LEARNING FROM ONE AND ONLY ONE SHOT

Anonymous authors
Paper under double-blind review

ABSTRACT

Humans can generalize from one or a few examples, and even from very little pre-training on similar tasks. Machine learning (ML) algorithms, however, typically require large data to either learn or pre-learn to transfer. Inspired by nativism, we directly model very basic human innate priors in abstract visual tasks like character or doodle recognition. The result is a white-box model that learns transformation-based topological similarity akin to how a human would naturally and unconsciously "distort" an object when first seeing it. Using the simple Nearest-Neighbor classifier in this similarity space, our model approaches human-level character recognition using only one to ten examples per class and nothing else (no pre-training). This is in contrast to one-shot and few-shot settings that require significant pre-training. On standard benchmarks including MNIST, EMNIST-letters, and the harder Omniglot challenge, our model outperforms both neural-network-based and classical ML methods in the "tiny-data" regime, including few-shot learning models that use an extra background set to perform transfer learning. Moreover, mimicking simple clustering methods like k-means but in a non-Euclidean space, our model can adapt to an unsupervised setting and generate human-interpretable archetypes of a class.

1 INTRODUCTION

Modern machine learning (ML) systems have made remarkable progress but this has been accompanied by increasing model complexity, requiring hundreds of neural layers (e.g., ResNet-152) and millions of parameters (e.g., AlexNet: 62.3M, VGG16: 138M, BERT: 110M, GTP-3: 175B) for performance boosts. This results in a huge appetite for data as well as increasing difficulty in model interpretability—both for users to understand and for developers to tune (e.g., hyperparameters, architecture). As such, AI researchers have pushed for ML models that are both *prior-efficient* and *data-efficient* (Chollet, 2019) as well as that perform *human-like* learning (Lake et al., 2015) and that exhibit *human-interpretable* behaviors (Adadi & Berrada, 2018).

Think about MNIST. Whereas humans can perform reasonably well using just a handful of training examples, state-of-the-art models—those achieving near-perfection using all 60K training images—deteriorate fast when training size is greatly reduced. Being data-hungry is a real challenge in data-scarce scenarios, e.g., in low resource languages or one-shot learning as in Omniglot (Lake et al., 2015). Inspired by humans' innate topological visual intuitions, we design a white-box model that achieves near-human performance, e.g., reaching 80%, 90% MNIST test accuracies using only the first and first four training images per class, respectively, and achieving 6.75% error rate (human performance is 4.5%) in Omniglot one-shot learning without pre-training—one and only one shot.

The underlying idea is nativist: each time an example is presented to us humans, we make *abstractions*, so effectively we construct an equivalence class of unseen examples in our mind that are equivalent to the given example in several abstract senses. Many abilities of abstraction are inborn in humans and happen unconsciously. When a baby sees a mug, (s)he can immediately recognize the mug no matter if it is translated, rotated, scaled due to distance, or even deformed due to perspective. Such abstraction abilities are considered innate cf. the Core Knowledge priors (Spelke & Kinzler, 2007), rather than acquired later in life such as learning that a mug is topologically a donut.

We computationally realize the above intuition via our so-called *distortable canvas*—imagining every image being smoothly painted on a rubber canvas that can be distorted in many ways, e.g., bent, stretched, squeezed (Figure 1A). Due to rubber's viscosity, more distorted canvas transformations

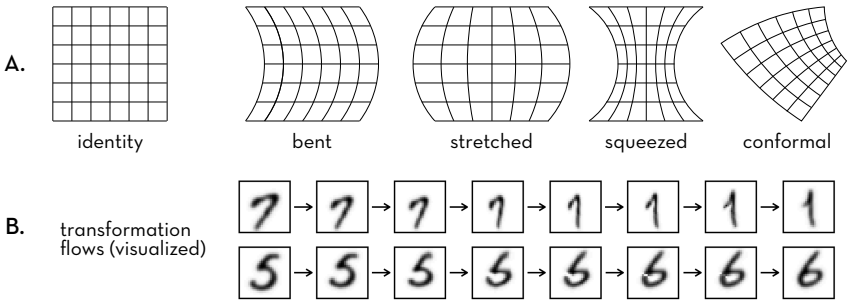


Figure 1: Canvas transformations (A) and transformation flows (B).

cost more energy. This induces a topological similarity, or distance, based on *minimal energy*: two images are similar if one can transform into the other without distorting too much. The distance is computed by minimizing canvas and color distortions (i.e., how well colors match after transforming). In general, one considers visual similarity up to not only a canvas transformation but also a color transformation. In this paper, we focus on canvas transformation, dealing with grayscale images only.

Besides a desired transformation (and distance) in the end, we apply the minimal-energy principle to the entire transformation process, i.e., to keep canvas and color distortions small along the whole optimization process (Mesa et al., 2019). When perceiving a translation, our mind does not process it as a sudden displacement from one location to another, but auto-completes a translation path—continuous and preferably short. Gradient descent naturally fits this goal by always choosing the steepest descent. Yet, it suffers from the *curse of local minima*. Our solution is to *lift gradient descent to multiple levels of abstraction* via multiscale canvas lattices and color blurring, mimicking again human abstraction ability that is extremely flexible in multiscale optimization. This yields visualizable and interpretable transformation flows (Figure 1B) that match existing human intuition (e.g., what humans would naturally do to transform a "7" to a "1") or provide human intuition to initially nonintuitive settings. The latter case suggests new human-interpretable transformations: ah, I did not realize this other way of transforming "7" to "1", but now I see it and it makes perfect sense!

We demostrate our model on visual tasks like character or doodle recognition. Our model can be directly used with the simple, human-intuitive Nearest-Neighbor method to classify images and with k-means style to cluster images. On benchmarks including MNIST (LeCun et al., 1998), EMNIST-letters (Cohen et al., 2017), and the harder Omniglot challenge (Lake et al., 2015), running Nearest-Neighbor in our learned similarity space outperforms both neural-network-based and classical ML methods in the "tiny data" or one-shot regime. This includes outperforming few-shot models that use extra background sets (which we do not need) to perform pre-training or transfer learning. For example, on MNIST, we need one and only one training example (per class) to reach 80% accuracy and four to reach 90%. In an unsupervised setting, we show how our model can do k-means-style clustering but in our human-like similarity space. We generate cluster centroids as archetypes of a particular class, e.g., different ways of writing "7" or doodling a giraffe.

