# Computing Linear Restrictions of Neural Networks

**Matthew Sotoudeh** and Aditya V. Thakur
masotoudeh@ucdavis.edu, avthakur@ucdavis.edu
Davis Automated Reasoning Group

Conference on Neural Information Processing Systems, 2019

UCDAVIS
COMPUTER SCIENCE

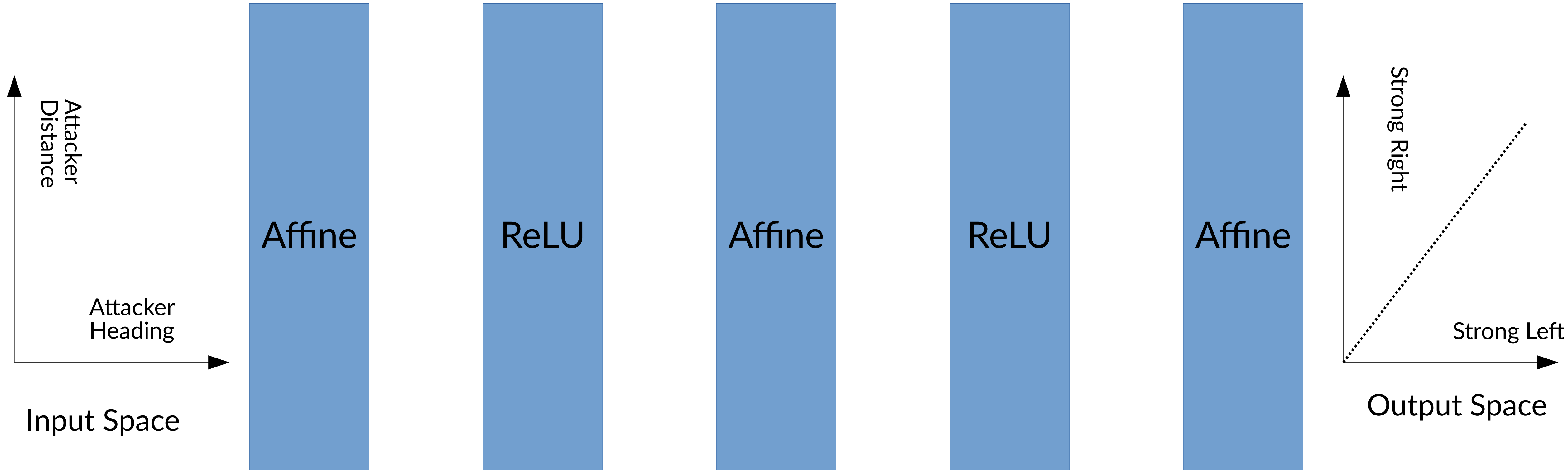
Davis
Automated
Reasoning
Group

# **Our Work:** A Technique for Examining Trained Neural Networks

Specifically, computing *succinct representation of the network restricted to a line.*

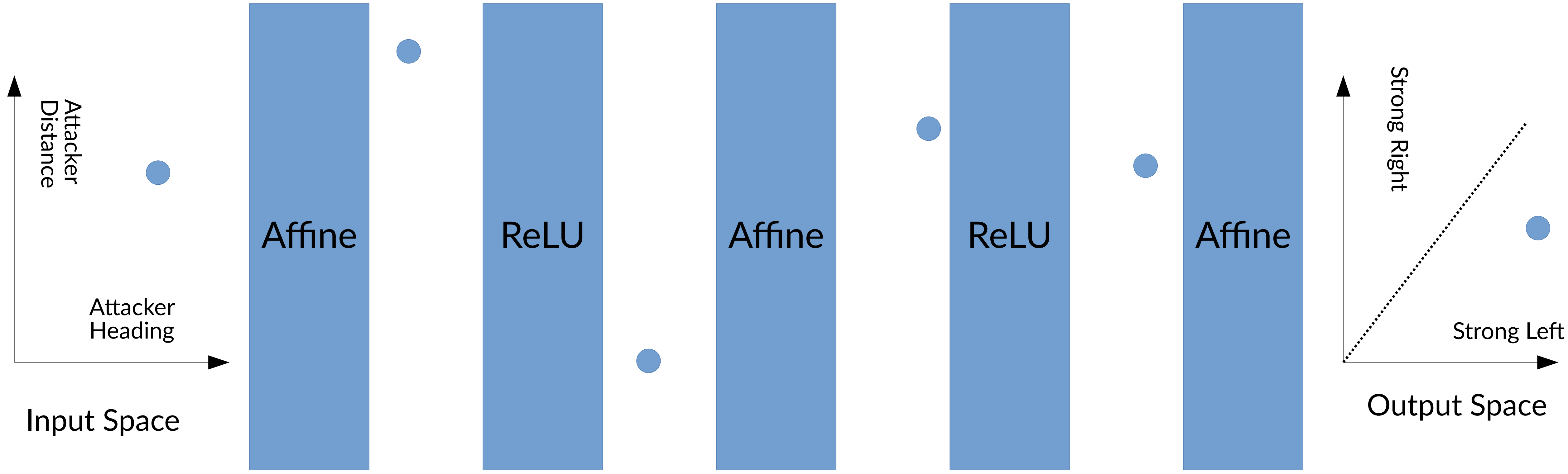
# Overview: Neural Networks

Neural Networks: sequential composition of other functions. Can transform individual points through each layer to find the output of the network.



