Adversarial Reprogramming Revisited

Matthias Englert[*]
University of Warwick
m.englert@warwick.ac.uk

Ranko Lazić[*]
University of Warwick
r.s.lazic@warwick.ac.uk

Abstract

Adversarial reprogramming, introduced by Elsayed, Goodfellow, and Sohl-Dickstein, seeks to repurpose a neural network to perform a different task, by manipulating its input without modifying its weights. We prove that two-layer ReLU neural networks with random weights can be adversarially reprogrammed to achieve arbitrarily high accuracy on Bernoulli data models over hypercube vertices, provided the network width is no greater than its input dimension. We also substantially strengthen a recent result of Phuong and Lampert on directional convergence of gradient flow, and obtain as a corollary that training two-layer ReLU neural networks on orthogonally separable datasets can cause their adversarial reprogramming to fail. We support these theoretical results by experiments that demonstrate that, as long as batch normalisation layers are suitably initialised, even untrained networks with random weights are susceptible to adversarial reprogramming. This is in contrast to observations in several recent works that suggested that adversarial reprogramming is not possible for untrained networks to any degree of reliability.

1 Introduction

Elsayed, Goodfellow, and Sohl-Dickstein [2019] proposed *adversarial reprogramming*: given a neural network $\mathcal{N}$ which performs a task $F : X \to Y$ and given an adversarial task $G : \widetilde{X} \to \widetilde{Y}$, repurpose $\mathcal{N}$ to perform G by finding mappings $h_{\text{in}} : \widetilde{X} \to X$ and $h_{\text{out}} : Y \to \widetilde{Y}$ such that $G \approx h_{\text{out}} \circ F \circ h_{\text{in}}$. They focused on a setting where F and G are image classification tasks, X consists of large images, $\widetilde{X}$ consists of small images, and h_{in} and h_{out} are simple and computationally inexpensive: $h_{\text{in}}(\widetilde{x}) = p + \widetilde{x}$ draws the input $\widetilde{x}$ at the centre of the *adversarial program* p, and h_{out} is a hard coded mapping from the class labels Y to the class labels $\widetilde{Y}$. Then the challenge of adversarial reprogramming is to find p such that $h_{\text{out}} \circ F \circ h_{\text{in}}$ approximates well the adversarial task G.

Remarkably, Elsayed et al. [2019] showed that a range of realistic neural networks can be adversarially reprogrammed to perform successfully several tasks. They considered six architectures trained on ImageNet [Russakovsky, Deng, Su, Krause, Satheesh, Ma, Huang, Karpathy, Khosla, Bernstein, Berg, and Fei-Fei, 2015] and found adversarial programs that achieve very good accuracies on a counting task, MNIST [LeCun, Bottou, Bengio, and Haffner, 1998], and CIFAR-10 [Krizhevsky, 2009]. In addition to the basic setting, they investigated limiting the visibility of the adversarial program by restricting its size or scale, or even concealing both the input and the adversarial program by hiding them within a normal image from ImageNet, and obtained good results.

Adversarial reprogramming can be seen as taking the crafting of adversarial examples [Biggio, Corona, Maiorca, Nelson, Šrndić, Laskov, Giacinto, and Roli, 2013, Szegedy, Zaremba, Sutskever, Bruna, Erhan, Goodfellow, and Fergus, 2014] to a next level. Like the universal adversarial perturbation of Moosavi-Dezfooli, Fawzi, Fawzi, and Frossard [2017], a single adversarial program is combined with every input from the adversarial task, but in contrast to the former where the goal is to cause

[*]Equal contribution.

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

natural images to be misclassified with high probability, adversarial reprogramming seeks a high accuracy of classification for the given adversarial task.

Adversarial reprogramming is also related to transfer learning [Raina, Battle, Lee, Packer, and Ng, 2007, Mesnil, Dauphin, Glorot, Rifai, Bengio, Goodfellow, Lavoie, Muller, Desjardins, Warde-Farley, Vincent, Courville, and Bergstra, 2011], however with important differences. Whereas transfer learning methods use the knowledge obtained from one task as a base for learning to perform another, and allow model parameters to be changed for the new task, in adversarial reprogramming the new task may bear no similarity with the old, and the model cannot be altered but has to be manipulated through its input.

The latter features are suggestive of potential nefarious uses of adversarial reprogramming, and Elsayed et al. [2019] list several, such as repurposing computational resources to perform a task which violates the ethics code of system providers. However, its further explorations have demonstrated utility for virtuous deployments to medical image classification [Tsai, Chen, and Ho, 2020], natural language processing [Hambardzumyan, Khachatrian, and May, 2021, Neekhara, Hussain, Dubnov, and Koushanfar, 2019, Neekhara, Hussain, Du, Dubnov, Koushanfar, and McAuley, 2022], molecule toxicity prediction [Vinod, Chen, and Das, 2020], and time series classification [Yang, Tsai, and Chen, 2021]. In the last three works, adversarial reprogramming was achieved between different domains, e.g. repurposing neural networks trained on ImageNet to perform classification of DNA sequences, and of natural language sentiments and topics.

In spite of the variety of fruitful applications, it is still largely a mystery when adversarial reprogramming is possible and why. In the only work in this direction, Zheng, Feng, Xia, Jiang, Demontis, Pintor, Biggio, and Roli [2021] proposed the alignment $\|g\|_1/(1/n \sum_i \|g_i\|_1)$ of the gradients g_i of the inputs from the adversarial task with their average g as the main indication of whether adversarial reprogramming will succeed. However, their experiments did not show a significant correlation between accuracy after reprogramming and the gradient alignment before reprogramming. The correlation was statistically significant with the gradient alignment after reprogramming, but that is unsurprising and it is unclear how to use it for predicting success.

Specifically, a central question on adversarial reprogramming is:

Can neural networks with random weights be adversarially reprogrammed, and more generally, how does training impact adversarial reprogrammability?

First, addressing this question is important for assessing scope of two claims made in the literature:

***"[Adversarial] reprogramming usually fails when applied to untrained networks"* [Zheng et al., 2021].** In addition to networks trained on ImageNet, Elsayed et al. [2019] experimented using the same architectures untrained, i.e. with random weights, and obtained generally poor results. Similarly, the experimental results of Neekhara et al. [2019] and Zheng et al. [2021] for random networks are significantly worse than their experimental results for trained networks. However, they remarked that this was surprising given the known richness of random networks (see e.g. He, Wang, and Hopcroft [2016c], Lee, Bahri, Novak, Schoenholz, Pennington, and Sohl-Dickstein [2018]), and that it was possibly due to simple reasons such as poor scaling of the random weights.

***"The original task the neural networks perform is important for adversarial reprogramming"* [Elsayed et al., 2019].** Nevertheless a number experimental results including those of Tsai et al. [2020], Neekhara et al. [2022], Vinod et al. [2020], Yang et al. [2021], Zheng et al. [2021] demonstrated successful adversarial reprogramming between tasks that are seemingly unrelated (e.g. repurposing networks trained on ImageNet to act as HCL2000 [Zhang, Guo, Chen, and Li, 2009] classifiers) or from different domains, so the interaction between the original and adversarial tasks remains unclear.

Second, the question above is important because of implications for the following two applied considerations:

Disentangling architecture and training as factors in adversarial reprogrammability. Clarifying the respective bearings on adversarial reprogramming success of the network architecture and of the task (if any) it was trained for would improve decision making, either to maximise the success in beneficial scenarios or to minimise it in detrimental ones.

Managing the cost of adversarial reprogramming. Understanding when training the network is not essential, and when training for longer does not help or even hinders adversarial reprogrammability, should make it possible to control better the economic and environmental costs.

1.1 Our contributions

We initiate a theoretical study of adversarial reprogramming. In it we focus on two-layer neural networks with ReLU activation, and on adversarial tasks given by the Bernoulli distributions of Schmidt, Santurkar, Tsipras, Talwar, and Mądry [2018] over hypercube vertices. The latter are binary classification data models in which the two classes are represented by opposite hypercube vertices, and when sampling a data point for a given class, we flip each coordinate of the corresponding class vertex with a certain probability. This probability is a parameter that controls the difficulty of the classification task. These data models are inspired by the MNIST dataset because MNIST images are close to binary (many pixels are almost fully black or white). The remaining parameters are the radius of the hypercube, the input dimension of the neural network, and its width.

We prove that, in this setting, for networks with random weights, adversarial programs exist that achieve arbitrarily high accuracy on the Bernoulli adversarial tasks. This holds for a wide variety of parameter regimes, provided the network width is no greater than its input dimension. The adversarial programs we construct depend on the weights of the network and on the class vertices (i.e. the direction) of the Bernoulli data model, and their Euclidean length is likely to be close to the square root of the input dimension. We present these results in Section 2.

We also prove that, in the same setting, training the network on orthogonally separable datasets can cause adversarial reprogramming to fail. Phuong and Lampert [2021] recently showed that, under several assumptions, training a two-layer ReLU network on such datasets by gradient flow makes it converge to a linear combination of two maximum-margin neurons; and subsequently Wang and Pilanci [2022] obtained in a different manner the same conclusion under the same assumptions. We provide a simpler proof of a significantly stronger result: we show that the assumptions in Phuong and Lampert [2021] and Wang and Pilanci [2022] of small initialisation, and of positive and negative support examples spanning the whole space, are not needed; and we generalise to the exponential loss function as well as the logistic one. We then observe that, for any Bernoulli data model whose direction is in a half-space of the difference of the maximum-margin neurons, and for any adversarial program, the accuracy tends to $1/2$ (i.e. approaches guessing) under a mild assumption on the growth rate of the difficulty of the data model. We present these results in Section 3.

Both for the neworks with random weights and for the networks trained to infinity on orthogonally separable datasets, we then show that similar theoretical results can be obtained with a different kind of adversarial task, namely those given by the Gaussian distributions also of Schmidt et al. [2018]. The latter are mixtures of two spherical multivariate Gaussians, one per data class. Please see Appendix G.

In the experimental part of our work, we demonstrate that, as long as batch normalisation layers are suitably initialised, even untrained networks with random weights are susceptible to adversarial reprogramming. Both the random weights and the batch normalisation layers are kept fixed throughout the finding of adversarial programs and their evaluation. Our experiments are conducted with six realistic network architectures and MNIST as the adversarial task. We investigate two different schemes to combine input images with adversarial programs: replacing the centre of the program by the image as was done by Elsayed et al. [2019], and scaling the image to the size of the program and then taking a convex combination of the two. Each of the two schemes has a ratio parameter, and we explore their different values. We find that the second scheme gives better results in our experiments, and that for some choices of the ratio parameter, the accuracies on the test set across all six architectures are not far below what Elsayed et al. [2019] reported for networks trained on ImageNet. Please see Section 4.

We conduct the same experiments also on the more challenging Fashion-MNIST [Xiao, Rasul, and Vollgraf, 2017] and Kuzushiji-MNIST [Clanuwat, Bober-Irizar, Kitamoto, Lamb, Yamamoto, and Ha, 2018] datasets, and obtain broadly similar results, however with lower test accuracies in several cases; they are reported in Appendix H.1.

For a further discussion of relations with other works, please see Appendix A.

We conclude the paper in Section 5, where we discuss limitations of our work and suggest directions for future work.

2 Random networks

Basic notations. We write: $[n]$ for the set $\{1, \ldots, n\}$, $\|\boldsymbol{v}\|$ for the Euclidean length of a vector $\boldsymbol{v}$, $\angle(\boldsymbol{v}, \boldsymbol{v}')$ for the angle between $\boldsymbol{v}$ and $\boldsymbol{v}'$, $\mathbb{H}^d$ for the d-dimensional unit hypercube $\{\pm 1/\sqrt{d}\}^d$, and $\mathbb{S}^{d-1}$ for the d-dimensional unit sphere $\{\boldsymbol{v} \in \mathbb{R}^d \mid \|\boldsymbol{v}\| = 1\}$.

Two-layer ReLU networks. We consider two-layer neural networks $\mathcal{N}$ with the ReLU activation. We write d for the input dimension, k for the width, $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_k \in \mathbb{R}^d$ for the weights of the first layer, and $a_1, \ldots, a_k \in \mathbb{R}$ for the weights of the second layer. For an input $\boldsymbol{x} \in \mathbb{R}^d$, the output is thus

$$\mathcal{N}(\boldsymbol{x}) \coloneqq \sum_{j=1}^{k} a_j \psi(\boldsymbol{w}_j^\top \boldsymbol{x}) \ ,$$

where $\psi(u) = \max\{u, 0\}$ is the ReLU function.

Random weights. In this section, we assume that the weights in $\mathcal{N}$ are random as follows:

- each $\boldsymbol{w}_j$ consists of d independent centred Gaussians with variance $1/d$, and
- each a_j is independently uniformly distributed in $\{\pm 1/\sqrt{k}\}$.

This distribution is as in Bubeck, Cherapanamjeri, Gidel, and Tachet des Combes [2021], and standard for theoretical investigations. It is similar to He's initialisation [He, Zhang, Ren, and Sun, 2015], with the second layer discretised for simplicity.

The variances of the weights are such that, for any input $\boldsymbol{x} \in \mathbb{R}^d$ of Euclidean length $\sqrt{d}$, each $\boldsymbol{w}_j^\top \boldsymbol{x}$ is a standard Gaussian, and for large widths k the distribution of $\mathcal{N}(\boldsymbol{x})$ is close to centred Gaussian with variance $1/2$.

Bernoulli data models. Adapting from Schmidt et al. [2018], given a hypercube vertex $\boldsymbol{\phi} \in \mathbb{H}^d$, a radius $\rho > 0$, and a class bias parameter $0 < \tau \leq 1/2$, we define the $(\boldsymbol{\phi}, \rho, \tau)$-Bernoulli distribution over $(\boldsymbol{x}, y) \in \rho\mathbb{H}^d \times \{\pm 1\}$ as follows:

- first draw the label y uniformly at random from $\{\pm 1\}$,
- then sample the data point $\boldsymbol{x}$ by taking $y\rho\boldsymbol{\phi}$ and flipping the sign of each coordinate independently with probability $1/2 - \tau$.

These binary classification data models on hypercube vertices are inspired by the MNIST dataset [LeCun et al., 1998]. The class bias parameter τ controls the difficulty of the classification task, which increases as τ tends to zero, i.e. as $1/\tau$ tends to infinity.

In this section and the next, we consider adversarial tasks that are $(\boldsymbol{\phi}, \rho, \tau)$-Bernoulli data models, and investigate variations of the parameters $\boldsymbol{\phi}$, ρ and τ.

Adversarial program. In this section, we assume that $k \leq d$, i.e. the network width is no greater than the input dimension, and we define an adversarial program $\boldsymbol{p}$ which depends on the weights of the network $\mathcal{N}$ and on the direction $\boldsymbol{\phi}$ of the Bernoulli data model.

With probability 1, for all $j \in [k]$, we have that $a_j \boldsymbol{w}_j^\top \boldsymbol{\phi} \neq 0$. Let us write K^+ for $\{j \in [k] \mid a_j \boldsymbol{w}_j^\top \boldsymbol{\phi} > 0\}$, and K^- for $\{j \in [k] \mid a_j \boldsymbol{w}_j^\top \boldsymbol{\phi} < 0\}$. Then, for all $j \in [k]$, let:

$$\boldsymbol{p}'_j = \begin{cases} 0 & \text{if } j \in K^+, \\ -\sqrt{d/|K^-|} & \text{if } j \in K^-. \end{cases}$$

Since $k \leq d$, with probability 1, the weights vectors $\boldsymbol{w}_j$ are linearly independent, i.e. the $k \times d$ matrix $\boldsymbol{W}$ whose rows are $\boldsymbol{w}_j$ has a positive smallest singular value $s_{\min}(\boldsymbol{W})$. Hence $\boldsymbol{p} \in \mathbb{R}^d$ exists

such that

$$\boldsymbol{p}' = \boldsymbol{W}\boldsymbol{p} \quad \text{and} \quad \frac{\|\boldsymbol{p}'\|}{s_{\max}(\boldsymbol{W})} \leq \|\boldsymbol{p}\| \leq \frac{\|\boldsymbol{p}'\|}{s_{\min}(\boldsymbol{W})} \ . \tag{1}$$

The neurons in K^+ can be thought of as "helpful" for the adversarial task, and those in K^- as "unhelpful". This definition of an adversarial program $\boldsymbol{p}$ ensures that its effect is to introduce as first-layer biases the entries of the vector $\boldsymbol{p}'$. They are 0 (i.e. do nothing) for every "helpful" neuron, and the negative value $-\sqrt{d/|K^-|}$ (i.e. reduce the contribution to the network output) for every "unhelpful" neuron.

From the inequalities in (1), we can expect $\|\boldsymbol{p}\| \approx \sqrt{d}$ if $k = o(d)$ (see Appendix B for details).

Expected reprogramming accuracy. The accuracy for an adversarial task $\mathcal{D}$ is the probability that the sign of the output of the reprogrammed network conforms to the input label, i.e.

$$\mathbb{P}_{(\boldsymbol{x},y)\sim\mathcal{D}}\{y\,\mathcal{N}(\boldsymbol{p}+\boldsymbol{x}) > 0\} \ .$$

Our main result in this section is that, for sufficiently large input dimensions d, and under mild restrictions on the growth rates of the network width k, the radius ρ and the difficulty $1/\tau$ of the Bernoulli data model, in expectation over the random network weights, the reprogramming accuracy is at least $(1 - C_1\gamma)(1 - \gamma^\dagger)$, which by tuning the probability parameters γ and $\gamma^\dagger$ can be arbitrarily close to 100%. The growth rate restrictions indicate that networks with smaller widths can be reprogrammed for tasks with larger radii, but that networks with larger widths can be reprogrammed for tasks that are more difficult. The proof proceeds by analysing concentrations of the underlying distributions and involves establishing a theorem that bounds the reprogrammed network output; the details can be found in Appendix C.

