scientific reports

Check for updates

OPEN

Exact feature collisions in neural networks

Utku Ozbulak[1,2,3], Shodhan Rao[1,4], Wesley De Neve[1,2], Joris Vankerschaver[1,5], Arnout Van Messem[6,7] & Manvel Gasparyan[1,4,7]✉

Predictions made by deep neural networks have been shown to be highly sensitive to small changes made in the input space where such maliciously crafted data points containing small perturbations are being referred to as adversarial examples. On the other hand, recent research suggests that the same networks can also be extremely insensitive to changes of large magnitude, where predictions of two largely different data points are mapped to approximately the same output. In such cases, features of two data points are said to *approximately collide*, thus leading to largely similar predictions. Our results improve and extend prior work on approximate feature collisions in neural networks and provide specific criteria for data points to have colliding features from the perspective of weights of neural networks, revealing that neural networks (theoretically) not only suffer from features that approximately collide but also suffer from features that *exactly collide*. We identify sufficient conditions for the existence of such scenarios, hereby investigating a large number of deep neural networks that have been used to solve various computer vision problems. Furthermore, we propose the Null-space search, a numerical approach that does not rely on heuristics, to create data points with colliding features for any input and for any task, including, but not limited to, classification, segmentation, and localization.

Keywords Activation functions, Input transformations, Model vulnerability, Neural network classifiers, Null-space search, Rank and nullity

Abbreviations

ASMA	Adaptive segmentation mask attack
CAM	Class activation map
DNN	Deep neural network
FOSNN	Fractional order spiking neural network
FTCNTZNN	Fixed time convergence and noise tolerant zeroing neural network
LPIPS	Learned perceptual image patch similarity
LSTM	Long short term memory
MOQRNN	Multiple output quantile regression neural networks
PAC	Probably approximately correct
PGD	Projected gradient descent
PSNR	Peak signal to noise ratio
ReLU	Rectified linear unit
ResNet	Residual network
ResLNet	Deep residual long short term memory network
SSIM	Structural similarity index measure

Since the inception of deep neural networks (DNNs) with AlexNet[1], computer vision problems involving classification, segmentation, and localization found quickly adopted solutions[2–4]. Shortly after the wide-range adoption of DNNs in computer vision, other domains such as natural language processing[5], sequence analysis[6,7],

[1]Center for Biosystems and Biotech Data Science, Ghent University Global Campus, Incheon, South Korea. [2]IDLab-Department of Electronics and Information Systems, Ghent University, Ghent, Belgium. [3]Computational and Data Sciences Department, George Mason University Korea, Incheon, South Korea. [4]Department of Data Analysis and Mathematical Modelling, Ghent University, Ghent, Belgium. [5]Department of Mathematics, Computer Science, and Statistics, Ghent University, Ghent, Belgium. [6]Department of Mathematics, University of Liège, Liège, Belgium. [7]Arnout Van Messem and Manvel Gasparyan contributed equally to this work. ✉email: manvel.gasparyan@ugent.be

and radar-based detection[8] embraced the same networks and utilized them to solve a wide-range of complex problems.

Much to the surprise of the research community, features learned by DNNs were found to be easily abusable by adversarial examples[9], malicious data points created with the sole purpose of misleading machine learning models. Such adversarial examples are created by adding a small perturbation to an image, which then changes the prediction of that image drastically[10], meaning that small changes in inputs may lead to large changes in outputs (i.e., predictions). Although adversarial examples are shown to also be a threat in other domains, the vision domain in particular suffers greatly from this phenomenon due to the exercised perturbation often being invisible to the naked eye. As a result, mission-critical tasks such as self-driving cars[11] and medical image diagnosis[12], which typically take advantage of the predictive power of DNNs, suffer greatly from this vulnerability.

After the discovery of adversarial examples[9], substantial research efforts were spent to find a solution to this security flaw[13]. Among those research efforts, a subset focused their attention on demystifying the black-box nature of neural networks and improving the understanding of the feature space[14]. While research on the topic of adversarial examples has shown that neural networks are extremely sensitive to small amounts of (adversarial) perturbations, the work of Jacobsen et al.[15] and Li et al.[16] have revealed that the same neural networks are also impartial to certain other types of perturbations. In particular, Li et al.[16] have shown that neural networks that contain Rectified Linear Unit (ReLU) activations suffer from what they called *approximate feature collisions* where multiple (dissimilar) data points may have similar feature maps, leading to similar predictions.

In this work, we investigate the phenomenon of colliding features in order to improve the understanding of the neural network feature space. In particular, we improve and extend the work of Li et al.[16] to trainable weights and reveal the existence of data points with *exactly* colliding features, irrespective of the selected activation function. Based on our findings, we propose the Null-space search, a novel numerical method that uses the null-space of weights in creating colliding data points. Finally, expanding on the angle of adversarial vulnerability, we proceed to show the existence of adversarial examples with colliding features, which exacerbates the adversarial risk associated with DNNs.

Related work

Iterative generation of adversarial examples has gained traction in recent years[17] where many recently-proposed iterative attacks follow in the footsteps of Szegedy et al.[10] and Kurakin et al.[18]. The work of Sabour et al.[19] on the other hand approached the topic of adversarial examples from a different angle by trying to create adversarial examples that come with particular and pre-defined feature representations. Such approaches were employed to fool the interpretability of DNNs[20]. This line of work also led to the discovery of invariance-based adversarial examples[15, 21], which not only come with limited perturbation, but also with specified predictions. Such adversarial examples give rise to an identical (or almost identical) prediction compared to a data point in the dataset, thus making it impossible to differentiate from that genuine data point based on the output of the model[22, 23], which makes them relevant to feature collisions.

Another line of work, and the one on which our paper is mostly based, is the work of Li et al.[16] where approximate feature collisions based on activation functions were discovered. This work mainly takes advantage of the shortcomings of DNN activation functions, and in particular, rectifiers[24]. The shortcomings of rectifiers and their relation to the injectivity of networks are also thoroughly discussed in the works of Hein et al.[25] and Puthawala et al.[26]. Although our work follows the path ushered by the aforementioned works, we move our focus away from activation functions and towards trainable weights.

More recently, several studies have proposed novel neural network architectures and approaches that complement our discussion on network design, feature representations, and robustness. For example, Wang et al.[27] introduce Deep Residual Long Short Term Memory Network (ResLNet), a deep residual Long Short Term Memory (LSTM) network for action recognition that effectively captures long-range spatial-temporal features, highlighting advances in deep network architectures for complex data. Chen et al.[28] propose hybrid stochastic computing techniques that enable fault-tolerant neural network computations with low hardware cost, emphasizing efficiency and robustness in network implementations. In addition, Hao and Yang[29] develop Multiple-Output Quantile Regression Neural Networks (MOQRNN) to model multivariate conditional distributions, providing insights into neural network theory for structured outputs. Finally, Jin et al.[30] present a Fixed-Time Convergence and Noise-Tolerant Zeroing Neural Network (FTCNTZNN) for time-varying matrix inversion problems, illustrating theoretical advances in network design with guaranteed stability and robustness.

Recent studies further expand the landscape of neural network design, robustness, and decision-making relevant to feature collisions. For example, Zhou et al.[31] propose a hybrid neural-physical architecture for longitudinal vehicle dynamics modeling, demonstrating how hybrid networks can achieve accurate predictions with limited data while maintaining robustness across varying conditions. Zhang et al.[32] introduce Fractional-Order Spiking Neural Networks (FOSNNs), providing theoretical insights into multistability and global attractivity, which relate to robustness and reliability in neural systems. In the context of decision-making, Yuan et al.[33] survey Transformer-based reinforcement learning models, highlighting how modern architectures combine deep networks with sequential decision-making tasks. Finally, Cao et al.[34] analyze optimization strategies for low-resource learning within a Probably Approximately Correct (PAC) framework, showing approaches to achieve robust generalization under data scarcity.

