Sketching based Representations for Robust Image Classification with Provable Guarantees

Nishanth Dikkala[*]
Google Research
nishanthd@google.com

Sankeerth Rao Karingula[*]
Google Research
sankeerth1729@gmail.com

Raghu Meka[*]
UC Los Angeles
raghuvardhan@gmail.com

Jelani Nelson[*]
UC Berkeley
minilek@gmail.com

Rina Panigrahy[*]
Google Research
rinap@google.com

Xin Wang[*]
Google Research
wanxin@google.com

Abstract

How do we provably represent images succinctly so that their essential latent attributes are correctly captured by the representation to as high level of detail as possible? While today's deep networks (such as CNNs) produce image embeddings they do not have any provable properties and seem to work in mysterious non-interpretable ways. In this work we theoretically study synthetic images that are composed of a union or intersection of several mathematically specified shapes using thresholded polynomial functions (for e.g. ellipses, rectangles). We show how to produce a succinct sketch of such an image so that the sketch "smoothly" maps to the latent-coefficients producing the different shapes in the image. We prove several important properties such as: easy reconstruction of the image from the sketch, similarity preservation (similar shapes produce similar sketches), being able to index sketches so that other similar images and parts of other images can be retrieved, being able to store the sketches into a dictionary of concepts and shapes so parts of the same or different images that refer to the same shape can point to the same entry in this dictionary of common shape attributes.

1 Introduction

A first step in many computer vision systems is to learn a succinct representation of an input image (or a set of images) which retains all the *useful* information about the image. Today, there are several deep networks that can take images and produce useful embedding representations [4, 7, 2]. Deep approaches are extremely complex and hard to understand. Moreover, for the tasks they aim to solve, it is unclear if we need such a non-interpretable ([6, 1]) approach which is also known to be very brittle ([17]). Previously hand-crafted feature construction approaches such as SIFT [13] used to dominate. These were also heuristic and not theoretically guaranteed to work. In addition, they focused heavily on local features which might not be good at understanding important global information such as shape.

Ideally we would like representations of images to be such that they capture the essential latent semantic attributes of the image. If the images are produced by some generative process then the latent attributes correspond to parameters used in the generative process. Further, given a large collection of images, it can be desirable to create a dictionary or an index of commonly occurring latent attributes so that each image may be viewed as a set of pointers into this global dictionary. To study this problem theoretically, we use a simple generative process for producing images using

[*]Alphabetical author order.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

polynomial functions. By taking thresholded versions of polynomial functions one can generate simple shapes such as circles, ellipses, hyperbolas and by taking intersections of such functions one can generate polygons such as rectangle, triangles etc. By taking a union of such intersections one can generate fairly complex images – in fact, much of computer graphics is generated using such principles.

Since this process is fully formalized, it is now feasible to study questions such as: *Do deep networks learn representations that are isomorphic to the parameters that generated the image?* While today's deep networks do not provably have such properties about being able to recover the polynomial coefficients used in an image, we provide a method that can produce a sketch that is provably equivalent to the parameters that generated the image. While our sketch is not identical to the polynomial coefficients, we show that it is equivalent in a smooth manner – that is there exist smooth functions mapping one to the other and vice versa. Further our sketches can be easily used to reconstruct the original image which means that our representations are robust across different types of shapes.

The main idea behind our sketching algorithm is fairly simple and uses a locality sensitive hash (LSH) table. We partition the pixels of the image into regions based on the LSH and train specific parameters within each hash bucket so that those parameters can be used to reconstruct that region of the image. By using a few such LSH tables and training parameters in those buckets we obtain a succinct sketch that is able to provably reconstruct the image. This provides several advantages: 1) our sketches are invertible, that is one can reconstruct the original image; 2) they are semantically meaningful since they are isomorphic (using a smooth transform) to the latent parameters. This in turn means that any classification of shapes that can be done using the latent parameters can also be done using LSH sketches. If we think of each shape type (such as ellipse) as a separate region in coefficient space that is well separated from other regions by a polynomial threshold function (PTF) over the coefficient vector, then by training such a PTF classifier over the sketches of such images we obtain a robust classifier for such shapes. In fact in the coefficient space (which is homeomorphic to the set of produced sketches), we can view each shape type itself as an intersection of PTFs and thus the above tools for representing shapes in images could be used for representing shape-types in sketch space.

Our sketches also have the nice property that the sketch of a part of the image can be easily obtained as some of the dimensions in the full sketch. This means that one can create a secondary LSH index of sketches for different parts and different shapes in the image over a large collection of images. This gives a global dictionary of shape attributes so that each image merely becomes a set of points into this global dictionary. If we think of each shape-type as a disjoint compact region in coefficient space, then each contiguous set of buckets in this secondary index forms a shape-type and thus this secondary index can be viewed as a dictionary of shape-types.

We show the following results. An image can be viewed as map from pixel location to pixel values.

Proposition 1.1 (Informal, Image sketches and their properties)**.**

1. ***Low degree polynomials****: Given a grayscale image where the map from pixel position to the pixel value is a degree s polynomial, we construct sketch vector of size $O(s)$ that is able to reconstruct the image (see Theorem 4.4). Further the coefficient vector of the polynomial can be recovered by applying a linear transform over the sketch vector (see Appendix).*

2. ***Polynomial threshold functions****: If the image is a PTF where the underlying polynomial is of degree s we can get a sketch vector of size $O(s)$ that can reconstruct the image (Theorem 5.4).*

3. ***Parametric isomorphism****: The sketch is "isomorphic" to the original parameters used in producing the image. Furthermore, the transform from the coefficient parameters to the sketch is "smooth" (continuous and differentiable) (Lemma 5.5); whereas for PTFs the image itself (or a random projection of it) is not differentiable in the underlying coefficients.*

4. ***Union/Intersections of PTFs****: A union of intersection of k PTFs each of degree at most s can be converted into a sketch of size $poly(ks)$ from which the image can be reconstructed. (claim 6.2)*

5. ***Local property****: By projecting to specific dimensions, for any patch in the image one can obtain a sub-sketch from the sketch corresponding to that patch. One can create an index of*

patch sketches that can be used to retrieve patches from other images that are similar to a given patch (claim 6.3).

Theorem 1.2 (Learning Shape concepts, see claims 7.3, 7.4)**.** *Assuming shapes (such as ellipse) can be expressed as a region in coefficient space, and that each such region can be expressed as a PTF over the coefficient vector, and that these regions are well separated, our sketching methods can be used to build a robust high margin classifier for shapes that identifies these regions in sketch space and classifies an image sketch to its shape.*

1.1 Related Work

There is a tremendous amount of work on building features specifically for learning images similar to the ones we consider in the computer vision literature (for instance, *SIFT* features [13]). The literature is too vast to survey here — see [18] and references therein for instance. However, by and large while these methods are empirically well-understood they do not come with rigorous guarantees in terms of compressing images.

Learning good representations has also been posited as a key reason for the success of deep neural networks [2]. Deep representations of complex inputs such as text and images are often used to compare the underlying objects and transfer to new classification problems [20, 16]. However there is little theoretical understanding of the neural representations computed by such networks.

Locality sensitive hashing was originally introduced by [8] in the context of speeding up nearest-neighbor computations but has since found many applications in sketching in the streaming community. The LSH framework has been applied to deep learning and vision settings as well but our approach is tangential to these. In particular, it has mainly been used to speed up computations of various neural network architectures (see e.g., [9, 3]). We also study a more general paradigm recursive LSH based sketching. We also refer the reader to [19] for a survey of LSH.

There is also extensive work on learning PTFs and intersections of PTFs (e.g., [12, 10, 11]) in the high-dimensional setting under distributional assumptions. We also have distributional assumptions however the focus is on the very low-dimensional setting and dealing with an unsupervised learning problem with emphasis on computing compact representations.

2 Setup and Preliminaries

We focus on inputs which are 2D grayscale images which are square in shape for simplicity. All our algorithms and results easily extend to non-grayscale rectangular images as well. We view an image I as a real-valued function on a continuous 2-D domain $I : [0, 1]^2 \to [0, 1]$ mapping positions within a rectangular window to grayscale values.

Sampling Pixels of an Image: Given an image, we will assume the learner has n evaluations of I at points within $[0, 1]^2$ sampled uniformly at random. That is we are given $(x_1^i, x_2^i)_{i=1}^n \sim U[0, 1]^2$. These points can be thought of as pixels in a digital representation of the image. The vector of pixel values at the sampled positions is denoted by $\mathbf{v} \in [0, 1]^n$. We denote these n points as samples of I.

Underlying Generative Process for Images: We will model images as being the outputs of a generative process. We will consider the following simple generative processes.

Definition 2.1 (Polynomial Images)**.** The map $I : [0, 1]^2 \to [0, 1]$ is a degree-s polynomial in 2 variables (x_1, x_2).

Definition 2.2 (Thresholded Polynomial Images)**.** Given a polynomial $P(x)$, a polynomial threshold function (PTF) $f_P : \mathbb{R}^d \to \{0, 1\}$ is defined as $f_P(x) = \mathbb{1}\{P(x) \geq 0\}$. A thresholded polynomial image is a PTF on $[0, 1]^2$.

Rather than bounding the degree explicitly, we sometimes impose a 'smoothness' constraint on the polynomial generating our image.

Definition 2.3 (γ-Smoothness)**.** We say a function $f : [0, 1]^2 \to [0, 1]$ is γ-smooth if $\sum_{k,l=0}^{\infty} c_{k,l}^2 (k^2 + 1)(l^2 + 1) \leq \gamma^2$ where $c_{k,l}$ are the coefficients of the Fourier series of f restricted to $[0, 1]^2$.

We will also talk about classes of images which together can be thought of as constituting a *concept*. We assume that a class of images forms a contiguous region in the underlying parameter space which generates our image.

Locality Sensitive Hashing We will construct our image sketches using a form of hashing which is sensitive to distance called locality sensitive hashing which is a form of hashing defined on a metric space where nearby points hash to same buckets and far away points hash to different buckets.

Originally introduced for solving the approximate of exact Nearest Neighbor Search problem, LSH has since found applications in many other problem areas. Next we define the specific LSH family we use.