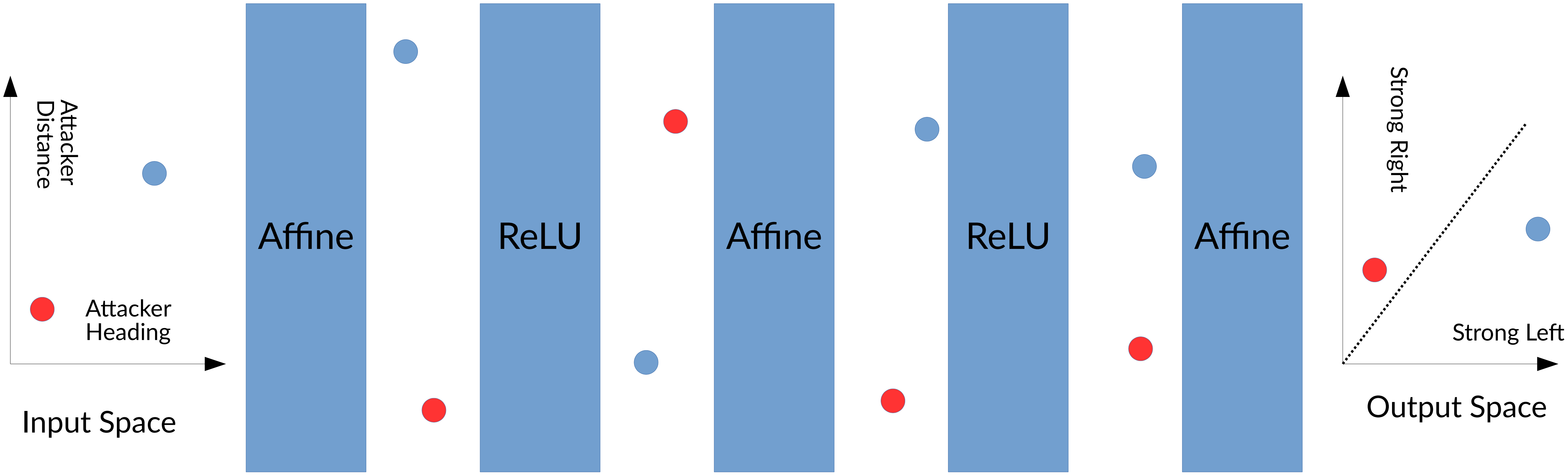
# Overview: Neural Networks

Neural Networks: sequential composition of other functions. Can transform individual points through each layer to find the output of the network.



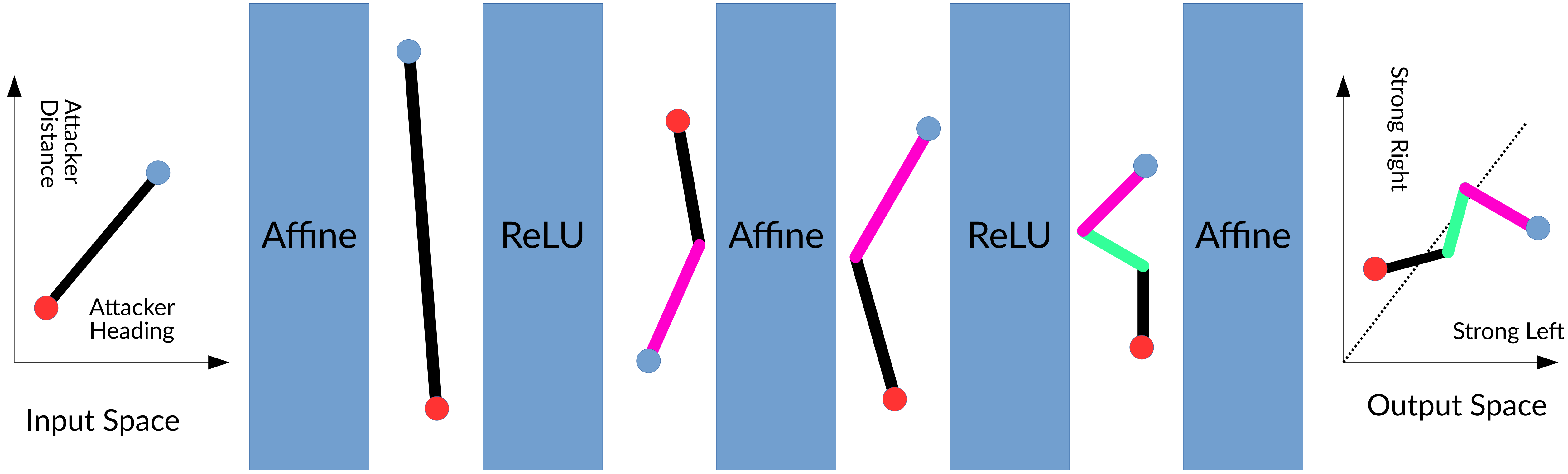
# Overview: Neural Networks

Neural Networks: sequential composition of other functions. Can transform individual points through each layer to find the output of the network.



# Overview: Neural Networks

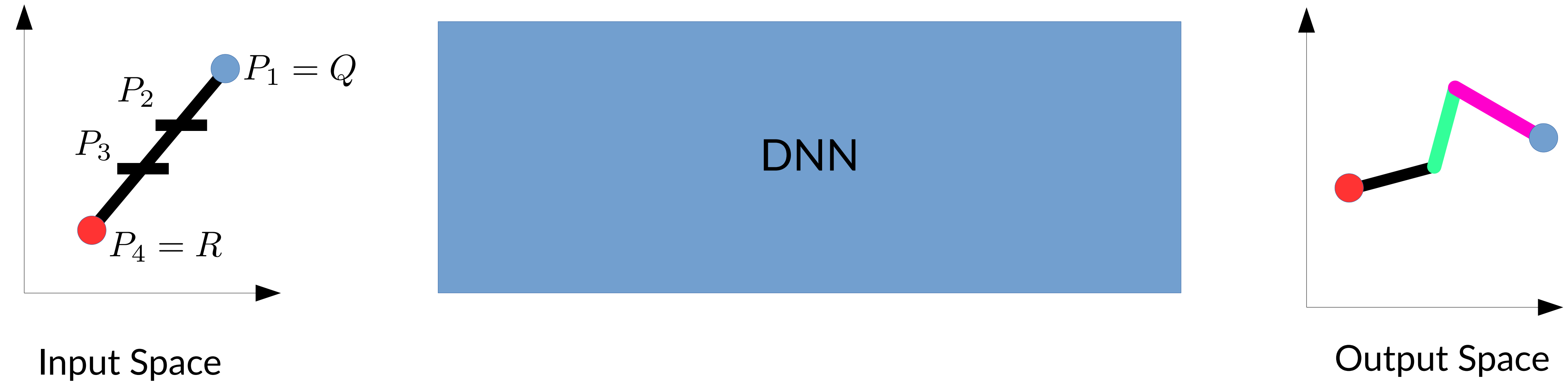
Neural Networks: sequential composition of other functions. Can transform individual points through each layer to find the output of the network.