Related work: few-shot learning, transfer learning, and meta-learning. The so-called few-shot or one-shot learning (Rezende et al., 2016; Salakhutdinov et al., 2012) is similar to our considered tiny-data regime. The difference lies in whether additional background data is used to pre-learn. Few-shot learning is a special case of transferring learning (Pan & Yang, 2009) or dataset shift (Storkey, 2009). However, one downside of transferring learning is that the transferred knowledge may not be directly human-interpretable, making it hard to predict whether a transfer would be effective. Likewise meta-learning or structure tuning, which also requires an extra or held-out dataset. We do not require any additional background or validation data. We learn from one and only that one shot.

Related work: transformation and data augmentation. Our distortable canvas is a transformation-based model, conceptually similar to all other such models e.g., optimal transportation maps (Villani, 2009), as well as transformation-induced symmetries e.g., SIFT descriptors (Lowe, 1999), equivariances (Bronstein et al., 2017), or data augmentation techniques (Krizhevsky et al., 2012). Our model considers all transformations (not just isometries or affine), effectively does data augmentation without actually generating data, and can be solved efficiently via our abstracted gradient descent.

2 SMOOTH IMAGE ON DISTORTABLE CANVAS

We introduce a *distortable canvas* model where all images in the world are thought of as smoothly painted on a rubber canvas that can be bent, stretched, etc. This allows us to further introduce *canvas transformations* that can flexibly "jitter" or "distort" the image like we naturally do in our mind. More specifically, we define a *smooth image* by a piecewise differentiable function $\mathcal{M} : \mathbb{R}^2 \to \mathbb{R}_+$, where $\mathbb{R}^2$ is viewed as a *canvas* and $\mathbb{R}_+$ denotes *color* in a single channel—grayscale in this paper. We define a *canvas transformation* by a function $\alpha : \mathbb{R}^2 \to \mathbb{R}^2$, which "reshapes" the underlying canvas of a smooth image. Common examples include translation $\alpha(x) := x + v$, scaling $\alpha(x) := sx$, rotation $\alpha(x) := Rx$, and affine $\alpha(x) := Ax + v$. We also define a *color transformation* by a function $\chi : \mathbb{R}_+ \to \mathbb{R}_+$, which "repaints" a color. In this paper, we make color transformation easy, only used to adjust image contrast via affine $\chi(c) := ac + b$. On the contrary, we do *not* restrict canvas transformation to any pre-designated type (e.g., isometries or affine), but consider *all* 2D transformations in general.

Given a smooth image $\mathcal{M}$, a canvas transformation α, and a color transformation χ (possibly identity), the composition $\chi \circ \mathcal{M} \circ \alpha$ denotes the transformed image. Notably, canvas transformation is like *change of coordinates* (of the canvas), so visually the transformed image $\mathcal{M} \circ \alpha$ inversely corresponds to the transformation α, e.g., if $\alpha(x) = 2x$, the image is scaled down by $1/2$.

To mimic humans' innate topological intuition about visual similarity, we introduce *canvas distortion* $\mathcal{D}_V(\alpha)$ for any canvas transformation α and *color distortion* $\mathcal{D}_C(\mathcal{M}, \mathcal{M}')$ between any two smooth images $\mathcal{M}, \mathcal{M}'$. The high-level idea is to search for a transformation that mimics what a human would naturally do to transform one image to another. That is, a low-distorted α that makes little difference in color between $\mathcal{M}$ and the transformed $\mathcal{M}'$, or more precisely, an α that minimizes both

$$\mathcal{D}_V(\alpha) \quad \text{and} \quad \mathcal{D}_C(\mathcal{M},\ \chi \circ \mathcal{M}' \circ \alpha). \tag{1}$$

Representating digital and smoothed images. An $m \times n$ *digital image* is a discrete function $\mathsf{M} : [m] \times [n] \to [0, 1]$, where $[k] := \{0, 1, \ldots, k-1\}$. We call $[m] \times [n]$ the *canvas grid* and any $z \in [m] \times [n]$ a *grid point*. For any $m \times n$ digital image M, we smooth it to $\mathcal{M}$ via a *sum of kernels*:

$$\mathcal{M}(x) := \sum_{z \in [m] \times [n]} \mathsf{M}(z) \cdot \kappa(\rho(z, x)) \quad \text{for any } x \in \mathbb{R}^2, \tag{2}$$

where a kernel $\kappa : \mathbb{R}_+ \to \mathbb{R}_+$ is a decaying function (e.g., linear, polynomial, exponential decay) and ρ is a metric on $\mathbb{R}^2$ (e.g., $\ell_1, \ell_2, \ell_\infty$). In this paper, we use linear decay for κ and ℓ_∞ for ρ, i.e., $\kappa(\rho(z, x)) = 1 - \frac{1}{\rho_c}\|z - x\|_\infty$ if $\|z - x\|_\infty < \rho_c$ (for some cutoff radius $\rho_c > 0$) and $\kappa(\rho(z, x)) = 0$ otherwise. Note the smoothed image $\mathcal{M}$ is well-defined everywhere on $\mathbb{R}^2$. This differs from Gaussian blurring as we do not discretize kernels. It is key for our model to use the smoothed image as input. So, we always smooth any digital image at first and then only deal with the smoothed image.

