What do AI algorithms actually learn? – On false structures in deep learning

Laura Thesing*   Vegard Antun†   Anders Hansen* †

June 4, 2019

Abstract

There are two big unsolved mathematical questions in artificial intelligence (AI): (1) Why is deep learning so successful in classification problems and (2) why are neural nets based on deep learning at the same time universally unstable, where the instabilities make the networks vulnerable to adversarial attacks. We present a solution to these questions that can be summed up in two words; false structures. Indeed, deep learning does not learn the original structures that humans use when recognising images (cats have whiskers, paws, fur, pointy ears, etc), but rather different false structures that correlate with the original structure and hence yield the success. However, the false structure, unlike the original structure, is unstable. The false structure is simpler than the original structure, hence easier to learn with less data and the numerical algorithm used in the training will more easily converge to the neural network that captures the false structure. We formally define the concept of false structures and formulate the solution as a conjecture. Given that trained neural networks always are computed with approximations, this conjecture can only be established through a combination of theoretical and computational results similar to how one establishes a postulate in theoretical physics (e.g. the speed of light is constant). Establishing the conjecture fully will require a vast research program characterising the false structures. We provide the foundations for such a program establishing the existence of the false structures in practice. Finally, we discuss the far reaching consequences the existence of the false structures has on state-of-the-art AI and Smale's 18th problem.

1   Introduction

It is now well established through the vast literature on adversarial attacks [7, 8, 17, 19, 20, 21, 29, 31, 33, 36] (we can only cite a small subset here) on neural networks for image classification that deep learning provides highly successful, yet incredibly unstable neural networks for classification problems. Moreover, recently, the instability phenomenon has also been shown [2] for deep learning in image reconstruction and inverse problems [1, 13, 15, 18, 30, 32, 40]. Thus, this phenomenon of instability seems to be universal. What is fascinating is that despite intense research trying to solve the instability issue [6, 10, 14, 16, 23, 37], the problem is still open. As a result, there is a growing concern regarding the consequences of the instabilities of deep learning methods in the sciences. Indeed, *Science* [9] recently reported on researchers warning about potential fatal consequences in medicine due to the instabilities. Hence, we are left with the following fundamental question:

> *Why are current deep learning methods so successful in image classification, yet universally unstable and, hence, vulnerable to adversarial attacks?*

In this paper we provide a radical conjecture answering the above question in classification and explaining why this problem will not be solved unless there is a fundamental rethink of how to approach learning. We provide the first steps towards such a theory.

---

*Department of Applied Mathematics and Theoretical Physics, University of Cambridge, UK
†Department of Mathematics, University of Oslo, Norway

Figure 1: Image of a fire truck with a horizontal blue line in the lower right corner, as suggested in the thought experiment in Example 2.1. As discussed in the example, the blue line yields many potential false structures.

Conjecture 1.1 (False structures in classification)**.** *The current training process in deep learning for classification forces the neural network to learn a different (false) structure and not the actual structure of the classification problem. There are three main components:*

  (i) (**Success**) *The false structure correlates well with the original structure, hence one gets a high success rate.*
 (ii) (**Instability**) *The false structure is unstable, and thus the network is susceptible to adversarial attacks.*
(iii) (**Simplicity**) *The false structure is simpler than the desired structure, and hence easier to learn e.g. fewer data is needed and the numerical algorithm used in the training easily converges to the neural network that captures the false structure.*

Remark 1.2 (Structure)*.* One can think of the word structure to mean the concept of what would describe a classification problem. Considering an image, we could think of what makes humans recognise a cat or a fire truck. In particular, a structure describing a cat would encompass all the features that make humans recognise cats. However, classification problems extend beyond image recognition. For example, one may want to classify sound patterns or patters in meteorological data, seismic data etc. Thus, we need a proper mathematical definition of what we mean by structure and also false structure.

1.1   Smale's 18th problem

Based on a request from V. Arnold, inspired by Hilbert's list of mathematical problems for the 20th century, S. Smale created a list of mathematical problems for the 21st century [28]. The last problem on this list is more relevant than ever and echoes Turing's paper from 1950 [35] on the question of artificial intelligence. Turing asks if a computer can think, and suggests the imitation game as a test for his question about artificial intelligence. Smale takes the question even further and asks in his 18th problem the following:

> " *What are the limits of intelligence, both artificial and human?"*
>
>   — Smale's 18th problem (from mathematical problems for the 21st century [28]).

Smale formulates the question in connection to foundations of computational mathematics and discusses different models of computation appropriate for the problem [5, 34]. The results in

this paper should be viewed in connection with Smale's 18th problem and the foundations of computation (see §2.2). Our contribution is part of a larger program on foundations of computational mathematics and the Solvability Complexity Index hierarchy [3, 4, 11, 12, 22, 25] established to determine the boundaries of computational mathematics and scientific computing.

2 Why the concept of false structures is needed

To illustrate the concept of false structure and the conjecture we continue with a thought experiment.

Example 2.1 (A thought experiment explaining false structures and Conjecture 1.1)**.** Suppose a human is put in a room in a foreign country with an unknown language. The person is to be trained to recognise the label "fire truck", and in order to do so, she is given a large complex collection of images with different types of fire trucks and images without fire trucks. Each time an image with a fire truck is shown to the person, she hears the foreign language word for fire truck. In particular, the person is trying to learn the function $f : \mathcal{M} \to \{0, 1\}$, where 1 means that there is a fire truck in the image, and $\mathcal{M}$ denotes a large set of images. We will refer to f as the original structure. However, on the set $\mathcal{T} \subset \mathcal{M}$ of images shown in the training process there is a small, but clearly visible, horizontal blue line on each of the images showing a fire truck (this is visualised in Figure 1). On the images without fire trucks there is no line. The question is: will the person think that the foreign language word for fire truck means horizontal blue line, blue line, line or actual fire truck. This is an example of the original structure f (describing fire trucks) and three false structures g_1 (horizontal blue line), g_2 (blue line, not necessary horizontal), and g_3 (line, colour and geometry are irrelevant). All of them could have been learned from the same data. However, the structure or false structure that the person has learned will yield wildly different results on different tests:

(i) (**False structure** g_1): Suppose the person learns the false structure g_1 describing the horizontal blue-line structure that is chosen. Suppose also that the test set of images is chosen such that every image containing a fire truck also has a horizontal blue line. On this test set one will have 100% success, yet the false structure g_1 will give incredible instabilities in several different ways. First, a tiny perturbation in terms of removing the blue line will yield a miss classification. Second, a slight rotation of the blue line would mean wrong output, and a slight change in colour will result in an incorrect decision. Thus, there are at least three types of adversarial attacks that would succeed in reducing the success rate from 100% to 0% on the test set.

(ii) (**False structure** g_3): Suppose it is just the line structure described by the false structure g_3 that is learned. Repeating the experiment in (i) with all images in the test set containing a fire truck also having a small visible green line would yield the same success as in (i) as well as instabilities, however, now rotations of the line would not have an effect nor changes in the colour. Thus, two less adversarial attacks would be successful. If we did the experiment with the false structure g_2, there would have been at least two forms of adversarial attacks available.

(iii) (**The original structure** f): If the actual fire truck structure, described by f, was chosen, one would be as successful and stable as would be expected from a human when given any test set. Note that the original structure f is much more complex and likely harder to learn than the false structures g_1, g_2, g_3.

