{1}$$

Multi-head dot-product attention concatenates a series of k independent *attention heads*

$$\text{MHAtt}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \underset{\text{axis}=h}{\text{concat}}(\boldsymbol{O}_1, \ldots, \boldsymbol{O}_k)\boldsymbol{W}^O, \text{ where } \boldsymbol{O}_j = \text{Att}(\boldsymbol{Q}\boldsymbol{W}_j^Q, \boldsymbol{K}\boldsymbol{W}_j^K, \boldsymbol{V}\boldsymbol{W}_j^V). \quad (2)$$

We learn embedding matrices $\boldsymbol{W}_j^Q, \boldsymbol{W}_j^K, \boldsymbol{W}_j^V \in \mathbb{R}^{h \times h/k}, j \in \{1, \ldots, k\}$ for each head j, and $\boldsymbol{W}^O \in \mathbb{R}^{h \times h}$ mixes outputs from different heads. Here, we focus on multi-head *self*-attention, $\text{MHSelfAtt}(\boldsymbol{H}) = \text{MHAtt}(\boldsymbol{Q} = \boldsymbol{H}, \boldsymbol{K} = \boldsymbol{H}, \boldsymbol{V} = \boldsymbol{H})$, which uses the *same* inputs for queries, keys, and values. Following Transformer best practices to improve performance [15, 23, 51, 56, 76], we first add a residual branch and apply Layer Normalization (LN) [3] followed by $\text{MHSelfAtt}(\cdot)$,

$$\text{Res}(\boldsymbol{H}) = \boldsymbol{H}\boldsymbol{W}^{\text{res}} + \text{MHSelfAtt}(\text{LN}(\boldsymbol{H})), \quad (3)$$

with learnable weight matrix $\boldsymbol{W}^{\text{res}} \in \mathbb{R}^{h \times h}$. Then, we add another residual branch with LN and a row-wise feed-forward network (rFF), finally giving the full multi-head self-attention layer as

$$\text{MHSA}(\boldsymbol{H}) = \text{Res}(\boldsymbol{H}) + \text{rFF}(\text{LN}(\text{Res}(\boldsymbol{H})) \in \mathbb{R}^{n \times h}. \quad (4)$$

2.4 Attention Between Datapoints (ABD)

The **Attention Between Datapoints (ABD)** layer is a key operation for NPT. It explicitly transforms data by reasoning about pairwise relationships between all datapoints, see Fig. 2c. As input to ABD, we flatten the output of the previous layer $\boldsymbol{H}^{(\ell)}$ from $\mathbb{R}^{n \times d \times e}$ to $\mathbb{R}^{n \times h}$ with $h = d \cdot e$. Then, we perform multi-head self-attention between the datapoints $\{\boldsymbol{H}_i^{(\ell)} \in \mathbb{R}^{1 \times h} \mid i \in 1 \ldots n\}$ as

$$\text{ABD}(\boldsymbol{H}^{(\ell)}) = \text{MHSA}(\boldsymbol{H}^{(\ell)}) = \boldsymbol{H}^{(\ell+1)} \in \mathbb{R}^{n \times h}. \quad (5)$$

At the first ABD layer, we input $\boldsymbol{H}^{(0)} \in \mathbb{R}^{n \times d \times e}$, the linearly embedded input data. After applying ABD, we reshape the output again, from $\mathbb{R}^{n \times h}$ to $\mathbb{R}^{n \times d \times e}$.

Note that this is distinct from how $\text{MHSA}(\cdot)$ is usually applied in the literature, as we compute attention between *different datapoints* and not between the *features of a single datapoint* [23, 24, 39, 76]. For example, in natural language processing, attention is usually applied between the tokens (features) of a sentence (datapoint) but not between different sentences. For example, NPT could learn to attend between two datapoints with indices i and i' by embedding $\boldsymbol{Q}_i$ and $\boldsymbol{K}_{i'}$ in close proximity. Following (1), datapoint i will then attend more closely to i' because $\boldsymbol{Q}_i \boldsymbol{K}_{i'}^T$ will be large. By stacking many ABD layers, NPT can learn higher-order interactions between datapoints [23, 76].

2.5 Attention Between Attributes (ABA)

We now introduce **Attention Between Attributes (ABA)**, which is always performed following ABD. ABA layers can help the model learn better per-datapoint representations for the between-datapoint interactions, see Fig. 2d. In ABA, we apply MHSA independently to each row (corresponding to a single datapoint) in the input $\boldsymbol{H}_i^{(\ell)} \in \mathbb{R}^{d \times e}, i \in \{1, \ldots, n\}$, giving

$$\text{ABA}(\boldsymbol{H}^{(\ell)}) = \underset{\text{axis}=n}{\text{stack}}(\text{MHSA}(\boldsymbol{H}_1^{(\ell)}), \ldots, \text{MHSA}(\boldsymbol{H}_n^{(\ell)})) = \boldsymbol{H}^{(\ell+1)} \in \mathbb{R}^{n \times d \times e}. \quad (6)$$

Just like in standard Transformers [23, 24, 39, 76], ABA is used to transform attribute representations of single datapoints independently. We batch over the n dimension to compute ABA efficiently. By alternating between attention over datapoints (ABD) and attributes (ABA), NPTs can model both complex dependencies between points as well as learn suitable transformations of datapoints individually. Next, we describe the use of masking mechanisms during NPT training and evaluation.