Representing arbitrary canvas transformations. We consider all possible 2D transformations (not necessarily given by a formula), but how do we represent them in a computer? We can start with the standard grid $[m] \times [n]$ of the original digital image (or any other grid) and use $\alpha([m] \times [n])$—*the transformed grid*—to represent α digitally. Accordingly, any canvas transformation is represented by a matrix $\boldsymbol{\alpha} \in \mathbb{R}^{(mn)\times 2}$ containing the 2D coordinates of each transformed grid point row by row. We use the lexicographical order of a 2D grid. For example, the $[2] \times [3]$ grid is ordered as the list $[(0, 0), (0, 1), (0, 2), (1, 0), (1, 1), (1, 2)]$. With respect to this $[2] \times [3]$ grid, an arbitrary canvas transformation and as a special case, the identity transformation, are represented respectively as

$$\alpha \overset{d}{=} \boldsymbol{\alpha} := \begin{bmatrix} \boldsymbol{\alpha}_{00} & \boldsymbol{\alpha}_{01} \\ \boldsymbol{\alpha}_{10} & \boldsymbol{\alpha}_{11} \\ \boldsymbol{\alpha}_{20} & \boldsymbol{\alpha}_{21} \\ \boldsymbol{\alpha}_{30} & \boldsymbol{\alpha}_{31} \\ \boldsymbol{\alpha}_{40} & \boldsymbol{\alpha}_{41} \\ \boldsymbol{\alpha}_{50} & \boldsymbol{\alpha}_{51} \end{bmatrix} \quad \text{and} \quad \mathrm{id} \overset{d}{=} \mathbf{id} := \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 0 & 2 \\ 1 & 0 \\ 1 & 1 \\ 1 & 2 \end{bmatrix}, \tag{3}$$

where the ith row $\boldsymbol{\alpha}_i := (\boldsymbol{\alpha}_{i0}, \boldsymbol{\alpha}_{i1})$ records the coordinates of the ith grid point after transformation. In the above and throughout this paper, we use $\overset{d}{=}$ to mean "is digitally represented by". Under this notation, any transformed image $\mathcal{M} \circ \alpha \overset{d}{=} \mathcal{M}(\boldsymbol{\alpha}) := (\ \mathcal{M}(\boldsymbol{\alpha}_0),\ \ldots,\ \mathcal{M}(\boldsymbol{\alpha}_{(mn-1)})\ ) \in \mathbb{R}^{(mn)}$, i.e., a (vectorized) digital image sampled from $\mathcal{M}$ at the transformed grid $\boldsymbol{\alpha}$.

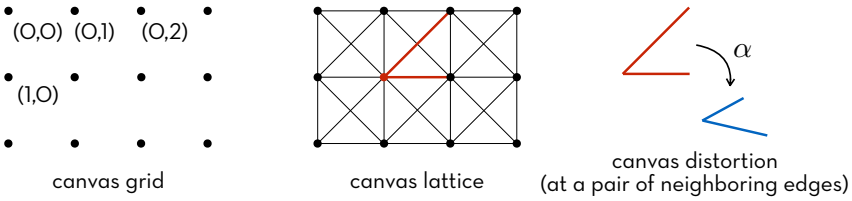


Figure 2: Canvas grid and its corresponding lattice. Local distortions are computed at every pair of neighboring edges, with one example of neighboring edges is highlighted in red.

Representing color and canvas distortions. The color distortion $\mathcal{D}_C$ measures the color discrepancy between $\mathcal{M}(\mathbf{id})$ and $\mathcal{M}'(\boldsymbol{\alpha})$ up to an affine color transformation χ. The canvas distortion $\mathcal{D}_V$ measures the distortion between the original grid $\mathbf{id}$ and the transformed grid $\boldsymbol{\alpha}$. Formally,

$$\mathcal{D}_C(\mathcal{M},\ \chi\circ\mathcal{M}'\circ\alpha) \overset{d}{=} \mathcal{D}_C(\ \mathcal{M}(\mathbf{id}),\ \chi(\mathcal{M}'(\boldsymbol{\alpha}))\ ) := \|a\mathcal{M}'(\boldsymbol{\alpha}) + b - \mathcal{M}(\mathbf{id})\|_2^2 \tag{4}$$

$$\mathcal{D}_V(\alpha) \overset{d}{=} \mathcal{D}_V(\mathbf{id},\boldsymbol{\alpha}) := \max_{\{\{i,j\},\{i',j'\}\}\in B_E} \left|\Delta^{\boldsymbol{\alpha}}_{\{i,j\}} - \Delta^{\boldsymbol{\alpha}}_{\{i',j'\}}\right|,\ \Delta^{\boldsymbol{\alpha}}_{\{i,j\}} := \log\frac{\|\boldsymbol{\alpha}_i - \boldsymbol{\alpha}_j\|_2}{\|\mathbf{id}_i - \mathbf{id}_j\|_2} \tag{5}$$

Here B_E comprises all pairs of neighboring edges in a *canvas lattice*, which we will introduce below. The whole formula (5) is derived from the mathematical definition of *distortion of a function* by discretizing it across the canvas lattice. This formula measures how far an arbitrary transformation is from being *conformal*, which is flexible for *local* isometries and scaling. Given a canvas grid $[m]\times[n]$, its corresponding *canvas lattice* is an undirected graph $\mathcal{L}=(V,E)$, with the set of vertices being the grid $V=[m]\times[n]$ and the set of edges obtained by connecting neighboring vertices in the ℓ_∞ sense: $E=\{\{i,j\}\mid \|v_i-v_j\|_\infty=1 \text{ for } v_i,v_j\in V\}$. We say two edges are neighbors if they form a 45° angle (Figure 2).

Computing topological distance by minimizing distortions. To minimize color and canvas distortions (4) and (5), we consider two dual views: minimizing $\mathcal{D}_C$ among low-distorted α's or minimizing $\mathcal{D}_V$ among best-matching α's. We write the two views as the following two constrained optimization problems, together with their respective unconstrained equivalents: with $\epsilon\to 0_+$ and $\mu\to 0_+$,

$$\min_{\alpha,\chi}\ \mathcal{D}_C(\mathcal{M},\ \chi\circ\mathcal{M}'\circ\alpha) \quad \text{s.t. } \mathcal{D}_V(\alpha)\le\epsilon \iff \min_{\alpha,\chi}\ \mathcal{D}_V(\alpha) + \mu\mathcal{D}_C(\mathcal{M},\ \chi\circ\mathcal{M}'\circ\alpha) \tag{6}$$

$$\min_{\alpha,\chi}\ \mathcal{D}_V(\alpha) \quad \text{s.t. } \mathcal{D}_C(\mathcal{M},\ \chi\circ\mathcal{M}'\circ\alpha)\le\epsilon \iff \min_{\alpha,\chi}\ \mathcal{D}_C(\mathcal{M},\ \chi\circ\mathcal{M}'\circ\alpha) + \mu\mathcal{D}_V(\alpha) \tag{7}$$

We let the optimal values $\mathcal{D}_C^\star$ for (6) and $\mathcal{D}_V^\star$ for (7) denote the two versions of our desired topological distance aimed for mimicking human's innate topological intuition. We call them *$\mathcal{D}_C$-distance* and *$\mathcal{D}_V$-distance*, respectively. It remains to characterize their metric properties, but we nevertheless use them to measure inverse topological similarity.