Motivated by the above thought experiment we can now formally define the original structure and false structures.

Definition 2.2 (The original structure and false structures)**.** Consider $\mathcal{M} \subset \mathbb{R}^d$ and a string of unique (see Remark 2.3) predicates $L = \{\alpha_0, \dots, \alpha_N\}$ on $\mathcal{M}$, where $N \in \mathbb{N}$ such that for each $x \in \mathcal{M}$ there is a unique $\alpha_j \in L$ such that $\alpha_j(x) = \text{true}$. For such x define $f(x) = j$. We say that the pair (f, L) is *the original structure* on $\mathcal{M}$. A *false structure for (f, L) relative to $\mathcal{T} \subsetneq \mathcal{M}$* is

a pair (g, L'), where $L' = \{\beta_0, \ldots, \beta_N\}$ a string of unique predicates with $g : \mathcal{M} \to \{0, \ldots, N\}$ such that $g(x) = j$ iff $\beta_j(x) = \text{true}$. Moreover, $\beta_j \neq \alpha_j$ for all $j \in \{0, \ldots, N\}$ and

$$g(x) = j \text{ iff } f(x) = j \,\forall\, x \in \mathcal{T}, \tag{1}$$

as well as

$$\mathcal{C} = \{x \in \mathcal{M} \backslash \mathcal{T} \mid f(x) = i, \, g(x) = j, \text{ for some } i \neq j\} \neq \varnothing. \tag{2}$$

We say that g is a partial false structure if $\alpha_j \neq \beta_j$ for at least two different $j \in \{0, \ldots, N\}$ (as opposed to all).

Remark 2.3 (Unique predicates). By unique predicates $L = \{\alpha_0, \ldots, \alpha_N\}$ we mean that the support of the characteristic functions induced by the predicates in L do not intersect.

Remark 2.4 (How bad is the false structure?). Note that Definition 2.2 does not consider how 'far' the false structure g is from the original structure f. This is beyond the scope of this paper, however, this can easily be done. For example, suppose $\mathcal{M}$ is equipped with some probability measure P, then assuming $\mathcal{C}$ from (2) is measurable, $P(\mathcal{C})$ would indicate how severe it would be to learn the false structure instead of the original structure.

The motivation behind the idea of a false structure can be understood as follows. Suppose one is interested in learning the original structure (f, L) as in Definition 2.2. In Example 2.1 the list $L = \{\alpha_0, \alpha_1\}$ of predicates are $\alpha_1(x) = x$ demonstrates a fire truck, and $\alpha_0(x) = x$ does not demonstrate a fire truck. In order to learn (f, L) we have a training set $\mathcal{T} \subset \mathcal{M}$. However, if we have a false structure relative to $\mathcal{T}$, as in Example 2.1, (g_1, L') where $L' = \{\beta_0, \beta_1\}$ with

$$\beta_1(x) = x \text{ demonstrates a horizontal blue line,}$$
$$\beta_0(x) = x \text{ does not demonstrate a horizontal blue line,}$$

how do we know that we have not learned (g_1, L') instead? In Example 2.1 there are three different false structures, each with its different instability issues.

Remark 2.5 (Formulation of the predicate). By "x demonstrates a z" (where z was a horizontal blue line above) in the previous predicate we mean that the main object showing in x is z, and that there is only one main object. The word demonstrate is slightly ambiguous, however, for simplicity we use this formulation.

Example 2.1 illustrates the issues in Conjecture 1.1 very simply. Indeed, a simple false structure could give great performance yet incredible instabilities. Let us continue with the thought experiment, however, now we will replace the human in Example 2.1 with a machine, and in particular, we consider the deep learning technique.

Example 2.6. We consider the same problem as in Example 2.1, however, we replace the human by a neural network that we shall train. Indeed, we let $f : \mathcal{M} \subset \mathbb{R}^d \to \{0, 1\}$ be the function deciding if there is a fire truck in the image, where $\mathcal{M}$ is as in Example 2.1. The training set $\mathcal{T} = \{x^1, \ldots, x^r\}$ and test set $\mathcal{C} = \{y^1, \ldots, y^s\}$ consists of images x^j and y^j with and without fire trucks. However, all fire truck images also contain a small blue horizontal line, and there is no blue line in the images without fire trucks. We choose a cost function C, a class of neural networks $\mathcal{NN}$ and approximate the optimisation problem

$$\Psi \in \underset{\Phi \in \mathcal{NN}}{\arg\min}\, C(v, w) \text{ where } v_j = \Phi(x^j), w_j = f(x^j) \text{ for } 1 \leqslant j \leqslant r. \tag{3}$$

The question is now, given that there are three false structures (g_1, L_1'), (g_2, L_2'), (g_3, L_3') (the predicates in L_j' would come from the description in Example 2.1) that would also fit (3):

> *Why should we think that the trained neural network has picked up the correct structure, and not any of the the false structures?*

Remark 2.7 (Simplicity). Note that there could be a minimiser Ψ of (3) such that $\Psi = f$, however, this minimiser may be hard to reach and hence one finds another minimiser $\tilde{\Psi}$ of (3) such that $\tilde{\Psi} = g_1$, say. We will see examples of this phenomenon below.

2.1 Support for the conjecture and how to establish it

Unlike common conjectures in mathematics, Conjecture 1.1 can never be proven with standard mathematical tools. The issue is that all neural networks that are created are done so with inaccurate computations. Thus, actual minimisers are rarely found, if ever, but rather approximations in one form or another. Thus Conjecture 1.1 should be treated more like a postulate in theoretical physics, like 'the speed of light is constant'. One can never establish this with a mathematical proof, however, mathematical theory and experiments can help support the postulate.

Note that there is already an overwhelming amount of numerical evidence that Conjecture 1.1 is true based on the myriad of experiments done over the last five years. Indeed, we have the following documented cases: *(I) Unrecognizable and bizarre structures are labeled as natural images* [21]. Trained successful neural nets classify unrecognizable and bizarre structures as natural images with standard labels with high confidence. Such mistakes would not be possible if the neural network actually captured the correct structure that allows for image recognition in the human brain. *(II) Perturbing one pixel changes the label* [31, 36]. It has been verified that trained and successful networks change the label of the classification even when only one pixel is perturbed. Clearly, the structures in an image that allows for recognition by humans are not affected by a change in a single pixel. *(III) Universal invisible perturbations change more than 90% of the labels* [19, 20]. The DeepFool software [8] demonstrates how a single almost invisible perturbation to the whole test set dramatically changes the failure rate. Different structures in images, allowing for successful human recognition, are clearly not susceptible to misclassification by a single near-invisible perturbation. However, the false structure learned through training is clearly unstable.

There is quite a bit of work on establishing which part of the data is crucial for the decision of the classifier [24, 26, 27, 38, 39]. This is a rather different program compared to establishing our conjecture. Indeed, our conjecture is about the unstable false structures. However, one should not rule out that there might be connections that could help detecting and understanding the false structures.

2.2 Consequences of Conjecture 1.1

The correctness of Conjecture 1.1 may have several consequences both negative and positive.
Negative consequences:

(i) The success of deep learning in classification is not due to networks learning the structures that humans associate with image recognition, but rather that the network picks up unstable false structures in images that are potentially impossible for humans to detect. This means that instability, and hence vulnerability to adversarial attacks, can never be removed until one guarantees that no false structure is learned. This means a potential complete overhaul of modern AI.