2.6 Masking and Optimization

Masking. Much like in masked language modeling [23], we use masks to indicate which values NPT is expected to predict, and to prevent the model from accessing ground truth values. Recall that NPT needs to predict $p(\boldsymbol{X}^M \mid \boldsymbol{X}^O)$, with masked values $\boldsymbol{X}^M = \{\boldsymbol{X}_{i,j} \mid \boldsymbol{M}_{i,j} = 1\}$ and observed values $\boldsymbol{X}^O = \{\boldsymbol{X}_{i,j} \mid \boldsymbol{M}_{i,j} = 0\}$. Masked values can be either features or targets. Canonically, masked language modeling is used to perform self-supervised learning on a sequence of tokens in a sentence [23]. We use such *stochastic feature masking* to mask a feature value $\boldsymbol{X}_{i,j}, j \neq d$ with probability p_{feature} during training. *Stochastic target masking* is done in the same manner on the targets of the training set $\boldsymbol{X}_{:,d}$ with p_{target}. Note that we take great care to avoid test set leakage, and *never* reveal targets of the test set to NPT. Appendix C.6 gives full details on the masking procedure.

NPT Objective. During training, we compute the negative log-likelihood loss at training targets $\mathcal{L}^{\text{Targets}}$ as well as the auxiliary loss from masked-out features $\mathcal{L}^{\text{Features}}$. We write the NPT training objective as $\mathcal{L}^{\text{NPT}} = (1-\lambda)\mathcal{L}^{\text{Targets}} + \lambda\mathcal{L}^{\text{Features}}$, where λ is a hyperparameter. At test time, we only mask and compute a loss over the targets of test points. See Appendix C.7 for optimization details.

This objective has a few notable elements. Feature masking requires NPTs to make predictions over all attributes, encouraging the models to learn a representation of the entire dataset. This increases the difficulty of the task and adds more supervision, which we find tends to have a beneficial regularizing effect. Interestingly, stochastic target masking means that many training targets are *unmasked* to the model at training time. This allows NPTs to learn to predict, at each epoch, the masked targets of certain training datapoints using the *targets of other training datapoints* in addition to all training data features.[1] NPTs no longer have to memorize a mapping between training inputs and outputs in their parameters $\boldsymbol{\theta}$, and can instead use their representational capacity to learn functions using other *training features and targets as input*. For example, NPTs could learn to assign test datapoints to clusters of training datapoints, and predict on those points using interpolation of the training targets in their respective cluster. We explore the ability of NPTs to solve such complex reasoning tasks in §4.2.

Handling Large Datasets. Avoiding the poor $\mathcal{O}(n^2)$ time and space complexity of naïve self-attention, we resort to approximations once the data grows too large. For example, we reach 24 GB of GPU memory for standard NPT model sizes at about 8000 datapoints. We find that processing the data in random subsets for model training and prediction, i.e., *minibatching*, is a simple and effective solution. We construct minibatches such that, at test time, training and test data are both present in the same batch, to allow NPTs to attend to training datapoints. In §4.3, we show that NPTs make use of attention between datapoints with minibatching enabled. See §5 for further discussion and ideas for future work.

3 Related Work

Deep Non-Parametric Models. Deep Gaussian Processes [21] and Deep Kernel Learning (DKL) [78] extend ideas from Gaussian Processes [64] to representation learning. Deep GPs stack standard GPs with the aim to learn more expressive relationships between input points, sharing motivation with NPTs. However, unlike NPTs, deep GPs are difficult to work with in practice, requiring complex approximate inference schemes [13, 20, 66]. DKL applies a neural network to each datapoint *independently* before passing points on to a standard Gaussian Process, making predictions based directly on similarity in embedding space instead of *learning* the interactions themselves.

Neural Processes. Similar to GPs, Neural Processes (NPs) [32, 33] define a distribution over functions. They use a latent variable model parametrized by neural networks, fulfilling specific architectural constraints to approximately preserve consistency of finite-dimensional marginals. Attentive Neural Processes (ANPs) [42] extend Neural Processes to allow for direct attention between a context set and targets. However, as the authors themselves stress, "NPs and GPs have different training regimes" [42]. While a GP can be trained on a single dataset, *(A)NPs require multiple realizations of the dataset*. The authors further note that *"a direct comparison between the two is usually not plausible"* [42], which is why we cannot compare (A)NPs to NPT on our standard tasks.

