Vision Transformers learn patch association

Anonymous Author(s)
Affiliation
Address
email

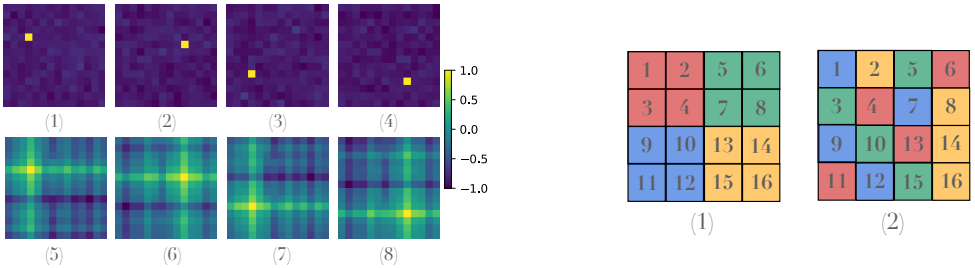
Abstract

Vision Transformers (ViTs) have recently achieved comparable or superior performance to Convolutional neural networks (CNNs) in computer vision. This empirical breakthrough is even more remarkable since ViTs discards spatial information by mixing patch embeddings and positional encodings and do not embed any visual inductive bias (e.g. spatial locality). Yet, recent work showed that while minimizing their training loss, ViTs specifically learn convolution-like patterns. This raises a central question: how do ViTs learn this pattern by solely minimizing their training loss using gradient-based methods from *random initialization*? We propose a structured classification dataset and a simplified ViT model to provide preliminary theoretical justification of this phenomenon. The structure in our dataset is general and includes convolution-like patterns. Our model relies on a simplified attention mechanism –the positional attention mechanism– where the attention matrix solely depends on the positional encodings. While the problem admits multiple solutions that generalize, we show that our model implicitly learns the spatial structure of the dataset while generalizing. We finally prove that learning the structure helps to sample-efficiently transfer to downstream datasets that share the same structure as the pre-training one but with different features. We empirically verify that ViTs using only the positional attention mechanism perform similarly to the original one on CIFAR-10/100, SVHN and ImageNet.

1 Introduction

Transformers are deep learning models built on self-attention [32], and in the past several years they have increasingly formed the backbone for state-of-the-art models in domains ranging from Natural Language Processing (NLP) [32, 14] to computer vision [15], reinforcement learning [8, 22], program synthesis [2] and symbolic tasks [24]. Beyond their remarkable performance, several works reported the ability of Transformers to simultaneously minimize their training loss and learn inductive biases tailored to specific datasets e.g. in computer vision [28], in NLP [6, 34] or in mathematical reasoning [37]. In this paper, we focus on computer vision where convolutions are considered to be an adequate and biologically plausible inductive bias since they capture local spatial information [18]. This seems intuitively reasonable: nearby pixels encode the presence of small scale features, whose patterns in turn determine more abstract features at longer and longer length scales. Several seminal works [9, 15, 28] *empirically* show that although randomly initialized, the positional encodings in Vision Transformers (ViTs) [15] learn structured patterns where closer patches have more similar positional encodings – see Figure 1a-(5,6,7,8). We refer to this structure as "convolution-like" patterns. A priori, learning such specific structure is surprising. Indeed, in contrast to convolutional neural networks (CNNs), ViTs are not built with the inductive bias of spatial locality and weight sharing. They start by replacing an image by a collection of D patches $(\boldsymbol{X}_1, \ldots, \boldsymbol{X}_D) \in \mathbb{R}^{d \times D}$, each of dimension d. While each $\boldsymbol{X}_i$ represents (an embedding of) a spatially localized portion of the original image, the relative positions of the patches $\boldsymbol{X}_i$ in the image are disregarded. Instead, relative spatial information is supplied through *image-independent* positional encodings $\boldsymbol{P} = (\boldsymbol{p}_1, \ldots, \boldsymbol{p}_D) \in \mathbb{R}^{d \times D}$. Unlike

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.



(a) (b)

Figure 1: (a) Visualization of the positional encodings similarities at initialization (1-2-3-4) and after training on Imagenet (5-6-7-8) using a "ViT-large-patch16-224" [15]. In contrast with the initial arrays that are random, the final ones show a convolution-like pattern around the patch in yellow. (b) Partition of the patches into sets $\mathcal{S}_\ell$ as in Definition 2.1. Each square depicts a patch along with its index e.g. "1" denotes patch $\boldsymbol{X}_1$. Squares in the same color belong to the same set $\mathcal{S}_\ell$. (1) corresponds to convolutional sets $\mathcal{S}_\ell = \{4(\ell-1)+1, 4(\ell-1)+2, 4(\ell-1)+3, 4\ell\}$ for $\ell \in \{1,2,3,4\}$. Definition 2.1 also covers sets with non-contiguous elements as (2).

CNNs, each layer of a ViT then learns, via trainable self-attention, a non-local set of filters that non-linearly depend on both the values of all patches $\boldsymbol{X}_j$ and their positional encodings $\boldsymbol{p}_j$. Despite their spatial non-locality, ViTs surprisingly learn the convolution inductive bias in vision datasets as shown in Figure 1a.

This empirical observation sets a central question: from a *theoretical perspective*, how do ViTs manage to learn these convolution patterns by simply minimizing their training loss using gradient descent from random initialization? While it is known that attention can express convolution [9], it remains unclear how ViTs learn it. In this paper, we present a simple spatially-structured classification dataset for which it is sufficient (but not necessary) to learn the structure in order to generalize. We also present a simplified ViT model which we prove implicitly learns a convolution-like inductive bias when it minimizes its training loss via gradient descent (GD). We name this inductive bias *patch association*. We prove that our ViT model leverages this bias to generalize and sample-efficiently transfer to downstream datasets that share the same structure. To our knowledge, our work is the first showing that there exists simple (yet natural) data distributions where simple ViTs implicitly learn convolution-like patterns during training. Our contributions are divided as follows:

- In Section 2, we introduce our framework to formalize patch association.
- In Section 3, we introduce our structured classification dataset and a simplified ViT model to theoretically study how ViTs learn their inductive bias and generalize.
- In Section 4, we present our main theorems (Theorem 4.1 and Theorem 4.2) along with the intermediate lemmas. We prove that a one-layer single-head ViT model trained with GD on our synthetic dataset learns its structure and thus generalizes.
- In Section 5, we show (Theorem 5.1) that after pre-training in our synthetic dataset, our model can be sample-efficiently fine-tuned to transfer to a downstream dataset that shares the same structure as the source dataset (and may have different features).
- In Section 6, we conduct a set of experiments to validate our theoretical model.

Related work

CNNs and ViTs. Many computer vision architectures can be considered as a form of hybridization of Transformers with CNNs. For example, DeTR [7] uses a CNN to generate features that are fed to a Transformer. [16] show that self-attention can be initialized or regularized to behave like a convolution and [11, 19] add convolution operations to Transformers. Conversely, [5, 29, 4] introduce self-attention or attention-like operations to supplement or replace convolution in ResNet-like models. In constrast, our paper attempts to explain how ViTs learn convolution-like patterns using GD.