(ii) The success is dependent of the simple yet unstable structures, thus the AI does not capture the intelligence of a human.

(iii) Since one does not know which structure the network picks up, it becomes hard to conclude what the neural network actually learns, and thus harder to trust its prediction. What if the false structure gives wrong predictions?

Positive consequences:

(i) Deep learning captures structures that humans cannot detect, and these structures require very little data and computing power in comparison to the true original structures, however, they generalise rather well compared to the original structure. Thus, from an efficiency point of view, the human brain may be a complete overkill for certain classification problems, and deep learning finds a mysterious effective way of classifying.

(ii) The structure learned by deep learning may have information that the human may not capture. This structure could be useful if characterised properly. For example, what if there is

structural information in the data that allows for accurate prediction that the original structure could not do?

Consequences - Smale's 18th problem:

(i) Conjecture 1.1 suggests that there is a fundamental difference between state-of-the-art AI and human intelligence as neural networks based on deep learning learn completely different structures compared to what humans learn. Hence, in view of Smale's 18th problem, correctness of Conjecture 1.1 implies both limitations of AI as well as human intelligence. Indeed, the false unstable structures learned by modern AI limits its abilities to match human intelligence regarding stability. However, in view of the positive consequences mentioned above, correctness of Conjecture 1.1 implies that there is a limitation to human intelligence when it comes to detecting other structures that may provide different information than the structure detected by humans.

3 Establishing Conjecture 1.1 - Do false structures exist in practice?

Our starting point for establishing Conjecture 1.1 is Theorem 3.1 below, for which the proof captures all the three components of the conjecture. We will demonstrate how this happens in actual computations. To introduce some notation, we let $\mathcal{NN}_{\mathbf{N},L}$, with $\mathbf{N} = (N_L, N_{L-1}, \ldots, N_1, N_0)$ denote the set of all L-layer neural networks. That is, all mappings $\Phi : \mathbb{C}^{N_0} \to \mathbb{C}^{N_L}$ of the form

$$\Phi(x) = W_L(\rho(W_{L-1}(\rho(\ldots \rho(W_1(x)))))),$$

with $x \in \mathbb{R}^{N_0}$, where the W_js are affine maps with dimensions given by $\mathbf{N}$, and ρ is a fixed non-linear function acting component-wise on a vector. We consider a binary classifier $f : \mathcal{M} \subset \mathbb{R}^d \to \{0, 1\}$, where $\mathcal{M}$ is some subset. To make sure that we consider stable problems we define the family of well separated and stable sets $\mathcal{S}_\delta^f$ with separation at least $\delta > 0$:

$$\mathcal{S}_\delta^f = \{\{x^1, \ldots, x^r\} \mid \|x^j\| \leqslant 1, \min_{i \neq j} \|x^i - x^j\|_\infty \geqslant \delta, f(x^j + y) = f(x^j) \text{ for } \|y\|_\infty < \delta\}. \quad (4)$$

Moreover, the cost function C used in the training is in

$$\mathcal{CF} := \{C : \mathbb{R}^{rN_L} \times \mathbb{R}^{rN_L} \to \mathbb{R}_+ : C(v, w) = 0 \text{ iff } v = w\}. \quad (5)$$

Theorem 3.1 (Bastounis, Hansen, Vlacic [3])**.** *There is an uncountable family $\mathcal{F}$ of classification functions $f : \mathbb{R}^{N_0} \to \{0, 1\}$ such that for each $f \in \mathcal{F}$ and neural network dimensions $\mathbf{N} = (N_L, N_{L-1}, \ldots, N_0)$ with $N_0, L \geqslant 2$, any $\epsilon > 0$, and any integers s, r with $r \geqslant 3(N_1 + 1) \cdots (N_{L-1} + 1)$, there exist uncountably many non-intersecting training sets $\mathcal{T} = \{x^1, \ldots, x^r\} \in \mathcal{S}_{\varepsilon(r+s)}^f$ of size r (where $\varepsilon(n) := [(4n + 3)(2n + 2)]^{-1}$) and uncountably many non-intersecting classification sets $\mathcal{C} = \{y^1, \ldots, y^s\} \in \mathcal{S}_{\varepsilon(r+s)}^f$ of size s such that we have the following. For every $C \in \mathcal{CF}$ there is a neural net*

$$\Psi \in \underset{\Phi \in \mathcal{NN}_{\mathbf{N},L}}{\arg\min} \, C(v, w) \text{ where } v_j = \Phi(x^j), w_j = f(x^j) \text{ for } 1 \leqslant j \leqslant r \quad (6)$$

such that $\Psi(x) = f(x) \quad \forall x \in \mathcal{T} \cup \mathcal{C}$. However, there exist uncountably many $v \in \mathbb{R}^{N_0}$ such that

$$|\Psi(v) - f(v)| \geqslant 1/2, \qquad \|v - x\|_\infty \leqslant \epsilon \text{ for some } x \in \mathcal{T}. \quad (7)$$

Moreover, there exists a stable neural network

$$\hat{\Psi} \notin \mathcal{NN}_{\mathbf{N},L} \text{ with } \hat{\Psi}(x) = f(x), \, \forall x \in B_{\varepsilon(r+s)}^\infty(\mathcal{T} \cup \mathcal{C}), \quad (8)$$

where $B_{\varepsilon(r+s)}^\infty(\mathcal{T} \cup \mathcal{C})$ denotes the $\varepsilon(r + s)$ neighbourhood, in the l^∞ norm, of $\mathcal{T} \cup \mathcal{C}$.

The message of Theorem 3.1 and its proof [3] (which serves as a basis for §3.1) is summarised below.

(i) (**Success**) The successful neural network learns a false structures that correlates well with the true structure of the problem and hence the great success (100% success on an arbitrarily large test set).

(ii) (**Instability**) The false structure is completely unstable despite the original problem being stable (see $\mathcal{S}_\delta^f$ in (4)). Because of the training process of seeking a minimiser of the optimisation problem, the neural network learns the false structure, and hence becomes completely unstable. Indeed, it will fail on uncountably many instances that are ϵ-away from the training set. Moreover, ϵ can be made arbitrarily small.

(ii) (**Simplicity**) The false structure is very simple and easy to learn. Moreover, paradoxically, there exists another neural network with different dimensions that becomes stable and has the same success rate as the unstable network, however, there is no known way to construct this network.

Theorem 3.1 and its proof provides the starting point in the program on establishing Conjecture 1.1. However, the missing part is to show that: the false structure is learned in practice, it is much easier to learn than the original structure, and that the original structure is very unlikely to be learned in the presence of the false structure in the training set. This is done in the next section.