Transformation flow. To obtain both transformations and transformation processes that are human-like, we run gradient descent, or projected gradient descent for constrained problems. The iterative gradient steps allow us to not only obtain a desired transformation $\alpha^\star$ in the end, but also elicit a *transformation flow* $\mathrm{id}=\alpha^{(0)}\to\alpha^{(1)}\to\cdots\to\alpha^\star$. The resulting sequence of the transformed images $\mathcal{M}'=\mathcal{M}'\circ\alpha^{(0)}\to\mathcal{M}'\circ\alpha^{(1)}\to\cdots\to\mathcal{M}'\circ\alpha^\star\approx\mathcal{M}$ (we omit χ for simplicity) makes up an animation, which helps with human intuition on transforming $\mathcal{M}'$ to $\mathcal{M}$. However, directly running (projected) gradient descent on (6) or (7) does not work, as it suffers greatly from the curse of local minima, which we discuss and solve in the next section.

3 Gradient Descent at Higher Levels of Abstraction

The canvas distortion $\mathcal{D}_C$ is invariant under a variety of transformations (e.g., $\mathcal{D}_V(\alpha)=0$ for any conformal α) which nicely mimics humans' flexible transformation options, but this also implies lots of local and global minima, and other critical points where the gradient is zero. How much the color distortion $\mathcal{D}_C$ fluctuates as a function of α depends on the images $\mathcal{M},\mathcal{M}'$. But in most cases, $\mathcal{D}_C$ also has lots of local and global minima, the majority of which represent unwanted "short cuts"—unnatural transformations that make $\mathcal{D}_C\to 0$ but would break the rubber canvas or create holes in it. The curse of vanishing gradient can freeze up the gradient descent method. To unfreeze the

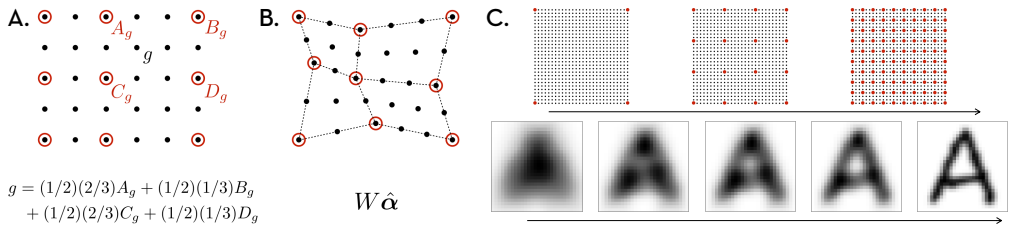


Figure 3: An anchor system (A) and its transformation (B). (C) exemplifies a configuration of $(\hat{G}, \rho_c)$-solution path consisting of a chain of anchor grids/lattices and a chain of blurring.

gradient steps, we lift gradient descent to higher levels, mimicking once again humans' abstraction power, as our internal optimization system is quite flexible in making "gradient-descent" moves at different levels of abstraction. We design two abstraction techniques: a chain of anchor lattices to make hierarchical abstractions of canvas transformations and a chain of color blurring to make hierarchical abstractions of the image painted on the canvas.

Anchor grids and lattices. An *anchor grid* and its corresponding *anchor lattice* offer a simpler parameterization, i.e., an abstraction, of canvas transformations. Without such an abstraction, any transformed $[m] \times [n]$ grid $\boldsymbol{\alpha} \in \mathbb{R}^{(mn)\times 2}$ consists of $2mn$ free parameters. So, the optimization problems (6) and (7) are in dimension $2mn+2$, which is not only computationally inefficient for large images but has too much freedom for vanishing gradient. We use a simpler $\boldsymbol{\alpha}$-parameterization that regularizes transformation, lowers distortion, and agrees with our intuition on rubber transformations.

Formally, an *anchor system* $(G, \hat{G}) = (M \times N, \hat{M} \times \hat{N})$ uses two layers of grids: an *underlying grid* G and an *anchor grid* $\hat{G}$ atop, satisfying $\hat{M} \subseteq M$, $\hat{N} \subseteq N$, and $G \subseteq \text{ConvexHull}(\hat{G})$. Figure 3A shows one example, where $G = [5] \times [6] = \{0, \ldots, 4\} \times \{0, \ldots, 5\}$ and $\hat{G} = \{0, 2, 4\} \times \{0, 2, 5\}$. Under an anchor system, we can uniquely represent any grid point $g \in G$ via four anchors $A_g, B_g, C_g, D_g \in \hat{G}$ via proportional interpolation, or more precisely, the double convex combination

$$g = (1-\lambda_g)(1-\nu_g)A_g + (1-\lambda_g)\nu_g B_g + \lambda_g(1-\nu_g)C_g + \lambda_g\nu_g D_g. \tag{8}$$

Here $A_gB_gD_gC_g$ can be uniquely picked as the smallest rectangle in $\hat{G}$'s lattice containing g; the two weight parameters λ_g, ν_g can be readily computed based on the relative position, e.g., as in Figure 3A. The above relation between grid points and anchors can be succinctly encoded by a weight matrix $W \in \mathbb{R}^{|G|\times|\hat{G}|}$ with the ith row encoding the weights of the ith grid point, say g, as in (8). Particularly, the ith row contains at most four non-zero entries $(1-\lambda_g)(1-\nu_g), (1-\lambda_g)\nu_g, \lambda_g(1-\nu_g), \lambda_g\nu_g$ located at the columns corresponding to A_g, B_g, C_g, D_g, respectively.