Definition 2.4 (Hyperplane LSH Family for Euclidean Distance)**.** The hyperplane LSH family $\mathcal{H}_{\|\|_2}(k, w)$ for the metric space $(\mathbb{R}^d, \|.\|_2)$ is one where each hash function is defined by a set of hyperplane inequalities (we use axis-aligned hyperplanes in some settings). We sample k random directions $a_1, \ldots, a_k$ where each coordinate of each a_i is an independent normal variable. In addition we sample $b_i \sim Unif[0, \epsilon]$ independently for $i \in [k]$. Given a hash function $h \in \mathcal{H}_{\|\|_2}(k, w)$ defined by the hyperplane directions $\{a_i\}_{i=1}^k$ and shifts $\{b_i\}_{i=1}^k$, $h(x) = \left(\left\lfloor \frac{a_i^\top x + b_i}{\epsilon} \right\rfloor\right)_{i=1}^D$.

3 Representation Learning using LSH

A natural assumption about images is that how quickly the pixel-values change with respect to distance in pixel locations conveys important information for most image-processing tasks. Convolutions crucially exploit this by using the same filter across various patches. We propose to exploit this by using LSH.

Definition 3.1. Simple LSH-Sketch of an Image I. Given n uniformly random samples of I, we define the Simple LSH-Sketch $S(I; m, k, w)$ of an image I as follows.

1. Using m random functions $\{f_j\}_{j=1}^m \in \mathcal{H}_{\|\|_2}(k, w)$, map the n pixel positions to corresponding hash buckets.

2. For each hash function f_j, we allow each hash bucket to store a scalar value. We denote the vector of the resulting bucket values by $\mathbf{z}_j \in \mathbb{R}^K$ where K is the number of hash buckets. The learnt value in each bucket will act as an approximation to the image in the region covered by that bucket.

Effectively, the Simple LSH sketch is a piecewise regression model where the pieces are determined by partitions induced by LSH. It can also be considered as a form of a smoothening of the image I (analogous to Gaussian smoothening).

Procedure for Reconstructing an Image from its Simple LSH-Sketch Given a Simple LSH-Sketch $S(I; m, k, w)$, we can reconstruct an approximation of I as follows. For any position (x_1, x_2), the reconstructed pixel value is defined as the average of the values stored in all the buckets that the point (x_1, x_2) hashes to under the m hash functions constituting the sketch. Let K denote the total number of buckets per each hash function. For the pixel values at the train locations $\mathbf{v} \in [0, 1]^n$, since the reconstruction is a linear operation it can also be represented in matrix form as $\mathbf{u} = A^\top \mathbf{z}$, where $\mathbf{z} = (\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_m) \in \mathbb{R}^{mK}$ is the vector of all bucket values concatenated and $A \in \mathbb{R}^{mk \times n}$ is a matrix mapping pixels to buckets such that

$$A(i, j) = \frac{1}{m} \text{ if } (x_1^j, x_2^j) \text{ maps to bucket } i, \quad A(i, j) = 0 \text{ otherwise.}$$

Computing the Optimal Bucket Values Given the above procedure to reconstruct the image, we now define the learning problem which we solve to learn the bucket values $\mathbf{z}_j$ for each hash function in the Simple LSH-Sketch. The values stored in the buckets aim to minimize the reconstruction error between the predicted pixel values $\mathbf{u}$ and the true pixel values $\mathbf{v}$. Our goal is to minimize the reconstruction error on a randomly drawn test pixel.

$$\mathcal{L}(\mathbf{u}, I) = \mathop{\mathbb{E}}_{\mathbf{x} \sim U[0,1]^2} \left[\ell(u(\mathbf{x}), I(\mathbf{x}))\right], \tag{1}$$

where $\ell(u, v) = (u - v)^2$ for polynomial images and $\ell(u, v) = \max(0, 1 - uv/\epsilon)$ (hinge loss) for PTF images. To achieve this goal, we minimize the ℓ_2-regularized training loss $\mathcal{L}_n(\mathbf{u}, I) = \sum_{i=1}^{n} \frac{1}{n}\ell(u_i, v_i) + \lambda\|\mathbf{z}\|_2^2$.

3.1 Multi-Level LSH Sketch (LSH-DAG Sketch)

The Simple LSH-Sketch works well for simple images defined using polynomials. For more complex images, it helps to build a multi-level LSH-Sketch defined using an underlying directed acyclic graph (DAG). We call this the LSH-DAG sketch.

Definition 3.2 (LSH-DAG Sketch)**.** A function class $F = \{f : [n]^2 \to \mathbb{R}\}$. is a DAG where for every internal node the outgoing edges are labeled according to the indices of buckets of a function from a LSH family. In addition, the nodes with no outgoing edges (i.e., leaves) are labeled with functions $f_\ell : [0, 1]^2 \to [0, 1]$ that are obtained as the threshold of a degree k polynomial on the input.

Recursive LSH sketching algorithm: Algorithm 1 describes how to compute a good multi-level LSH-Tree sketch of an image. At a high level, it computes an LSH-Tree where regions that are hard to learn at a certain resolution are recursively expanded into smaller regions to get a Tree of sketches.

Algorithm 1 Recursive Sketching Subroutine. Image or the region in a bucket may optionally be centred before sketching

1: Given a function $f : [0, 1]^2 \to [0, 1]$ pick a resolution r and map the pixels to LSH buckets with constant bucket width parameter r
2: Train a sketch in the buckets to best predict the pixel values in the image restricted to the region mapping to that bucket
3: Mark a bucket as invalid if the prediction error for the pixels mapping to the bucket is not negligible
4: Recursively compute a finer LSH sketch if the bucket is marked invalid by dividing the region into smaller regions with resolution $r/2$ (do not expand buckets whose region is fully covered by other valid buckets).

Merging near duplicate sketches in the LSH-Tree to get an LSH-DAG: we can merge sub-sketches and even entire sub-trees that are near duplicates to get a DAG structure instead of a recursive tree by using secondary LSH table of sketches. **Inputs**: A bucket B of a recursive LSH sketch, A global secondary LSH table of sketches $H_{sketches}$.

Algorithm 2 Recursive merging into LSH table of sketches

1: If the bucket B is a marked as a valid (thus leaf level bucket that is not further expanded) then store the sketch in the bucket into LSH table $H_{sketches}$. And replace the bucket contents by a pointer to the index of the entry in $H_{sketches}$
2: If a bucket B is not a leaf level bucket (so it is expanded into further buckets) then recursively merge the sketches in each of the component buckets. Look at the vector of merged sketches in the component buckets and hash to store this vector into $H_{sketches}$ and store only the index in B

4 Warm-up: Sketching Low-Complexity Polynomials

Polynomial images are simpler to sketch. For instance, a random projection of the image of a degree-s polynomial to $O(s^2)$ dimensions gives all 4 desiderata, (i) reconstruction, (ii) small sketch, (iii) recoverability of parameters, (iv) smoothness as a function of polynomial coefficients. However, it will be beneficial to see how we can use LSH to sketch polynomial images as the analysis will help us extend to more complex images in later sections. In fact, we will show that LSH works not only for sketching degree-bounded polynomials but for polynomials of arbitrary degree as long as their Fourier spectra are appropriately bounded as described in Assumption 4.1.

Assumption 4.1. We assume that the polynomial $p(x_1, x_2)$ generating the image satisfies:

1. Suppose $p(x_1, x_2) = \sum_{k,l=0}^{\infty} c_{k,l} \exp(2\pi i k x_1) \exp(2\pi i l x_2)$ is the Fourier series of p. We assume that $\sum_{k,l=0}^{\infty} c_{k,l}^2 (k^2+1)(l^2+1) \leq \gamma^2$ for some constant γ.

2. In the region $[0,1]^2$, the Lipschitz constant of $p(x_1, x_2)$ is bounded by Λ.

Note that our assumption on the Fourier spectrum of p is satisfied by bounded polynomials with bounded degrees in particular.

We start by first studying the limiting behavior of the axis-aligned LSH-Sketch as the number of hash functions $m \to \infty$.

Definition 4.2 (Infinite LSH-Sketch)**.** Given an image I, we sample m hash functions from $\mathcal{H}_{\perp}(m, w_1, w_2)$ for $w_1, w_2 \sim U[1/4, 1/2]$ and we analyze the properties of the resulting sketch as $m \to \infty$.

In this scenario, the metric we focus on for reconstruction is the mean squared error (MSE). Given the LSH Sketch $S(I; m, w_1, w_2)$, let $K \leq \lceil 1/w_1 \rceil \times \lceil 1/w_2 \rceil$ denote the total number of buckets per hash function. Let $A \in \mathbb{R}^{mK \times n}$ denote the mapping from the n sampled pixel positions $\{(x_1^i, x_2^i\}$ to buckets. We minimize $\frac{1}{n}\|A^\top \mathbf{z} - \mathbf{v}\|_2^2 + \frac{\lambda}{n}\|\mathbf{z}\|_2^2$.

We can obtain a closed form expression for the $\hat{\mathbf{z}}$ which minimizes the above objective. $\hat{\mathbf{z}} = A(A^\top A + \lambda I_n)^{-1}\mathbf{v}$.

Next, we observe that $\lim_{m\to\infty} A^\top A$ is a shift-invariant smoothening operator (convolution matrix) which attenuates higher frequencies in the image signal.

Lemma 4.3. *$\lim_{m\to\infty} A^\top A$ is a shift-invariant (convolution) matrix which acts also as a smoothening matrix. In the Fourier space, convolution of a signal I with $\lim_{m\to\infty} A^\top A$, corresponds to pointwise multiplication of the Fourier spectrum of I given by $\tilde{I}(k,l)$ with $\operatorname{sinc}(w_1 k)^2 \operatorname{sinc}(w_2 l^2)$.*

This realization from Lemma 4.3 suggests a Fourier style analysis of the solution $\hat{\mathbf{z}}$. Due to Assumption B.1 as well we can focus on the first few Fourier coefficients of the image and ignore the rest. This enables us to show the following.

Theorem 4.4 (Informal)**.** *With probability ≥ 0.9, for $n = \Theta(\gamma\Lambda/\operatorname{poly}(\epsilon))$ samples, $\mathbb{E}_{\boldsymbol{x}\sim U[0,1]^2}\left[(u(\boldsymbol{x}) - I(\boldsymbol{x}))^2\right] \leq \tilde{O}(\epsilon)$.*

In addition to Theorem 4.4 we also show that the resulting sketch has continuous and bounded derivatives with respect to the $\boldsymbol{\theta}$ (Lemma B.13 and that there exists a linear transformation R such that $R\hat{\mathbf{z}} = \boldsymbol{\theta}$ with high probability (Lemma B.14).

Finally, we show how we can obtain a sketch with a finite m which approximately still satisfies the above properties.

Remark 4.5. [sketches are resilient to sampled pixel positions] The above methods let us produce sketches of smooth images from sampling a few pixels in the image. The sketches only need random sampling and we can compare images even when different random pixel positions are sampled.