Attention. NPTs are part of a line of recent work that explores the use of Transformer-based architectures outside of natural language processing, e.g., Transformers in computer vision [24, 39, 57] or architectures exploiting desirable invariances or equivariances [30, 37, 51, 53]. Like NPTs, Set Transformer [51] attends to a set of input points. However, unlike NPTs, Set Transformer relies on the existence of multiple independent sets for training and makes only a single prediction for each set. Like NPTs, Axial Transformers [35] and MSA Transformers [63] attend to multiple dimensions of matrix-shaped input. However, Axial Transformers process single images as input, i.e., no attention across datapoints is performed. MSA Transformers use attention within individual protein sequences and across an aligned protein family for contact prediction, but do not consider a more general setting. Recent works have improved neural network performance on tabular data using attention. AutoInt [68] is a direct application of multi-head attention to tabular data, and TabNet [2] sequentially attends to sparse subsets of the features inspired by tree-based models. Both approaches do not reason about interactions between datapoints, a key contribution that we introduce with NPT in this work.

[1] A potential concern is that the model will memorize training targets and fail to generalize. In practice, we do not observe generalization issues, likely because (i) a loss is never backpropagated on an unmasked value, and (ii) BERT-style masking [23] uses token randomization to prevent memorization. See Appendix C.6.

Table 1: Average rank order of various methods ($\pm$ standard error) on UCI benchmarks, across binary classification, multi-class classification, and regression tasks. We determine rank using the test area under the receiver operating characteristic (AUROC) curve on binary classification (4 of 10 datasets), accuracy on multi-class classification (2 of 10), and root mean squared error (RMSE) on regression (4 of 10), and sort methods by ascending rank for each metric. See Appendix B.5 for full results.

Method	AUROC
NPT	**2.50 $\pm$ 0.87**
CatBoost	2.75 $\pm$ 0.85
LightGBM	3.50 $\pm$ 1.55
XGBoost	4.75 $\pm$ 1.25
Gradient Boosting	5.00 $\pm$ 0.71
MLP	5.75 $\pm$ 1.49
Random Forest	6.00 $\pm$ 0.71
TabNet	6.50 $\pm$ 1.32
k-NN	8.25 $\pm$ 0.48

Method	Accuracy
NPT	**2.50 $\pm$ 0.50**
XGBoost	**2.50 $\pm$ 1.50**
MLP	3.00 $\pm$ 2.00
CatBoost	3.50 $\pm$ 0.50
Gradient Boosting	3.50 $\pm$ 1.50
Random Forest	6.50 $\pm$ 0.50
TabNet	7.50 $\pm$ 0.50
LightGBM	7.50 $\pm$ 1.50
k-NN	8.50 $\pm$ 0.50

Method	RMSE
CatBoost	**3.00 $\pm$ 0.91**
XGBoost	3.25 $\pm$ 0.63
NPT	3.25 $\pm$ 1.31
Gradient Boosting	4.00 $\pm$ 1.08
Random Forest	4.50 $\pm$ 0.87
MLP	5.00 $\pm$ 1.22
LightGBM	6.50 $\pm$ 1.55
TabNet	6.75 $\pm$ 0.95
k-NN	8.75 $\pm$ 0.25

Few-Shot Learning, Meta-Learning, and Prompting. In §4.2, we apply NPTs to tasks that require learning of relational structure between datapoints on training data to achieve good generalization performance on novel test inputs. This setup shares motivations with meta-learning [6, 8, 26, 48], in which a model is pre-trained on a variety of tasks, such that it can then learn new tasks using only a small number of additional training points from the new task. However, we consider evaluation without any additional gradient updates, unlike recent meta-learning methods [26, 80] which are therefore inapplicable to this setting. Recent works on few-shot learning with text prompting [12, 62] provide a trained Transformer-based language model with a few examples of a novel relationship in a prompt at prediction time, and observe strong generalization on the task. Similarly, we consider attention between a "context" of datapoints. While ground-truth input-output pairs are provided for prompting, we consider settings in which no ground-truth is given at prediction time (cf. Appendix B.1.2), but the model can solve the task if it has learned the underlying relational structure.

Due to the unique properties of NPTs, we believe that there are many other exciting connections to be drawn. We discuss a selection of possible areas of application including semi-supervised learning, graph neural networks, and relational learning in Appendix D, and leave other areas such as prediction on missing data, semi-supervised learning, and continual learning to future research. In this initial study, we instead concentrate on questions at the core of NPTs.

4 Experiments

We seek to answer the following set of questions in our evaluation[2] of NPTs: (**Q1**) How do NPTs perform on standard benchmarks for supervised machine learning? (**Q2**) Can NPTs successfully model interactions between datapoints in idealized settings? (**Q3**) Do NPTs actually learn to rely on interactions between datapoints for prediction on real-world datasets? (**Q4**) If so, what is the nature of these interactions, e.g., which other datapoints are relevant for prediction?