Theory for attention models. Early theoretical works have focused on the expressivity of attention. [33, 17] addressed this question in the context of self-attention blocks and [12, 35, 21] for Transformers. On the optimization side, [38] investigate the role of adaptive methods in attention models and [31] analyze the dynamics of a single-head attention head to approximate the learning of a Seq2Seq architecture. Our work rather belongs to the implicit bias line of research: we exhibit a setting where a single-head ViT trained with gradient descent provably learns convolution-like patterns and generalizes. To our knowledge, this work is the first that proves generalization of attention models through the lens of optimization.

Notation. We use lower case letters for scalars, lower case bold for vectors and upper case bold for matrices. Given an integer D, we define $[D] = \{1, \dots, D\}$. Any statement made "with high probability" holds with probability at least $1 - 1/\text{poly}(d)$. Given a vector $\boldsymbol{a} \in \mathbb{R}^d$ and $k \leqslant d$, we define $\text{Top}_k\{a_j\}_{j=1}^d = \{a_{i_1}, \dots, a_{i_k}\}$ where $a_{i_1}, \dots, a_{i_k}$ are the k-largest elements. For a function F that implicitly depend on parameters $\boldsymbol{A}$ and $\boldsymbol{v}$, we often write $F_{\boldsymbol{A},\boldsymbol{v}}$ to highlight its parameters.

2 Defining patch association

The goal of this section is to define the notion of patch association. Before, we define the data structure we use to model convolution-like patterns.

Definition 2.1 (Data structure)**.** *Let $\mathcal{D}$ be a distribution over $\mathbb{R}^{d\times D} \times \{-1, 1\}$ where each patch $\boldsymbol{X} = (\boldsymbol{X}_1, \dots, \boldsymbol{X}_D) \in \mathbb{R}^{d\times D}$ has label $y \in \{-1, 1\}$. We partition $[D]$ into L disjoint subsets i.e. $[D] = \bigcup_{\ell=1}^L \mathcal{S}_\ell$ such that $\mathcal{S}_\ell \subsetneq D$ and $|\mathcal{S}_\ell| = C$.*

Note that Definition 2.1 is not an assumption since for any matrix $\boldsymbol{X}$, it is possible to define a partition on its columns. We thus consider distributions $\mathcal{D}$ where there exists an underlying partition of the patches into disjoint sets $\mathcal{S}_\ell$. A particular case of sets $\mathcal{S}_\ell$ is the one of convolutional sets which we use to model convolution-like patterns from Figure 1a. We define them as follows.

Definition 2.2 (Informal: Convolutional sets)**.** *For $\boldsymbol{X} = (\boldsymbol{X}_1, \dots, \boldsymbol{X}_D)$, we say that the sets $\mathcal{S}_\ell$ are convolutional sets if $\forall \ell \in [L]$, $\forall i \in \mathcal{S}_\ell$, the $\boldsymbol{X}_i$'s form a rectangle.*

An example of convolutional sets is $\mathcal{S}_1 = \{1, \dots, C\}, \dots, \mathcal{S}_{\lfloor D/C \rfloor} = \{D - C, \dots, D\}$ that we depict in Figure 1b-(1) in the case $D = 16$ and $C = 4$. We emphasize that Definition 2.1 is not limited to convolutional sets and also covers non-contiguous sets as Figure 1b-(2). Since we aim to show that ViTs learn the sets $\mathcal{S}_\ell$ while fitting the labels y, we need to assume the existence of a labeling function f^* that respects the partition in Definition 2.1.

Assumption 1 (Data structure)**.** *Let $\mathcal{D}$ as in Definition 2.1. We assume that there exists a labeling function f^* such that $\mathbb{P}[yf^*(\boldsymbol{X}) > 0] = 1 - d^{-\omega(1)}$ and,*

$$f^*(\boldsymbol{X}) := \sum_{\ell\in[L]} \phi\big((\boldsymbol{X}_i)_{i\in\mathcal{S}_\ell}\big), \quad \textit{where } \phi\colon \mathbb{R}^{d\times C} \to \mathbb{R} \textit{ is some function,} \tag{1}$$

Assumption 1 states that there exists a labeling function that preserves the underlying structure by applying the same function ϕ to each $\mathcal{S}_\ell$ as in (1). For instance, when the sets $\mathcal{S}_\ell$'s are convolutional, f^* can be a one-hidden layer convolutional network. In this paper, we are interested in *patch association* which refers to the ability of an algorithm to identify the sets $\mathcal{S}_\ell$'s. Following [9, 15, 28], we investigate whether ViTs can learn patch association using its positional encodings.

Definition 2.3 (Patch association for ViTs)**.** *Let Assumption 1 holds. Let $\mathcal{M}\colon \mathbb{R}^{d\times D} \to \{-1, 1\}$ be a transformer and $\boldsymbol{P}^{(\mathcal{M})}$ its positional encodings matrix. We say that $\mathcal{M}$ performs patch association if for all $\ell \in [L]$ and $i \in \mathcal{S}_\ell$, we have $\text{Top}_C\ \{\langle \boldsymbol{p}_i^{(\mathcal{M})}, \boldsymbol{p}_j^{(\mathcal{M})}\rangle\}_{j=1}^D = \mathcal{S}_\ell$.*

Definition 2.3 states that patch association is learned when for a given $i \in \mathcal{S}_\ell$, its positional encoding mainly attends those of j such that $i, j \in \mathcal{S}_\ell$. In this way, the transformer groups the X_i according to $\mathcal{S}_\ell$ just like the true labeling function – It is thus natural to ask would ViTs really learn those $\mathcal{S}_\ell$ after training to match the labeling function f^*?

ViTs do not always learn patch association under Assumption 1. Figure 1a shows that ViTs learn convolution-like patterns on ImageNet. This raises a question: while minimizing their loss, do ViTs always learn the structure of $\mathcal{S}_\ell$ for any data distribution satisfying Assumption 1? We give a negative answer through the following synthetic experiment. Consider the case where all the patches $\boldsymbol{X}_j$ are i.i.d. standard Gaussian and f^* being a one-hidden layer convolutional network with cubic activation. As shown in Figure 2, one-layer ViT reaches small test error. However, $\boldsymbol{P}^\top\boldsymbol{P}$ does not match the convolution pattern encoded in f^*. To prove that ViTs learn patch association, we need additional assumptions on $\mathcal{D}$.

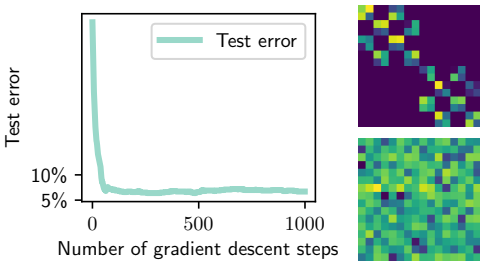


Figure 2: Left: Test error of the ViT on the convolutional structured dataset. Upper Right: convolution structure, non zero coefficients indicates neighbouring patches in the input. Lower Right: Learnt $\boldsymbol{P}^\top\boldsymbol{P}$ looks random compared to upper one.