Corollary 1. *Suppose that*

$$k = \Theta(d^{\eta_{(k)}}) \ , \quad \rho = O(d^{\eta_{(\rho)}}) \ , \quad 1/\tau = O(d^{\eta_{(\tau)}}) \quad \textit{and} \quad 1/\tau = \omega_d(1) \ ,$$

where $\eta_{(k)}, \eta_{(\rho)}, \eta_{(\tau)} \in [0,1]$ *are arbitrary constants that satisfy*

$$\eta_{(\rho)} < 1 - \eta_{(k)}/2 \quad \textit{and} \quad \eta_{(\tau)} < \eta_{(k)}/2 \ .$$

Then, for sufficiently large input dimensions d*, the expected accuracy of the adversarially reprogrammed network* $\mathcal{N}$ *on the* $(\boldsymbol{\phi}, \rho, \tau)$*-Bernoulli data model is arbitrarily close to* 100%.

3 Implicit bias

Orthogonally separable dataset. In this section, we consider training the network on a binary classification dataset $S = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\} \subseteq \mathbb{R}^d \times \{\pm 1\}$ which is orthogonally separable [Phuong and Lampert, 2021], i.e. for all $i, i' \in [n]$ we have:

$$\boldsymbol{x}_i^\top \boldsymbol{x}_{i'} > 0 \quad \text{if} \quad y_i = y_{i'} \ , \quad \text{and} \quad \boldsymbol{x}_i^\top \boldsymbol{x}_{i'} \leq 0 \quad \text{if} \quad y_i \neq y_{i'} \ .$$

In other words, every data point can act as a linear separator, although some data points from the opposite class may be exactly orthogonal to it.

Gradient flow with exponential or logistic loss. For two-layer ReLU networks with input dimension d and width k as before (but without the assumption $k \leq d$, which is not needed in this section), we denote the vector of all weights by

$$\boldsymbol{\theta} \coloneqq (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_k, a_1, \ldots, a_k) \in \mathbb{R}^{k(d+1)} \ ,$$

and we write $\mathcal{N}_{\boldsymbol{\theta}}$ for the network whose weights are the coordinates of the vector $\boldsymbol{\theta}$.

The empirical loss of $\mathcal{N}_{\boldsymbol{\theta}}$ on S is $\mathcal{L}(\boldsymbol{\theta}) \coloneqq \sum_{i=1}^n \ell(y_i \mathcal{N}_{\boldsymbol{\theta}}(\boldsymbol{x}_i))$, where ℓ is either the exponential $\ell_{\exp}(u) = e^{-u}$ or the logistic $\ell_{\log}(u) = \ln(1 + e^{-u})$ loss function.

A trajectory of gradient flow is a function $\boldsymbol{\theta}(t) : [0, \infty) \to \mathbb{R}^{k(d+1)}$ that is an arc, i.e. it is absolutely continuous on every compact subinterval, and that satisfies the differential inclusion

$$\frac{\mathrm{d}\boldsymbol{\theta}}{\mathrm{d}t} \in -\partial\mathcal{L}(\boldsymbol{\theta}(t)) \quad \text{for almost all} \quad t \in [0, \infty) \ ,$$

where $\partial\mathcal{L}$ denotes the Clarke subdifferential [Clarke, 1975] of the locally Lipschitz function $\mathcal{L}$.

Gradient flow is gradient descent with infinitesimal step size. We work with the Clarke subdifferential in order to handle the non-differentiability of the ReLU function at zero: $\partial\psi(0)$ is the whole interval $[0, 1]$. At points of continuous differentiability, the Clarke subdifferential amounts to the gradient, e.g. $\partial\mathcal{L}(\boldsymbol{\theta}) = \{\nabla\mathcal{L}(\boldsymbol{\theta})\}$. For some further background, see Appendix D.

Initialisation of network weights. In this section, we assume that the initialisation is

balanced: for all $j \in [k]$, at time $t = 0$ we have $|a_j| = \|\boldsymbol{w}_j\| > 0$; and

live: for both signs $s \in \{\pm 1\}$ there exist $i_s \in [n]$ and $j_s \in [k]$ such that $y_{i_s} = s$ and at time $t = 0$ we have $y_{i_s} a_{j_s} \psi(\boldsymbol{w}_{j_s}^\top \boldsymbol{x}_{i_s}) > 0$.

The balanced assumption has featured in previous work (see e.g. Phuong and Lampert [2021], Lyu, Li, Wang, and Arora [2021]). It ensures that it remains to hold throughout the training, and that the signs of the second-layer weights a_j do not change (see the proof of Theorem 2). The live assumption (present probabilistically in Phuong and Lampert [2021]) is mild: it states that at least one positively initialised neuron is active for at least one positive input, and the same for negative ones.

Convergence of gradient flow. Our main result in this section establishes that the early phase of training necessarily reaches a point where the empirical loss is less than $\ell(0)$, which implies that then every input is classified correctly by the network. Perhaps surprisingly, no small initialisation assumption is needed, however the proof makes extensive use of orthogonal separability of the dataset (see Appendix F, which contains all proofs for this section).

Theorem 2. *There exists a time t_0 such that $\mathcal{L}(\boldsymbol{\theta}(t_0)) < \ell(0)$.*

This enables us to apply to the late phase recent results of Lyu and Li [2020], Ji and Telgarsky [2020], Lyu et al. [2021] and obtain the next corollary, which is significantly stronger than the main result of Phuong and Lampert [2021], extending it to exponential loss, and showing that assumptions of small initialisation, and of positive and negative support examples spanning the whole space, are not needed. The corollary establishes that each neuron converges to one of three types: a scaling of the maximum-margin vector for the positive data class, a scaling of the maximum-margin vector for the negative data class, or zero. The two maximum-margin vectors are defined as follows: for both signs $s \in \{\pm 1\}$, let $I_s \coloneqq \{i \in [n] \mid y_i = s\}$, and let $\boldsymbol{v}_s$ be the unique minimiser of the quadratic problem

$$\text{minimise} \quad \frac{1}{2}\|\boldsymbol{v}\|^2 \quad \text{subject to} \quad \forall i \in I_s : \boldsymbol{v}^\top \boldsymbol{x}_i \geq 1 \ .$$

That a trajectory $\boldsymbol{\theta}(t)$ converges in direction to a vector $\widetilde{\boldsymbol{\theta}}$ means $\lim_{t\to\infty} \boldsymbol{\theta}(t)/\|\boldsymbol{\theta}(t)\| = \widetilde{\boldsymbol{\theta}}/\|\widetilde{\boldsymbol{\theta}}\|$.

Corollary 3. *As the time tends to infinity, we have that the empirical loss converges to zero, the Euclidean norm of the weights converges to infinity, and the weights converge in direction to some $\boldsymbol{\theta}$ such that for all $j \in [k]$ we have $|a_j| = \|\boldsymbol{w}_j\|$, and if $a_j \neq 0$ then*

$$\frac{\boldsymbol{w}_j}{\|\boldsymbol{w}_j\|} = \frac{\boldsymbol{v}_{\operatorname{sgn}(a_j)}}{\sum_{\operatorname{sgn}(a_{j'})=\operatorname{sgn}(a_j)} a_{j'}^2} \ .$$

Thanks to homogeneity, the sign of the network output does not depend on the norm of the weights, only on their direction. Examining networks whose weights are directional limits as in Corollary 3 is therefore informative of consequences for adversarial reprogramming of long training. The following result tells us that, for any Bernoulli data model whose direction is in a half-space of the difference of the maximum-margin vectors, and for any adversarial program, the accuracy tends to $1/2$ provided that the difficulty $1/\tau$ of the data model increases slower than the square root of the input dimension d. The latter assumption is considerably weaker than in the results of Schmidt et al. [2018] on the Bernoulli data model, where the bound is in terms of the fourth root. The statement also tells us that this failure cannot be fixed by choosing in advance a different mapping from the class labels of the original task to the class labels of the adversarial task. Since we consider binary classification tasks here, the mapping can be represented by a multiplier $m \in \{\pm 1\}$.

Proposition 4. *Suppose network weights $\boldsymbol{\theta}$ are is in Corollary 3, class label mapping $m \in \{\pm 1\}$ is arbitrary, data model $\mathcal{D}$ is any $(\boldsymbol{\phi}, \rho, \tau)$-Bernoulli distribution such that $m \cos\angle(\boldsymbol{v}_1 - \boldsymbol{v}_{-1}, \boldsymbol{\phi}) < 0$, and adversarial program $\boldsymbol{p}$ is arbitrary. Then we have that*

$$\mathbb{P}_{(\boldsymbol{x},y)\sim\mathcal{D}}\{m\, y\, \mathcal{N}_{\boldsymbol{\theta}}(\boldsymbol{p} + \boldsymbol{x}) > 0\} \ \leq \ \frac{1}{2} + \frac{1}{2}\, e^{-2d\tau^2 \cos^2\angle(\boldsymbol{v}_1 - \boldsymbol{v}_{-1}, \boldsymbol{\phi})} \ .$$

4 Experiments

Network architectures and initialisation. Our experiments[2] are conducted using the following six network architectures: ResNet-50 [He, Zhang, Ren, and Sun, 2016a], ResNet-50V2, ResNet-101V2, ResNet-152V2 [He, Zhang, Ren, and Sun, 2016b], Inception-v3 [Szegedy, Vanhoucke, Ioffe, Shlens, and Wojna, 2016], and EfficientNet-B0 [Tan and Le, 2019].

We use the networks exactly as implemented in Keras in TensorFlow 2.8.1 including the method for randomly initialising the trainable weights. For biases, this means they are initialised with 0. For all other trainable weights, mostly, the Glorot uniform initialiser [Glorot and Bengio, 2010] is used in this implementation. EfficientNet is an exception, where many layers instead use a truncated normal distribution that has mean 0 and standard deviation $\sqrt{2/\text{number of output units}}$.

All the networks we experiment with (ResNet-50, ResNet-50V2, ResNet-101V2, ResNet-152V2, Inception-v3, and EfficientNet-B0) involve batch normalisation layers [Ioffe and Szegedy, 2015]. Every such layer maintains a moving mean and a moving variance based on batches it has seen during training. The inputs are then normalised accordingly: they are shifted by the recorded mean and scaled by the inverse of the recorded variance. Note that the moving mean and variance values are not trainable, i.e., they are not subject to updates by the optimiser during training. During inference, the moving mean and variance values are no longer updated, and the normalisation is performed based on the last values recorded during training.

Crucially, in the default implementation of these networks, these moving mean and moving variance values are initialised as 0 and 1, respectively. Therefore, an untrained network initialised in this way will behave as if the batch normalisation layers were not present.

To obtain more sensible random networks, i.e., ones that can still make use of batch normalisation, we initialise the moving mean and variance of batch normalisation layers differently. We generate a batch of 50 random images (each pixel value is chosen independently and uniformly at random in the allowed range). This single batch is then fed through the random network and each batch normalisation layer records the mean and variance values it sees at its input (and normalises its output accordingly). The trainable weights of the network are not changed during this process.

We should note that, in addition to the moving mean and variance, batch normalisation layers can have *trainable* weights, by means of which the output mean and variance can be tuned. Specifically, such a layer may have trainable weights γ and β, and will scale its otherwise normalised output by γ and shift it by β. If present, these trainable weights are initialised as 1 and 0 respectively, and hence have no effect. Our initialisation procedure does not modify these trainable weights and they are therefore not used in our random networks.

For each network, after randomly initialising its weights as set out in Section 4 and its batch normalisation layers as described above, we keep it completely fixed: neither its weights nor its batch normalisation layers (i.e., their moving means and variances, and their weights if any) change in any way.

Combining input images with adversarial programs. Our adversarial programs are colour images whose sizes match the expected input size of the networks. This is 224×224 for all networks except Inception-v3, where it is 299×299.

We use two different schemes to combine input images with the adversarial program. The first scheme, used by Elsayed et al. [2019], is to take the adversarial program and overwrite a portion of it by the input image. We do this in such a way that the input image is, up to rounding, centred in the adversarial program. We can vary the construction by scaling the input image up or down before applying this procedure. In particular, we use a parameter $r \in [0, 1]$ and scale the input image, using bilinear interpolation, in such a way that r times the width of the adversarial program is equal to the width of the scaled image. That we focus on the width is not important because all our inputs and programs are square. An illustration is shown in Figure 1.

Our second scheme involves scaling the image to the same size as the adversarial program and then taking a convex combination of the two. We use a parameter $v \in [0, 1]$ to specify how much weight

[2] We are making code to run the experiments available at `https://github.com/englert-m/adversarial_reprogramming`.

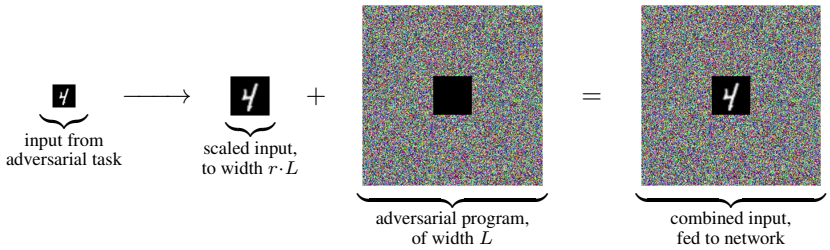

Figure 1: Scheme 1 for combining input images with adversarial programs. In this example, the width and height of the adversarial program are 224 and the parameter r equals $2^{-20/9} \approx 0.214$, so the input image is scaled to width and height $r \cdot 224$ rounded, which is 48.

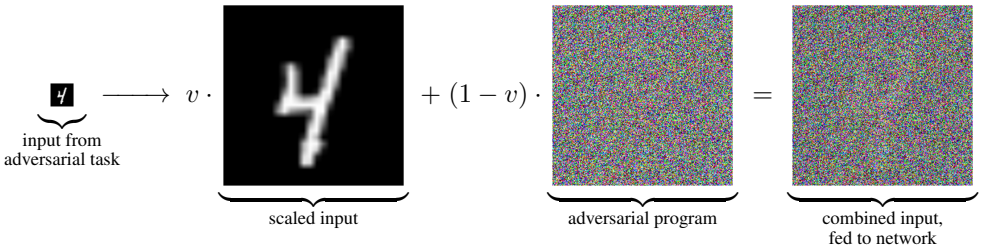

Figure 2: Scheme 2 for combining input images with adversarial programs. In this example, the sizes of the scaled input image and the adversarial program are 224×224. The parameter v equals $2^{-40/9} \approx 0.046$, so the weight of the input image in the convex combination with the adversarial program is approximately 4.6%, which makes it faintly visible.

the input image should get in this convex combination. Specifically, the combined image is obtained by calculating $v \cdot I + (1 - v) \cdot P$, where I is the input image and P is the adversarial program. An illustration is shown in Figure 2.

Adversarial task dataset. Elsayed et al. [2019] evaluate adversarial reprogramming on random networks using the MNIST [LeCun et al., 1998] dataset. In other words, they were asking whether it is possible to repurpose a random network for the task of classifying the handwritten digits from the MNIST dataset. We use the same dataset, which consists of $60{,}000$ training images and $10{,}000$ test images, for our experiments. It is available under the Creative Commons Attribution Share-Alike 3.0 licence.

The networks we use classify inputs into $1{,}000$ classes. We map the 10 labels of the MNIST dataset onto the first 10 of these classes.

For additional experimental results on the Fashion-MNIST and Kuzushiji-MNIST datasets, please see Appendix H.1.

Finding and evaluating adversarial programs. Internally, we represent adversarial programs using unconstrained weights. We then apply a softsign function to the weights to map them into the range $(-1, 1)$, and further shift and scale the program such that the pixel values lie in the same range that is used for the input images. The program is initialised in such a way that after the application of the softsign function, each value lies uniformly at random in $(-1, 1)$.

We use the $60{,}000$ training images to run an Adam optimiser [Kingma and Ba, 2015] with learning rate 0.01 and a batch size of 50 to optimise the unconstrained weights of the adversarial program. We report the accuracy on the $10{,}000$ test images after 20 epochs, please see Figure 3.

The experiments were mainly run on two internal clusters utilising a mix of NVIDIA GPUs such as GeForce RTX 3080 Ti, Quadro RTX 6000, GeForce RTX 3060, GeForce RTX 2080 Ti, and GeForce GTX 1080. Depending on the network, optimising a single adversarial program for 20 epochs takes between 30 minutes and 1.5 hours on a standard desktop computer with two NVIDIA GeForce RTX 3080 Ti GPUs.

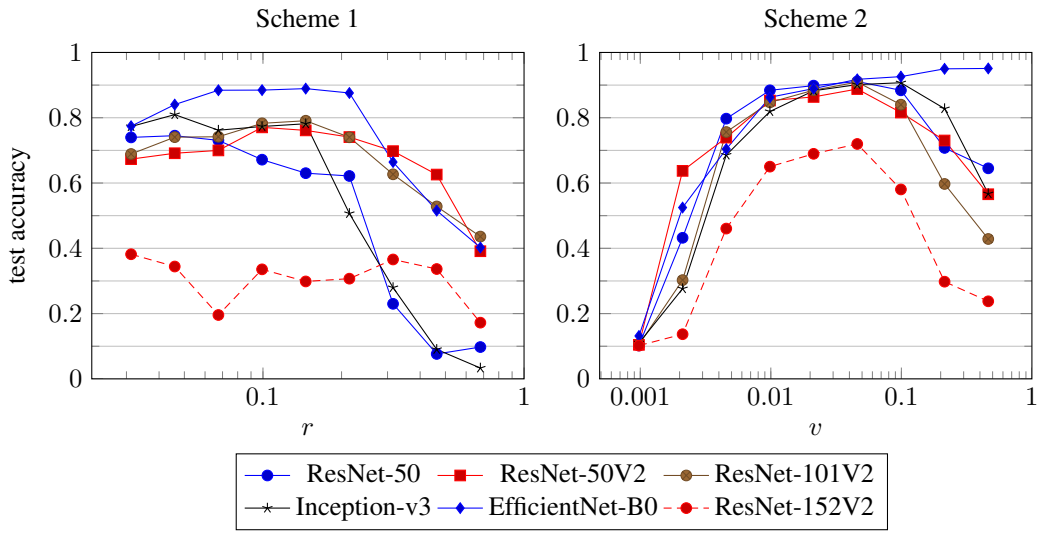

Figure 3: The accuracy achieved by the adversarial program on the MNIST test set for different parameters of the two schemes of combining input images with adversarial programs. The horizontal axes are logarithmic. The values plotted are averages over 5 trials, which are listed together with the standard deviations in Appendix H.2.