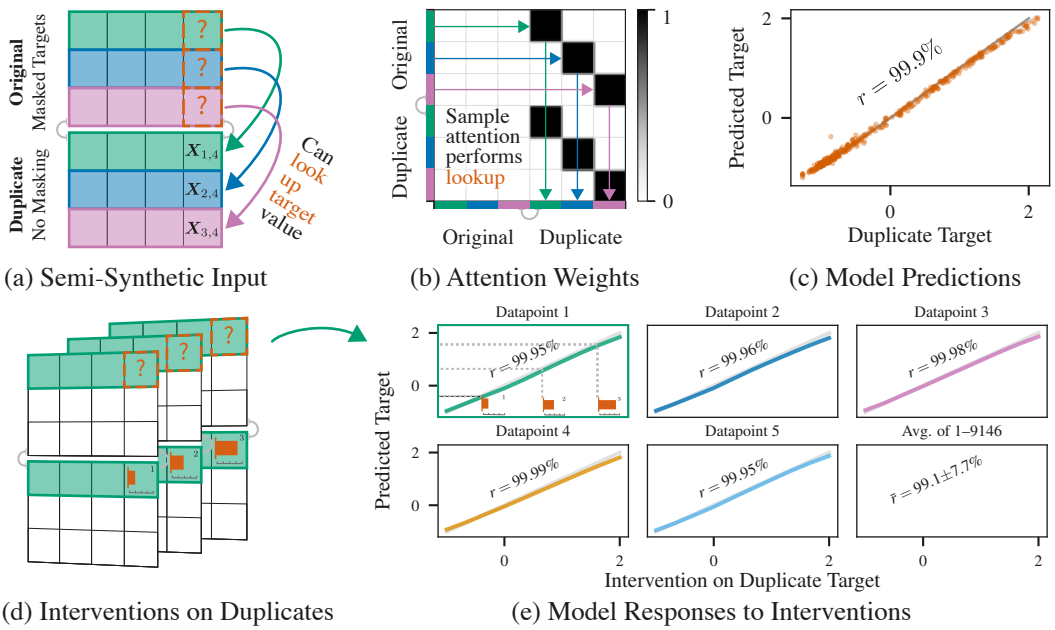
4.1 NPTs Perform Competitively on Established Benchmarks

To answer (**Q1**), we evaluate NPTs on tabular data from the UCI Repository [25] as well as the CIFAR-10 [47] and MNIST [50] image classification datasets. Tabular data is ubiquitous in real-world machine learning [19] but notoriously challenging for general purpose deep neural networks, which consistently underperform boosting models [67] and are rarely used in practice.[3]

Tabular Datasets, Setup, and Baselines. We evaluate NPTs over 10 datasets varying across the number of samples, number of features, composition (categorical or continuous) of features, and task. 4 of the 10 are binary classification, 2 are multi-class classification, and 4 are regression. We compare NPT against a wide set of standard or state-of-the-art baselines: Random Forests [10], Gradient Boosting Trees [29], XGBoost [16], CatBoost [61], LightGBM [41], MLPs, k-NN [1, 27], and TabNet [2]. For additional background on tree-based models see Appendix D.2. We

[2] We share code for NPTs, which is anonymized for review purposes, at rebrand.ly/NPTNeurIPS2021.

[3] We conduct an informal survey of all Kaggle [38] competitions using tabular data completed in 2020 with a public leaderboard. In 11 out of a total of 13 cases, the winning entries relied on some form of boosting.



(a) Semi-Synthetic Input (b) Attention Weights (c) Model Predictions

(d) Interventions on Duplicates (e) Model Responses to Interventions

Figure 3: Demonstrating NPT's ability to predict from Attention Between Datapoints (ABD). (a) We append to the original data with masked targets **[?]** a copy of the same data with all masked values revealed, such that perfect prediction via lookup is possible. (b) Attention weights indicate that the ideal lookup behavior is learned by NPT. Shown are actual values learned by NPT at head 0 and depth 4 for the first 3 datapoints. (c) NPT predictions closely match the ideal values. (d) Additionally, we intervene on the values of individual targets, (e) finding that NPT predictions adjust accordingly.

tune the parameters of all models on validation sets and use 10-fold cross-validation whenever computationally feasible. Note that while we perform an extensive grid search for the baselines, we only search over a small set of configurations for NPTs. We refer the reader to Appendix E for further details on the datasets and baseline setups, and Appendix C.1 for NPT hyperparameters.

Tabular Data Results. We report the average rank order for NPT and various tree-based and deep learning baselines in Table 1. NPT achieves the highest average ranking on binary and multi-class classification tasks, outperforming CatBoost and XGBoost, two popular state-of-the-art boosting methods designed specifically for tabular data. On regression tasks, NPT ties in average rank with XGBoost, and is outperformed only by CatBoost. In addition to its strong rank-wise performance, NPT achieves best performance on 4 of the 10 benchmark datasets – more than any other method. We find that these are remarkable results for a general purpose model that does not include tabular-specific design, supporting our hypothesis that attention between datapoints is a useful architectural inductive bias for prediction. For all metrics across all datasets, i.e., NLL for classification, AUROC/accuracy for binary/multi-class classification, and (R)MSE for regression, we refer the reader to Appendix B.5.