3 Setting to learn patch association

In this section, we introduce our theoretical setting to analyze how ViTs learn patch association. We first define our binary classification dataset and finally present the ViT model we use to learn it.

Data distribution. The binary classification problem analyzed in this paper is defined as follows.

Definition 3.1 (Data distribution)**.** *Let $\mathcal{D}$ be a distribution satisfying Assumption 1 and $\boldsymbol{w}^* \in \mathbb{R}^d$ be the underlying feature. For each data-point $\boldsymbol{X}$, we define two types of sets:*

- ***Signal set**: uniformly sample an index $\ell(\boldsymbol{X})$ from $[L]$ and for $j \in \mathcal{S}_{\ell(\boldsymbol{X})}$, $\boldsymbol{X}_j = y\boldsymbol{w}^* + \boldsymbol{\xi}_j$, where $y\boldsymbol{w}^*$ is the informative feature and $\boldsymbol{\xi}_j \overset{i.i.d.}{\sim} \mathcal{N}(0, \sigma^2 \mathbf{I}_D)$.*
- ***Random sets**: for $\ell \in [L] \backslash \{\ell(\boldsymbol{X})\}$ and $j \in \mathcal{S}_\ell$, $\boldsymbol{X}_j = \delta_j \boldsymbol{w}^* + \boldsymbol{\xi}_j$, where the spurious feature $\delta_j = 1$ with probability $q/2$, -1 with same probability and 0 otherwise, and $\boldsymbol{\xi}_j \overset{i.i.d.}{\sim} \mathcal{N}(0, \sigma^2 \mathbf{I}_D)$.*

Besides, $\mathcal{D}$ admits the labeling function $f^*(\boldsymbol{X}) = \sum_{\ell \in [L]} \mathrm{Threshold}_{0.9C}(\sum_{i \in \mathcal{S}_\ell} \langle \boldsymbol{w}^*, \boldsymbol{X}_i \rangle)$, where $\mathrm{Threshold}_C(z) = z$ if $|z| > C$ and 0 otherwise.

We sketch a data-point of $\mathcal{D}$ in Figure 3. Our dataset can be viewed as an extreme simplification of real-world object-recognition datasets where there is a set of adjacent patches that contain useful feature (e.g. the nose of a dog) and many patches that have uninformative or spurious features e.g. the background of the image. We make the following assumptions on the parameters of $\mathcal{D}$.

Assumption 2. *We set $d = \mathrm{poly}(D)$, $C = \mathrm{polylog}(d)$, $q = \mathrm{poly}(C)/D$, $\|\boldsymbol{w}^*\|_2 = 1$ and $\sigma^2 = 1/d$. This implies $C \ll D$ and $q \ll 1$.*

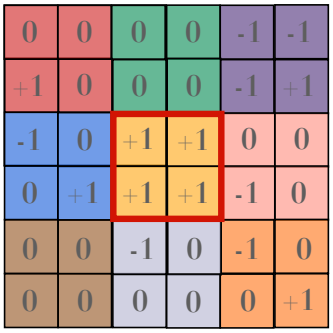


Figure 3: Visualization of a data-point $\boldsymbol{X}$ in $\mathcal{D}$ when the $\mathcal{S}_\ell$'s are convolutional. Each square depicts a patch $\boldsymbol{X}_j$ and squares of the same color belong to the same set $\mathcal{S}_\ell$. "0" indicates that the patch does not have a feature, "1" stands for feature $1 \cdot \boldsymbol{w}^*$ and "-1" for feature $-1 \cdot \boldsymbol{w}^*$. The large red square depicts the signal set $\ell(\boldsymbol{X})$. Although there are more "-1"'s than "+1"'s, the label of $\boldsymbol{X}$ is $+1$ since there are only "+1"'s inside the signal set.

Assumption 2 may be justified by considering a "ViT-base-patch16-224" model [15] on Imagenet. In this case, $d = 384$, $D = 196$. σ is set to have $\|\boldsymbol{\xi}_j\|_2 \approx \|\boldsymbol{w}^*\|_2$. q is chosen so that there are more spurious features than informative ones (low signal-to-noise regime) which makes the data non-linearly separable. Under Assumption 2, our dataset is non-trivial to learn since generalized linear networks fail to generalize.

Theorem 3.1. *Let $\mathcal{D}$ be as defined in Definition 3.1. Let $g(\boldsymbol{X}) = \phi\left(\sum_{j=1}^{D} \langle \boldsymbol{w}_j, \boldsymbol{X}_j \rangle\right)$ be a generalized linear model. Then, g does not fit the labeling function i.e. $\mathbb{P}[f^*(\boldsymbol{X}) g(\boldsymbol{X}) \leqslant 0] \geqslant 1/8$.*

Intuitively, g fails to generalize because it is does not have any knowledge on the underlying partition and the number of random sets is much higher than the signal one. Thus, a model must have a minimal (but not necessarily complete) knowledge about the $\mathcal{S}_\ell$'s in order to generalize.

Theorem 3.2. *Let $\mathcal{D}$ be defined as in Definition 3.1. There exists a (one-layer) transformer $\mathcal{M}$ so that $\mathbb{P}[f^*(\boldsymbol{X}) \mathcal{M}(\boldsymbol{X}) \leqslant 0] = d^{-\omega(1)}$ but for all $\ell \in [L]$, $i \in \mathcal{S}_\ell$, $\mathrm{Top}_C \left\{\langle \boldsymbol{p}_i^{(\mathcal{M})}, \boldsymbol{p}_j^{(\mathcal{M})} \rangle\right\}_{j=1}^{D} \cap \mathcal{S}_\ell = \varnothing$.*

Theorem 3.2 states that there exist classifiers that generalize without being able to learn patch association (see Appendix for proof). In Section 4, we show that there exist a ViT model that learns patch association via minimizing its training objective.

Simplified ViT model. We here define our simplified ViT model. Before, we remind the definition of attention [3, 32] in the single-head case with average pooling.

Definition 3.2 (Self-attention)**.** *Let $\boldsymbol{X} \in \mathbb{R}^{d \times D}$ be a patch from $\mathcal{D}$ and $\boldsymbol{P} \in \mathbb{R}^{d \times D}$ the positional encoding. The attention mechanism computes*

1. *the sum of patches and positional encodings i.e.* $\boldsymbol{\mathcal{X}} = \boldsymbol{X} + \boldsymbol{P} \in \mathbb{R}^{d\times(D+1)}$.
2. *the attention matrix* $\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{K}^\top$ *where* $\boldsymbol{Q} = \boldsymbol{\mathcal{X}}^\top \boldsymbol{W_Q}$, $\boldsymbol{K} = \boldsymbol{\mathcal{X}}^\top \boldsymbol{W_K}$, $\boldsymbol{W_Q}, \boldsymbol{W_K} \in \mathbb{R}^{d\times d}$.
3. *the score matrix* $\boldsymbol{S} \in \mathbb{R}^{D\times D}$ *with coefficients* $S_{i,j} = \exp(A_{i,j}/\sqrt{d})/\sum_{r=1}^{D}\exp(A_{i,r}/\sqrt{d})$.
4. *the matrix* $\boldsymbol{V} = \boldsymbol{\mathcal{X}}^\top \boldsymbol{W_V}$, *where* $\boldsymbol{W_V} \in \mathbb{R}^{d\times d}$.