We did not explore different optimisers and learning rates, since our first choices already resulted in suitable adversarial programs for these random networks. We only reduced the batch size to 50 after first trying 100, in order to reduce the requirement on GPU memory and be able to easily run the experiments on a wider range of hardware. However, we did extensively explore the two different schemes of combing input images with adversarial programs, and different values for the respective parameters r and v. For each network, and each value of r and v, we ran 5 experiments, each with a new random initialisation of the network, and are reporting the average of the test accuracy.

Discussion. Overall, the second scheme of combining input images with adversarial programs appears to give better and more reliable results in our experiments. For both schemes, the choice of parameters is important. Clearly, when r or v is 0, the input image is not visible to the network at all. On the other hand, when r or v is 1, there no longer is an adversarial program. In most cases, best results are achieved when the adversarial program is significantly larger (either by actual size in the first scheme, or in terms of pixel value ranges in the second scheme) than the input image.

In the second scheme in particular, we see accuracies on the test set which are lower, but not much lower than what Elsayed et al. [2019] reported for networks trained on ImageNet. For $v \approx 0.046$ for example, we see accuracies of 91.8%, 91.1%, 90.9%, 90.2%, 88.8%, 72.0% for EfficientNet-B0, ResNet-50, ResNet-101V2, Inception-v3, ResNet-50V2, ResNet-152V2, respectively. This suggests that, while training, say, on ImageNet may impact the possibility of finding suitable adversarial programs, such a training may be less important than previously thought and other factors are of significant importance.

5 Conclusion and future work

We proved the first theoretical results on adversarial reprogrammability of neural networks, in which we focused on architectures with two layers and ReLU activations, and on Bernoulli and Gaussian adversarial tasks. Provided the input dimension is sufficiently large, and for a wide variety of parameter regimes, our results show that: firstly, arbitrarily high reprogramming accuracies are achievable in expectation for networks with random weights; and secondly, reprogramming accuracies that are no better than guessing may be unavoidable for networks that were trained for many iterations with small learning rates on orthogonally separable datasets.

In the theoretical results that conclude arbitrarily high expected reprogramming accuracies, we assumed that the width of the random network is no grater than its input dimension, which is similar to the assumption on widths in e.g. Daniely and Shacham [2020] and enabled us to show existence of suitable adversarial programs by matrix inversion. Interesting directions for future work include relaxing this assumption, extending the whole theoretical analysis to deeper networks, and investigating derivation of adversarial programs by gradient methods.

It would also be interesting to consider more permissive data models; however, our theoretical results on the failure of adversarial reprogramming on networks that were trained to infinity rely on implicit bias properties of gradient methods, and in that area separability assumptions on training data are common and appear challenging to lift (see e.g. Lyu et al. [2021]).

The outcomes of our experiments, which are on six realistic convolutional network architectures designed for image classification, and on three adversarial tasks provided by the MNIST, Fashion-MNIST and Kuzushiji-MNIST datasets, are consistent with our theoretical results on high reprogramming accuracies. Both the experimental plots of test accuracies and the theoretical bounds of network outputs indicate existence of "sweet spot" maximising parameter values.[3] Since in the experiments, the network architectures have widths, depths and other features that are currently beyond our theoretical assumptions, and the adversarial programs are derived by gradient descent, their outcomes provide further motivation for extending the theory as suggested above.

A conclusion that emerges across our experimental results is that the EfficientNet-B0 architecture tends to be more susceptible to adversarial reprogramming, and the ResNet-152V2 architecture tends to be less susceptible, than the remaining four. We suggest for future work investigating the causes of this, as well as seeking to reprogram random networks for adversarial tasks that are more difficult than MNIST, Fashion-MNIST and Kuzushiji-MNIST. In the context of the latter, it would be interesting to consider also vision transformer [Dosovitskiy, Beyer, Kolesnikov, Weissenborn, Zhai, Unterthiner, Dehghani, Minderer, Heigold, Gelly, Uszkoreit, and Houlsby, 2021] architectures.

Another direction for experimental work is to verify that long training can cause adversarial reprogramming to fail.

Acknowledgments and Disclosure of Funding

We are grateful to the anonymous reviewers, whose comments helped us improve the paper. We also thank Maria Ovens for linguistic advice.

This research was partially supported by the Centre for Discrete Mathematics and Its Applications (DIMAP) at the University of Warwick.

References

Peter L. Bartlett, Sébastien Bubeck, and Yeshwanth Cherapanamjeri. Adversarial Examples in Multi-Layer Random ReLU Networks. In *NeurIPS*, pages 9241–9252, 2021. 15

Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion Attacks against Machine Learning at Test Time. In *ECML PKDD*, pages 387–402, 2013. 1

Tom B. Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. Adversarial Patch. *CoRR*, abs/1712.09665, 2017. 16

Sébastien Bubeck, Yeshwanth Cherapanamjeri, Gauthier Gidel, and Rémi Tachet des Combes. A single gradient step finds adversarial examples on random two-layers neural networks. In *NeurIPS*, pages 10081–10091, 2021. 4, 15

Tarin Clanuwat, Mikel Bober-Irizar, Asanobu Kitamoto, Alex Lamb, Kazuaki Yamamoto, and David Ha. Deep Learning for Classical Japanese Literature. *CoRR*, abs/1812.01718, 2018. 3, 28

Frank H. Clarke. Generalized gradients and applications. *Trans. Amer. Math. Soc.*, 205:247–262, 1975. 6, 21

[3]The varying of the scaling parameters r and v in the experiments corresponds in the Bernoulli data model case to varying the radius ρ while keeping all other parameters fixed, and in the Gaussian data model case to varying the mean radius ϱ and the variance parameter ς while keeping fixed their ratio and all other parameters.

Amit Daniely and Hadas Shacham. Most ReLU Networks Suffer from ℓ^2 Adversarial Perturbations. In *NeurIPS*, 2020. 10, 15

Damek Davis, Dmitriy Drusvyatskiy, Sham M. Kakade, and Jason D. Lee. Stochastic Subgradient Method Converges on Tame Functions. *Found. Comput. Math.*, 20(1):119–154, 2020. 21

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *ICLR*, 2021. 10

Joydeep Dutta, Kalyanmoy Deb, Rupesh Tulshyan, and Ramnik Arora. Approximate KKT points and a proximity measure for termination. *J. Glob. Optim.*, 56(4):1463–1499, 2013. 21

Gamaleldin F. Elsayed, Ian Goodfellow, and Jascha Sohl-Dickstein. Adversarial Reprogramming of Neural Networks. In *ICLR*, 2019. 1, 2, 3, 7, 8, 9

Justin Gilmer, Luke Metz, Fartash Faghri, Samuel S. Schoenholz, Maithra Raghu, Martin Wattenberg, and Ian J. Goodfellow. Adversarial Spheres. In *ICLR (Workshop)*, 2018. 26

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, pages 249–256, 2010. 7

Karen Hambardzumyan, Hrant Khachatrian, and Jonathan May. WARP: Word-level Adversarial ReProgramming. In *ACL/IJCNLP*, pages 4921–4933, 2021. 2

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *ICCV*, pages 1026–1034, 2015. 4

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *CVPR*, pages 770–778, 2016a. 7

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity Mappings in Deep Residual Networks. In *ECCV*, pages 630–645, 2016b. 7

Kun He, Yan Wang, and John E. Hopcroft. A Powerful Generative Model Using Random Weights for the Deep Image Representation. In *NeurIPS*, pages 631–639, 2016c. 2

Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Mądry. Adversarial Examples Are Not Bugs, They Are Features. In *NeurIPS*, pages 125–136, 2019. 15, 26

Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *ICML*, pages 448–456, 2015. 7

Ziwei Ji and Matus Telgarsky. Directional convergence and alignment in deep learning. In *NeurIPS*, 2020. 6, 16, 24

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *ICLR*, 2015. 8

Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images. Master's thesis, University of Toronto, 2009. 1

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proc. IEEE*, 86(11):2278–2324, 1998. 1, 4, 8

Jaehoon Lee, Yasaman Bahri, Roman Novak, Samuel S. Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein. Deep Neural Networks as Gaussian Processes. In *ICLR*, 2018. 2

Kaifeng Lyu and Jian Li. Gradient Descent Maximizes the Margin of Homogeneous Neural Networks. In *ICLR*, 2020. 6, 16, 21, 24

Kaifeng Lyu, Zhiyuan Li, Runzhe Wang, and Sanjeev Arora. Gradient Descent on Two-layer Nets: Margin Maximization and Simplicity Bias. In *NeurIPS*, pages 12978–12991, 2021. 6, 10, 16, 24

Grégoire Mesnil, Yann N. Dauphin, Xavier Glorot, Salah Rifai, Yoshua Bengio, Ian J. Goodfellow, Erick Lavoie, Xavier Muller, Guillaume Desjardins, David Warde-Farley, Pascal Vincent, Aaron C. Courville, and James Bergstra. Unsupervised and Transfer Learning Challenge: a Deep Learning Approach. In *ICML Workshop on Unsupervised and Transfer Learning*, pages 97–110, 2011. 2

Andrea Montanari and Yuchen Wu. Adversarial Examples in Random Neural Networks with General Activations. *CoRR*, abs/2203.17209, 2022. 15

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal Adversarial Perturbations. In *CVPR*, pages 86–94, 2017. 1

Paarth Neekhara, Shehzeen Hussain, Shlomo Dubnov, and Farinaz Koushanfar. Adversarial Reprogramming of Text Classification Neural Networks. In *EMNLP-IJCNLP*, pages 5215–5224, 2019. 2

Paarth Neekhara, Shehzeen Hussain, Jinglong Du, Shlomo Dubnov, Farinaz Koushanfar, and Julian J. McAuley. Cross-modal Adversarial Reprogramming. In *WACV*, pages 2898–2906, 2022. 2

Mary Phuong and Christoph H. Lampert. The inductive bias of ReLU networks on orthogonally separable data. In *ICLR*, 2021. 3, 5, 6, 16, 22

Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Y. Ng. Self-taught learning: transfer learning from unlabeled data. In *ICML*, pages 759–766, 2007. 2

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *Int. J. Comput. Vis.*, 115(3):211–252, 2015. 1

Ludwig Schmidt, Shibani Santurkar, Dimitris Tsipras, Kunal Talwar, and Aleksander Mądry. Adversarially Robust Generalization Requires More Data. In *NeurIPS*, pages 5019–5031, 2018. 3, 4, 6, 19, 26, 27, 28

Adi Shamir, Odelia Melamed, and Oriel BenShmuel. The Dimpled Manifold Model of Adversarial Examples in Machine Learning. *CoRR*, abs/2106.10151, 2021. 15

Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K. Reiter. Accessorize to a Crime: Real and Stealthy Attacks on State-of-the-Art Face Recognition. In *CCS*, pages 1528–1540, 2016. 16

Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. The Implicit Bias of Gradient Descent on Separable Data. *J. Mach. Learn. Res.*, 19:70:1–70:57, 2018. 15

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *ICLR*, 2014. 1

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the Inception Architecture for Computer Vision. In *CVPR*, pages 2818–2826, 2016. 7

Mingxing Tan and Quoc V. Le. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In *ICML*, pages 6105–6114, 2019. 7

Yun-Yun Tsai, Pin-Yu Chen, and Tsung-Yi Ho. Transfer Learning without Knowing: Reprogramming Black-box Machine Learning Models with Scarce Data and Limited Resources. In *ICML*, pages 9614–9624, 2020. 2

Gal Vardi, Gilad Yehudai, and Ohad Shamir. Gradient Methods Provably Converge to Non-Robust Networks. In *NeurIPS*, 2022. 16

Roman Vershynin. Introduction to the non-asymptotic analysis of random matrices. In Yonina C. Eldar and Gitta Kutyniok, editors, *Compressed Sensing*, pages 210–268. Cambridge University Press, 2012. URL `http://arxiv.org/abs/1011.3027`. 16

Roman Vershynin. *High-dimensional probability: An introduction with applications in data science*, volume 47 of *Cambridge Series in Statistical and Probabilistic Mathematics*. Cambridge University Press, 2018. 18, 19

Ria Vinod, Pin-Yu Chen, and Payel Das. Reprogramming Language Models for Molecular Representation Learning. *CoRR*, abs/2012.03460, 2020. 2

Martin J. Wainwright. *High-dimensional statistics: A non-asymptotic viewpoint*, volume 48 of *Cambridge Series in Statistical and Probabilistic Mathematics*. Cambridge University Press, 2019. 19

Yifei Wang and Mert Pilanci. The Convex Geometry of Backpropagation: Neural Network Gradient Flows Converge to Extreme Points of the Dual Convex Program. In *ICLR*, 2022. 3, 16

Yunjuan Wang, Enayat Ullah, Poorya Mianjy, and Raman Arora. Adversarial Robustness is at Odds with Lazy Training. In *NeurIPS*, 2022. 15

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. *CoRR*, abs/1708.07747, 2017. 3, 28

Chao-Han Huck Yang, Yun-Yun Tsai, and Pin-Yu Chen. Voice2Series: Reprogramming Acoustic Models for Time Series Classification. In *ICML*, pages 11808–11819, 2021. 2

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *ICLR*, 2017. 16

Honggang Zhang, Jun Guo, Guang Chen, and Chun-Guang Li. HCL2000 - A Large-scale Handwritten Chinese Character Database for Handwritten Character Recognition. In *ICDAR*, pages 286–290, 2009. 2

Yang Zheng, Xiaoyi Feng, Zhaoqiang Xia, Xiaoyue Jiang, Ambra Demontis, Maura Pintor, Battista Biggio, and Fabio Roli. Why Adversarial Reprogramming Works, When It Fails, and How to Tell the Difference. *CoRR*, abs/2108.11673, 2021. 2

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] They are determined by the assumptions of the theoretical results and by the restrictions of the experiments ran, and we stated both.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] They stem from potential nefarious uses of adversarial reprogramming, which we indicated on page 2 with a reference to a fuller discussion in the literature.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] In the paper together with its appendix.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] At the URL referenced in Section 4.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 4 and Appendix H.1.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] We reported standard deviations in Appendix H.2.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Section 4 and Appendix H.1.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Contents

1 Introduction **1**
1.1 Our contributions 3
2 Random networks **4**
3 Implicit bias **5**
4 Experiments **7**
5 Conclusion and future work **9**
Acknowledgments and Disclosure of Funding **10**
References **10**
Checklist **14**
A Additional related works **15**
B Length of the adversarial program **16**
C Proofs for random networks **16**
D Clarke subdifferential **21**
E Karush-Kuhn-Tucker conditions **21**
F Proofs for implicit bias **21**
G Gaussian data models **26**
G.1 Random networks 26
G.2 Implicit bias 27
H Further details on experiments **28**
H.1 Experiments using other datasets 28
H.2 Data from experiments 28

A Additional related works

We focus on literature that is most related to this work and not already discussed in Section 1.

Understanding adversarial examples. Adversarial reprogramming can be seen as a challenging type of adversarial attack. Instead of finding perturbations that push examples over a classification boundary, the goal is to find an adversarial program which is an offset that, when added to any input from the adversarial task, with high probability makes it classified as desired. Equivalently, adversarial reprogramming seeks a single perturbation that pushes all inputs from the adversarial task to their respective target classes. Therefore understanding when and why neural networks are susceptible to adversarial reprogramming is an interesting piece of the puzzle of understanding adversarial examples. For example, our results on random networks suggest that sensitivity to well-generalising features in training data, which was identified by Ilyas, Santurkar, Tsipras, Engstrom, Tran, and Mądry [2019] as a key cause of adversarial vulnerability, does not fully explain susceptibility to adversarial reprogramming. A different conceptual framework for understanding adversarial examples, in terms of a dimpled manifold model, was proposed by Shamir, Melamed, and BenShmuel [2021]; here our result that training for longer can be a defence against adversarial reprogramming suggests that it may be interesting to explore how its success depends on shapes of the dimples in classification boundaries as they evolve during training.

More directly related to our work are the results of Bartlett, Bubeck, and Cherapanamjeri [2021] who, building on the works of Daniely and Shacham [2020] and Bubeck et al. [2021], proved that random ReLU networks of constant depth have small adversarial perturbations that can be found in one step along the direction opposite to the input gradient. Those were advanced further recently by Montanari and Wu [2022] removing a restriction on layer widths, and by Wang, Ullah, Mianjy, and Arora [2022] encompassing two-layer networks trained in the so-called lazy regime. In comparison, our results show existence of adversarial programs that are points where the classification boundary behaves approximately as required for the adversarial task, and are sensitive to the radius and the difficulty of the latter as parameters; however we leave theoretical investigations of obtaining adversarial programs by gradient methods for future work.

Implicit bias of gradient descent. By showing that, for linear logistic regression on linearly separable data, gradient descent always converges to the maximum-margin solution, Soudry, Hoffer, Nacson, Gunasekar, and Srebro [2018] initiated a fruitful research direction on implicit bias of gradient

descent, which tackles one of the greatest open questions in deep learning: why do overparameterised deep neural networks generalise well [Zhang, Bengio, Hardt, Recht, and Vinyals, 2017].