3.1 Establishing the conjecture: Case 1

We will first start by demonstrating that Conjecture 1.1 is true in the many unique classification cases suggested by Theorem 3.1. A simple example is the infinite class of functions

$$f_a : \mathcal{M} \to \{0, 1\}, \quad \mathcal{M} = [b, 1] \times [0, 1], \quad f_a(x) = \left\lceil \frac{a}{x_1} \right\rceil \mod 2, \tag{9}$$

for $a > 0$ and $0 < b < 1$. Consider the predicates

$$\alpha_0(x) = \left\lceil \frac{a}{x_1} \right\rceil \text{ is odd}, \qquad \alpha_1(x) = \left\lceil \frac{a}{x_1} \right\rceil \text{ is even.}$$

Let $L = \{\alpha_1, \alpha_2\}$ then (f_a, L) is the original structure, as in Definition 2.2. We note that f_a is constant on each of the intervals $\frac{a}{k+1} \leqslant x_1 < \frac{a}{k}$, where $k \in \mathbb{N}$, and hence may be viewed as a very simple classification problem: given x, is $f_a(x) = 1$ or $f_a(x) = 0$? To simplify the learning and analysis further we will assume that $a \in \mathbb{N}$, $a < K$ for some chosen $K \in \mathbb{N}$, and we let $b = \frac{a}{K+1}$. This means that f_a has $K - a$ jump discontinuities on the interval $[b, 1)$. To ensure f_a is stable with respect to perturbations of size $\epsilon > 0$ on its input, we will ensure that each of our samples of f_a lies at least ϵ away from each of these jump discontinuities. Hence, we define

$$\mathcal{S}_\epsilon = \bigcup_{k=a}^{K} \left(\frac{a}{k+1} + \epsilon, \frac{a}{k} - \epsilon \right). \tag{10}$$

choose the samples from the set $\mathcal{S}_\epsilon \times [0, 1]$. This is similar to (4) used in Theorem 3.1. To avoid that $\mathcal{S}_\epsilon$ is a union of empty sets we will always assume that $\epsilon < b^2/(2(a - b))$. It will not be a goal in itself to learn f_a for any x, but rather to learn the right value of f_a within the ϵ-stable region $\mathcal{S}_\epsilon$. Indeed, given that this is a decision problem, inputs close to the boundary are always going to be hard to classify. However, the decision problem stays stable on $\mathcal{S}_\epsilon \times [0, 1]$.

For the learning task we now consider two sets of size r,

$$\mathcal{T}_0^r = \{(x_1^i, 0)\}_{i=1}^r, \quad \mathcal{T}_\delta^r = \{(x_1^i, x_2^i)\}_{i=1}^r, \ x_2^i = \delta f_a(x_1^i), \qquad x_1^i \in \mathcal{S}_\epsilon, \tag{11}$$

where $0 < \delta < \epsilon$. Note that $\mathcal{T}_\delta^r$ gives rise to a false structure as the next proposition shows.
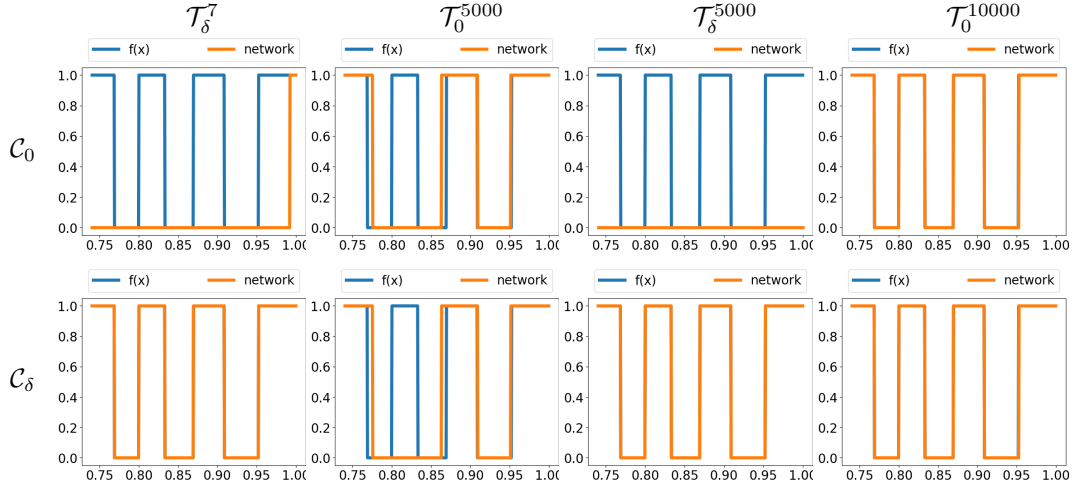
Figure 2: (**Experiment I**) The graphs shows the output of $f_a(x)$ and $\lfloor \sigma(\Psi_i(x)) \rceil$ for x in the two sets $\mathcal{C}_0$ (top row) and $\mathcal{C}_\delta$ (bottom row). The networks Ψ_i, $i = 1, \ldots, 4$ have been trained on the sets $\mathcal{T}_\delta^7$, $\mathcal{T}_0^{5000}$, $\mathcal{T}_\delta^{5000}$ and $\mathcal{T}_\delta^{10000}$, respectively, and are shown from left to right.

Proposition 3.2. *Consider the predicates β_0 and β_1 on $\mathcal{M}$ defined by*

$$\beta_0(x) = x_2 \text{ is } 0, \qquad \beta_1(x) = x_2 \text{ is not } 0.$$

Define $g : \mathcal{M} \to \{0, 1\}$ by $g(x) = 0$ when $\beta_0(x) = \text{true}$ and $g(x) = 1$ when $\beta_1(x) = \text{true}$. Let $L' = \{\beta_0, \beta_1\}$. Then (g, L') is a false structure for (f_a, L) relative to $\mathcal{T}_\delta^r$.

Note that for small δ, g becomes unstable on $\mathcal{T}_\delta^r$. Moreover, the false structure (g, L') appears much simpler than the original structure f_a. In order to train a neural network to learn the original structure f_a we choose the set of networks (architecture) to be $\mathcal{NN}_{\mathbf{N},2}$. That is all fully connected 2-layer networks with dimensions $\mathbf{N} = [1, 4K, 2]$ and non-linear function $\rho : \mathbb{R} \to \mathbb{R}$ being the ReLU function. What is crucial is that the set $\mathcal{NN}_{\mathbf{N},2}$ is rich enough to predict the value of f_a. By prediction we here mean that for a network ϕ the value of $\lfloor \sigma \circ \phi \rceil$, agrees with f_a, where $\sigma(x) = 1/(1 + \exp(-x))$, $x \in \mathbb{R}$ is the sigmoid function, and $\lfloor \cdot \rceil$ means rounding to nearest integer. For the function class $\mathcal{NN}_{\mathbf{N},2}$ above, it is indeed, possible to find such a network whose prediction agrees with f_a on the stable area $\mathcal{S}_\epsilon \times [0, 1]$ in (10). This is formalised in the following statement.

Proposition 3.3 (Existence of stable and accurate network). *Let $\sigma(x) = 1/(1+\exp(-x))$, $x \in \mathbb{R}$ be the sigmoid function, and let $C : \mathbb{R}^r \times \mathbb{R}^r \to \mathbb{R}$ be the cross entropy cost function for binary classification, that is*

$$C(v, w) = \sum_{j=1}^{r} -w_j \log(\sigma(v_j)) - (1 - w_j) \log(1 - \sigma(v_j)). \tag{12}$$

Let f_a be as in (9). Then, for any $\eta > 0$, there exists a two layer neural network $\Psi \in \mathcal{NN}_{\mathbf{N},2}$ with $\mathbf{N} = [1, 4K, 2]$ using the ReLU activation function, such that

$$f_a(x) = \lfloor \sigma(\Psi(x)) \rceil, \quad x \in \mathcal{S}_\epsilon \times [0, 1],$$

where $\lfloor \cdot \rceil$ denotes rounding to the closest integer, and for any subset $\mathcal{T} = \{x^{(1)}, \ldots, x^{(r)}\} \subset \mathcal{S}_\epsilon \times [0, 1]$, $v = \{\Psi(x)\}_{x \in \mathcal{T}}$ and $w = \{f_a(x)\}_{x \in \mathcal{T}}$ we have $C(v, w) \leqslant \eta$.