5 Sketching Thresholded Polynomial Images

Similar to the polynomials setting, we assume that the Fourier spectrum of the underlying polynomial p which we threshold is well-behaved. In addition the previous assumptions we had in Section 4 we also assume an additional condition about the boundary of the PTF. First we present a definition.

Definition 5.1 (ϵ-Boundary of a PTF)**.** The region where the absolute value of the underlying polynomial is $\leq \epsilon$.

The assumptions we make about our PTF image are given next.

Assumption 5.2.

1. Satisfy Assumptions 4.1.

2. The total probability mass of the ϵ-boundary of p is at most $O(\epsilon)$. For example, this holds for an ellipse represented by $ax^2 + by^2 - 1$ when $\sqrt{a^2+b^2} > c$ for a constant c.

We see how the sketch we construct can accurately reconstruct a PTF image. Due to the binary nature of the true pixel values, an objective such as cross-entropy or the hinge loss is found to

be more appropriate for computing the optimal bucket values. In particular, we use the following parametrization of hinge loss. For $y \in \{0, 1\}$, and for $\hat{y} \in \mathbb{R}$, $\ell_\epsilon(\hat{y}, y) = \max\left(0, 1 - \frac{\hat{y}y}{\epsilon}\right)$.

Similar to Section 4, we show that the sketch we get here has a low reconstruction error. In particular, we will only make errors near the boundary of the PTF and we can control this error by increasing the number of buckets per each hash function. In addition, we show that the manifold containing the sketches is Lipschitz and has bounded derivatives with respect to the coefficients of p.

5.1 Properties of The Infinite Sketch

As in Section 4, we begin by analyzing the sketch as $m \to \infty$. We first observe that as $m \to \infty$, our reconstruction procedure allows us the flexibility to express arbitrary smooth functions over a finite numbe of pixels.

Lemma 5.3. *As $m \to \infty$, any function f which is Lipschitz and has bounded derivatives can be expressed as $A^\top \boldsymbol{z}$ where $\boldsymbol{z}$ are the bucket values we learn.*

Lemma 5.3 allows us to argue that we can get a small reconstruction error for PTF images.

Theorem 5.4 (Informal)**.** *Constructing infinite sketch such that each hash function uses $O(1/\epsilon)$ buckets with $n = \Theta(\gamma\Lambda/\operatorname{poly}(\epsilon))$ samples will yield a solution with expected $0/1$-reconstruction error at most $\tilde{O}(\epsilon)$.*

A short overview of the proof of Theorem 5.4 is as follows. Recall that the hinge loss we optimize is $\max(0, 1 - uv/\epsilon)$ where $v \in \{0, 1\}$ is the true pixel value and u is the predicted reconstruction value. Note that this loss is 0 when $uv \geq \epsilon$. Given infinitely many hash functions, we can realize a reconstruction such that the reconstructed value at a pixel u is equal to the underlying polynomial value at v. This reconstruction will set $|uv| > \epsilon$ for v outside the ϵ-boundary and hence incur 0 hinge loss there. With our assumption on the boundedness of ϵ-boundary of p, we get that this solution incurs a train reconstruction loss of $O(\epsilon)$ implying that the optimal solution also incurs a loss of $O(\epsilon)$. The last step is a generalization bound to ensure that this behavior approximately holds for a randomly sampled test point as well.

5.2 Smoothness

We show that the sketch we compute is an analytically smooth function in terms of the latent parameters used to generate the image. Note that this property is not true if we just think of the mapping to the image itself. That is, say we have a degree d polynomial specified by coefficient parameters $\boldsymbol{\theta}$. Let $\mathbf{v}_{\boldsymbol{\theta}}$ denote the image generated by the PTF represented by coefficients $\boldsymbol{\theta}$. The mapping $\boldsymbol{\theta} \to \mathbf{v}_{\boldsymbol{\theta}}$ is non-differentiable whereas the mapping $\boldsymbol{\theta} \to S(\mathbf{v}_{\boldsymbol{\theta}}; m, \mathcal{H})$ is an analytic function with derivatives bounded independent of the number of pixels. This is a desirable property as it implies that any nice space of images whose latent parameters form a low-dimensional manifold will also be mapped to a low-dimensional manifold in the sketch space. Note that this not true for the image mapping as it preserves continuity but not differentiability.

Lemma 5.5 (Sketch-Smoothness)**.** *The infinite sketch produced is smooth in the PTF parameters and vice versa. If we use the image directly as the sketch then it is not smooth (first derivative is not defined).*

The main idea behind the proof is that as the sketch is the $\arg\min$ of a convex optimization problem, we can compute derivatives of the $\arg\min$ with respect to the parameters of the objective function using the *implicit function theorem*. See C.0.2 for the details of the proof.

As in Section 4, we can again go from the infinite sketch to a finite sized sketch with both a small reconstruction error and (approximate) smoothness. Details are given in the supplementary material.

6 Intersection of PTFs

Learning a rectangle We will now show how to construct a recursive sketch of a (constant-sized) rectangle image via our algorithm 1. We will first focus only on reconstruction. Instead of keeping a scalar in each bucket, we will keep a sketch vector within each bucket that can be used to reconstruct

the region that maps to that bucket. For simplicity, let us say we use an LSH that does say a square tiling with a random offset (shift) where each bucket is a square of side length say $1/c$. This would give about c^2 buckets. Within each bucket we run a PTF learning algorithm giving a $O(d)$ sized sketch that can learn any edge that can be expressed as a PTF of degree d as long as the tile intersects purely with at most one side of the rectangle.

Any tile-sketch that does not predict its content with high accuracy is marked as invalid and we recurse further by dividing into smaller tiles and to compute sketches at a higher resolution. We will only need to recurse to depth $\log n$ to go to the resolution of n pixels. Note that we can also amplify the coverage of valid tiles by using a constant number of random shifts of such LSH tilings and in this case we need to recurse into a tile only if it is not fully covered by tiles (from the different shifts) marked as valid. In expectation at least a constant fraction of each side will be covered by some tile that predicts correctly. The same idea works for an intersection of PTFs and also to a union of an intersection of PTFs. For the union of intersection of k PTFs if the boundary curve based on a PTF appears multiple times along the boundary split by intersections we will count them separately for each appearance. This proves the following claim.

Claim 6.1. W.h.p. a recursive LSH sketch of depth $O(\log n)$ and size $O(\log n)$ will be able to reconstruct the rectangle. W.h.p. a recursive LSH sketch of depth $O(\log n)$ and size $O(dk \log n)$ will be able to reconstruct an intersection (or union of intersection) of k PTFs of degree d each.

We will see how to further simplify these sketches by merging regions that have similar sketches.

Stable sketches: We will call a sketching method stable if for a smooth function, by sketching the function in a constant sized interval (or patch) we get the same sketch regardless of the interval (patch) position. By modifying the A matrix (for e.g. based on the polynomial kernel) we can get stable sketches (see Appendix).

Hashing sketches to merge duplicates: We can compress the LSH sketch by merging all near duplicate copies of the sketches from different parts of the same curve – this can be done by further hashing the sketches into another secondary LSH table of sketches and storing only one copy there and keeping only pointers to that table. This can also be done across different images; so the same curve being repeated across images gets replaced by a pointer. This merging can also be done recursively bottom up in a recursive fashion where each patch sketch at a leaf level is replaced by a pointer into the secondary LSH table and going up recursively an entire subtree may be replaced by a pointer so that all copies of a subtree point to the same entry in the global LSH table of sketches. Thus this global LSH table becomes a 'dictionary of sketches of shapes'. This gives us the following.

Claim 6.2 (Reconstruction)**.** By using stable sketches and merging similar sketches, w.h.p. the above sketch can be compressed to size $O(dk + k \log n)$ for intersection/union of k PTFs of degree d each.

Claim 6.3 (Indexing)**.** One can use an LSH table as an index of tile-sketches to find shifted copies of the same shape (specified by an intersection of PTFs). Further for a PTF, only a patch containing a part of the PTF is sufficient to index and lookup the shape.

7 Learning shapes as concepts

So far we have been viewing images as maps from pixel-locations to pixel-colors. However the same view can be applied to coefficient vectors or (sketch-vectors that can be viewed as smooth transforms of coefficient vectors) for a given shape type (such as ellipses). For simplicity let us just think of shapes in terms of their coefficient vectors that are d-dimensional assuming each image corresponding to a shape can be specified by a PTF of degree d. Each shape-type can now be viewed as a region in the coefficient space. For example all ellipses can be easily shown to be representable as an intersection of two PTFs in coefficient space .

We proved that the transform from coefficient to sketch space is smooth and invertible; we will extend this to the following assumption. (Note that for PTFs there is a linear map from the sketch vector to the coefficient vector it maps to.)

Assumption 7.1. We will assume that transform from the sketch space to the coefficient space is analytic and all derivatives are bounded by some constant. This immediately implies that any low degree polynomial in the coefficient space can be approximated by a low degree polynomial in the sketch space.

Assumption 7.2 (Model of Shapes in Real-World)**.** A shape is defined by a region (open/closed body with volume) in the coefficient space. These regions are non-overlapping and have a constant margin separating regions corresponding to any distinct shapes. We will assume each region is an intersection of PTFs.

This means that a constant margin PTF separating two shape-types remain separable by a constant margin PTF in the sketch-space. This gives the following theorem.

Claim 7.3. Given two shape-types assuming that there is a (robust) high margin PTF that separates them in coefficient space, a polynomial kernel can be used to separate the two regions with a high margin in sketch space.

Forming a dictionary of shapes: In fact, we will show how the recursive LSH algorithm described for classifying pixel-locations can actually be applied to classify coefficient vectors. One difference here is that we only get +vely labeled sketches that correspond to some shape in an image; however this is not an issue as we can still study this set of sketches by treating them as "pixels" at a meta level and handle the absence of -vely labeled pixels by randomly sample points as -ve labels (see Appendix). Even for an unsupervised set of images, each contiguous region in coefficient space can now be treated as a new latent shape label and its boundaries can be specified in a robust manner to precisely characterize the region spanned by the sketches corresponding to that shape. This gives the following claim.

Claim 7.4. The recursive LSH sketch algorithm when applied to sketches of images can be used to decompose images of shapes into distinct concepts (regions) in sketch space by learning a union of intersection of PTFs. Even when images are not labelled, we can treat each contiguous region in sketch space as a distinct latent concept and characterize the region robustly as intersection of PTFs.

8 Experiments

We demonstrate the smoothness of sketch vectors in the latent parameters used to generate the image in this section. As shown in Lemma 5.5, the infinite sketch is smooth as a function of the PTF parameters. Here we further demonstrate that finite sketch vectors also exhibit similar smoothness as we vary the latent parameters.