Some of the works closest to ours are: Lyu and Li [2020], Ji and Telgarsky [2020], who established that, for positively homogeneous deep networks and either exponential or logistic loss, if at some time training attains perfect accuracy and a small loss, then continuing the training makes the loss converge to 0 and the weights converge in direction to a Karush-Kuhn-Tucker point of a constrained optimisation problem on margin maximisation; Phuong and Lampert [2021], who showed that, from small and balanced initialisations, and when trained with logistic loss on orthogonally separable data whose positive and negative support examples span the whole space, two-layer ReLU networks converge to a linear combination of two maximum-margin neurons; Wang and Pilanci [2022], who derived a new proof via characterising the implicit bias of unregularised non-convex gradient flow as convex regularisation of an equivalent convex model; Lyu et al. [2021], who proved that, from small initialisations, and when trained with logistic loss on symmetric linearly separable data, two-layer networks with the leaky ReLU activation converge to a globally maximum-margin linear classifier, and that the result is fragile with respect to the symmetry assumption; and Vardi, Yehudai, and Shamir [2022], who showed that, for two-layer ReLU networks with first-layer biases and for data whose points are neither too many nor too correlated, if training converges then it produces non-robust solutions in spite of robust ones existing. To supplement our discussion in Section 1 of the motivations for considering random versus trained networks in this work, we remark that we see merit in the point of Vardi et al. [2022] that *"trained networks are clearly not random, and properties that hold in random networks may not hold in trained networks."*

Physical adversarial examples. Currently more distant but with potential for interesting connections to our work is the vibrant research direction on adversarial examples in the physical world. For example, our results may inform attempts to design adversarial accessories with prescribed effects for a set of participants [Sharif, Bhagavatula, Bauer, and Reiter, 2016], or adversarial patches that cause prescribed classifications for a set of objects [Brown, Mané, Roy, Abadi, and Gilmer, 2017].

B Length of the adversarial program

To estimate the length of the adversarial program defined in Section 2, we make use of the following known bounds on the smallest and largest singular values of matrices of independent identical Gaussians.

Theorem 5 (see e.g. Vershynin [2012, Corollary 5.35])**.** *Let $\boldsymbol{A}$ be an $n \times m$ matrix whose entries are independent centred Gaussians with variance $1/d$. Then with probability at least $1-\gamma$ its smallest and largest singular values satisfy*

$$\frac{\sqrt{m}-\sqrt{n}-\sqrt{2\ln(2/\gamma)}}{\sqrt{d}} \leq s_{\min}(\boldsymbol{A}) \leq s_{\max}(\boldsymbol{A}) \leq \frac{\sqrt{m}+\sqrt{n}+\sqrt{2\ln(2/\gamma)}}{\sqrt{d}} \ .$$

It follows that, in the regime $k = o(d)$, applying Theorem 5 to the $k \times d$ matrix $\boldsymbol{W}$ with $\gamma = o_d(1)$ and $\gamma = e^{-o(d)}$ gives us that with probability $1 - o_d(1)$ the singular values are within $1 \pm o_d(1)$. Then, from the inequalities in (1), we have that $\|\boldsymbol{p}\|$ is close to $\|\boldsymbol{p}'\|$ for large d. Observe that $\|\boldsymbol{p}'\| = \sqrt{d}$ whenever K^- is not empty, which occurs with probability $1 - 2^{-k}$ because the events $\{j \in K^-\}$ are independent and have probability $1/2$.

C Proofs for random networks

Here we work with the notations and assumptions from Section 2, so we have: a random two-layer ReLU network $\mathcal{N}$ with input dimension d, width k such that $k \leq d$, and weights $\boldsymbol{w}_j$ and a_j for $j \in [k]$; a $(\boldsymbol{\phi}, \rho, \tau)$-Bernoulli data model; and an adversarial program $\boldsymbol{p}$ defined as in (1).

First we prove a lemma that provides a lower (respectively, upper) bound on the sum of outputs of the "helpful" neurons for the adversarial program $\boldsymbol{p}$ together with inputs $\boldsymbol{x}$ that are positively (respectively, negatively) correlated with the direction $\boldsymbol{\phi}$ of the data model. In the proof, we use concentration properties of the weights of $\mathcal{N}$ to show that these neurons get close to computing together the inner product $\boldsymbol{\phi}^\top \boldsymbol{x}$, which equals $\rho \cos \alpha$ where α is the angle between $\boldsymbol{\phi}$ and $\boldsymbol{x}$. The

probability parameters γ and γ' can be regarded as constants that can be chosen to be arbitrarily small. If $1/|\cos\alpha|$ grows slower than the square root of the width k, then the bound, up to a constant factor, is essentially $\sqrt{k/d}\,\rho\cos\alpha$.

Lemma 6. *Suppose $\boldsymbol{x} \in \mathbb{R}^d$ is such that $\|\boldsymbol{x}\| = \rho$ and $\boldsymbol{\phi}^\top \boldsymbol{x} \neq 0$. Let $\alpha \coloneqq \angle(\boldsymbol{\phi}, \boldsymbol{x})$ and $y \coloneqq \operatorname{sgn}(\cos\alpha)$. Then with probability at least $1 - \gamma - \gamma'/\sqrt{2\pi}$ one has*

$$y \sum_{j \in K^+} a_j \psi(\boldsymbol{w}_j^\top(\boldsymbol{p} + \boldsymbol{x})) \;>\; \frac{\sqrt{k}\rho}{\sqrt{d}} \left(\frac{\sqrt{\pi}}{4} |\cos\alpha| \left(\frac{1}{4\sqrt{2}} - \sqrt{\frac{\ln(1/\gamma)}{k}} \right) - \sqrt{\frac{2\ln(1/\gamma')}{k}} \right) ,$$

provided that $\gamma' \leq 1/\sqrt{e}$.

Proof. Let us consider the case $y = 1$, i.e. $\boldsymbol{\phi}^\top \boldsymbol{x} = \rho\cos\alpha > 0$.

We reason as follows, where $\overset{\mathrm{d}}{=}$ denotes distributional equivalence:

$$\begin{aligned}
&\sum_{j \in K^+} a_j \psi(\boldsymbol{w}_j^\top(\boldsymbol{p} + \boldsymbol{x})) \\
&= \sum_{j \in K^+} a_j \psi(\boldsymbol{w}_j^\top \boldsymbol{x}) && \text{since } j \in K^+ \text{ implies } \boldsymbol{w}_j^\top \boldsymbol{p} = 0 \\
&= \sum_{j \in K^+} \frac{\operatorname{sgn}(\boldsymbol{w}_j^\top \boldsymbol{\phi})}{\sqrt{k}} \psi(\boldsymbol{w}_j^\top \boldsymbol{x}) && \text{by definition of } K^+ \\
&\overset{\mathrm{d}}{=} \sum_{j \in [k]} B_j \frac{\operatorname{sgn}(\boldsymbol{w}_j^\top \boldsymbol{\phi})}{\sqrt{k}} \psi(\boldsymbol{w}_j^\top \boldsymbol{x}) && \text{where } B_j \text{ are independent } \mathrm{Ber}(1/2) \\
&= \sum_{\substack{B_j = 1 \\ \boldsymbol{w}_j^\top \boldsymbol{\phi} > 0 \\ \boldsymbol{w}_j^\top \boldsymbol{x} > 0}} \frac{\boldsymbol{w}_j^\top \boldsymbol{x}}{\sqrt{k}} - \sum_{\substack{B_{j'} = 1 \\ \boldsymbol{w}_{j'}^\top \boldsymbol{\phi} < 0 \\ \boldsymbol{w}_{j'}^\top \boldsymbol{x} > 0}} \frac{\boldsymbol{w}_{j'}^\top \boldsymbol{x}}{\sqrt{k}} && \text{by definition of } \psi \\
&= \sum_{\substack{B_j = 1 \\ \boldsymbol{w}_j^\top \boldsymbol{\phi} > 0 \\ \boldsymbol{w}_j^\top \boldsymbol{x} > 0}} \frac{\boldsymbol{w}_j^\top \boldsymbol{x}}{\sqrt{k}} + \sum_{\substack{B_{j'} = 1 \\ -\boldsymbol{w}_{j'}^\top \boldsymbol{\phi} > 0 \\ -\boldsymbol{w}_{j'}^\top \boldsymbol{x} < 0}} \left(-\frac{\boldsymbol{w}_{j'}^\top \boldsymbol{x}}{\sqrt{k}} \right) && \text{and observe } j \text{ and } j' \text{ are disjoint} \\
&\overset{\mathrm{d}}{=} \sum_{\substack{B_j = 1 \\ \boldsymbol{w}_j^\top \boldsymbol{\phi} > 0 \\ \boldsymbol{w}_j^\top \boldsymbol{x} > 0}} \frac{\boldsymbol{w}_j^\top \boldsymbol{x}}{\sqrt{k}} + \sum_{\substack{B_{j'} = 1 \\ \boldsymbol{w}_{j'}^\top \boldsymbol{\phi} > 0 \\ \boldsymbol{w}_{j'}^\top \boldsymbol{x} < 0}} \frac{\boldsymbol{w}_{j'}^\top \boldsymbol{x}}{\sqrt{k}} && \text{by independence and symmetry of } \boldsymbol{w}_{j'} \\
&= \sum_{\substack{B_j = 1 \\ \boldsymbol{w}_j^\top \boldsymbol{\phi} > 0}} \frac{\boldsymbol{w}_j^\top \boldsymbol{x}}{\sqrt{k}} && \text{by merging the sums} \\
&= \sum_{\substack{B_j = 1 \\ \boldsymbol{w}_j^\top \boldsymbol{\phi} > 0}} \frac{\boldsymbol{w}_j^\top \boldsymbol{\phi}\boldsymbol{\phi}^\top \boldsymbol{x}}{\sqrt{k}} + \sum_{\substack{B_j = 1 \\ \boldsymbol{w}_j^\top \boldsymbol{\phi} > 0}} \frac{\boldsymbol{w}_j^\top (\boldsymbol{I}_d - \boldsymbol{\phi}\boldsymbol{\phi}^\top) \boldsymbol{x}}{\sqrt{k}} && \text{projecting onto } \boldsymbol{\phi} \text{ and orthogonal hyperplane} \\
&\overset{\mathrm{d}}{=} \underbrace{\sum_{B'_j = 1} \frac{|\boldsymbol{w}_j^\top \boldsymbol{\phi}| \boldsymbol{\phi}^\top \boldsymbol{x}}{\sqrt{k}}}_{\text{(I)}} + \underbrace{\sum_{B'_j = 1} \frac{\boldsymbol{w}_j^\top (\boldsymbol{I}_d - \boldsymbol{\phi}\boldsymbol{\phi}^\top) \boldsymbol{x}}{\sqrt{k}}}_{\text{(II)}} && \text{where } B'_j \text{ are independent } \mathrm{Ber}(1/4).
\end{aligned}$$

We now analyse the two sums separately, in both cases obtaining a lower bound:

(I) Each of the k terms is the absolute value of an independent centred Gaussian with variance $(\rho\cos\alpha)^2/(dk)$, so it is with probability $1/2$ greater than

$$\frac{\sqrt{2}\,\mathrm{erf}^{-1}(1/2)\rho\cos\alpha}{\sqrt{dk}} > \frac{\sqrt{\pi}\rho\cos\alpha}{2\sqrt{2dk}}\,.$$

Also each of the k terms is present in the sum independently with probability $1/4$. Hence by Hoeffding's inequality (see e.g. Vershynin [2018, Theorem 2.2.6]), we have that with probability at least $1-\gamma$ the sum is greater than

$$\frac{\sqrt{\pi}\rho\cos\alpha}{2\sqrt{2dk}}\left(\frac{k}{8}-\sqrt{\frac{k\ln(1/\gamma)}{2}}\right) = \frac{\sqrt{\pi k}\rho\cos\alpha}{4\sqrt{d}}\left(\frac{1}{4\sqrt{2}}-\sqrt{\frac{\ln(1/\gamma)}{k}}\right)\,. \tag{2}$$

(II) Each term is an independent centred Gaussian with variance $(\rho\sin\alpha)^2/(dk)$, so a sum of k' such terms is a centred Gaussian with variance $(\rho\sin\alpha)^2k'/(dk)$, which is with probability at least $1-\gamma'/\sqrt{2\pi}$ at least

$$-\rho\,|\sin\alpha|\sqrt{\frac{2k'\ln(1/\gamma')}{dk}} \geq -\rho\sqrt{\frac{2\ln(1/\gamma')}{d}}\,, \quad \text{provided} \quad \gamma' \leq 1/\sqrt{e}\,. \tag{3}$$

Summing the bounds in (2) and (3) yields the bound in the statement, and we can combine the probabilities by the union bound.

The case $y=-1$, i.e. $\boldsymbol{\phi}^\top\boldsymbol{x} = \rho\cos\alpha < 0$, is analogous. □

Second we prove a lemma that gives us a bound on how much the sum of outputs of the"unhelpful" neurons can spoil the work of the "helpful" neurons which the previous lemma addressed. The function f is the density of a standard Gaussian, i.e. $f(u) = e^{-u^2/2}/\sqrt{2\pi}$. It bounds the first summand inside the outer parentheses, which is therefore exponentially small for large $d/(\sqrt{k}\rho)$. Again viewing the probability parameter γ'' as arbitrarily small but constant, the second summand approaches zero at the rate of the square root of the width k. The techniques we use in the proof are similar to those for the previous one.

Lemma 7. *Suppose $\boldsymbol{x} \in \mathbb{R}^d$ is such that $\|\boldsymbol{x}\| = \rho$ and $\boldsymbol{\phi}^\top\boldsymbol{x} \neq 0$. Let $\alpha \coloneqq \angle(\boldsymbol{\phi},\boldsymbol{x})$ and $y \coloneqq \mathrm{sgn}(\cos\alpha)$. Then with probability at least $1-\gamma''$ one has*

$$y\sum_{j\in K^-} a_j\psi(\boldsymbol{w}_j^\top(\boldsymbol{p}+\boldsymbol{x})) \;>\; -\frac{\sqrt{k}\rho}{\sqrt{d}}\left(f\left(\frac{d}{\sqrt{k}\rho}\right)\min\left\{1,\left(\frac{\sqrt{k}\rho}{d}\right)^2\right\}+\frac{2\pi}{\pi-1}\sqrt{\frac{\ln(1/\gamma'')}{k}}\right)\,.$$

Proof. Let us consider the case $y=1$, i.e. $\boldsymbol{\phi}^\top\boldsymbol{x} = \rho\cos\alpha > 0$.

We estimate as follows:

$$\begin{aligned}
&\sum_{j\in K^-} a_j\psi(\boldsymbol{w}_j^\top(\boldsymbol{p}+\boldsymbol{x})) \\
&= \sum_{j\in K^-} a_j\psi(\boldsymbol{w}_j^\top\boldsymbol{x}-\sqrt{d/|K^-|}) && \text{since } j\in K^- \text{ implies } \boldsymbol{w}_j^\top\boldsymbol{p} = -\sqrt{d/|K^-|} \\
&\geq -\sum_{j\in K^+}\frac{\psi(\boldsymbol{w}_j^\top\boldsymbol{x}-\sqrt{d/|K^-|})}{\sqrt{k}} && \text{since } a_j \geq -1/\sqrt{k} \\
&\geq -\sum_{j\in K^+}\frac{\psi(\boldsymbol{w}_j^\top\boldsymbol{x}-U)}{\sqrt{k}} && \text{where } U \coloneqq \sqrt{d/k} \\
&\geq -\sum_{j\in [k]}\frac{\psi(\boldsymbol{w}_j^\top\boldsymbol{x}-U)}{\sqrt{k}} && \text{since } \psi(u)\geq 0 \text{ for all } u.
\end{aligned}$$

Each $\boldsymbol{w}_j^\top \boldsymbol{x}$ is a centred Gaussian with variance $\rho^2/d =: \sigma^2$. Hence (see e.g. Vershynin [2018, Proposition 2.1.2])

$$\mathbb{P}\{\boldsymbol{w}_j^\top \boldsymbol{x} \geq U\} \geq \max\{\sigma/U - (\sigma/U)^3, 0\} \cdot f(U/\sigma) \ .$$

We therefore have

$$\begin{aligned}\mu \coloneqq \mathbb{E}[\psi(\boldsymbol{w}_j^\top \boldsymbol{x} - U)] &= \int_U^\infty (u-U)\frac{f(u/\sigma)}{\sigma}\,\mathrm{d}u \\ &= \sigma f(U/\sigma) - U\,\mathbb{P}\{\boldsymbol{w}_j^\top \boldsymbol{x} \geq U\} \leq \sigma f(U/\sigma)\min\{1, (\sigma/U)^2\} \ .\end{aligned}$$

Writing X_j for the centred random variable $\psi(\boldsymbol{w}_j^\top \boldsymbol{x} - U) - \mu$, there are the following two cases.

Case $0 \leq s \leq \mu$.

$$\begin{aligned}\mathbb{P}\{|\boldsymbol{w}_j^\top \boldsymbol{x}| \geq s\} \geq \mathbb{P}\{|\boldsymbol{w}_j^\top \boldsymbol{x}| \geq \mu\} &= 1 - \int_{-\mu}^{\mu} \frac{f(u/\sigma)}{\sigma}\,\mathrm{d}u > 1 - 2\mu f(0)/\sigma \\ &\geq 1 - 2f(0)f(U/\sigma) > 1 - \frac{1}{\pi} \geq \left(1 - \frac{1}{\pi}\right)\mathbb{P}\{|X_j| \geq s\} \ .\end{aligned}$$

Case $\mu < s$.

$$\mathbb{P}\{|\boldsymbol{w}_j^\top \boldsymbol{x}| \geq s\} \geq 2\,\mathbb{P}\{\boldsymbol{w}_j^\top \boldsymbol{x} \geq U + \mu + s\} = 2\,\mathbb{P}\{\psi(\boldsymbol{w}_j^\top \boldsymbol{x} - U) \geq \mu + s\} = 2\,\mathbb{P}\{|X_j| \geq s\} \ .$$

Hence X_j is sub-Gaussian with parameter $\sqrt{2}\pi\sigma/(\pi - 1)$ (see e.g. Wainwright [2019, proof of Theorem 2.6]), so by Hoeffding's bound (see e.g. Wainwright [2019, Proposition 2.5]), with probability at least $1 - \gamma''$ one has

$$\begin{aligned}\sum_{j \in K^-} a_j \psi(\boldsymbol{w}_j^\top(\boldsymbol{p} + \boldsymbol{x})) &\geq -\sum_{j \in [k]} \frac{\mu + X_j}{\sqrt{k}} = -\sqrt{k}\mu - \sum_{j \in [k]} \frac{X_j}{\sqrt{k}} \\ &> -\sqrt{k}\mu - \frac{2\pi}{\pi - 1}\sigma\sqrt{\ln(1/\gamma'')} \geq -\sqrt{k}\,\sigma f(U/\sigma)\min\{1, (\sigma/U)^2\} - \frac{2\pi}{\pi - 1}\sigma\sqrt{\ln(1/\gamma'')} \\ &= -\frac{\sqrt{k}\rho}{\sqrt{d}} f\left(\frac{d}{\sqrt{k}\rho}\right)\min\left\{1, \left(\frac{\sqrt{k}\rho}{d}\right)^2\right\} - \frac{2\pi}{\pi - 1}\rho\sqrt{\frac{\ln(1/\gamma'')}{d}} \ ,\end{aligned}$$

which equals the bound in the statement.

