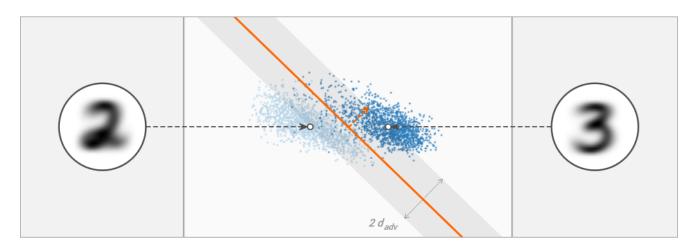# A New Angle on L2 Regularization

(interactive version available at `https://thomas-tanay.github.io/post--L2-regularization/`*)*

L2 Regularization:

$err_{\mathrm{train}} = 2.6\%$

$d_{\mathrm{adv}} = 1.5$

Imagine two high dimensional clusters and a hyperplane separating them.

Consider in particular **the angle between**:
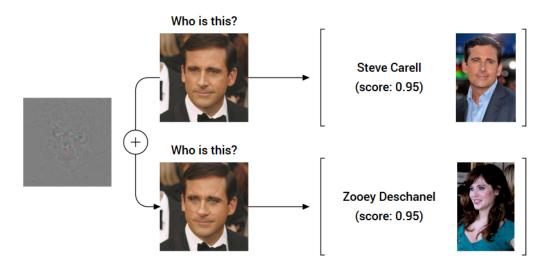the direction joining the two clusters' centroids and the normal to the hyperplane.

In linear classification, this angle depends on the level of L2 regularization used.
— *Can you explain why?* —

Thomas Tanay　　**Lewis D Griffin**

CoMPLEX, UCL　　**CoMPLEX, UCL**

Deep neural networks have been shown to be vulnerable to the adversarial example phenomenon: all models tested so far can have their classifications dramatically altered by small image perturbations [1, 2]. The following predictions were for instance made by a state-of-the-art network trained to recognize celebrities [3]:

This result is puzzling for two reasons. First, it challenges a common belief according to which good generalization to novel data and robustness to small perturbations go hand in hand. Second, it constitutes a potential threat to real-world applications [4–6]. Researchers at MIT have for instance recently constructed 3D objects that are misclassified under a wide distribution of angles and viewpoints [7]. Understanding this phenomenon and improving deep networks' robustness has thus become an important research objective.

Several approaches have been explored already. The phenomenon has been described in detail [8, 9] and some theoretical analysis has been provided [10–12]. Attempts have been made at designing more robust architectures [13–16] or at detecting adversarial examples during evaluation [17–20]. *Adversarial training* has also been introduced as a new regularization technique penalising adversarial directions [2, 21–23]. Unfortunately, the problem remains largely unresolved [24, 25]. Confronted with this difficulty, we propose to proceed from fundamentals: focusing on linear classification first and then increasing complexity incrementally.
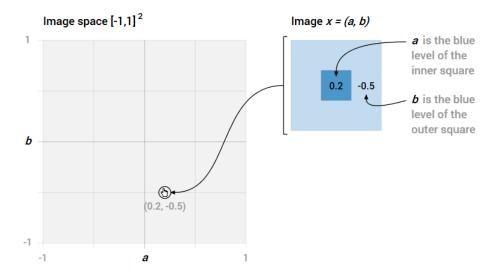
A Toy Problem

In linear classification, adversarial perturbations are often understood as a property of the dot product in high dimension. A widespread intuition is that: "for high dimensional problems, we can make many infinitesimal changes to the input that add up to one large change to the output" [2]. Here, we challenge this intuition and argue instead that adversarial examples exist when the classification boundary lies close to the data manifold—independently of the image space dimension.[1]
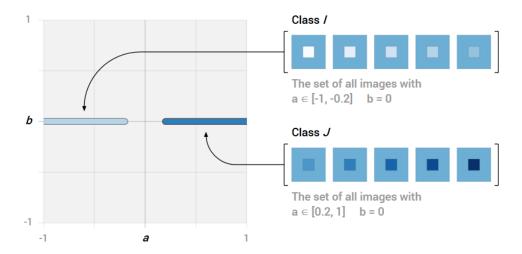
Setup

Let's start with a minimal toy problem: a two-dimensional image space where each image is a function of a and b.
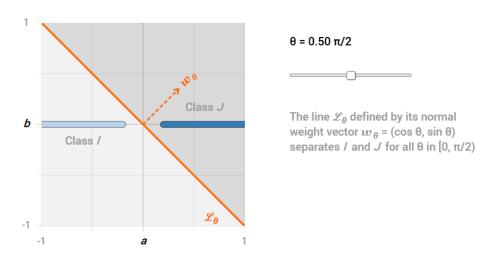
---

[1]This idea was originally inspired by the work of Marron et al. [26] on the *data piling phenomenon* affecting SVMs on high-dimensional low sample size data.

Image space [-1,1]^2

Image $x = (a, b)$

a is the blue level of the inner square

0.2   -0.5

b is the blue level of the outer square

(0.2, -0.5)

In this simple image space, we define two classes of images...

Class I

The set of all images with $a \in [-1, -0.2]$ $b = 0$
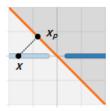
Class J

The set of all images with $a \in [0.2, 1]$ $b = 0$

...which can be separated by an infinite number of linear classifiers. Consider for instance the line $\mathscr{L}_\theta$.

w_θ

Class J

Class I

$\mathscr{L}_\theta$

$\theta = 0.50 \, \pi/2$

The line $\mathscr{L}_\theta$ defined by its normal weight vector $w_\theta = (\cos\theta, \sin\theta)$ separates I and J for all θ in $[0, \pi/2)$

This raises a first question: if all the linear classifiers $\mathscr{L}_\theta$ separate I and J equally well, are they all equally robust to image perturbations?

Projected and mirror images

Consider an image x in class I. The closest image classified in the opposite class is *the projected image* of x on $\mathscr{L}_\theta$:
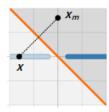

$$x_p := x - (x \cdot w_\theta)\, w_\theta$$

Low confidence adversarial example

When x and x_p are very close to each other, we say that x_p is an *adversarial example* of x. Observe though that x_p is classified with a low confidence score (it lies on the boundary) and it is perhaps more interesting to consider *high-confidence adversarial examples* [25].