It finally outputs $\mathrm{SA}((\boldsymbol{X};\boldsymbol{P})) = \boldsymbol{S}\boldsymbol{V} \in \mathbb{R}^{d\times D}$.

Self-attention *mixes* patches and positional encodings which makes it hard to use to show that ViTs learn patch association. We introduce the positional attention mechanism that circumvents this issue.

Definition 3.3 (Positional attention)**.** *Let* $\boldsymbol{X} \in \mathbb{R}^{d\times D}$ *be a datum from* $\mathcal{D}$ *and* $\boldsymbol{P} \in \mathbb{R}^{d\times D}$ *the positional encoding. The positional attention mechanism takes as input the pair* $(\boldsymbol{X};\boldsymbol{P})$ *and computes:*

1. *the attention matrix* $\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{K}^\top$ *where* $\boldsymbol{Q} = \boldsymbol{P}^\top \boldsymbol{W_Q}$, $\boldsymbol{K} = \boldsymbol{P}^\top \boldsymbol{W_K}$ *and* $\boldsymbol{W_Q}, \boldsymbol{W_K} \in \mathbb{R}^{d\times d}$.
2. *the matrix* $\boldsymbol{V} = \boldsymbol{X}^\top \boldsymbol{W_V}$, *where* $\boldsymbol{W_V} \in \mathbb{R}^{d\times d}$.

Similarly to the attention mechanism, it computes the score matrix and outputs $\mathrm{PA}((\boldsymbol{X};\boldsymbol{P})) = \boldsymbol{S}\boldsymbol{V}$.

In Definition 3.3, one of the main differences with self-attention is having $\boldsymbol{A} = \boldsymbol{P}^\top \boldsymbol{W_Q}\boldsymbol{W_K}\boldsymbol{P}$. Thus, the advantage of positional attention is to *isolate* the positional encoding from the patches. Using this framework, we can thus use $\boldsymbol{A}$ to monitor the learning of patch association. On the other hand, the matrix $\boldsymbol{V}$ encodes the data-dependent part. In summary, a model that fits the labeling function and learns patch association in our setting must have i) $\boldsymbol{A}$ learns patch association ii) $\boldsymbol{V}$ learns $\boldsymbol{w}^*$ in order to generalize. Lastly, we make the following simplification in the parameters to ease our analysis.

Simplification 3.1. *In the positional attention mechanism, we set* $d = D$, $\boldsymbol{W_K} = \boldsymbol{I}_D$ *and* $\boldsymbol{W_Q} = \boldsymbol{I}_D$ *which implies* $\boldsymbol{A} = \boldsymbol{P}^\top \boldsymbol{P}$. *Besides, we set* $\boldsymbol{W_V} = [\boldsymbol{v}, \ldots, \boldsymbol{v}] \in \mathbb{R}^{D\times D}$ *where* $\boldsymbol{v} \in \mathbb{R}^D$.

In Assumption 3.1, we set $\boldsymbol{W_K}$ and $\boldsymbol{W_Q}$ so that $\boldsymbol{A} = \boldsymbol{P}^\top \boldsymbol{P}$. This Gram matrix encodes the inductive bias of the ViT as previously seen in Figure 1a. Besides, since fitting the labeling function requires to learn one feature $\boldsymbol{w}^*$, it is sufficient to parametrize $\boldsymbol{W_V}$ with a vector $\boldsymbol{v}$. Under Assumption 3.1, our simplified ViT model is thus one-hidden attention layer with a single head defined by

$$F(\boldsymbol{X}) = \sum_{i=1}^{D} \sigma\left(D \sum_{j=1}^{D} S_{i,j} \langle \boldsymbol{v}, \boldsymbol{X}_j \rangle\right), \tag{T}$$

where σ is an activation function. Since we aim to the simplest ViT model, we opt for a polynomial activation i.e. $\sigma(x) = x^p + \nu x$ where $p \geqslant 3$ is an odd integer and $\nu = 1/\mathrm{poly}(d)$. Note that this choice of polynomial activation is common in the deep learning theory literature – see e.g. [25, 1, 36] among others. The degree p is odd to make the ViT model compatible with the labeling function and strictly larger than 1 because the data is not linearly separable (Assumption 2). We add a linear part in the activation function to ensure that the gradient is non-zero when $\boldsymbol{v}$ has tiny coefficients.

With these simplifications, we formally prove that F is able to learn patch association and generalize. We underline that this very simplified model is *not meant to be representative* of real-world ViTs but rather to exhibit a setting where a simplified ViT model can provably learn an inductive bias. We believe that positional attention reflects some aspects of real-world ViTs since in Section 6, it numerically competes with self-attention.

Trainable parameters. The trainable parameters of our model (T) are specified as follows.

Assumption 3. *In the model* (T)*, we set* $\boldsymbol{A}$ *and* $\boldsymbol{v}$ *as trainable parameters. Besides, without loss of generality, we train all* $A_{i,j}$ *for* $i \neq j$ *and leave the diagonals of* $\boldsymbol{A}$ *fixed.*

Although $\boldsymbol{A} = \boldsymbol{P}^\top \boldsymbol{P}$ and $\boldsymbol{P}$ is trainable, we choose for simplicity to only optimize over $\boldsymbol{A}$. Besides, we leave the $A_{i,i}$'s fixed because the softmax is invariant under the uniform shift of the input.

Learning problems. Given a dataset $\mathcal{Z} = \{(\boldsymbol{X}[i], y[i])\}_{i=1}^{N}$ sampled from $\mathcal{D}$, we solve the empirical risk minimization problem for the logistic loss defined by:

$$\min_{\widehat{\boldsymbol{A}}, \widehat{\boldsymbol{v}}} \ \frac{1}{N} \sum_{i=1}^{N} \log\left(1 + e^{-y[i]F(\boldsymbol{X}[i])}\right) := \widehat{\mathcal{L}}(\widehat{\boldsymbol{A}}, \widehat{\boldsymbol{v}}). \tag{E}$$

Instead of directly analyzing (E), we introduce a proxy where we minimize the population risk

$$\min_{\boldsymbol{A},\boldsymbol{v}} \ \mathbb{E}_{\mathcal{D}}\big[\log\big(1+e^{-yF(\boldsymbol{X})}\big)\big] := \mathcal{L}(\boldsymbol{A},\boldsymbol{v}). \tag{P}$$

We refer to (E) as the *realistic* problem while (P) as the *idealized* problem.