The case $y = -1$, i.e. $\boldsymbol{\phi}^\top \boldsymbol{x} = \rho\cos\alpha < 0$, is again analogous. □

Our main technical result on random networks provides a lower (respectively, upper) bound on the output of the adversarially reprogrammed network for positively (respectively, negatively) labelled inputs sampled from the Bernoulli data model.

Theorem 8. *There exist positive constants C_1, C_2, C_3, C_4 and C_5 such that, with probability at least $(1 - C_1\gamma)(1 - \gamma^\dagger)$ over the draw of the network $\mathcal{N}$ and the labelled data point $(\boldsymbol{x}, y)$ from the $(\boldsymbol{\phi}, \rho, \tau)$-Bernoulli distribution, provided that $2d\tau^2 \geq \ln(1/\gamma^\dagger)$, we have*

$$y\,\mathcal{N}(\boldsymbol{p}+\boldsymbol{x}) > \frac{\sqrt{k}\rho}{\sqrt{d}}\left(C_2\tau - C_3\exp\left(-\frac{d^2}{2k\rho^2}\right)\min\left\{1, \frac{k\rho^2}{d^2}\right\} - C_4\sqrt{\frac{\ln(1/\gamma)}{k}} - C_5\sqrt{\frac{\ln(1/\gamma^\dagger)}{d}}\right) \ .$$

Proof. Let $\alpha \coloneqq \angle(\boldsymbol{\phi}, \boldsymbol{x})$.

Since $(\boldsymbol{x}, y)$ is sampled from the $(\boldsymbol{\phi}, \rho, \tau)$-Bernoulli distribution, Schmidt et al. [2018, Lemma 24] tell us that, for all $u \in (0, 2\tau]$,

$$\mathbb{P}\{y\cos\alpha > 2\tau - u\} \geq 1 - e^{-du^2/2} \ ,$$

i.e. that with probability at least $1 - \gamma^\dagger$ one has

$$y\cos\alpha > 2\tau - \sqrt{\frac{2\ln(1/\gamma^\dagger)}{d}} \ ,$$

provided that $2d\tau^2 \geq \ln(1/\gamma^\dagger)$.

It remains to apply Lemmas 6 and 7 with $\gamma = \gamma' = \gamma''$ and $C_1 = 2 + 1/\sqrt{2\pi}$, observe that $1 - C_1\gamma \geq 0$ implies $\gamma \leq 1/\sqrt{e}$, and simplify as follows:

$$\begin{aligned}
&\frac{\sqrt{\pi}}{4} y \cos\alpha \left(\frac{1}{4\sqrt{2}} - \sqrt{\frac{\ln(1/\gamma)}{k}} \right) - \sqrt{\frac{2\ln(1/\gamma)}{k}} \\
&- f\left(\frac{d}{\sqrt{k}\rho}\right) \min\left\{1, \left(\frac{\sqrt{k}\rho}{d}\right)^2\right\} - \frac{2\pi}{\pi - 1}\sqrt{\frac{\ln(1/\gamma)}{k}} \\
&\geq \frac{\sqrt{\pi}}{16\sqrt{2}} y \cos\alpha - f\left(\frac{d}{\sqrt{k}\rho}\right) \min\left\{1, \left(\frac{\sqrt{k}\rho}{d}\right)^2\right\} \\
&\quad - \left(\sqrt{2} + \frac{\sqrt{\pi}}{4} + \frac{2\pi}{\pi - 1}\right)\sqrt{\frac{\ln(1/\gamma)}{k}} \\
&> \frac{\sqrt{\pi}}{8\sqrt{2}}\tau - f\left(\frac{d}{\sqrt{k}\rho}\right) \min\left\{1, \left(\frac{\sqrt{k}\rho}{d}\right)^2\right\} \\
&\quad - \left(\sqrt{2} + \frac{\sqrt{\pi}}{4} + \frac{2\pi}{\pi - 1}\right)\sqrt{\frac{\ln(1/\gamma)}{k}} - \frac{\sqrt{\pi}}{16}\sqrt{\frac{\ln(1/\gamma^\dagger)}{d}} \ ,
\end{aligned}$$

which establishes the statement with

$$C_2 = \frac{\sqrt{\pi}}{8\sqrt{2}} \ , \quad C_3 = 1/\sqrt{2\pi} \ , \quad C_4 = \sqrt{2} + \frac{\sqrt{\pi}}{4} + \frac{2\pi}{\pi - 1} \quad \text{and} \quad C_5 = \frac{\sqrt{\pi}}{16} \ . \quad \square$$

We now restate and prove Corollary 1 from Section 2.

Corollary 1. *Suppose that*

$$k = \Theta(d^{\eta_{(k)}}) \ , \quad \rho = O(d^{\eta_{(\rho)}}) \ , \quad 1/\tau = O(d^{\eta_{(\tau)}}) \quad \textit{and} \quad 1/\tau = \omega_d(1) \ ,$$

where $\eta_{(k)}, \eta_{(\rho)}, \eta_{(\tau)} \in [0, 1]$ *are arbitrary constants that satisfy*

$$\eta_{(\rho)} < 1 - \eta_{(k)}/2 \quad \textit{and} \quad \eta_{(\tau)} < \eta_{(k)}/2 \ .$$

Then, for sufficiently large input dimensions d*, the expected accuracy of the adversarially reprogrammed network* $\mathcal{N}$ *on the* (ϕ, ρ, τ)*-Bernoulli data model is arbitrarily close to* 100%.

Proof. Fix γ and $\gamma^\dagger$ such that $(1 - C_1\gamma)(1 - \gamma^\dagger)$ is as close to 100% as required.

By the assumptions in the statement, we have $\eta_{(\tau)} < 1/2$, so $\tau = \Omega(d^{-\eta_{(\tau)}}) = \omega(d^{-1/2})$ and hence $2d\tau^2 \geq \ln(1/\gamma^\dagger)$ for large enough d; moreover, we have

$$\exp\left(\frac{d^2}{k\rho^2}\right) = \omega((1/\tau)^2) \ , \quad \frac{k}{\ln(1/\gamma)} = \omega((1/\tau)^2) \quad \text{and} \quad \frac{d}{\ln(1/\gamma^\dagger)} = \omega((1/\tau)^2) \ .$$

The corollary follows by applying Theorem 8 and observing that

$$\begin{aligned}
&C_2\tau - C_3 \exp\left(-\frac{d^2}{2k\rho^2}\right) \min\left\{1, \frac{k\rho^2}{d^2}\right\} - C_4\sqrt{\frac{\ln(1/\gamma)}{k}} - C_5\sqrt{\frac{\ln(1/\gamma^\dagger)}{d}} \\
&\qquad = \frac{1}{O(1/\tau)} - \frac{1}{\omega(1/\tau)\ln(\omega((1/\tau)^2))} - \frac{1}{\omega(1/\tau)} = \frac{1}{O(1/\tau)} \ ,
\end{aligned}$$

which is positive for large enough d. $\square$

D Clarke subdifferential

By Rademacher's theorem, every locally Lipschitz function $g : \mathbb{R}^m \to \mathbb{R}$ is differentiable almost everywhere. Its Clarke subdifferential [Clarke, 1975] ∂g at a point $\boldsymbol{z}$ is the convex closure of the set of all limits of gradients along sequences that converge to $\boldsymbol{z}$:

$$\partial g(\boldsymbol{z}) := \operatorname{conv}\left\{ \lim_{i\to\infty} \nabla g(\boldsymbol{z}_i) \mid g \text{ differentiable at all } \boldsymbol{z}_i, \text{ and } \lim_{i\to\infty} \boldsymbol{z}_i = \boldsymbol{z} \right\} ,$$

which is nonempty and compact. If g is continuously differentiable at $\boldsymbol{z}$, then $\partial g(\boldsymbol{z}) = \{\nabla g(\boldsymbol{z})\}$.

For example, the Clarke subdifferential of the ReLU function is:

$$\partial \psi(u) = \begin{cases} \{0\} & \text{if } u < 0, \\ [0,1] & \text{if } u = 0, \\ \{1\} & \text{if } u > 0. \end{cases}$$

For a fuller introduction to the Clarke subdifferential and its applications to gradient flow, and for further references to related literature, we refer the reader to e.g. Lyu and Li [2020, Section 3 and Appendix I].

E Karush-Kuhn-Tucker conditions

Following Dutta, Deb, Tulshyan, and Arora [2013, Section 2.2], supposing $g, h_1, \dots, h_n : \mathbb{R}^m \to \mathbb{R}$ are locally Lipschitz, we have that $\boldsymbol{z} \in \mathbb{R}^m$ is a Karush-Kuhn-Tucker point of the single-objective constrained optimisation problem

$$\text{minimise} \quad g(\boldsymbol{z}) \quad \text{subject to} \quad \forall i \in [n] : \; h_i(\boldsymbol{z}) \le 0$$

if and only if there exist Lagrange multipliers $\lambda_1, \dots, \lambda_n \ge 0$ such that:

(feasibility) $\forall i \in [n] : \; h_i(\boldsymbol{z}) \le 0$,

(equilibrium inclusion) $\mathbf{0} \in \partial g(\boldsymbol{z}) + \sum_{i=1}^n \lambda_i \partial h_i(\boldsymbol{z})$, and

(complementary slackness) $\forall i \in [n] : \; \lambda_i h_i(\boldsymbol{z}) = 0$.

F Proofs for implicit bias

Here we work with the notations and assumptions from Section 3, so we have a trajectory $\boldsymbol{\theta}(t) : [0,\infty) \to \mathbb{R}^{k(d+1)}$ of gradient flow for a two-layer ReLU network $\mathcal{N}$ with input dimension d and width k, trained on an orthogonally separable binary classification dataset $S = \{(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_n, y_n)\}$, from a balanced and live initialisation, using either the exponential $\ell_{\exp}(u) = e^{-u}$ or the logistic $\ell_{\log}(u) = \ln(1 + e^{-u})$ loss function.

For $t \in [0,\infty)$, we denote by $\mathcal{N}_{\boldsymbol{\theta}(t)}$ the network at time t of gradient flow, i.e. whose weights are the coordinates of the trajectory vector at time t: $\boldsymbol{w}_1(t), \dots, \boldsymbol{w}_k(t)$ for the first layer, and $a_1(t), \dots, a_k(t)$ for the second layer.

Since our network functions, loss functions, and their compositions, are definable (see Davis, Drusvyatskiy, Kakade, and Lee [2020, Corollary 5.11]), they admit the chain rule (see Davis et al. [2020, Theorem 5.8]), and hence we have the following basic fact on the empirical loss $\mathcal{L}(\boldsymbol{\theta}) = \sum_{i=1}^n \ell(y_i \mathcal{N}_{\boldsymbol{\theta}}(\boldsymbol{x}_i))$ decreasing along gradient flow.

Proposition 9 (by Davis et al. [2020, Lemma 5.2])**.** *We have* $\mathrm{d}\mathcal{L}(\boldsymbol{\theta})/\mathrm{d}t = -\|\mathrm{d}\boldsymbol{\theta}/\mathrm{d}t\|^2$ *for almost all* $t \in [0,\infty)$.

From the differential inclusion of gradient flow, i.e. that $\mathrm{d}\boldsymbol{\theta}/\mathrm{d}t \in -\partial\mathcal{L}(\boldsymbol{\theta}(t))$ for almost all $t \in [0,\infty)$, it is straightforward to obtain the following expressions for the derivatives of the weights during training. Note that for the first layer we have a membership rather than an equation because the Clarke subdifferentials of the ReLU function may be at zero.

Proposition 10. *For all $j \in [k]$ and almost all $t \in [0, \infty)$ we have*

$$\frac{\mathrm{d}\boldsymbol{w}_j}{\mathrm{d}t} \in -\sum_{i=1}^{n} \ell'(y_i \mathcal{N}_{\boldsymbol{\theta}}(\boldsymbol{x}_i)) y_i a_j \partial\psi(\boldsymbol{w}_j^\top \boldsymbol{x}_i)\boldsymbol{x}_i$$
$$\frac{\mathrm{d}a_j}{\mathrm{d}t} = -\sum_{i=1}^{n} \ell'(y_i \mathcal{N}_{\boldsymbol{\theta}}(\boldsymbol{x}_i)) y_i \psi(\boldsymbol{w}_j^\top \boldsymbol{x}_i) \ .$$

We now extend to exponential loss the observation made for logistic loss by Phuong and Lampert [2021, Lemma A.4]: that during training second-layer weights remain balanced and keep their signs.

Lemma 11. *Throughout the training, for all $j \in [k]$, we have that $|a_j| = \|\boldsymbol{w}_j\|$ and that a_j maintains its sign.*

Proof. Observe that the ReLU function satisfies $\partial\psi(u)u = \{\psi(u)\}$ for all $u \in \mathbb{R}$. Observe also that $|\ell'(u)| \leq \ell(u)$ for all $u \in \mathbb{R}$, which for exponential loss is trivial, and for logistic loss follows from $e < (1 + 1/e^u)^{e^u+1}$, i.e. $1 < \ell_{\log}(u)/|\ell'_{\log}(u)|$.

From Proposition 10 it follows that $\mathrm{d}\|\boldsymbol{w}_j\|^2/\mathrm{d}t = 2\boldsymbol{w}_j^\top(\mathrm{d}\boldsymbol{w}_j/\mathrm{d}t)$ and $\mathrm{d}a_j^2/\mathrm{d}t = 2a_j(\mathrm{d}a_j/\mathrm{d}t)$, and so $\mathrm{d}\|\boldsymbol{w}_j\|^2/\mathrm{d}t = \mathrm{d}a_j^2/\mathrm{d}t$ almost everywhere during the training.

Hence, by our assumption of balanced initialisation, we have that $|a_j| = \|\boldsymbol{w}_j\|$ throughout the training. Then, by Proposition 9, for almost all $t \in [0, \infty)$ we have that $a_j(t) \neq 0$ implies

$$\left|\frac{\mathrm{d}\ln a_j^2(t)}{\mathrm{d}t}\right| \leq 2\sum_{i=1}^{n} |\ell'(y_i \mathcal{N}_{\boldsymbol{\theta}(t)}(\boldsymbol{x}_i))| \, \|\boldsymbol{x}_i\| \leq 2\mathcal{L}(\boldsymbol{\theta}(t)) \max_{i=1}^{n} \|\boldsymbol{x}_i\| \leq 2\mathcal{L}(\boldsymbol{\theta}(0)) \max_{i=1}^{n} \|\boldsymbol{x}_i\| \ .$$

Therefore $\ln a_j^2$ is bounded below by a linear function of t, so a_j does not cross zero throughout the training. □

We now observe several key properties that the weights have during the training in our setting. Namely: the offsets of the positive neurons remain in the cone spanned by the positive inputs and the opposites of the negative inputs, their inner products with all the latter vectors are nondecreasing, and their norms either remain bounded or tend to infinity; and the analogous holds for the negative neurons. For $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_m \in \mathbb{R}^l$, let $\mathrm{cone}(\boldsymbol{u}_1, \ldots, \boldsymbol{u}_m) := \{\sum_{i=1}^m b_i \boldsymbol{u}_i \mid b_1, \ldots, b_m \in [0, \infty)\}$. Also let $\boldsymbol{X} := \{y_i \boldsymbol{x}_i \mid i \in [n]\}$.

Lemma 12. *(i) For all $j \in [k]$ and all $t \in [0, \infty)$, we have $\boldsymbol{w}_j(t) - \boldsymbol{w}_j(0) \in \mathrm{cone}(\mathrm{sgn}(a_j)\boldsymbol{X})$.*

(ii) For all $i \in [n]$, all $j \in [k]$, and almost all $t \in [0, \infty)$, we have $\mathrm{d}(\boldsymbol{w}_j^\top \mathrm{sgn}(a_j) y_i \boldsymbol{x}_i)/\mathrm{d}t \geq 0$.

(iii) For each $j \in [k]$, either $|a_j|$ is bounded, or $|a_j| \to \infty$ as $t \to \infty$.

Proof. We have (i) by Proposition 10 and the negativity of the derivatives of both the exponential and the logistic loss functions.

For (ii), we have by Proposition 10 that

$$\frac{\mathrm{d}(\boldsymbol{w}_j^\top \mathrm{sgn}(a_j) y_i \boldsymbol{x}_i)}{\mathrm{d}t} \in -\sum_{i'=1}^{n} \ell'(y_{i'} \mathcal{N}_{\boldsymbol{\theta}}(\boldsymbol{x}_{i'})) |a_j| \partial\psi(\boldsymbol{w}_j^\top \boldsymbol{x}_{i'}) y_i y_{i'} \boldsymbol{x}_i^\top \boldsymbol{x}_{i'} \ ,$$

which never contains negative values since the dataset is orthogonally separable.

By Lemma 11, it suffices to show (iii) for $\|\boldsymbol{w}_j\|$, and so letting $\widehat{\boldsymbol{w}}_j(t) := \boldsymbol{w}_j(t) - \boldsymbol{w}_j(0)$, it suffices to show (iii) for $\|\widehat{\boldsymbol{w}}_j\|$.