Image Data Results. NPT achieves 68.2% accuracy on CIFAR-10 and 98.3% accuracy on MNIST. Similar to previous work on Transformers for computer vision, we would expect (pre)-training on millions of images to significantly boost NPT's performance [22, 39, 65, 71, 74]. We perform no pre-training, and therefore a direct comparison of our results to this line of work is inappropriate. Crucially, we show in §4.3 that NPTs learn to make use of interactions between images, indicating that attention between datapoints is valuable for image classification. Appendix B.6 contains further discussion.

4.2 NPTs Can Learn to Predict Using Attention Between Datapoints

To determine if NPTs can successfully learn to exploit interactions between datapoints **(Q2)**, we introduce a task with strong input correlations for which we know ground-truth interactions. Concretely, we take the UCI Protein regression dataset (cf. §4.1), to construct the following semi-synthetic task: for each batch, we input the original data with masked target values as well as a *copy* of the original data where all target values have been revealed, i.e., no masking is applied (Fig. 3a). NPTs can use attention between datapoints to achieve arbitrarily good performance by *learning* to look up the target

Table 2: Drop in NPT performance after destroying information from other datapoints. Shown are changes in test set performance, where negative values indicate worse performance after corruption.

Δ *Accuracy*	CIFAR-10	Poker	Income	Higgs	MNIST	Forest	Kick	Breast Cancer
	−5.1	−1.1	−1.1	−0.5	−0.4	−0.1	−0.1	0.0
$\Delta_{RMSE}/_{RMSE}$ (%)	Yacht	Protein	Boston	Concrete				
	−52%	−21%	−20%	−7%				

values in the matching duplicate row. At test time, we input novel semi-synthetic test data to ensure that NPT has learned the correct relational mechanism and not just memorized target values.

NPTs successfully learn to perform this lookup between original and duplicate datapoints. The ABD attention weights, visualized for the first three datapoints in Fig. 3b, clearly show the model correctly attending to the duplicates. As a result, NPT predictions are Pearson-correlated with the duplicate targets at $r = 99.9\%$ (Fig. 3c). This equals an RMSE of only 0.44, about a magnitude lower than the error on the original Protein dataset (Table 8). We conclude that NPTs learn to predict by looking up the target values from matching points. Further discussion and attention maps are in Appendix B.1.1.

Purely parametric models cannot exploit information from other datapoints, limiting their performance. For example, MLPs achieve an RMSE of 3.62 on this task. Non-parametric approaches also cannot solve this task in its original form, because unlike NPTs they must be *told* which datapoints are the originals (training data) and which the duplicates (test data) as well as which columns contain features and which target values. We demonstrate in Appendix B.1.2 that even when we make these concessions, we can easily adapt the task such that both k-Nearest Neighbors and Deep Kernel Learning fail to solve it. In fact, we are not aware of any other model that can solve the adapted task.

Additionally, we perform an *interventional* experiment to investigate the extent to which NPTs have actually learned the causal mechanism underlying the lookup task. As illustrated in Fig. 3d, we now intervene on individual duplicate datapoints at test time by varying their target value across a wide range. We stress that we perform these experiments without retraining the model, using exactly the same NPT from Figs. 3a-c. The model is now confronted with target values associated with features that are highly unlikely under the training data. This label distribution shift [31] is a challenging setting for neural networks. However, NPT predictions follow the intervened target values with near-perfect correlation, Fig. 3e, continuing to predict by correctly looking up targets.

We now confidently conclude that NPTs robustly learn the causal data-generating mechanism underlying the semi-synthetic dataset. This requires NPTs to *learn* a non-trivial sequence of compuational steps. They must learn to match rows based on similarity of relevant features; to look up the target value of the duplicated datapoint; and, to copy that value into the target of the masked datapoint.

4.3 NPTs Learn to Use Attention Between Datapoints on Real Data

We next consider **(Q3)**: do NPTs actually learn to use attention between datapoints for prediction on real data? We design a test that allows us to quantify the extent to which NPT predictions depend on relationships between datapoints at test time. Concretely, for each target value in the input we randomize the data for all *other* datapoints by independently shuffling each of their attributes across the rows. We then evaluate the loss on the prediction at the target entry and repeat this procedure for all test datapoints. This completely corrupts the information from all datapoints except the one for which we evaluate. Hence, a model that relies meaningfully on attention between datapoints will show deteriorating performance. We give an algorithm for the corruption procedure in Appendix B.2.1.