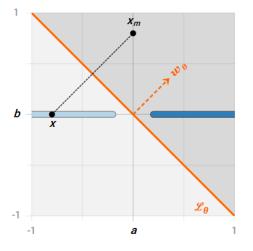In the following, we focus on the *mirror image* of x through $\mathscr{L}_\theta$:

$$x_m := x - 2 (x \cdot w_\theta)\, w_\theta$$
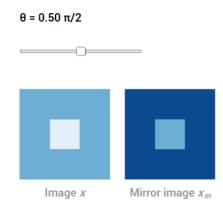
High confidence adversarial example

By construction, x and x_m are at the same distance from the boundary and are classified with the same confidence level.
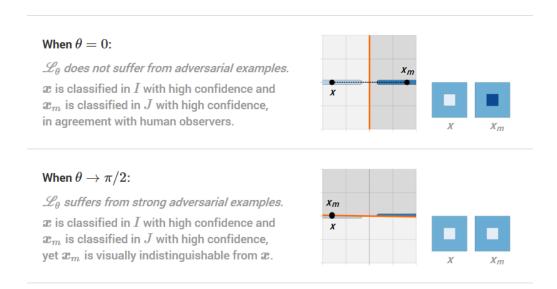
A mirror image as a function of θ

Coming back to our toy problem, we can now plot an image x and its mirror image x_m as a function of θ.

$\theta = 0.50\ \pi/2$

Image x Mirror image x_m

We see that the distance between $\boldsymbol{x}$ and $\boldsymbol{x}_m$ depends on the angle θ. The two borderline cases are of particular interest.
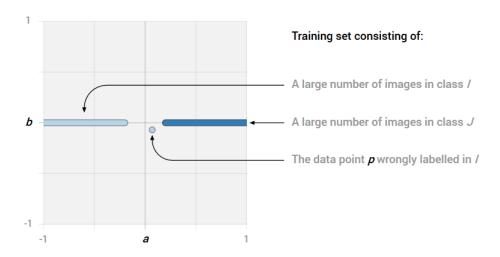

When $\theta = 0$:

$\mathscr{L}_\theta$ does not suffer from adversarial examples.

$\boldsymbol{x}$ is classified in I with high confidence and $\boldsymbol{x}_m$ is classified in J with high confidence, in agreement with human observers.

When $\theta \to \pi/2$:

$\mathscr{L}_\theta$ suffers from strong adversarial examples.

$\boldsymbol{x}$ is classified in I with high confidence and $\boldsymbol{x}_m$ is classified in J with high confidence, yet $\boldsymbol{x}_m$ is visually indistinguishable from $\boldsymbol{x}$.

This raises a second question: if adversarial examples exist when $\mathscr{L}_\theta$ is strongly tilted, what makes $\mathscr{L}_\theta$ tilt in practice?

Overfitting and L2 regularization

Our working hypothesis is that the classification boundary defined by standard linear learning algorithms tilts by overfitting noisy data points in the training set. This hypothesis is supported by the theoretical result of Xu et al. [27] relating robustness to regularization in Support Vector Machines (SVM). It can also be tested experimentally: techniques designed to reduce overfitting such as L2 regularization are expected to mitigate the adversarial example phenomenon.

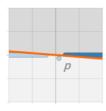Consider for instance a training set containing one noisy data point $\boldsymbol{p}$.

Training set consisting of:

A large number of images in class I

A large number of images in class J

The data point $\boldsymbol{p}$ wrongly labelled in I

If we train an SVM or a logistic regression model on this training set, we observe two possible behaviours.

Without L2 regularization:

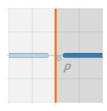*The classification boundary is strongly tilted.*

Most of the leeway available to fit the training data resides in the tilting angle of the boundary. Here, the data point p can be classified correctly, but the classifier obtained is then vulnerable to adversarial examples.

With L2 regularization:

The classification boundary is not tilted.

L2 regularization reduces overfitting by allowing some training samples to be misclassified. When enough regularization is used, the data point p is ignored and the classifier obtained is robust to adversarial examples.

At this point, one might legitimately wonder—what does a 1-dimensional data manifold lying in a 2-dimensional image space have to do with high-dimensional natural images?

Adversarial Examples in Linear Classification

In the following, we show that the two main ideas introduced in the previous toy problem stay valid in the general case: adversarial examples exist when the classification boundary lies close to the data manifold and L2 regularization controls the tilting angle of the boundary.

Scaling the Loss Function

Let's start with a simple observation: during training, the norm of the weight vector acts as a scaling parameter on the loss function.

Setup

Let I and J be two classes of images and $\mathcal{C}$ a hyperplane boundary defining a linear classifier in $\mathbb{R}^d$. $\mathcal{C}$ is specified by a normal weight vector $\boldsymbol{w}$ and a bias b. For an image $\boldsymbol{x}$ in $\mathbb{R}^d$, we call *raw score* of $\boldsymbol{x}$ through $\mathcal{C}$
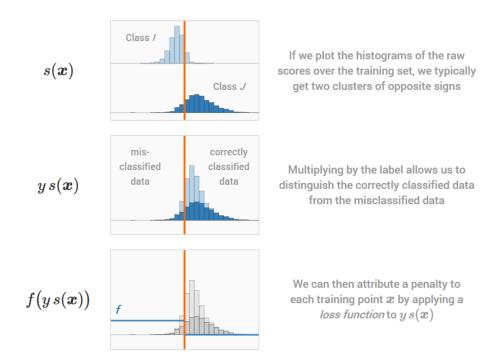
the value:

$$s(\boldsymbol{x}) := \boldsymbol{w} \cdot \boldsymbol{x} + b$$

The raw score can be seen as a *signed distance* between $\boldsymbol{x}$ and the classification boundary defined by $\mathcal{C}$. In particular:

$$\boldsymbol{x} \text{ is classified in } \left| \begin{array}{l} I \text{ if } s(\boldsymbol{x}) \leq 0 \\ J \text{ if } s(\boldsymbol{x}) \geq 0 \end{array} \right.$$