However, when analyzing the network we would like to understand its behavior over infinitely-many points, eg. a line.

# ExactLine Formal Definition

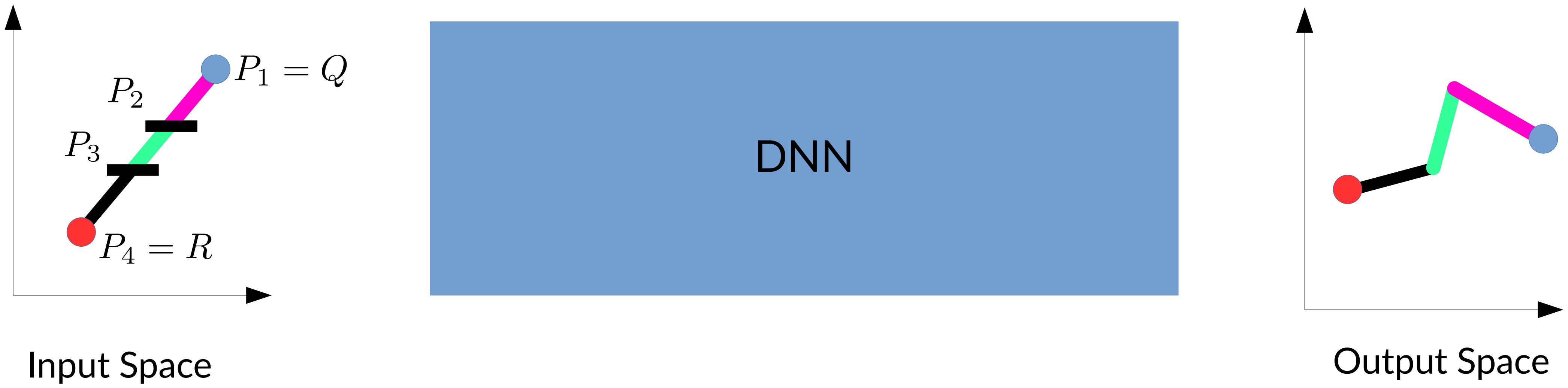
**Definition 1.** *Given a function $f : A \to B$ and line segment $\overline{QR} \subseteq A$, a tuple $(P_1, P_2, P_3, \ldots, P_n)$ is a linear partitioning of $f_{\restriction \overline{QR}}$, denoted $\mathcal{P}(f_{\restriction \overline{QR}})$ and referred to as "*ExactLine *of $f$ over $\overline{QR}$," if:*



# ExactLine Formal Definition

**Definition 1.** *Given a function $f : A \to B$ and line segment $\overline{QR} \subseteq A$, a tuple $(P_1, P_2, P_3, \ldots, P_n)$ is a linear partitioning of $f_{\restriction \overline{QR}}$, denoted $\mathcal{P}(f_{\restriction \overline{QR}})$ and referred to as "*ExactLine *of $f$ over $\overline{QR}$," if:*

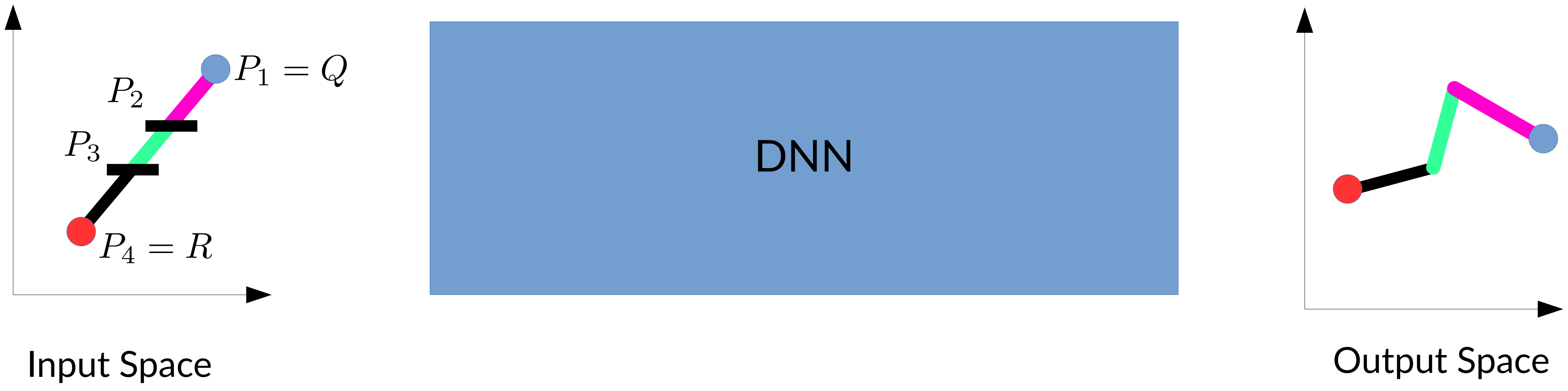
*1. $\{\overline{P_i P_{i+1}} \mid 1 \leq i < n\}$ partitions $\overline{QR}$ (except for overlap at endpoints).*