In particular, we consider the following three functions that generate ellipses, hyperbolas and rectangles images, respectively: $f_1(x,y) = (x-x_c)^2/a^2 + (y-y_c)^2/b^2 - 1$, $f_2(x,y) = (x-x_c)^2/a^2 - (y-y_c)^2/b^2 - 1$, $f_3(x,y) = I_{|x-x_c|\leq a}(x,y) \cdot I_{|y-y_c|\leq b}(x,y)$[2] We sample (x,y) uniformly in $[0,28] \times [0,28]$ and generate pixel values in $\{0,1\}$ with the above functions. The LSH network is trained to predict the correct pixel values and the trained LSH tables are the sketches for the corresponding images.

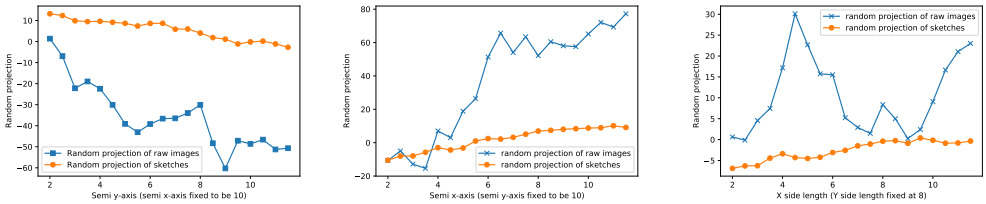
To examine the smoothness of the sketches, we apply projection of the sketches to a random direction. We also obtain a "raw image" from the functions by computing the pixel values on a uniform 56×56 grid in $[0,28] \times [0,28]$, and compare with the random projection of the raw images. Examples of such random projections are shown in Figure 1. In Table 1, we calculate the average absolute discrete derivatives of the projection with respect to the parameter, i.e. $\sum_i |\mathrm{Proj}(v_{i+1}) - \mathrm{Proj}(v_i)| / \sum_i |\theta_{i+1} - \theta_i|$, where v_i is the raw image vector (sketch vector) corresponding to parameter θ_i. The sketch vectors have much smaller average absolute discrete derivatives (in Figure 1, the sketch projections are almost flat, compared to large variations in the image projections), indicating a smooth dependency of the sketch vectors on the PTF parameters in the finite dimensional sketch case.

Table 1: Average absolute discrete derivative for varying function parameters

Shape	**Raw image**	**Sketch**
ellipse	12.22	2.29
hyperbola	18.31	2.63
rectangle	10.25	1.32

We also performed an initial set of experiments on using sketches for image classification. In this experiment, we sampled 10% of the train/test split of the fashion MNIST dataset, and obtained LSH

[2] I_A is the indicator function of set A.



(a) Ellipses (b) Hyperbolas (c) Rectangles

Figure 1: Examples of 1D random projection of raw images versus sketches. Projections of sketches have smaller variations when we change the parameters in its defining polynomials.

sketches of the images. We then train a 2 layer neural network (with a hidden layer of size 128 and ReLU activation) to classify the sketches. We found a 100% train accuray and 79.3% test accuracy for this task. This demonstrates that sketches have the potential to be used in practical applications.

References

[1] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. *Advances in Neural Information Processing Systems*, 31, 2018.

[2] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[3] Beidi Chen, Zichang Liu, Binghui Peng, Zhaozhuo Xu, Jonathan Lingjie Li, Tri Dao, Zhao Song, Anshumali Shrivastava, and Christopher Re. Mongoose: A learnable lsh framework for efficient neural network training. In *International Conference on Learning Representations*, 2020.

[4] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[5] Charles L Epstein. How well does the finite fourier transform approximate the fourier transform? *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 58(10):1421–1435, 2005.

[6] Feng-Lei Fan, Jinjun Xiong, Mengzhou Li, and Ge Wang. On interpretability of artificial neural networks: A survey. *IEEE Transactions on Radiation and Plasma Medical Sciences*, 2021.

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

[8] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *STOC*, pages 604–613. ACM, 1998.

[9] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *ICLR*. OpenReview.net, 2020.

[10] Adam R. Klivans, Ryan O'Donnell, and Rocco A. Servedio. Learning intersections and thresholds of halfspaces. *J. Comput. Syst. Sci.*, 68(4):808–840, 2004.

[11] Adam R. Klivans, Ryan O'Donnell, and Rocco A. Servedio. Learning geometric concepts via gaussian surface area. In *FOCS*, pages 541–550. IEEE Computer Society, 2008.

[12] Adam R. Klivans and Rocco A. Servedio. Learning DNF in time $2^{\tilde{o}(n^{1/3})}$. *J. Comput. Syst. Sci.*, 68(2):303–318, 2004.

[13] David G Lowe. Object recognition from local scale-invariant features. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 1150–1157. Ieee, 1999.

[14] Sheheryar Mehmood and Peter Ochs. Differentiating the value function by using convex duality. In Arindam Banerjee and Kenji Fukumizu, editors, *The 24th International Conference on Artificial Intelligence and Statistics, AISTATS 2021, April 13-15, 2021, Virtual Event*, volume 130 of *Proceedings of Machine Learning Research*, pages 3871–3879. PMLR, 2021.

[15] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[16] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1199–1208, 2018.

[17] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[18] Richard Szeliski. *Computer Vision - Algorithms and Applications, Second Edition*. Texts in Computer Science. Springer, 2022.

[19] Jingdong Wang, Heng Tao Shen, Jingkuan Song, and Jianqiu Ji. Hashing for similarity search: A survey. *arXiv preprint arXiv:1408.2927*, 2014.

[20] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of Big data*, 3(1):1–40, 2016.

A Additional Preliminaries

Definition A.1 (Fourier Series)**.** Given a T-periodic function $f(x)$, its Fourier series representation is a decomposition into a sum of Fourier basis functions.

$$f(x) = \sum_{k=0}^{\infty} c_k \exp(2\pi i k x/T). \tag{2}$$

The $\{c_k\}_{k=0}^{\infty}$ are referred to as the Fourier coefficients of x. For a (T_1, T_2)-periodic 2D function $f(x, y)$, its Fourier decomposition is analogously defined as

$$f(x_1, x_2) = \sum_{k,l=0}^{\infty} c_{k,l} \exp(2\pi i k x_1/T_1) \exp(2\pi i l x_2/T_2). \tag{3}$$

Definition A.2 (Discrete Fourier Transform)**.** The discrete Fourier transform transforms a sequence of N complex numbers $\{x_n\}_{n=1}^{N}$ into another sequence $\{\tilde{x}_k\}_{k=1}^{N}$ defined by

$$\tilde{x}_k = \sum_{n=1}^{N} x_n \exp\left(-\frac{i2\pi kn}{N}\right)$$

Given a function $f : [0, 1]^2 \to [0, 1]$ we can define a 1-periodic extension of f to $\mathbb{R}^2$ as follows $f_{periodic}(x_1, x_2) = f(x_1 - \lfloor x_1 \rfloor, x_2 - \lfloor x_2 \rfloor)$. In our work, when we need to consider a Fourier series representation for an image I, we use the Fourier series of $I_{periodic}$.

Definition A.3 (Rademacher Complexity)**.** The Rademacher complexity of a function class $\mathcal{F}$ is a useful quantity to understand how fast function averages for any $f \in \mathcal{F}$ converge to their mean value. Formally, the empirical Rademacher complexity of $\mathcal{F}$ on a sample set $S = (\mathbf{x_1}, \ldots, \mathbf{x_n})$ where each sample $\mathbf{x_i} \sim \mathcal{D}$, is defined as

$$\mathcal{R}_n(\mathcal{F}) = \frac{1}{n} \mathop{\mathbb{E}}_{\boldsymbol{\xi}} \left[\sup_{f \in \mathcal{F}} \sum_{i=1}^{n} \xi_i f(\mathbf{x_i}) \right],$$

where $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_n)$ is a vector of n i.i.d. Rademacher random variables (each is $+1$ w.p. $1/2$ and -1 w.p. $1/2$). The expected Rademacher complexity is then defined as

$$\mathbb{E}[\mathcal{R}_n(\mathcal{F})] = \frac{1}{n} \mathop{\mathbb{E}}_{\boldsymbol{\xi}, \{\mathbf{x_i}\}_{i=1}^{n} \sim \mathcal{D}^n} \left[\sup_{f \in \mathcal{F}} \sum_{i=1}^{n} \xi_i f(\mathbf{x_i}) \right]$$

Given the above definition of Rademacher complexity, we have the following lemma to bound the worst deviation of the population average from the corresponding sample average over all $f \in \mathcal{F}$ (also known as uniform convergence).

Lemma A.4 (Theorem 26.5 from [15])**.** *Given a function class $\mathcal{F}$ of functions on inputs $\mathbf{x}$, if for all $f \in \mathcal{F}$, and for all $\mathbf{x}$, $|f(\mathbf{x})| \leq c$, we have with probability $\geq 1 - \delta$,*

$$\mathop{\mathbb{E}}_{\mathbf{x} \sim \mathcal{D}}[f(\mathbf{x})] \leq \mathop{\mathbb{E}}_{n}[f(\mathbf{x})] + 2\,\mathbb{E}[\mathcal{R}_n(\mathcal{F})] + c\sqrt{\frac{2\log(2/\delta)}{n}}.$$

Definition A.5 (Locality Sensitive Hashing (LSH))**.** A locality sensitive hash function is one which maps similar inputs into the same hash bucket. Given a metric space $\mathcal{M} = (M, d)$, an LSH family $\mathcal{F} = \{f : M \to \mathbb{Z}\}$ mapping points of M to the integers is defined as follows. For any $p, q \in M$ and for any f chosen randomly from $\mathcal{F}$

1. if $d(p, q) \leq R$, then $\Pr[f(p) = f(q)] \geq p_1$,
2. if $d(p, q) > cR$, then $\Pr[f(p) = f(q)] \leq p_2$.

We say the family $\mathcal{F}$ is an (R, c, p_1, p_2)-LSH family.

Definition A.6 (Axis-Aligned Hyperplane LSH for Images)**.** The axis-aligned hyperplane LSH family for 2D images $\mathcal{H}_{\perp}(w_1, w_2)$ is a special case of $\mathcal{H}_{\|\|_2}(k, w)$ where $k = 2$ and the two hyperplanes are chosen to be the x_1 and x_2 axes respectively. The width parameters for each hyperplane are chosen to be w_1, w_2 respectively. Note that we allow for w_1 to be different from w_2 here.