We report the resulting change in performance after corruption in Table 2 for all datasets from §4.1. We find that for most datasets, the corruption of other rows at test time significantly decreases the performance of the trained NPT models. This indicates that the NPTs have successfully learned to make predictions supported by attention between datapoints. For some datasets, the corruption experiment deteriorates performance completely. For example, for the Protein regression dataset NPT achieves state-of-the-art performance, but corrupting the input leads to NPT performing worse than all of the baselines considered in §4.1. We note that minor differences in performance are often still significant, as differences between competing models in §4.1 are often likewise small.

Interestingly, on certain datasets such as Forest Cover, Kick, and Breast Cancer, corrupted inputs do not significantly affect performance. It appears that when NPTs do not find it advantageous to rely

on attention between datapoints during training, they can learn to completely ignore other inputs, essentially collapsing into a standard parametric model. This supports our earlier claims that NPTs can learn end-to-end from data the extent to which they rely on other datapoints for prediction. We think this is extremely interesting behavior and are unaware of prior work reporting similar results. However, we stress that these results reflect inductive biases of the NPT architecture and do not lend themselves to general statements about the performance of parametric versus non-parametric models.

4.4 NPTs Rely on Similar Datapoints for Predictions on Real Data

So far, we have presented convincing evidence that NPTs (sometimes strongly) depend on attention between datapoints. However, we do not know what kind of interactions are learned in practice on real data (**Q4**). As an initial step towards understanding this, we now present two experiments investigating *to which* other datapoints NPT attends.

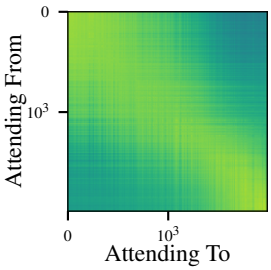


Fig. 4: Attention weights.

Qualitative Evidence. Figure 4 shows an attention map for attention between datapoints (ABD) of NPT on a batch of the Protein regression dataset. We sort the input data with respect to their feature space distance such that similar datapoints are now close to each other. The diagonal pattern in Fig. 4 indicates that NPT attends more strongly to datapoints that are similar in feature space. Appendix B.3.1 discusses this further and gives additional attention maps.

Quantitative Evidence. Seeking a quantitative measure for this hypothesis, the *data deletion* experiment repeats the following procedure for all test set points: iteratively delete other datapoints from the input if they do not significantly affect the prediction. We stop if less than 2% of the original datapoints remain, or if the total change in prediction for the target (relative to the original prediction with all data) exceeds 10%. We investigate the average feature space distances between the test point and the *kept* datapoints, as well as the distances between the test point and the *deleted* datapoints. We find that kept datapoints have a significantly lower average feature space distance to the test point than those deleted. This indicates that two datapoints i, i' that are similar in feature space, such that $\sum_{j<d}(X_{i,j} - X_{i',j})^2$ is low, have a larger effect on the predictions of one another. A Wilcoxon signed-rank test is significant at $p \approx 8.77 \cdot 10^{-130}$. We give full details on this in Appendix B.3.2.

Both experiments support the hypothesis that NPTs rely on similar datapoints for prediction in real data settings. One possible explanation is that similar datapoints might have different realizations of observation noise which NPTs could learn to average out. Altogether, we conclude that NPTs can and do learn representations which rely on interactions between datapoints for prediction.

5 Limitations, Future Work, and Conclusions

Limitations. NPTs share scaling limitations with all naïvely non-parametric approaches [64] and Graph Convolutional Networks [44]. While we have seen success with minibatching (§2.6), NPT justifies future work in principled attention approximations, such as learning representative input points [51], kernelization [18, 40], or other sparsity-inducing methods [5, 17, 72].

Future Work. We believe that the unique predictive mechanism of NPTs makes them an interesting object of study for other tasks including continual learning, multi-task learning, few-shot generalization, and domain adaptation. For example, when predicting under distribution shift, general relations between datapoints and attributes may remain valid and allow NPTs to accommodate such scenarios better. Additionally, future work could explore the connections to stochastic processes, e.g., extending NPTs to be approximately consistent, similar to Neural Processes [32, 33, 42].

Conclusions. We have introduced Non-Parametric Transformers (NPTs), a novel deep learning architecture that takes the entire dataset as input and uses self-attention to model complex relationships *between* datapoints. NPTs challenge and naturally extend parametric modeling as the dominant paradigm of deep learning. They have the additional flexibility to learn to predict by directly attending to other datapoints. Notably, NPTs learn this end-to-end from the data at hand. Empirically, NPTs achieve highly competitive performance on a variety of benchmarks, and additional experiments demonstrate their ability to solve complex reasoning tasks over datapoints. Further, we show that on real data, NPTs learn to rely on attention between datapoints for prediction. We believe that the characteristics of NPTs will make them an exciting object of further study.

References

[1] Naomi S Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46, 1992.

[2] Sercan O Arik and Tomas Pfister. Tabnet: Attentive interpretable tabular learning. *arXiv:1908.07442*, 2019.

[3] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv:1607.06450*, 2016.

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*, 2015.