In particular, Proposition 3.3 states, similarly to the last statement in Theorem 3.1, that there is a stable and accurate network approximating f_a that provides arbitrarily small values of the cost function C. The problem, however, as we will see in the next experiment, is that it is very unlikely to be found it in the presence of the false structure.
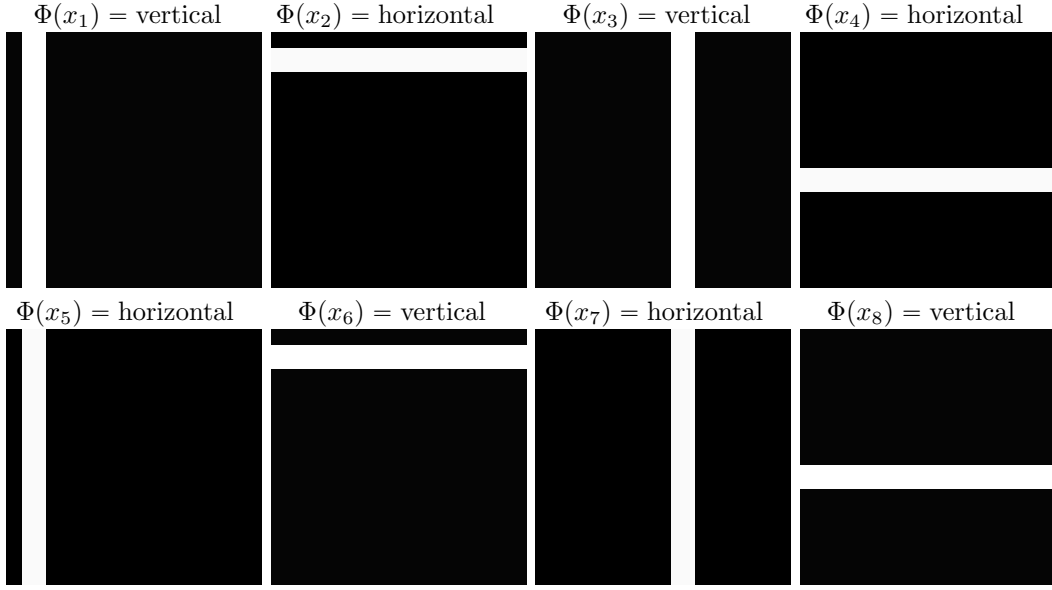
8

Figure 3: (**Experiment II**) Upper figures: Elements $\{x^j\}_{j=1}^4 \subset \tilde{\mathcal{C}}^{b,c}$ ($b = 0.009$, $c = 0.01$), as well as the network classifications $\Phi(x^j)$. Lower figures: Elements $\{x^j\}_{j=5}^8 \in \hat{\mathcal{C}}^{b,c}$ as well as the network classifications $\Phi(x^j)$. Note how the network is completely unstable as the label changes with perturbations that are not visible to the human eye.

3.1.1 Experiment I

The experiment is done as follows. We fixed $a = 20$, $K = 26$, $\epsilon = 10^{-2}$ and $\delta = 10^{-4}$ and trained four neural networks $\Psi_1, \Psi_2, \Psi_3, \Psi_4 \in \mathcal{NN}_{\mathbf{N},2}$ with $\mathbf{N} = [1, 4K, 2]$ and ReLU activation function. The networks Ψ_i, $i = 1, \ldots, 4$ were trained on the sets $\mathcal{T}_\delta^7$, $\mathcal{T}_0^{5000}$, $\mathcal{T}_\delta^{5000}$ and $\mathcal{T}_0^{10000}$, respectively. For the set $\mathcal{T}_\delta^7$, we ensured that the first components x_1^i were located in separate intervals of $\mathcal{S}_\epsilon$. Otherwise it would be infeasible to learn f_a from 7 samples. For the other sets we distributed the first components approximately equally between the disjoint intervals of $\mathcal{S}_\epsilon$. To investigate which structure the networks had learned, we define the two sets,

$$\mathcal{C}_0 = \{(x_1, 0) \mid x_1 \in [b, 1]\} \quad \text{and} \quad \mathcal{C}_\delta = \{(x_1, x_2) \mid x_1 \in [b, 1], x_2 = \delta f_a(x_1)\}. \tag{13}$$

We have plotted $f_a(x)$ and $\lfloor \sigma(\psi_i(x)) \rfloor$ for x in $\mathcal{C}_0$ and $\mathcal{C}_\delta$. The results are displayed in Figure 2 and give the following conclusions. If a network has learned the false structure (g, L'), then $\lfloor \sigma(\Psi_i(x)) \rfloor$ should agree with f_a on $\mathcal{C}_\delta$, while it should be all zero on $\mathcal{C}_0$. On the other hand if the network has not learned (g, L'), then it should have the same output on both $\mathcal{C}_\delta$ and $\mathcal{C}_0$. If the network has learned (f_a, L), it should agree with f_a on both $\mathcal{C}_\delta$ and $\mathcal{C}_0$.

Conclusion (Exp I): Ψ_1, trained on $\mathcal{T}_\delta^7$ (7 samples) learns the false structure (g, L'). Ψ_2, trained on $\mathcal{T}_0^{5000}$, does not learn the original structure (f_a, L) nor the false structure (g, L'). Ψ_3, trained on $\mathcal{T}_\delta^{5000}$, learns the false structure (g, L'). Ψ_4, trained on $\mathcal{T}_0^{10000}$, learns the original structure (f_a, L). Note that the conclusion supports Conjecture 1.1: The false structure (g, L') for (f_a, L) relative to $\mathcal{T}_\delta^r$ is unstable and simple and is learned with only 7 samples. It also gives fantastic success on $\mathcal{C}_\delta$. The original structure (f_a, L) is difficult to learn (10000 samples are needed to succeed, yet 5000 samples are too few). Moreover, when training on $\mathcal{T}_\delta^r$, it is impossible to learn (f_a, L), even with an excessive amount of samples, and the false structure (g, L') is always learned. In particular, despite Proposition 3.3 assuring that the architecture chosen is rich enough to include stable networks that approximate f_a well, and should be found when the cost function is small, the good network is not found in nearly all cases.
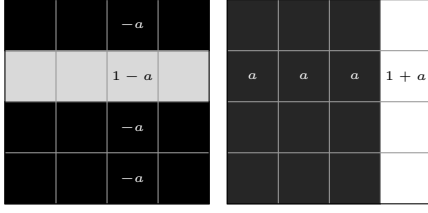
Figure 4a: The colour code for the images defining $\tilde{\mathcal{M}}$. Horizontal stripe: the light coloured pixels have value $1 - a$ and the dark coloured pixels have value $-a$. Vertical stripe: the light coloured pixels have value $1 + a$ and the dark coloured pixels have value a.
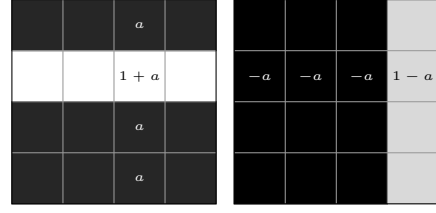
Figure 4b: The colour code for the images defining $\widehat{\mathcal{M}}$. Horizontal stripe: the light coloured pixels have value $1 + a$ and the dark coloured pixels have value a. Vertical stripe: the light coloured pixels have value $1 - a$ and the dark coloured pixels have value $-a$.