Algorithm. We solve (P) and (E) using gradient descent (GD) for T iterations. The update rule in the case of (P) for $t \in [T]$ and $i, j \in [D]$ is

$$A_{i,j}^{(t+1)} = A_{i,j}^{(t)} - \eta\partial_{A_{i,j}}\mathcal{L}(\boldsymbol{A}^{(t)},\boldsymbol{v}^{(t)}), \quad \boldsymbol{v}^{(t+1)} = \boldsymbol{v}^{(t)} - \eta\nabla_{\boldsymbol{v}}\mathcal{L}(\boldsymbol{A}^{(t)},\boldsymbol{v}^{(t)}), \tag{GD}$$

where $\eta > 0$ is the learning rate. A similar update may be written for (E). We now detail how to set the parameters in (GD).

Parametrization 3.1. *When running GD on (P) and (E), the number of iterations is any $T \geqslant \mathrm{poly}(d)/\eta$. We set the learning rate as $\eta \in \big(0, \frac{1}{\mathrm{poly}(d)}\big)$. The diagonal coefficient of the attention matrix are set for $i \in [D]$ as $A_{i,i} = \widehat{A}_{i,i} = \sigma_{\boldsymbol{A}}\mathbf{I}_D$ where $\sigma_{\boldsymbol{A}} = \mathrm{polyloglog}(d)$. The off-diagonal coefficients of $\boldsymbol{A}$ and the value vector are initialized as:*

1. *Idealized case: $\boldsymbol{v}^{(0)} = \alpha^{(0)}\boldsymbol{w}^*$ where $\alpha^{(0)} = \nu^{1/(p-1)}$ and $A_{i,j}^{(0)} = 0$ for $i \neq j$.*
2. *Realistic case: $\widehat{\boldsymbol{v}}^{(0)} \sim \mathcal{N}(0, \omega^2\mathbf{I}_d)$ and $\widehat{A}_{i,j}^{(0)} \sim \mathcal{N}(0,\omega^2)$ where $i \neq j$ and $\omega = 1/\mathrm{poly}(d)$.*

In the idealized case, we initialize $\boldsymbol{v}^{(0)}$ in $\mathrm{span}(\boldsymbol{w}^*)$ even though this latter should be unknown to the algorithm. We remind that the idealized case is only an easy-to-study proxy to ultimately characterize the realistic dynamics.

4 Learning patch association via matching the labeling function

As announced above, we show that our ViTs (T) implicitly learns patch association and fits the labeling function by minimizing the training objective. We first study the dynamics in (P). Using the analysis in the idealized case, we then characterize the solution found in the realistic problem (E).

4.1 Learning process in the idealized case

In this section, we analyze dynamics of (P). Our main result is that the final iterates of (P) learn patch association while generalize using GD on infinitely many examples sampled from $\mathcal{D}$.

Theorem 4.1. *Assume that we run GD on (P) for T iterations with parameters set as in Parametrization 3.1. With high probability, the ViT model (T)*

1. *learns patch association i.e. for all $\ell \in [L]$ and $i \in \mathcal{S}_\ell$, $\mathrm{Top}_C\ \{A_{i,j}^{(T)}\}_{j=1}^D = \mathcal{S}_\ell$.*
2. *learns the labeling function f^* i.e. $\mathbb{P}_{\mathcal{D}}[f^*(\boldsymbol{X})F_{\boldsymbol{A}^{(T)},\boldsymbol{v}^{(T)}}(\boldsymbol{X}) > 0] \geqslant 1 - o(1)$.*

We now sketch the main ideas to prove the theorem and refer to Appendix D for a complete proof.

Invariance and symmetries. The main feature making (P) appealing is the fact that we take the expectation over $\mathcal{D}$. Indeed, since (T) is permutation-invariant and the data distribution is symmetric, we can thus dramatically simplify the variables in (P). An illustration of this is the next lemma that shows that $\boldsymbol{A}$ can be reduced to three variables in (P).

Lemma 4.1. *There exist $\beta = \sigma_{\boldsymbol{A}}$, $\gamma^{(t)}, \rho^{(t)} \in \mathbb{R}$ such that for all $t \geqslant 0$:*

1. *for all $i \in [D]$, $A_{i,i}^{(t)} = \beta$.*
2. *for all $i, j \in [D]$ such that $i, j \in \mathcal{S}_\ell$ for some $\ell \in [L]$, $A_{i,j}^{(t)} = \gamma^{(t)}$.*
3. *for all $i, j \in [D]$ such that $i \in \mathcal{S}_\ell$ and $j \in \mathcal{S}_m$ for some $\ell, m \in [L]$ with $\ell \neq m$, $A_{i,j}^{(t)} = \rho^{(t)}$.*

Besides, using the initialization in Parametrization 3.1, we can show that $\boldsymbol{v}$ always lies in $\mathrm{span}(\boldsymbol{w}^*)$.

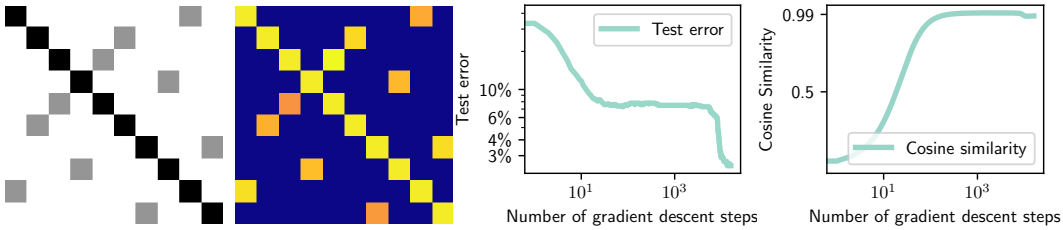


Figure 4: Illustration of Theorem 4.2. From left to right, we first show in grey the tuples (i, j) such that $(i, j) \in \mathcal{S}_\ell$. We then plot the learned matrix $\boldsymbol{A}$ and see that coefficients with high value exactly correspond to their grey scale counterpart in the left plot. We also show test error and cosine similarity of $\boldsymbol{w}^*$ and $\boldsymbol{v}$ w.r.t the number of training steps.

Lemma 4.2. *For all $t \in [T]$, there exists $\alpha^{(t)} \in \mathbb{R}$ such that $\boldsymbol{v}^{(t)} = \alpha^{(t)} \boldsymbol{w}^*$.*

In summary, Lemma 4.1 and Lemma 4.2 imply that instead of optimizing over $\boldsymbol{A}$ and $\boldsymbol{v}$ in (P), we can instead consider the scalar variables $\alpha^{(t)}$, $\gamma^{(t)}$ and $\rho^{(t)}$. The remaining of this section consists in analyzing the dynamics of these three quantities.

Learning patch association. We first analyze the dynamics of $\gamma^{(t)}$ and $\rho^{(t)}$. To this end, we introduce the following softmax terms:

$$\Lambda^{(t)} = \frac{e^{\beta}}{e^{\beta} + (C-1)e^{\gamma^{(t)}} + (D-C)e^{\rho^{(t)}}}, \qquad \Gamma^{(t)} = \frac{e^{\gamma^{(t)}}}{e^{\beta} + (C-1)e^{\gamma^{(t)}} + (D-C)e^{\rho^{(t)}}},$$

$$\Xi^{(t)} = \frac{e^{\rho^{(t)}}}{e^{\beta} + (C-1)e^{\gamma^{(t)}} + (D-C)e^{\rho^{(t)}}}, \qquad G^{(t)} = D(\Lambda^{(t)} + (C-1)\Gamma^{(t)}).$$

Note that $\Lambda^{(t)}$, $\Gamma^{(t)}$ and $\Xi^{(t)}$ respectively correspond to the coefficients on the diagonal, those for which $i, j \in \mathcal{S}_\ell$ for some ℓ and all the other coefficients of the attention matrix $\boldsymbol{S}$. Using these notations, we first derive the GD updates of $\gamma^{(t)}$ and $\rho^{(t)}$.

