A model is worth tens of thousands of examples for estimation and thousands for classification

Thomas Dagès, Laurent D. Cohen, Alfred M Bruckstein

A model is worth tens of thousands of examples for estimation and thousands for classification

Thomas Dagès*[1], Laurent D. Cohen[2], and Alfred M. Bruckstein[1]

[1]Department of Computer Science, Technion Israel Institute of Technology
[2]Ceremade, University Paris Dauphine, PSL Research University, UMR CNRS 7534

Abstract

Traditional signal processing methods relying on mathematical data generation models have been cast aside in favour of deep neural networks, which require vast amounts of data. Since the theoretical sample complexity is nearly impossible to evaluate, these amounts of examples are usually estimated with crude rules of thumb. However, these rules only suggest when the networks should work, but do not relate to the traditional methods. In particular, an interesting question is: how much data is required for neural networks to be on par or outperform, if possible, the traditional model-based methods? In this work, we empirically investigate this question in three simple examples covering estimation and classification, where the data is generated according to precisely defined mathematical models, and where well-understood optimal or state-of-the-art mathematical data-agnostic solutions are known. A first problem is deconvolving one-dimensional Gaussian signals, a second one is estimating a circle's radius and location in random grayscale images of disks, and a third one both classifies the presence of a line and locates it when present in a binary random dot image. By training various networks, either naive custom designed or well-established ones, with various amounts of training data, we find that networks require tens of thousands of examples for estimation in comparison to the traditional methods and thousands for classification, whether the networks are trained from scratch or even with transfer-learning or finetuning.

1 Introduction

Neural network-based machine learning has widely replaced the traditional methods for solving many signal and image processing tasks that relied on mathematical models for the data [1, 2]. In some cases, the assumed models provided ways to optimally address the tasks at hand and resulted in well-performing estimation and prediction methods with theoretical guarantees [3, 4, 5]. Nowadays, gathering raw data and applying gradient descent-like processes to neural network structures [6, 7, 8, 9] largely replaced modelling and mathematically developing provably optimal solutions.

It is commonly accepted that, if the networks are complex enough and when vast amounts of data are available, neural networks outperform traditionally designed methods [10, 7] or even humans [11, 12, 13, 14]. In many real-world applications, the availability of vast amounts of data is often limited by factors such as cost, privacy concerns, or logistical challenges. This makes it critical for the field to understand the data requirements for neural networks to achieve high performance, especially as these networks are increasingly applied in data-limited domains like healthcare and finance, where accurately estimating the minimum data needed for successful training is becoming ever more important. The required amount of data is called in statistical learning theory the sample complexity and is related to the input-output information relation [15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28] and to the VC-dimension of the problem [29, 30, 31, 32, 33, 34], which is usually intractable for non trivial networks [35]. Instead, various rules of thumb have been used in the field to guess how many samples are needed: at least 10-50 times the number of parameters [36], at least 10 times per class in classification (and 50 times in regression) the data dimensionality [37] and at least 50-1000 times the output dimension [36].

However, these rules only suggest how much data is needed to get a "good" network, but they do not relate to the traditional data-generation model-based methods. A natural question hence arises: do the neural network-based solutions perform as well as, or even outperform, the processing methods based on traditional data-generation models when lots of data is available, and if so how much data is necessary? We address this question in three simple empirical examples, where the data is produced according to precisely defined models, and where well understood optimal or state-of-the-art mathematical solutions are available. The first is the deconvolution of Gaussian signals, optimally solved with the Wiener filter [3]. The second is the estimation of the radius and centre coordinates

*thomas.dages@cs.technion.ac.il
This is a post peer-review, pre-copyedit version of an article published in Pattern Recognition. The final authenticated version is available online at: https://doi.org/10.1016/j.patcog.2024.110904

of a disk in an image, which can be elegantly solved using a Pointflow method [38]. The third is the classification of a random dot image to have a significant line in it based on the estimation of position of the candidate line, which is classically achieved by the Hough transform [39, 40]. This work aids engineers to decide when to use model-based classical methods or simply feed lots of data (if available) to deep neural networks. This paper extends our previous study [41].

Section 2 presents our comparison for the one-dimensional signal recovery, Section 3 deals with estimation of disk characteristics in an image, and Section 4 tackles the classification of binary random dot images to possess a significant line and, when present, estimate its location.

2 One-dimensional signal recovery

We suggest to first analyse a simple and well-understood problem in the one-dimensional case where the optimal solution is provingly known.

2.1 Data model and optimal solution

The original data consists of real random vectors φ of size D that are centred, i.e. $\mathbb{E}(\varphi) = 0$, and with known autocorrelation $R_\varphi = \mathbb{E}(\varphi\varphi^\top) \in \mathbb{R}^{D\times D}$. However, φ is degraded by blur and noise producing the observed data φ^{data} as follows:

$$\varphi^{data} = H\varphi + n, \tag{1}$$

where $H \in \mathbb{R}^{D\times D}$ is a known deterministic matrix and n is random additive noise independent from φ that is centred $\mathbb{E}(n) = 0$ and with known autocorrelation matrix $R_n \in \mathbb{R}^{D\times D}$. It is well-known [3] that the best linear recovery of φ in the L^2 sense, i.e. minimising the Expected Squared Error (ESE) $ESE(\hat{\varphi}, \varphi) = \mathbb{E}\left(\|\hat{\varphi} - \varphi\|_2^2\right)$ with respect to the matrix $M \in \mathbb{R}^{D\times D}$ such that $\hat{\varphi} = M\varphi^{data}$, is given by applying the Wiener filter $W = R_\varphi H^\top (HR_\varphi H^\top + R_n)^{-1}$, i.e. $\hat{\varphi}^* = W\varphi^{data}$. Moreover, if we further assume both $\varphi \sim \mathcal{N}(0, R_\varphi)$ and $n \sim \mathcal{N}(0, R_n)$ are Gaussian, then the Wiener filter W minimises the ESE over all possible recoveries including nonlinear ones. Furthermore, note that if R_φ is circulant, i.e. φ is cyclostationary, and so is n, e.g. if n has independent entries implying R_n is diagonal, and if H is circulant, then W is also circulant and Wiener filtering is a pointwise multiplication in the Fourier domain given by applying the unitary Discrete Fourier Transform $[DFT]$ with (k,l)-th entry $[DFT]_{k,l} = \frac{1}{\sqrt{D}} e^{-i\frac{2\pi kl}{D}}$.

In our tests, the dimensionality is $D = 32$ and the problem is circulant. We use an interpretable symmetric positive-definite autocorrelation matrix R_φ parameterised by a large number $\rho = 0.95$ to create high spatial correlation over a large support decaying with distance and H is a local smoothing convolution. The first lines of R_φ and H are $\begin{pmatrix}1 & \rho & \rho^2 & \rho^3 & \cdots & \cdots & \rho^3 & \rho^2 & \rho\end{pmatrix}$ and $\begin{pmatrix}1 & 1 & 0 & \text{——} & 0 & 1\end{pmatrix}$. The noise is i.i.d. $n \sim \mathcal{N}(0, \sigma_n^2 I)$ with $\sigma_n = 0.1$. We display example data in Figure 1, the designed H, R_φ, and R_n along with their associated Wiener filter W in Figure 2.

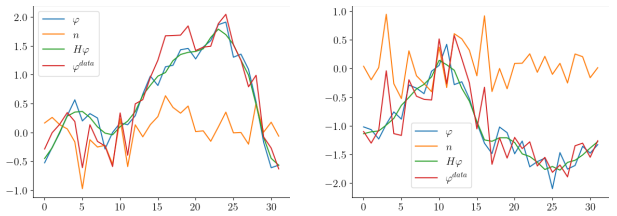


Figure 1: Two example signals φ^{data}, with their associated blur $H\varphi$ and noise n.

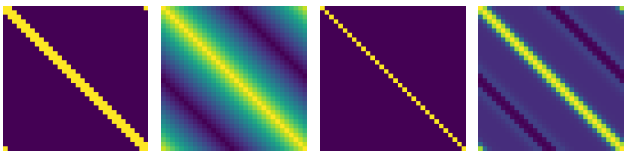


Figure 2: Chosen model matrices and associated optimal Wiener filter.

2.2 Neural models

We wish to evaluate the capabilities of neural networks by comparing them to humanly designed methods by classical experts using no training. Our criterion is the amount of random training samples N needed to reach or overtake human expertise. Working in the Gaussian case for the data model of equation (1), we create various random training datasets containing N data samples ranging in $N \in \{10, 100, 1000, 10000, 100000\}$. We train a variety of small Convolutional Neural Networks (CNNs) of various depths $k \in \{0, 1, 2, 3\}$. The depth of the network is measured as the number of successions of convolution-pointwise-nonlinearity layers. Each network ends with a final fully connected layer A (with bias b_A), i.e. a final unconstrained affine transformation. For simplicity, our CNNs will be single-channel only and without various architecture tricks, e.g. dropout, batch normalisation, or pooling. The network functions, denoted f_k for $k \in \{1, \ldots, K\}$ can thus be written as:

$$f_k(\varphi^{data}) = A\sigma \circ \tilde{C}_k \circ \sigma \circ \tilde{C}_{k-1} \circ \cdots \circ \sigma \circ \tilde{C}_1(\varphi^{data}) + b_A, \tag{2}$$

where $\tilde{C}_i(x) = C_i x + b_i$ is the i-th convolution layer comprising the circulant matrix C_i for the convolution and its additive unconstrained bias b_i and $\sigma = \text{ReLU}$ the standard pointwise nonlinearity in neural networks. Note that a CNN with depth 0 degenerates to an unconstrained affine transformation in $\mathbb{R}^D$ (no pointwise nonlinearity or convolution): $f_0(\varphi^{data}) = A\varphi^{data} + b_A$.