3.2 Establishing the conjecture: Case 2

Consider $\mathcal{M}$ to be the collection of 32×32 grey scale images with a 3-pixel wide either horizontal or vertical light stripe on a dark background as shown in Figure 3. The colour code is as follows: -0.01 is black and $1 + 0.01$ is white. Hence, numbers between -0.01 and 0.01 yield variations of black and numbers between $1 - 0.01$ and $1 + 0.01$ give variations of white. Thus, there are slight differences in the black and white colours, however, they are typically not visible to the human eye.

Define the original structure (f, L) on $\mathcal{M}$ with $L = \{\alpha_0, \alpha_1\}$, where

$$\alpha_0(x) = x \text{ has a light horizontal stripe}, \quad \alpha_1(x) = x \text{ has a light vertical stripe}, \qquad (14)$$

and note that the original structure (f, L) is very robust to any small perturbations.

3.2.1 Experiment II

The experiment is inspired by an example from [8], and is done as follows. Define the two sets

$$\tilde{\mathcal{M}} = \{x \in \mathcal{M} \mid x \text{ has a specific colour code described in Figure 4a with } a > 0\},$$

$$\widehat{\mathcal{M}} = \{x \in \mathcal{M} \mid x \text{ has a specific colour code described in Figure 4b with } a > 0\},$$

and notice that both are non-intersecting subsets of $\mathcal{M}$. Next let $\tilde{\mathcal{C}}^{b,c} \subset \tilde{\mathcal{M}}$ and $\widehat{\mathcal{C}}^{b,c} \subset \widehat{\mathcal{M}}$, each contain 1000 elements from their respective sets and where the value of $a \in [b, c]$. Each element of $\tilde{\mathcal{C}}^{b,c}$ and $\widehat{\mathcal{C}}^{b,c}$, is chosen with an equal probability of being a vertical or horizontal stripe, and the value of a is chosen from the uniform distribution on $[b, c]$.

We have trained a neural network Φ on a training set $\mathcal{T}$ which contains exactly the 60 unique elements in $\tilde{\mathcal{M}}$ for which $a = 0.01$. The network Φ has a 100% success rate on the set $\tilde{\mathcal{C}}^{b,c}$, $b = 0.009$, $c = 0.01$, yet its success rate on $\widehat{\mathcal{C}}^{b,c}$ is 0%. As is evident from Figure 3, Φ misclassifies images that look exactly the same as the ones that it successfully classifies. Hence, it is completely unstable and Φ has clearly not learned the original structure (f, L). Thus, a pertinent question is:

What is the false structure that Φ has learned?

To answer the question we begin with the following proposition.

Proposition 3.4. *Consider the predicates β_0 and β_1 on $\mathcal{M}$ defined by*

$$\beta_0(x) = \text{The sum of the pixel values of } x \text{ are } \leqslant 96,$$

$$\beta_1(x) = \text{The sum of the pixel values of } x \text{ are } > 96,$$

and let $L' = \{\beta_0, \beta_1\}$. Let $g(x) = 0$ when $\beta_0(x) = \text{true}$, and $g(x) = 1$ when $\beta_1(x) = \text{true}$. Then (g, L') is a false structure for (f, L) relative to $\tilde{\mathcal{M}}$.

b	c	$\tilde{\mathcal{C}}^{b,c}$	$\widehat{\mathcal{C}}^{b,c}$
0.009	0.010	100%	0.0%
0.008	0.009	100%	0.0%
0.007	0.008	98.7%	0.0%
0.006	0.007	98.5%	0.6%

Table 1: Accuracy of the network Φ, on the two test sets $\tilde{\mathcal{C}}^{b,c}$ and $\widehat{\mathcal{C}}^{b,c}$, for various values of b and c. In all cases the two sets contain 1000 elements.

Note that, contrary to the original structure (f, L), that considers the geometry of the problem, the false structure (g, L) is clearly completely unstable on $\tilde{\mathcal{M}}$. Indeed, a tiny perturbation in the pixel values will change the label. The question is whether it is this false structure that is actually learned by the network Φ. This turns out to be a rather delicate question. Indeed, the fact that we get 100% success rate on $\tilde{\mathcal{C}}^{b,c}$, $b = 0.009, c = 0.01$ and 0% success rate on $\widehat{\mathcal{C}}^{b,c}$ suggests that the false structure that Φ learns is (g, L'). However, by choosing smaller values of c and d, we see from Table 1 that the success rate of Φ on $\tilde{\mathcal{C}}^{b,c}$ decreases, whereas the success rate of $\widehat{\mathcal{C}}^{b,c}$ increases. This implies that this is not entirely the case; if Φ had learned the false structure (g, L') it should have 100% success rate on $\tilde{\mathcal{C}}^{b,c}$ and 0% success rate on $\widehat{\mathcal{C}}^{b,c}$ for all $0 < b < c \leqslant 0.01$. This example illustrates how delicate the task of determining exactly the false structure actually is, even on the simplest examples. The actual false structure learned by Φ is likely not too far from (g, L'), but making this statement mathematically rigorous, as well as a full test is beyond the scope of this paper.

Conclusion (Exp II): The above numerical examples suggest that Φ learns a false structure, however, it may not always be (g, L') from Proposition 3.4. It should also be noted that if the experiments are done with larger test sets $\tilde{\mathcal{C}}^{b,c}$ and $\widehat{\mathcal{C}}^{b,c}$ the conclusion stays the same. The experiments support Conjecture 1.1 as the false structures are simple to learn (only 60 samples needed) and completely unstable. Indeed, tiny perturbations make the network change its label. Moreover, the network Φ that learned the false structures become successful on large test sets.

4 Final conclusion

The correctness of Conjecture 1.1 may have far reaching consequences on how we understand modern AI, and in particular on how to get to the heart of the problem of universal instability throughout neural networks based on deep learning. The conjecture is inspired by Theorem 3.1 and its proof in addition to the many numerical examples demonstrating instabilities in deep learning and suggesting learning of false structures. This paper provides the foundations for a larger program to establish the conjecture fully. However, as we have demonstrated in this paper, Conjecture 1.1 appears to be true even in the simplest cases.

References

[1] J. Adler and O. Öktem. Solving ill-posed inverse problems using iterative deep neural networks. *Inverse Problems*, 33(12):124007, 2017.

[2] V. Antun, F. Renna, C. Poon, B. Adcock, and A. C. Hansen. On instabilities of deep learning in image reconstruction – Does AI come at a cost? *arXiv:1902.05300*, 2019.

[3] A. Bastounis, A. C. Hansen, and V. Vlacic. On computational barriers and paradoxes in estimation, regularisation, learning and computer assisted proofs. *Preprint*, 2019.

[4] J. Ben-Artzi, A. C. Hansen, O. Nevanlinna, and M. Seidel. New barriers in complexity theory: On the solvability complexity index and the towers of algorithms. *Comptes Rendus Mathematique*, 353(10):931 – 936, 2015.