Notations

We denote by $\boldsymbol{x}_n \in \mathbb{R}^q$ and $y_n \in \{1, \ldots, M\}$ a single data point and its categorical information, respectively. Let $\mathcal{D} = \{\boldsymbol{x}_n \in \mathbb{R}^q \,|\, n = 1, \ldots, N\}$ be a dataset containing N data points. In this setting, let $\theta \in \mathcal{P} \subseteq \mathbb{R}^{p_1 \times p_2}$ denote a parameter set, and let $g : \mathcal{P} \times \mathbb{R}^q \to \mathbb{R}^M$ denote a neural network classifier that yields a vector of

real-valued logit scores for each category. Logits obtained by the usage of a neural network with parameters θ are represented by $g(\theta, \boldsymbol{x}) \in \mathbb{R}^M$. In this system, the index that contains the largest logit is assigned as the categorical classification for a data point, $G(\theta, \boldsymbol{x}_n) = \arg\max_m \left(g(\theta, \boldsymbol{x}_n)_m\right)$. If $G(\theta, \boldsymbol{x}_n) = y_n$, the data point is correctly classified.

We represent the internal working of a neural network as a composition of layers

$$g(\theta, \boldsymbol{x}) = f^{(t)}(\boldsymbol{x}) := f_t \circ f_{t-1} \circ \ldots \circ f_1(\boldsymbol{x}), \tag{1}$$

with f_k and $f^{(k)}$ indicating a forward pass at the k^{th} layer and a forward pass until the k^{th} layer, respectively. Each of the layers can perform any of the commonly used operations in standard neural network architectures, such as convolution, attention, activation, and pooling[24,35,36]. The activation function is a mathematical operation applied to a neuron's output to introduce non-linearity, enabling the neural network to learn complex patterns.

In a feedforward neural network, the layers typically consist of alternating fully connected layers and activation layers,

$$f_1(\boldsymbol{x}) = \boldsymbol{x}^\top \boldsymbol{W}_1, \quad \ldots, \quad f_{k-1}(\boldsymbol{x}) = \alpha(\boldsymbol{x}), \quad f_k(\boldsymbol{x}) = x^\top \boldsymbol{W}_k, \quad f_{k+1}(\boldsymbol{x}) = \alpha(\boldsymbol{x}), \quad \ldots$$

where $\boldsymbol{W}_k \in \mathbb{R}^{d_k \times d_{k-1}}$ is the weight matrix at layer k, and α is the activation function applied elementwise.

Feature collisions

For neural networks of the form of Eq. (1), feature collisions can be defined as follows.

Definition 1 *(Generic Feature Collision)* Let $f^{(k)} : \mathbb{R}^{d_0} \to \mathbb{R}^{d_k}$ be the output of the k-th layer of a neural network. A *feature collision* occurs at layer k if there exist distinct inputs $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^{d_0}$ such that $f^{(k)}(\boldsymbol{x}_1) = f^{(k)}(\boldsymbol{x}_2)$.

It is important to note that if a feature collision occurs at layer k, i.e., $f^{(k)}(\boldsymbol{x}_1) = f^{(k)}(\boldsymbol{x}_2)$ for some distinct $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^{d_0}$, then this collision propagates to all subsequent layers: $f^{(\ell)}(\boldsymbol{x}_1) = f^{(\ell)}(\boldsymbol{x}_2), \quad \text{for all } \ell > k$, since $f^{(\ell)}$ depends on $f^{(\ell-1)}$.

Although colliding features have a number of implications, the most important one is having two different inputs being mapped to the same output. Here, we would like to make the following remark about inference using neural networks.

Remark 1 (Inference with neural networks) Since the layer operations for a neural network described in Eq. (1) are deterministic during inference time, if at the moment of inference we have $f^{(k)}(\boldsymbol{x}_1) = f^{(k)}(\boldsymbol{x}_2)$, then $g(\theta, \boldsymbol{x}_1) = g(\theta, \boldsymbol{x}_2)$.

Remark 1 is directly related to the property of injectiveness. It implies that if a neural network has data points with colliding features, this network is not injective, and vice versa. Then, given a neural network with parameters θ, if

$$\min_{\boldsymbol{x}_1 \neq \boldsymbol{x}_2} ||g(\theta, \boldsymbol{x}_1) - g(\theta, \boldsymbol{x}_2)|| = 0 \tag{2}$$

holds, that neural network is not injective. In this context, a non-injective transformation indicates (roughly speaking) that the domain of the transformation is larger than its image, and as such there will always be at least two different elements in the domain that will be mapped to the same output.

Proposition 1 For any neural network with parameters θ, Eq. (2) holds if $g(\theta, \boldsymbol{x})$ is non-injective with respect to its second argument.

In order to investigate the plausibility of Proposition 1, we need to determine the conditions for which Eq. (2) holds. A simple sufficient condition for the occurrence of feature collisions is that the dimension M of the target space is strictly smaller than the dimension q of the input space, i.e., $M < q$. In this case, Brouwer's invariance of domain theorem ensures that the map $G(\theta, \cdot) : \mathbb{R}^q \to \mathbb{R}^M$ cannot be injective. However, this condition is by no means necessary: architectures such as (variational) autoencoders or U-Nets define maps between spaces of the same dimensionality but often include intermediate layers of lower dimensionality, which can also give rise to feature collisions.

Feature collisions can occur due to different mechanisms within a neural network. Li et al.[16] investigated feature collisions and the validity of Proposition 1 based on activation functions. They defined colliding features based on activations as follows:

Definition 2 *(Feature collisions on activations)* Let α be the activation function and $f^{(k-1)}$ the output of layer $k-1$. Given two data points $\boldsymbol{x}_1 \neq \boldsymbol{x}_2$, a *feature collision* occurs at layer k if $\alpha(f^{(k-1)}(\boldsymbol{x}_1)) = \alpha(f^{(k-1)}(\boldsymbol{x}_2))$, even in the case where the pre-activations differ, i.e., $f^{(k-1)}(\boldsymbol{x}_1) \neq f^{(k-1)}(\boldsymbol{x}_2)$.

The Definitions 1 and 2 capture feature collisions at the level of layer outputs and activations. Definition 1 considers collisions directly on the layer outputs: two distinct inputs $\boldsymbol{x}_1 \neq \boldsymbol{x}_2$ collide if their outputs are identical, $f^{(k)}(\boldsymbol{x}_1) = f^{(k)}(\boldsymbol{x}_2)$. Definition 2 refines this notion by incorporating the activation function α: even if the pre-activation outputs differ, a collision occurs when the activated outputs are equal,

$\alpha(f^{(k-1)}(\boldsymbol{x}_1)) = \alpha(f^{(k-1)}(\boldsymbol{x}_2))$. The key difference is that Definition 2 accounts for the nonlinearity introduced by activations, which may induce collisions not present in the raw layer outputs.

In their work, Li et al.[16] scrutinized in particular the ReLU activation. ReLU, as defined by $\mathrm{ReLU}(z) = \max\{0, z\}$, is indeed non-injective: by definition, $\forall z_1 \neq z_2 \in \mathbb{R}_-$ it holds that $\mathrm{ReLU}(z_1) = \mathrm{ReLU}(z_2) = 0$.

Let us define the features of $\boldsymbol{x}$ at the k^{th} layer as $\boldsymbol{l}^{(\boldsymbol{x})} = \mathrm{ReLU}(f^{(k-1)}(\boldsymbol{x}))$ based on Definition 2 and ReLU activation. Two data points $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ have a feature collision if $\boldsymbol{l}_+^{(x_1)} = \boldsymbol{l}_+^{(x_2)}$, where $\boldsymbol{l}_+$ denotes the non-negative elements of $\boldsymbol{l}$, since all the negative elements will be mapped to 0 due to the ReLU activation. Because of the property highlighted above, thus far, the creation of data points with colliding features for DNNs containing such non-injective activation functions has been possible[15].

For other (injective) activation functions such as sigmoid or tanh, Li et al.[16] argue that the property of saturation (e.g., $\lim_{x\to\infty} \mathrm{sigmoid}(x) = 1$) may lead to *approximate feature collisions*. As a result, the observations regarding the colliding features made in the aforementioned work are mostly applicable to ReLU networks and do not generalize easily to other networks that do not have this type of activation.