The networks are trained to minimise the Mean Squared Error (MSE)[1], a proxy for the ESE, using the N generated samples. Denoting $f_{k,N,\eta}$ the resulting networks (where η a hyperparameter of the optimisation algorithm), we have:

$$MSE_{train}(f_{k,N,\eta}) = \frac{1}{N}\sum_{i=1}^{N} \|f_{k,N,\eta}(\varphi^{data}_{train,i}) - \varphi_{train,i}\|_2^2, \tag{3}$$

where for a sample collection set, $\varphi_{set,i}$ and $\varphi^{data}_{set,i}$ denote the i-th original and degraded samples. This quantity is to be compared with $ESE(f_{k,N}(\varphi^{data}), \varphi)$, which evaluates the performance on all possible data of a network trained on N instances only. Naturally, this quantity cannot be computed by hand and is approximated by another MSE calculation on a large test set using N_t test samples independently generated from the training ones:

$$\begin{aligned} MSE_{test}(f_{k,N,\eta}) &= \frac{1}{N_t}\sum_{i=1}^{N_t} \|f_{k,N,\eta}(\varphi^{data}_{test,i}) - \varphi_{test,i}\|_2^2 \\ &\xrightarrow[N_t\to\infty]{} ESE(f_{k,N,\eta}(\varphi^{data}), \varphi). \end{aligned} \tag{4}$$

In our tests, $N_t = 100000$. Note that implicitly in $ESE(f_{k,N,\eta}(\varphi^{data}), \varphi)$ the network $f_{k,N,\eta}$ is the given result of a minimisation algorithm. For randomised algorithms, it is thus to be understood as the expectation conditional to the learned network $f_{k,N,\eta}$: $ESE(f_{k,N,\eta}(\varphi^{data}), \varphi) = \mathbb{E}(\|f_{k,N,\eta}(\varphi^{data}) - \varphi\|_2^2 \mid f_{k,N,\eta})$.

We train our networks using Stochastic Gradient Descent with Nesterov momentum parameter equal to 0.9. We train the networks using various learning rates $\eta \in \{0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1\}$ over 50 epochs, performing $N_r = 50$ independent training trials per learning rate, and compute the final median performance per learning rate on a validation set generated independently of the train and test data comprising $N_v = 100000$ validation samples:

$$\begin{aligned} MSE_{val}(f_{k,N,\eta}) &= \frac{1}{N_v}\sum_{i=1}^{N_v} \|f_{k,N,\eta}(\varphi^{data}_{val,i}) - \varphi_{val,i}\|_2^2 \\ &\xrightarrow[N_v\to\infty]{} ESE(f_{k,N,\eta}(\varphi^{data}), \varphi). \end{aligned} \tag{5}$$

The validation set is used to choose the best learning rate for each amount of training data $\eta^*(N)$ by taking:

$$\eta^*(N) = \operatorname*{argmin}_{\eta} MEDIAN_r(MSE_{val}(f_{k,N,\eta})), \tag{6}$$

where $MEDIAN_r$ takes the median over the $r \leq N_r$ best independent runs on the validation set per η. Given that a significant amount of runs do not converge or get trapped early in a poor local minimum depending on the random initialisation, choosing $r \ll N_r$ ensures that only the networks finding a good local minimum are considered. The final performance of CNNs $SCORE_{k,r}(N)$ for each amount of data N is then the median of the test performance over those selected r trials[2] of the final test score at the chosen learning rate $\eta^*(N)$:

$$SCORE_{k,r}(N) = MEDIAN_r(MSE_{test}(f_{k,N,\eta^*(N)})). \tag{7}$$

See Table 1 for a summary of our training procedure and hyperparameters.

We display the evolution of the networks' performance on the amount of training data N in Figure 3 for each depth k, with detailed scores in Table 2, along with the performance of the Wiener filter. Regardless of N, the Wiener filter outperforms the neural models as expected by the theory, but their performance converges to the Wiener's one when a lot of data is available, with similar performance when at least 10000 training samples are available. We can thus consider this study as providing a criterion that a model would be preferable for estimation if data is limited to fewer than 10000 samples to train on.

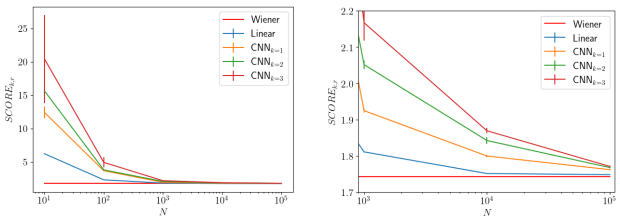


Figure 3: Median test scores for CNNs with depth $k \in \{0, 1, 2, 3\}$ on $r = 10$ selected runs ($k = 0$ is just a linear layer). Vertical bars represent the standard deviation of the MSE of these runs. The right figure is a zoomed-in plot of the left one for large N.

3 Two-dimensional geometric estimation

We next analyse a more complicated yet well-understood problem based on Euclidean geometry. The goal is to estimate basic geometric properties on simple data: the radius and centre location of a random disk in an image. It was shown in [42] that this seemingly trivial task is more complex than expected for neural models even when focusing on radius estimation of centred disks.

3.1 Data model

The original data now consists of $D \times D$ random two-dimensional grayscale images of disks. Images are centred at $(0, 0)$, and for a pixel $p \in [-\frac{D-1}{2}, \frac{D-1}{2}]^2$:

$$\varphi(p) = \begin{cases} b & \text{if } \|p - c\|_2 > r \\ f & \text{if } \|p - c\|_2 \leq r, \end{cases} \tag{8}$$

[1] The loss function is actually scaled to $\frac{1}{D}MSE_{train}$ as is commonly done in practice.

[2] Selected on the validation set.

	1D estimation	2D estimation	2D classification
CNN architecture	Custom	{AlexNet, VGG11, ResNet18}	{AlexNet, VGG11, ResNet18}
k	$\{0, 1, 2, 3\}$	—	—
N_r	50	1	1
r	10	1	1
η	{0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1}	{0.000001, 0.00005, 0.00001, 0.0005, 0.0001, 0.005, 0.001, 0.05, 0.01, 0.5, 0.1}	{0.000001, 0.00005, 0.00001, 0.0005, 0.0001, 0.005, 0.001, 0.05, 0.01, 0.5, 0.1}
Batch size	10	10	10
Epochs	50	50	50
N	$\{10, 100, 1000, 10000, 100000\}$	$\{10, 100, 1000, 10000, 100000\}$	$\{10, 100, 1000, 10000, 100000\}$
N_v	100000	100000	100000
N_t	100000	100000	100000
Loss	*MSE*	*MSE*	$BCE + MSGE + \lambda_{reg} L_{reg}$

Table 1: Summary of training procedures and hyperparameters for all CNNs in this work.

N	0	10	100	1000	10000	100000
Wiener	**1.743**	—	—	—	—	—
Linear ($k = 0$)	—	6.204	2.295	1.811	1.751	1.748
CNN ($k = 1$)	—	12.386	3.662	1.924	1.799	1.762
CNN ($k = 2$)	—	15.614	3.789	2.051	1.842	1.767
CNN ($k = 3$)	—	20.395	4.911	2.167	1.869	1.771

Table 2: Median MSE scores $SCORE_{k,r}$ of the CNNs on $r = 10$ selected runs, compared to the theoretically optimal Wiener filter.

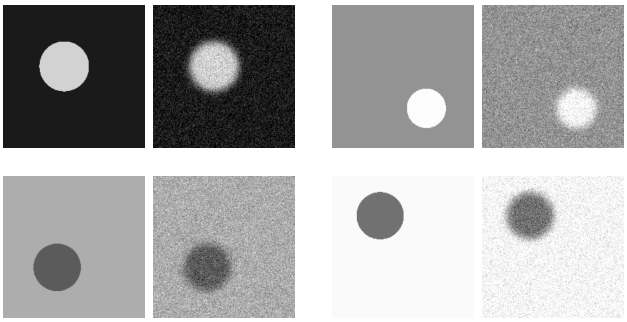


Figure 4: Four examples of clean φ and degraded φ^{data} disk images.

where r is the circle's radius, $c = (c_x, c_y)$ its centre, and f (resp. b) is the foreground (resp. background) intensity. These parameters are independently[3] and uniformly chosen at random: $r \sim \mathcal{U}([\frac{\varepsilon_r}{2}\frac{D-1}{4}, (1-\frac{\varepsilon_r}{2})\frac{D-1}{4}])$ with $\varepsilon_r = 0.4$, $c \sim \mathcal{U}([(D-1)\frac{\varepsilon_c}{2} - \frac{D-1}{2}, (D-1)(1-\frac{\varepsilon_c}{2}) - \frac{D-1}{2}]^2)$ with $\varepsilon_c = 0.5$, $b \sim \mathcal{U}([0, 1])$, and $f \mid b \sim \mathcal{U}([0, 1] \setminus [b - \delta, b + \delta])$ with $\delta = \frac{50}{255}$ the minimum contrast[4]. However, φ is degraded with blur and noise giving the observed data φ^{data} as follows:

$$\varphi^{data} = g_{\sigma_b} * \varphi + n, \tag{9}$$

where $g_{\sigma_b}(p) = \frac{1}{2\pi}\exp(-\frac{\|p\|_2^2}{2\sigma_b^2})$ is a Gaussian convolution kernel, and n is i.i.d. white noise $n \sim \mathcal{N}(0, \sigma_n I_{D^2})$. We plot example data in Figure 4. The task is to estimate the three geometric numbers $(r, c) = (r, c_x, c_y)$ from φ^{data}.

3.2 Expert engineer's solution

Unlike in the Wiener case, the optimal estimator minimising the ESE is not so trivial to find. Instead, we choose a method called Pointflow designed by an expert engineer that perfectly tackles the problem at hand.

