

# Neuronale Netze und adversarial attacks: Wie der Panda zum Gibbon wurde

Data Science in Forschung und  
Industrie

Tim Roith<sup>1</sup>

<sup>1</sup>Helmholtz Imaging, Deutsches Elektronen-Synchrotron  
DESY

31. Oktober 2025

 HELMHOLTZ  
IMAGING

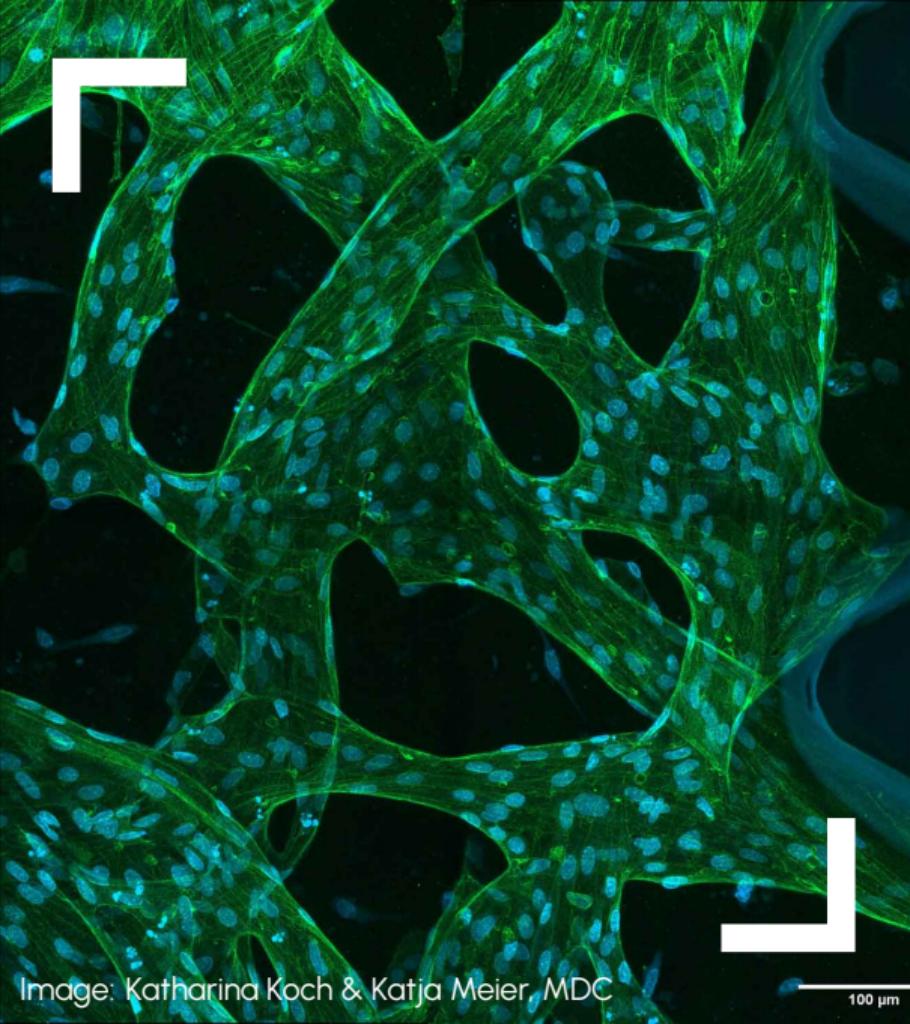


Image: Katharina Koch & Katja Meier, MDC

100 µm

# Klassifizierung mit neuronalen Netzten

# Bildklassifizierung: Was sind Bilder?



- Wir konzentrieren uns in diesem Vortrag auf Bildklassifizierung.

# Bildklassifizierung: Was sind Bilder?



- Wir konzentrieren uns in diesem Vortrag auf Bildklassifizierung.
- Viele der folgenden Konzepte lassen sich auf andere Aufgaben übertragen.

# Bildklassifizierung: Was sind Bilder?



- Wir konzentrieren uns in diesem Vortrag auf Bildklassifizierung.
- Viele der folgenden Konzepte lassen sich auf andere Aufgaben übertragen.
- Wir repräsentieren ein Bild als ein Element

$$x \in [0, 1]^{3 \times H \times W}$$

mit  $H, W \in \mathbb{N}$ .

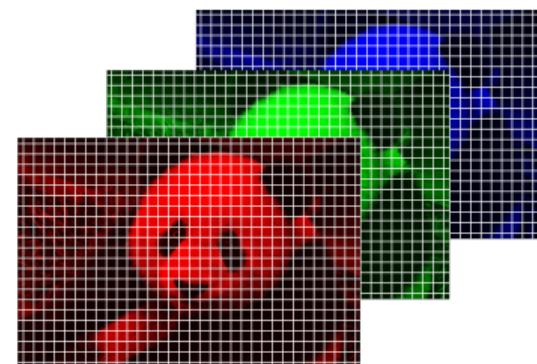
# Bildklassifizierung: Was sind Bilder?



- Wir konzentrieren uns in diesem Vortrag auf Bildklassifizierung.
- Viele der folgenden Konzepte lassen sich auf andere Aufgaben übertragen.
- Wir repräsentieren ein Bild als ein Element

$$x \in [0, 1]^{3 \times H \times W}$$

mit  $H, W \in \mathbb{N}$ .



# Bildklassifizierung: Was ist Klassifizierung?



- Wir betrachten eine Menge an möglichen Klassen

$$\mathcal{C} = \{\text{Panda, Hydrant, Spaghetti, ..., PKW}\}.$$

# Bildklassifizierung: Was ist Klassifizierung?



- Wir betrachten eine Menge an möglichen Klassen

$$\mathcal{C} = \{\text{Panda, Hydrant, Spaghetti, ..., PKW}\}.$$

- Wir möchten nun eine Abbildung  $f$  erhalten, die jedem Bild  $x$  eine Klasse zuordnet:

$$f \left( \begin{array}{c} \text{Image of a Panda} \end{array} \right) = \text{Panda}$$

# Bildklassifizierung: Was ist Klassifizierung?



- Wir betrachten eine Menge an möglichen Klassen

$$\mathcal{C} = \{\text{Panda, Hydrant, Spaghetti, ..., PKW}\}.$$

- Wir möchten nun eine Abbildung  $f$  erhalten, die jedem Bild  $x$  eine Klasse zuordnet:

$$f \left( \begin{array}{c} \text{Image of a Panda} \end{array} \right) = \text{Panda}$$

- Oftmals gibt  $f$  nicht direkt eine Klasse zurück, sondern einen Vektor  $y \in \mathbb{R}^C$

$$f(x) = (0.6, \dots, 0.3, \dots, 0.1).$$

Panda   Gibbon   PKW

# Bildklassifizierung: Wie gut ist der Klassifizierer?



- Wir befinden uns im „supervised“ Setting, d. h. wir haben einen Datensatz von Input-Output Daten gegeben,

$$\mathcal{T} = \{(x_1, \text{PKW}), (x_2, \text{Katze}), \dots, (x_N, \text{Panda})\}.$$

# Bildklassifizierung: Wie gut ist der Klassifizierer?



- Wir befinden uns im „supervised“ Setting, d. h. wir haben einen Datensatz von Input-Output Daten gegeben,

$$\mathcal{T} = \{(x_1, \text{PKW}), (x_2, \text{Katze}), \dots, (x_N, \text{Panda})\}.$$

- Wir können die Performanz des Klassifizierers  $f$  nun messen: für ein Datum  $(x, y) \in \mathcal{T}$  betrachten wir

$$\ell(f(x), y).$$

Typische Wahl für  $\ell(\cdot, \cdot)$ :  $\ell(y', y) = \|y' - y\|$ , oder  $\ell(\cdot, \cdot)$  = Kreuzentropie.

# Bildklassifizierung: Wie gut ist der Klassifizierer?



- Wir befinden uns im „supervised“ Setting, d. h. wir haben einen Datensatz von Input-Output Daten gegeben,

$$\mathcal{T} = \{(x_1, \text{PKW}), (x_2, \text{Katze}), \dots, (x_N, \text{Panda})\}.$$

- Wir können die Performanz des Klassifizierers  $f$  nun messen: für ein Datum  $(x, y) \in \mathcal{T}$  betrachten wir

$$\ell(f(x), y).$$

Typische Wahl für  $\ell(\cdot, \cdot)$ :  $\ell(y', y) = \|y' - y\|$ , oder  $\ell(\cdot, \cdot)$  = Kreuzentropie.

- Die Performanz auf allen Daten ist dann gegeben durch

$$\sum_{(x,y) \in \mathcal{T}} \ell(f(x), y).$$

# Bildklassifizierung: Wie lernt man den Klassifizierer?



- Um einen Klassifizierer modifizieren zu können, betrachten wir Parameter  $\theta \in \Theta$  und eine Parametrisierung  $f_\theta$ .

# Bildklassifizierung: Wie lernt man den Klassifizierer?



- Um einen Klassifizierer modifizieren zu können, betrachten wir Parameter  $\theta \in \Theta$  und eine Parametrisierung  $f_\theta$ .
- Beispiel lineare Funktion:  $f_\theta(x) = w \cdot x + b$  mit Parametern  $\theta = (w, b)$ .