# ExactLine Formal Definition

**Definition 1.** *Given a function $f : A \to B$ and line segment $\overline{QR} \subseteq A$, a tuple $(P_1, P_2, P_3, \ldots, P_n)$ is a linear partitioning of $f_{\restriction \overline{QR}}$, denoted $\mathcal{P}(f_{\restriction \overline{QR}})$ and referred to as "*ExactLine *of $f$ over $\overline{QR}$," if:*

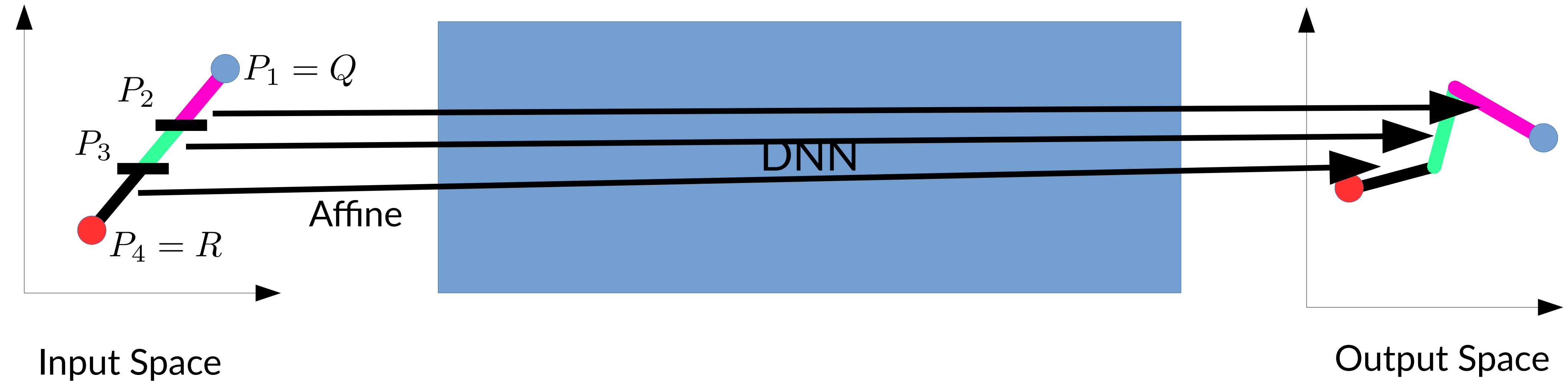
1. *$\{\overline{P_i P_{i+1}} \mid 1 \leq i < n\}$ partitions $\overline{QR}$ (except for overlap at endpoints).*
2. *$P_1 = Q$ and $P_n = R$.*



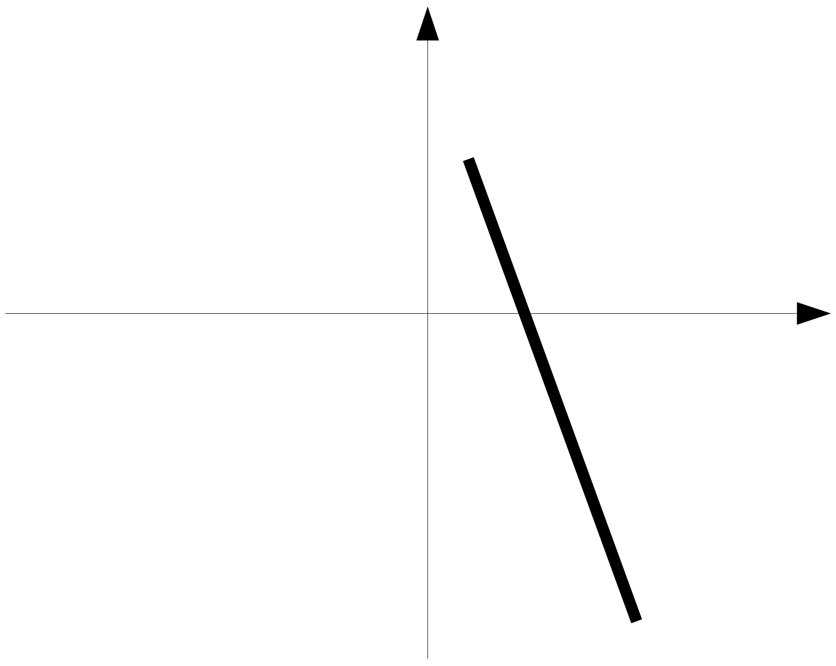
# ExactLine Formal Definition

**Definition 1.** *Given a function $f : A \to B$ and line segment $\overline{QR} \subseteq A$, a tuple $(P_1, P_2, P_3, \ldots, P_n)$ is a linear partitioning of $f_{\restriction \overline{QR}}$, denoted $\mathcal{P}(f_{\restriction \overline{QR}})$ and referred to as "*ExactLine *of $f$ over $\overline{QR}$," if:*

1. *$\{\overline{P_i P_{i+1}} \mid 1 \leq i < n\}$ partitions $\overline{QR}$ (except for overlap at endpoints).*
2. *$P_1 = Q$ and $P_n = R$.*
3. *For all $1 \leq i < n$, there exists an affine map $A_i$ such that $f(x) = A_i(x)$ for all $x \in \overline{P_i P_{i+1}}$.*