Pointflow [38, 43] is an elegant subpixel level contour integrator and edge detector in images requiring no learning whatsoever. It consists in defining potential vector fields V along which random points P flow: $\frac{dP}{dt}(t) = V(P(t))$, such that end trajectories lie on edges of the image I. The vanilla Pointflow [38] uses two fields V_+ and V_- from the edge attraction V_a and rotating V_r fields based on the image gradients as follows:

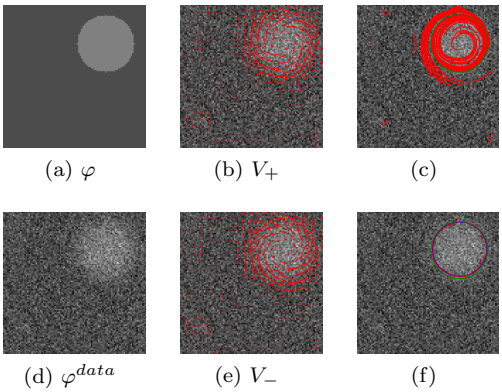
$$V_a = \nabla\|\nabla I_b\|_2, \quad V_r = \nabla I_b^{\perp}, \quad V_\pm = \tfrac{1}{2}(V_a \pm V_r), \tag{10}$$

where $I_b = g_{\sigma_{Pf}} * I$ is a blurred version of I with a Gaussian kernel $g_{\sigma_{Pf}}$. Various stopping conditions and uses of $V_\pm$ exist to detect edges in natural images, however on our data containing a single circular edge per image, we need only consider the basic ones. Indeed, the possible cases for trajectories are: it loops (C_l), it leaves the image domain (C_o), or it is stuck in an area with small magnitude $\|V\|_2$ (C_s). Flowing initially from V_+, if we loop (C_l), then the point has reached the circle and it suffices to reflow along V_+ to extract just the circle's contour. If we end up outside the image domain (C_o), which is rare in our data, we reflow from the exit point along V_-. If we are in a low flow magnitude area (C_s), which is not rare, then we discard the trajectory. In total, $N_{Pf} = 200$ points are randomly uniformly sampled in the image domain and used for Pointflow, and a fraction of them end up flowing on the disk's edge with subpixel precision, as the other ones lead to discarded trajectories. For more details on our implementation of Pointflow see A. For some illustrations of Pointflow results on our data see Figure 5.

To estimate the disk's radius and centre from pointflow contours (C_l), we can simply compute the average length of the reflown closed contours and divide

[3] Except f and b which are slightly correlated to ensure a minimal contrast $|f - b| > \delta$.

[4] In our tests, we take $D = 201$ implying that $r \sim \mathcal{U}([10, 40])$ and $c \sim \mathcal{U}([-50, 50]^2)$.



(a) φ (b) V_+ (c)

(d) φ^{data} (e) V_- (f)

Figure 5: Pointflow in practice. (5a): clean data. (5d): degraded data. (5b) and (5e): pointflow fields sampled every five pixels. (5c): initial flows of points without reflowing with groundtruth boundary in green. (5f): all final trajectories that have reflown in a closed loop (C_l) with groundtruth boundary in green and all regressed circles using least squares on each looped trajectory in blue (used for estimating the centre).

it by 2π. To compute the centre's coordinates, we could compute for each closed trajectory the average of its points, and then average over these estimations. However, this method empirically did not best perform on validation data, so we refined it by applying least-squares regression on the equation of a circle to estimate from it its location per trajectory and then average the estimations. Note that the least-squares regression did not provide a better estimation of the radius so we keep the crude length integration strategy for it.

3.3 Neural models

As in the one-dimensional case, the expert's method is to be compared with a convolutional neural network. Although the learning problem seems trivial, it is actually harder than expected for networks, as has been shown in [42] even when the circles are centred. Empirically, we were not able to train correctly a small custom model similar to those previously used having just three layers and even many channels per layers and no further deep learning tricks. To overcome this limitation, we use famous networks in the deep learning literature: the CNNs Alexnet [7], VGG [8] and ResNet [9], and the transformer VIT[5] [44]. To adapt the model to our task, we change the final fully connected layer to have 3 outputs only.

For each architecture, we either train the networks from scratch (SC), or initialise the weights, except those of the final fully connected layers, to those publicly available obtained by classification on Imagenet [45], as is commonly done in the field. The pretrained weights can be either frozen for transfer-learning (TL) or retrained as well for finetuning (FT). Although the task and data are fundamentally different from ours, it is generally believed that the wide variety of natural images encourages the famous networks to learn features that generalise quite well to most reasonable tasks.

Once again, the *MSE* loss is used for training[6]. As it is significantly more expensive to train such networks compared to the tiny ones in the Wiener case, we only perform $N_r = 1$ run per learning rate configuration, ranging in $\eta \in \{0.000001, 0.00005, 0.00001, 0.0005, 0.0001, 0.005, 0.001, 0.05, 0.01, 0.5, 0.1\}$, with a batch size of 10 for 50 epochs. As previously, the optimal learning rate is chosen on the performance on validation data. Both the test and validation data use $N_v = N_t = 100000$ independently randomly sampled images, whereas the training sets have $N \in \{10, 100, 1000, 10000, 100000\}$ ones. See Table 1 for a summary of our training procedure and choice of hyperparameters.

3.4 Results

We present the results in Figure 6 and Table 3. Although the networks are simultaneously trained for both the radius and centre location estimation, we also present the MSE on the estimation of each geometric concepts separately.

First, the transfer-learning network ResNet-TL is not able to correctly estimate the radius or centre's location, meaning that its learned features on classification of Imagenet are not able to handle our simple data: they are not so general after all. Likewise, the prelearned features of Alexnet-TL, VGG-TL, and VIT-TL do not generalise well to this toy problem, requiring significantly more data than the maximum available to compare with the simple data-agnostic pointflow. However, when the networks are entirely trained, either finetuned or from scratch, they are either flatly beaten by pointflow when using small amounts of training data or on par or slightly outperform it when $N \geq 10000$. The only networks beating pointflow overall are VGG-SC, ResNet-FT, and VIT-FT when $N = 100000$, but more (VGG-FT, VGG-SC, ResNet-FT, ResNet-SC, and VIT-SC) significantly outperform pointflow on the radius estimation when $N \geq 10000$. The difference between finetuning and training from scratch seems to only appear when small amounts of training data are available, and then finetuning is better. However, in these cases, both approaches pale in comparison to the reference data-agnostic method.

From this experiment, we conclude that a realistic neural network is worth at least tens of thousands of examples on a fairly simple task (toy problem vs real world challenge) compared to an expert engineer. We thus provide a criterion that a data model is preferable

[5] We use the simplest ones VGG11, ResNet18, and VIT_B_16, as larger ones are here unnecessary.

[6] To help the networks converge, the radius and centre coordinates are scaled to $[-1, 1]$ using $r_s = \frac{8}{D-1}(r - \frac{D-1}{8})$ and $c_s = \frac{2}{D-1}c$. In all plots and numbers provided in this paper, the results are rescaled to the original scale: r and c and not r_s and c_s.

for estimation if fewer than 10000 training samples are available.

	N	Pointflow	AlexNet			VGG			ResNet			VIT		
			TL	FT	SC	TL	FT	SC	TL	FT	SC	TL	FT	SC
(r,c)	0	**0.66**												
	10		826	524	1748	1581	754	1748	5256	3358	2590	2257	2282	2099
	100		314	66	291	448	69	215	2064	1124	793	1183	694	1799
	1000		183	23	9.0	189	6.5	7.6	1092	48	45	669	18	1788
	10000		140	31	4.8	71	2.9	1.2	814	3.7	4.4	289	1.0	1681
	100000		67	17	1.5	40	0.68	**0.42**	826	**0.51**	0.93	213	**0.21**	5.6
r	0	**0.26**												
	10		66	65	75	75	69	75	110	86	82	30	21	85
	100		36	5.9	42	52	5.6	40	42	25	31	5.3	6.9	70
	1000		26	2.4	1.3	20	0.57	0.62	24	2.2	1.3	1.7	0.58	28
	10000		16	2.0	1.5	9.9	**0.23**	**0.11**	19	**0.19**	0.27	0.95	**0.065**	0.72
	100000		6.8	1.7	0.69	4.9	**0.075**	**0.051**	19	**0.066**	**0.078**	0.73	**0.034**	**0.12**
c	0	**0.40**												
	10		759	454	1673	1506	675	1673	5142	3206	2507	2195	2282	2001
	100		277	60	249	393	63	175	2020	1099	762	1178	687	1724
	1000		157	21	7.7	169	5.9	6.9	1067	46	43	667	18	1717
	10000		118	29	3.0	61	2.6	1.1	795	3.5	4.1	288	0.96	1680
	100000		57	6.4	0.79	36	0.57	**0.37**	807	0.45	0.86	212	**0.17**	5.5

Table 3: MSE scores of the networks compared to Pointflow on test data, computed on both r and c (top), just r (middle), or just c (bottom). Networks were trained on joint prediction of r and c. For the separate r (resp. c) scores, the selected networks were those providing the best r (resp. c) error on validation data.

4 Two-dimensional geometric classification

Most traditional prediction problems fall in one of the following categories: estimation – also known as regression – and classification. Both previously studied problems belong to estimation, where precise numbers need to be predicted. The problem we next analyse is a classification task: what primarily matters is the accuracy of the class attribution scheme rather than the precision of the numbers used to determine it. It is commonly believed that classification is easier than regression.

Classification methods, both traditional and neural, usually follow the same scheme: find an embedding that (approximately) linearly separates the data and then perform linear classification for the class attribution. While linear classification is well-known, understanding the good non unique embedding strategies is more complex. For this reason, the problem we analyse next is once again simple and based on geometric data. To control the embeddings used for classification, we also study a well-understood geometric estimation problem. The main goal is thus to classify whether a random dot image stems from a line image or is pure white noise. The auxiliary estimation task consists in locating the line: estimating its distance from the image centre and orientation. The random dot images we use have been studied in detail both theoretically and empirically for joint classification and line estimation on both algorithms and humans [46, 47].