# Bildklassifizierung: Wie lernt man den Klassifizierer?



- Um einen Klassifizierer modifizieren zu können, betrachten wir Parameter  $\theta \in \Theta$  und eine Parametrisierung  $f_\theta$ .
- Beispiel lineare Funktion:  $f_\theta(x) = w \cdot x + b$  mit Parametern  $\theta = (w, b)$ .
- Die Funktion zu lernen, bedeutet Parameter  $\theta$  zu finden, s.d. die **Loss**-Funktion

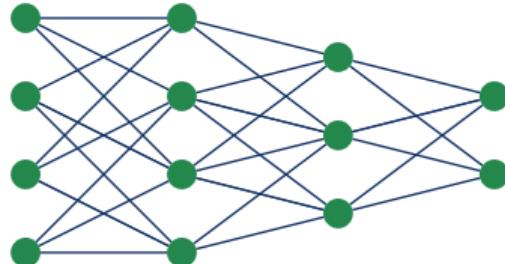
$$\mathcal{L}(\theta) = \sum_{(x,y) \in \mathcal{T}} \ell(f_\theta(x), y)$$

minimiert wird.

# Parametrisierung durch neuronale Netze



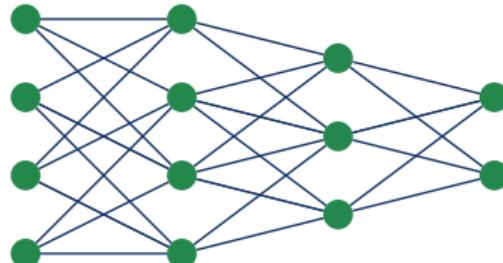
- Wir parametrisieren den Klassifizierer im Folgenden als **neuronales Netz**.



# Parametrisierung durch neuronale Netze



- Wir parametrisieren den Klassifizierer im Folgenden als **neuronales Netz**.



- Ein neuronales Netz besteht aus Layern der Form

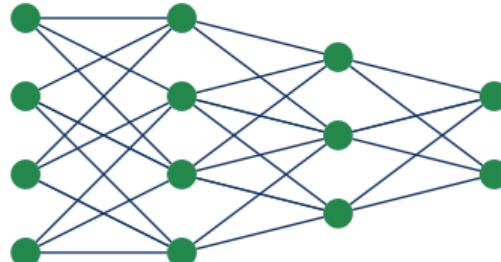
$$\Phi_i(z) := \sigma(W_i z + b_i),$$

$W_i$ : Gewichtsmatrix,  $b_i$ : Bias,  $\sigma$ : Aktivierungsfunktion.

# Parametrisierung durch neuronale Netze



- Wir parametrisieren den Klassifizierer im Folgenden als **neuronales Netz**.



- Ein neuronales Netz besteht aus Layern der Form

$$\Phi_i(z) := \sigma(W_i z + b_i),$$

$W_i$ : Gewichtsmatrix,  $b_i$ : Bias,  $\sigma$ : Aktivierungsfunktion.

- Das gesamte Netz entsteht durch das Zusammenstecken aller Layer

$$f_{\theta} = \Phi_L \circ \dots \circ \Phi_2 \circ \Phi_1$$

mit Parametern  $\theta = ((W_L, b_L), \dots, (W_2, b_2), (W_1, b_1))$ .

# Adversarial Attacks

# Wie gut funktionieren neuronale Netze?



- Positive Resultate (um 2015):

# Wie gut funktionieren neuronale Netze?



- Positive Resultate (um 2015):
  - He, Zhang, Ren und Sun: „*Delving deep into rectifiers: Surpassing human-level performance on imagenet classification*“

# Wie gut funktionieren neuronale Netze?



- Positive Resultate (um 2015):
  - He, Zhang, Ren und Sun: „*Delving deep into rectifiers: Surpassing human-level performance on imagenet classification*“
  - He, Zhang, Ren und Sun: „*Deep residual learning for image recognition*“  
(siehe auch Srivastava, Greff und Schmidhuber: „*Highway networks*“)

# Wie gut funktionieren neuronale Netze?



- Positive Resultate (um 2015):
  - He, Zhang, Ren und Sun: „*Delving deep into rectifiers: Surpassing human-level performance on imagenet classification*“
  - He, Zhang, Ren und Sun: „*Deep residual learning for image recognition*“  
(siehe auch Srivastava, Greff und Schmidhuber: „*Highway networks*“)
- Bedenkliche Resultate (etwas früher in 2014):

# Wie gut funktionieren neuronale Netze?



- Positive Resultate (um 2015):
  - He, Zhang, Ren und Sun: „*Delving deep into rectifiers: Surpassing human-level performance on imagenet classification*“
  - He, Zhang, Ren und Sun: „*Deep residual learning for image recognition*“  
(siehe auch Srivastava, Greff und Schmidhuber: „*Highway networks*“)
- Bedenkliche Resultate (etwas früher in 2014):
  - Szegedy u. a.: „*Intriguing properties of neural networks*“

# Wie gut funktionieren neuronale Netze?



- Positive Resultate (um 2015):
  - He, Zhang, Ren und Sun: „*Delving deep into rectifiers: Surpassing human-level performance on imagenet classification*“
  - He, Zhang, Ren und Sun: „*Deep residual learning for image recognition*“  
(siehe auch Srivastava, Greff und Schmidhuber: „*Highway networks*“)
- Bedenkliche Resultate (etwas früher in 2014):
  - Szegedy u. a.: „*Intriguing properties of neural networks*“
  - I. J. Goodfellow, Shlens und Szegedy: „*Explaining and harnessing adversarial examples*“

# Was sind adversarial attacks?



Das berühmteste Beispiel von [GSS14]:

# Was sind adversarial attacks?

Das berühmteste Beispiel von [GSS14]:



$x$

“panda”

57.7% confidence

# Was sind adversarial attacks?

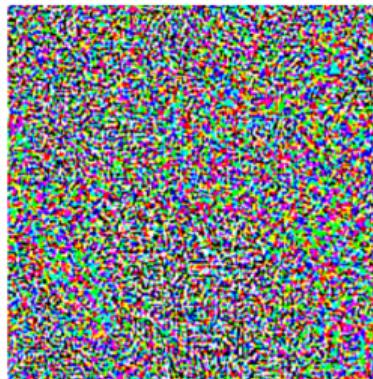
Das berühmteste Beispiel von [GSS14]:



$x$

“panda”  
57.7% confidence

$$+ .007 \times$$



$$\text{sign}(\nabla_x J(\theta, x, y))$$

“nematode”  
8.2% confidence

# Was sind adversarial attacks?



Das berühmteste Beispiel von [GSS14]:



$$+ .007 \times$$



=



$x$

“panda”

57.7% confidence

$$\text{sign}(\nabla_x J(\theta, x, y))$$

“nematode”

8.2% confidence

$$\epsilon \text{sign}(\nabla_x J(\theta, x, y)) + x$$

“gibbon”

99.3 % confidence



# Was sind adversarial attacks?

- Gegeben sei ein Bild  $x$ , welches durch  $f_\theta$  „korrekt“ klassifiziert wird [Bun+23].



# Was sind adversarial attacks?

- Gegeben sei ein Bild  $x$ , welches durch  $f_\theta$  „korrekt“ klassifiziert wird [Bun+23].
- Ein (untargeted) adversarial example ist ein Bild  $x'$  „nahe“ bei  $x$ , s.d.  $f_\theta x'$  falsch klassifiziert.

# Was sind adversarial attacks?



- Gegeben sei ein Bild  $x$ , welches durch  $f_\theta$  „korrekt“ klassifiziert wird [Bun+23].
- Ein (untargeted) adversarial example ist ein Bild  $x'$  „nahe“ bei  $x$ , s.d.  $f_\theta x'$  falsch klassifiziert.
- Intuitive, bedeutet „nahe“

# Was sind adversarial attacks?



- Gegeben sei ein Bild  $x$ , welches durch  $f_\theta$  „korrekt“ klassifiziert wird [Bun+23].
- Ein (untargeted) adversarial example ist ein Bild  $x'$  „nahe“ bei  $x$ , s.d.  $f_\theta x'$  falsch klassifiziert.
- Intuitive, bedeutet „nahe“
  - $x$  sieht ähnlich aus wie  $x'$ , der Unterschied ist kaum wahrnehmbar,

# Was sind adversarial attacks?



- Gegeben sei ein Bild  $x$ , welches durch  $f_\theta$  „korrekt“ klassifiziert wird [Bun+23].
- Ein (untargeted) adversarial example ist ein Bild  $x'$  „nahe“ bei  $x$ , s.d.  $f_\theta x'$  falsch klassifiziert.
- Intuitive, bedeutet „nahe“
  - $x$  sieht ähnlich aus wie  $x'$ , der Unterschied ist kaum wahrnehmbar,
  - ein Mensch würde  $x$  und  $x'$  gleich klassifizieren.