Although ReLU activations used to be popular building blocks of neural networks, they have been increasingly replaced with other types of activation functions in recently proposed architectures[37–39]. In contrast to those building blocks, trainable parameters are a mainstay of DNN architectures, either in the form of convolutional or fully-connected layers. As such, different from previous research discussed thus far, we focus our attention on trainable weights and analyze their propensity for colliding features, independent of the data used for training as well as activation functions present in the model.

Feature collision on weights

For neural networks of the form of Eq. (1), we define feature collisions on trainable weights as follows.

Definition 3 *(Feature collisions induced by weights)* Let $\boldsymbol{W}_k \in \mathbb{R}^{d_k \times d_{k-1}}$ be the weight matrix at layer k. A *feature collision* occurs if there exist distinct inputs $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^{d_{k-1}}$ such that $\boldsymbol{W}_k(\boldsymbol{x}_1 - \boldsymbol{x}_2) = \boldsymbol{0}$, i.e., $\boldsymbol{x}_1 - \boldsymbol{x}_2 \in \ker(\boldsymbol{W}_k)$.

Definition 3 considers collisions from the perspective of the layer weights. Two inputs collide if their difference lies in the kernel of the weight matrix, $\boldsymbol{x}_1 - \boldsymbol{x}_2 \in \ker(\boldsymbol{W}_k)$. This formulation emphasizes the linear transformation performed by the weights and identifies the structural sources of collisions independent of activations. Compared to Definitions 1 and 2 (see Table 1), it provides a more algebraic viewpoint: while Definitions 1 and 2 detect collisions on the outputs or activated outputs of a layer, Definition 3 highlights how the network's weights can inherently induce feature collisions before any activation is applied.

Given two inputs with feature collisions at the k^{th} layer. Following Remark 1, we know that the outputs of the layers after k will also collide. Notably, if two inputs collide at the first layer, their features will be exactly the same for the rest of the network. Keeping that in mind, our analysis starts with investigating fully connected layers, particularly the first fully connected layer, where we make the following observation regarding the eigenvalues of the weights and the feasibility of having colliding features.

Lemma 1 (Feature collision on fully connected layers) *Let $\boldsymbol{v}_k = f^{(k-1)}(\boldsymbol{x}) \in \mathbb{R}^{q\times 1}$ be the column vector obtained from composite operations up to the $(k-1)^{\mathrm{th}}$ layer, and let f_k be a fully connected layer defined as $f_k(\boldsymbol{v}_k) = \boldsymbol{v}_k^T \mathrm{W}_k \in \mathbb{R}^{1\times n_k}$, where $\mathrm{W}_k \in \mathbb{R}^{q\times n_k}$ are the trainable weights. Then a necessary condition for a feature collision on the weights to occur at the k^{th} layer is that zero is an eigenvalue of $\mathcal{W}_k = \mathrm{W}_k \mathrm{W}_k^T \in \mathbb{R}^{q\times q}$. This condition is also sufficient if f_k is the first layer of the network.*

Proof Having a zero eigenvalue for $\mathcal{W}_k$ implies that $\mathrm{rank}(\mathcal{W}_k) < q$. Since $\mathrm{rank}(\mathcal{W}_k) = \mathrm{rank}(\boldsymbol{W}_k^\top)$, it holds that $\mathrm{rank}(\boldsymbol{W}_k^\top) < q$. From the rank-nullity theorem, we have

$$\mathrm{nullity}(\boldsymbol{W}_k^\top) + \mathrm{rank}(\boldsymbol{W}_k^\top) = q, \tag{3}$$

where $\mathrm{nullity}(\boldsymbol{W}_k^\top) = \dim(\ker(\boldsymbol{W}_k^\top))$. This implies that $\mathrm{nullity}(\boldsymbol{W}_k^\top) \geq 1$ (i.e., $\ker(\boldsymbol{W}_k^\top)$ is not trivial). Let $\boldsymbol{\varphi} \in \ker(\boldsymbol{W}_k^\top)$ be a non-zero basis vector. Then $\boldsymbol{W}_k^\top \boldsymbol{\varphi} = \boldsymbol{0}$. Transposing both sides, we obtain $\boldsymbol{\varphi}^\top \boldsymbol{W}_k = \boldsymbol{0}^\top$.

If $\boldsymbol{x}$ is an input to the network with $\boldsymbol{v}_k = f^{(k-1)}(\boldsymbol{x})$ the features at the k^{th} layer, then adding $\boldsymbol{\varphi}$ to $\boldsymbol{v}_k$ does not change the pre-activation at layer k,

Aspect	Definition 1: Generic features	Definition 2: Activation-induced	Definition 3: Weight-induced
Depends on layer output?	Yes	Yes (after activation)	Yes (linear layer output)
Depends on activation?	No	Yes	No
Depends on weights?	Indirectly	Indirectly	Directly
Mechanism of collision	Any equality of layer outputs	Non-injective activation maps multiple pre-activations to same output	Non-full-rank or zero-eigenvalue weight matrix produces collisions
Example	$f^{(k)}(\boldsymbol{x}_1) = f^{(k)}(\boldsymbol{x}_2)$	$\mathrm{ReLU}(-5) = 0$ and $\mathrm{ReLU}(-10) = 0$	$\boldsymbol{W}_k(\boldsymbol{x}_1 - \boldsymbol{x}_2) = 0$

Table 1. Comparison of Feature Collision Definitions.

$$(\boldsymbol{v}_k + \boldsymbol{\varphi})^\top \boldsymbol{W}_k = \boldsymbol{v}_k^\top \boldsymbol{W}_k + \boldsymbol{\varphi}^\top \boldsymbol{W}_k = \boldsymbol{v}_k^\top \boldsymbol{W}_k. \tag{4}$$

Equivalently, $f_k(\boldsymbol{v}_k) = f_k(\boldsymbol{v}_k + \boldsymbol{\varphi})$. If there exists an input $\boldsymbol{x}'$ such that

$$f^{(k-1)}(\boldsymbol{x}') = \boldsymbol{v}_k + \boldsymbol{\varphi} \tag{5}$$

then $\boldsymbol{x}$ and $\boldsymbol{x}'$ collide at the k^{th} layer. This condition is automatically satisfied for the first layer, since in that case $\boldsymbol{x}' = \boldsymbol{x} + \boldsymbol{\varphi}$. □

For feature collisions in the layers beyond the first one, Eq. (5) represents a non-linear equation that must be solved for $\boldsymbol{x}'$. A sufficient condition for this equation to have a solution is that $\boldsymbol{v}_k$ is in the interior of the range of $f^{(k-1)}$: in this case $\boldsymbol{\varphi}$ can be re-scaled so that $\boldsymbol{v}_k + \boldsymbol{\varphi}$ is still within the range, thus guaranteeing the existence of $\boldsymbol{x}'$. However, as our focus is for the most part on collisions at the first layer, we will not pursue this observation any further.

The observation made in Lemma 1 for fully connected layers can be easily extended to convolutional layers.

Remark 2 (Feature collision on convolutional layers) In the proof of Lemma 1, using the fact that $\ker(\boldsymbol{W})$ is non-trivial, we constructed a vector $\boldsymbol{v}_k + \boldsymbol{\varphi} \neq \boldsymbol{v}_k \in \mathbb{R}^q$ such that Eq. (2) holds. If the k^{th} layer is a convolutional layer such that $f_k(\boldsymbol{V}_k) = \boldsymbol{V}_k \boldsymbol{W}_k$, where $\boldsymbol{V}_k \in \mathbb{R}^{l \times q}$, we can construct a non-zero matrix $\boldsymbol{P} \in \mathbb{R}^{l \times q}$ such that $(\boldsymbol{V}_k + \boldsymbol{P})\boldsymbol{W}_k = \boldsymbol{V}_k \boldsymbol{W}_k$, by, for example, taking $\boldsymbol{P} = [\boldsymbol{\varphi} \dots \boldsymbol{\varphi}]^\top \in \mathbb{R}^{l \times q}$.