4.1 Data model

The original data now consists of two-dimensional $D \times D$ binary images that are either empty or of a random line. Images are centred at $(0,0)$ and now the y-axis is pointed upwards to fit the natural mathematical convention on axes and angle orientations[7]. For a pixel $p \in [-\frac{D-1}{2}, \frac{D-1}{2}]^2$:

$$\varphi(p) = \begin{cases} 0 & \text{if } l = 0, \\ \begin{cases} 0 & \text{if } |u_n^\top (p - p_0)| > \frac{w}{2}, \\ 1 & \text{if } |u_n^\top (p - p_0)| \leq \frac{w}{2}, \end{cases} & \text{if } l = 1, \end{cases} \tag{11}$$

where $l \in \{0,1\}$ is the binary label line-no line of the image, w is the fixed width of all straight lines, p_0 is the orthogonal projection of the image centre onto the medial axis of the line, and $u_n \in \mathbb{R}^2$ is the normal vector to the medial axis of the line (see Figure 7). The lines are randomly chosen by selecting the distance $r = \|p_0\|_2$ and orientation θ at random, giving $u_n = (-\sin\theta, \cos\theta)^\top$ and $p_0 = -ru_n$. To ensure uniformly sampling lines in the image in the natural sense, r and θ are independently uniformly sampled: $r \sim \mathcal{U}([\iota_r, (1-\varepsilon_r)\frac{D-1}{2}])$ with $\varepsilon_r = 0.3$ and $\theta \sim \mathcal{U}([0, 2\pi])$. We enforce $\iota_r > 0$ to remove label instability that occurs with magnitude π for θ when lines pass close to the origin[8]. In our tests, we take $D = 201$ pixels, $w = 8$ and $\iota_r = 3$ pixels, and so $r \sim \mathcal{U}([3, 70])$. Note that all lines[9] are uniquely mapped to a point in the polar coordinate system (r, θ).

However, φ is degraded by a sampling process on the line along with additive white noise points, generating the observed data of random dot images φ^{data} as follows:

$$\varphi^{data} = \varphi_{\mathfrak{p}_f} \vee n_{\mathfrak{p}_b}, \tag{12}$$

where $\vee$ is the logical OR operator replacing a logical addition, $n_{\mathfrak{p}_b}$ is a white noise image where each pixel value is drawn independently, a Bernoulli with parameter[10] $\mathfrak{p}_b$, and $\varphi_{\mathfrak{p}_f}$ is a sampled version of φ where pixels outside the line[11] all take value 0, but pixels on the line have independent sampled values of Bernoulli random variables with parameter $\mathfrak{p}_f$. We followed the model and notations of [46, 47] and this is similar in style to the proposed degradations previously studied. Note that φ^{data} would have been equivalently given by the alternative model consisting in taking pixel values independently according to Bernoulli variables of parameters $\mathfrak{p}_0 = \mathfrak{p}_b$ outside the line and $\mathfrak{p}_1 = \mathfrak{p}_b + \mathfrak{p}_f - \mathfrak{p}_b\mathfrak{p}_f$ inside it, as the probability of a pixel on the line to be white is the sum of the probabilities of it being white in the sample image $\varphi_{\mathfrak{p}_f}$ ($\mathfrak{p}_f$) or in the noise image $n_{\mathfrak{p}_b}$ ($\mathfrak{p}_b$) minus the probability that it is white in both of them ($\mathfrak{p}_b\mathfrak{p}_f$). We have $\mathfrak{p}_1 > \mathfrak{p}_0$. An image with a line should thus possess a straight region of width w with a higher point density $\mathfrak{p}_1$ compared to the background density $\mathfrak{p}_0$. Following the analysis of [46, 47],

[7] Which differs from the image convention of a downwards y-axis. Angles with respect to the x-axis are opposites up to 2π between conventions.

[8] The same issue would occur for a different choice of origin, e.g. the top left corner.

[9] Except those passing through the origin, which we exclude by $\iota_r > 0$.

[10] With our notations, $p = (x, y)$ denotes a pixel position, whereas $\mathfrak{p} > 0$ is a scalar parameter of a Bernoulli distribution.

[11] In the absence of a line, we simply have $\varphi_{\mathfrak{p}_f} = \varphi \equiv 0$.

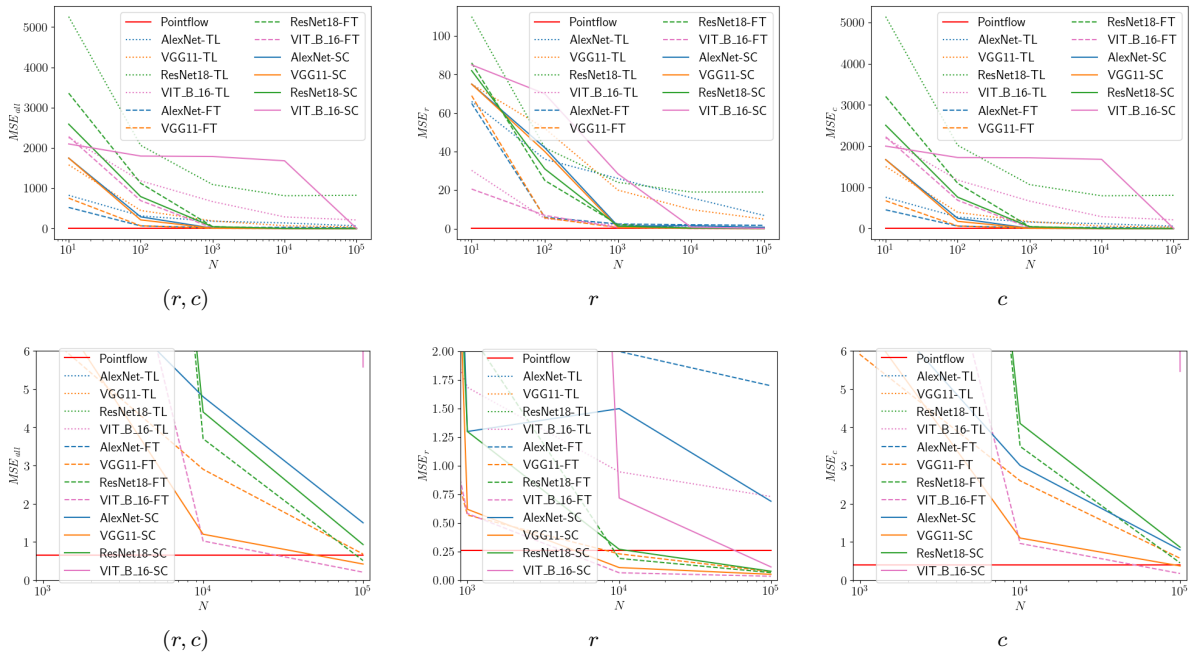


Figure 6: Test scores of the data-agnostic Pointflow and of the best networks learned from scratch, transfer-learned, or finetuned. Left: MSE computed on both the radius and the centre's location estimation. Middle: same but only on the radius estimation. Right: same but only on the centre location estimation. We zoom-in in the bottom set of figures.

we took $\mathfrak{p}_b = 1 - (1 - \mathfrak{p}'_b)^w$ with $\mathfrak{p}'_b = 0.005$ and for $w = 8$ we chose $\mathfrak{p}_f = 0.05$, making it possible to easily see and retrieve algorithmically the line without many false alarms in most but not all cases. We plot example data in Figure 8. In all our work, classes l are balanced out unless explicitly mentioned otherwise. The primary goal is line-no line classification from φ^{data}. This classification must be based on features that allow to achieve the secondary goal: estimating the two geometric numbers (r, θ) when $l = 1$, and both goals must be simultaneously performed.

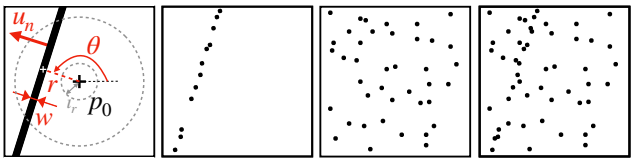


Figure 7: Line data generation process. The clean image (left) is generated by randomly sampling the white cross, at angle θ and distance r from the origin, which is associated to a unique line of width w. If a line at angle θ passes too close to the origin ($r \leq \iota_r$), then its image and that of a line of angle $\theta + \pi$ are almost identical, yet have very different angles, which is an issue for angle regression, so it is banned. Given a clean line image, we randomly sample points on it (middle left), and points anywhere in the image with a different density (middle right), and merge the two with a logical union (right). For visualisation purposes, we draw negated images (points are black instead of white).

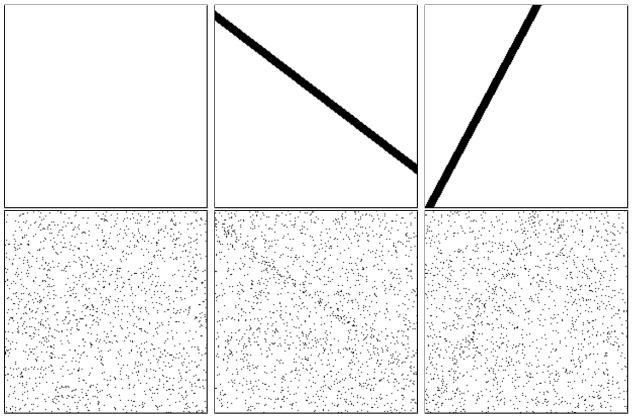


Figure 8: Three examples of clean and degraded random dot images, that have been negated, i.e. $1-\varphi$ and $1 - \varphi^{data}$, for better visualisation. Electronic zooming in is recommended for best viewing conditions.