Lemma 4.3. *Let $t \leqslant T$. The attention weights $\gamma^{(t)}$ and $\rho^{(t)}$ satisfy:*

$$\gamma^{(t+1)} = \gamma^{(t)} + \eta \mathrm{polylog}(d)(\alpha^{(t)})^p \cdot \Gamma^{(t)}(G^{(t)})^{p-1},$$
$$|\rho^{(t+1)}| \leqslant |\rho^{(t)}| + \eta \mathrm{polylog}(d)(\alpha^{(t)})^p \Big(\frac{1}{D} + \frac{1}{D}\Gamma^{(t)}(G^{(t)})^{p-1}\Big).$$

Lemma 4.3 shows that the increment of $\gamma^{(t)}$ is larger than the one of $\rho^{(t)}$. Since $\gamma^{(0)} = \rho^{(0)} = 0$, this implies that $\gamma^{(t)} \geqslant \rho^{(t)}$ for all $t \geqslant 0$. This observation proves the first item of Theorem 4.1. We now show how learning patch association leads to $\boldsymbol{v}$ highly correlated with $\boldsymbol{w}^*$.

Learning process. We now analyze the GD dynamics of $\alpha^{(t)}$. We start by writing $\langle \nabla_{\boldsymbol{v}} \mathcal{L}(\boldsymbol{A}^{(t)}, \boldsymbol{v}^{(t)}), \boldsymbol{w}^* \rangle = \boldsymbol{\mathcal{S}}^{(t)} + \boldsymbol{\mathcal{N}}^{(t)}$ for some $\boldsymbol{\mathcal{S}}^{(t)}, \boldsymbol{\mathcal{N}}^{(t)}$ such that $\boldsymbol{\mathcal{S}}^{(t)} \geqslant 0$ and $\boldsymbol{\mathcal{N}}^{(t)} \in \mathbb{R}$. We divide the idealized learning process as follows.

- **Event I** ($t \in [0, \mathcal{T}_0]$): Since (P) involves the logistic loss, a sigmoid term appears in the gradient. Around the small $\alpha^{(0)}$, this sigmoid term stays constant. This implies that $\boldsymbol{\mathcal{N}}^{(t)} = 0$ which yields $\alpha^{(t)}$ to increase until reaching a specific value where the sigmoid is not constant anymore.
- **Event II** ($t \in [\mathcal{T}_0, \mathcal{T}_1]$): at time $\mathcal{T}_0$, $\alpha^{(t)}$ is large. This fact along with Lemma 4.3 imply that $\gamma^{(t)}$ increases. Eventually, $\Gamma^{(\mathcal{T}_1)}$ becomes large enough so that $\boldsymbol{\mathcal{S}}^{(t)} \geqslant \max_{\tau \leqslant T} |\boldsymbol{\mathcal{N}}^{(\tau)}|$.
- **Event III** ($t \in [\mathcal{T}_1, T]$): Since $\boldsymbol{\mathcal{S}}^{(t)} \geqslant \max_{\tau \leqslant T} |\boldsymbol{\mathcal{N}}^{(\tau)}|$, $\alpha^{(t)}$ increases until being large enough so that the population risk is at most $o(1)$.

This process leads to $\boldsymbol{v}^{(T)} = \alpha^{(T)} \boldsymbol{w}^*$ with $\alpha^{(T)} \gg 1$ and thus to the second item of Theorem 4.1.

4.2 From the idealized to the realistic learning process

The real learning process differs from the idealized one in that i) we have a finite number of samples ii) we initialize both $\widehat{\boldsymbol{A}}$ and $\widehat{\boldsymbol{v}}$ as Gaussian random variables. Using a polynomial number of samples, we show that (T) is still able to learn patch association and generalize

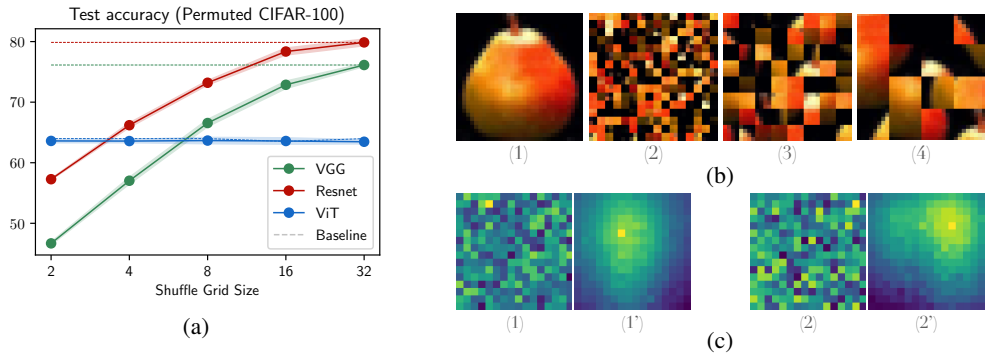


Figure 5: (a): Test accuracy obtained with ViT (patch size 2), ResNet-18 and VGG-19 on permuted (in solid lines) and on original (in dashed lines) CIFAR-100. While convolutional models are very sensitive to permutations, the ViT performs equally whether the dataset is permuted or not. (b): (2) CIFAR-100 image (1) and Permuted CIFAR-100 image when shuffle grid size is 2 (2), 4 (3) and 8 (4). (c): (1-2) Visualization of positional encoding similarities after training a ViT (patch size 2) on permuted CIFAR-100 (shuffle grid size 2). These latter do not have a convolution-like structure. However, when applying the inverse of the permutation, we recover a convolution-like pattern in (1'-2').

Theorem 4.2. *Assume that we run GD on (E) for T iterations with parameters set as in Parametrization 3.1. Assume that the number of samples is $N = \mathrm{poly}(d)$. With high probability, the model*

1. *learns patch association i.e. for all $\ell \in [L]$ and $i \in \mathcal{S}_\ell$, $\mathrm{Top}_C\ \{\widehat{A}_{i,j}^{(T)}\}_{j=1}^{D} = \mathcal{S}_\ell$.*
2. *fits the labeling function i.e. $\mathbb{P}_{\mathcal{D}}[f^*(\boldsymbol{X})F_{\widehat{\boldsymbol{A}}^{(T)},\widehat{\boldsymbol{v}}^{(T)}}(\boldsymbol{X}) > 0] \geqslant 1 - o(1)$.*

Figure 4 numerically illustrates Theorem 4.2. Similarly to [26], the proof introduces a "semi-realistic" learning process that is a mid-point between the idealized and realistic processes. We show that $\widehat{\boldsymbol{A}}^{(T)}$ and $\widehat{\boldsymbol{v}}^{(T)}$ are close to their semi-realistic counterparts – see Appendix E for a complete proof.