Now, consider a training set T of n pairs $(\boldsymbol{x}, y)$ where $\boldsymbol{x}$ is an image and $y = \{-1 \text{ if } \boldsymbol{x} \in I \mid 1 \text{ if } \boldsymbol{x} \in J\}$ is its label. We are interested in the distributions of the following quantities over T:

This leads to the notion of *empirical risk* $R(\boldsymbol{w}, b)$ for the classifier $\mathcal{C}$ defined as the average penalty over the training set T:

$$R(\boldsymbol{w}, b) := \frac{1}{n} \sum_{(\boldsymbol{x},y) \in T} f\big(y\, s(\boldsymbol{x})\big)$$

In general, learning a linear classifier consists of finding a weight vector $\boldsymbol{w}$ and a bias b minimising $R(\boldsymbol{w}, b)$ for a well chosen loss function f.

In binary classification, three notable loss functions are:
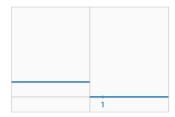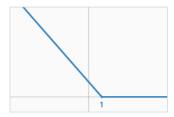
The 0-1 indicator function

$$f : z \to \begin{cases} 1 & \text{if } z \leq 0 \\ 0 & \text{if } z > 0 \end{cases}$$

The hinge loss

$$f : z \to \max(1 - z, 0)$$

The softplus loss

$$f : z \to \ln(1 + e^{-z})$$

With the 0-1 indicator function, the empirical risk is simply the *error rate* on T. In a sense, this is the optimal loss function as minimizing the error rate is often the desired objective in practice. Unfortunately, it is incompatible with gradient descent (there is no gradient to descend: the derivative is null everywhere).

This limitation is overcome in the hinge loss (used in SVM) and the softplus loss (used in logistic regression) by replacing the unit penalty on the misclassified data with a strictly decreasing penalty. Note that both the hinge loss and the softplus loss also penalize some correctly classified data in the neighbourhood of the boundary, effectively enforcing a safety margin.

The scaling parameter $\|w\|$

An important point previously overlooked is that the signed distance $s(x)$ is scaled by the norm of the weight vector. If $d(x)$ is the actual *signed Euclidean distance* between x and $\mathcal{C}$, we have:

$$d(x) := \hat{w} \cdot x + b' \qquad \text{where} \qquad \hat{w} := \frac{w}{\|w\|} \qquad b' := \frac{b}{\|w\|}$$

$$\text{and} \quad s(x) = \|w\| \, d(x)$$
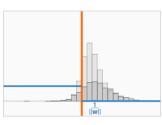
Hence, the norm $\|w\|$ can be interpreted as a scaling parameter for the loss function in the expression of the empirical risk:

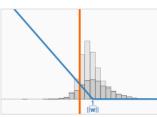$$R(w, b) = \frac{1}{n} \sum_{(x,y) \in T} f\big(\underbrace{\|w\|}_{\text{scaling parameter for } f} \times y \, d(x)\big)$$

Let us define the *scaled loss function* $f_{\|w\|} : z \to f(\|w\| \times z)$.

8

We observe that the 0-1 indicator function is invariant to rescaling while the hinge loss and the softplus loss are strongly affected.
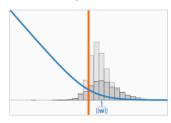
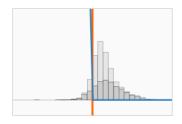| 0-1 indicator function	hinge loss	softplus loss

$\|\boldsymbol{w}\| = 10$ 0.00

Influence of the scaling parameter $\|\boldsymbol{w}\|$. The histograms represent the values of $y\,d(\boldsymbol{x})$ over T for a given $\boldsymbol{w}$ and b and the blue lines represent the values of $f_{\|\boldsymbol{w}\|}$.

Remarkably, the hinge loss and the softplus loss behave in the same way for extreme values of the scaling parameter.

When $\|\boldsymbol{w}\|$ is large

The hinge loss and the softplus loss penalize only the misclassified data—and do so linearly.

More precisely, both losses satisfy:[2]

$$f_{\|\boldsymbol{w}\|}\big(y\,d(\boldsymbol{x})\big) \underset{\|\boldsymbol{w}\|\to+\infty}{\approx} \|\boldsymbol{w}\|\,\max\big(-y\,d(\boldsymbol{x}),0\big)$$

For convenience, we name the set of misclassified data:

$$M := \{(\boldsymbol{x}, y) \in T \mid y\,d(\boldsymbol{x}) \le 0\}$$

and we can then write the empirical risk as:

$$R(\boldsymbol{w}, b) \underset{\|\boldsymbol{w}\|\to+\infty}{\approx} \|\boldsymbol{w}\| \left(\frac{1}{n} \sum_{(\boldsymbol{x}, y)\in M} (-y\,d(\boldsymbol{x})) \right)$$

---

[2] Hinge loss

$$\max(1 - \|\boldsymbol{w}\|\,y\,d(\boldsymbol{x}), 0) \quad = \quad \|\boldsymbol{w}\|\,\max\big(\|\boldsymbol{w}\|^{-1} - y\,d(\boldsymbol{x}), 0\big)$$
$$\underset{\|\boldsymbol{w}\|\to+\infty}{\approx} \|\boldsymbol{w}\|\,\max\big(-y\,d(\boldsymbol{x}), 0\big)$$

Softplus loss

$$\ln\big(1 + e^{-\|\boldsymbol{w}\|\,y\,d(\boldsymbol{x})}\big) \underset{\|\boldsymbol{w}\|\to+\infty}{\approx} \begin{cases} -\|\boldsymbol{w}\|\,y\,d(\boldsymbol{x}) & \text{if } y\,d(\boldsymbol{x}) \le 0 \\ 0 & \text{if } y\,d(\boldsymbol{x}) > 0 \end{cases}$$
$$\underset{\|\boldsymbol{w}\|\to+\infty}{\approx} \|\boldsymbol{w}\|\,\max\big(-y\,d(\boldsymbol{x}), 0\big)$$