[5] Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv:2004.05150*, 2020.

[6] Y. Bengio, S. Bengio, and J. Cloutier. Learning a synaptic learning rule. In *International Joint Conference on Neural Networks*, volume 2, 1991.

[7] J. L. Bentley. Multidimensional binary search trees used for associative searching. In *Communications of the ACM*, volume 18, 1975.

[8] John B Biggs. The role of metalearning in study processes. *British journal of educational psychology*, 55, 1985.

[9] Leo Breiman. Bagging predictors. *Machine learning*, 24, 1996.

[10] Leo Breiman. Random forests. *Machine learning*, 45, 2001.

[11] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984.

[12] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv:2005.14165*, 2020.

[13] Thang Bui, Daniel Hernández-Lobato, Jose Hernandez-Lobato, Yingzhen Li, and Richard Turner. Deep gaussian processes for regression using approximate expectation propagation. In *International Conference on Machine Learning*, 2016.

[14] Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. Extracting training data from large language models. *arXiv:2012.07805*, 2020.

[15] Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Mike Schuster, Noam Shazeer, Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Zhifeng Chen, Yonghui Wu, and Macduff Hughes. The best of both worlds: Combining recent advances in neural machine translation. In *Annual Meeting of the Association for Computational Linguistics*, volume 56, 2018.

[16] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Knowledge Discovery and Data Mining*, volume 22, 2016.

[17] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv:1904.10509*, 2019.

[18] Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, David Benjamin Belanger, Lucy J Colwell, and Adrian Weller. Rethinking attention with performers. In *International Conference on Learning Representations*, 2021.

[19] Michael Chui, James Manyika, Mehdi Miremadi, Nicolaus Henke, Rita Chung, Pieter Nel, and Sankalp Malhotra. Notes from the AI frontier: Insights from hundreds of use cases, 2018.

[20] Zhenwen Dai, Andreas Damianou, Javier González, and Neil Lawrence. Variational auto-encoded deep gaussian processes. In *International Conference on Learning Representations*, 2016.

[21] Andreas Damianou and Neil D Lawrence. Deep gaussian processes. In *International Conference on Artificial Intelligence and Statistics*, volume 16, 2013.

[22] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Conference on Computer Vision and Pattern Recognition*, 2009.

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

[24] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[25] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

[26] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, volume 34, 2017.

[27] Evelyn Fix. *Discriminatory analysis: nonparametric discrimination, consistency properties*, volume 1. USAF school of Aviation Medicine, 1985.

[28] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55, 1997.

[29] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, 2001.

[30] Fabian Fuchs, Daniel Worrall, Volker Fischer, and Max Welling. Se(3)-transformers: 3d roto-translation equivariant attention networks. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

[31] Saurabh Garg, Yifan Wu, Sivaraman Balakrishnan, and Zachary Lipton. A unified view of label shift estimation. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

[32] Marta Garnelo, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Rezende, and SM Ali Eslami. Conditional neural processes. In *International Conference on Machine Learning*, volume 35, 2018.

[33] Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J Rezende, SM Eslami, and Yee Whye Teh. Neural processes. *arXiv:1807.01622*, 2018.

[34] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585, 2020.

[35] Jonathan Ho, Nal Kalchbrenner, Dirk Weissenborn, and Tim Salimans. Axial attention in multidimensional transformers. *arXiv:1912.12180*, 2019.

[36] James Honaker and Gary King. What to do about missing values in time series cross-section data. *American Journal of Political Science*, 2010.

[37] Michael Hutchinson, Charline Le Lan, Sheheryar Zaidi, Emilien Dupont, Yee Whye Teh, and Hyunjik Kim. Lietransformer: Equivariant self-attention for lie groups. *arXiv:2012.10885*, 2020.

[38] Google Inc. Kaggle. *https://www.kaggle.com/*, 2021.

[39] Andrew Jaegle, Felix Gimeno, Andrew Brock, Andrew Zisserman, Oriol Vinyals, and Joao Carreira. Perceiver: General perception with iterative attention. *arXiv:2103.03206*, 2021.

[40] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*, volume 37, 2020.

[41] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in neural information processing systems*, volume 30, 2017.

[42] Hyunjik Kim, Andriy Mnih, Jonathan Schwarz, Marta Garnelo, Ali Eslami, Dan Rosenbaum, Oriol Vinyals, and Yee Whye Teh. Attentive neural processes. In *International Conference on Learning Representations*, 2019.

[43] Gary King, James Honaker, Anne Joseph, and Kenneth Scheve. Analyzing incomplete political science data: An alternative algorithm for multiple imputation. *American Political Science Review*, 2001.

[44] Thomas Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

[45] Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. Neural relational inference for interacting systems. In *International Conference on Machine Learning*, volume 35, 2018.

[46] Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big transfer (bit): General visual representation learning. In *European Conference on Computer Vision*, 2020.

[47] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images, 2009.

[48] Brenden M. Lake, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350, 2015.