# Was sind adversarial attacks?

- Gegeben sei ein Bild  $x$ , welches durch  $f_\theta$  „korrekt“ klassifiziert wird [Bun+23].
- Ein (untargeted) adversarial example ist ein Bild  $x'$  „nahe“ bei  $x$ , s.d.  $f_\theta x'$  falsch klassifiziert.
- Intuitive, bedeutet „nahe“
  - $x$  sieht ähnlich aus wie  $x'$ , der Unterschied ist kaum wahrnehmbar,
  - ein Mensch würde  $x$  und  $x'$  gleich klassifizieren.
- In Realität wählen wir oft eine  $\ell^p$  Norm, um die Distanz zu messen,

$$\|x' - x\|_p^p = \sum_{i,j} |x_{i,j} - x'_{i,j}|^p$$

# Sind adversarial attacks rein additiv?



**Clean Image**

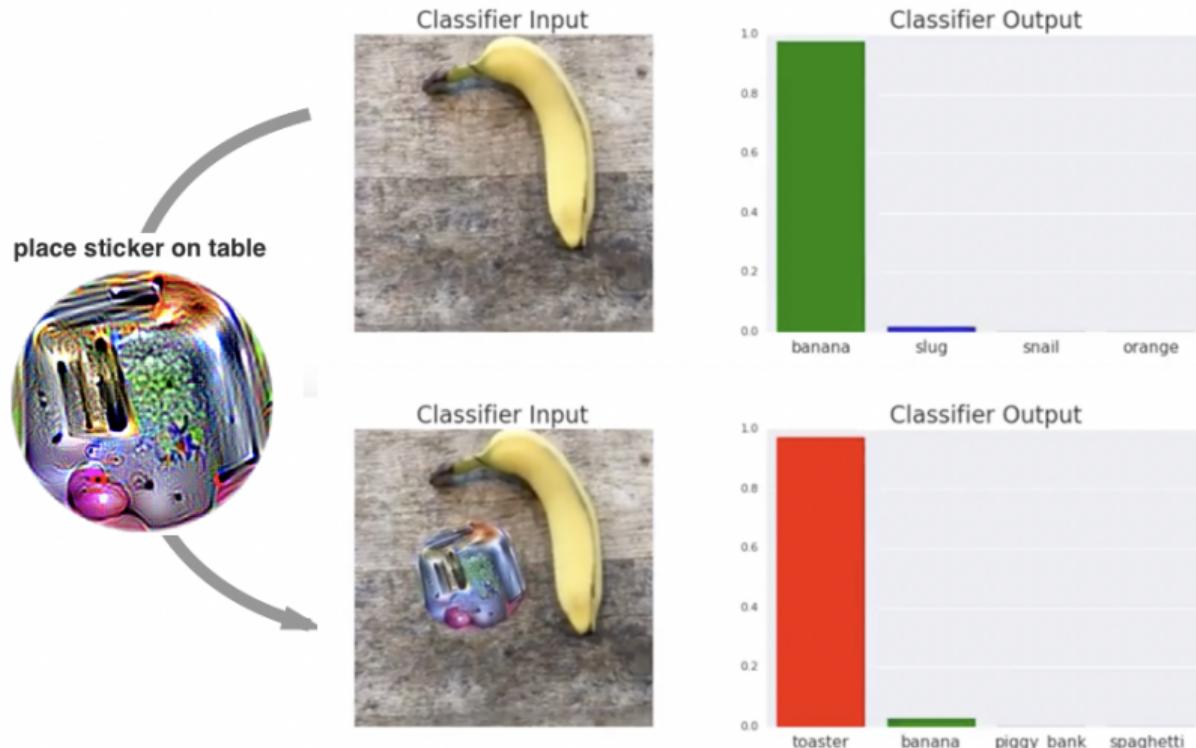
35.3% Soccer Ball  
7.4% Pick  
4.3% Golf Ball



**Adversarial Example**

46.1% Shower Cap  
9.2% Bonnet  
8.0% Tissue

# Adversarial Patches: [Bro+17]



# Wie produziert man adversarial Examples?



Gegeben sei ein Bild  $x$ , mit echtem Label  $y$  und das Budget  $\varepsilon$ . Wir betrachten das Problem

$$\max_{x': \|x-x'\| \leq \varepsilon} \ell(y, f_\theta(x'))$$

# Wie produziert man adversarial Examples?



Gegeben sei ein Bild  $x$ , mit echtem Label  $y$  und das Budget  $\varepsilon$ . Wir betrachten das Problem

$$\max_{x': \|x - x'\| \leq \varepsilon} \ell(y, f_\theta(x'))$$

Projezierte Gradienten Aufstiegs Methode: [GSS14]:

$$x^{k+1} = \Pi_{B_\varepsilon(x)} \left( x^k + \eta \nabla_x \ell(y, f_\theta(x^k)) \right).$$

# Wie produziert man adversarial Examples?



Gegeben sei ein Bild  $x$ , mit echtem Label  $y$  und das Budget  $\varepsilon$ . Wir betrachten das Problem

$$\max_{x': \|x - x'\| \leq \varepsilon} \ell(y, f_\theta(x'))$$

Projezierte Gradienten Aufstiegs Methode: [GSS14]:

$$x^{k+1} = \Pi_{B_\varepsilon(x)} \left( x^k + \eta \nabla_x \ell(y, f_\theta(x^k)) \right).$$

„Open-box“: Wir kennen die Gradienten des Netzwerks auswerten.

„Closed-box“: Gradienten des Netzwerks werden nicht benutzt [Ily+18].

# Verteidigung gegen Adversarial Attacks

# Adversarial Training



Folgende Arbeiten:

- Kurakin, I. Goodfellow, Bengio u. a.: *Adversarial examples in the physical world*,
- Madry, Makelov, Schmidt, Tsipras und Vladu: „*Towards deep learning models resistant to adversarial attacks*“,

# Adversarial Training



Folgende Arbeiten:

- Kurakin, I. Goodfellow, Bengio u. a.: *Adversarial examples in the physical world,*
- Madry, Makelov, Schmidt, Tsipras und Vladu: „*Towards deep learning models resistant to adversarial attacks*“,

schlagen „adversarial training“ vor, welches das folgende Problem betrachtet:

$$\min_{\theta} \mathbb{E}_{(x,y)} \max_{\delta \in B_{\varepsilon}(0)} \ell(f_{\theta}(x + \delta), y).$$

# Andere Ideen?: Lipschitz Stetigkeit



Lipschitz Stetigkeit:

$$(\forall x, x' \in \mathcal{X}) \quad \|f_\theta(x) - f_\theta(x')\|_{\mathcal{Y}} \leq L \|x - x'\|_{\mathcal{X}}.$$

# Andere Ideen?: Lipschitz Stetigkeit



Lipschitz Stetigkeit:

$$(\forall x, x' \in \mathcal{X}) \quad \|f_\theta(x) - f_\theta(x')\|_{\mathcal{Y}} \leq L \|x - x'\|_{\mathcal{X}}.$$

Die Lipschitzkonstante ist dann gegeben durch

$$\text{Lip}(f_\theta) := \sup_{x, x' \in \mathcal{X}} \frac{\|f_\theta(x) - f_\theta(x')\|_{\mathcal{Y}}}{\|x - x'\|_{\mathcal{X}}}.$$

# Regularisierung mit der Lipschitzkonstante



Wir betrachten

$$\min_{\theta \in \Theta} \frac{1}{|\mathcal{T}|} \sum_{(x,y) \in \mathcal{T}} \ell(f_\theta(x), y) + \lambda \text{Lip}(f_\theta).$$

# Regularisierung mit der Lipschitzkonstante



Wir betrachten

$$\min_{\theta \in \Theta} \frac{1}{|\mathcal{T}|} \sum_{(x,y) \in \mathcal{T}} \ell(f_\theta(x), y) + \lambda \text{Lip}(f_\theta).$$

Aber:  $\text{Lip}(f_\theta)$  zu berechnen ist NP hart [SV18].

# Regularisierung mit der Lipschitzkonstante



Wir betrachten

$$\min_{\theta \in \Theta} \frac{1}{|\mathcal{T}|} \sum_{(x,y) \in \mathcal{T}} \ell(f_\theta(x), y) + \lambda \text{Lip}(f_\theta).$$

Aber:  $\text{Lip}(f_\theta)$  zu berechnen ist NP hart [SV18].

Alternativ: Benutze die Abschätzung

$$\text{Lip}(f_\theta) \leq \prod_{l=1}^L \text{Lip}(\Phi_l) \leq C_\sigma \prod_{l=1}^L \|W_l\|, \quad (1)$$

# Regularisierung mit der Lipschitzkonstante



Wir betrachten

$$\min_{\theta \in \Theta} \frac{1}{|\mathcal{T}|} \sum_{(x,y) \in \mathcal{T}} \ell(f_\theta(x), y) + \lambda \text{Lip}(f_\theta).$$

Aber:  $\text{Lip}(f_\theta)$  zu berechnen ist NP hart [SV18].

Alternativ: Benutze die Abschätzung

$$\text{Lip}(f_\theta) \leq \prod_{l=1}^L \text{Lip}(\Phi_l) \leq C_\sigma \prod_{l=1}^L \|W_l\|, \quad (1)$$

Aber: Überschätzt  $\text{Lip}(f_\theta)$  [ALG19; Gou+20; KMP20; RKH19].

# Lipschitz Training Algorithmus: [Bun+21]



Die Idee aus [Bun+21]: Approximiert die Lipschitzkonstante  $\text{Lip}(f_\theta)$  auf einer endlichen aber variablen Menge  $\mathcal{X}_{\text{Lip}} \subset \mathcal{X} \times \mathcal{X}$ :

$$\text{Lip}(f_\theta) \approx \max_{(x,x') \in \mathcal{X}_{\text{Lip}}} \frac{\|f_\theta(x) - f_\theta(x')\|}{\|x - x'\|}.$$