This expression contains a term which we call the *error distance*:

$$d_{\text{err}} := \frac{1}{n} \sum_{(\boldsymbol{x},y) \in M} \big(-y\,d(\boldsymbol{x}) \big)$$

It is *positive* and can be interpreted as the average distance by which each training sample is misclassified by $\mathcal{C}$ (with a null contribution for the correctly classified data). It is related—although not exactly equivalent—to the training error.[3]
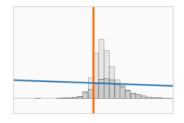
Finally we have:

$$\text{minimize: } R(\boldsymbol{w},b) \underset{\|\boldsymbol{w}\| \to +\infty}{\iff} \text{minimize: } d_{\text{err}}$$

In words, when $\|\boldsymbol{w}\|$ is large, *minimizing the empirical risk for the hinge loss or the softplus loss is equivalent to minimizing the error distance, which is similar to minimizing the error rate on the training set.*

When $\|\boldsymbol{w}\|$ is small

The hinge loss and the softplus loss penalize the entire training data linearly.

More precisely, both losses satisfy:[4]

$$f_{\|\boldsymbol{w}\|}\big(y\,d(\boldsymbol{x})\big) \underset{\|\boldsymbol{w}\| \to 0}{\approx} \alpha - \beta\,\|\boldsymbol{w}\|\,y\,d(\boldsymbol{x})$$

for some positive values α and β.

We can then write the empirical risk as:

$$R(\boldsymbol{w},b) \underset{\|\boldsymbol{w}\| \to 0}{\approx} \alpha - \beta\,\|\boldsymbol{w}\| \left(\frac{1}{n} \sum_{(\boldsymbol{x},y) \in T} y\,d(\boldsymbol{x}) \right)$$

---

[3]A small error distance d_{err} does not *guarantee* the training error err_{train} to be small. In the worst case, when all the data lies on the boundary, $d_{\text{err}} = 0$ and $err_{\text{train}} = 100\%$.

[4] Hinge loss

$$\max(1 - \|\boldsymbol{w}\|\,y\,d(\boldsymbol{x}), 0) \underset{\|\boldsymbol{w}\| \to 0}{=} 1 - \|\boldsymbol{w}\|\,y\,d(\boldsymbol{x})$$

$$\alpha = 1 \quad \text{and} \quad \beta = 1$$

Softplus loss

$$\ln\big(1 + e^{-\|\boldsymbol{w}\|\,y\,d(\boldsymbol{x})}\big) \underset{\|\boldsymbol{w}\| \to 0}{\approx} \ln(2) - \frac{1}{2}\,\|\boldsymbol{w}\|\,y\,d(\boldsymbol{x})$$

$$\alpha = \ln(2) \quad \text{and} \quad \beta = \frac{1}{2}$$

This expression contains a term which we call the *adversarial distance*:

$$d_{\mathrm{adv}} := \frac{1}{n} \sum_{(\boldsymbol{x},y)\in T} y\, d(\boldsymbol{x})$$

It is the mean distance between the images in T and the classification boundary $\mathcal{C}$ (with a negative contribution for the misclassified images). It can be viewed as a measure of robustness to adversarial perturbations: when d_{adv} is high, the number of misclassified images is limited and the correctly classified images are far from $\mathcal{C}$.

Finally we have:

$$\text{minimize: } R(\boldsymbol{w},b) \underset{\|\boldsymbol{w}\|\to 0}{\Longleftrightarrow} \text{ maximize: } d_{\mathrm{adv}}$$

In words, when $\|\boldsymbol{w}\|$ is small, *minimizing the empirical risk for the hinge loss or the softplus loss is equivalent to maximizing the adversarial distance, which can be interpreted as minimizing the phenomenon of adversarial examples.*

Closing remarks

In practice, the value of $\|\boldsymbol{w}\|$ can be controlled by adding a regularization term to the empirical risk, yielding the *regularized loss*:

$$R(\boldsymbol{w},b) = \frac{1}{n} \sum_{(\boldsymbol{x},y)\in T} f\big(\underbrace{\|\boldsymbol{w}\|}_{\text{scaling parameter for } f} \times\, y\, d(\boldsymbol{x})\big)$$

A small *regularization parameter* λ lets $\|\boldsymbol{w}\|$ grow unchecked while a larger λ encourages $\|\boldsymbol{w}\|$ to shrink.

> In summary, the two standard models used in linear classification (SVM and logistic regression) balance between two objectives: they *minimize the error distance* when regularization is low and they *maximize the adversarial distance* when regularization is high.

Adversarial Distance and Tilting Angle

The adversarial distance emerged in the previous section as a measure of robustness to adversarial perturbations. Rather conveniently, it can be expressed as a function of a single parameter: the angle between the classification boundary and the nearest centroid classifier.

If T_I and T_J are the restrictions of T to the elements in I and J respectively, we can write:

$$d_{\text{adv}} = \frac{1}{n} \sum_{(\boldsymbol{x},y) \in T} y\, d(\boldsymbol{x})$$

$$= \frac{1}{n} \left[\sum_{\boldsymbol{x} \in T_I} (-\hat{\boldsymbol{w}} \cdot \boldsymbol{x} - b') \;+\; \sum_{\boldsymbol{x} \in T_J} (\hat{\boldsymbol{w}} \cdot \boldsymbol{x} + b') \right]$$

If T_I and T_J are balanced ($n = 2\, n_I = 2\, n_J$):

$$d_{\text{adv}} = -\frac{1}{2\, n_I} \sum_{\boldsymbol{x} \in T_I} \hat{\boldsymbol{w}} \cdot \boldsymbol{x} \;+\; \frac{1}{2\, n_J} \sum_{\boldsymbol{x} \in T_J} \hat{\boldsymbol{w}} \cdot \boldsymbol{x}$$

$$= \frac{1}{2}\, \hat{\boldsymbol{w}} \cdot \left[\left(\frac{1}{n_J} \sum_{\boldsymbol{x} \in T_J} \boldsymbol{x} \right) - \left(\frac{1}{n_I} \sum_{\boldsymbol{x} \in T_I} \boldsymbol{x} \right) \right]$$