Using Lemma 1 and Remark 2, we show the existence of colliding features for neural networks that have at least a single trainable weight matrix with zero as an eigenvalue. Moreover, having larger weight matrices (i.e., increasing the width of the network) also makes it more likely to have zero as an eigenvalue, thus increasing the likelihood of the architecture at hand to have colliding features. This is made more precise in the supplementary material (see, Lemma 1).

In Lemma 1 and Remark 2, we established the existence of colliding features with respect to eigenvalues of the weights. A special case where this effect occurs is when the dimension of the domain of the linear transformation (i.e., $\boldsymbol{x}^\top \boldsymbol{W}$) is smaller than the dimension of its range.

Theorem 1 (Inevitability of colliding features in the first weight) *For all neural networks that contain a first trainable weight $\boldsymbol{W}_1 \in \mathbb{R}^{q \times p}$ such that $p < q$, Eq. (2) holds.*

Proof For any $\boldsymbol{W}_k \in \mathbb{R}^{q \times p}$ such that $p < q$ we have $\operatorname{rank}(\boldsymbol{W}_k) = \operatorname{rank}(\boldsymbol{W}_k^\top) \leq p$. Recalling the rank-nullity theorem, we have $\operatorname{nullity}(\boldsymbol{W}_k^\top) \geq 1$. □

Furthermore, neural networks with trainable weights $\boldsymbol{W}_k \in \mathbb{R}^{q \times p}$ satisfying $p < q$ inherently contain directions in which distinct inputs are mapped to identical or nearly identical representations.

In order to demonstrate the practicality of our observations, in Table 2, we evaluate the properties of a large number of classification, segmentation, and localization models, including the recently-proposed vision transformer models. Specifically, for all trainable weights of the considered models, we calculate

$$\nu(\theta) = \sum_{\boldsymbol{W}_i \in \theta} \mathbb{1}_{\{\operatorname{nullity}(W_i)\}},$$

the number of trainable weights that have zero as an eigenvalue, as shown in Lemma 1 and Remark 2, and

$$\mu(\theta) = \sum_{\boldsymbol{W}_i \in \theta} \mathbb{1}_{\{\operatorname{row}(W_i) < \operatorname{col}(W_i)\}},$$

the number of weights that perform a linear transformation, as shown in Theorem 1. With $\frac{1}{n_\theta}\nu(\theta)$ and $\frac{1}{n_\theta}\mu(\theta)$, we also provide the percentage of weights that fit the aforementioned descriptions compared to all weights. Moreover, with $\sum_{\boldsymbol{W}_i \in \theta} \operatorname{nullity}(\boldsymbol{W}_i)$, we provide the number of basis vectors in the kernel of weights satisfying Lemma 1 and Remark 2. Based on Table 2, it is clear that, due to a large number of trainable weighs satisfying the conditions we discussed thus far, all of the models listed in Table 2 indeed suffer from potentially colliding features. In the appendix, we also evaluate a number of adversarially-secure models.

Our observations indicate that it is highly likely to have colliding features when using popular DNNs. In the following section, we discuss methods of finding colliding features.

Finding data points with colliding features

To show the existence of colliding features, thus far, a variety of heuristic-based approaches were employed. Specifically, the method proposed in the work of Sabour et al.[19] allows for the creation of data points that come with similar predictions to other, dissimilar data points. This approach was used in the work of Jacobsen et al.[15] to create so-called invariance-based adversarial examples. These adversarial examples are constructed by minimizing an ℓ_p norm distance between two logit vectors, $g(\theta, \boldsymbol{x})$ and $g(\theta, \hat{\boldsymbol{x}})$, through the use of gradient descent: $\hat{\boldsymbol{x}} = \min_{\widetilde{\boldsymbol{x}}} ||g(\theta, \boldsymbol{x}) - g(\theta, \widetilde{\boldsymbol{x}})||_p$. The created invariance-based adversarial example $\hat{\boldsymbol{x}}$ then has similar logit predictions as $\boldsymbol{x}$. Independent from the aforementioned work, Li et al.[16] also proposed another heuristic-

Task	Model	$\nu(\theta)$	$\frac{1}{n_\theta}\nu(\theta)$	$\mu(\theta)$	$\frac{1}{n_\theta}\mu(\theta)$	$\sum_{W_i \in \theta} \text{nullity}(W_i)$	$\text{nullity}(W_1)$
Classification	AlexNet[1]	8	100%	7	87%	16, 547	299
	Squeezenet[40]	16	61%	16	61%	3, 392	0
	VGG-16[41]	15	93%	14	87%	53, 357	0
	ResNet-50[2]	36	66%	33	61%	40, 701	84
	DenseNet-121[42]	121	100%	117	96%	90, 388	85
	Inception-V3[43]	95	96%	95	96%	80, 938	0
	ViT-Base[39]	18	36%	12	24%	27, 665	13
	ViT-Large[39]	46	47%	25	25%	73, 780	0
	DeiT-Tiny[38]	15	30%	13	26%	7, 490	576
	DeiT-Base[38]	16	32%	12	24%	27, 653	1
Segmentation	UNet[44]	44	93%	44	93%	73, 626	91
	Fcn[45]	39	68%	36	63%	67, 254	84
	PSPNet[46]	38	90%	38	90%	61, 446	90
	MobileNet-V3[4]	45	71%	42	66%	110, 267	84
	Lite R-ASPP[4]	32	48%	27	40%	8, 756	11
Localization	Mask R-CNN[3]	54	73%	50	68%	74, 677	83
	Faster R-CNN[47]	38	52%	32	43%	27, 366	11
	Keypoint R-CNN[3]	57	75%	53	69%	97, 230	84
	RetinaNet[48]	53	74%	50	70%	72, 845	72

Table 2. For the models provided in the second column, the number of trainable weights that satisfy Lemma 1 (3^{rd} column) and Theorem 1 (5^{th} column) with the percentage of those weights compared to all weights (4^{th} and 6^{th} columns) are provided. Moreover, the number of basis vectors in the kernel of all trainable weights (7^{th} column) and the number of basis vectors in only the first trainable weight (8^{th} column) are also given.

based minimization method similar to the one given above that allows the creation of data points with colliding features.

Different from the approaches detailed above, we will take advantage of the observations made in Lemma 1 and Remark 2 and employ the basis vectors in the kernel of the trainable weights, which in turn allows us to numerically create data points with colliding features. In order to demonstrate the feasibility of this approach, we define the Null-space search, a numerical method that allows for the creation of data points with colliding features.

Definition 4 (Null-space search) Let $\boldsymbol{W}_1 \in \mathbb{R}^{q \times p}$ denote the weights of the first layer in a neural network for which $\text{nullity}(\boldsymbol{W}_1) \geq 1$. We define the Null-space search as $\psi_{\boldsymbol{W}}(\boldsymbol{x}, \boldsymbol{\varphi}) = \boldsymbol{x} + \beta\boldsymbol{\varphi}$, where β is a scaling factor and where the perturbation vector $\boldsymbol{\varphi} \in \ker(\boldsymbol{W}_1)$ is one of the basis vectors selected from the kernel of $\boldsymbol{W}_1$, thus satisfying $\boldsymbol{\varphi} \in \mathbb{R}^p$, $\boldsymbol{W}_1\boldsymbol{\varphi} = \boldsymbol{0}, \forall \boldsymbol{\varphi} \in \ker(\boldsymbol{W}_1)$.

Data points created with the Null-space search not only have colliding features for the first weight but for all layers, including the final one (i.e., the prediction). Although this method uses the basis vectors of the first weight for the sake of straightforward calculation, it is also possible to use the subsequent weights to create colliding data points (provided that they satisfy the conditions discussed in Lemma 1 and Remark 2). However, in that scenario, one has to solve a system of non-linear equations, see Eq. (5), to find the necessary modifications.