Suppose $\widehat{\boldsymbol{w}}_j^\top(t) \, \mathrm{sgn}(a_j(t)) y_i \boldsymbol{x}_i$ is bounded for all $i \in [n]$. By orthogonal separability of the dataset, every two vectors in $\mathrm{cone}(\mathrm{sgn}(a_j)\boldsymbol{X})$ have nonnegative inner product, and so by (i) we have that

$$\|\widehat{\boldsymbol{w}}_j(t)\| \leq \sum_{i \in [n]} \frac{\widehat{\boldsymbol{w}}_j^\top(t) \, \mathrm{sgn}(a_j(t)) y_i \boldsymbol{x}_i}{\|\boldsymbol{x}_i\|} \ .$$

Hence $\|\widehat{\boldsymbol{w}}_j(t)\|$ is also bounded.

Otherwise, by (ii), there exists $i \in [n]$ such that $\widehat{\boldsymbol{w}}_j^\top(t)\operatorname{sgn}(a_j(t))y_i\boldsymbol{x}_i \to \infty$ as $t \to \infty$. By orthogonal separability of the dataset, every two vectors in $\operatorname{cone}(\operatorname{sgn}(a_j)\boldsymbol{X})$ have nonnegative inner product, and so by (i) we have that

$$\frac{\widehat{\boldsymbol{w}}_j^\top(t)\operatorname{sgn}(a_j(t))y_i\boldsymbol{x}_i}{\|\boldsymbol{x}_i\|} \leq \|\widehat{\boldsymbol{w}}_j(t)\| \,.$$

Hence also $\|\widehat{\boldsymbol{w}}_j(t)\| \to \infty$ as $t \to \infty$. □

The following is our main lemma, whose statement is simple: for every input, there must be at least one ReLU whose output tends to infinity during the training. For its proof, recall from Section 3 the notation $I_s = \{i \in [n] \mid y_i = s\}$.

Lemma 13. *For all $i \in [n]$, there exists $j \in [k]$ such that $\boldsymbol{w}_j^\top\boldsymbol{x}_i \to \infty$ as $t \to \infty$.*

Proof. Consider the case $i \in I_1$.

Recalling that $\boldsymbol{w}_{j_1}^\top\boldsymbol{x}_{i_1}$ is positive at $t = 0$ by our assumption of live initialisation, and that it is nondecreasing during the training by Lemma 12 (ii), we have by Lemma 11 that a_{j_1} is bounded below by a positive constant. Since $\mathrm{d}(\boldsymbol{w}_{j_1}^\top\boldsymbol{x}_{i_1})/\mathrm{d}t \geq -\ell'(\mathcal{N}_{\boldsymbol{\theta}}(\boldsymbol{x}_{i_1}))a_{j_1}\|\boldsymbol{x}_{i_1}\|^2$ for almost all $t \in [0,\infty)$, we have that either $\mathcal{N}_{\boldsymbol{\theta}}(\boldsymbol{x}_{i_1})$ or $\boldsymbol{w}_{j_1}^\top\boldsymbol{x}_{i_1}$ is not bounded above. In either case, by Lemma 12 (iii), there exists $j \in [k]$ such that $a_j \to \infty$ as $t \to \infty$.

Suppose $-\boldsymbol{w}_j^\top\boldsymbol{x}_{i'}$ is unbounded for some $i' \in I_{-1}$. Recalling that, by Lemma 12 (i), $\boldsymbol{w}_j(t)-\boldsymbol{w}_j(0) \in \operatorname{cone}(\boldsymbol{X})$ for all $t \in [0,\infty)$, it follows from orthogonal separability of the dataset that $-\boldsymbol{w}_j^\top\boldsymbol{x}_{i'}$ is unbounded for all $i' \in I_{-1}$. Hence there exists $T \in [0,\infty)$ such that, for all $i' \in I_{-1}$ and $t \in [T,\infty)$, we have $\boldsymbol{w}_j^\top\boldsymbol{x}_{i'} < 0$. Then $\boldsymbol{w}_j(t) - \boldsymbol{w}_j(T) \in \operatorname{cone}\{\boldsymbol{x}_i \mid i \in I_1\}$ for all $t \in [T,\infty)$, so $\boldsymbol{w}_j^\top\boldsymbol{x}_{i'}$ is unbounded for some $i' \in I_1$.

Therefore $\boldsymbol{w}_j^\top\boldsymbol{x}_{i'}$ is unbounded for some $i' \in I_1$. Arguing as before, since $\boldsymbol{w}_j(t)-\boldsymbol{w}_j(0) \in \operatorname{cone}(\boldsymbol{X})$ for all $t \in [0,\infty)$, we have that $\boldsymbol{w}_j^\top\boldsymbol{x}_{i'}$ is unbounded for all $i' \in I_1$, in particular for $i' = i$ as required.

The proof for the case $i \in I_{-1}$ is analogous. □

That brings us to a position where we can restate and prove Theorem 2 from Section 3. Using Lemma 13, we show that, whereas the empirical loss is bounded below by zero, eventually its rate of decrease remains at least the sum of squares of derivatives of the loss function, which implies that the latter and hence also the empirical loss get arbitrarily small.

Theorem 2. *There exists a time t_0 such that $\mathcal{L}(\boldsymbol{\theta}(t_0)) < \ell(0)$.*

Proof. By Lemma 13, for each $i \in [n]$, let $j(i) \in [k]$ be such that $\boldsymbol{w}_{j(i)}^\top\boldsymbol{x}_i \to \infty$ as $t \to \infty$. Then let $T \in [0,\infty)$ be such that, for all $i \in [n]$ and all $t \in [T,\infty)$, we have $|a_{j(i)}| \geq 1/\|\boldsymbol{x}_i\|$ and $\boldsymbol{w}_{j(i)}^\top\boldsymbol{x}_i > 0$.

Recalling Propositions 9 and 10, for almost all $t \in [T,\infty)$ we have

$$\begin{aligned}
\frac{\mathrm{d}\mathcal{L}(\boldsymbol{\theta})}{\mathrm{d}t} &= -\left\|\frac{\mathrm{d}\boldsymbol{\theta}}{\mathrm{d}t}\right\|^2 \leq -\sum_{j=1}^k\left\|\frac{\mathrm{d}\boldsymbol{w}_j}{\mathrm{d}t}\right\|^2 \\
&\in -\sum_{j=1}^k\left\|\sum_{i=1}^n \ell'(y_i\mathcal{N}_{\boldsymbol{\theta}}(\boldsymbol{x}_i))y_i a_j\partial\psi(\boldsymbol{w}_j^\top\boldsymbol{x}_i)\boldsymbol{x}_i\right\|^2 \\
&\subseteq -\sum_{j=1}^k\left(\sum_{i=1}^n \ell'(y_i\mathcal{N}_{\boldsymbol{\theta}}(\boldsymbol{x}_i))y_i a_j\partial\psi(\boldsymbol{w}_j^\top\boldsymbol{x}_i)\boldsymbol{x}_i\right)^\top\left(\sum_{i'=1}^n \ell'(y_{i'}\mathcal{N}_{\boldsymbol{\theta}}(\boldsymbol{x}_{i'}))y_{i'} a_j\partial\psi(\boldsymbol{w}_j^\top\boldsymbol{x}_{i'})\boldsymbol{x}_{i'}\right) \\
&= -\sum_{j=1}^k\sum_{i=1}^n\sum_{i'=1}^n \ell'(y_i\mathcal{N}_{\boldsymbol{\theta}}(\boldsymbol{x}_i))\ell'(y_{i'}\mathcal{N}_{\boldsymbol{\theta}}(\boldsymbol{x}_{i'}))a_j^2\partial\psi(\boldsymbol{w}_j^\top\boldsymbol{x}_i)\partial\psi(\boldsymbol{w}_j^\top\boldsymbol{x}_{i'})y_i y_{i'}\boldsymbol{x}_i^\top\boldsymbol{x}_{i'}
\end{aligned}$$

$$\leq \left\{ - \min \sum_{i=1}^{n} \sum_{j=1}^{k} \left(\ell'(y_i \mathcal{N}_{\boldsymbol{\theta}}(\boldsymbol{x}_i)) a_j \partial\psi(\boldsymbol{w}_j^\top \boldsymbol{x}_i) \|\boldsymbol{x}_i\| \right)^2 \right\}$$

$$\leq \left\{ - \min \sum_{i=1}^{n} \left(\ell'(y_i \mathcal{N}_{\boldsymbol{\theta}}(\boldsymbol{x}_i)) a_{j(i)} \partial\psi(\boldsymbol{w}_{j(i)}^\top \boldsymbol{x}_i) \|\boldsymbol{x}_i\| \right)^2 \right\}$$

$$\leq \left\{ - \sum_{i=1}^{n} (\ell'(y_i \mathcal{N}_{\boldsymbol{\theta}}(\boldsymbol{x}_i)))^2 \right\} ,$$

where the second inequality is by orthogonal separability of the dataset.

Hence $\mathcal{L}(\boldsymbol{\theta}(T)) \geq \int_T^\infty \sum_{i=1}^n (\ell'(y_i \mathcal{N}_{\boldsymbol{\theta}}(\boldsymbol{x}_i)))^2 \, \mathrm{d}t$, so for all $\nu > 0$ there exists $t \in [T, \infty)$ such that $\sum_{i=1}^n (\ell'(y_i \mathcal{N}_{\boldsymbol{\theta}}(\boldsymbol{x}_i)))^2 < \nu$.

For exponential loss, let t_0 be such that $\sum_{i=1}^n (\ell'_{\exp}(y_i \mathcal{N}_{\boldsymbol{\theta}(t_0)}(\boldsymbol{x}_i)))^2 < 1/n^2$. Then for all $i \in [n]$ we have $y_i \mathcal{N}_{\boldsymbol{\theta}(t_0)}(\boldsymbol{x}_i) > \ln n$, so $\mathcal{L}(\boldsymbol{\theta}(t_0)) < 1 = \ell_{\exp}(0)$.

For logistic loss, let t_0 be such that $\sum_{i=1}^n (\ell'_{\log}(y_i \mathcal{N}_{\boldsymbol{\theta}(t_0)}(\boldsymbol{x}_i)))^2 < 1/(2n+1)^2$. Then for all $i \in [n]$ we have $y_i \mathcal{N}_{\boldsymbol{\theta}(t_0)}(\boldsymbol{x}_i) > \ln(2n)$ and thus $\ell_{\log}(y_i \mathcal{N}_{\boldsymbol{\theta}(t_0)}(\boldsymbol{x}_i)) < \ln(1 + 1/(2n)) < 1/(2n)$, so $\mathcal{L}(\boldsymbol{\theta}(t_0)) < 1/2 < \ln 2 = \ell_{\log}(0)$. □

The following theorem from recent literature is on the late phase of gradient flow, i.e. after attaining perfect accuracy and a small loss. Its assumptions on the network are satisfied in our setting (see Ji and Telgarsky [2020, Section 2]), in particular two-layer ReLU networks are positively 2-homogeneous, i.e. for all $\alpha > 0$, $\boldsymbol{\theta}$ and $\boldsymbol{x}$, we have $\mathcal{N}_{\alpha\boldsymbol{\theta}}(\boldsymbol{x}) = \alpha^2 \mathcal{N}_{\boldsymbol{\theta}}(\boldsymbol{x})$. For KKT conditions for nonsmooth optimisation problems, see Appendix E.

Theorem 14 (Lyu and Li [2020, Theorem A.8] and Ji and Telgarsky [2020, Theorem 3.1])**.** *Suppose $\mathcal{N}_{\boldsymbol{\theta}}$ is a neural network which is locally Lipschitz, positively homogeneous, and definable in some o-minimal structure that includes the exponential function. Consider minimising either the exponential or the logistic loss over a binary classification dataset $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)$ using gradient flow. If $\mathcal{L}(\boldsymbol{\theta}(0)) < \ell(0)$, then as $t \to \infty$ we have that $\mathcal{L}(\boldsymbol{\theta}(t)) \to 0$, $\|\boldsymbol{\theta}(t)\| \to \infty$, and the weights converge in direction to a Karush–Kuhn–Tucker point of the following maximum margin problem:*

$$\text{minimise} \quad \frac{1}{2}\|\boldsymbol{\theta}\|^2 \quad \text{subject to} \quad \forall i \in [n] : \; y_i \mathcal{N}_{\boldsymbol{\theta}}(\boldsymbol{x}_i) \geq 1 \; .$$

Another recent result is the next lemma, which shows that maximum-margin KKT points for two-layer ReLU networks on orthogonally separable datasets are particularly streamlined. Recall from Section 3 that $\boldsymbol{v}_1$ and $\boldsymbol{v}_{-1}$ denote the maximum margin vectors for the positive and negative data classes, respectively. We remark that it may be interesting that a part of the statement says that, regardless of whether the initialisation was balanced, the weights of any KKT point are balanced. We provide a proof of the lemma, which is perhaps simpler, and also it removes a marginal concern about applicability to our setting due to the leaky ReLU activation having been assumed in Lyu et al. [2021, Lemma B.9].

Lemma 15 (Lyu et al. [2021, Lemma H.3])**.** *Suppose $\boldsymbol{\theta}$ is a Karush–Kuhn–Tucker point of the problem in Theorem 14 for a two-layer ReLU network $\mathcal{N}_{\boldsymbol{\theta}}$ and an orthogonally separable dataset. Then for all $j \in [k]$ we have $|a_j| = \|\boldsymbol{w}_j\|$, and if $a_j \neq 0$ then*

$$\frac{\boldsymbol{w}_j}{\|\boldsymbol{w}_j\|} = \frac{\boldsymbol{v}_{\operatorname{sgn}(a_j)}}{\sum_{\operatorname{sgn}(a_{j'}) = \operatorname{sgn}(a_j)} a_{j'}^2} \; .$$

Proof. We have that $\boldsymbol{\theta}$ is feasible, i.e. $\forall i \in [n] : \; y_i \mathcal{N}_{\boldsymbol{\theta}}(\boldsymbol{x}_i) \geq 1$, and some $\lambda_1, \ldots, \lambda_n \geq 0$ satisfy

$$\boldsymbol{\theta} \in \sum_{i=1}^{n} \lambda_i \partial(y_i \mathcal{N}_{\boldsymbol{\theta}}(\boldsymbol{x}_i)) \quad \text{and} \quad \forall i \in [n] : \; \lambda_i = 0 \; \vee \; y_i \mathcal{N}_{\boldsymbol{\theta}}(\boldsymbol{x}_i) = 1 \; .$$

Thus, for all $j \in [k]$, we have $\boldsymbol{w}_j = a_j \sum_{i=1}^n \lambda_i y_i \psi'_{i,j} \boldsymbol{x}_i$ where $\psi'_{i,j} \in \partial\psi(\boldsymbol{w}_j^\top \boldsymbol{x}_i)$ are some values, and $a_j = \sum_{i=1}^n \lambda_i y_i \psi(\boldsymbol{w}_j^\top \boldsymbol{x}_i)$.

That $a_j = 0$ implies $\|\boldsymbol{w}_j\| = 0$ is immediate, so suppose $a_j > 0$.

For all $i' \in I_{-1}$ such that $\lambda_{i'} > 0$, we have by orthogonal separability that

$$\boldsymbol{w}_j^\top \boldsymbol{x}_{i'} = a_j \sum_{i=1}^n \lambda_i y_i \psi'_{i,j} \boldsymbol{x}_i^\top \boldsymbol{x}_{i'} \leq -a_j \lambda_{i'} \psi'_{i',j} \|\boldsymbol{x}_{i'}\|^2 \ ,$$

so $\psi'_{i',j} > 0$ is impossible, and thus $\psi'_{i',j} = 0$.

Hence $a_j = \sum_{i \in I_1} \lambda_i \psi(\boldsymbol{w}_j^\top \boldsymbol{x}_i)$, so there exists $i \in I_1$ such that $\lambda_i > 0$ and $\boldsymbol{w}_j^\top \boldsymbol{x}_i > 0$. Then for all $i' \in I_1$ we have $\boldsymbol{w}_j^\top \boldsymbol{x}_{i'} \geq a_j \lambda_i \boldsymbol{x}_i^\top \boldsymbol{x}_{i'} > 0$.

Therefore $a_j^2 = \boldsymbol{w}_j^\top a_j \sum_{i \in I_1} \lambda_i \boldsymbol{x}_i = \|\boldsymbol{w}_j\|^2$, so $a_j = \|\boldsymbol{w}_j\|$ and $\boldsymbol{w}_j / \|\boldsymbol{w}_j\| = \sum_{i \in I_1} \lambda_i \boldsymbol{x}_i =: \boldsymbol{u}_1$.

Analogously, if $a_j < 0$, then $-a_j = \|\boldsymbol{w}_j\|$ and $\boldsymbol{w}_j / \|\boldsymbol{w}_j\| = \sum_{i \in I_{-1}} \lambda_i \boldsymbol{x}_i =: \boldsymbol{u}_{-1}$.

For both signs $s \in \{\pm 1\}$, let $\kappa_s := \sum_{\mathrm{sgn}(a_j)=s} a_j^2$. By orthogonal separability, it follows that for all $i \in I_s$ we have $\mathcal{N}_{\boldsymbol{\theta}}(\boldsymbol{x}_i) = \sum_{\mathrm{sgn}(a_j)=s} a_j \boldsymbol{w}_j^\top \boldsymbol{x}_i = \kappa_s \boldsymbol{u}_s^\top \boldsymbol{x}_i$. Hence $\kappa_s \boldsymbol{u}_s$ is a Karush-Kuhn-Tucker point of the maximum margin problem

$$\text{minimise} \quad \frac{1}{2}\|\boldsymbol{v}\|^2 \quad \text{subject to} \quad \forall i \in I_s : \ \boldsymbol{v}^\top \boldsymbol{x}_i \geq 1 \ ,$$

and therefore its unique Karush-Kuhn-Tucker point, which is the maximum margin vector $\boldsymbol{v}_s$. □

The following corollary, restated from Section 3, is now an immediate consequence of Theorem 2, Theorem 14 and Lemma 15.