If $\boldsymbol{i}$ and $\boldsymbol{j}$ are the centroids of T_I and T_J respectively:

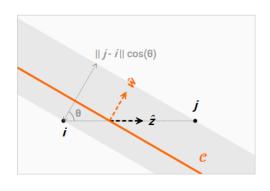$$d_{\text{adv}} = \frac{1}{2}\, \hat{\boldsymbol{w}} \cdot (\boldsymbol{j} - \boldsymbol{i})$$

We now introduce the *nearest centroid classifier*, which has unit normal vector $\hat{\boldsymbol{z}} = (\boldsymbol{j} - \boldsymbol{i})/\|\boldsymbol{j} - \boldsymbol{i}\|$:

$$d_{\text{adv}} = \frac{1}{2}\, \|\boldsymbol{j} - \boldsymbol{i}\|\, \hat{\boldsymbol{w}} \cdot \hat{\boldsymbol{z}}$$

Finally, we call the plane containing $\hat{\boldsymbol{w}}$ and $\hat{\boldsymbol{z}}$ the *tilting plane* of $\mathcal{C}$ and we we call the angle θ between $\hat{\boldsymbol{w}}$ and $\hat{\boldsymbol{z}}$ the *tilting angle* of $\mathcal{C}$:

$$d_{\text{adv}} = \frac{1}{2}\, \|\boldsymbol{j} - \boldsymbol{i}\|\, \cos(\theta)$$

This equation can be interpreted geometrically in the tilting plane:

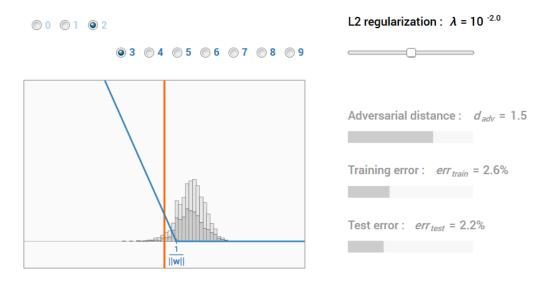d_{adv} is the average distance between the classification boundary $\mathcal{C}$ and the two centroids $\boldsymbol{i}$ and $\boldsymbol{j}$

On a given training set T where the distance between the two centroids $\|\boldsymbol{j} - \boldsymbol{i}\|$ is fixed, d_{adv} depends only on the tilting angle θ. Two observations follow:

- The adversarial example phenomenon is minimized by the nearest centroid classifier ($\theta = 0$).[5]

- Adversarial examples can be arbitrarily strong when $\theta \to \pi/2$ (as was the case with the classifier $\mathscr{L}_\theta$ in the toy problem section).

Example: SVM on MNIST

We now illustrate the previous considerations on binary classification of MNIST digits. For each possible pair of digit classes, we train multiple SVM models $(\boldsymbol{w}, b)$ for a regularization parameter $\lambda \in [10^{-1}, 10^7]$ using a training set of 3000 images per class.[6]

We start by plotting the distributions of the distances $y\,d(\boldsymbol{x})$ between the training data and the boundary as a function of the regularization parameter λ (grey histograms). We superimpose the loss function $f_{\|\boldsymbol{w}\|}$ as scaled after the convergence of each model (blue line).

○ 0 ○ 1 ● 2
● 3 ○ 4 ○ 5 ○ 6 ○ 7 ○ 8 ○ 9

L2 regularization : $\lambda = 10^{-2.0}$

Adversarial distance : $d_{adv} = 1.5$

Training error : $err_{train} = 2.6\%$

Test error : $err_{test} = 2.2\%$

---

[5]...and the classifiers parallel to it: d_{adv} is independent of the bias b. This explains why the models defined by SVM are poorly adjusted when regularization is high, resulting in very high training and test errors (see for instance the classification of 1s vs 8s when $\lambda = 10^7$ in the next section).

[6]More precisely, we train for each pair of digit classes 81 models with a regularization parameter $\lambda = 10^\alpha$ with α ranging from -1 to 7 by steps of 0.1.

We see that the scaling of the hinge loss has a clear influence on the model obtained. Unfortunately, minimising the training error and maximizing the adversarial distance are conflicting goals: err_{train} is minimized when λ is small and d_{adv} is maximized when λ is large. Note that the test error is minimized for an intermediate level of regularization λ_{optimal}. When $\lambda < \lambda_{\text{optimal}}$, the classifier is *overfitted* and when $\lambda > \lambda_{\text{optimal}}$, the classifier is *underfitted*.

To get a better understanding of how the two objectives are balanced, we can look at the training data under a different point of view. We first compute the unit weight vector $\hat{\boldsymbol{z}}$ of the nearest centroid classifier. Then for each SVM model $(\boldsymbol{w}, b)$, we compute the unit vector $\hat{\boldsymbol{n}}$ such that $(\hat{\boldsymbol{z}}, \hat{\boldsymbol{n}})$ is an orthonormal basis of the tilting plane of $\boldsymbol{w}$.[7] Finally, we project the training data in $(\hat{\boldsymbol{z}}, \hat{\boldsymbol{n}})$:
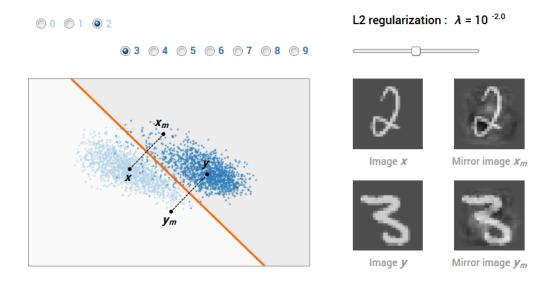

The horizontal direction passes through the two centroids and the vertical direction is chosen such that $\boldsymbol{w}$ belongs to the plane (the hyperplane boundary then appears as a line). Remark also that since $(\hat{\boldsymbol{z}}, \hat{\boldsymbol{n}})$ is an orthonormal basis, the distances in this plane are actual pixel distances. To understand why the data points appear to be moving around when λ varies, one needs to imagine the tilting plane rotating around $\hat{\boldsymbol{z}}$ in the 784-dimensional input space (thus showing a different section of the 784-dimensional training data for each value of λ).