# Adversarial Updates: Toy Example



# Adversarial Updates: Toy Example

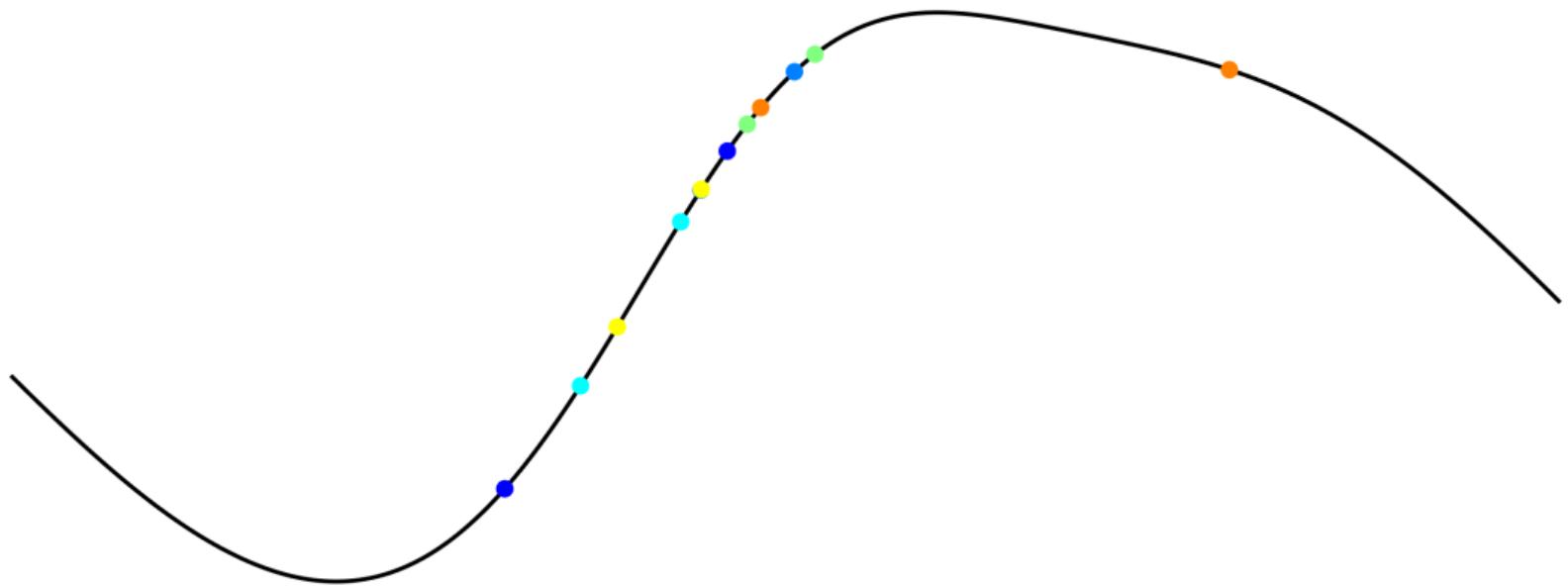


Abbildung: Adversarial Paare für eine 1D function.

# Adversarial Updates: Toy Example

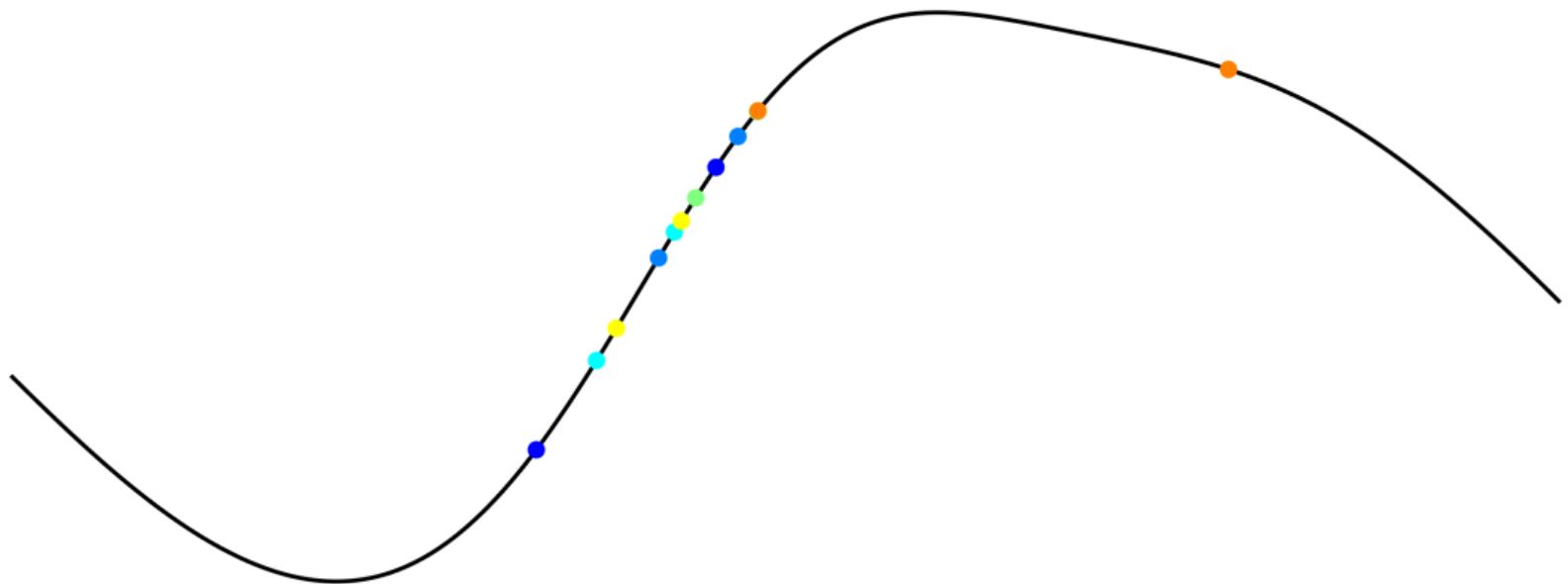


Abbildung: Adversarial Paare für eine 1D function.

# Adversarial Updates: Toy Example

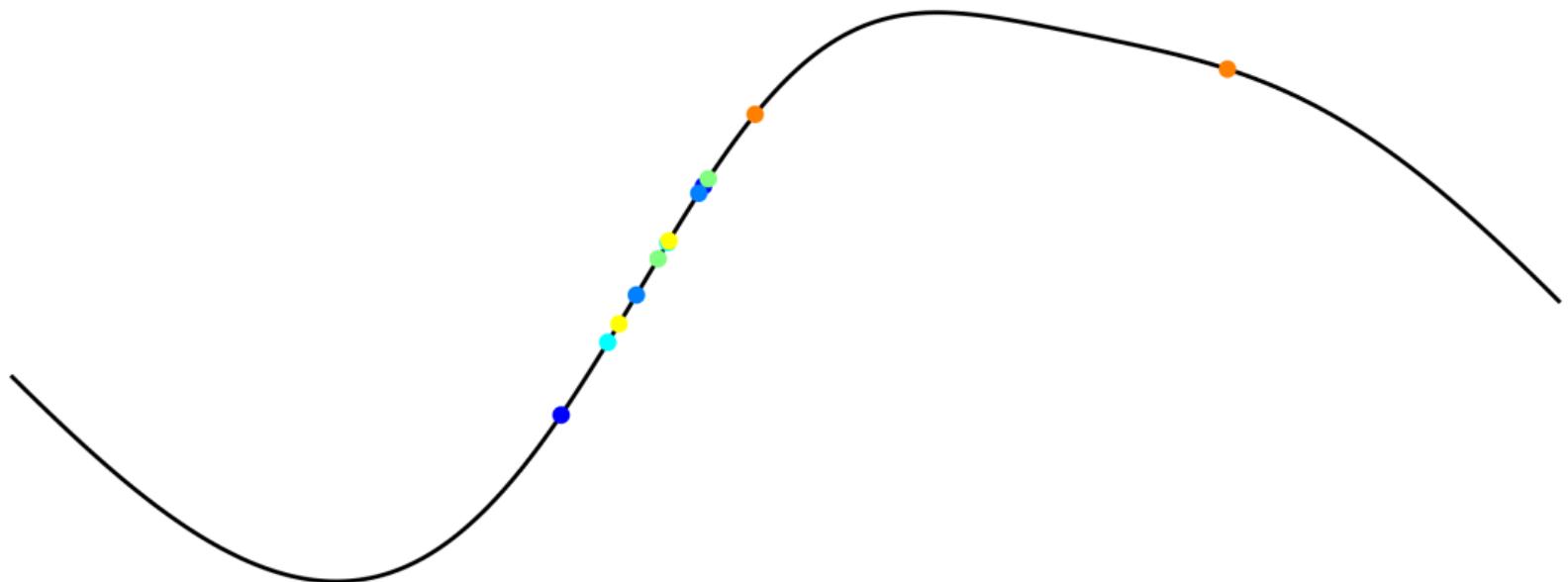


Abbildung: Adversarial Paare für eine 1D function.

# Adversarial Updates: Toy Example

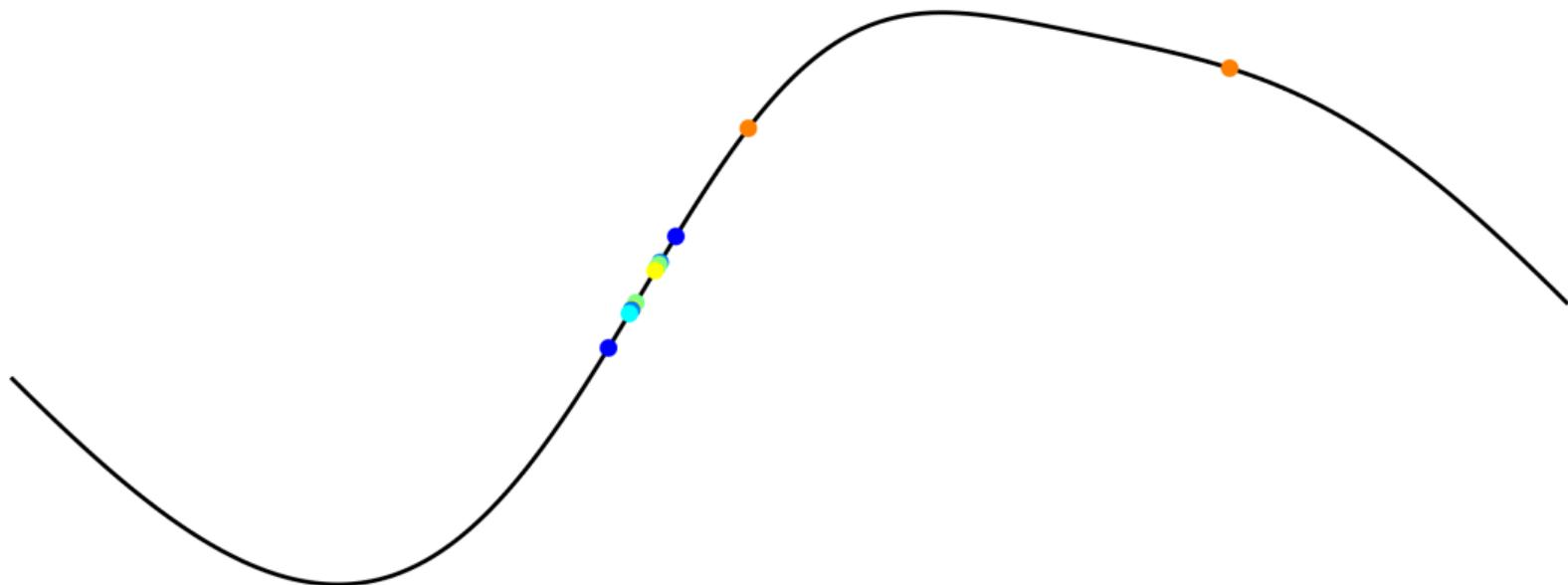


Abbildung: Adversarial Paare für eine 1D function.