Given an anchor system $(G, \hat{G})$, any canvas transformation α is digitally represented by $\boldsymbol{\alpha} \in \mathbb{R}^{|G|\times 2}$ under G and by $\hat{\boldsymbol{\alpha}} \in \mathbb{R}^{|\hat{G}|\times 2}$ under $\hat{G}$. By definition, $\hat{\boldsymbol{\alpha}}$ is a submatrix of $\boldsymbol{\alpha}$, which induces an equivalence relation on the set of all canvas transformations: two transformations α, β, or $\boldsymbol{\alpha}, \boldsymbol{\beta}$ under the digital representation, are equivalent if and only if $\hat{\boldsymbol{\alpha}} = \hat{\boldsymbol{\beta}}$. So, any given $\hat{\boldsymbol{\alpha}}$ abstracts the equivalence class $\{\boldsymbol{\beta} \mid \hat{\boldsymbol{\beta}} = \hat{\boldsymbol{\alpha}}\}$. Based on the maximum entropy principle (Jaynes, 1957), a reasonable selection of a representative of this equivalence class is $W\hat{\boldsymbol{\alpha}}$. One can check that $W\hat{\boldsymbol{\alpha}}$ is indeed an element in $\{\boldsymbol{\beta} \mid \hat{\boldsymbol{\beta}} = \hat{\boldsymbol{\alpha}}\}$ and evenly distributes the transformed grid points by preserving their relative positions to anchors. Figure 3B illustrates this type of even distribution, which agrees with the intuition on how a rubber surface would naturally react when transformation forces are applied at anchors.

Using an anchor system in optimization problems (6) and (7) adds very little to computing distortions and gradients, where we reuse the earlier computation by letting $\boldsymbol{\alpha} = W\hat{\boldsymbol{\alpha}}$ and perform only one additional chain-rule step $\partial\boldsymbol{\alpha}/\partial\hat{\boldsymbol{\alpha}} = W$. By doing so, however, the total number of optimization variables in (6) or (7) is reduced from $|G| + 2$ to $|\hat{G}| + 2$. For example, if G is the canvas grid of an 28×28 image and $\hat{G} = \{0, 27\} \times \{0, 27\}$, the number of optimization variables reduces from 1570 to 10. It is important to note that using a simpler anchor grid is *not* the same as downsampling. If it were downsampling, one would plug in $\alpha \leftarrow \hat{\boldsymbol{\alpha}}$, but what we plug in is $\alpha \leftarrow W\hat{\boldsymbol{\alpha}}$. In our case, image colors are still sampled from the underlying grid rather than downsampled from the anchor grid (in fact, one may even use an underlying grid that is denser than the original image grid). As

a result, using our anchor system does not lose information from the image, and still benefits from the reduced optimization size. Running gradient descent (with respect to anchors) in the abstracted optimization space effectively avoids getting stuck at critical points.

Blurring. Another view to lifting gradient descent to a high-level, abstracted optimization space, is to blur the image. Intuitively, blurring ignores low-level fluctuation, like how we naturally abstract an image. As a result, blurring helps alleviate the vanishing-gradient problem, and it can be easily done in our image smoothing process at the beginning. The cutoff radius ρ_c in the kernel κ in (2) controls the blurring extent: a larger ρ_c refers to a slower decay and thus, a more blurred image.

Guided gradient descent. Putting the two abstraction techniques together yields our guided gradient descent, starting from higher-level abstractions to lower-level ones. Given an anchor grid $\hat{G}$ and a cutoff radius ρ_c, we denote the two optimization problems (6) and (7) by $DC(\hat{G}, \rho_c)$ and $DV(\hat{G}, \rho_c)$, respectively. For either problem, we solve for a $(\hat{G}, \rho_c)$-solution path, starting from smaller $\hat{G}$ and larger ρ_c then gradually refining $\hat{G}$ and decreasing ρ_c. Let $\hat{G}_k$ denote a $k \times k$ anchor grid that distributes anchors most evenly across the underlying grid G and let $\hat{L}_k$ denote the corresponding anchor lattice of $\hat{G}_k$. Figure 3C illustrates a chain of anchor lattices $\{\hat{L}_{3^i+1}\}_{i=0,1,2,\ldots}$ starting from the $\hat{G}_2$ grid, and a chain of cutoff radii $\{\eta^j \rho_{c_0}\}_{j=0,1,2,\ldots}$ starting from, e.g., ρ_{c_0} being $1/3$ of the original image size. Initially, it is easy to align two largely blurred blobs via a small canvas adjustment, implying only a small number of iterations needed to converge to $\mathcal{D}_C \approx \mathcal{D}_V \approx 0$. As we proceed along the solution path with larger $\hat{G}_k$ and smaller ρ_c, the image pattern gets more intricate but the finer anchor grid/lattice helps handle the increased intricacy. Moreover, in a solution path, an earlier solution is always used to warm-start the subsequent solve step, which further alleviates the curse of vanishing gradient by providing a good starting point. Notably, even the starting $\hat{G}_2$ grid consisting of only four anchors (i.e., the corners of the underlying grid) is very expressive, parameterizing a large family of transformations, including all affine transformations. Finer anchor grids/lattices can express all sorts of flexible transformations (including local, global, piecewise affine, *etc.*), which can approach human-level flexibility in transforming one image to another.

4 Image Classification in the Tiny Data Regime

We leverage the $\mathcal{D}_C$ / $\mathcal{D}_V$-distance learned from our distortable canvas model to classify grayscale images, where we simply use the Nearest-Neighbor classifier. As desired, the entire process including metric learning and classification is human-intuitive and human-interpretable (i.e., a white box). We name the whole model $\mathcal{D}_C$-Nearest-Neighbor, or $\mathcal{D}_V$-Nearest-Neighbor in its dual version, which reduces to vanilla Nearest-Neighbor under Euclidean distance. We show classification performances on three standard benchmarks: the MNIST and EMNIST datasets of handwritten digits and letters restricted to the tiny-data regime, as well as the Omniglot challenge.