4.2 Expert engineer's solution

Once again, the optimal estimator or classifier are not easily derived. Instead, we choose the classical and perfectly suited Hough transform [39, 40] (see Figure 9). The Hough transform was initially designed to retrieve edges in images. Once binary edge maps have been extracted, edges consist in locally maximal alignments of white pixels. To locate them, the Hough transform maps lines to a polar coordinate system (r_h, θ_h) centred in p_h, where the equation of a line is

$$r_h = x\cos(\theta_h) + y\sin(\theta_h). \tag{13}$$

Points are thus mapped to continuous non straight curves, each point on the curve corresponds to a line passing through the point at some angle. More precisely, all lines that pass by a given point (x, y) have an equation with parameters (r_h, θ_h) such that $r_h = x\cos(\theta_h) + y\sin(\theta_h)$. This means that r_h is a sinusoidal function of θ_h and to each point $p = (x, y)$ corresponds a sinusoidal curve in the Hough space (r_h, θ_h). If many points $p_i = (x_i, y_i)$ are aligned, it means that their sinusoidal curves will intersect. In reverse, many intersections of sinusoidal curves in the Hough space reveal the existence of aligned points in the image. The Hough transform is usually implemented by a voting strategy: each point in the (edge map) image votes for all the lines passing through it up to an angular resolution δ_θ^h. Votes are accumulated between all points in an array, on which we then extract the peaks to detect the lines.

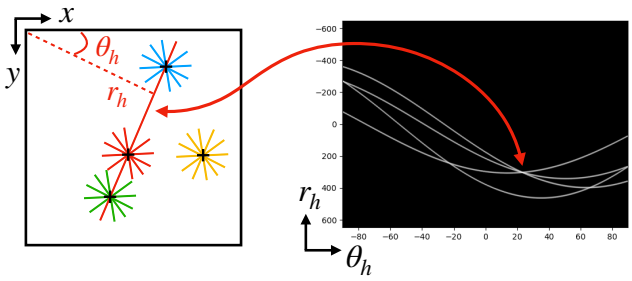


Figure 9: The Hough transform maps image points to curves, where each point on a curve is a line passing through a point in the image. Intersection of many curves happens when points in the image are aligned. It is usually implemented by a voting strategy where each image point votes for the set of sampled lines passing through it.

Magnitudes of the peaks correspond to the number of votes, as such the higher the peak, the more points are located along the line. This information provides a confidence score for the Hough transform related to the density of points along the estimated line. Since in our data a higher density of $\mathfrak{p}_1$ is expected[12] along lines and $\mathfrak{p}_0$ everywhere else, the proposed expert's solution is to extract the maximum peak on the Hough transform of φ^{data} and compare its intensity $\mathcal{P}^h_{\max}(\varphi^{data})$ to a threshold λ, which is global since $\mathfrak{p}_b$, $\mathfrak{p}_f$, and w are shared between all images. The classification result at threshold λ is thus $\hat{l}_h = \mathbb{1}_{\mathcal{P}^h_{\max}(\varphi^{data})>\lambda}$, where $\mathbb{1}$ is the indicator function on the set of data functions. This classification strategy is based on line estimation to be provided by the line with the estimated line parameters (r_h^*, θ_h^*) of the peak. Due to possible convention differences, conversion is needed to get the final estimate $(\hat{r}_h, \hat{\theta}_h)$. See B for more information on implementation details.

[12]This is expected on average but differs randomly in each sample. Random events can also lead to seeing lines in white noise, which in psychophysics is called apophenia or the clustering illusion [48] and is mathematically likely [49].

4.3 Neural models

The expert's method is to be compared with convolutional neural networks. We chose the exact same networks as in the previous section, namely AlexNet [7], VGG [8], ResNet [9], and VIT [44], trained either from scratch (SC), with transfer-learning (TL), or finetuning (FT) like before.

As previously motivated, simultaneous classification and line position estimation is performed. As such, three numbers need to be extracted from the output of the network $\hat{r}_n$, $\hat{\theta}_n$, and $\hat{l}_n$. However, due to the periodic nature, but not values, of angles, careless black box angular regression can lead to unstable behaviours around the border of the angle interval, here 0 and 2π. While it is partly mitigated by using geodesic angle errors rather than crude differences in angle values, the common recipe is to regress periodic values, usually $x^\theta = \cos\theta$ and $y^\theta = \sin\theta$, and then compute the angle via the arctangent[13] rather than let the network find this operation on its own. As such, we modify the last linear layer of the neural architecture to output four numbers: r_n, x_n^θ, y_n^θ, and the logit $y_n^{\log}$, and once trained we denote them respectively r_n^*, $x_n^{\theta^*}$, $y_n^{\theta^*}$, and $l_n^{\log^*}$. The final estimates based on these numbers are $\hat{r}_n = r_n^*$ and $\hat{\theta}_n = \arctan2\,(y_n^{\theta^*}, x_n^{\theta^*})$ up to rescaling[14], and $\hat{l}_n = \mathbb{1}_{l_n^*>\lambda}$ where $l_n^* = \sigma(l_n^{\log^*})$ and $\sigma = \frac{1}{1+e^{-x}}$ is the sigmoid operator mapping logits to $[0, 1]$.

For the estimation part (r, θ), we wish to minimise the minimum squared geodesic error *MSGE*, a modification of the *MSE* loss where the distance between angles is now geodesic rather than the crude value difference. The geodesic distance between two angles θ_1 and θ_2 is given by

$$G(\theta_1, \theta_2) = \mathbb{1}_{|\theta_2-\theta_1|\leq\pi}|\theta_2 - \theta_1| + \mathbb{1}_{|\theta_2-\theta_1|>\pi}\Big(2\pi - \max(\theta_1, \theta_2) + \min(\theta_1, \theta_2)\Big), \tag{14}$$

with $G(\theta_1, \theta_2) \in [0, \pi]$, therefore the *MSGE* is given by

$$MSGE = \frac{1}{N_b^{(1)}}\sum_{i=1}^{N_b^{(1)}} l_i\Bigg[\Big((r_n)_i - r_i\Big)^2 + G\Big(\arctan2\,\big((y_n^{\theta_i})_i, (x_n^{\theta_i})_i\big), \theta_i\Big)^2\Bigg], \tag{15}$$

where $N_b^{(1)} \leq N_b$ is the number of line images in the batch and N_b is the batch size, the index i in r_i is the value of r for the i-th batch sample, and the same goes

[13]This is done with the 2-argument arctangent $\theta = \arctan2(y, x)$, which essentially computes $\arctan(\frac{y}{x})$ the regular scalar arctangent, but can add or subtract π to it depending on the quadrant of (x, y), and also handles cases where one of the arguments is 0. This allows the predicted angle to lie in an interval of length 2π.

[14]As in the previous section, the estimation training labels were scaled to $[-1, 1]$ using $r_s = \frac{D-1}{4}(r - \frac{D-1}{4})$ and $\theta_s = \frac{1}{\pi}(\theta - \pi)$. The results presented in this paper, either formulae, numbers, or figures, have been rescaled to the original scale: r and θ and not r_s and θ_s.

for the other symbols $(r_n)_i$, $(x_n^{\theta_i})_i$, $(y_n^{\theta_i})_i$, θ_i, l_i but also $(l_n^{\log})_i$. We abused notations when no line is present, i.e. $N_b^{(1)} = 0$, as then $MSGE = 0$.

Without further constraints, the network does not respect the condition $(x_n^\theta)^2 + (y_n^\theta)^2 = 1$, leading often to a collapse to the origin. To stabilise training, we add the regularisation loss

$$L_{reg} = \frac{1}{N_b^{(1)}} \sum_{i=1}^{N_b^{(1)}} l_i \Big((x_n^\theta)_i^2 + (y_n^\theta)_i^2 - 1 \Big)^2 . \tag{16}$$

Note that an alternative strategy to bypass the geodesic issue on angles and the previously mentioned collapse would be to use the regular *MSE* on the cosine and sine estimates directly. However, naturally random uniform lines in the plane do not have an orientation of uniform cosine and sine, contaminating the training labels with a bias that the networks could exploit. For this reason we did not use this strategy and work in the proposed angular domain.

The binary classifier is trained as is the norm using the binary cross entropy loss

$$BCE = \frac{1}{N_b} \sum_{i=1}^{N_b} -\Big[l_i \ln \Big(\sigma\big((y_n^{\log})_i\big) \Big) + (1 - l_i) \ln \Big(1 - \sigma\big((l_n^{\log})_i\big) \Big) \Big]. \tag{17}$$

The total loss used for training the networks is then

$$L = BCE + MSGE + \lambda_{reg} L_{reg}, \tag{18}$$

where λ_{reg} is a Lagrangian multiplier chosen to be 1 in our experiments. With this new loss, the training methodology is unchanged from the previous case: only $N_r = 1$ run is performed per learning rate configuration η ranging in the same values, with the same batch size and number of epochs. See Table 1 for a summary of our training procedure and choice of hyperparameters. Since classification is the main goal, the optimal learning rate is decided based only on the classification performance. The cross entropy loss is a differentiable proxy for an interesting classification score, usually taken to be the accuracy at $\lambda = \frac{1}{2}$. Instead, we prefer the area under the curve $AUC \in [0, 1]$ of the receiver operating characteristic (ROC) curve that summarises the balance between the true and false positive rates TPR and FPR respectively over all λ. The higher the AUC is the better, and for the blind random predictor it has value $\frac{1}{2}$. As such, the optimal learning rate $\eta^*(N)$ for N training samples is chosen based on the maximum AUC score on $N_v = 100000$ independent validation samples. The test set uses $N_t = N_v$ samples whereas the training sets have $N \in \{10, 100, 1000, 10000, 100000\}$ ones. Note that in the validation, test, and all training sets, half the data comes from a line image $l = 1$. For example, our largest training set contains 50000 random dot images of lines and 50000 pure white noise images.