Corollary 3. *As the time tends to infinity, we have that the empirical loss converges to zero, the Euclidean norm of the weights converges to infinity, and the weights converge in direction to some $\boldsymbol{\theta}$ such that for all $j \in [k]$ we have $|a_j| = \|\boldsymbol{w}_j\|$, and if $a_j \neq 0$ then*

$$\frac{\boldsymbol{w}_j}{\|\boldsymbol{w}_j\|} = \frac{\boldsymbol{v}_{\mathrm{sgn}(a_j)}}{\sum_{\mathrm{sgn}(a_{j'})=\mathrm{sgn}(a_j)} a_{j'}^2} \ .$$

It remains to prove Proposition 4, also restated from Section 3.

Proposition 4. *Suppose network weights $\boldsymbol{\theta}$ are is in Corollary 3, class label mapping $m \in \{\pm 1\}$ is arbitrary, data model $\mathcal{D}$ is any $(\boldsymbol{\phi}, \rho, \tau)$-Bernoulli distribution such that $m \cos \angle(\boldsymbol{v}_1 - \boldsymbol{v}_{-1}, \boldsymbol{\phi}) < 0$, and adversarial program $\boldsymbol{p}$ is arbitrary. Then we have that*

$$\mathbb{P}_{(\boldsymbol{x},y)\sim\mathcal{D}}\{m\, y\, \mathcal{N}_{\boldsymbol{\theta}}(\boldsymbol{p}+\boldsymbol{x}) > 0\} \ \leq \ \frac{1}{2} + \frac{1}{2}\, e^{-2d\tau^2 \cos^2 \angle(\boldsymbol{v}_1 - \boldsymbol{v}_{-1}, \boldsymbol{\phi})} \ .$$

Proof. The cases $m = 1$ and $m = -1$ are symmetric by swapping $\boldsymbol{\phi}$ and y with $-\boldsymbol{\phi}$ and $-y$, so we can assume that $m = 1$.

Since

$$\begin{aligned}
&\mathbb{P}_{(\boldsymbol{x},y)\sim\mathcal{D}}\{y\, \mathcal{N}_{\boldsymbol{\theta}}(\boldsymbol{p}+\boldsymbol{x}) > 0\} \\
&\quad = \frac{1}{2}\, \mathbb{P}_{(\boldsymbol{x},y)\sim\mathcal{D}}\{y\, \mathcal{N}_{\boldsymbol{\theta}}(\boldsymbol{p}+\boldsymbol{x}) > 0 \ \mid \ y = 1\} + \frac{1}{2}\, \mathbb{P}_{(\boldsymbol{x},y)\sim\mathcal{D}}\{y\, \mathcal{N}_{\boldsymbol{\theta}}(\boldsymbol{p}+\boldsymbol{x}) > 0 \ \mid \ y = -1\} \ ,
\end{aligned}$$

it suffices to establish the following claim: there exists a data class $s \in \{\pm 1\}$ such that

$$\mathbb{P}_{(\boldsymbol{x},y)\sim\mathcal{D}}\{y\, \mathcal{N}_{\boldsymbol{\theta}}(\boldsymbol{p}+\boldsymbol{x}) > 0 \ \mid \ y = s\} \ \leq \ e^{-2d\tau^2 \cos^2 \angle(\boldsymbol{v}_1 - \boldsymbol{v}_{-1}, \boldsymbol{\phi})} \ .$$

Suppose $\boldsymbol{v}_1^\top \boldsymbol{p} \leq \boldsymbol{v}_{-1}^\top \boldsymbol{p}$, and let $s = 1$. Then we have that $\psi(\boldsymbol{v}_1^\top(\boldsymbol{p}+\boldsymbol{x})) > \psi(\boldsymbol{v}_{-1}^\top(\boldsymbol{p}+\boldsymbol{x}))$ implies $\boldsymbol{v}_1^\top \boldsymbol{x} > \boldsymbol{v}_{-1}^\top \boldsymbol{x}$, so

$$\begin{aligned}
\mathbb{P}_{(\boldsymbol{x},y)\sim\mathcal{D}}\{y\, \mathcal{N}_{\boldsymbol{\theta}}(\boldsymbol{p}+\boldsymbol{x}) > 0 \ \mid \ y = s\} &= \mathbb{P}_{(\boldsymbol{x},1)\sim\mathcal{D}}\{\mathcal{N}_{\boldsymbol{\theta}}(\boldsymbol{p}+\boldsymbol{x}) > 0\} \\
&= \mathbb{P}_{(\boldsymbol{x},1)\sim\mathcal{D}}\big\{\psi(\boldsymbol{v}_1^\top(\boldsymbol{p}+\boldsymbol{x})) > \psi(\boldsymbol{v}_{-1}^\top(\boldsymbol{p}+\boldsymbol{x}))\big\} \leq \mathbb{P}_{(\boldsymbol{x},1)\sim\mathcal{D}}\big\{\boldsymbol{v}_1^\top \boldsymbol{x} > \boldsymbol{v}_{-1}^\top \boldsymbol{x}\big\} \\
&\leq e^{-2d\tau^2 \cos^2 \angle(\boldsymbol{v}_1 - \boldsymbol{v}_{-1}, \boldsymbol{\phi})}
\end{aligned}$$

as required, where the last inequality is by the assumption $\cos\angle(\boldsymbol{v}_1 - \boldsymbol{v}_{-1}, \boldsymbol{\phi}) < 0$ and Schmidt et al. [2018, Lemma 26].

If $\boldsymbol{v}_1^\top \boldsymbol{p} \geq \boldsymbol{v}_{-1}^\top \boldsymbol{p}$, the argument to show the claim is analogous, with taking $s = -1$ and observing that $\psi(\boldsymbol{v}_1^\top(\boldsymbol{p}+\boldsymbol{x})) < \psi(\boldsymbol{v}_{-1}^\top(\boldsymbol{p}+\boldsymbol{x}))$ implies $\boldsymbol{v}_1^\top \boldsymbol{x} < \boldsymbol{v}_{-1}^\top \boldsymbol{x}$. □

G Gaussian data models

We show here how our main theoretical results can be adapted if the Bernoulli data models (see Section 2) we considered are replaced by data models based on the multivariate Gaussian distribution. The latter distribution has been standard for modelling data in theoretical investigations of adversarial examples (see e.g. Gilmer, Metz, Faghri, Schoenholz, Raghu, Wattenberg, and Goodfellow [2018], Schmidt et al. [2018], Ilyas et al. [2019]).

Definition. Following Schmidt et al. [2018], given a per-class mean vector $\varrho\boldsymbol{\varphi}$ where $\varrho > 0$ is its radius and $\boldsymbol{\varphi} \in \mathbb{S}^{d-1}$ is its direction, and given a variance parameter $\varsigma > 0$, we define the $(\boldsymbol{\varphi}, \varrho, \varsigma)$-Gaussian distribution over $(\boldsymbol{x}, y) \in \mathbb{R}^d \times \{\pm 1\}$ as follows:

- first draw the label y uniformly at random from $\{\pm 1\}$,
- then sample the data point $\boldsymbol{x}$ from the spherical multivariate Gaussian distribution whose mean is $y\varrho\boldsymbol{\varphi}$ and covariance matrix is $\varsigma^2 \boldsymbol{I}_d$.

These binary classification data models are thus mixtures of two spherical Gaussians, one per data class. The overlapping of the classes, and therefore the difficulty of the classification task, is controlled by the ratio between the spherical radius $\sqrt{d}\varsigma$ and the mean radius ϱ, i.e. classification becomes harder as $\sqrt{d}\varsigma/\varrho$ increases.

G.1 Random networks

We assume that, as in Section 2, we have a random two-layer ReLU network $\mathcal{N}$ with input dimension d, width k such that $k \leq d$, and weights $\boldsymbol{w}_j$ and a_j for $j \in [k]$.

Let $\boldsymbol{p}$ be an adversarial program defined as in (1), with respect to the direction $\boldsymbol{\varphi}$ of the Gaussian data model.

By adapting the proof of Theorem 8 to the Gaussian adversarial task, we obtain the following result, which provides similar lower (respectively, upper) bounds on the output of the adversarially reprogrammed network for positively (respectively, negatively) labelled adversarial inputs.

Theorem 16. *There exist positive constants C_1, C_2, C_3 and C_4 such that, with probability at least $(1 - C_1\gamma)(1 - 2\gamma^\dagger)$ over the draw of the network $\mathcal{N}$ and the labelled data point $(\boldsymbol{x}, y)$ from the $(\boldsymbol{\varphi}, \varrho, \varsigma)$-Gaussian distribution, provided that $\varrho^2/\varsigma^2 \geq 2\ln(1/\gamma^\dagger)$, we have*

$$y\,\mathcal{N}(\boldsymbol{p}+\boldsymbol{x}) > \frac{\sqrt{k}}{\sqrt{d}}\left(\frac{C_2}{2}\beta - \beta'\left(C_3 \exp\left(-\frac{d^2}{2k\beta'^2}\right)\min\left\{1, \frac{k\beta'^2}{d^2}\right\} + C_4\sqrt{\frac{\ln(1/\gamma)}{k}}\right)\right) ,$$

where

$$\beta \coloneqq \varrho - \varsigma\sqrt{2\ln(1/\gamma^\dagger)} \quad \textit{and} \quad \beta' \coloneqq \varrho + \varsigma(\sqrt{d} + \sqrt{2\ln(1/\gamma^\dagger)}) .$$

Proof. For any fixed $(\boldsymbol{x}, y)$ such that $y\boldsymbol{\varphi}^\top\boldsymbol{x} > 0$, we can apply Lemmas 6 and 7 as in the proof of Theorem 8, which gives us that with probability at least $1 - C_1\gamma$ over the draw of the network $\mathcal{N}$ one has

$$y\,\mathcal{N}(\boldsymbol{p}+\boldsymbol{x}) > \frac{\sqrt{k}}{\sqrt{d}}\left(\frac{C_2}{2}y\boldsymbol{\varphi}^\top\boldsymbol{x} - \|\boldsymbol{x}\|\left(C_3 \exp\left(-\frac{d^2}{2k\|\boldsymbol{x}\|^2}\right)\min\left\{1, \frac{k\|\boldsymbol{x}\|^2}{d^2}\right\} - C_4\sqrt{\frac{\ln(1/\gamma)}{k}}\right)\right) ,$$

where C_1, C_2, C_3 and C_4 are positive constants defined as in the proof of Theorem 8.

On the other hand, for $(\boldsymbol{x}, y)$ sampled from the $(\boldsymbol{\varphi}, \varrho, \varsigma)$-Gaussian distribution, Schmidt et al. [2018, Lemma 13] tell us that with probability at least $1 - \gamma^\dagger$ one has

$$\|\boldsymbol{x}\| < \varrho + \varsigma(\sqrt{d} + \sqrt{2\ln(1/\gamma^\dagger)}) ,$$

and again Schmidt et al. [2018, Lemma 15] tell us that with probability at least $1-\gamma^{\ddagger}$ one has

$$y\boldsymbol{\varphi}^{\top}\boldsymbol{x} > \varrho - \varsigma\sqrt{2\ln(1/\gamma^{\ddagger})}\ .$$

To complete the proof, we take $\gamma^{\dagger}=\gamma^{\ddagger}$ and combine the inequalities above. □

Next we obtain a counterpart of Corollary 1. It shows that, for sufficiently large input dimensions d, and under relatively mild restrictions on the growth rates of the network width k, the mean radius ϱ and the variance parameter ς of the Gaussian data model, expected reprogramming accuracy can be arbitrarily close to 100%. Particularly interesting is the constraint $\eta_{(\varrho)}-\eta_{(\varsigma)} > 1/2-\eta_{(k)}/2$, which amounts to $\sqrt{d}\varsigma/\varrho = o(\sqrt{k})$, i.e. the adversarial task is required to be not too difficult, where greater difficulty may be allowed by greater network width.

Corollary 17. *Suppose that*

$$k=\Theta(d^{\eta_{(k)}})\ ,\quad \varrho=\Theta(d^{\eta_{(\varrho)}})\quad \textit{and}\quad \varsigma=O(d^{\eta_{(\varsigma)}})\ ,$$

where $\eta_{(k)},\eta_{(\varrho)},\eta_{(\varsigma)}\in[0,1]$ *are arbitrary constants that satisfy*

$$\eta_{(k)}>0\ ,\quad \eta_{(\varrho)}<1-\eta_{(k)}/2\ ,\quad \eta_{(\varsigma)}<1/2-\eta_{(k)}/2\quad \textit{and}\quad \eta_{(\varrho)}-\eta_{(\varsigma)}>1/2-\eta_{(k)}/2\ .$$

Then, for sufficiently large input dimensions d*, the expected accuracy of the adversarially reprogrammed network* $\mathcal{N}$ *on the* $(\boldsymbol{\varphi},\varrho,\varsigma)$*-Gaussian data model is arbitrarily close to* 100%.

Proof. Fix γ and $\gamma^{\dagger}$ such that $(1-C_1\gamma)(1-2\gamma^{\dagger})$ is as close to 100% as required.

By the assumptions in the statement, we have

$$\varrho/\varsigma=\Omega(d^{\eta_{(\varrho)}})/O(d^{\eta_{(\varsigma)}})=\Omega(d^{\eta_{(\varrho)}-\eta_{(\varsigma)}})=\omega_d(1)\ ,$$

so $\varrho^2/\varsigma^2\geq 2\ln(1/\gamma^{\dagger})$ for large enough d; moreover, we have

$$\beta=\Omega(d^{\eta_{(\varrho)}})\ ,\quad \beta'=O(d^{\max\{\eta_{(\varrho)},\eta_{(\varsigma)}+1/2\}})\quad \text{and}\quad \exp\left(\frac{d^2}{k\beta'^2}\right)=\omega(d^2)\ .$$

The corollary follows by applying Theorem 16 and observing that

$$\begin{aligned}
&\frac{C_2}{2}\beta-\beta'\left(C_3\exp\left(-\frac{d^2}{2k\beta'^2}\right)\min\left\{1,\frac{k\beta'^2}{d^2}\right\}+C_4\sqrt{\frac{\ln(1/\gamma)}{k}}\right)\\
&\qquad=\Omega(d^{\eta_{(\varrho)}})-O(d^{\max\{\eta_{(\varrho)},\eta_{(\varsigma)}+1/2\}})\left(\frac{1}{\omega(d)\ln(\omega(d^2))}+\frac{1}{\Omega(d^{\eta_{(k)}/2})}\right)\\
&\qquad=\Omega(d^{\eta_{(\varrho)}})-O(d^{\max\{\eta_{(\varrho)},\eta_{(\varsigma)}+1/2\}-\eta_{(k)}/2})=\Omega(d^{\eta_{(\varrho)}})-o(d^{\eta_{(\varrho)}})=\Omega(d^{\eta_{(\varrho)}})\ ,
\end{aligned}$$

which is positive for large enough d. □

G.2 Implicit bias

We consider, as in Section 3, a trajectory of gradient flow for a two-layer ReLU network $\mathcal{N}_{\boldsymbol{\theta}}$ with input dimension d and width k, trained on an orthogonally separable binary classification dataset, from a balanced and live initialisation, using either the exponential or the logistic loss function. Also as in Section 3, the assumption $k\leq d$ is not needed here.

Hence Corollary 3 still applies, and we restate here for convenience. Recall that $\boldsymbol{w}_j$ are first-layer network weights, a_j are second-layer network weights, and $\boldsymbol{v}_1$ and $\boldsymbol{v}_{-1}$ are the maximum-margin vectors for the positive and negative data classes (respectively).

Corollary 3. *As the time tends to infinity, we have that the empirical loss converges to zero, the Euclidean norm of the weights converges to infinity, and the weights converge in direction to some* $\boldsymbol{\theta}$ *such that for all* $j\in[k]$ *we have* $|a_j|=\|\boldsymbol{w}_j\|$*, and if* $a_j\neq 0$ *then*

$$\frac{\boldsymbol{w}_j}{\|\boldsymbol{w}_j\|}=\frac{\boldsymbol{v}_{\operatorname{sgn}(a_j)}}{\sum_{\operatorname{sgn}(a_{j'})=\operatorname{sgn}(a_j)}a_{j'}^2}\ .$$

By adapting the proof of Proposition 4 to the Gaussian adversarial task, we obtain the following result about consequences for adversarial reprogramming of the long training of the network. It tells us that, for any class label mapping from the original to the adversarial task, for any Gaussian data model whose mean direction is in a half-space of the difference of the maximum-margin vectors, and for any adversarial program, the accuracy tends to $1/2$ provided that ϱ/ς tends to infinity, i.e. the difficulty $\sqrt{d}\varsigma/\varrho$ of the data model increases slower than the square root of the input dimension d. The latter constraint mirrors the corresponding conclusion for the Bernoulli data model in Proposition 4, and is again considerably weaker than in the results of Schmidt et al. [2018] on the Gaussian data model, where $\varrho = \sqrt{d}$ and $\varsigma = O(\sqrt[4]{d})$ are assumed.

Proposition 18. *Suppose network weights $\boldsymbol{\theta}$ are is in Corollary 3, class label mapping $m \in \{\pm 1\}$ is arbitrary, data model $\mathcal{D}$ is any $(\boldsymbol{\varphi}, \varrho, \varsigma)$-Gaussian distribution such that $m \cos \angle(\boldsymbol{v}_1 - \boldsymbol{v}_{-1}, \boldsymbol{\varphi}) < 0$, and adversarial program $\boldsymbol{p}$ is arbitrary. Then we have that*

$$\mathbb{P}_{(\boldsymbol{x},y)\sim\mathcal{D}}\{m\, y\, \mathcal{N}_{\boldsymbol{\theta}}(\boldsymbol{p}+\boldsymbol{x}) > 0\} \;\leq\; \frac{1}{2} + \frac{1}{2}\exp\left(-\frac{\varrho^2 \cos^2 \angle(\boldsymbol{v}_1 - \boldsymbol{v}_{-1}, \boldsymbol{\varphi})}{2\varsigma^2}\right) .$$

Proof. The argument is as in the proof of Proposition 4, except that we invoke Schmidt et al. [2018, Lemma 17] to show that $\cos \angle(\boldsymbol{v}_1 - \boldsymbol{v}_{-1}, \boldsymbol{\varphi}) < 0$ implies

$$\mathbb{P}_{(\boldsymbol{x},1)\sim\mathcal{D}}\left\{\boldsymbol{v}_1^\top \boldsymbol{x} > \boldsymbol{v}_{-1}^\top \boldsymbol{x}\right\} \leq \exp\left(-\frac{\varrho^2 \cos^2 \angle(\boldsymbol{v}_1 - \boldsymbol{v}_{-1}, \boldsymbol{\varphi})}{2\varsigma^2}\right) . \quad \square$$

H Further details on experiments

Here we supplement Section 4 by presenting experimental results on adversarial tasks other than MNIST, and listing the data behind the test accuracy plots.