---

[7]We do this by using the Gram-Schmidt process:

$$\hat{\boldsymbol{n}} = \frac{\boldsymbol{n}}{\|\boldsymbol{n}\|} \qquad \text{with} \qquad \boldsymbol{n} = \boldsymbol{w} - (\boldsymbol{w} \cdot \hat{\boldsymbol{z}}) \, \hat{\boldsymbol{z}}$$

For high regularization levels, the model is parallel to the nearest centroid classifier and the adversarial distance is maximized. As λ decreases, the classification boundary improves its fit of the training data by tilting towards directions of low variance. Eventually, a small number of misclassified training samples is overfitted, resulting in a very small adversarial distance and a weight vector that is hard to interpret.

Finally, we can look at two representative images $\boldsymbol{x}$, $\boldsymbol{y}$ (one per class) and their mirror images $\boldsymbol{x}_m$, $\boldsymbol{y}_m$ for each model. Their projections in the tilting plane of $\boldsymbol{w}$ give a very intuitive picture of the adversarial example phenomenon in linear classification:

The model is sensitive to strong adversarial examples ($||\boldsymbol{x}_m - \boldsymbol{x}|| \to 0$ and $||\boldsymbol{y}_m - \boldsymbol{y}|| \to 0$) when the tilting angle approaches $\pi/2$. This is a symptom of strong overfitting, and whether it occurs or not depends on the difficulty to separate the two classes (compare for instance the classification of $7s$ versus $9s$ and the classification of $0s$ versus $1s$).

Adversarial Examples in Neural Networks

Thanks to the equivalence between adversarial distance and tilting angle, the linear case is simple enough to be visualized in the plane. In neural networks however, the class boundary is not flat and the adversarial distance cannot be reduced to a single parameter. Nonetheless, some similarities with the linear case remain.

First Step: a 2-Layer Binary Network

Let $\mathcal{N}$ be a 2-layer network with a single output defining a non-linear binary classifier in $\mathbb{R}^d$. The first layer of $\mathcal{N}$ is specified by a weight matrix W_1 and a bias vector b_1 and the second layer of $\mathcal{N}$ is specified by a weight vector W_2 and bias b_2. We assume that the two layers are separated by a layer ϕ of rectified linear units applying the function $z \rightarrow \max(0, z)$ element-wise. For an image x in $\mathbb{R}^d$, we call the *raw score* of x through $\mathcal{N}$ the value:

$$s(x) := W_2 \, \phi(W_1 \, x + b_1) + b_2$$

Similarly to the linear case, the *empirical risk* on T for a *loss function f* can be written:

$$R(W_1, b_1; W_2, b_2) := \frac{1}{n} \sum_{(x,y) \in T} f\big(y \, s(x)\big)$$

and training $\mathcal{N}$ consists in finding W_1, b_1, W_2 and b_2 minimizing R for a well chosen f.

ϕ is piecewise linear and around each image x there is a local linear region $\mathcal{L}_x$ within which:

$$\phi(W_1 \, x + b_1) = W_1^x \, x + b_1^x$$

where W_1^x and b_1^x are obtained by zeroing some lines in W_1 and b_1 respectively.[8] Within $\mathcal{L}_x$, the raw score can thus be written:

$$s(x) = W_2 W_1^x \, x + W_2 b_1^x + b_2$$

This can be seen as the raw score of a local linear classifier $\mathcal{C}_x$ and our analysis of the linear case then applies almost without modifications. First, we observe that $s(x)$ is a scaled distance. If $d(x)$ is the actual *signed Euclidean distance* between x and $\mathcal{C}_x$, we have:

$$s(x) = \|W_2 W_1^x\| \, d(x)$$

The norm $\|W_2 W_1^x\|$ can then be interpreted as a scaling parameter for the loss function (the scaling is now local, dependent on x). One simple way to control all the local scalings simultaneously is by adding an L2 regularization term to the empirical risk acting on the norms $\|W_1\|$ and

---

[8]More precisely, the ith lines in W_1^x and b_1^x are:

$$(W_1^x \, , \, b_1^x)_i = \begin{cases} (W_1 \, , \, b_1)_i & \text{if } (W_1 \, x + b_1)_i > 0 \\ 0 & \text{if } (W_1 \, x + b_1)_i \le 0 \end{cases}$$

Notes ▸ $d(x)$ can also be viewed as a linear approximation of the distance between x and the boundary defined by $\mathcal{N}$ (the distance to the nearest adversarial example).

▸ $W_2 W_1^x$ is the gradient of $\mathcal{N}$ within $\mathcal{L}_x$. It is the adversarial direction for x, computed by backpropagation in practice.

$\|W_2\|$ independently (remember that the weights in W_1^x are a subset of the weights in W_1). With gradient descent, this is equivalent to decaying the weights W_1 and W_2 at every iteration. More precisely, for a learning rate η and a decaying factor λ, the *weight decay update* is:

$$W_1 \leftarrow W_1 - \eta\,\lambda\,W_1 \quad \text{and} \quad W_2 \leftarrow W_2 - \eta\,\lambda\,W_2$$

- **With a small decaying factor λ,** the scaling parameter $\|W_2 W_1^x\|$ is allowed to grow unrestricted and the loss penalizes only the misclassified data. Minimizing the empirical risk is then equivalent to minimizing the error on the training set.

- **As the decaying factor λ increases,** the scaling parameter $\|W_2 W_1^x\|$ decreases and the loss starts penalizing more and more of the correctly classified data, pushing it further away from the boundary. Under this light, L2 weight decay can be seen as a form of *adversarial training*.

> In summary, L2 regularization acts as a scaling mechanism on the loss function, both in linear classification and in small neural nets. With gradient descent, using a ***high weight decay*** results in a simple form of ***adversarial training***.