4.4 Results

We present the results in Figures 10 and 11 and Table 4 for both the main classification and auxiliary estimation tasks.

Classification Once again the transfer-learning networks perform poorly and are incapable to beat the simple data-agnostic Hough transform with the provided amount of data. However, when training the entire networks from scratch or with finetuning, the CNN networks all outperform the Hough transform with over $N = 1000$ training samples, even reaching close to perfect classification with $N = 100000$ samples. Unlike the CNNs, the transformer VIT networks require more data to outperform the Hough transform, needing at least $N = 10000$ samples. This illustrates the widespread belief in the field that transformers require significantly more data to be properly trained than CNNs. Overall, these results show that this classification task is fairly easy for neural networks, with ResNet-SC and AlexNet-SC on par or better than the Hough transform with only $N = 100$ training samples.

Estimation When observing the *MSGE* for radius or angle prediction, the Hough transform seems to struggle compared to the neural networks, being on par or slightly better than the transfer-learning networks for high number of training samples. All CNNs trained from scratch or with finetuning, along with transfer-learned VIT, outperform the Hough transform with at least $N = 10000$ training samples, some of them (VGG-FT and AlexNet-FT) doing so with only $N = 1000$, and the gap with the Hough transform is large at $N = 100000$. These results would suggest that neural networks need few amounts of data to compete with the Hough transform. However, this analysis is not fully correct. There is an artefact in the presented results stemming from the choice of summariser: a mean of errors. When a method fails to predict the correct line, the error can become large in the authorised range. In the Hough transform approach, predicting an incorrect line, e.g. by hallucinating a line in white noise, leads to a somewhat random estimation uncorrelated to the groundtruth geometric quantities, and thus to high errors. Since the Hough transform is not a perfect classifier, with an AUC of approximately 90%, these large errors act like large outliers, which heavily bias the mean score. To complement the analysis, we analyse the full distribution of estimation errors and found that the Hough transform error is heavy tailed with 20% of large errors that do not reflect the acute precision of the method on the other 80% of data. In comparison, most successful convolutional neural networks tend to have an error distribution that is significantly more progressive and at higher levels than the Hough transform approach, which reflects a lack of precision in most cases of these methods. To quantitatively reveal this effect, we study the median of squared (geodesic) errors *MdSGE* as a replacement of their mean *MSGE*. It becomes clear that the

Hough transform is an extremely precise approach in most cases, and networks need at least $N = 10000$ samples to be on par with it. To outperform the Hough transform in precision for the radius estimation, the methods require $N = 100000$ training samples (VGG-FT, VGG-SC, AlexNet-SC, ResNet-FT). No method achieves better *MdSGE* for angle estimation than the Hough transform even with $N = 100000$ training samples. Furthermore, none of the VITs manage to outperform the Hough transform for either radius or angle estimation, requiring more data than the maximum available.

From this experiment, we conclude that classification is indeed simpler than regression, requiring at least hundreds to thousands of samples to compare with or outperform an expert method for classification. Furthermore, we found once again that the number climbs to tens of thousands of samples for estimation. Interestingly, this experiment also shows that with fewer examples, e.g. a thousand, they can avoid random estimations in failure cases to maximise their average performance, which was not handled in the design of our expert method. We thus provide a criterion that a data model is preferable if fewer than 1000 samples are available if only classification is of interest, and otherwise 10000 samples if precise estimates are required.

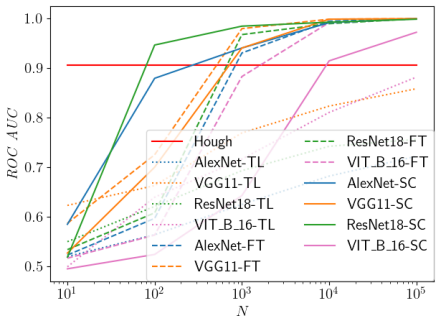


Figure 10: Test classification AUC scores of the data-agnostic Hough transform and of the best networks learned from scratch, transfer-learned, or finetuned. The AUC scores are the area under the ROC curves for networks trained with $N = 10, 100, 1000, 10000, 100000$ samples.

	N	Hough	AlexNet			VGG			ResNet			VIT		
			TL	FT	SC	TL	FT	SC	TL	FT	SC	TL	FT	SC
AUC	0	**0.906**												
	10		0.520	0.523	0.585	0.623	0.587	0.527	0.550	0.534	0.519	0.499	0.517	0.495
	100		0.564	0.597	0.879	0.663	0.725	0.700	0.621	0.608	**0.946**	0.640	0.563	0.524
	1000		0.625	**0.930**	**0.940**	0.769	**0.979**	**0.940**	0.693	**0.967**	**0.984**	0.716	0.826	0.643
	10000		0.682	**0.994**	**0.991**	0.823	**0.998**	**0.998**	0.742	**0.989**	**0.992**	0.810	**0.991**	**0.914**
	100000		0.722	**0.999**	**0.999**	0.858	**0.999**	**0.999**	0.756	**0.999**	**0.998**	0.881	**0.999**	**0.972**
r (mean)	0	**330**												
	10		380	390	450	420	440	420	1400	1400	390	490	1200	430
	100		490	530	380	400	370	370	580	540	400	480	390	1300
	1000		400	**160**	380	360	**120**	380	520	380	**330**	390	450	810
	10000		360	**84**	**87**	**280**	**22**	**28**	410	**220**	**260**	430	**45**	550
	100000		340	**23**	**6.8**	**240**	**7.7**	**6.9**	390	**8.7**	**40**	380	**14**	380
θ (mean)	0	**0.647**												
	10		0.819	0.828	0.824	0.825	0.826	0.825	0.919	0.919	0.829	0.823	0.823	0.824
	100		0.822	0.828	0.819	0.817	0.841	0.829	0.866	0.947	0.825	0.817	0.838	0.817
	1000		0.827	0.789	0.816	0.831	0.719	0.815	1.05	0.809	0.700	0.813	**0.607**	0.929
	10000		0.815	**0.607**	0.655	0.817	**0.276**	**0.479**	0.824	**0.582**	0.738	0.777	**0.423**	0.797
	100000		**0.632**	**0.089**	**0.048**	0.766	**0.053**	**0.099**	0.816	**0.036**	**0.233**	0.746	**0.161**	0.805
r (median)	0	**3.3**												
	10		280	280	280	280	280	280	880	850	280	280	710	280
	100		280	290	280	270	270	280	310	300	280	280	280	640
	1000		280	60	280	250	37	280	290	230	190	270	220	390
	10000		260	47	29	160	6.4	10	270	100	150	260	12	300
	100000		240	11	**1.2**	120	**2.0**	**1.8**	270	**2.1**	15	240	4.5	270
θ (median)	0	**0.0005**												
	10		0.614	0.625	0.621	0.622	0.622	0.625	0.636	0.643	0.623	0.621	0.617	0.623
	100		0.617	0.627	0.611	0.603	0.623	0.626	0.630	0.628	0.621	0.605	0.623	0.605
	1000		0.623	0.568	0.602	0.623	0.461	0.610	0.632	0.536	0.436	0.606	0.344	0.629
	10000		0.606	0.317	0.387	0.614	0.096	0.203	0.615	0.305	0.491	0.544	0.057	0.576
	100000		0.344	0.010	0.002	0.531	0.011	0.031	0.605	0.002	0.033	0.487	0.018	0.590

Table 4: Classification (top) and estimation scores – mean (middle) and median (bottom) of the geodesic squared errors – of the networks compared to the Hough transform on test data. The estimation scores are computed on r and θ separately. Networks were trained for joint line classification and (r, θ) estimation. All scores correspond to the selected networks providing the best AUC on validation data.

5 Conclusion

We analysed the amount of data required by neural networks, either shallow custom ones or deep famous ones, trained from scratch, finetuned, or transfer-learned, to compete with optimal or state-of-the-art traditional data-agnostic methods based on mathematical data generation models. To do so, we mathematically generated data, and fed various amounts of samples to the networks for training. We found that tens of thousands of data examples are needed for the networks to be on par or beat the traditional methods, if they are able to, for estimation problems, whereas only thousands suffice for classification. For mathematical accuracy, we did not investigate real-world problems, which are commonly harder with less accurate or non-existent mathematical data generation models, but more data should be needed in those complex tasks. We also left for further work a thorough investigation of the dependence on the dimensionality of the data. We have empirically derived a simple criterion, enabling researchers working on tasks where data is not easily available, to choose whether to use model-based traditional methods, by using either preexisting or newly created data generation models, or simply feed data to deep neural networks.

A Pointflow implementation details

The pointflow dynamics are implemented by discretising time and approximating the time derivative with a forward finite difference scheme, although it could be improved with a Runge-Kutta 4 implementation [50]. Given the small magnitudes of the fields, we found that a large time step $dt = 50$ works well. We define three thresholds, $\tau_l = 0.9$ for C_l, $\tau_s = 10^{-6}$ for C_s, and $\tau_{len} = 0.001$. we consider having looped C_l if a point reaches a previous point within squared Euclidean distance τ_l while having on the trajectory between the looping points at least one point with squared distance to them of at least τ_l. A trajectory is stuck if it reaches a point where the current flow V has small magnitude $\|V\|_2^2 \leq \tau_s$. Each flow is run for $N_i = 1000$ iterations, and trajectories shorter than τ_{len} are discarded, e.g. trajectories of type C_s. We used $\sigma_{Pf} = 5$ for blurring out the noise before computing the fields. The implemented pointflow algorithm for finding contours in our circle images is presented in Algorithm 1.