MNIST in the tiny-data regime. The original MNIST benchmark has a total of 60K 28×28 grayscale images for training and 10K for testing, spanning 10 classes. To evaluate how a model performs in the tiny data regime, we train the model on the first N images per class from the original MNIST training set, test the model on the entire test set, and see how the test set accuracy varies with $N = 1, 2, 3, \ldots$. We run both versions of our model and compare them to both neural-network-based and classical ML models. These include the capsule-network variant TextCaps (Jayasundara et al., 2019) with the-state-of-art performance in the small-data regime ($98.68 \pm 0.30\%$ accuracy, trained from 200 images per class), as well as SVM, Nearest-Neighbor, etc. We include the classical ones to show that excelling in the tiny-data regime does not mean just using simple models. For any non-deterministic model that returns different results from the same setting (e.g., neural networks, random forests, decision trees), we do 5 independent runs and record the mean and standard deviation. Notably, TextCaps does not run when the (per-class) training size $N < 4$ and may occasionally return a random-guess accuracy around 10%. So, for TextCaps, we start with $N = 4$ and do 11 independent runs for each training size N, trim the best two and worst four, and record the trimmed mean and standard deviation (so in reality, TextCaps is more unstable than our reported statistics). We also directly include results from other references that experimented with MNIST in a similar tiny-data setting, including a few-shot setting that uses extra data to pre-learn and transfer (whereas all our other selected models do not). These results are obtained under the same training-testing sizes but not the same training set or test set or both, and thus, serve as indirect comparisons (gray in Figure 4A).

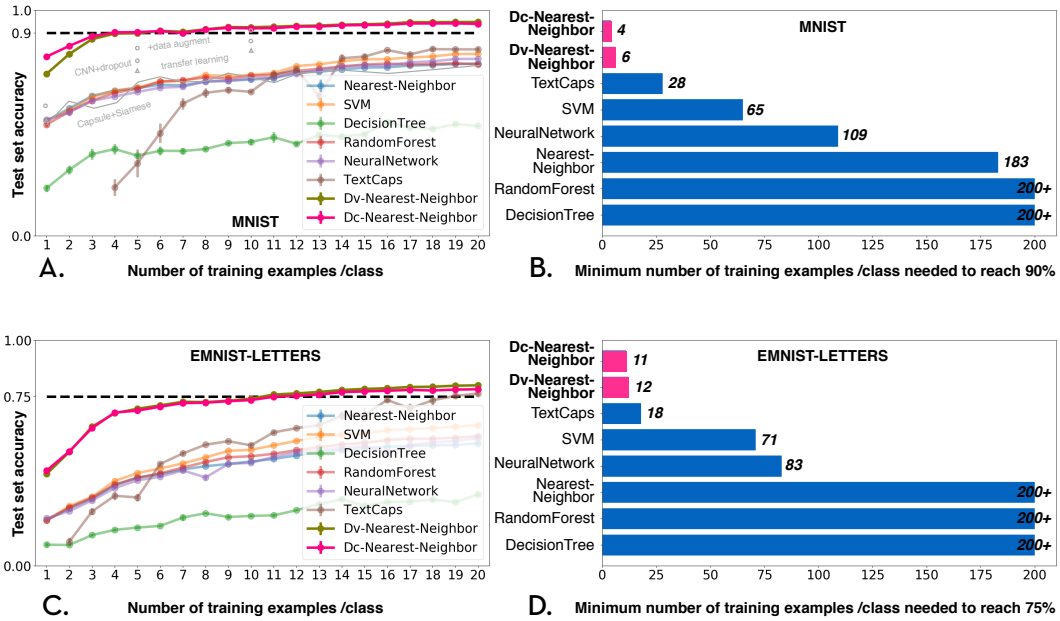
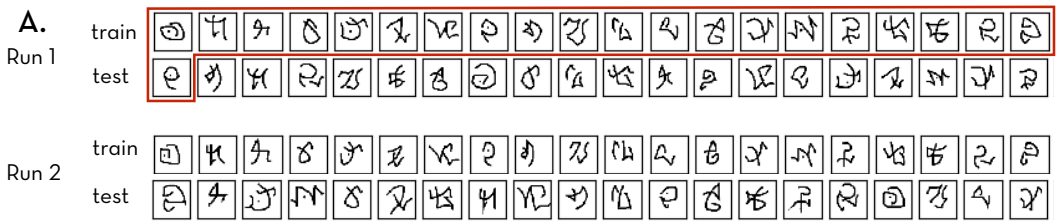


Figure 4: Model comparisons on MNIST/EMNIST-letters in the tiny-data regime.

For each model, we plot its test set accuracy versus the training size N in Figure 4A, and we also display the smallest size N needed to reach a threshold of 90% accuracy in Figure 4B. Both versions of our model outperform all other models at all values of N ranging from 1 to 20 training examples per class. Our model requires only the first four (or six in our other version) training examples per class to reach 90% accuracy, which is far less than what all the other models need.

EMNIST-letters in the tiny-data regime. The original EMNIST-letters benchmark comprises $4,800$ training images per class and 800 per class for testing, spanning 26 classes for the English letters (case-insensitive). The images are in the same format: 28×28, grayscale. We run the same experiment as in MNIST for EMNIST-letters (except for TextCaps being less unstable in this case: we do 7 independent runs per training size N and trim only the best and worst) and plot the results in Figures 4C and 4D. Like in MNIST, our model outperforms all other models at all values of training size N ranging from 1 to 20, and requires eleven (or twelve in our other version) training examples per class to reach 75% accuracy (due to the difficulty of this dataset), which is also less than what all other models need.