# Adversarial Updates: Toy Example

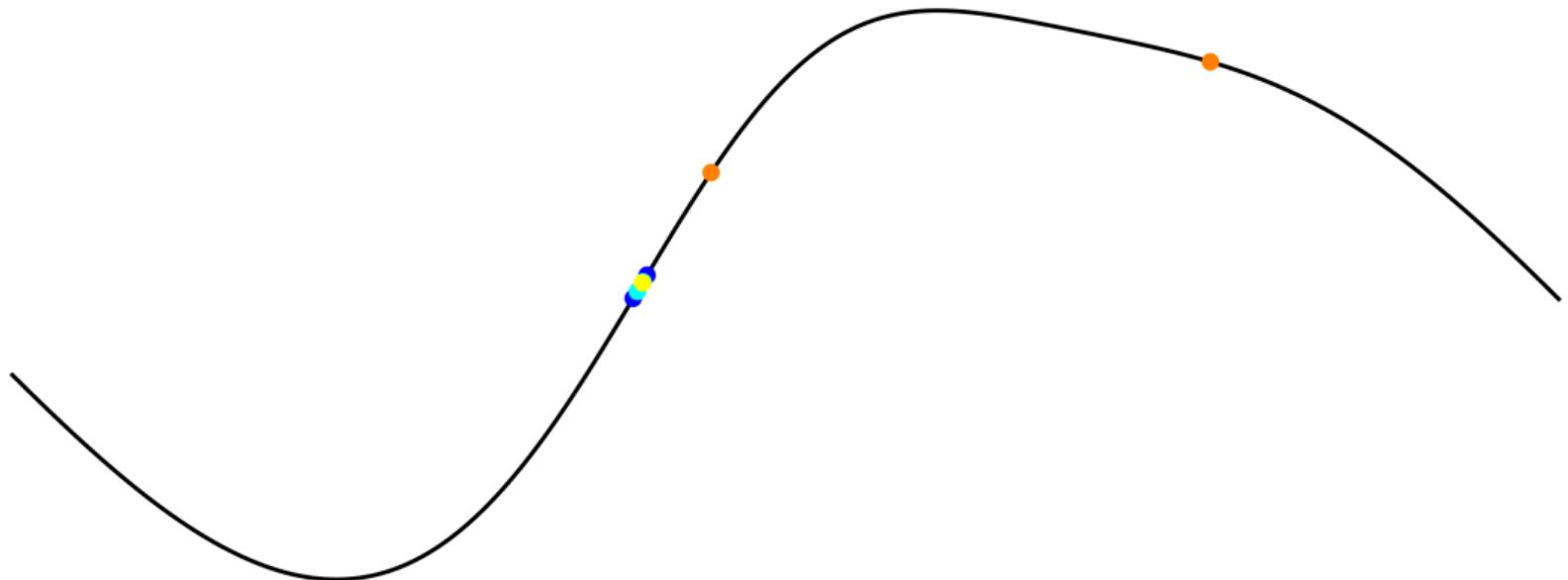


Abbildung: Adversarial Paare für eine 1D function.

# Adversarial Updates: Toy Example

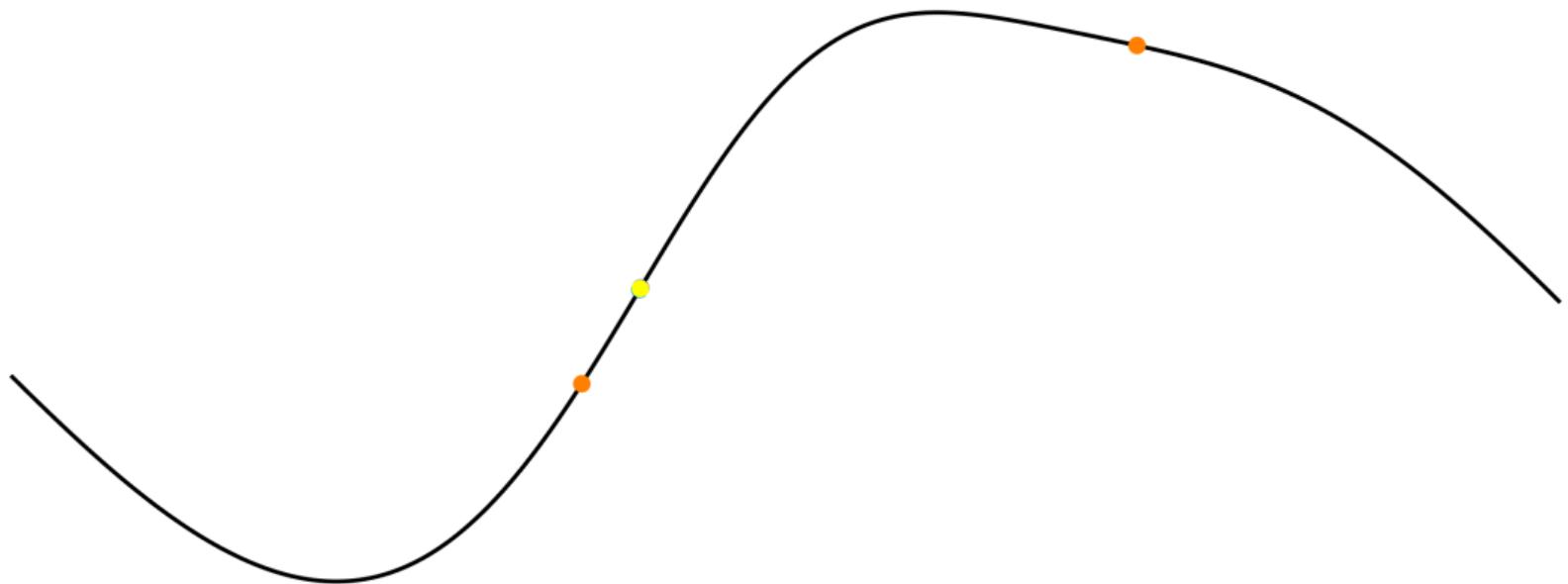


Abbildung: Adversarial Paare für eine 1D function.

# Adversarial Updates: Toy Example

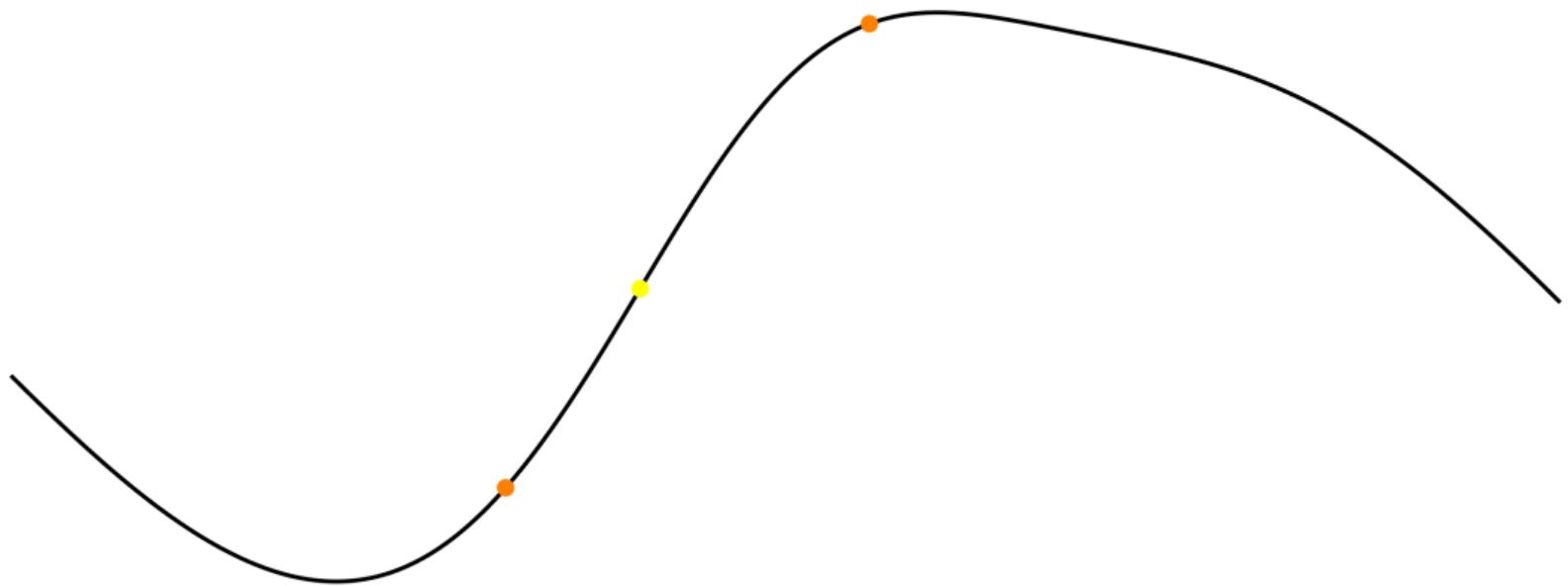


Abbildung: Adversarial Paare für eine 1D function.