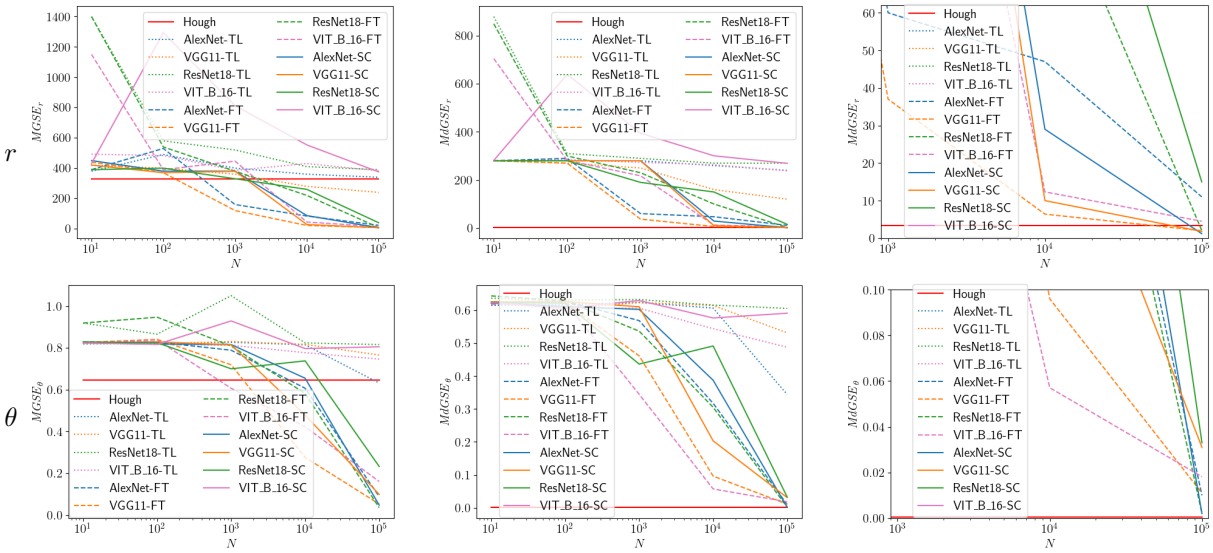


Figure 11: Test estimation scores of the data-agnostic Hough transform and of the best networks learned from scratch, transfer-learned, or finetuned. Our summary quantitative scores are the *MSGE* and *MdSGE*, i.e. the mean (left) and median (middle and zoomed-in right) of the sorted squared geodesic errors on the test set. Top: errors for the distance estimation. Bottom: same but for the angle estimation.

Algorithm 1 Contour integration with Pointflow on image I

Compute $I_b = g_{\sigma_{Pf}} * I$
Compute $V_a = \nabla \|\nabla I_b\|_2$ and $V_r = \nabla I_b^\top$
Compute $V_+ = \frac{1}{2}(V_a + V_r)$ and $V_- = \frac{1}{2}(V_a - V_r)$
Choose N_{Pf} random points independently and uniformly in the image domain $[-\frac{D-1}{2}, \frac{D-1}{2}]^2$
Let $\mathcal{C} = []$ be an empty list of computed contours
for $i = 1 \dots N_{Pf}$ **do**
  Let $(traj_+, C)$ be the flow along V_+ starting from the i-th point
  if $C = C_l$ and length$(traj_+) \geq \tau_{len}$ **then**
    Let $(traj_l, C_l)$ be the reflow along V_+ starting from the endpoint of $traj_+$
    $\mathcal{C}$.append$(traj_l)$
  else if $C = C_o$ and length$(traj_+) \geq \tau_{len}$ **then**
    Let $(traj_-, C_-)$ be the reflow along V_- starting from the endpoint of $traj_+$
    if $C_- = C_l$ and length$(traj_-) \geq \tau_{len}$ **then**
      Let $(traj_l, C)$ be the reflow along V_- starting from the endpoint of $traj_-$
      $\mathcal{C}$.append$(traj_l)$
    end if
  end if
end for
Return $\mathcal{C}$

After computing the list of contours $\mathcal{C}$ in the image I, we estimate the radius using the average curve length $\hat{r} = \frac{1}{2\pi}\sum_{i=1}^{|\mathcal{C}|} \text{length}(\mathcal{C}_i)$. Since the average of the points did not yield the best estimation of the circle centre, we estimate it instead using least squares. The equation of a circle is naturally given by $(x-c_x)^2 + (y-c_y)^2 = r^2$, which can be written as $\theta_1 x + \theta_2 y + \theta_3 = x^2 + y^2$, where $\theta_1 = 2c_x$, $\theta_2 = 2c_y$, and $\theta_3 = r^2 - c_x^2 - c_y^2$. We can thus estimate for each contour $\theta = (\theta_1, \theta_2, \theta_3)^\top$ by least squares as $\hat{\theta} = A^\top (AA^\top)^{-1} B$, with $A_{i,:} = (x_i, y_i, 1)$ and $B_i = x_i^2 + y_i^2$ and i ranging in the number of computed points on the contour. From $\hat{\theta}$ we can estimate $\hat{c} = (\frac{\theta_1}{2}, \frac{\theta_2}{2})$. The final centre estimation is then given by the average of this estimation over all contours. Note that we can also estimate r using θ_3 but we found that it did not outperform the lenght strategy so we do not use it.

B Hough transform implementation details

Theoretically, we could take the same conventions for the Hough transform as for the data (r, θ), namely that the origin is the image centre and that the y-axis is oriented downwards. However, in the Hough transform the origin is usually taken at the top left pixel $p_h = (-\frac{D-1}{2}, \frac{D-1}{2})$ and the y-axis is oriented downwards as is common for images. For such a choice, angles for lines passing through the image need only be taken in $[-\frac{\pi}{2}, \frac{\pi}{2}]$ rather than in an interval of length 2π. This different convention is used in most built-in methods, e.g., in the scikit-image package [51]. While conventions differ, there is a one-to-one mapping $(r_h^*, \theta_h^*) \mapsto (\hat{r}_h, \hat{\theta}_h)$ between representations in both sys-

tems.

In our experiments, we use the built-in Hough transform from the scikit-image package. Each point votes for k_h uniformly spaced lines in $[-\frac{\pi}{2}, \frac{\pi}{2}]$, which means an angular precision of $\delta_\theta^h = \frac{\pi}{k_h}$ rad $= \frac{180}{k_h}^\circ$. In our experiments we tested with $k_h \in \{360, 1800, 10800\}$ i.e. $\delta_\theta^h \in \{\frac{1}{2}^\circ, \frac{1}{10}^\circ, 1'\}$, which gave similar results so we only present $k_h = 10800$ and $\delta_\theta^h = 1'$. The maximum peak is chosen giving the line estimate (r_h^*, θ_h^*) in the usual convention that differs from ours. The conversion is given by the following result.

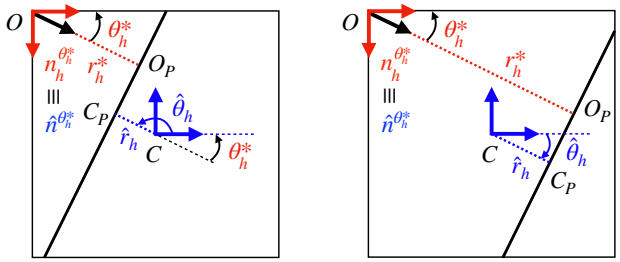


Figure 12: Conversion of the lines estimates in the Hough top-left centred and downwards y-axis to the image centre and upwards y-axis. The conversion depends on whether the line separates (figure on the left) or not (figure on the right) both origins.

Theorem 1. *Denote* $\bar{p}_h = (\frac{D-1}{2}, \frac{D-1}{2})^\top$ *and* $n_h^{\theta_h^*} = (\cos(\theta_h^*), \sin(\theta_h^*))$. *The conversion from the top-left centred and downwards y-axis to the image centre with an upwards y-axis is given by*

$$\begin{cases} \hat{r}_h = \left| r_h^* - \bar{p}_h^\top n_h^{\theta_h^*} \right|, \\ \hat{\theta}_h = -\theta_h^* + \mathbb{1}_{r_h^* - \bar{p}_h^\top n_h^{\theta_h^*} > 0} \times \pi \mod 2\pi. \end{cases} \tag{19}$$

Proof. We give an analytical formal proof and a lighter geometric one. For conciseness, we call the top-left centred and downwards y-axis system the Hough system of coordinates.

Analytical proof Denote (x, y) and (x^*, y^*) pixel coordinates in our and the Hough systems respectively. In the Hough system, the line is parametrised by

$$r_h^* = x^* \cos(\theta_h^*) + y^* \sin(\theta_h^*). \tag{20}$$

Since the coordinates between systems is given by

$$x = x^* - \bar{p}_{h_x} \qquad \text{and} \qquad y = \bar{p}_{h_y} - y^*, \tag{21}$$

where $\bar{p}_h = (\bar{p}_{h_x}, \bar{p}_{h_y})^\top$, the line can be reparametrised as

$$r_h^* - \bar{p}_h^\top n_h^{\theta_h^*} = x \cos(-\theta_h^*) + y \sin(-\theta_h^*). \tag{22}$$

If $r_h^* - \bar{p}_h^\top n_h^{\theta_h^*} > 0$, we recognise the equation of a line in our coordinate system with distance to the image centre of $\hat{r} = r_h^* - \bar{p}_h^\top n_h^{\theta_h^*}$ and angle $\hat{\theta}_h = -\theta_h^* \mod 2\pi$. If $r_h^* - \bar{p}_h^\top n_h^{\theta_h^*} < 0$, then flipping the sign in the equation gives $\hat{r} = -(r_h^* - \bar{p}_h^\top n_h^{\theta_h^*})$ and $\hat{\theta}_h = -\theta_h^* + \pi \mod 2\pi$. The case $r_h^* - \bar{p}_h^\top n_h^{\theta_h^*} = 0$ implies that the line passes through the image centre, which has been excluded, and in that case $\hat{\theta}_h$ is not defined.