B Full Details on Sketching Low-Complexity Polynomials (Appendix)

Polynomial images are simpler to sketch. For instance, a random projection of the image of a degree-s polynomial to $O(s^2)$ dimensions gives all 4 desiderata, (i) reconstruction, (ii) small sketch, (iii) recoverability of parameters, (iv) smoothness in the polynomial's coefficients. However, it will be beneficial to see how we can use LSH to sketch polynomial images as the analysis will help us extend to more complex images in later sections. In fact, we will show that LSH works not only for sketching degree-bounded polynomials but for polynomials of arbitrary degree as long as their Fourier spectra are appropriately bounded as described in Assumption B.1.

Assumption B.1. We assume that the polynomial $p(x_1, x_2)$ generating the image satisfies two properties:

1. Suppose $p(x_1, x_2) = \sum_{k,l=0}^{\infty} c_{k,l} \exp(2\pi i k x_1) \exp(2\pi i l x_2)$ is the Fourier series of p. We assume that $\sum_{k,l=0}^{\infty} c_{k,l}^2 (k^2+1)(l^2+1) \leq \gamma^2$ for some constant γ.
2. In the region $[0,1]^2$, the Lipschitz constant of $p(x_1, x_2)$ is bounded by Λ.

Lemma B.2 (The Image Norm is Bounded)**.** *We have that*

$$\int_{[0,1]^2} I(x_1, x_2)^2 dx_1 dx_2 \leq \gamma^2.$$

Proof. The statement follows from Assumption B.1 and an application of Parseval's theorem. From Parseval's theorem, we have that,

$$\int_{[0,1]^2} I(x_1, x_2)^2 dx_1 dx_2 = \sum_{k,l=0}^{\infty} c_{k,l}^2 \leq \gamma^2. \tag{4}$$

□

We start by studying the limiting behavior of the axis-aligned LSH-Sketch as the number of hash functions $m \to \infty$.

Definition B.3 (Infinite LSH-Sketch)**.** Given an image I, we sample m hash functions from $\mathcal{H}_\perp(m, w_1, w_2)$ for $w_1, w_2 \sim U[1/4, 1/2]$ and we analyze the properties of the resulting sketch as $m \to \infty$.

We will use Fourier analysis for our argument. Since we only have access to a set of finite samples from I, we will need the following lemma which says that with a large enough number of samples, the discrete Fourier transform of the sequence of samples is a good approximation to the coefficients of the Fourier series of the continuous function we sampled from.

Lemma B.4 (Approximation Error of Discrete Fourier Transform (Folklore, for e.g. see [5]))**.** *Given an image $I : [0,1]^2 \to [0,1]$ and n randomly sampled pixels from I, let $\boldsymbol{v}$ represent the sequence of pixel values sampled. If I has continuous derivatives and is Λ-Lipschitz,*

$$|\tilde{I}_n(k) - \tilde{I}(k)| \leq O\left(\frac{\Lambda}{n^2}\right).$$

In this scenario, the metric we focus on for reconstruction is the mean squared error (MSE). Given the LSH Sketch $S(I; m, w_1, w_2)$, let $K = \lceil 1/w_1 \rceil \times \lceil 1/w_2 \rceil$ denote the total number of buckets per hash function. Let $A \in \mathbb{R}^{mK \times n}$ denote the mapping from the n sampled pixel positions $\{(x_1^i, x_2^i\}$ to buckets. The regularized reconstruction loss we minimize is

$$\|A^\top \mathbf{z} - \mathbf{v}\|_2^2 + \lambda \|\mathbf{z}\|_2^2. \tag{5}$$

Since the above loss resembles the standard ridge regression objective, we can obtain a closed form expression for the $\hat{\mathbf{z}}$ which minimizes the above objective.

Lemma B.5. *The sketch which minimizes the reconstruction MSE on train samples is given by*

$$\hat{\boldsymbol{z}} = A(A^\top A + \lambda I_n)^{-1} \boldsymbol{v}.$$

Proof. Note that since $A^\top A$ is positive semi-definite, for any $\lambda > 0$, $A^\top A + \lambda I_n$ is invertible. The resulting closed form is well-known as the solution for the ridge regression loss. □

Next few Lemmas state some useful properties of the A matrix.

Lemma B.6. $\lim_{m\to\infty} A^\top A$ *is a shift-invariant (convolution) matrix which acts also as a smoothening matrix.*

Proof. We begin by observing that the $(i,j)^{th}$ entry of $\lim_{m\to\infty} A^\top A$ is proportional the probability that pixels i, j map to same bucket for a random hash function from $\mathcal{H}_\perp(w_1, w_2)$. That is,

$$\lim_{m\to\infty} (A^\top A)[i,j] = \frac{1}{m^2} \Pr_{h\sim\mathcal{H}_\perp(w_1,w_2)}[h(x_1^i, x_2^i) = h(x_1^j, x_2^j)] \tag{6}$$

$$= \frac{1}{m^2} \Pr[E_1] \Pr[E_2], \tag{7}$$

where E_1 (r. E_2) is the event that x_1^i, x_2^i (r. x_2^i, x_2^j) map to same bucket under the first (r. second) hyperplane. Hence we have established that the entries of $\lim_{m\to\infty} A^\top A$ only depend on the distance between the corresponding pixels and hence $\lim_{m\to\infty} A^\top A$ is a shift-invariant (convolution) matrix. In particular the effect of applying $\lim_{m\to\infty} A^\top A$ is similar to that of Gaussian smoothening which leads to attenuation of high frequency components in the input image I. To understand the amount of this attenuation we first focus on E_1. We observe that $\Pr[E_1] = \max(0, 1 - |x_1^i - x_1^j|/w_1)$. Define the rectangle function

$$\mathrm{rect}(x) = \begin{cases} 1 & \text{if } |x| \le 0.5 \\ 0 & \text{if otherwise} \end{cases}.$$

If $x_1^i - x_1^j = t$, then $\Pr[E_1]$ is exactly the convolution of $\mathrm{rect}(t/w_1)$ with itself. Therefore the Fourier transform of $\Pr[E_1]$ is the Fourier transform of $\mathrm{rect}(t/w_1)) * \mathrm{rect}(t/w_1)$ which is given by $\tilde{f}(k) = \mathrm{sinc}^2(\pi w_1 k) = \sin^2(\pi w_1 k)/((k^2+1)(l^2+1))$. Therefore, if we were dealing with 1D inputs, $m^2 \lim_{m\to\infty} A^\top A$ is essentially a convolution of our input function with the triangle function $\max(0, 1 - |t|/w_1)$. In 2D, we get a convolution of our input with the separable function $\max(0, 1 - |t_1|/w_1)\max(0, |t_2|/w_2)$ where $t_1 = x_1^i - x_1^j$ and $t_2 = x_2^i - x_2^j$. Due to separability, this can be viewed as convolving the image with $\max(0, 1 - |t_1|/w_1)$ first and then convolving with $\max(0, |t_2|/w_2)$. Suppose the Fourier transform of the image is denoted by $\tilde{I}(u,v)$. After the application of $\lim_{m\to\infty} A^\top A$, the resulting output's Fourier transform would be given by $\frac{1}{m^2}\tilde{I}(u,v)\,\mathrm{sinc}^2(\pi w_1 u)\,\mathrm{sinc}^2(\pi w_2 v)$.

If $w_1, w_2 \sim U[1/3, 2/3]$, then due to linearity of the Fourier transform, the resulting output's Fourier transform would be

$$\frac{1}{m^2} \mathop{\mathbb{E}}_{w_1,w_2}[\tilde{I}(u,v)\,\mathrm{sinc}^2(\pi w_1 u)\,\mathrm{sinc}^2(\pi w_2 v)],$$

which is (i) non-zero for all frequencies and crucially (ii) is $\Theta(1/m^2(k^2l^2+1))$ for all frequencies large than k. □

Remark B.7 (Polar co-ordinates). In 2D (or higher dimensions) it is also possible to carry out the Fourier analysis in polar coordinates in a radially symmetric manner via the Hankel transform. This can also be done inside a unit circle instead of $[0,1]^2$ using spherical harmonics. The analysis becomes more complicated but should produce essentially the same properties.

Lemma B.8 (Train Reconstruction Error of the Optimal Sketch)**.** *Given n samples, the train reconstruction error of a sketch using m hash functions with $\lambda = O\left(\frac{n}{m}\right)^{1/3}$ is*

$$\|A^\top \hat{\boldsymbol{z}} - \boldsymbol{v}\|_2^2 \le O\left(\frac{n}{m}\right)^{2/3} \gamma^2,$$

and the regularization term

$$\|\hat{\boldsymbol{z}}\|_2^2 \le O\left(\frac{n}{m}\right)^{1/3} \gamma^2.$$

Proof. We have that the train reconstruction error is

$$= \underbrace{\|A^\top \hat{\mathbf{z}} - \mathbf{v}\|_2^2}_{(1)} + \underbrace{\lambda \|\hat{\mathbf{z}}\|_2^2}_{(2)}. \tag{8}$$

Recall that $\hat{\mathbf{z}} = A(A^\top A + \lambda I_n)^{-1}\mathbf{v}$. Let $\lim_{m\to\infty} A^\top A = QDQ^{-1}$ be the eigendecomposition of $\lim_{m\to\infty} A^\top A$. Since A is a convolution matrix, its eigenvectors (columns of Q) are the Fourier basis vectors for the DFT over sequences of length n. D is a diagonal matrix containing the eigenvalues. We have,

$$\begin{aligned}
(1) &= \|(A^\top A(A^\top A + \lambda I_n)^{-1} - I_n)\mathbf{v}\|_2^2 && (9)\\
&= \|(QDQ^{-1}(Q(D+\lambda I_n)Q^{-1})^{-1} - I_n)\mathbf{v}\|_2^2 && (10)\\
&= \|Q(D(D+\lambda I_n)^{-1} - I_n)Q^{-1}\mathbf{v}\|_2^2. && (11)
\end{aligned}$$

Now we express $\mathbf{v} = Q\tilde{\mathbf{v}}$ in the Fourier representation. Then we have,

$$\begin{aligned}
(1) &= \|Q(D(D+\lambda I_n)^{-1} - I_n)Q^{-1}Q\tilde{\mathbf{v}}\|_2^2 && (12)\\
&= \|(D(D+\lambda I_n)^{-1} - I_n)\tilde{\mathbf{v}}\|_2^2 && (13)
\end{aligned}$$