5 Patch association yields sample-efficient fine-tuning with ViTs

A fundamental byproduct of our theory is that after pre-training on a dataset sampled from $\mathcal{D}$, our model (T) sample-efficiently transfers to datasets that are structured as $\mathcal{D}$ but differ in their features.

Downstream dataset. Let $\widetilde{\mathcal{D}}$ a downstream data distribution defined as in Definition 3.1 such that its underlying feature is $\widetilde{\boldsymbol{w}}^*$ with $\|\widetilde{\boldsymbol{w}}^*\|_2 = 1$ and $\widetilde{\boldsymbol{w}}^*$ potentially different from $\boldsymbol{w}^*$. In other words, the downstream $\widetilde{\mathcal{D}}$ and source $\mathcal{D}$ distributions share the same structure but not necessarily the same feature. We sample a downstream dataset $\widetilde{\mathcal{Z}} = \{(\widetilde{\boldsymbol{X}}[i], \widetilde{y}[i])\}_{i=1}^{\widetilde{N}}$ from $\widetilde{\mathcal{D}}$.

Learning problem. We consider the model (T) pre-trained as in subsection 4.2. We assume that $\widehat{\boldsymbol{A}}$ is kept *fixed* from the pre-trained model and we only optimize the value vector $\widetilde{\boldsymbol{v}}$ to solve:

$$\min_{\widetilde{\boldsymbol{v}}} \ \frac{1}{\widetilde{N}} \sum_{i=1}^{\widetilde{N}} \log\left(1 + e^{-\widetilde{y}[i]F(\widetilde{\boldsymbol{X}}[i])}\right) := \widetilde{\mathcal{L}}(\widetilde{\boldsymbol{v}}). \tag{$\widetilde{\text{E}}$}$$

We run GD on ($\widetilde{\text{E}}$) with parameters set as in Parametrization 3.1 except that the $\widehat{A}_{i,j}$'s are fixed and $\widetilde{\boldsymbol{v}}^{(0)} \sim \mathcal{N}(0, \omega^2 \mathbf{I}_d)$ with $\omega = 1/\mathrm{poly}(d)$. Our main results states that this fine-tuning procedure requires a few samples to achieve high test accuracy in $\widetilde{\mathcal{D}}$. In contrast, any algorithm without patch association needs a large number of samples to generalize.

Theorem 5.1. *Let $\widehat{\boldsymbol{A}}$ be the attention matrix obtained after pre-training as in subsection 4.2. Assume that we run GD for T iterations on ($\widetilde{\text{E}}$) to fine-tune the value vector. Using $\widetilde{N} \leqslant \mathrm{polylog}(D)$ samples, the model (T) transfers to $\widetilde{\mathcal{D}}$ i.e. $\mathbb{P}_{\widetilde{\mathcal{D}}}[f^*(\boldsymbol{X})F_{\widehat{\boldsymbol{A}},\widetilde{\boldsymbol{v}}^{(T)}}(\boldsymbol{X}) > 0] \geqslant 1 - o(1)$.*

Theorem 5.2. *Let $\mathcal{A}\colon \mathbb{R}^{d\times D} \to \{-1, 1\}$ be a binary classification algorithm without patch association knowledge. Then, it needs at least training $\widetilde{N} \geqslant D^{\Omega(1)}$ samples to generalize on $\widetilde{\mathcal{D}}$.*

The proofs of Theorem 5.1 and Theorem 5.2 are in Appendix F. These theorems hightlight that learning patch association is required for efficient transfer. We believe that they offer a new perspective on explaining why ViTs are widely used in transferring to downstream tasks. While it is possible that ViTs learn shared (with the downstream dataset) features during pretraining, our theory hints that learning the inductive bias of the labeling function is also central for transfer.

6 Numerical experiments

In this section, we empirically verify that i) ViTs learn patch association while miniziming its training loss and ii) the positional attention mechanism competes with the vanilla one on small-scale datasets such as CIFAR-10/100 [23], SVHN [27] and large-scale ones such as ILSVRC-2012 ImageNet [13]. For the small datasets, we use a ViT with 7 layers, 12 heads and hidden/MLP dimension 384. For ImageNet, we train a "ViT-tiny-patch16-224" [15]. Both models are trained with standard augmentations techniques [10] and using AdamW with a cosine learning rate scheduler. We run all the experiments for 300 epochs, with batch size 1024 for Imagenet and 128 otherwise and average our results over 5 seeds. We refer to Appendix A for the training details.

ViTs learn patch association. We consider the CIFAR-100 dataset where we divide each image into grids of size $s \times s$ pixels. For a fixed $s \in \{2, 4, 8, 16, 32\}$, we permute the grids according to π_s to create the permuted CIFAR-100 dataset. We call s the grid shuffle size. Figure 5b-(1) shows a CIFAR-100 image and its corresponding shuffling in the permuted CIFAR-100 dataset Figure 5b-(2-3-4). We train a ViT and convolutional models Resnet18 [20] and VGG-19 [30] on the permuted CIFAR-100 dataset. For the ViT, we set the patch size to 2, although this is sub-optimal in terms of accuracy, because the patch size needs to stay smaller or equal to s. Figure 5a reports the test accuracy of these three models for different values of s. When s is small, the image does not have a coherent structure e.g. Figure 5b-(2) and thus, convolutional models struggle to generalize. As s increases e.g. Figure 5b-(4), the information inside a patch is meaningful and thus, the convolutional models well-perform. In contrast, ViTs are *insensitive to permutations* as their performance remains unchanged for all s – see Figure 5a. We believe that this is due to the ability of ViTs to learn patch association. Indeed, intuitively, when we permute the grids in Figure 5b, we lose the convolution-like structure and create new sets $\mathcal{S}_\ell$'s and a new labeling function f^*. Despite this change, we verify that the ViT is able to recover the new $\mathcal{S}_\ell$'s: we feed the ViT with the shuffled pear image (Figure 5b-(2)) and see that $\boldsymbol{P}^\top \boldsymbol{P}$ does not have a convolution-like structure in Figure 5c-(1,2). We then apply π_s^{-1} to $\boldsymbol{P}^\top \boldsymbol{P}$ and observe that we recover the convolution-like patterns Figure 5c-(1',2'). This experiment highlights that ViTs do not just group nearby pixels together. They learn patch association even when the $\mathcal{S}_\ell$'s are non-convolutional.

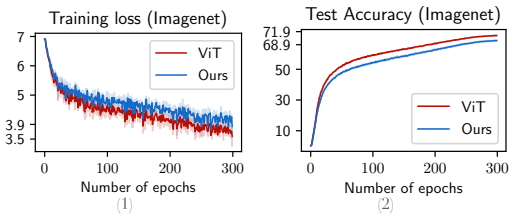


Figure 6: Training loss (1) and test accuracy (2) obtained using a ViT-tiny-patch16-224 on Imagenet. ViT using positional attention (Ours) gets 68.9% test accuracy while vanilla ViT (ViT) gets 71.9%.