# Adversarial Updates: Toy Example

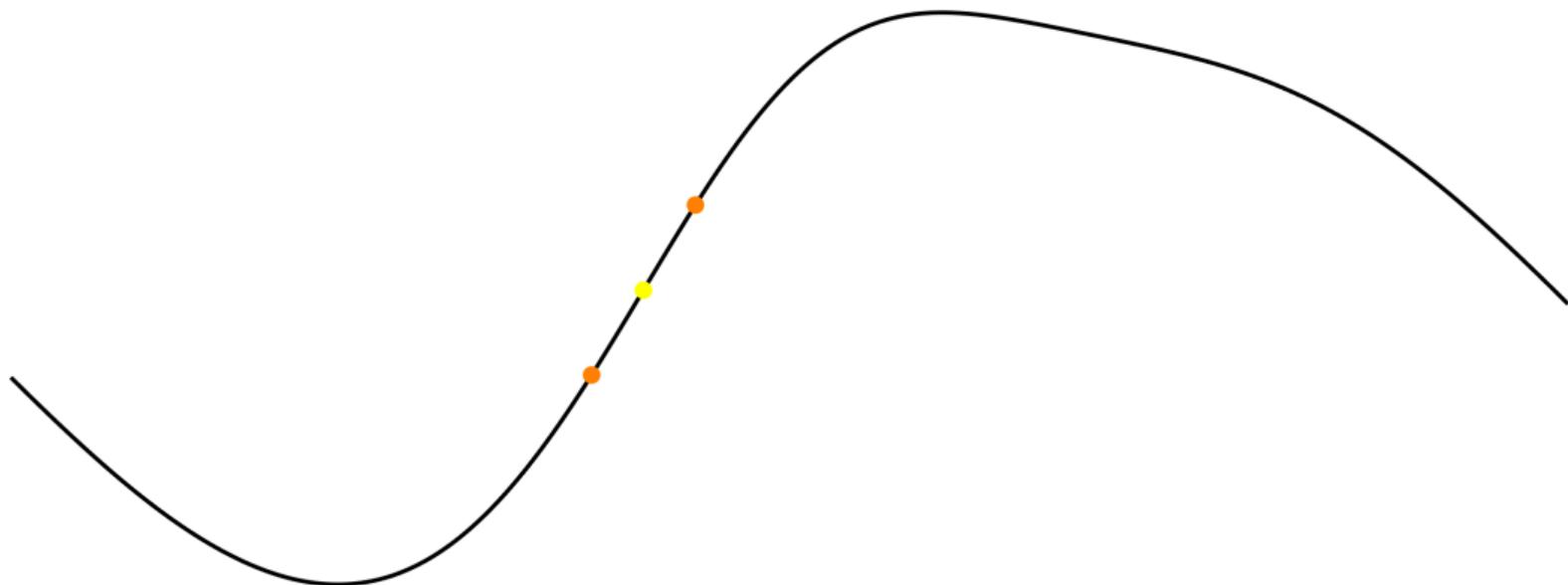


Abbildung: Adversarial Paare für eine 1D function.

# Adversarial Updates: Toy Example

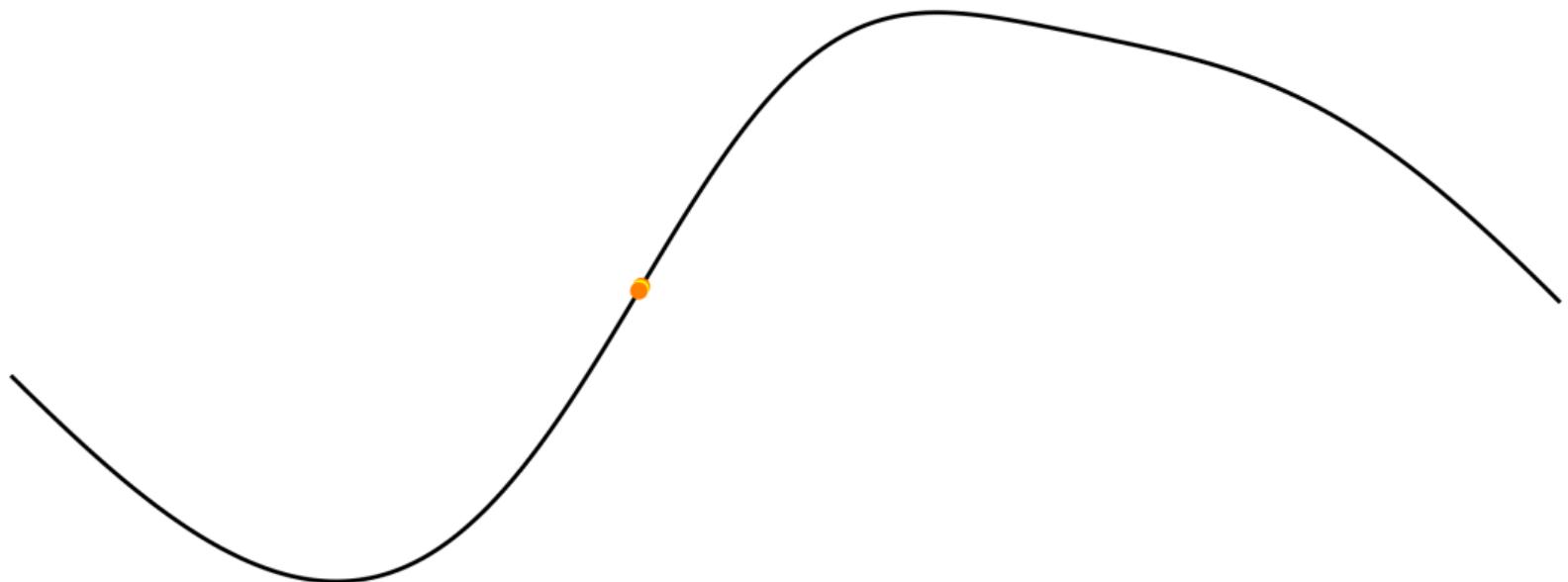


Abbildung: Adversarial Paare für eine 1D function.