In the rightmost column of Table 2, we provided the number of basis vectors in the kernel of the first trainable weight matrix of the evaluated models. As can be seen, most of the trained architectures indeed have $\text{nullity}(\boldsymbol{W}_1) \geq 1$, thus allowing the Null-space search to create colliding data points. Note that having a single basis vector in the kernel of the first weight is sufficient to generate countless colliding data points using different linear combinations of $\boldsymbol{x}$ and $\boldsymbol{\varphi}$.

In Fig. 1, we provide a visual illustration of the Null-space search for a classification scenario, through the usage of a trained Residual Network (ResNet)-50 on ImageNet. As shown in Fig. 1, the original image and its colliding counterpart have the same prediction, even though the adversarial example created with the proposed attack is visibly distorted by the perturbation obtained from the kernel of the first weight matrix.

Let us now outline a number of properties of the Null-space search.

Perturbation – Using the Null-space search when the first layer is a fully connected one is straightforward: we simply use a basis vector $\boldsymbol{\varphi}$ obtained from the kernel of the weight matrix. In the case when the first layer is a convolutional layer, we create the perturbation with $\boldsymbol{P} = [\boldsymbol{\varphi} \dots \boldsymbol{\varphi}]^\top$. In both cases, the degree of perturbation can be controlled with β.

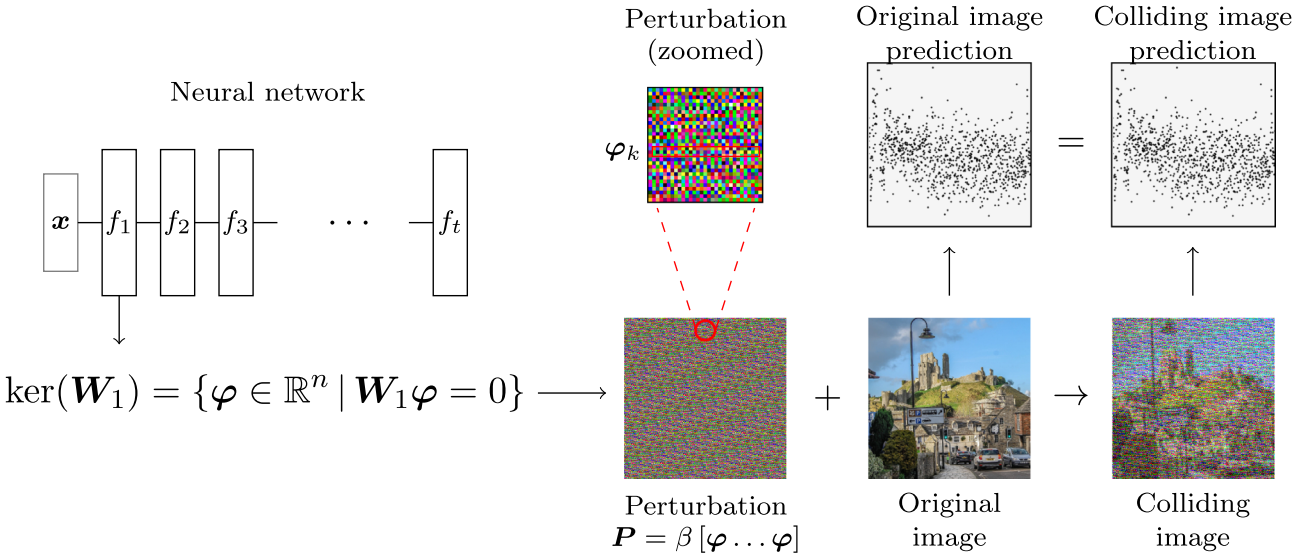

Fig. 1. A colliding image created with the Null-space search. The perturbation is produced through the repetition of a randomly selected basis vector obtained from the kernel of the first trainable weight matrix of a ResNet-50. Both the original image and the colliding image have exactly the same output.

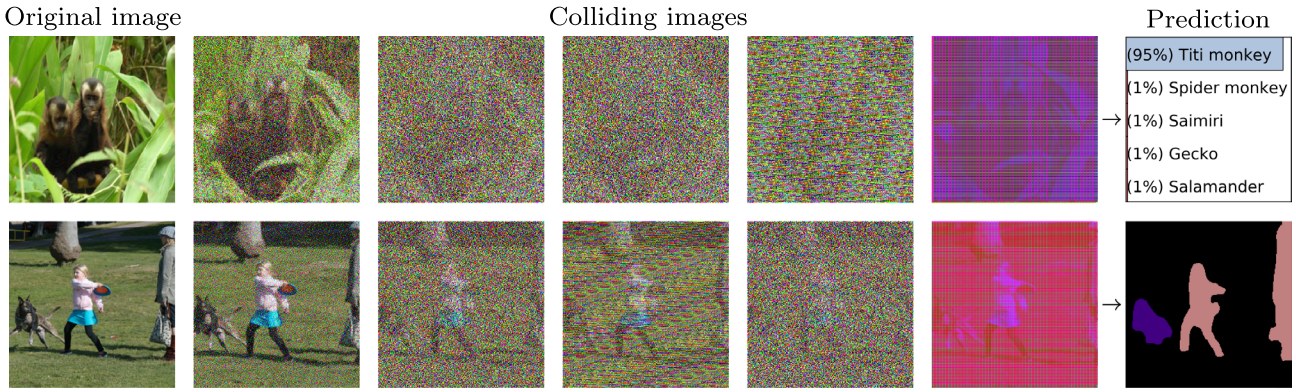

Fig. 2. Applications of the Null-space search for (top) classification and (bottom) segmentation. For classification and segmentation, we use Vit-B and MobileNet-V3, respectively. In the case of classification we provide the top-5 predictions and for segmentation, the predicted segmentation mask. Best viewed digitally with different levels of zoom.

In order to create the perturbation used in Fig. 1, we relied on a single basis vector scaled with the same perturbation multiplier $\boldsymbol{P} = \beta\,[\boldsymbol{\varphi} \ldots \boldsymbol{\varphi}]^\top$, $\beta \in \mathbb{R}$, where this basis vector originates from the kernel of $\boldsymbol{W}_1$ (see the red rectangle). Although we used the same basis vector for producing the perturbation in Fig. 1, we are not obliged to do so. Since any basis vector produces the same outcome (i.e., $(\boldsymbol{x} + \boldsymbol{\varphi}_i)\boldsymbol{W}_1 = \boldsymbol{x}\boldsymbol{W}_1$, $\forall \boldsymbol{\varphi}_i \in \ker(\boldsymbol{W}_1)$), it is possible to take different basis vectors (i.e., $\boldsymbol{P} = \beta\,[\boldsymbol{\varphi}_1\, \boldsymbol{\varphi}_2\, \ldots\, \boldsymbol{\varphi}_t]^\top$), thus producing different perturbations. Moreover, we are also not limited to the usage of a single perturbation multiplier. In this case, using individual perturbation multipliers (i.e., $\boldsymbol{P} = [\beta_1\boldsymbol{\varphi}_1\, \beta_2\boldsymbol{\varphi}_2\, \ldots\, \beta_t\boldsymbol{\varphi}_t]^\top$) makes it is possible to perturb a selected portion of the image while leaving other parts intact. In Fig. 2, we provide adversarial examples created with such approaches for a ViT-Base[39] and MobileNet-V3[4] trained on ImageNet.

Task-agnostic approach—Note that the Null-space search is task-agnostic, meaning that it is applicable for all types of neural networks that satisfy the properties listed in Lemma 1 or Remark 2 built to solve any kind of problem (e.g., classification, segmentation, or localization).