Second Step: General Case

The previous analysis can be generalized to more layers and even to non-piecewise-linear activation functions. The important observation is that we always have:

$$s(x) = \|\nabla_x s\|\, d(x)$$

Where $\nabla_x s$ is the gradient of the raw score on x, and $d(x)$ is a linear approximation of the distance between x and the boundary defined by the network. The norm $\|\nabla_x s\|$ then constitutes a scaling parameter for the loss function which can be controlled with weight decay.
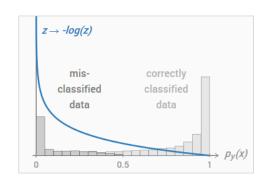
This idea can also be extended beyond binary classification. In the multiclass case, the raw score becomes a vector whose elements are called

the *logits*. Each logit $s_i(x)$ is then transformed into a *probability* $p_i(x)$ by applying the softmax function:

$$p_i(x) := \text{softmax}_i(s(x)) := \frac{e^{s_i(x)}}{\sum_j e^{s_j(x)}}$$

For an image/label pair (x, y), the probability associated with the correct class is $p_y(x)$. The *log-likelihood* loss function encourages it to come close to 1 by attributing the following penalty to (x, y):

$$f(x, y) := -\log(p_y(x))$$

Log-likelihood loss function

Now, varying weight decay influences the scaling of the logits, effectively acting as a *temperature* parameter for the softmax function.[9] When weight decay is very low, the probability distributions generated are close to one-hot encodings ($p_y(x) \approx 0$ or 1) and only the misclassified data produces non-zero penalties. With higher weight decay however, the probability distributions generated become smoother and the correctly classified data participates to the training, thus preventing overfitting.

In practice, a number of observations suggest that modern deep networks are under-regularized:

1. They are often poorly calibrated and produce overconfident predictions [29].

2. They often converge to zero training error, even on a random labelling of the data [30].

3. They are often vulnerable to linear attacks of small magnitude [2].

---

[9]We can redefine the softmax function as:

$$\text{softmax}_i(s(x), t) := \frac{e^{\frac{s_i(x)}{t}}}{\sum_j e^{\frac{s_j(x)}{t}}}$$
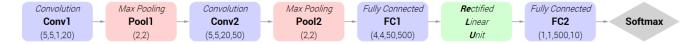
where t is the temperature parameter. It was introduced in the context of network distillation [28] and controls the softness of the softmax function.
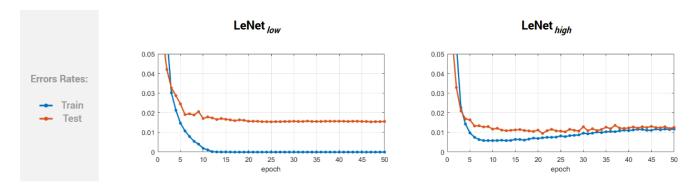
Example: LeNet on MNIST

Is it possible to regularize a neural network against adversarial examples by only using weight decay? The idea is simple enough and has been considered before: Goodfellow et al. [2] have observed that adversarial training is *"somewhat similar to L1 regularization"* in the linear case. However, the authors reported that when training maxout networks on MNIST, an L1 weight decay coefficient of 0.0025 *"was too large, and caused the model to get stuck with over 5% error on the training set. Smaller weight decay coefficients permitted successful training but conferred no regularization benefit."* We put the idea to the test once again and our observations are more nuanced. If using a high weight decay is clearly not the panacea, we found that it does help reduce the adversarial examples phenomenon, at least in simple setups.

Consider LeNet on MNIST (10-class problem). We use the baseline MatConvNet [31] implementation with the following architecture:
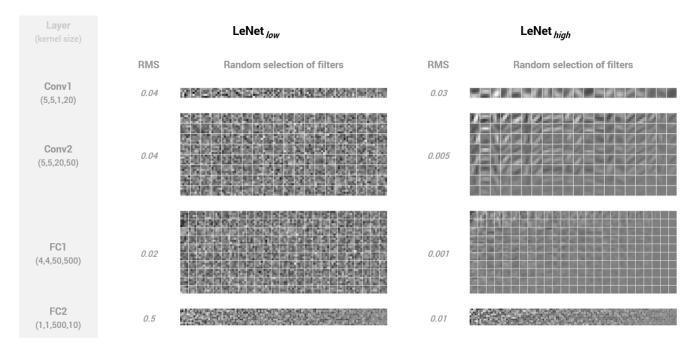

We train one version of this network with a *low* weight decay of 10^{-4} and one version with a *high* weight decay of 10^{-1} (we refer to the two versions as LeNet$_{low}$ and LeNet$_{high}$ respectively). We keep all the other parameters fixed: we train for 50 epochs, use a batch size of 300, a learning rate of 0.0005 and a momentum of 0.9.

We can make several observations. To start, let's plot the training and test errors for the two networks as a function of the epoch.

We see that LeNet$_{high}$ is less overfitted (the train and test errors are approximately equal at the end of the training) and performs slightly better than LeNet$_{low}$ (final test error of 1.2% versus 1.6%).

19

We can also inspect the weights that have been learnt. Below, we compute their root mean square value (RMS) and show a random selection of filters for each convolutional layer.
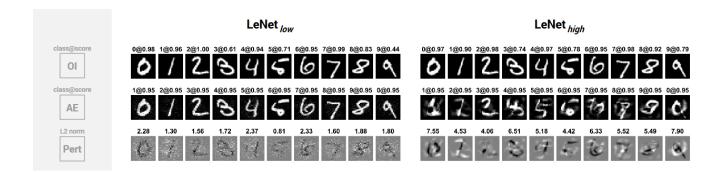
Layer (kernel size)	LeNet$_{low}$		LeNet$_{high}$	
	RMS	Random selection of filters	RMS	Random selection of filters
Conv1 (5,5,1,20)	0.04		0.03	
Conv2 (5,5,20,50)	0.04		0.005	
FC1 (4,4,50,500)	0.02		0.001	
FC2 (1,1,500,10)	0.5		0.01	

As expected, the weights learnt with a higher weight decay have a much lower RMS. The filters of LeNet$_{high}$ are also smoother than the filters of LeNet$_{low}$ (see the presence of clean edge detectors in Conv1 and Conv2) and their magnitudes vary more within each convolutional layer (see the presence of uniformly gray filters in Conv2 and FC1).