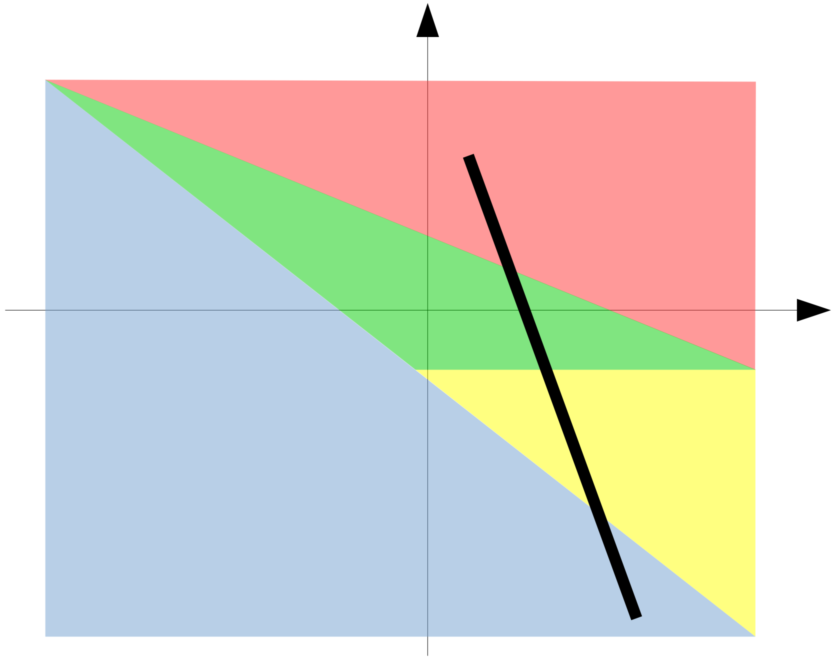


# Computing ExactLine: Single Layer




$$\mathcal{P}(f_{\restriction \overline{QR}})$$

# Computing ExactLine: Single Layer

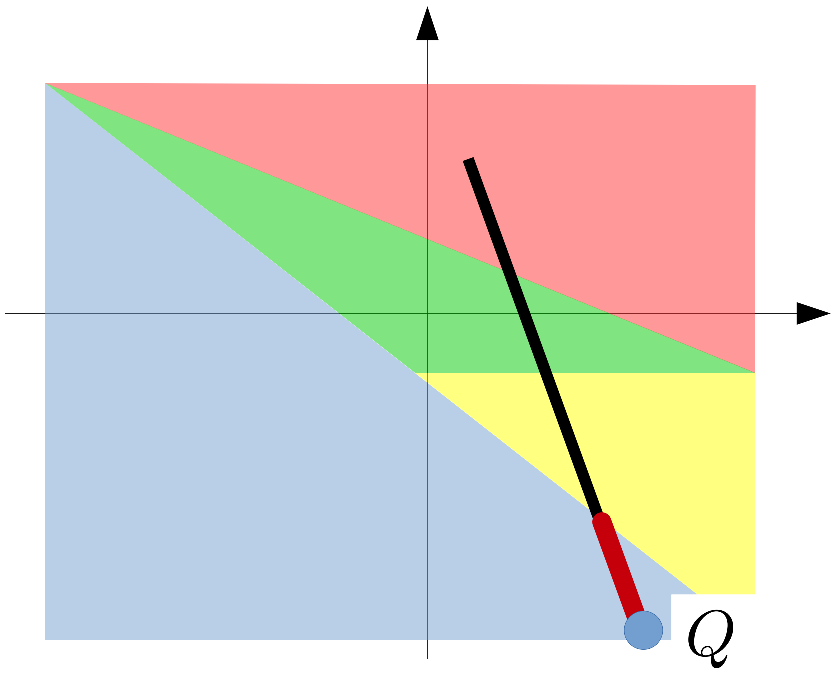


Input Space

$$\mathcal{P}(f_{\restriction \overline{QR}})$$

1. Partition input space according to PWL function.

# Computing ExactLine: Single Layer



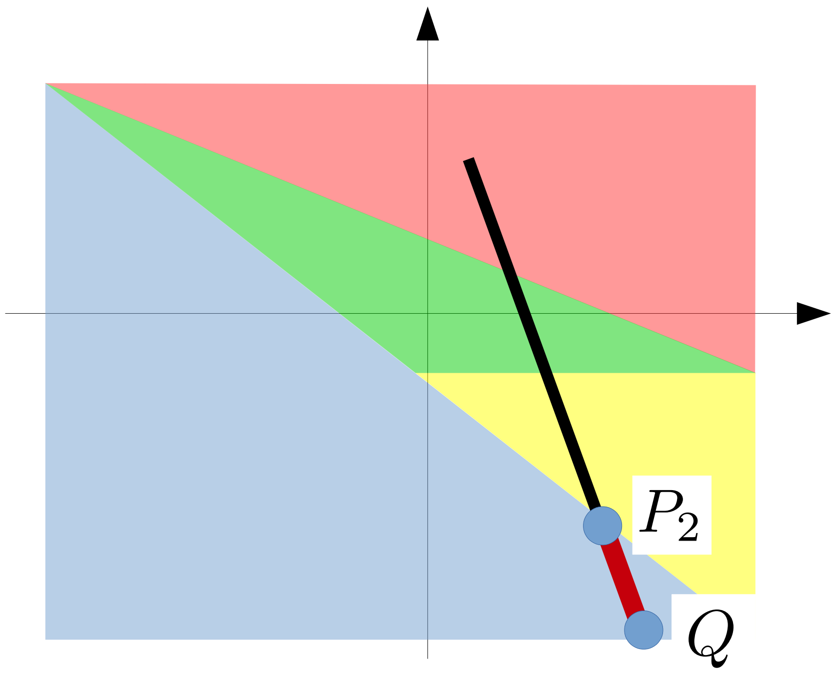

Input Space

1. Partition input space according to PWL function.

2. “Follow” line from an endpoint.