Utilizing heuristic-based perturbations—Since the Null-space search is a numerical method, it can also be used in conjunction with other heuristic methods that produce perturbations such as adversarial attacks in order to create colliding adversarial examples. For this type of approach, instead of a genuine data point, an adversarial example generated by any other attack is used as an input for the Null-space search in the following way:

$$\hat{\boldsymbol{x}} = \boldsymbol{x}' + \psi_{\boldsymbol{W}}(\boldsymbol{x}', \boldsymbol{\varphi}), \quad \boldsymbol{x}' = \Omega_\epsilon(\boldsymbol{x}), \tag{6}$$

where Ω_ϵ represents any adversarial attack such as Projected Gradient Descent (PGD)[49]. Even though the resulting adversarial example created with the Null-space search will have a different perturbation than the adversarial example created with the adversarial attack, they will have the same prediction $g(\theta, \hat{\boldsymbol{x}}) = g(\theta, \boldsymbol{x}')$. This outcome shows that it is indeed possible to have numerous adversarial examples that have the same colliding features and prediction but different perturbations, thus indicating that the security flaws of DNNs with regards to various types of adversarial examples are far worse than what our intuition tells us.

Impact of exact feature collisions

We now discuss the implications of having data points with colliding features.

Colliding features—First and foremost, two images with colliding features will have feature maps that are exactly the same. To add, the usage of basis vectors as the foundation of perturbation allows absurd scenarios where a colliding image created with the Null-space search is not recognizable due to added perturbation yet it has the same feature maps as well as prediction of its unperturbed counterpart.

Colliding predictions—Continuing on the topic above, perhaps the most straightforward consequence of data points with colliding features is that those data points will have the same prediction. This consequence, depending on the data points and the dataset at hand, can be seen as both a positive and a negative behavior of DNNs. In Fig. 2 we provide images with colliding features where the predictions of images on the same row are exactly the same. When the perturbation is mild and the object(s) of interest in those images are identifiable as their respective categories, colliding predictions may seem as behavior that is beneficial to DNNs. However, it is also possible to create data points with extremely large perturbation budgets where the objects of interest are not identifiable. In those cases, having a prediction that is the same as the original one becomes a detrimental behavior, since a confident prediction for a particular category is made for an essentially unrecognizable image.

Colliding interpretations—Two images with colliding features will have the same output with explainability methods that use *only* the forward pass such as Score-CAM)[50]. For the interpretability methods that use the gradient such as Grad-CAM[51] or Integrated Gradients[52], the result depends on the layer of first collision (i.e., the layer where the collision occurs for the first time) as well as the layer targeted for the creation of the interpretability map. In scenarios when the collision occurs at an earlier layer than the target of the interpretability method, the resulting output will be the same, thus further amplifying the confusion and distrust regarding the explainability of DNNs. Conversely, if the target layer for the interpretability method is earlier than or at the layer of the collision, the resulting interpretability maps differ.

DNN-based feature similarity methods—Capturing the complexity of human perception has been one of the biggest challenges in computer vision. A recent trend in the field is to replace shallow perception metrics such as the Structural Similarity Index Measure (SSIM) or Peak Signal-to-noise Ratio (PSNR) with more complex ones that rely on DNN features[53]. One such method that gained popularity is the Learned Perceptual Image Patch Similarity (LPIPS) proposed by Zhang et al.[54]. Although this method has been shown to have certain benefits over shallow metrics, its reliance on the features of DNNs makes it vulnerable to certain exploits, including images with colliding features. Specifically, according to LPIPS measurement, an initial image and colliding images created with the null-space search from such image are the same, regardless of the degree of perturbation (since the feature maps are exactly the same). In Fig. 3, we provide PSNR measurements calculated between 100 images as well as their colliding counterparts created with differing perturbation multipliers (β). As can be seen, PSNR can correctly identify the added perturbation which indicates that only relying on DNNs for image similarity may be risky and that there is still merit to the usage of shallow metrics such as PSNR.

Adversarial examples—As described in the "Finding data points with colliding features" section, it is possible to employ the Null-space search with ϵ-adversarial examples in order to generate even more adversarial examples with the same prediction. In order to demonstrate this, in Fig. 4, we provide approaches where adversarial examples are created by combining the Null-space search with PGD[49] or the Adaptive Segmentation Mask Attack (ASMA)[12] for classification and segmentation, respectively. In this scenario, colliding adversarial examples have the same prediction as the initial adversarial examples they are created from. Note that the degree of additive perturbation exercised by the Null-space search can be controlled with the selection of β and differing basis vectors. In this case, the added perturbation respects the initial ℓ_∞ limit set by the attacks, thus satisfying the property of invisible perturbation. This unsettling result indicates that the adversarial risk associated with DNNs

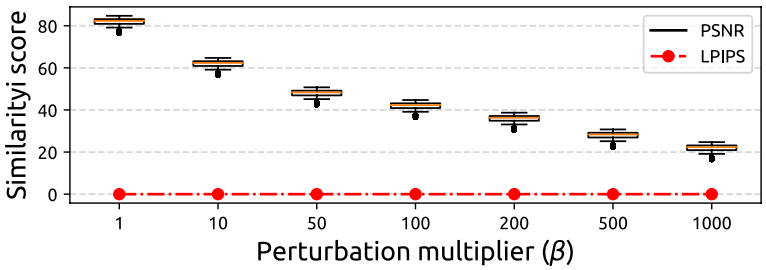

Fig. 3. PSNR and LPIPS similarity measurements calculated between the initial images and their perturbed counterparts for various values of β employed in the Null-space search. In terms of image similarity, a low PSNR score indicates a large perturbation whereas a low LPIPS score indicates a small perturbation, with an LPIPS score of 0 indicating that two images under consideration are the same.

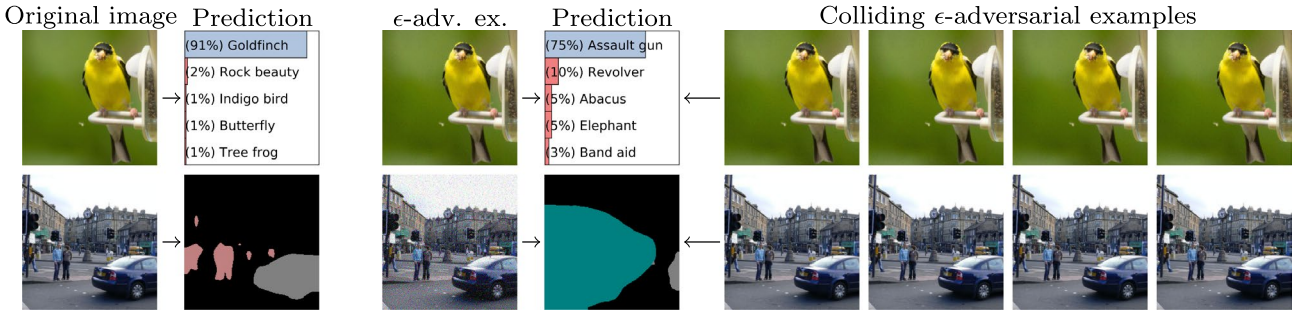

Fig. 4. Adversarial examples created with the Null-space search in conjunction with other adversarial attacks (PGD for classification and ASMA for segmentation). For classification and segmentation, we use Vit-B and MobileNet-V3, respectively. For classification and segmentation, we provide the top-6 predictions and the predicted segmentation masks, respectively.

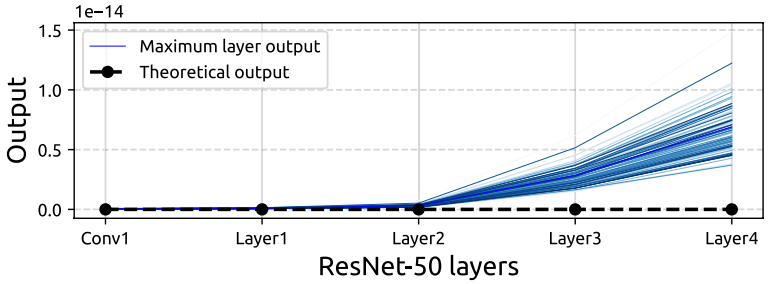

Fig. 5. Given an input $\boldsymbol{x}$ created with repetition of a basis vector $\boldsymbol{\varphi} \in \ker(\boldsymbol{W}_1)$ obtained from the first convolutional layer of ResNet-50, we plot $\max(f^{(k)}(\boldsymbol{x}))$, the maximum elements of outputs of the first convolutional layer as well as the subsequent residual bottlenecks of ResNet-50 using 84 different basis vectors (corresponding to 84 blue lines). Note that the y-axis is within the range of $[0, 1.5 \times 10^{-14}]$.

is worse than previously thought, since every adversarial examples created with an attack may have countless colliding data points that are also adversarial examples.