Now $D(D+\lambda I_n)^{-1} - I_n$ is a diagonal matrix whose $(i,i)^{th}$ entry is $\frac{\lambda}{\lambda+\lambda_i}$ where λ_i is the eigenvalue corresponding to the ith eigenvector of $\lim_{m\to\infty} A^\top A$. From Lemm B.6, the set of $\{\lambda_i\}_{i=1}^n$ will correspond to the set $\{\lambda_{k,l}\}_{(k+1)(l+1)\le O(n)}$ of the 2D Fourier coefficients of $\lim_{m\to\infty} A^\top A$. We have also shown in Lemma B.6 that $\lambda_{k,l} = \Theta(1/(kl)^2)$. This implies that the diagonal entries of $D(D+\lambda I_n)^{-1} - I_n$ are of the form

$$\frac{\lambda}{\lambda + O(1/(k+1)^2(l+1)^2)}.$$

In addition, we have from Lemma B.4, that the entries of $\mathbf{v}$ correspond approximately to the Fourier coefficients of $p(x_1, x_2)$ up to $\pm O(\Lambda/n^2)$. This implies that

$$\begin{aligned}
\|(D(D+\lambda I_n)^{-1} - I_n)\tilde{\mathbf{v}}\|_2^2 &= \sum_{k,l:(k+1)(l+1)\le O(n)} \left(\frac{\lambda}{\lambda + O(1/(k+1)^2(l+1)^2)}\right)^2 \left(c_{k,l} + \frac{\Lambda}{n^2}\right)^2 && (14)\\
&\le \lambda^2 O(\gamma^2). && (15)
\end{aligned}$$

Next, we bound (2). Let $A = USV^\top$ be the SVD of A.

$$\begin{aligned}
(2) &= \lambda\|A(A^\top A + \lambda I_n)^{-1}\mathbf{v}\|_2^2 && (16)\\
&= \lambda\|USV^\top(VS^\top SV^\top + \lambda I_n)^{-1}\mathbf{v}\|_2^2 && (17)\\
&= \lambda\|USV^\top(V(S^\top S + \lambda I_n)V^\top)^{-1}\mathbf{v}\|_2^2 && (18)\\
&= \lambda\|US(S^\top S + \lambda I_n)^{-1}V^\top\mathbf{v}\|_2^2 && (19)\\
&\le \lambda\sigma_{\max}^2(S(S^\top S + \lambda I_n)^{-1})\|\mathbf{v}\|_2^2. && (20)
\end{aligned}$$

Now let $\sigma_1, \ldots, \sigma_n$ be the n largest singular values of A (the rest will all be 0). From Lemma B.10, we have that $\sigma_i \le \sqrt{n/m}$ for all i.

$$\begin{aligned}
\sigma_{\max}^2(S(S^\top S + \lambda I_n)^{-1}) &= \max_i \left(\frac{\sigma_i}{\sigma_i^2 + \lambda}\right)^2\\
&\le \frac{n}{m\lambda^2}
\end{aligned}$$

The combined train error is then bounded by

$$\lambda^2 O(\gamma^2) + \frac{n}{m\lambda} O(\gamma^2), \tag{21}$$

which is minimized at $\lambda = \left(\frac{n}{m}\right)^{1/3}$. Plugging in this value of λ we get total train error is bounded by

$$\le \left(\frac{n}{m}\right)^{2/3} O(\gamma^2). \tag{22}$$

□

Lemma B.9. *The expected test reconstruction error of the optimal sketch is* $O\left(\frac{\gamma}{\sqrt{n}}\right)$.

Proof. Since we have a linear problem we can apply we can apply Rademacher complexity based generalization bounds for linear hypothesis classes in our setting. □

Lemma B.10. $\|A\|_2 \leq \sqrt{\frac{n}{m}}$.

Proof. We have $\|A\|_2 \leq \|A\|_F = \sqrt{\frac{n}{m}}$. □

Lemma B.11 ($A^\top A + \lambda I_n$ is well-conditioned)**.** $\lim_{m\to\infty} \kappa(A^\top A + \lambda I_n) = 1$.

Proof. Let the SVD of $A = USV^\top$. Then,

$$A^\top A + \lambda I_n = VS^\top SV^\top + \lambda I \tag{23}$$

$$= VS^\top SV^\top + \lambda V I_n V^\top \tag{24}$$

$$= V(S^\top S + \lambda I_n)V^\top. \tag{25}$$

Therefore,

$$\kappa(A^\top A + \lambda I_n) = \kappa(S^\top S + \lambda I_n) \leq \frac{\sigma^2_{\max}(A) + \lambda}{\lambda} \tag{26}$$

$$\leq 1 + \frac{n}{m} \cdot \frac{m^{1/3}}{n^{1/3}} = 1 + \left(\frac{n}{m}\right)^{2/3} \tag{27}$$

□

Claim B.12. A has linearly independent columns as $m \to \infty$.

Proof. For any pixel position and a random hash function, the probability that it maps to a bucket where none of the other pixels map to is > 0. Hence as $m \to \infty$ the probability of this event happening with at least one hash function goes to 1. This is enough to get linear independence. The only case to take care of is if we get degenerate samples which happens with vanishingly small probability in the first place. □

Lemma B.13 (Smoothness of the Sketch)**.** *The derivatives of the sketch with respect to the polynomial coefficients which generate our image are continuous and bounded.*

Proof. Since p is Λ-Lipschitz, we have for a specific sample i,

$$\|\nabla_{\boldsymbol{\theta}} v_i\|_2 \leq \Lambda. \tag{28}$$

As a function of $\mathbf{v}$, our sketch is given by $\hat{\mathbf{z}} = A(A^\top A + \lambda I_n)^{-1} A\mathbf{v}$. Since $A^\top A + \lambda I_n$ is well-conditioned (Lemma B.11), and $\|A\|_2 \leq \sqrt{n/m}$ (Lemma B.10), this will imply that the derivatives of our sketch with respect to $\mathbf{v}$ are also bounded. By chain rule, we get that the derivatives of our sketch with respect to the polynomial coefficients are also bounded. □

Lemma B.14 (Recovering the Polynomial's Coefficients from its Sketch)**.** *Given the sketch* $\boldsymbol{z}$*, there exists a linear transformation* R *such that* $R\boldsymbol{z} = \boldsymbol{\theta}$.

Proof. Recall that $\mathbf{z} = A(A^\top A + \lambda I_n)^{-1}\mathbf{v}$. Here we note that $\mathrm{rank}((A^\top A + \lambda I_n)^{-1}) = \mathrm{rank}(A)$ because $(A^\top A + \lambda I_n)^{-1}$ is a full-rank square matrix and $\lim_{m\to\infty} \mathrm{rank}(A) = n$. With probability 1, we have that the map $\mathbf{v} = G\boldsymbol{\theta}$ also satisfies $rank(G) = p$. This is because G is a Vandermonde matrix which is known to be of full rank as long as none of the points in the sample set coincide. Two points in the sample set coinciding is a measure zero event. This implies that the linear transform from $\boldsymbol{\theta}$ to $\mathbf{z}$ is full-rank with probability 1 implying that it can be inverted to recover $\boldsymbol{\theta}$. □

B.0.1 Going to a Finite Sized Sketch

So far we have been assuming infinite sized sketch due to infinitely many LSH functions. However we will show how this can be reduced by picking bucket boundaries at a $O(\epsilon)$-grid; that is, the hyperplane shifts are multiples of ϵ along each axis. For simplicity let us look at the 1-D case and argue that a smooth function can still be approximated using these buckets. In 1-D there are at most $O(1/\epsilon)$ possible distinct buckets. We will only consider the pixels from this $O(\epsilon)$-grid. In this case the Fourier Transform becomes a DFT with $n' = O(1/\epsilon)$ pixels. We already know based on the weighted norm of the image that it can be approximated within error $O(\epsilon)$ using the first $O(1/\epsilon)$ Fourier basis functions which are exactly the DFT basis functions. The above Fourier analysis easily extends to DFT giving a smooth solution with $O(\epsilon)$ error on the n' pixels: the A matrix will be probability of mapping each of these discrete pixels to a bucket with the discrete boundaries; the DFT of the rectangle function also drops in magnitude as $O(1/k)$ for the kth frequency.

For constant γ, this solution can be extended to the entire domain $[0, 1]$ by predicting the same value for the entire interval between two successive grid points. We will argue that this will introduce at most $O(\epsilon)$ additional error: to see this, note that in the original image the discretization error within an interval depends on the derivative at that grid point. From the γ-smoothness, We already know the image has a bounded weighted sum of Fourier coefficients which means that the mean squared value of the derivative at these grid-points is also bounded. Thus the discretization error in the original image is at most $O(\epsilon)$. Since the predicted values are within $O(\epsilon)$ average error of the original values at the discretized pixel positions, extending those to the grid intervals will also introduce at most $O(\epsilon)$ error. In general we will need grid length of ϵ/γ.

Claim B.15. By discretizing the set of pixels to $O(\epsilon/\gamma)$-grid we can get the same guarantees as before by using sketches of size $O(\gamma^2/\epsilon^2)$ in 1-D and $O(\gamma^4/\epsilon^4)$ in 2-D.

C Full Details about Sketching Thresholded Polynomial Images

Similar to the polynomials setting, we assume that the Fourier spectrum of the underlying polynomial p which we threshold is well-behaved. In addition the previous assumptions we had in Section 4 we also assume an additional condition about the boundary of the PTF. First we present a definition.

Definition C.1 (ϵ-Boundary of a PTF)**.** The region where the absolute value of the underlying polynomial is $\leq \epsilon$.

The assumptions we make about our PTF image are given next.

Assumption C.2.

1. Suppose $p(x_1, x_2) = \sum_{k,l=0}^{\infty} c_{k,l} \exp(2\pi i k x_1) \exp(2\pi i l x_2)$ is the Fourier series of p. We assume that $\sum_{k,l=0}^{\infty} c_{k,l}^2 (k^2 + 1)(l^2 + 1) \leq \gamma^2$ for some constant γ.
2. In the region $[0, 1]^2$, the Lipschitz constant of $p(x_1, x_2)$ is bounded by Λ.
3. The total probability mass of the ϵ-boundary of p is at most $O(\epsilon)$. For example, this holds for an ellipse represented by $ax^2 + by^2 - 1$ when $\sqrt{a^2 + b^2} > c$ for a constant c.