ViTs with positional attention are competitive. We numerically verify that ViTs using positional attention compete with those with vanilla attention. In Section 3, we introduced positional attention to define our theoretical learner model. Figure 6 and Figure 7 show that ViTs using positional attention compete with vanilla ViTs on a range of datasets. These experiments strengthen our intuition that for images, having an attention matrix that only depends on the positional encodings is sufficient to have a good test accuracy.

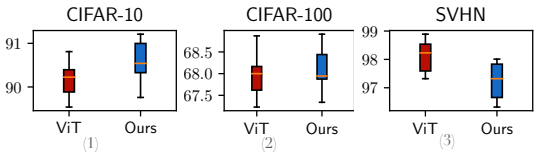


Figure 7: Test accuracy obtained with a ViT using vanilla attention (ViT) and positional attention (Ours) on CIFAR-10 (1), CIFAR-100 (2) and SVHN (3). Our model competes with the vanilla ViT. Patch size 4 and average over 10 seeds for this experiment.

Conclusion, limitations and future works

Our work is a first step towards understanding how Transformers learn tailored inductive biases when trained on large datasets with gradient descent. Our analysis heavily relies on the positional attention mechanism that disentangles patches and positional encodings. In practice, self-attention mixes these two quantities. An interesting direction is to understand the impact of patch embeddings on the inductive bias learned by ViTs. Moreover, our experiment on the Gaussian data shows that ViTs do not always learn the correct inductive bias under Assumption 1: characterizing the distributions under which ViTs recover the structure of the function is an important question. Lastly, this work also paves the way to many extensions beyond convolution. For example, can ViTs learn other inductive bias? What are the inductive biases learnt by Transformers in NLP? Answering those questions is central to better understand the underlying mechanism of attention.

References

[1] Zeyuan Allen-Zhu and Yuanzhi Li. Towards understanding ensemble, knowledge distillation and self-distillation in deep learning. *arXiv preprint arXiv:2012.09816*, 2020.

[2] Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[4] Irwan Bello. Lambdanetworks: Modeling long-range interactions without attention. *arXiv preprint arXiv:2102.08602*, 2021.

[5] Irwan Bello, Barret Zoph, Ashish Vaswani, Jonathon Shlens, and Quoc V Le. Attention augmented convolutional networks. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3286–3295, 2019.

[6] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[7] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020.

[8] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34, 2021.

[9] Jean-Baptiste Cordonnier, Andreas Loukas, and Martin Jaggi. On the relationship between self-attention and convolutional layers. *arXiv preprint arXiv:1911.03584*, 2019.

[10] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2018.

[11] Zihang Dai, Hanxiao Liu, Quoc V Le, and Mingxing Tan. Coatnet: Marrying convolution and attention for all data sizes. *Advances in Neural Information Processing Systems*, 34:3965–3977, 2021.

[12] Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. *arXiv preprint arXiv:1807.03819*, 2018.

[13] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[14] 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.

[15] 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.

[16] Stéphane d'Ascoli, Hugo Touvron, Matthew L Leavitt, Ari S Morcos, Giulio Biroli, and Levent Sagun. Convit: Improving vision transformers with soft convolutional inductive biases. In *International Conference on Machine Learning*, pages 2286–2296. PMLR, 2021.

[17] Benjamin L Edelman, Surbhi Goel, Sham Kakade, and Cyril Zhang. Inductive biases and variable creation in self-attention mechanisms. *arXiv preprint arXiv:2110.10090*, 2021.

[18] Kunihiko Fukushima. Neocognitron for handwritten digit recognition. *Neurocomputing*, 51:161–180, 2003.

[19] Jianyuan Guo, Kai Han, Han Wu, Chang Xu, Yehui Tang, Chunjing Xu, and Yunhe Wang. Cmt: Convolutional neural networks meet vision transformers. *arXiv preprint arXiv:2107.06263*, 2021.

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

[21] Jiri Hron, Yasaman Bahri, Jascha Sohl-Dickstein, and Roman Novak. Infinite attention: Nngp and ntk for deep attention networks. In *International Conference on Machine Learning*, pages 4376–4386. PMLR, 2020.

[22] Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34, 2021.

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

[24] Guillaume Lample and François Charton. Deep learning for symbolic mathematics. *arXiv preprint arXiv:1912.01412*, 2019.

[25] Yuanzhi Li, Tengyu Ma, and Hongyang Zhang. Algorithmic regularization in over-parameterized matrix sensing and neural networks with quadratic activations. In *Conference On Learning Theory*, pages 2–47. PMLR, 2018.

[26] Yuanzhi Li, Tengyu Ma, and Hongyang R Zhang. Learning over-parametrized two-layer neural networks beyond ntk. In *Conference on learning theory*, pages 2613–2682. PMLR, 2020.

[27] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

[28] Maithra Raghu, Thomas Unterthiner, Simon Kornblith, Chiyuan Zhang, and Alexey Dosovitskiy. Do vision transformers see like convolutional neural networks? *Advances in Neural Information Processing Systems*, 34, 2021.

[29] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jon Shlens. Stand-alone self-attention in vision models. *Advances in Neural Information Processing Systems*, 32, 2019.

[30] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[31] Charlie Snell, Ruiqi Zhong, Dan Klein, and Jacob Steinhardt. Approximating how single head attention learns. *arXiv preprint arXiv:2103.07601*, 2021.

[32] 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.

[33] James Vuckovic, Aristide Baratin, and Remi Tachet des Combes. A mathematical theory of attention. *arXiv preprint arXiv:2007.02876*, 2020.

[34] Alex Warstadt and Samuel R Bowman. Can neural networks acquire a structural bias from raw linguistic data? *arXiv preprint arXiv:2007.06761*, 2020.

[35] Colin Wei, Yining Chen, and Tengyu Ma. Statistically meaningful approximation: a case study on approximating turing machines with transformers. *arXiv preprint arXiv:2107.13163*, 2021.

[36] Blake Woodworth, Suriya Gunasekar, Jason D Lee, Edward Moroshko, Pedro Savarese, Itay Golan, Daniel Soudry, and Nathan Srebro. Kernel and rich regimes in overparametrized models. In *Conference on Learning Theory*, pages 3635–3673. PMLR, 2020.

[37] Yuhuai Wu, Markus N Rabe, Wenda Li, Jimmy Ba, Roger B Grosse, and Christian Szegedy. Lime: Learning inductive bias for primitives of mathematical reasoning. In *International Conference on Machine Learning*, pages 11251–11262. PMLR, 2021.

[38] Jingzhao Zhang, Sai Praneeth Karimireddy, Andreas Veit, Seungyeon Kim, Sashank Reddi, Sanjiv Kumar, and Suvrit Sra. Why are adaptive methods good for attention models? *Advances in Neural Information Processing Systems*, 33:15383–15393, 2020.

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] See Section 4 and Section 5.
   (b) Did you describe the limitations of your work? [Yes] See Section 6.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] This is a theory paper.
   (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] See Section 3.
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix.
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] See supplementary material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Section 6
   (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] See Appendix.
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]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
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]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]