$$\mathcal{P}(f_{\restriction \overline{QR}}) = (Q,$$

# Computing ExactLine: Single Layer




Input Space

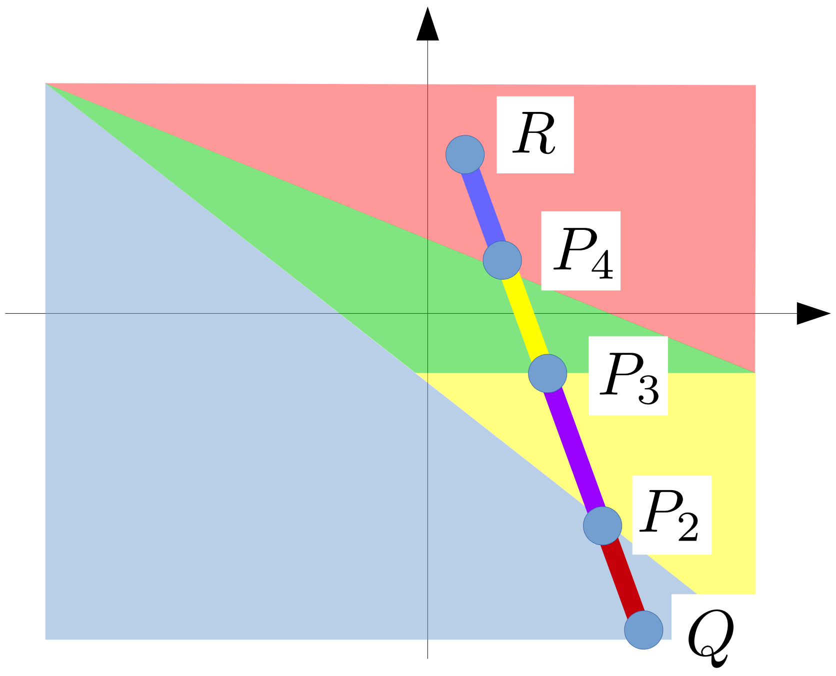
1. Partition input space according to PWL function.

2. “Follow” line from an endpoint.

3. When a PWL boundary reached, add an endpoint.

$$\mathcal{P}(f_{\restriction \overline{QR}}) = (Q, P_2,$$

# Computing ExactLine: Single Layer




$$\mathcal{P}(f_{\restriction \overline{QR}}) = (Q, P_2, P_3, P_4, R)$$

1. Partition input space according to PWL function.

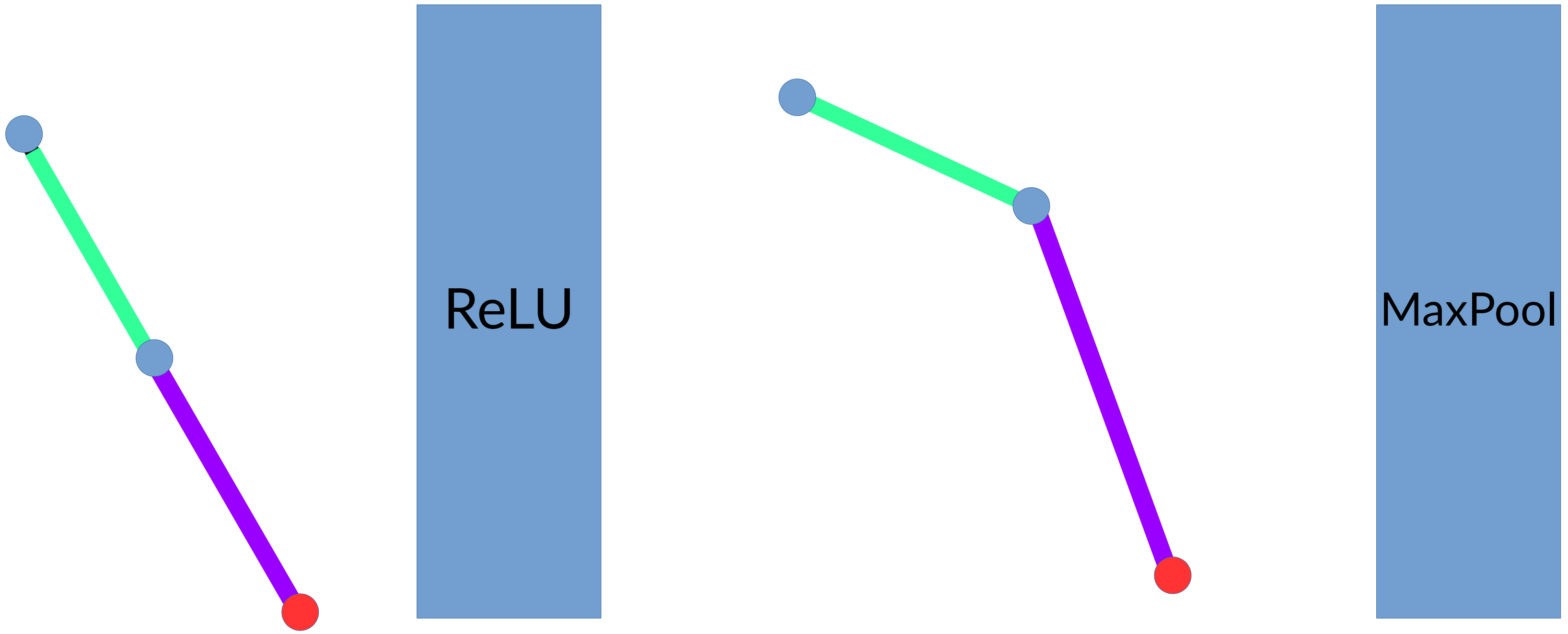
2. “Follow” line from an endpoint.