[5] L. Blum, F. Cucker, M. Shub, and S. Smale. *Complexity and Real Computation*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1998.

[6] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2018.

[7] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song. Robust physical-world attacks on deep learning visual classification. *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1625–1634, 2018.

[8] A. Fawzi, S. Moosavi-Dezfooli, and P. Frossard. The robustness of deep networks: A geometrical perspective. *IEEE Signal Processing Magazine*, 34:50–62, 2017.

[9] S. G. Finlayson, J. D. Bowers, J. Ito, J. L. Zittrain, A. L. Beam, and I. S. Kohane. Adversarial attacks on medical machine learning. *Science*, 363(6433):1287–1289, 2019.

[10] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *International conference on learning representations*, 2015.

[11] A. C. Hansen. On the approximation of spectra of linear operators on hilbert spaces. *J. Funct. Anal.*, 254(8):2092 – 2126, 2008.

[12] A. C. Hansen. On the solvability complexity index, the n-pseudospectrum and approximations of spectra of operators. *J. Amer. Math. Soc.*, 24(1):81–124, 2011.

[13] K. H. Jin, M. T. McCann, E. Froustey, and M. Unser. Deep convolutional neural network for inverse problems in imaging. *IEEE Transactions on Image Processing*, 26(9):4509–4522, 2017.

[14] J. Lu, T. Issaranon, and D. Forsyth. Safetynet: Detecting and rejecting adversarial examples robustly. In *IEEE International Conference on Computer Vision*, pages 446–454, 2017.

[15] A. Lucas, M. Iliadis, R. Molina, and A. K. Katsaggelos. Using deep neural networks for inverse problems in imaging: beyond analytical methods. *IEEE Signal Processing Magazine*, 35(1):20–36, 2018.

[16] X. Ma, B. Li, Y. Wang, S. M. Erfani, S. Wijewickrema, G. Schoenebeck, M. E. Houle, D. Song, and J. Bailey. Characterizing adversarial subspaces using local intrinsic dimensionality. In *International Conference on Learning Representations*, 2018.

[17] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

[18] M. T. McCann, K. H. Jin, and M. Unser. Convolutional neural networks for inverse problems in imaging: A review. *IEEE Signal Processing Magazine*, 34(6):85–95, Nov 2017.

[19] S. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard. Universal adversarial perturbations. In *IEEE Conference on computer vision and pattern recognition*, pages 86–94, 07 2017.

[20] S. M. Moosavi Dezfooli, A. Fawzi, and P. Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2574–2582, 2016.

[21] A. Nguyen, J. Yosinski, and J. Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 427–436, 2015.

[22] P. Odifreddi. *Classical Recursion Theory (Volume I)*. North–Holland Publishing Co., Amsterdam, 1989.

[23] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *IEEE Symposium on Security and Privacy*, pages 582–597. IEEE, 2016.

[24] M. T. Ribeiro, S. Singh, and C. Guestrin. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144. ACM, 2016.

[25] H. Rogers, Jr. *Theory of recursive functions and effective computability*. MIT Press, Cambridge, MA, USA, 1987.

[26] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Gradcam: Visual explanations from deep networks via gradient-based localization. In *IEEE International Conference on Computer Vision*, pages 618–626, 2017.

[27] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *International conference on learning representations*, 2013.

[28] S. Smale. Mathematical problems for the next century. *Mathematical Intelligencer*, 20:7–15, 1998.

[29] D. Song, K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, F. Tramer, A. Prakash, and T. Kohno. Physical adversarial examples for object detectors. In *12th {USENIX} Workshop on Offensive Technologies ({WOOT} 18)*, 2018.

[30] R. Strack. AI transforms image reconstruction. *Nature Methods*, 15:309, 04 2018.

[31] J. Su, D. V. Vargas, and K. Sakurai. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 2019.

[32] J. Sun, H. Li, Z. Xu, et al. Deep ADMM-Net for compressive sensing MRI. In *Advances in Neural Information Processing Systems*, pages 10–18, 2016.

[33] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In *International conference on learning representations*, 2014.

[34] A. M. Turing. On Computable Numbers, with an Application to the Entscheidungsproblem. *Proc. London Math. Soc.*, S2-42(1):230, 1936.

[35] A. M. Turing. I.-Computing machinery and intelligence. *Mind*, LIX(236):433–460, 1950.

[36] D. V. Vargas and J. Su. Understanding the one-pixel attack: Propagation maps and locality analysis. *arXiv preprint arXiv:1902.02947*, 2019.

[37] H. Wang and C.-N. Yu. A direct approach to robust deep learning using adversarial networks. In *International Conference on Learning Representations*, 2019.

[38] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. *European conference on computer vision*, pages 818–833, 2014.

[39] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Learning deep features for discriminative localization. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2921–2929, 2016.

[40] B. Zhu, J. Z. Liu, S. F. Cauley, B. R. Rosen, and M. S. Rosen. Image reconstruction by domain-transform manifold learning. *Nature*, 555(7697):487, 03 2018.

5 Appendix

5.1 Proofs

5.1.1 Proof of Proposition 3.2

Proof of Proposition 3.2. First, we notice that the predicates in L' are unique, i.e. x_2 is either 0 or not and hence the classification is unambiguous. Next, we want to see that $g(x) = j \iff f(x) = j$ on $\mathcal{T}_\delta^r$. We have that $x_2 = \delta f_a(x_1)$ for all $x \in \mathcal{T}_\delta^r$. Hence for the first case that we get $g(x) = 0$ when $x_2 = 0$, which means that $f(x) = 0$ since $\delta > 0$. The other case follows from the same argument. Last, we see that $\mathcal{C}$ is non-empty. Let $x_2 = 0$, and choose $x_1 \in [b, 1]$ such that $f_a(x) = 1$ for $x = (x_1, x_2)$. Such an x_1 will always exists since $K > a$. Then $f(x) = 1$ and $g(x) = 0$. $\qquad\square$

5.1.2 Proof of Proposition 3.3

Proof of Proposition 3.3. Let $x = (x_1, x_2) \in [b, 1] \times [0, 1]$, and let

$$\phi_\epsilon^{c,d}(x) = \rho\left(\frac{x_1 - c}{2\epsilon} + \frac{1}{2}\right) - \rho\left(\frac{x_1 - c}{2\epsilon} - \frac{1}{2}\right) - \rho\left(\frac{x_1 - d}{2\epsilon} + \frac{1}{2}\right) + \rho\left(\frac{x_1 - d}{2\epsilon} - \frac{1}{2}\right), \quad (15)$$

where $\epsilon > 0$ and $\rho(t) = \max\{0, t\}$, $t \in \mathbb{R}$ is the ReLU activation function. We start by noticing that $\phi_\epsilon^{c,d}$ is a two layered neural network lying in $\mathcal{NN}_{\mathbf{N},2}$ with $\mathbf{N} = [1, 4, 2]$, where the coefficients in front of x_2 are all zero. Hence the support of $\phi_\epsilon^{c,d}$ equals $[c - \epsilon, d + \epsilon] \times [0, 1]$. We also have that $\phi_\epsilon^{c,d}(x) = 1$ for $x \in [c, d] \times [0, 1]$ and that the range of $\phi_\epsilon^{c,d}$ is $[0, 1]$. Next let

$$\Phi(x) = \sum_{k=a}^{K} \frac{1}{2}((-1)^k + 1)\phi_\epsilon^{c(k),d(k)}(x) \quad (16)$$

with

$$c(k) = \frac{a}{k+1} + \epsilon \quad \text{and} \quad d(k) = \frac{a}{k} - \epsilon$$

be a sum of $K - a + 1$ smaller networks with non-overlapping support in the first variable. We note that the coefficients in front of some of the functions will be zero, and hence could have been removed, but we will not bother to do so. We notice that $\Phi \in \mathcal{NN}_{[1,4(K-a+1)],2} \subset \mathcal{NN}_{\mathbf{N},2}$ with $\mathbf{N} = [1, 4K, 2]$, where the inclusion holds since $K > a$.