We will see how the sketch we construct can accurately reconstruct a PTF image. Due to the binary nature of the true pixel values, an objective such as cross-entropy or the hinge loss is found to be more appropriate for computing the optimal bucket values. In particular, we use the following parameterization of hinge loss. For $y \in \{0, 1\}$, and for $\hat{y} \in \mathbb{R}$,

$$\ell_\epsilon(\hat{y}, y) = \max\left(0, 1 - \frac{\hat{y}y}{\epsilon}\right). \tag{29}$$

Similar to Section 4, we show that the sketch we get here has a low reconstruction error. In particular, we will only make errors near the boundary of the PTF and we can trade off this error by increasing the number of buckets per each hash function. In addition, we show that the manifold containing the sketches is Lipschitz and has bounded derivatives with respect to the coefficients of p.

C.0.1 Properties of Infinite Sketch

Lemma C.3 (Reconstruction using an Infinite Sketch can generate any Smooth Function over a Finite number of Pixels)**.**

Proof. This is mainly because $A^\top$ is going to be full rank with high probability. Firstly, the number of buckets far outnumber the number of pixels in this setting. Next, every Fourier coefficient will be allowed to be non-zero by this transformation. Suppose a sinusoid of a particular frequency θ maps to 0 when convolved with an LSH of a particular width $w(\theta)$. Then it will necessarily not map to 0 for at least some widths in a small interval around $w(\theta)$. Since we are considering an expected value over randomly drawn widths, this means that at no frequency is the expectation going to be 0. Therefore, we get that reconstruction using an infinite sketch is full rank (equivalent to saying that all Fourier coefficients are free to be set to values we like). Moreover, by bounding the norm of the buckets vector (through regularization), we control the smoothness of the resulting reconstruction. □

Lemma C.4. *Constructing an infinite sketch such that each hash function uses constant number of buckets will yield a solution with train reconstruction hinge loss at most $O(\epsilon)$.*

Proof. Recall that the hinge loss we optimize is $\max(0, 1 - uv/\epsilon)$. Note that this is 0 when $uv \geq \epsilon$. Given infinitely many hash functions, we can realize a reconstruction such that the reconstructed value at a pixel u is equal to the underlying polynomial value at v. This reconstruction will set $|uv^*| > \epsilon$ for v outside the ϵ-boundary and hence incur 0 hinge loss. Therefore, this solution incurs a total loss of $O(\epsilon)$ implying that the optimal solution also incurs a total loss of $O(\epsilon)$. □

Lemma C.5. *Given $O(1/\epsilon^2)$ training samples drawn from uniform distribution, we can achieve an average test hinge loss and consequently test $0/1$ error of reconstruction ϵ via the infinite sketch.*

Proof. This is a generalization bound which can be shown by bounding the norm of the values stored in the buckets. This is achieved due to the regularization term. This regularization will imply that $\sum_k c_k^2 O(k^2)$ is bounded where c_k is the kth Fourier coefficient of the reconstruction (before the sign). □

The above two lemmas give us the following theorem about learning PTF images.

Theorem C.6 (Learning PTFs using Infinite Sketch)**.** *Given $O(1/\epsilon^2)$ uniformly drawn samples from a PTF image, infinite sketch with $O(1)$ buckets per hash function.*

C.0.2 Smoothness of Sketches

We next prove 5.5.

Proof of Lemma 5.5. Let $\mathbf{v}$ be the true pixel values as generated by parameters $\boldsymbol{\theta}$. We will study the case where the sketch is chosen to be the minimizer of the population (aka continuous) loss function. That is, instead of considering a set of n pixels let us look at the setting where we view as the image as a continuous function: $v : [0, 1] \times [0, 1] \to \mathbb{R}$ with $\mathbf{v}_{\boldsymbol{\theta}}(x) = sign(p_{\boldsymbol{\theta}}(x))$ where $p_{\boldsymbol{\theta}}$ denotes the polynomial with parameters $\boldsymbol{\theta}$. Thus, the sketch is the minimizer of the cost function

$$\mathcal{L}(\boldsymbol{\theta}, \mathbf{z}) = \int_{\mathbf{x}} \ell(\mathbf{v}_{\boldsymbol{\theta}}(\mathbf{x}), \mathbf{u}_{\mathbf{z}}(\mathbf{x}))d\mathbf{x} + \lambda\|\mathbf{z}\|_2^2,$$

where $\mathbf{u}_{\mathbf{z}}(\mathbf{x})$ denotes the predicted image with sketch $\mathbf{z}$ and ℓ is the hinge-loss $\ell(a, b) = max(0, 1 - ab/\epsilon)$. The basic idea is that as the sketch is the $\arg\min$ of the above cost function, we can compute its derivatives with respect to the parameters $\boldsymbol{\theta}$ by the *implicit function theorem*. See appendix C.0.2 for the details of the proof.

Abusing notation a little, let

$$\mathbf{z}^*(\boldsymbol{\theta}) = \arg\min_{\mathbf{z}} \mathcal{L}(\boldsymbol{\theta}, \mathbf{z}).$$

As $\mathcal{L}$ is a strongly-convex function, the minimizer $\mathbf{z}^*(\boldsymbol{\theta})$ is also implicitly defined by the equation $\nabla_{\mathbf{z}}\mathcal{L}_{\boldsymbol{\theta}}(\mathbf{z}) = 0$. By using the implicit function theorem, the derivative of $\mathbf{z}^*(\boldsymbol{\theta})$ can be written (cf. [14] for instance) as

$$\nabla_\theta z^*(\theta) = (\nabla_z^2 \mathcal{L}(\boldsymbol{\theta}, z^*))^{-1} \partial_{z^*} \partial_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, z^*). \tag{30}$$

Note that by the regularizer in $\mathcal{L}$, $\nabla_z^2 \mathcal{L}(\boldsymbol{\theta}, z) \succeq 2\lambda I$ so that $\|(\nabla_z^2 \mathcal{L}(\boldsymbol{\theta}, z^*))^{-1}\| \leq 1/2\lambda = O(poly(1/\epsilon))$. It remains to bound the norm of the second term.

It follows from elementary calculus that

$$\nabla_\theta \mathcal{L}(\theta, z^*) = \int_x \frac{u_{z^*}(x)}{\epsilon} \delta(p_\theta(x)) \nabla_\theta p_\theta(x) dx = \int_{x:p_\theta(x)=0} \frac{u_{z^*}(x)}{\epsilon} \nabla_\theta p_\theta(x) dx, \tag{31}$$

where $\delta(\ )$ denotes the Dirac delta function.

Now, recall that $u_z(x) = \sum_{j=1}^m z_j A_j(x)$ where $A_j(x) = 1$ if x is hashed to bucket j and 0 else. Therefore,

$$\partial_{z_j} \nabla_\theta \mathcal{L}(\theta, z^*) = \int_{x:p_\theta(x)=0} \frac{A_j(x)}{\epsilon} dx.$$

Therefore, the norm of the vector $\partial_{z^*} \partial_\theta \mathcal{L}(\theta, z^*)$ is bounded.

A similar argument with more detailed calculations shows that even higher order derivatives of $z^*(\theta)$ with respect to θ exist and are bounded (we start with identities in Equations 30, 31, and use convenient identities such as $\int_x \delta'(x)\phi(x)dx = -\int_x \delta(x)\phi'(x)dx = \phi'(0)$).

In contrast, we next argue that the mapping $\boldsymbol{\theta} \to \mathbf{v}_{\boldsymbol{\theta}}$ while continuous is not necessarily differentiable in $\boldsymbol{\theta}$. To see this, consider a simple one-dimensional scenario where the PTF itself is of degree 1 and corresponds to the simple step function with one underlying parameter $t \in \mathbb{R}$ the step threshold. We are interested in the mapping that maps t to the corresponding step function $f_t(x) = 1$ if $x \geq t$ and 0 otherwise. Viewing this as a mapping from $\mathbb{R}$ to the Banach space of square-norm bounded functions on $\mathbb{R}$, we have $\lim_{h\to 0}(f_{t+h} - f_t)/h$ is the Dirac delta function centered at t which technically is not bounded in 2-norm. Note that this argument essentially implies that if we take images of rectangles and slide one of the boundaries, random projections of these images is a Brownian motion. Since the space covered by a Brownian motion is highly non-smooth we can conclude that the region formed by such images is also non-smooth □

D Stable sketches

Our Fourier analysis shows that LSH sketch acts like a Fourier basis kernel where the jth frequency gets scaled by $1/j$. Further a somewhat different A matrix can be used to tune the level of smoothness scale down higher frequencies at different levels and can even be used to essentially restrict the reconstructed image to the first d Fourier frequencies. Instead of using the Fourier basis functions We could also use the polynomial kernel corresponding to the first $O(d)$ monomials then we can exactly get the polynomial coefficients representing the function. Note that we only need to know some upper bound on the degree of the polynomial.

Claim D.1. By using a polynomial kernel one can learn a degree d polynomial using an $O(d)$ sized sketch. Further the sketch can be obtained from any patch of constant size. This gives us a stable sketch; that is, the same sketch is obtained regardless of which patch it is computed on: precisely, given a d polynomial of degree at most d and its evaluations at pixels in an interval of constant length $[a, b]$ within error $\epsilon = exp(-\Omega(d \log d))$ we can determine its coefficients within error ϵ.

Proof. The main point is even if we pick d points at gaps of $1/d$ then the vandermonde matrix has condition number at most $exp(d \log d)$. □

Instead of using a polynomial kernel one can also a more natural hash matrix A that is inspired from the normal distribution. The main point is that the Fourier transform of a normal distribution of a high width is also a normal distribution of low width which becomes like a low pass filter. We can use a discretized version such a matrix to exactly have non-zero weights only on the first few frequencies.

Claim D.2. In 1-dimension with $2n' + 1$ equispaced pixels denoted by x in the range $[-n'..n']$ the DFT of the function $\cos^{2l}(\pi x/(2n'))$ is a low pass filter with non-zero values only for frequencies in the range $[-3l, 3l]$. We can set $n' = O(1/\epsilon)$ to get an A matrix $A(x, y) = \cos^{2l}(\pi(x - y)/(2n'))$ that maps the pixels to $O(1/\epsilon)$ buckets so that $A^\top A$ functions precisely as a low pass filter that zeros out frequencies higher than $O(l)$.