# Adversarial Updates: Toy Example

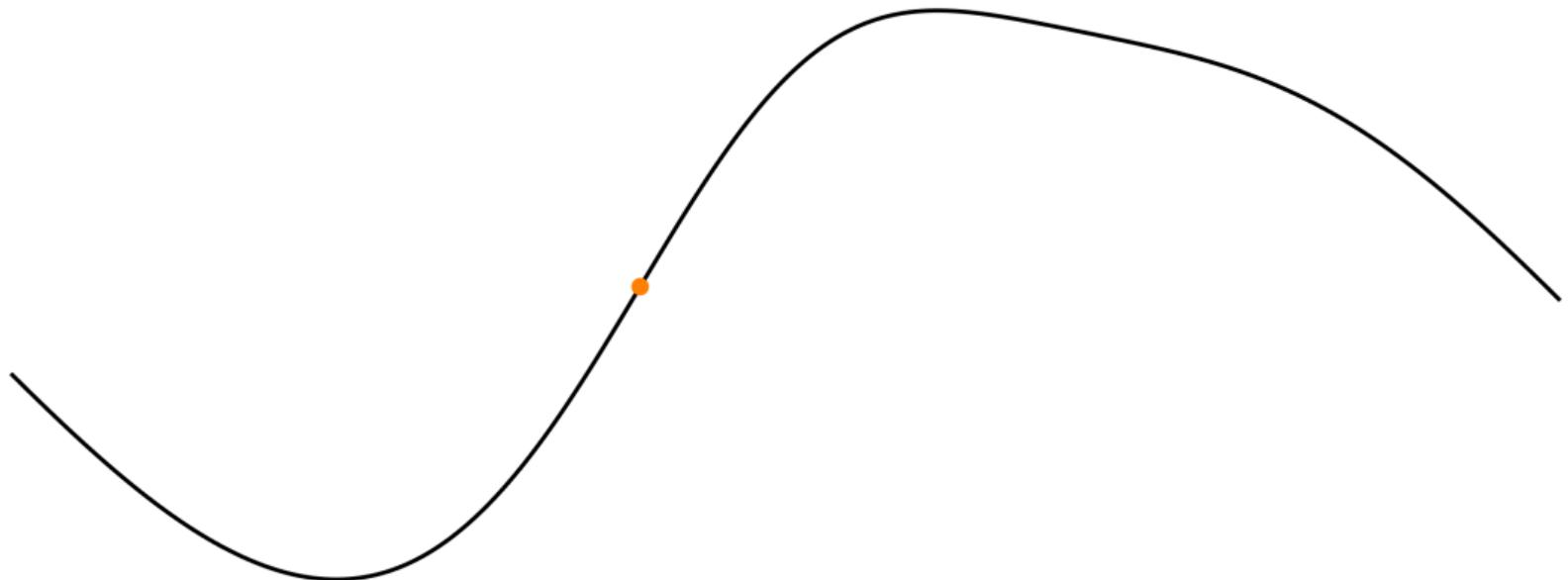
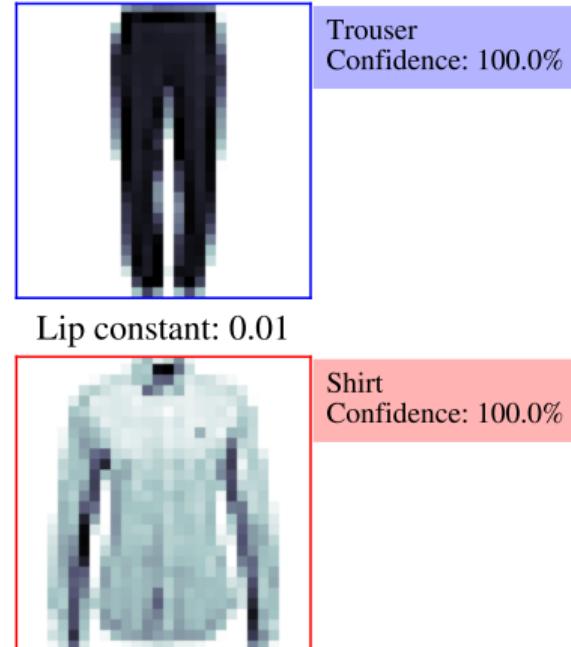
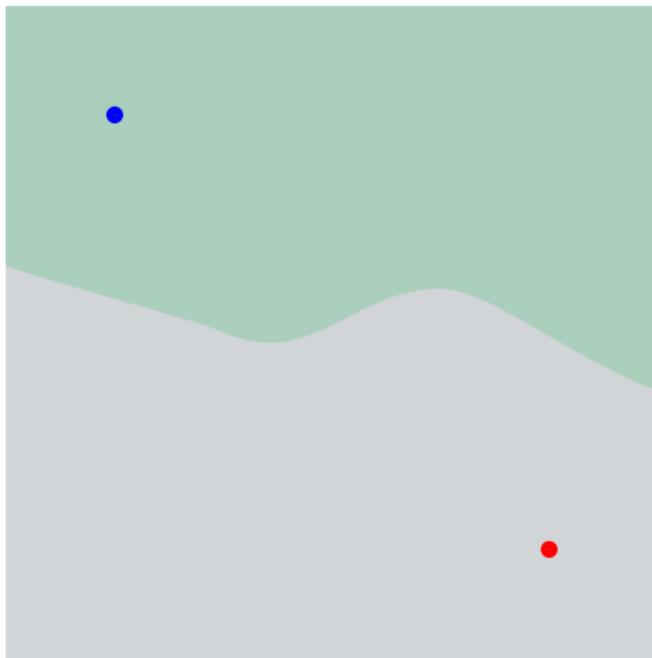
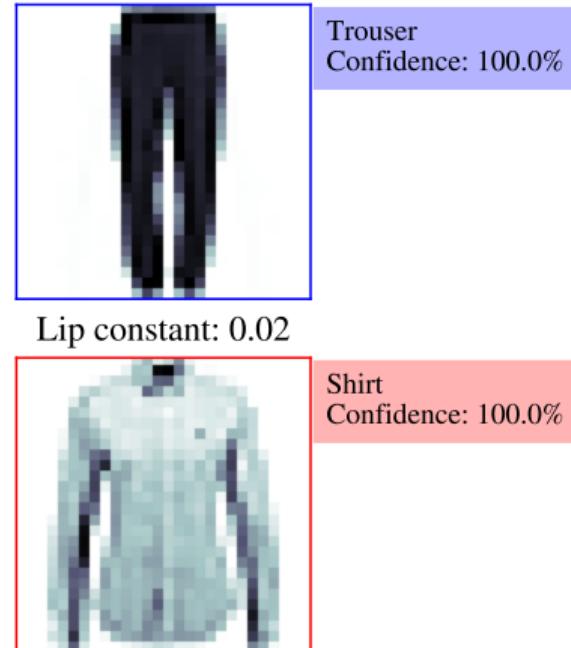
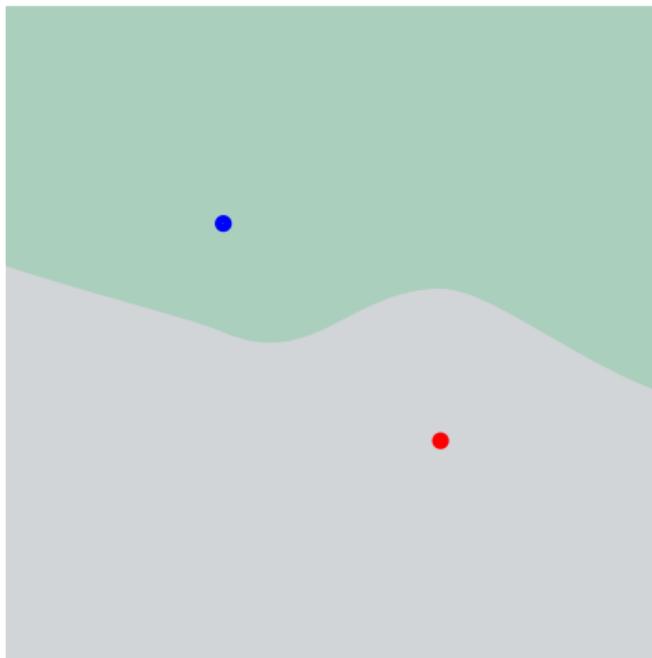


Abbildung: Adversarial Paare für eine 1D function.

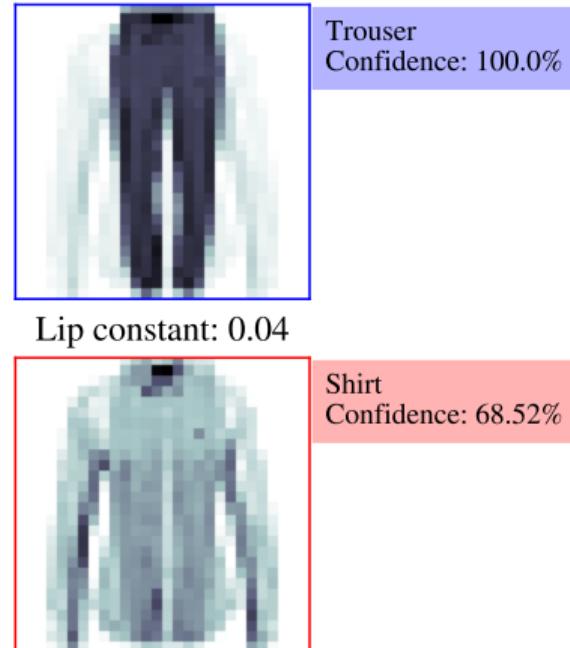
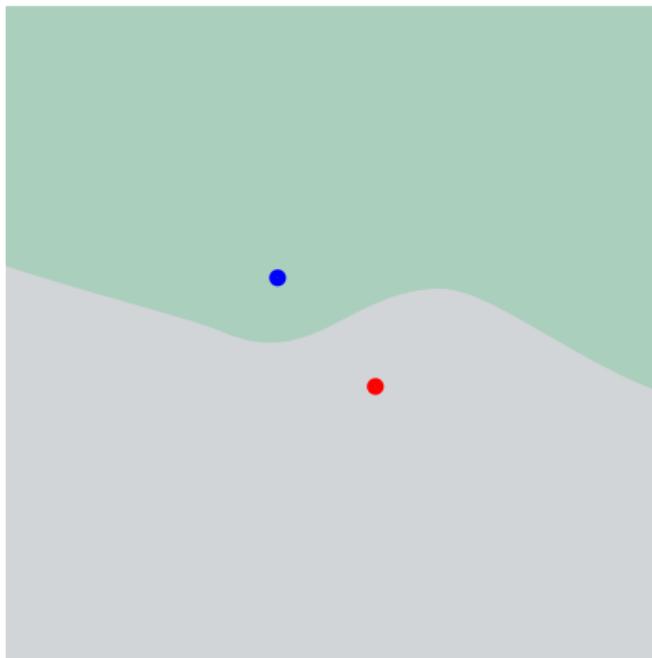
# Adversarial Updates: Fashion-MNIST



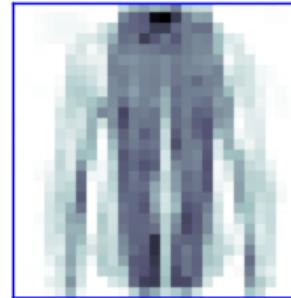
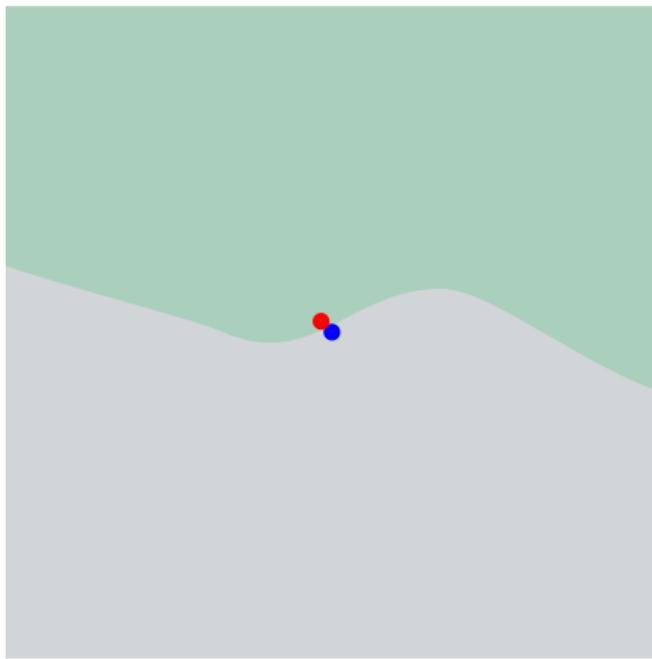
# Adversarial Updates: Fashion-MNIST