We note EMNIST-letters is harder, not only with more classes but also more intrinsic ambiguities, e.g., an *l* and an *I* can look identical, so can an *h* and an *n* when carelessly written. As a result, all models perform worse than in MNIST, which is why we lowered the target threshold from 90% to 75% in Figures 4C and 4D. The intrinsic ambiguity, together with possible labeling errors, narrows our superiority over other models when the training size increases. This is especially true for state-of-art deep learning models such as TextCaps, catching up quickly in Figures 4A and 4C when N grows larger. Being sensitive to ambiguities and outliers, however, is not a deficiency of our distortable canvas model, but a property of the Nearest-Neighbor method we used for classification. To improve, we may try integrating our model with more robust classifiers or a k-Nearest-Neighbor with proper voting. However, k-Nearest-Neighbor is not applicable in the tiny-data regime, not only because the training size may be about k but also there is very little room to further hold out a validation set from the tiny training set so as to select k. Ideally, adaptive k-Nearest-Neighbor is preferred, where k stays 1 in the tiny-data regime and becomes automatically tunable when an increased training size affords a held-out validation set. A closely-related issue is about picking a right model configuration but lacking validation data in the tiny-data regime. One may attain better results by trying new configurations for any selected model. Yet, it is not clear what strategy one may use. As such, for TextCaps, we directly used its original implementation and configuration; for the rest, we used their scikit-learn implementations with largely default configurations, except for tweaks to avoid triviality (e.g., a reasonable neural-network size) and those for the tiny-data setting (e.g., stronger



B.

		BPL	Humans	Ours	RCN	Siamese ConvNet	Simple ConvNet	Prototypical Net	VHE
Background set size	none			**6.75%**					
	smaller	4.2%					23.2%	30.1%	
	original	3.3%	4.5%		7.3%		13.5%	13.7%	18.7%
	augmented					8%			

Figure 5: One-shot classification in Omniglot challenge (A) and error-rate comparisons (B).

regularization). On the contrary, our distortable canvas model requires very little to tune, other than the $(\hat{G}, \rho_c)$-solution path. Theoretically speaking, the more gradual the path is, the better. We picked our $(\hat{G}, \rho_c)$ based on only the image size (28 in this case) and a reasonable runtime.

The Omniglot challenge for one-shot classification. The Omniglot dataset collects handwritten characters from 50 different alphabets, which include historical, present, and artificial scripts (e.g., Hebrew, Korean, Futurama) and thus, are far more complex than MNIST digits and EMNIST letters. The characters are 105×105 black-and-white images, together with the stroke-movement information. Unlike MNIST/EMNIST that comes with large training data, the Omniglot challenge was specifically designed to study human-level concept learning from small amount of data. In particular, its one-shot classification task was benchmarked to study how humans and machines can learn a new concept from just a single example. This benchmark contains 20 independent runs of 20-way within-alphabet classifications. The $(2k-1)$th and $(2k)$th runs for $k = 1, 2, \ldots, 10$ use the same set of 20 characters from a single alphabet. Each run contains 40 images in total: one training and one test image per character. The unit task there is to predict for each test image, which of the 20 characters it belongs to, based on the 20 training images. So, there is a total of 400 independent unit tasks across all the 20 runs. Figure 5A shows the first and second run in the benchmark, covering a total of 1 alphabet, 20 characters, and 80 distinct images. It also delineates a unit task consisting of 1 test and 10 training images via red bounding boxes.

The Omniglot benchmark adopted the standard one-shot learning setting, where it also provided a separate background set to pre-train one-shot learning models. The original background set contains 964 character classes from 30 alphabets; a smaller background set was proposed later to make the classification task more challenging. We run our $\mathcal{D}_C$-Nearest-Neighbor without any background set or any stroke-movement information. In other words, in each unit task, we predict the test image based on *one and only that* training image per character, reading all images from their raw pixels. Shown in Figure 5B, our model (with a 6.75% error rate) approaches human performance (4.5%) and outperforms all models as recorded in Lake et al. (2019), except for the BPL method developed specifically for the Omniglot Challenge. However, our model is more generic, applicable to a larger variety of images and learning tasks, e.g., doodle images for unsupervised learning, coming next.

5 Archetype Generation

Besides used with a classifier, our distortable canvas model can be extended to perform k-means-style clustering but under its human-intuitive topological similarity. Directly computing pairwise distances via our model and then running k-means is not realistic. Computing a distance in our case requires solving an optimization problem, which is not as cheap as computing an Euclidean distance. Moreover, computing a "centroid", or "averaging", in a non-Euclidean space requires solving another optimization problem (i.e., minimizing the sum of within-cluster distances to the centroid, so, an optimization problem containing multiple optimization problems), which is not as simple as an arithmetic mean. However, our setting of transformation flow between two images allows us to extend

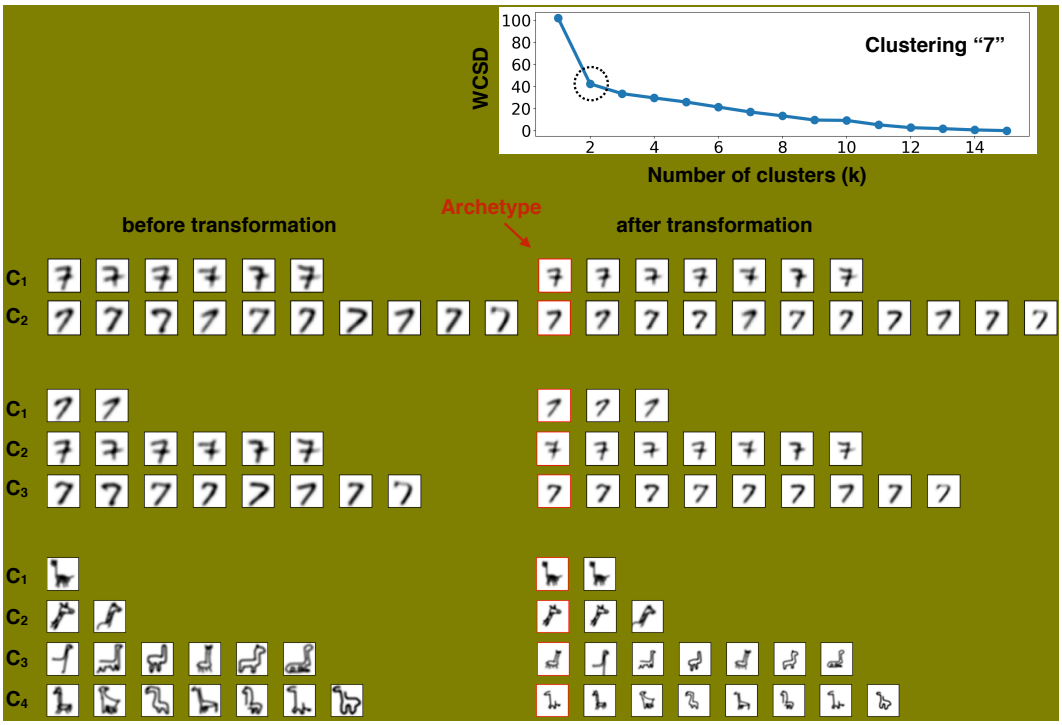


Figure 6: Archetype generation.