Geometric proof A more intuitive proof involves direct geometry, see Figure 12. In the Hough system, $n_h^{\theta_h^*}$ is the unit vector pointing from the top-left corner to its projection on the line. In our system, this unit vector becomes $\hat{n}^{\theta_h^*} = (\cos(\theta_h^*), -\sin(\theta_h^*))$. Calling O (resp. C) the geometric point representing the top-left (resp. centre) of the image, and O_P (resp. C_P) its projection on the line, the Chasles relation gives $OC = OO_P + O_PC_P + C_PC$. Projecting on the line normal, we get

$$OC^\top \hat{n}^{\theta_h^*} = OO_P^\top \hat{n}^{\theta_h^*} + C_PC^\top \hat{n}^{\theta_h^*} = r_h^* \pm r. \tag{23}$$

After noticing that $OC^\top \hat{n}^{\theta_h^*} = (p_h)_x \cos(\theta_h^*) - (p_h)_y \sin(-\theta_h^*) = \bar{p}_h^\top n_h^{\theta_h^*}$, we get that $\hat{r} = \left| r_h^* - \bar{p}_h^\top n_h^{\theta_h^*} \right|$. For the angle, it is easy to see that flipping the axis flips the sign of the angle, to which an angle of π must be added when the line separates O and C, i.e. $r_h^* < |OC^\top n^{\theta_h^*}| = \bar{p}_h^\top n_h^{\theta_h^*}$.

□

Acknowledgements

This work is in part supported by the French government under management of Agence Nationale de la Recherche as part of the "Investissements d'avenir" program, reference ANR-19-P3IA-0001 (PRAIRIE 3IA Institute).

References

[1] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444.

[2] I. Goodfellow, Y. Bengio, A. Courville, Deep learning, MIT press, 2016.

[3] N. Wiener, Extrapolation, interpolation, and smoothing of stationary time series: with engineering applications, Vol. 113, MIT press Cambridge, MA, 1949.

[4] A. B. Novikoff, On convergence proofs for perceptrons, Tech. rep., Stanford Research Institute, Menlo Park, CA, USA (1963).

[5] M. Elad, Sparse and redundant representations: from theory to applications in signal and image processing, Springer, 2010.

[6] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, et al., Handwritten digit recognition with a back-propagation network, Advances in neural information processing systems 2 (1989).

[7] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, Communications of the ACM 60 (6) (2017) 84–90.

[8] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556 (2014).

[9] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[10] A.-r. Mohamed, G. Dahl, G. Hinton, Deep belief networks for phone recognition, in: Nips workshop on deep learning for speech recognition and related applications, Vol. 1, 2009, p. 39.

[11] R. Geirhos, C. R. Temme, J. Rauber, H. H. Schütt, M. Bethge, F. A. Wichmann, Generalisation in humans and deep neural networks, Advances in neural information processing systems 31 (2018).

[12] R. Geirhos, K. Narayanappa, B. Mitzkus, T. Thieringer, M. Bethge, F. A. Wichmann, et al., Partial success in closing the gap between human and machine vision, Advances in Neural Information Processing Systems 34 (2021) 23885–23899.

[13] Y. Bengio, Y. Lecun, G. Hinton, Deep learning for AI, Communications of the ACM 64 (7) (2021) 58–65.

[14] B. Shlegeris, F. Roger, L. Chan, E. McLean, Language models are better than humans at next-token prediction, Transactions on Machine Learning Research (2024).

[15] R. Shwartz-Ziv, A. Painsky, N. Tishby, Representation compression and generalization in deep neural networks, arXiv preprint arXiv:2202.06749 (2018).

[16] D. Pereg, Information theoretic perspective on sample complexity, Neural Networks 167 (2023) 445–449.

[17] B. Adcock, N. Dexter, The gap between theory and practice in function approximation with deep neural networks, SIAM Journal on Mathematics of Data Science 3 (2) (2021) 624–655.

[18] P. Grohs, F. Voigtlaender, Proof of the theory-to-practice gap in deep learning via sampling complexity bounds for neural network approximation spaces, Foundations of Computational Mathematics (2023) 1–59.

[19] B. Adcock, S. Brugiapaglia, N. Dexter, S. Moraga, Learning smooth functions in high dimensions: from sparse polynomials to deep neural networks, arXiv preprint arXiv:2404.03761 (2024).

[20] J. Berner, P. Grohs, F. Voigtlaender, Learning relu networks to high uniform accuracy is intractable, arXiv preprint arXiv:2205.13531 (2022).

[21] A. Abdeljawad, P. Grohs, Sampling complexity of deep approximation spaces, arXiv preprint arXiv:2312.13379 (2023).

[22] N. B. Kovachki, S. Lanthaler, H. Mhaskar, Data complexity estimates for operator learning, arXiv preprint arXiv:2405.15992 (2024).

[23] J. Berner, P. Grohs, G. Kutyniok, P. Petersen, The modern mathematics of deep learning, arXiv preprint arXiv:2105.04026 78 (2021).

[24] H. Boche, A. Fono, G. Kutyniok, Limitations of deep learning for inverse problems on digital hardware, IEEE Transactions on Information Theory (2023).

[25] Z. Xu, X. Zhang, Stability of least square approximation under random sampling, arXiv preprint arXiv:2407.10221 (2024).

[26] S. N. Attoor, E. R. Dougherty, Classifier performance as a function of distributional complexity, Pattern Recognition 37 (8) (2004) 1641–1651.

[27] A. Broumand, M. S. Esfahani, B.-J. Yoon, E. R. Dougherty, Discrete optimal bayesian classification with error-conditioned sequential sampling, Pattern Recognition 48 (11) (2015) 3766–3782.

[28] B. Kadıoğlu, P. Tian, J. Dy, D. Erdoğmuş, S. Ioannidis, Sample complexity of rank regression using pairwise comparisons, Pattern Recognition 130 (2022) 108688.

[29] V. N. Vapnik, A. Y. Chervonenkis, On the uniform convergence of relative frequencies of events to their probabilities, in: Measures of complexity, Springer, 2015, pp. 11–30.

[30] W. Maass, Neural nets with superlinear vc-dimension, Neural Computation 6 (5) (1994) 877–884.

[31] A. Sakurai, Tight bounds for the vc-dimension of piecewise polynomial networks, Advances in neural information processing systems 11 (1998).

[32] L. Pinto, S. Gopalan, P. Balasubramaniam, On the stability and generalization of neural networks with vc dimension and fuzzy feature encoders, Journal of the Franklin Institute 358 (16) (2021) 8786–8810.

[33] W. Liu, Y. Yang, Analysis of autoencoders with vapnik-chervonenkis dimension, in: International Conference on Neural Computing for Advanced Applications, Springer, 2022, pp. 316–326.

[34] V. Cherkassky, E. H. Lee, To understand double descent, we need to understand vc theory, Neural Networks 169 (2024) 242–256.

[35] M. Anthony, P. L. Bartlett, Neural Network Learning: Theoretical Foundations, Cambridge University Press, 1999.

[36] A. Alwosheel, S. van Cranenburgh, C. G. Chorus, Is your dataset big enough? sample size requirements when using artificial neural networks for discrete choice analysis, Journal of choice modelling 28 (2018) 167–182.

[37] V. Lakshmanan, S. Robinson, M. Munn, Machine learning design patterns, O'Reilly Media, 2020.

[38] F. Yang, L. D. Cohen, A. M. Bruckstein, A model for automatically tracing object boundaries, in: 2017 IEEE International Conference on Image Processing (ICIP), IEEE, 2017, pp. 2692–2696.

[39] P. V. Hough, Method and means for recognizing complex patterns, uS Patent 3,069,654 (Dec. 18 1962).

[40] R. O. Duda, P. E. Hart, Use of the hough transformation to detect lines and curves in pictures, Communications of the ACM 15 (1) (1972) 11–15.

[41] T. Dagès, L. D. Cohen, A. M. Bruckstein, A model is worth tens of thousands of examples, in: International Conference on Scale Space and Variational Methods in Computer Vision, Springer, 2023, pp. 223–235.

[42] T. Dagès, M. Lindenbaum, A. M. Bruckstein, From compass and ruler to convolution and nonlinearity: On the surprising difficulty of understanding a simple cnn solving a simple geometric estimation task, arXiv preprint arXiv:2303.06638 (2023).

[43] B. Bai, F. Yang, L. Chai, Point flow edge detection method based on phase congruency, in: 2019 Chinese Automation Congress (CAC), IEEE, 2019, pp. 5853–5858.

[44] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, N. Houlsby, An image is worth 16x16 words: Transformers for image recognition at scale, in: International Conference on Learning Representations, 2021.

[45] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, et al., Imagenet large scale visual recognition challenge, International journal of computer vision 115 (3) (2015) 211–252.

[46] T. Dagès, M. Lindenbaum, A. M. Bruckstein, Seeing things in random-dot videos, arXiv preprint arXiv:1907.12195 (2019).

[47] T. Dagès, M. Lindenbaum, A. M. Bruckstein, Seeing things in random-dot videos, in: Pattern Recognition: 5th Asian Conference, ACPR 2019, Auckland, New Zealand, November 26–29, 2019, Revised Selected Papers, Part I 5, Springer, 2020, pp. 195–208.

[48] T. Gilovich, How we know what isn't so, Simon and Schuster, 2008.

[49] D. G. Kendall, W. S. Kendall, Alignments in two-dimensional random sets of points, Advances in Applied probability 12 (2) (1980) 380–424.

[50] J. Butcher, Numerical Methods for Ordinary Differential Equations, Wiley, 2008.

[51] S. Van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, T. Yu, scikit-image: image processing in python, PeerJ 2 (2014) e453.