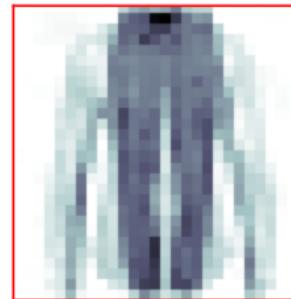
# Adversarial Updates: Fashion-MNIST



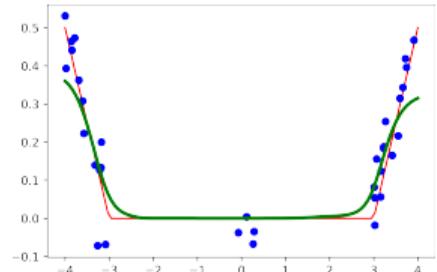
# Adversarial Updates: Fashion-MNIST



Lip constant: 17.21

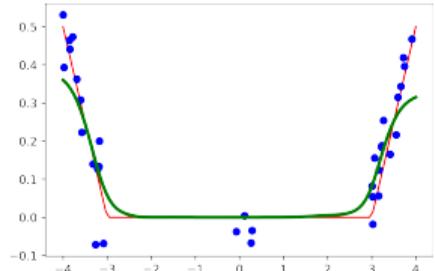


# Regression in 1D

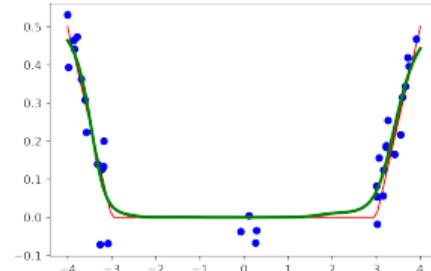


(a)  $\lambda = 10$

# Regression in 1D

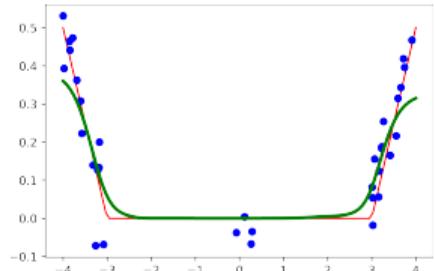


(a)  $\lambda = 10$

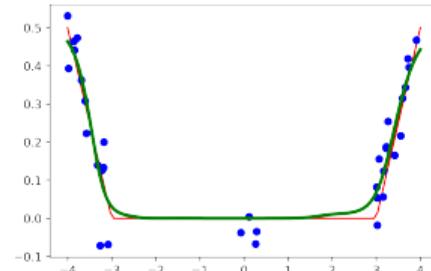


(b)  $\lambda = 1$

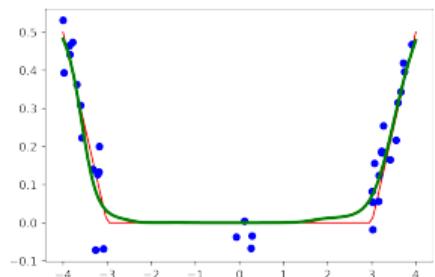
# Regression in 1D



(a)  $\lambda = 10$

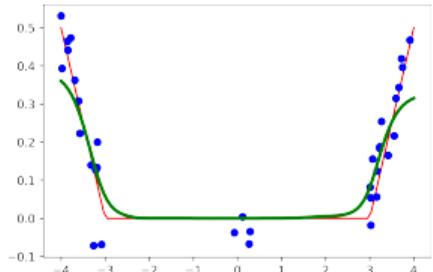


(b)  $\lambda = 1$

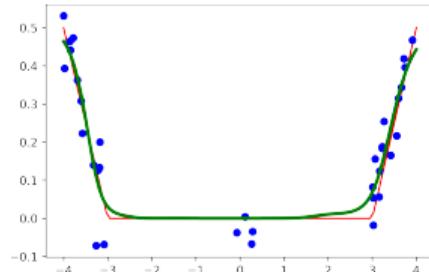


(c)  $\lambda = 10^{-10}$

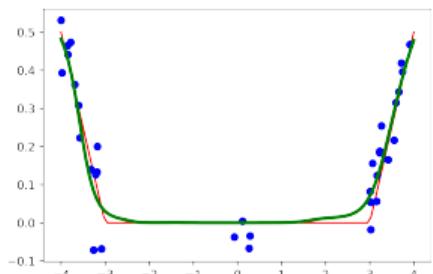
# Regression in 1D



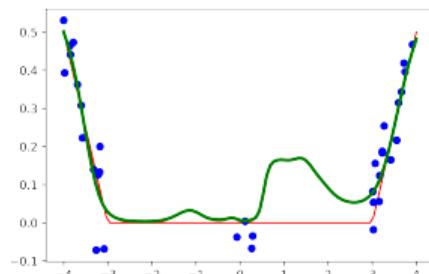
(a)  $\lambda = 10$



(b)  $\lambda = 1$



(c)  $\lambda = 10^{-10}$



(d)  $\lambda = 0$

# Some Empirical Results

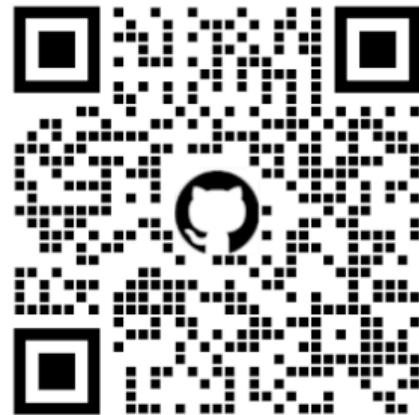


	Training Type	Train	Test	Noise	PGD	$\mu, \lambda$
MNIST	Standard	<b>95.6</b>	89.8	71.8	25.3	0
	Weight Reg. <sub>.95</sub>	93.1	89.0	71.4	25.4	0.0003
	Weight Reg. <sub>.90</sub>	89.8	89.7	71.9	24.8	0.0015
	Weight Reg. <sub>.85</sub>	84.8	87.2	72.3	24.9	0.0031
	<b>CLIP<sub>95</sub></b>	95.3	90.7	70.3	24.4	0.01
	<b>CLIP<sub>90</sub></b>	91.8	<b>91.6</b>	<b>78.0</b>	36.2	3.91
	<b>CLIP<sub>85</sub></b>	87.6	87.2	74.5	<b>37.7</b>	9.82
Fashion-MNIST	Standard	<b>98.5</b>	<b>92.0</b>	25.7	1.9	0
	Weight Reg. <sub>.95</sub>	94.8	91.2	26.9	3.5	0.0011
	Weight Reg. <sub>.90</sub>	89.8	89.6	28.1	6.0	0.0023
	Weight Reg. <sub>.85</sub>	85.3	85.4	<b>34.2</b>	10.2	0.0091
	<b>CLIP<sub>95</sub></b>	94.6	<b>91.9</b>	19.8	9.1	0.18
	<b>CLIP<sub>90</sub></b>	90.6	88.9	23.3	13.0	0.74
	<b>CLIP<sub>85</sub></b>	85.9	83.1	31.9	<b>14.9</b>	4.18

# Some Empirical Results



	Training Type	Train	Test	Noise	PGD	$\mu, \lambda$
MNIST	Standard	<b>95.6</b>	89.8	71.8	25.3	0
	Weight Reg. <sub>.95</sub>	93.1	89.0	71.4	25.4	0.0003
	Weight Reg. <sub>.90</sub>	89.8	89.7	71.9	24.8	0.0015
	Weight Reg. <sub>.85</sub>	84.8	87.2	72.3	24.9	0.0031
	<b>CLIP<sub>.95</sub></b>	95.3	90.7	70.3	24.4	0.01
	<b>CLIP<sub>.90</sub></b>	91.8	<b>91.6</b>	<b>78.0</b>	36.2	3.91
	<b>CLIP<sub>.85</sub></b>	87.6	87.2	74.5	<b>37.7</b>	9.82
Fashion-MNIST	Standard	<b>98.5</b>	<b>92.0</b>	25.7	1.9	0
	Weight Reg. <sub>.95</sub>	94.8	91.2	26.9	3.5	0.0011
	Weight Reg. <sub>.90</sub>	89.8	89.6	28.1	6.0	0.0023
	Weight Reg. <sub>.85</sub>	85.3	85.4	<b>34.2</b>	10.2	0.0091
	<b>CLIP<sub>.95</sub></b>	94.6	<b>91.9</b>	19.8	9.1	0.18
	<b>CLIP<sub>.90</sub></b>	90.6	88.9	23.3	13.0	0.74
	<b>CLIP<sub>.85</sub></b>	85.9	83.1	31.9	<b>14.9</b>	4.18



[https://github.com/  
TimRoith/CLIP](https://github.com/TimRoith/CLIP)

# Thank You



This talk is based on: [L. Bungert, R. Raab, T. R. L. Schwinn und D. Tenbrinck. „CLIP: Cheap Lipschitz Training of Neural Networks“](#). In: *Scale Space and Variational Methods in Computer Vision*. 2021

# Thank You



This talk is based on: L. Bungert, R. Raab, T. R. L. Schwinn und D. Tenbrinck. „CLIP: Cheap Lipschitz Training of Neural Networks“. In: *Scale Space and Variational Methods in Computer Vision*. 2021

Check out our GitHub repository  
<https://github.com/TimRoith/CLIP>