Finally, let's submit the two networks to the same visual evaluation: for a random instance of each digit, we generate a high confidence adversarial example targeted to perform a cyclic permutation of the labels $0 \to 1$, $1 \to 2$, ..., $9 \to 0$. Specifically, each adversarial example is generated by performing gradient ascent on the probability of the desired label until the median value of 0.95 is reached.[10] We show below the 10 original images OI with their corresponding adversarial examples AE and adversarial perturbations Pert for the two networks.

---

[10]On each network, we set the temperature t of the softmax layer such that the median classification score over the test set is 0.95. For a target label l, the gradient $\nabla_x \, p_l(x)$ (computed by backpropagation) gives the direction of steepest ascent towards l. Here, we generated each adversarial example x' by iterating the following update rule:

$$x' = \text{clip}_{[0,255]} \left(x' + 1.0 \times \frac{\nabla_{x'} \, p_l(x')}{\|\nabla_{x'} \, p_l(x')\|} \right)$$

until $p_l(x') = 0.95$.

We see that LeNet$_{high}$ is less susceptible to adversarial examples than LeNet$_{low}$: the adversarial perturbations have higher L2 norms and are more meaningful for human observers.

Thoughts Moving Forward

Despite the widespread interest it has generated for several years now, and despite its significance for the field of machine learning both in theory and in practice, the adversarial example phenomenon has so far retained much of its intrigue. Our main goal here was to provide a clear and intuitive picture of the phenomenon in the linear case, hopefully constituting a solid base from which to move forward. Incidentally, we showed that L2 weight decay plays a more significant role than previously suspected in a small neural net on MNIST.

Unfortunately, the story gets more complicated with deeper models on more sophisticated datasets. In our experience, the more non-linear the model becomes and the less weight decay seems to be able to help. This limitation may be superficial and it is perhaps worth exploring the ideas introduced here a bit further (for example, we should probably pay more attention to the scaling of the logits during training). Or the high non-linearity of deep networks might constitute a fundamental obstacle to the type of first-order adversarial training that L2 regularization implements. Our feeling is that a truly satisfying solution to the problem will likely require profoundly new ideas in deep learning.

References

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

[2] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[3] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, et al. Deep face recognition. In *BMVC*, volume 1, page 6, 2015.

[4] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against deep learning systems using adversarial examples. *arXiv preprint arXiv:1602.02697*, 2016.

[5] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

[6] Ivan Evtimov, Kevin Eykholt, Earlence Fernandes, Tadayoshi Kohno, Bo Li, Atul Prakash, Amir Rahmati, and Dawn Song. Robust physical-world attacks on machine learning models. *arXiv preprint arXiv:1707.08945*, 2017.

[7] Anish Athalye and Ilya Sutskever. Synthesizing robust adversarial examples. *arXiv preprint arXiv:1707.07397*, 2017.

[8] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2574–2582, 2016.

[9] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. *arXiv preprint arXiv:1608.04644*, 2016.

[10] Osbert Bastani, Yani Ioannou, Leonidas Lampropoulos, Dimitrios Vytiniotis, Aditya Nori, and Antonio Criminisi. Measuring neural net robustness with constraints. In *Advances in Neural Information Processing Systems*, pages 2613–2621, 2016.

[11] Alhussein Fawzi, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. Robustness of classifiers: from adversarial to random noise. In *Advances in Neural Information Processing Systems*, pages 1632–1640, 2016.

[12] Nicholas Carlini, Guy Katz, Clark Barrett, and David L Dill. Ground-truth adversarial examples. *arXiv preprint arXiv:1709.10207*, 2017.

[13] Shixiang Gu and Luca Rigazio. Towards deep neural network architectures robust to adversarial examples. *arXiv preprint arXiv:1412.5068*, 2014.

[14] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *Security and Privacy (SP), 2016 IEEE Symposium on*, pages 582–597. IEEE, 2016.

[15] Qiyang Zhao and Lewis D Griffin. Suppressing the unusual: towards robust cnns using symmetric activation functions. *arXiv preprint arXiv:1603.05145*, 2016.

[16] Andras Rozsa, Manuel Gunther, and Terrance E Boult. Towards robust deep neural networks with bang. *arXiv preprint arXiv:1612.00138*, 2016.

[17] Arjun Nitin Bhagoji, Daniel Cullina, and Prateek Mittal. Dimensionality reduction as a defense against evasion attacks on machine learning classifiers. *arXiv preprint arXiv:1704.02654*, 2017.

[18] Reuben Feinman, Ryan R Curtin, Saurabh Shintre, and Andrew B Gardner. Detecting adversarial samples from artifacts. *arXiv preprint arXiv:1703.00410*, 2017.

[19] Kathrin Grosse, Praveen Manoharan, Nicolas Papernot, Michael Backes, and Patrick McDaniel. On the (statistical) detection of adversarial examples. *arXiv preprint arXiv:1702.06280*, 2017.

[20] Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. On detecting adversarial perturbations. *arXiv preprint arXiv:1702.04267*, 2017.

[21] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*, 2016.

[22] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*, 2017.

[23] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[24] Ian Goodfellow, Nicolas Papernot, Sandy Huang, Yan Duan, Pieter Abbeel, and Jack Clark. Attacking machine learning with adversarial examples, 2017.

[25] Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. *arXiv preprint arXiv:1705.07263*, 2017.

[26] James Stephen Marron, Michael J Todd, and Jeongyoun Ahn. Distance-weighted discrimination. *Journal of the American Statistical Association*, 102(480):1267–1271, 2007.

[27] Huan Xu, Constantine Caramanis, and Shie Mannor. Robustness and regularization of support vector machines. *Journal of Machine Learning Research*, 10(Jul):1485–1510, 2009.

[28] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[29] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. *arXiv preprint arXiv:1706.04599*, 2017.

[30] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

[31] Andrea Vedaldi and Karel Lenc. Matconvnet: Convolutional neural networks for matlab. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 689–692. ACM, 2015.