it to multi-flows among multiple images. Under the multi-flow extension, we do not individually compute any pairwise distance to the end, i.e., we do not solve the inner optimizations first. Instead, we solve the inner and outer optimizations at the same time, turning the nested optimization into a single flat one. Formally, given N images $\mathcal{M}_1, \ldots, \mathcal{M}_N$, to cluster them into K clusters, we solve

$$\min_{\substack{\alpha_1,\ldots,\alpha_N \\ \overline{\alpha}_1,\ldots,\overline{\alpha}_K \\ C_1,\ldots,C_K}} \sum_{k=1}^{K} \sum_{i \in C_k} \mathcal{D}_C(\overline{\mathcal{M}}_k \circ \overline{\alpha}_k,\ \mathcal{M}_i \circ \alpha_i) \quad \text{s.t.} \sum_{i=1}^{N} \mathcal{D}_V(\alpha_i) \leq \epsilon, \tag{9}$$

where C_k denotes the kth cluster, $\overline{\mathcal{M}}_k \circ \overline{\alpha}_k$ denotes the transformed image representing the kth centroid, and $\mathcal{M}_i \circ \alpha_i$ denotes the ith transformed image flowing to its corresponding centroid together with all other $N-1$ transformed images. One can check (9) is an extension of (6) where we omitted χ for simplicity. Solving (9) is similar to k-means via the alternating refinement technique: the assignment step assigns each transformed image $\mathcal{M}_i \circ \alpha_i$ to $C_{k^\star}$ according to $k^\star = \arg\min_{k=1,\ldots,K} \mathcal{D}_C(\overline{\mathcal{M}}_k \circ \overline{\alpha}_k,\ \mathcal{M}_i \circ \alpha_i)$; the update step solves (9) for one gradient-descent step given the C_k's. Upon convergence, we obtain $C_1^\star, \ldots, C_K^\star$ as the clusters and $\overline{\mathcal{M}}_1 \circ \overline{\alpha}_1, \ldots, \overline{\mathcal{M}}_K \circ \overline{\alpha}_K$ as the centroids. We treat the learned centroids as *archetypes* of the N images given at the beginning.

We run the above method to generate archetypes of a particular image class, sharing many common practices with k-means such as trying different values of $k = 1, 2, \ldots$. For each k, we try multiple random starts and record the best within-cluster sum of distances (WCSD). We then use the elbow method to pick good values for k. Figure 6 shows the WCSD-versus-k curve obtained by running our clustering method on a set of 16 images of "7"s from the MNIST dataset. The curve indicates $k = 2$ as a potential elbow point. The two clusters of "7" coincide with human initiations on have two general ways of writing "7' depending on whether there is an extra stroke. In this case, the three clusters also make sense, further dividing the cluster of "simple 7"s based on the angle. Moving away from more strictly defined symbol systems towards arts, we try our model on doodle data where people can freely or even widely draw abstract figures of real-world objects. The bottom clustering in Figure 6 reveals four ways of doodling a giraffe, learned from the first 16 giraffes taken from Google's Quick Draw dataset. One can see the four learned archetypes cleanly separate outline sketch and detailed drawing, pose orientations, as well as zoomed-in views of the neck.

REFERENCES

Amina Adadi and Mohammed Berrada. Peeking inside the black-box: A survey on explainable artificial intelligence (XAI). *IEEE Access*, 6:52138–52160, 2018.

Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Process. Mag.*, 34(4):18–42, 2017.

François Chollet. On the measure of intelligence. arXiv:1911.01547v2 [cs.AI], 2019.

Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre van Schaik. EMNIST: An extension of MNIST to handwritten letters. In *Proceedings of the 2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 2921–2926, May 2017.

Vinoj Jayasundara, Sandaru Jayasekara, Hirunima Jayasekara, Jathushan Rajasegaran, Suranga Seneviratne, and Ranga Rodrigo. TextCaps: Handwritten character recognition with very small datasets. In *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, pp. 254–262, 2019.

E. T. Jaynes. Information theory and statistical mechanics. *Physical Review*, 106(4):620–630, May 1957.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances Neural Inf. Process. Syst.*, 25:1097–1105, 2012.

Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

Brenden M. Lake, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. The omniglot challenge: a 3-year progress report. *Current Opinion in Behavioral Sciences*, 29:97–104, October 2019.

Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998.

David G Lowe. Object recognition from local scale-invariant features. In *Proc. 7th IEEE Int. Conf. Computer Vision*, volume 2, pp. 1150–1157, 1999.

Diego A. Mesa, Justin Tantiongloc, Marcela Mendoza, Sanggyun Kim, and Todd P. Coleman. A distributed framework for the construction of transport maps. *Neural Computation*, 31(4):613–652, April 2019. doi: 10.1162/neco_a_01172.

Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Trans. Knowl. Data Eng.*, 22 (10):1345–1359, 2009.

Danilo Rezende, Shakir Mohamed, Ivo Danihelka, Karol Gregor, and Daan Wierstra. One-shot generalization in deep generative models. In *Proc. Int. Conf. Mach. Learn.*, pp. 1521–1529, 2016.

Ruslan Salakhutdinov, Joshua Tenenbaum, and Antonio Torralba. One-shot learning with a hierarchical nonparametric bayesian model. In *Proc. 2012 ICML Workshop Unsuperv. Transf. Learn. (UTL 2012)*, pp. 195–206, 2012.

Elizabeth S Spelke and Katherine D Kinzler. Core knowledge. *Developmental Sci.*, 10(1):89–96, 2007.

Amos Storkey. When training and test sets are different: characterizing learning transfer. *Dataset Shift Mach. Learn.*, 30:3–28, 2009.

Cedric Villani. *Optimal Transport: Old and New*. Springer, 2009.