H.1 Experiments using other datasets

We repeated our experiments with the Fashion-MNIST [Xiao et al., 2017] dataset as the adversarial task instead of MNIST, while keeping the rest of the experimental setup unchanged. Fashion-MNIST also consists of 60,000 training images and 10,000 test images, and it is available under the MIT licence. It has 10 labels: 0 for T-shirt/top, 1 for Trouser, 2 for Pullover, etc.

The average test accuracies are plotted in Figure 4. The compute involved is similar to that for MNIST, which we discussed in Section 4.

We then repeated our experiments with the Kuzushiji-MNIST [Clanuwat et al., 2018] dataset as the adversarial task, again keeping the rest of the experimental setup unchanged. Kuzushiji-MNIST also consists of 60,000 training images and 10,000 test images, and it is available under the Creative Commons Attribution Share-Alike 4.0 licence. It consists of 10 classes of handwritten Japanese characters.

The average test accuracies are plotted in Figure 5, and the compute involved is again similar to that for MNIST.

H.2 Data from experiments

Please see Tables 1, 2, 3, 4, 5, and 6.

We remark that, in the cases where the standard deviation is relatively large, that is due to some of the trials failing to find an adversarial program with non-trivial test accuracy, and the remaining trials succeeding. It might be interesting in future work to investigate this apparent bimodality.

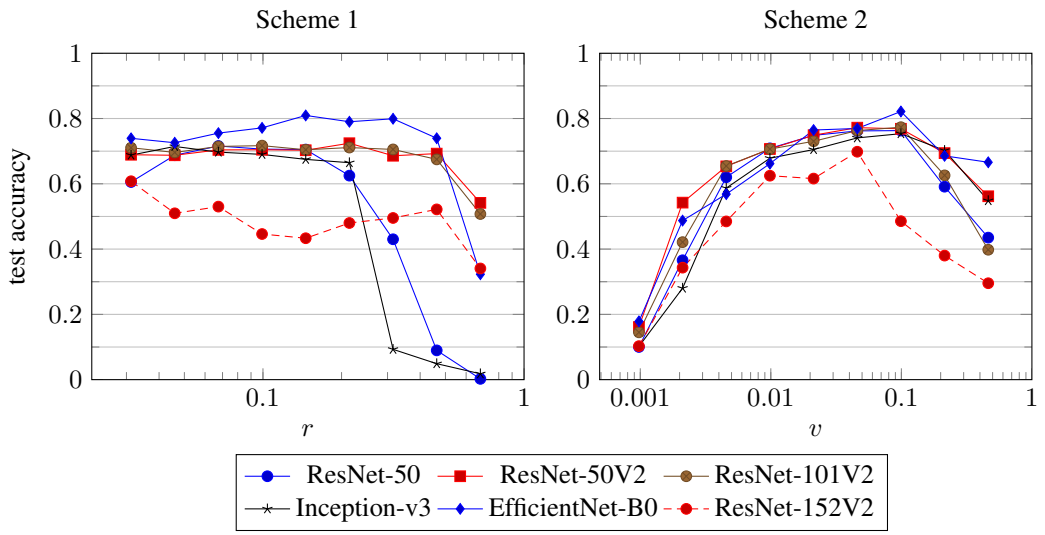

Figure 4: The accuracy achieved by the adversarial program on the Fashion-MNIST test set for different parameters of the two schemes of combining input images with adversarial programs. The horizontal axes are logarithmic. The values plotted are averages over 5 trials, which are listed together with the standard deviations in Appendix H.2.

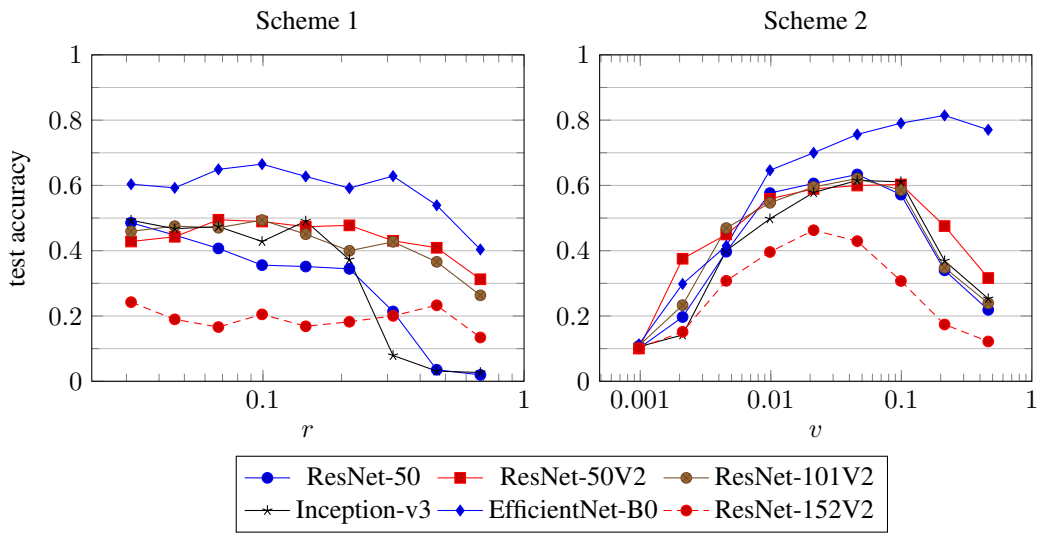

Figure 5: The accuracy achieved by the adversarial program on the Kuzushiji-MNIST test set for different parameters of the two schemes of combining input images with adversarial programs. The horizontal axes are logarithmic. The values plotted are averages over 5 trials, which are listed together with the standard deviations in Appendix H.2.

Table 1: Accuracy averages (%) on the MNIST test set, which are plotted in Figure 3 on the left, with standard deviations (%) shown in parentheses, over 5 trials.

r	Efficient Net-B0	Inception-v3	ResNet-101V2	ResNet-152V2	ResNet-50	ResNet-50V2
2^{-5}	77.4 (8.0)	77.3 (4.0)	68.9 (9.1)	38.2 (15.0)	74.0 (2.4)	67.3 (5.9)
$2^{-\frac{40}{9}}$	84.0 (4.8)	81.0 (8.2)	74.1 (4.7)	34.4 (14.2)	74.5 (8.9)	69.1 (4.4)
$2^{-\frac{35}{9}}$	88.4 (2.0)	76.2 (10.4)	74.2 (7.5)	19.6 (10.8)	73.1 (7.0)	70.0 (6.6)
$2^{-\frac{30}{9}}$	88.5 (1.6)	77.3 (11.0)	78.3 (4.8)	33.6 (25.9)	67.2 (15.1)	77.1 (2.9)
$2^{-\frac{25}{9}}$	88.9 (2.0)	78.2 (15.1)	79.1 (5.8)	29.9 (18.9)	63.0 (30.9)	76.1 (5.8)
$2^{-\frac{20}{9}}$	87.6 (2.1)	50.7 (35.7)	74.0 (3.0)	30.7 (11.6)	62.2 (10.7)	74.1 (3.6)
$2^{-\frac{15}{9}}$	66.4 (37.4)	28.0 (14.9)	62.7 (1.9)	36.6 (18.6)	23.0 (25.0)	69.8 (5.5)
$2^{-\frac{10}{9}}$	51.5 (45.1)	9.0 (9.9)	52.8 (22.2)	33.7 (9.4)	7.6 (7.6)	62.6 (9.1)
$2^{-\frac{5}{9}}$	40.2 (41.4)	3.3 (3.2)	43.6 (21.8)	17.2 (8.7)	9.8 (7.0)	39.1 (6.1)

Table 2: Accuracy averages (%) on the MNIST test set, which are plotted in Figure 3 on the right, with standard deviations (%) shown in parentheses, over 5 trials.

v	Efficient Net-B0	Inception-v3	ResNet-101V2	ResNet-152V2	ResNet-50	ResNet-50V2
2^{-10}	13.2 (3.7)	11.0 (1.6)	10.5 (0.5)	10.2 (0.3)	10.2 (0.8)	10.4 (0.7)
$2^{-\frac{80}{9}}$	52.5 (3.2)	27.6 (24.4)	30.3 (23.6)	13.7 (7.1)	43.2 (29.1)	63.7 (9.4)
$2^{-\frac{70}{9}}$	70.4 (4.8)	68.6 (11.0)	75.6 (7.8)	46.0 (24.9)	79.7 (6.4)	73.9 (4.9)
$2^{-\frac{60}{9}}$	86.3 (7.0)	81.9 (6.2)	84.7 (2.6)	65.0 (12.2)	88.4 (1.5)	85.3 (2.1)
$2^{-\frac{50}{9}}$	88.9 (6.2)	88.3 (1.6)	88.4 (0.9)	68.9 (15.9)	89.8 (1.6)	86.4 (3.5)
$2^{-\frac{40}{9}}$	91.8 (3.0)	90.2 (1.4)	90.9 (0.7)	72.0 (10.0)	91.1 (1.2)	88.8 (1.1)
$2^{-\frac{30}{9}}$	92.6 (6.6)	90.7 (1.5)	84.0 (3.6)	58.0 (16.5)	88.4 (1.4)	81.6 (9.3)
$2^{-\frac{20}{9}}$	95.0 (2.0)	82.8 (10.1)	59.7 (7.0)	29.8 (17.1)	70.8 (9.0)	73.0 (5.0)
$2^{-\frac{10}{9}}$	95.1 (2.2)	56.7 (15.5)	42.9 (12.5)	23.8 (6.7)	64.5 (10.1)	56.6 (11.0)

Table 3: Accuracy averages (%) on the Fashion-MNIST test set, which are plotted in Figure 4 on the left, with standard deviations (%) shown in parentheses, over 5 trials.

r	Efficient Net-B0	Inception-v3	ResNet-101V2	ResNet-152V2	ResNet-50	ResNet-50V2
2^{-5}	73.9 (2.6)	68.8 (3.5)	71.1 (1.4)	60.8 (10.1)	60.5 (28.3)	68.9 (1.6)
$2^{-\frac{40}{9}}$	72.6 (5.4)	71.4 (2.3)	69.6 (2.8)	51.0 (9.3)	68.7 (2.7)	68.7 (1.8)
$2^{-\frac{35}{9}}$	75.5 (4.3)	69.7 (5.1)	71.4 (3.3)	53.0 (10.9)	71.6 (3.8)	70.4 (2.4)
$2^{-\frac{30}{9}}$	77.1 (6.0)	69.0 (4.8)	71.7 (2.0)	44.6 (14.1)	70.6 (3.2)	70.4 (2.5)
$2^{-\frac{25}{9}}$	80.9 (2.8)	67.5 (6.0)	70.4 (3.5)	43.3 (13.3)	70.5 (2.3)	70.3 (2.5)
$2^{-\frac{20}{9}}$	79.0 (6.9)	66.5 (4.1)	71.1 (1.8)	48.0 (9.9)	62.5 (13.4)	72.4 (3.2)
$2^{-\frac{15}{9}}$	79.9 (6.4)	9.3 (5.9)	70.5 (4.3)	49.5 (12.4)	43.0 (17.8)	68.6 (8.6)
$2^{-\frac{10}{9}}$	74.0 (9.1)	4.9 (5.7)	67.5 (3.6)	52.2 (7.8)	9.0 (17.7)	69.2 (5.3)
$2^{-\frac{5}{9}}$	32.3 (44.2)	1.8 (3.3)	50.7 (8.0)	34.1 (12.6)	0.2 (0.6)	54.2 (8.5)

Table 4: Accuracy averages (%) on the Fashion-MNIST test set, which are plotted in Figure 4 on the right, with standard deviations (%) shown in parentheses, over 5 trials.

v	Efficient Net-B0	Inception-v3	ResNet-101V2	ResNet-152V2	ResNet-50	ResNet-50V2
2^{-10}	17.9 (11.5)	10.2 (0.2)	14.5 (8.1)	10.3 (0.6)	10.0 (0.2)	16.1 (10.5)
$2^{-\frac{80}{9}}$	48.7 (6.9)	28.0 (7.0)	42.1 (7.2)	34.3 (13.1)	36.6 (16.2)	54.2 (5.2)
$2^{-\frac{70}{9}}$	56.8 (9.5)	58.7 (4.4)	65.4 (2.6)	48.5 (5.7)	62.0 (4.1)	65.3 (5.4)
$2^{-\frac{60}{9}}$	66.2 (6.9)	67.8 (3.4)	70.6 (2.0)	62.5 (7.3)	70.9 (2.7)	70.7 (2.8)
$2^{-\frac{50}{9}}$	76.5 (1.4)	70.5 (1.6)	73.0 (1.9)	61.6 (6.2)	74.8 (1.2)	74.9 (1.6)
$2^{-\frac{40}{9}}$	77.0 (4.0)	74.1 (1.6)	76.3 (0.6)	69.8 (4.2)	76.2 (2.3)	77.2 (1.1)
$2^{-\frac{30}{9}}$	82.2 (2.0)	75.4 (1.5)	77.3 (1.0)	48.6 (8.1)	76.3 (2.3)	76.9 (0.9)
$2^{-\frac{20}{9}}$	68.6 (32.9)	70.2 (5.7)	62.6 (1.9)	38.0 (15.2)	59.1 (11.8)	69.4 (3.7)
$2^{-\frac{10}{9}}$	66.6 (37.3)	54.8 (12.3)	39.8 (10.3)	29.5 (6.7)	43.5 (5.6)	56.2 (4.1)

Table 5: Accuracy averages (%) on the Kuzushiji-MNIST test set, which are plotted in Figure 5 on the left, with standard deviations (%) shown in parentheses, over 5 trials.

r	Efficient Net-B0	Inception-v3	ResNet-101V2	ResNet-152V2	ResNet-50	ResNet-50V2
2^{-5}	60.4 (3.4)	49.3 (4.3)	45.9 (2.9)	24.2 (5.3)	48.6 (1.5)	42.8 (4.8)
$2^{-\frac{40}{9}}$	59.3 (1.2)	46.7 (5.5)	47.5 (4.7)	19.0 (1.4)	44.8 (3.7)	44.3 (1.8)
$2^{-\frac{35}{9}}$	64.9 (3.6)	47.4 (4.8)	47.0 (3.0)	16.6 (5.5)	40.7 (6.0)	49.5 (3.3)
$2^{-\frac{30}{9}}$	66.5 (0.7)	42.8 (3.4)	49.4 (2.4)	20.5 (4.3)	35.6 (9.5)	48.9 (3.8)
$2^{-\frac{25}{9}}$	62.7 (3.5)	49.1 (5.2)	45.0 (4.6)	16.8 (4.4)	35.2 (4.4)	47.4 (3.8)
$2^{-\frac{20}{9}}$	59.2 (13.6)	37.3 (17.2)	40.0 (2.5)	18.2 (3.3)	34.4 (7.0)	47.8 (1.0)
$2^{-\frac{15}{9}}$	62.9 (2.8)	7.9 (7.8)	42.7 (4.0)	20.0 (4.5)	21.4 (13.5)	43.0 (2.5)
$2^{-\frac{10}{9}}$	53.9 (6.0)	3.1 (4.1)	36.6 (13.6)	23.3 (5.6)	3.5 (3.5)	40.9 (5.6)
$2^{-\frac{5}{9}}$	40.4 (24.1)	2.7 (4.3)	26.3 (11.7)	13.4 (12.6)	2.0 (2.2)	31.3 (10.6)

Table 6: Accuracy averages (%) on the Kuzushiji-MNIST test set, which are plotted in Figure 5 on the right, with standard deviations (%) shown in parentheses, over 5 trials.

v	Efficient Net-B0	Inception-v3	ResNet-101V2	ResNet-152V2	ResNet-50	ResNet-50V2
2^{-10}	11.3 (1.8)	10.6 (1.3)	10.9 (0.9)	10.2 (0.2)	10.1 (0.2)	10.0 (0.0)
$2^{-\frac{80}{9}}$	29.8 (14.4)	14.2 (6.8)	23.3 (6.2)	15.2 (3.2)	19.7 (4.8)	37.5 (5.4)
$2^{-\frac{70}{9}}$	41.5 (17.6)	40.0 (2.0)	46.9 (2.6)	30.8 (3.1)	39.7 (16.9)	45.0 (5.2)
$2^{-\frac{60}{9}}$	64.6 (3.2)	49.8 (2.4)	54.7 (3.4)	39.6 (6.0)	57.7 (1.2)	55.9 (3.5)
$2^{-\frac{50}{9}}$	70.0 (3.8)	57.8 (2.2)	59.5 (2.2)	46.3 (1.9)	60.6 (1.2)	58.9 (2.8)
$2^{-\frac{40}{9}}$	75.6 (3.0)	61.5 (1.8)	62.1 (2.0)	42.9 (6.9)	63.4 (1.3)	60.0 (2.0)
$2^{-\frac{30}{9}}$	79.1 (3.0)	61.1 (2.4)	58.6 (5.1)	30.7 (7.5)	57.2 (3.4)	60.3 (3.0)
$2^{-\frac{20}{9}}$	81.4 (4.8)	36.9 (5.6)	34.8 (3.7)	17.4 (4.0)	34.0 (4.8)	47.6 (1.8)
$2^{-\frac{10}{9}}$	77.1 (3.8)	25.3 (8.1)	23.9 (5.9)	12.2 (7.4)	21.9 (5.2)	31.6 (6.9)