Proof. Since the DFT and its inverse are symmetric, look at the discrete function $f(t)$ on $2n' + 1$ pixels in the range $[-n'..n']$ where $f(0) = 1/2, f(-1) = f(+1) = 1/4$ and f is 0 at all other

points. In the frequency domain it can be written as $F(k) = e^{jk\pi/n'}/4 + e^{-jk\pi/n'}/4 + 1/2 = (\cos(k\pi/n') + 1)/2 = \cos^2(k\pi/(2n'))$

Now if we take the self convolution of f l times its DFT will be $F^l = \cos^{2l}(k\pi/(2n'))$. Further since f is non-zero only in the range $[-1, 1]$ by a simple inductive argument, f convoluted l times is non-zero only in the range $[-3l, 3l]$. And all its values in that range are at least $1/4^l$ □

The argument for learning stable sketches can also be applied to Taylor series. If the curve can be written as $y = f(x)$ where f is an analytic function, then one can learn the initial few coefficients of the Taylor series expansion of f that approximate f within a tiny error (if the curve takes multiple values at a given x we can break it into regions and use random rotations so that this is not the case in at least one rotation). Again these set of coefficients form a sketch that is independent of which patch it has been computed from as long as the patch has a significant part of the curve. If there are n pixels one can approximate the first $\Omega(\log n)$ coefficients. Thus the above result can be extended to analytic functions with bounded derivatives.

Claim D.3. Given an analytic function that converges in $[0, 1]$ with all derivatives bounded in magnitude by some constant in the range $[0, 1]$ and its evaluations at points in the range $[a, b]$ within error ϵ we can determine its value in the entire range within error $\epsilon^{O(1)}$. A sketch of this extrapolation can serve as a stable sketch of the function.

Proof. If is well known that if we take say d points $x_i, ., x_d$ for an analytic function $f(x)$ taking values $y_1, .., y_d$ then the polynomial $f(x) = \sum y_i \Pi_{j\neq i}(x - x_j)/\Pi_{j\neq i}(x_i - x_j)$ fitted on these d points has error at most $f^{(d+1)}(\eta)/(d + 1)!\Pi(x - x_i)$ in the entire interval $[0, 1]$ for some $\eta \in [0, 1]$. Since all derivatives are constant this is at most $1/d! = exp(-O(d \log d)) = \delta$ (say). There is also the contribution from error in the measurements y_i in the extrapolation formula $f(x) = \sum y_i \Pi_{j\neq i}(x-x_j)/\Pi_{j\neq i}(x_i-x_j)$. For the error in this to be less than δ we need that the measurements y_i need to be such that $y_i \Pi_{j\neq i}(x - x_j)/\Pi_{j\neq i}(x_i - x_j)$ has error at most δ/d. If we choose all x_i in some constant sized interval to be values at least $O(1/d)$ apart then it means that the we want y_i to be accurate up to $\delta O(1/d)^d = exp(-O(d \log d)) = \delta^{O(1)}$. Thus if the error in the measurements is at most ϵ then we can extrapolate within error $\epsilon^{O(1)}$ in the entire interval $[0, 1]$. This becomes the stable sketch for f. The value of $d = O(\log(1/\epsilon)/ \log \log(1/\epsilon))$ □

The Taylor series approach can be used even for PTFs by fitting the points on the boundary to a polynomial.

E Intersection of PTFs

Figure 2: Plot of a thresholded ellipse (PTF) and an ellipse as a polynomial

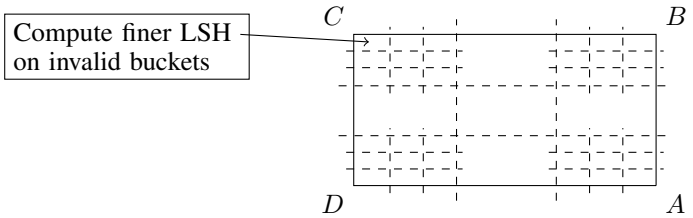


Figure 3: Rectangle tesselation based on Recursive Sketching Subroutine (Algorithm 1)

Remark E.1. [Tile sketches can be viewed as a smoother function]

The sketches can be viewed as a map from tile-positions to tile-sketches; this itself can be viewed as an "image mapping pixels (tile positions) to d-dimensional sketch vector (instead of value in R its a value in R^d). Note that this map can be viewed as a smoother than the original map from pixels to color as for most tiles of size $1/c$ their sketch will be the same if the tile is shifted by $O(1/c)$. The values stored at such tiles is not affected by a smoothening of the function by $O(1/c)$. Such tiles are sufficient to reconstruct the contents of other tiles that are covered by such tiles. This smoothend version of the map can be stored at a lower resolution to capture the image and hence methods described earlier can be applied to get a sketch of the map. Thus If we use a few random shifts of tilings and take a smoothened version of this map from tile-positions to tile-sketches, based on remark 4.5 we get a representation that is invariant to the random shift choices.

If we use these the sketches of the edges as features, then on top of these we will get a simple classifier for the rectangle. Note that a union of intersections over k features can be written as a degree k polynomial over the features which can be easily learned for small k.

Claim E.2. By LSH hashing the stable sketches from the tiles and using the top k sketches as features, we can build a simple classifier of size dk for a union of intersection of k PTFs of degree d.

LSH table of sketches as an inverse map: Note also that the LSH table of sketches can be viewed as a map from a patch-sketch to its tile-coordinate. This can be easily used to identify shifted copies of an image. For two shifted images the outputs of their maps will differ exactly by the shift value. One point to note is that a single patch bucket may correspond to multiple tiles – for such patches one would need to store a "set" of positions per bucket. It is possible to store a sketch of a set so that two shifted sets will have just shifted sketches. This can be done by first shifting the sets to make them mean zero and then sketching the sets; The set can again be viewed as a map from tile-positions to 0/1 depending on whether they are in set and then sketching this map based on Remark E.1.

Remark E.3 (Relation to Vision Transformers). Note that after merging tiles corresponding to similar sketches, one can learn a single sketch for the entire merged set of tiles. This is reminiscent of image transformer where related patches can attend to each other and merge into one token.

Remark E.4. [Shift resistant duplicate representation] Given two identical images that are shifts of each other they can be made "shift free" by first centering the image by say shifting the centroid to the origin. This will produce the same recursive sketch that can be stored once in the LSH table of sketches – this would be despite getting different random samples of pixels in the two images as per remark 4.5. This can even be applied to parts of an image. For example think about the same (say top-left) corner in two different (axis-aligned) rectangles. After the first level of LSH bucketing is done we get the tiles marked invalid that contain the corner. We can center the region covered by the union of these invalid tiles and then recursively continue the sketching. If we used sufficiently many shifts, the same corner from two different rectangles in different images will produce in expectation the same region after centering. Thus all say "top-left" corners of axis aligned rectangles will get the same recursive sketches in expectation.

So far there are several different methods that we have described for computing sketches: LSH sketch, polynomial kernel, Taylor series interpolation, the foward and the inverse maps from tile-positions to tile-sketches, mean and moments for a set. Different methods provide different guarantees. By keeping a "heterogenerous combination" of all these different sketches we get a single combined sketches different parts of which can be used for reconstruction, locality-properties, indexing etc.

LSH table of sketches as an inverse map: Note that the LSH table of sketches can be viewed as a map from a patch-sketch to its tile-coordinate. This can be easily used to identify shifted copies of an

image. For two shifted images the outputs of their maps will differ exactly by the shift value. One point to note is that a single patch bucket may correspond to multiple tiles – for such patches one would need to store a "set" of positions per bucket. We can also store a sketch of a set so that two shifted sets will have just shifted sketches. This can be done by first shifting the sets to make them mean zero and then sketching the sets; the set can again be viewed as a map from tile-positions to 0/1.

Note that since are sketch is a recursive sketch of tiles, it has the local property in the main proposition as the sketch of a part of an image easily be obtained form the full sketch.

Claim E.5 (Indexing)**.** One can use an LSH table as an index of tile-sketches to find shifted copies of the same shape (specified by an intersection of PTFs). Further for a PTF, only a patch containing a part of the PTF is sufficient to index and lookup the shape.

Remark E.6. [Intersections of PTFs in higher dimensions] For the intersection of PTFs we assumed so far 2-d images. There are some complexities that arise when we go to higher dimensions. For example if we look at the intersection of two halfspaces, if we use a recursive LSH, the edge formed from the intersection produces several small tiles (3-d tiles). If there are n grid points (pixels) total in 3-d then the edge will produce an $O(n^{1/3})$ sketch due to the tiles along the edge at the leaf level of the recursion; in k-dimensions there will be $O(n^{1-1/k})$ tiles. This can be avoided by clustering all such tiles at each depth of the recursion and learning a single smooth function for the entire union of those tiles at the next level of granularity that approximates the edge. Note that there is low degree polynomial (for example the equation of a cylinder along the edge) that is like a smoothened version of the edge. By decreasing the radius of the cylinder in each depth of the recursion we get better approximation to the edge each time. This gives an $O(\log n)$ sized sketch for each edge in a 3-d shape.

Note that even though we are using stable sketches for the above claims, even we if did not use such stable sketches and sketches of the same curve differ slightly across tiles we can look at similar sketches in adjacent tiles and try to merge them by finding if there is a single sketch that explains the union of adjacent tiles.

F Learning shapes as concepts

Claim F.1. Ellipses satisfy Assumption 7.2.

Proof. Look at the coefficients of the first six monomials in the expansion: $x^2, y^2, xy, x, y, 1$. Let the coefficients of the first three terms be a, b, c respectively. The coefficients will in fact define an ellipse if the corresponding 2×2 matrix $[[ac/2][c/2b]]$ is positive semi-definite. This in turn is equaivalent to three conditions: $a > 0$, $b > 0$ and $4ab > c^2$. Thus, in the coefficient space, ellipses correspond to an intersection of three PTFs. Look at the coefficients of the first six monomials in the expansion: $x^2, y^2, xy, x, y, 1$. Let the coefficients of the first three terms be a, b, c respectively. The coefficients will in fact define an ellipse if the corresponding 2×2 matrix $[[ac/2][c/2b]]$ is positive semi-definite. This in turn is equaivalent to three conditions: $a > 0$, $b > 0$ and $4ab > c^2$. Thus, in the coefficient space, ellipses correspond to an intersection of three PTFs. □

Remark F.2. [Sketching with only positively labeled pixels] Our sketching methods work even if do not have any negatively labeled pixels and only have pixels from inside the (union) of PTF. One difference is that we only get positively labeled points, that is, points inside the shape-region. We will show how negative labeled points can be generated by random sampling pixels from the space of pixels (say in R^k). Although this will introduce noise within the positive region this can be corrected by correcting each label to be positive if in some positive label in an ϵ-neighborhood around it. If we sample $\tilde{\Omega}((\sqrt{k}/\epsilon)^k)$ points in an $\epsilon/\sqrt{k}$-grid as negative labels then after correction all points will be correctly labeled except for points within ϵ of the boundary of a PTF.

We will assume that each shape type forms a disjoint region in coefficient space. And that union of these regions can be specified as a union of intersection of PTFs. Thus we can apply the algorithm from previous section (see remark E.6). A contiguous region of tiles can be identified as a separate shape/concept.