3. When a PWL boundary reached, add an endpoint.

4. Continue until last endpoint reached.

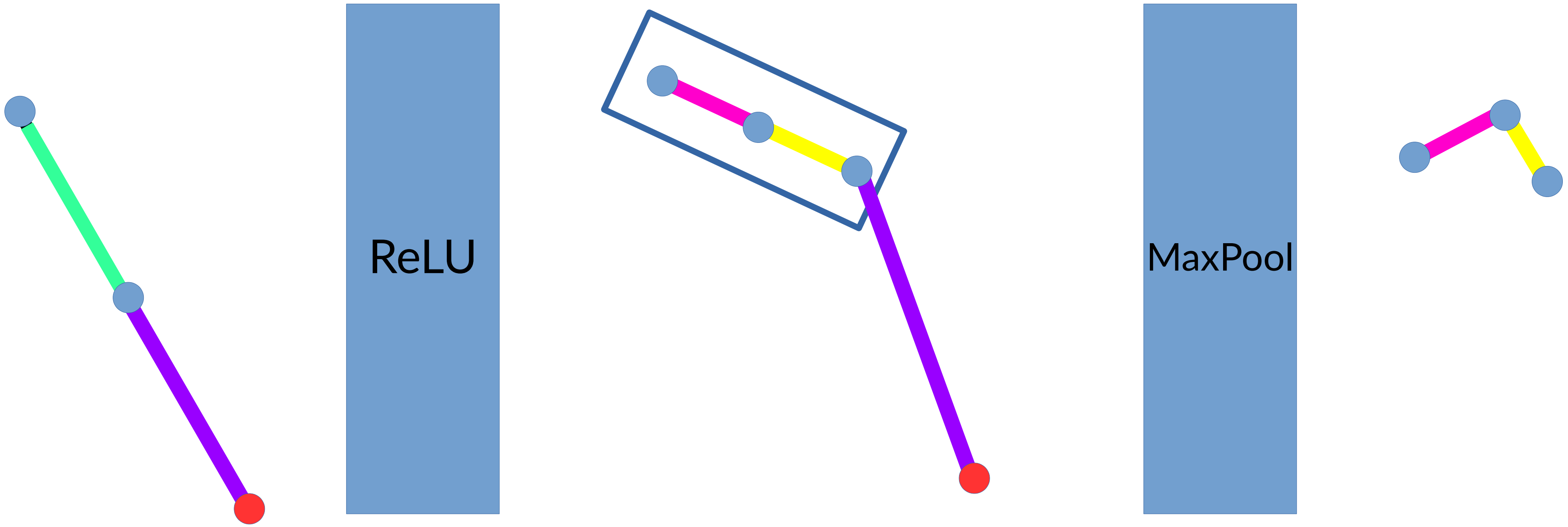
# Computing ExactLine: Multiple Layers

# Computing ExactLine: Multiple Layers




1. Transform by the first layer.

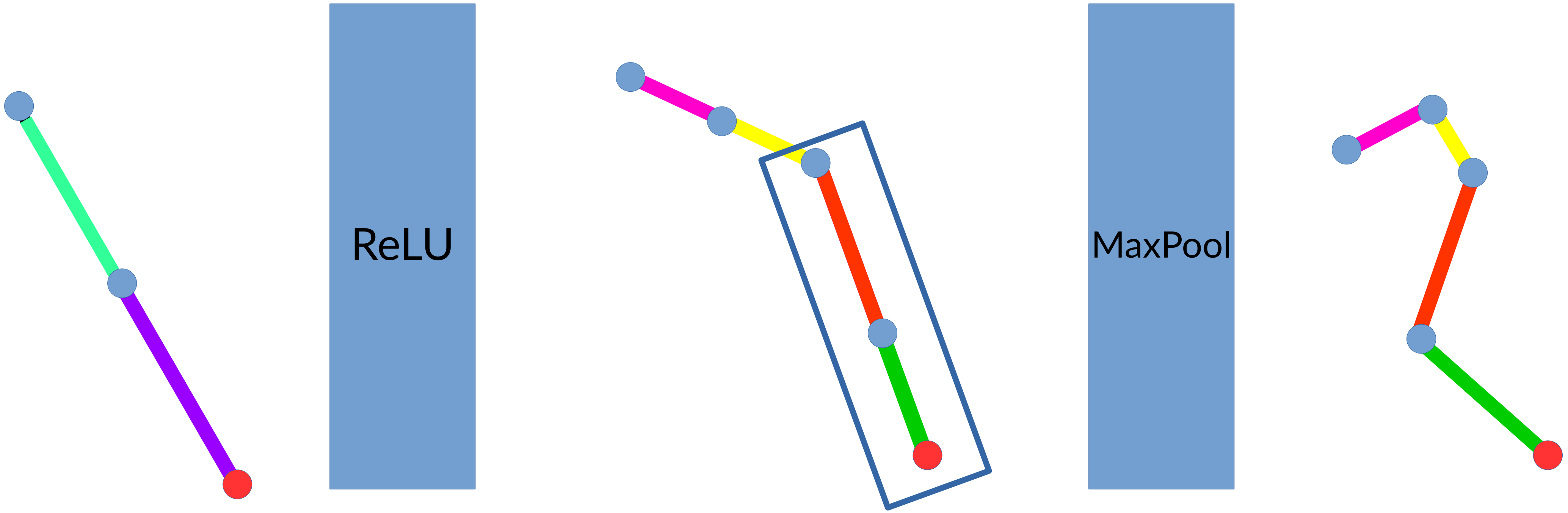
# Computing ExactLine: Multiple Layers




1. Transform by the first layer.

2. Transform intermediate-space segments by second layer.

# Computing ExactLine: Multiple Layers

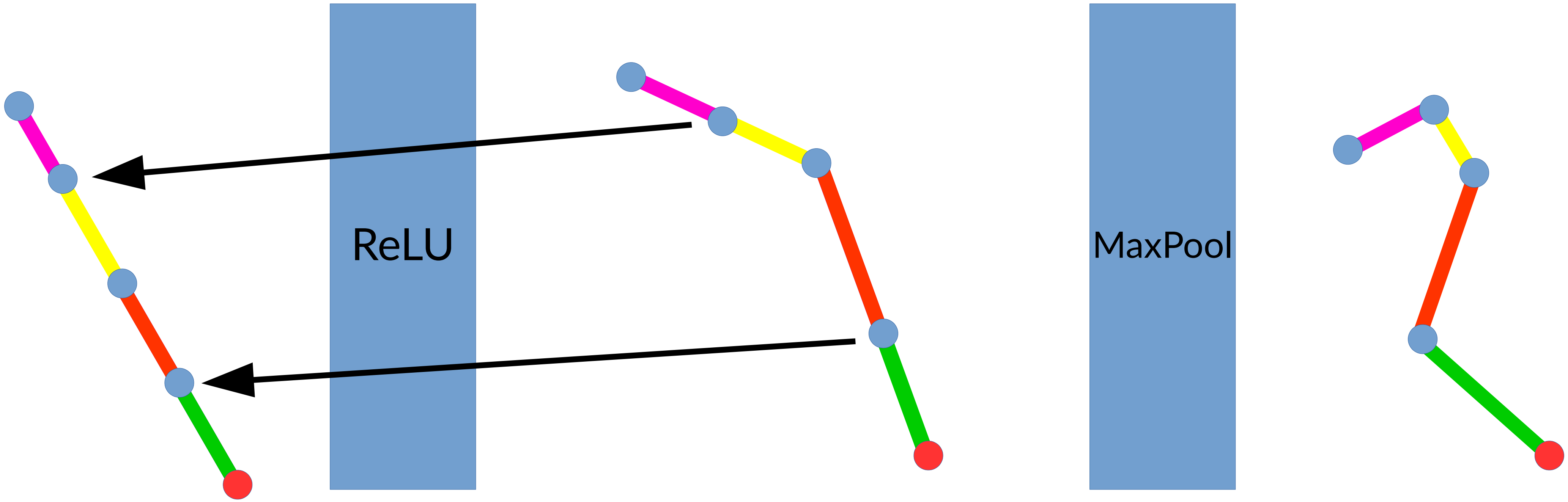



1. Transform by the first layer.

2. Transform intermediate-space segments by second layer.