[49] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86, 1998.

[50] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]*, 2, 2010.

[51] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International Conference on Machine Learning*, volume 36, 2019.

[52] T. Liu, A. Moore, and A. Gray. New algorithms for efficient high-dimensional nonparametric classification. In *Journal of Machine Learning Research*, volume 7, 2006.

[53] Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

[54] Wei-Yin Loh. Fifty years of classification and regression trees. *International Statistical Review*, 82, 2014.

[55] James N Morgan and John A Sonquist. Problems in the analysis of survey data, and a proposal. *Journal of the American statistical association*, 58, 1963.

[56] Sharan Narang, Hyung Won Chung, Yi Tay, William Fedus, Thibault Févry, Michael Matena, Karishma Malkan, Noah Fiedel, Noam Shazeer, Zhenzhong Lan, Yanqi Zhou, Wei Li, Nan Ding, Jake Marcus, Adam Roberts, and Colin Raffel. Do transformer modifications transfer across implementations and applications? *arXiv:2102.11972*, 2021.

[57] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In *International Conference on Machine Learning*, volume 35, 2018.

[58] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, 2019.

[59] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2011.

[60] Google Cloud AI Platform. Getting started with the built-in tabnet algorithm, 2021. URL cloud.google.com/ai-platform/training/docs/algorithms/tab-net-start.

[61] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: unbiased boosting with categorical features. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31, 2018.

[62] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 2019.

[63] Roshan Rao, Jason Liu, Robert Verkuil, Joshua Meier, John F Canny, Pieter Abbeel, Tom Sercu, and Alexander Rives. Msa transformer. *bioRxiv*, 2021.

[64] Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer school on machine learning*, 2003.

[65] Tal Ridnik, Emanuel Ben-Baruch, Asaf Noy, and Lihi Zelnik-Manor. Imagenet-21k pretraining for the masses. *arXiv:2104.10972*, 2021.

[66] Hugh Salimbeni and Marc Deisenroth. Doubly stochastic variational inference for deep gaussian processes. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30, 2017.

[67] Robert E Schapire. The strength of weak learnability. *Machine learning*, 5, 1990.

[68] Weiping Song, Chence Shi, Zhiping Xiao, Zhijian Duan, Yewen Xu, Ming Zhang, and Jian Tang. Autoint: Automatic feature interaction learning via self-attentive neural networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019.

[69] D.J. Stekhoven and P. Buehlmann. Missforest - nonparametric missing value imputation for mixed-type data. *Bioinformatics*, 2012.

[70] Yu-Sung Su, Andrew E. Gelman, Jennifer Hill, and Masanao Yajima. Multiple imputation with diagnostics (mi) in R: Opening windows into the black box. *Journal of Statistical Software*, 2012.

[71] C. Sun, A. Shrivastava, S. Singh, and A. Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017.

[72] Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey. *arXiv:2009.06732*, 2020.

[73] Hugo Touvron, Andrea Vedaldi, Matthijs Douze, and Herve Jegou. Fixing the train-test resolution discrepancy. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[74] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. *arXiv:2012.12877*, 2020.

[75] Stef van Buuren and Karin Groothuis-Oudshoorn. mice: Multivariate imputation by chained equations in r. *Journal of Statistical Software*, 2011.

[76] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[77] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

[78] Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P. Xing. Deep kernel learning. In *International Conference on Artificial Intelligence and Statistics*, volume 19, 2016.

[79] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

[80] Jaesik Yoon, Taesup Kim, Ousmane Dia, Sungwoong Kim, Yoshua Bengio, and Sungjin Ahn. Bayesian model-agnostic meta-learning. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31, 2018.

[81] Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training bert in 76 minutes. In *International Conference on Learning Representations*, 2020.

[82] Michael Zhang, James Lucas, Jimmy Ba, and Geoffrey E Hinton. Lookahead optimizer: k steps forward, 1 step back. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See §5.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] In Appendix F, we give a lengthy discussion of potential negative societal impacts.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] Appendix A gives proof that NPTs are equivariant with respect to a permutation of the datapoints.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Footnote in §4 gives link to anonymized code that should allow for result replication.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Appendix E discusses data details and baseline hyperparameters, and Appendix C.1 gives NPT hyperparameters.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] We perform 10-fold cross-validation whenever computationally feasible.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] This is addressed in Appendix G.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] We cite the creators of the UCI Repository, MNIST, and CIFAR. We also cite some of the most important libraries used in this work in Appendix G.
   (b) Did you mention the license of the assets? [Yes] Also in Appendix G.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] We include a link to code for NPT in a footnote in §4.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] We are using standard machine learning datasets and cite licenses in Appendix G.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] We are using standard machine learning datasets with a long history of academic usage.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A] Does not apply – no crowdsourcing or research including human subjects.
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A] Does not apply – no crowdsourcing or research including human subjects.
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A] Does not apply – no crowdsourcing or research including human subjects.