For all of the aforementioned use-cases, we provide additional examples for colliding features in the appendix.

Where theory slightly differs from practice

In the "Feature collision on weights" section we laid out theoretical evidence for the existence of colliding features in DNNs from the perspective of weights. According to Lemma 1, by definition of basis vectors ($\boldsymbol{\varphi} \in \ker(\boldsymbol{W}_1)$) we have $\boldsymbol{\varphi}^\top \boldsymbol{W}_1 = \boldsymbol{0}$ exactly. However, in practice, for large matrices such as the ones in DNNs, due to the limitations regarding the numerical precision, $\boldsymbol{\varphi}$ becomes an approximation up to machine precision, rather than an exact kernel element. As a consequence of this, $\boldsymbol{\varphi}^\top \boldsymbol{W}_1$ becomes numerically non-zero but remains on the order of floating-point round-off, instead of exactly zero. In order to illustrate the magnitude of this numerical effect, we provide Fig. 5, which plots the maximum output obtained from layers of ResNet-50 when the input is created with numerically computed basis vectors of the first weight. Note that this difference is extremely small (i.e., at the level of numerical precision) and does not have any impact on the observations made in the "Impact of exact feature collisions" section. We emphasize that the feature collisions are theoretically exact and numerically exact up to floating-point precision.

While our Null-space search constructs feature collisions explicitly using the null-space basis vectors of layer weights, it is important to note that such collisions can also arise naturally in trained networks. Even without adversarial construction, the presence of nontrivial null spaces implies that multiple inputs may map to identical or near-identical feature representations at certain layers. However, the specific collisions we generate in this work are designed to highlight and quantify these vulnerabilities, and may not always occur under normal data distributions. Understanding the frequency and conditions under which natural collisions occur is an important direction for future study.

Conclusions and outlook

In this paper, we revisited the phenomenon of colliding features from a formal and probabilistic standpoint. While the impossibility of injectivity for continuous maps from higher- to lower-dimensional spaces is classical, we extended this reasoning to trainable neural network layers and proposed the Null-space search method, which identifies colliding data points through the null-space of trained weights.

Our Null-space search constructs colliding data points by exploiting the null spaces of the layer weight matrices. While existing numerical approaches, such as Jacobian-based analysis of singular values or kernels[26, 55, 56], can

detect local non-injectivity, our method provides a complementary perspective. Specifically, it allows for global, layer-wise exploration of potential feature collisions rather than focusing only on local Jacobian properties.

Given the connection between colliding data points and adversarial examples, in future work, we believe the Null-space search, as well as the metrics evaluated in Table 2 (such as $\frac{1}{n_\theta}\nu(\theta)$, $\frac{1}{n_\theta}\mu(\theta)$, and $\sum_{\boldsymbol{W}_i \in \theta} \text{nullity}(\boldsymbol{W}_i)$) can be used as benchmarking methods to quantify and compare the adversarial risk associated with models. Given the lack of easily available quantitative evaluation metrics in the field of adversarial attacks, if such methods show correlation with the easiness of creation of adversarial examples, they can be expected to be helpful for fairly evaluating the adversarial risk associated with models.

Furthermore, as we have discussed in the "Feature collision on weights" section, having a larger (and wider) network also increases the chance of collisions. As such, supporting the ideas proposed by Heo et al.[57] and Goldblum et al.[58], we expect that knowledge distillation and having compact networks will be valuable in reducing collisions as well as to minimize the adversarial risk.

In addition, potential defenses could include careful weight initialization and regularization strategies aimed at reducing the nullity of weight matrices, such as structured pruning, spectral normalization, or weight decay. These techniques may help limit the dimension of the kernel, thereby reducing the likelihood of feature collisions. Investigating such strategies in future work could provide practical guidelines for improving the robustness of neural networks against adversarial vulnerabilities.

Finally, we plan to conduct a large-scale empirical study to investigate the quantitative relationship between the metrics reported in Table 2 ($\nu(\theta)$, $\mu(\theta)$, and nullity counts) and the adversarial robustness of neural networks. This will help validate whether models with higher nullity are indeed more susceptible to adversarial examples.

Data availability

The data used in this study are publicly available from the ImageNet dataset, a large-scale image database released for non-commercial research and educational purposes. ImageNet can be accessed at: https://www.image-net.org/. The dataset was originally introduced in: Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Received: 28 November 2025; Accepted: 13 February 2026
Published online: 21 February 2026

References

1. Krizhevsky, A., Sutskever, I. & Hinton, G. E. ImageNet classification with deep convolutional neural networks. *Adv. Neural Inf. Process. Syst.* (2012).
2. He, K., Zhang, X., Ren, S. & Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* https://doi.org/10.1109/cvpr.2016.90 (2016).
3. He, K., Gkioxari, G., Dollár, P. & Girshick, R. Mask R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision* (2017).
4. Howard, A. et al. Searching for Mobilenetv3. In *Proceedings of the IEEE International Conference on Computer Vision* https://doi.org/10.1109/iccv.2019.00140 (2019).
5. Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv:1810.04805 (2018).
6. Zuallaert, J. et al. SpliceRover: Interpretable convolutional neural networks for improved splice site prediction. *Bioinformatics* https://doi.org/10.1093/bioinformatics/bty497 (2018).
7. Ozbulak, U. et al. Mutate and observe: Utilizing deep neural networks to investigate the impact of mutations on translation initiation. *Bioinformatics* **39**(6), 338. https://doi.org/10.1093/bioinformatics/btad338. (2023).
8. Vandersmissen, B. et al. Indoor human activity recognition using high-dimensional sensors and deep neural networks. *Neural Comput. Appl.* https://doi.org/10.1007/s00521-019-04408-1 (2019).
9. Goodfellow, I., Shlens, J. & Szegedy, C. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations* (2015).
10. Szegedy, C. et al. Intriguing properties of neural networks. In *International Conference on Learning Representations* (2014).
11. Chernikova, A., Oprea, A., Nita-Rotaru, C. & Kim, B. G. Are self-driving cars secure? Evasion attacks against deep neural networks for steering angle prediction. *IEEE Security Privacy Workshops* https://doi.org/10.1109/spw.2019.00033 (2019).
12. Ozbulak, U., Van Messem, A. & De Neve, W. Impact of adversarial examples on deep learning models for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention* https://doi.org/10.1007/978-3-030-32245-8_34 (2019).
13. Tramer, F., Carlini, N., Brendel, W. & Madry, A. On adaptive attacks to adversarial example defenses. *Adv. Neural Inf. Process. Syst.* (2020).
14. Ilyas, A. et al. Adversarial examples are not bugs, they are features. *Adv. Neural Inf. Process. Syst.* 125–136 (2019).
15. Jacobsen, J., Behrmann, J., Zemel, R. & Bethge, M. Excessive invariance causes adversarial vulnerability. In *International Conference on Learning Representations* (2019).
16. Li, K., Zhang, T. & Malik, J. Approximate feature collisions in neural nets. *Adv. Neural Inf. Process. Syst.* (2019).
17. Wang, H., Wang, G., Li, Y., Zhang, D. & Lin, L. Transferable, controllable, and inconspicuous adversarial attacks on person re-identification with deep mis-ranking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* https://doi.org/10.1109/cvpr42600.2020.00042 (2020).
18. Kurakin, A., Goodfellow, I. & Bengio, S. Adversarial examples in the physical world. In *Workshop Track, International Conference on Learning Representations* (2016).
19. Sabour, S., Cao, Y., Faghri, F. & Fleet, D. J. Adversarial manipulation of deep representations. In *International Conference on Learning Representations* (2016).
20. Subramanya, A., Pillai, V. & Pirsiavash, H. Fooling network interpretation in image classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* https://doi.org/10.1109/iccv.2019.00211 (2019).
21. Tramèr, F., Behrmann, J., Carlini, N., Papernot, N. & Jörn-Henrik J. Fundamental tradeoffs between invariance and sensitivity to adversarial perturbations. In *International Conference on Machine Learning*, 9561–9571 (PMLR, 2020).
22. Hosseini, H., Kannan, S. & Poovendran, R. Are Odds Really Odd? Bypassing Statistical Detection Of Adversarial Examples. CoRR, arXiv:1907.12138 (2019).