Let $x_1 \in (a/(k+1) + \epsilon, a/k - \epsilon)$ and notice that if $k + 1$ is odd, then $\Phi(x) = f_a(x) = 1$, and if $k + 1$ is even, then $\Phi(x) = f_a(x) = 0$. Hence we conclude that $\Phi \in \mathcal{NN}_{\mathbf{N},2}$, with $\mathbf{N} = [1, 4K, 2]$, is a neural network such that $\Phi(x) = f_a(x)$ for all $x \in \mathcal{S}_\epsilon \times [0, 1]$.

Next let $N_1 > 0$ be a constant so that $-\log(\sigma(N_1)) < \eta/r$ and let $N_2 < 0$ be a constant so that $-\log(1 - \sigma(N_2)) < \eta/r$. Furthermore let $N = \max\{N_1, -N_2\}$. Then then

$$\Psi(x) = 2N\Phi(x) - N, \quad x \in [b, 1] \times [0, 1]$$

is a neural network in $\mathcal{NN}_{\mathbf{N},2}$, $\mathbf{N} = [1, 4K, 2]$. Finally notice that for any $\mathcal{T} = \{x^1, \ldots, x^r\} \subset \mathcal{S}_\epsilon \times [0, 1]$ we have $w = \{f(x^i)\}_{i=1}^r \subset \{0, 1\}^r$. Hence if $w_j = 0$, then $\Psi(x^i) \leqslant -N$ and if $w_j = 1$ then $\Psi(x^j) \geqslant N$. letting $v = \{\Psi(x^i)\}_{i=1}^r$, we readily see that

$$C(w, v) = \sum_{j=1}^r -w_j \log(\sigma(v_j)) - (1 - w_j)\sigma(1 - v_j) < \sum_{j=1}^r \eta/r = \eta,$$

moreover, by the same argument as above we see that $\lfloor \sigma(\Psi(x)) \rceil = f(x)$ for all $x \in \mathcal{S}_\epsilon \times [0, 1]$. $\quad\square$

5.1.3   Proof of Proposition 3.4

Proof of Proposition 3.4. The proof is similar to the proof of Proposition 3.2. We again start with the uniqueness of L'. It is clear that the number of pixels is either larger than 96 or smaller or equal. Hence, g is well defined by the predicates β_0 and β_1. Next, we recognise that for images in $\tilde{\mathcal{M}}$ the images with a vertical line sum up to $3 \cdot 32 + 32^2 a$ and for the horizontal lines they sum up to $3 \cdot 32 - 32^2 a$. Hence, for all all images $x \in \tilde{\mathcal{M}}$ we have that those with horizontal lines have values < 96 and for vertical lines > 96 and therefore coincide with the evaluation of the structure f. Last, we see that $\mathcal{C}$ in non-empty. All images in $x \in \mathcal{M}\backslash\tilde{\mathcal{M}}$ with vertical lines have that the pixel sum is 96. Therefore, they also get classified as having a horizontal line, which is then different to the classification by f. $\quad\square$

5.2   Description of training procedures

In this section we intend to describe the training procedure in detail, so that all the experiments becomes reproducible. A complete overview of the code, and the weights of the trained networks described in this paper can be downloaded from `https://github.com/vegarant/false_`
`structures`. Before we start we would also like to point out that in each of the experiments there is an inherit randomness due to random initialization of the network weights, and "on the fly" generation of some of the training and test sets. Hence, rerunning the code, might result in slightly different results. It is beyond the scope of this paper, to investigate how often, and under what circumstances a false structure is learned.

All the code have been implemented in Tensorflow version 1.13.1, and all layers have used Tensorflows default weight initializer, the so called "glorot_uniform" initializer. If not otherwise stated, the default options for other parameters is always assumed.
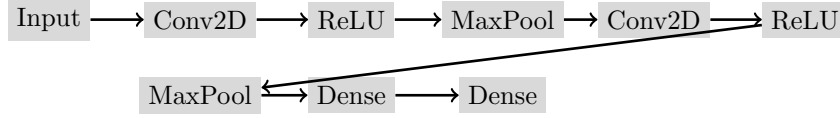
5.2.1   Experiment I

Network architecture. We considered two-layer neural networks with a ReLU activation function between the two layers. The output dimension from the first layer was set to $4K$, and the output dimension of the final layers was 1. An observant reader, who reads the proof of Proposition 3.3, might notice that it would be possible to decrease the output dimension of the first layer. In our initial tests, we did this, but doing so made it substantially harder to learn the true structure f_a.

Training parameters. We trained the networks using the cross entropy loss function for binary classification as described in Proposition 3.3. All networks was trained using the ADAM optimizer, running 30000 epochs. The network trained on the set containing only 7 samples used a batch size of 7, the three other networks used a batch size of 50, with a shuffling of the data samples in each epoch.

Training data. The training sets considered are $\mathcal{T}_\delta^7, \mathcal{T}_0^{5000}, \mathcal{T}_\delta^{5000}$ and $\mathcal{T}_\delta^{10000}$, and is described in the main document. We point out that for each of the training steps, each of these sets where drawn at random. Hence there is some randomness in the experiment itself.

5.2.2   Experiment II

Network architecture. The trained network Φ had the following architecture.

Input $\longrightarrow$ Conv2D $\longrightarrow$ ReLU $\longrightarrow$ MaxPool $\longrightarrow$ Conv2D $\Longrightarrow$ ReLU

MaxPool $\Longleftrightarrow$ Dense $\longrightarrow$ Dense

where the to 2D convolutional layers (Conv2D) had a kernel size of 5, strides equal $(1,1)$, padding equal "same". The first of the convolutional layers had 24 filters, whereas the last had 48 filters. After each of the convolutional layers we used a ReLU activation function. For the max pooling layers a pool size of $(2,2)$ was used, with padding equal "same". After the final max pool layer the output of the layer was reshaped into a vector and feed to a dense layer. The output dimension of the first dense layer was 10, whereas for the last it was 1.

Training parameters. We trained the networks using the cross entropy loss function for binary classification as described in Proposition 3.3. The network was trained using the ADAM optimizer for 10 epochs, with a batch size of 60. We ran the code a few times to capture the exact false structure as described in the main document, as it sometimes capture a slightly different false structure, which does not have exactly 0% success rate on $\widehat{\mathcal{C}}^{b,c}$, $b = 0.009$ and $c = 0.01$.

Training data. The network was trained on the training set $\mathcal{T} \subset \tilde{\mathcal{M}}$ which contains exactly the 60 unique elements in $\tilde{\mathcal{M}}$ for which $a = 0.01$.