23. Chaturvedi, A. & Garain, U. Mimic and fool: A task-agnostic adversarial attack. *IEEE Trans. Neural Netw. Learn. Syst.* https://doi.org/10.1109/tnnls.2020.2984972 (2020).
24. Glorot, X., Bordes, A. & Bengio, Y. Deep sparse rectifier neural networks. In *International Conference on Artificial Intelligence and Statistics* (2011).
25. Hein, M., Andriushchenko, M. & Bitterwolf, J. Why ReLU networks yield high-confidence predictions far away from the training data and how to mitigate the problem. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* https://doi.org/10.1109/cvpr.2019.00013 (2019).
26. Puthawala, M., Kothari, K., Lassas, M., Dokmanić, I. & de Hoop, M. Globally injective ReLU networks. *J. Mach. Learn. Res.* **23**(105), 1–55 (2022).
27. Wang, T. et al. ResLNet: Deep residual LSTM network with longer input for action recognition. *Front. Comp. Sci.* **16**(6), 166334. https://doi.org/10.1007/s11704-021-0236-9 (2022).
28. Chen, Y., Li, H., Song, Y. & Zhu, X. Recoding hybrid stochastic numbers for preventing bit width accumulation and fault tolerance. *IEEE Trans. Circuits Syst. I Regul. Pap.* https://doi.org/10.1109/TCSI.2024.3492054 (2024).
29. Hao, R. & Yang, X. Multiple-output quantile regression neural network. *Stat. Comput.* **34**(2), 89. https://doi.org/10.1007/s11222-024-10408-6 (2024).
30. Jin, J., Zhu, J., Zhao, L. & Chen, L. A fixed-time convergent and noise-tolerant zeroing neural network for online solution of time-varying matrix inversion. *Appl. Soft Comput.* **130**, 109691. https://doi.org/10.1016/j.asoc.2022.109691 (2022).
31. Zhou, Z. et al. Hybrid of neural network and physics-based estimator for vehicle longitudinal dynamics modeling using limited driving data. *IEEE Trans. Intell. Transp. Syst.* https://doi.org/10.1109/TITS.2025.3585346 (2025).
32. Shuo Zhang, L., Liu, C. W., Zhang, X. & Ma, R. Multistability and global attractivity for fractional-order spiking neural networks. *Appl. Math. Comput.* **508**, 129617. https://doi.org/10.1016/j.amc.2025.129617 (2026).
33. Yuan, W. et al. Transformer in reinforcement learning for decision-making: A survey. *Front. Inf. Technol. Electron. Eng.* **25**(6), 763–790. https://doi.org/10.1631/FITEE.2300548 (2024).
34. Cao, X. et al. Analytical survey of learning with low-resource data: From analysis to investigation. *ACM Comput. Surv.* **58**(6), 1–47. https://doi.org/10.1145/3773075 (2025).
35. LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE* https://doi.org/10.1109/9780470544976.ch9 (1998).
36. Vaswani, A. et al. Attention is all you need. In *Advances in Neural Information Processing Systems* (2017).
37. Hendrycks, D. & Gimpel, K. Gaussian Error Linear Units (GELUs). CoRR, arXiv:1606.08415 (2016).
38. Touvron, H. et al. Training Data-efficient Image Transformers & Distillation Through Attention. CoRR, arXiv:2012.12877 (2020).
39. Dosovitskiy, A. et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations* (2021).
40. Iandola, F. N. et al. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 MB model size. CoRR, arXiv:1602.07360 (2016).
41. Simonyan, K. & Zisserman, A. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations* (2015).
42. Huang, G., Liu, Z., Van Der Maaten, L. & Weinberger, K. Q. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017).
43. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. & Wojna, Z. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* https://doi.org/10.1109/cvpr.2016.308 (2016).
44. Ronneberger, O., Fischer, P. & Brox, T. U-Net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention* (eds Navab, N.et al.) https://doi.org/10.1007/978-3-319-24574-4_28 (2015).
45. Long, J., Shelhamer, E. & Darrell, T. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3431–3440 https://doi.org/10.1109/cvpr.2015.7298965 (2015).
46. Zhao, H., Shi, J., Qi, X., Wang, X. & Jia, J. Pyramid scene parsing network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* https://doi.org/10.1109/cvpr.2017.660 (2017).
47. Ren, S., He, K., Girshick, R. & Sun, J. Faster R-CNN: Towards real-time object detection with region proposal networks. *Adv. Neural Inf. Process. Syst.* (2015).
48. Lin, T-Y., Goyal, P., Girshick, R., He, K. & Dollár, P. Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision* https://doi.org/10.1109/iccv.2017.324 (2017).
49. Madry, A., Makelov, A., Schmidt, L., Tsipras, D. & Vladu, A. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations* (2018).
50. Wang, H. et al. Score-cam: Score-weighted visual explanations for convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 24–25 https://doi.org/10.1109/cvprw50498.2020.00020 (2020).
51. Selvaraju, R. R. et al. Grad-Cam: Why did you say that? Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016).
52. Sundararajan, M., Taly, A. & Yan, Q. Axiomatic attribution for deep networks. In *International Conference on Machine Learning* (2017).
53. Wang, Z., Bovik, A. C., Sheikh, H. R. & Simoncelli, E. P. Image quality assessment: From error visibility to structural similarity. *IEEE Trans. Image Process.* **13**(4), 600–612. https://doi.org/10.1109/tip.2003.819861 (2004).
54. Zhang, R., Isola, P., Efros, A. A., Shechtman, E. & Wang, O. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 586–595. https://doi.org/10.1109/cvpr.2018.00068 (2018).
55. Munier, N., Soubies, E. & Weiss, P. Jackpot: Approximating uncertainty domains with adversarial manifolds. *J. Mach. Learn. Res.* **26**(251), 1–41 (2025).
56. Li, X. & Short, K. M. Null space properties of neural networks with applications to image steganography. *Mathematics*, **13**(21) ISSN 2227–7390. https://doi.org/10.3390/math13213394 (2025).
57. Heo, B., Lee, M., Yun, S. & Choi, J. Y. Knowledge distillation with adversarial samples supporting decision boundary. In *Proceedings of the AAAI Conference on Artificial Intelligence* https://doi.org/10.1609/aaai.v33i01.33013771 (2019).
58. Goldblum, M., Fowl, L., Feizi, S. & Goldstein, T. Adversarially Robust Distillation. In *Proceedings of the AAAI Conference on Artificial Intelligence* https://doi.org/10.1609/aaai.v34i04.5816 (2020).

Author contributions

UO and MG conceived the study, designed the overall research methodology, and coordinated the project workflow. UO conducted the experiments, curated the datasets, and wrote the initial draft in close collaboration with MG. JV and WDN provided guidance on computational design choices, advised on experimental setup, and reviewed intermediate results to ensure robustness. AVM, SR, and MG developed and refined the mathematical formulations underlying the proposed approach. JV, WDN, and AVM critically reviewed, edited, and refined the manuscript for clarity and technical accuracy. All authors contributed to discussions throughout the project,

read, and approved the final version of the manuscript.

Declarations

Ethical approval

Not applicable. This study did not involve human participants, animal subjects, or data requiring ethics approval.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1038/s41598-026-40605-4.

Correspondence and requests for materials should be addressed to M.G.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.