# Computing ExactLine: Multiple Layers

3. Project the new endpoints (and partitions) back onto the input space.




1. Transform by the first layer.

2. Transform intermediate-space segments by second layer.

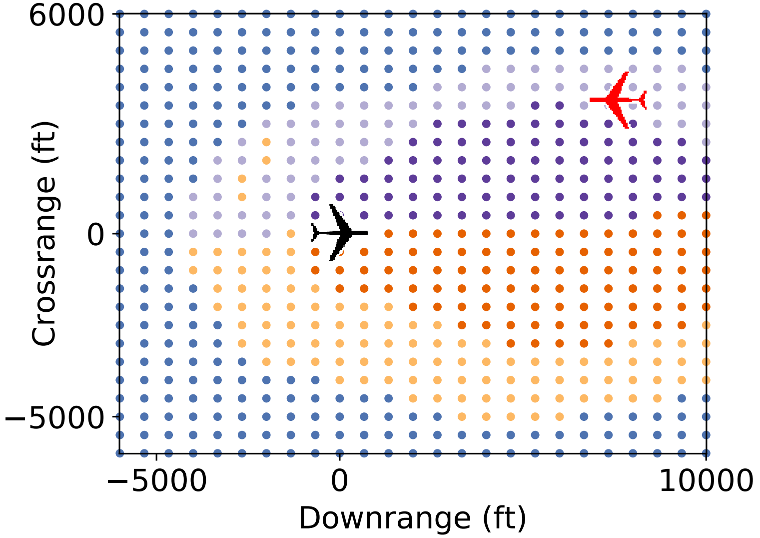
# Three Initial Applications

1. Understanding Decision Boundaries

# Visualizing ACAS Xu Decision Boundaries

ACAS Xu network:
Attacker Position (polar) → Advisory

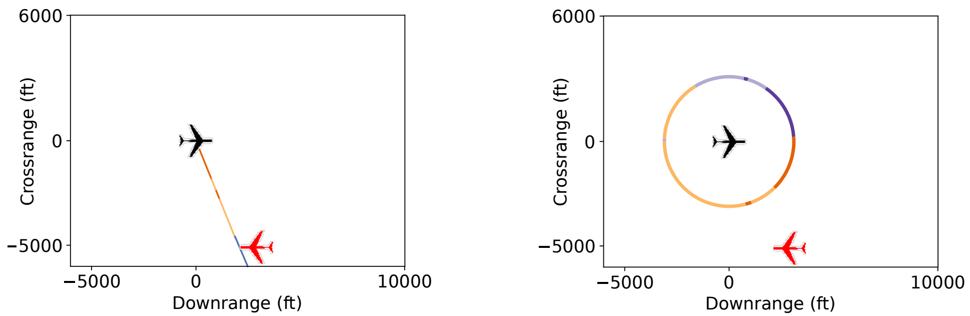
Prior work: sampling individual points.




Legend: Clear-of-Conflict Weak Right Strong Right Strong Left Weak Left

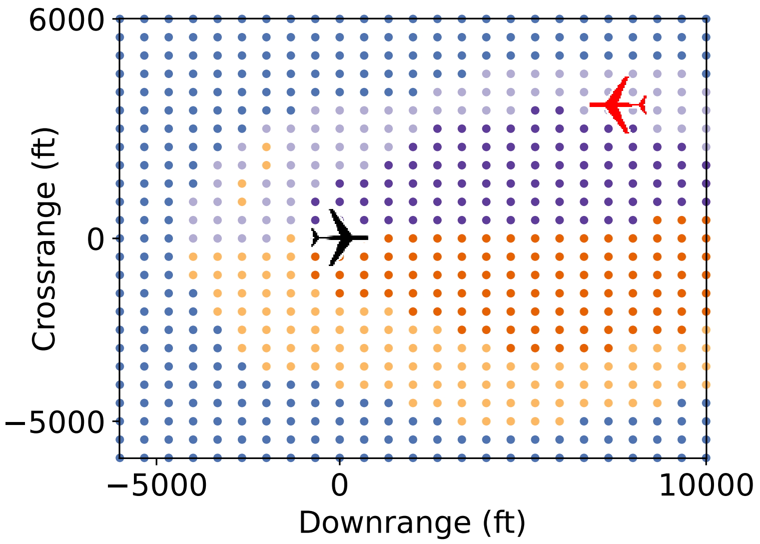
# Visualizing ACAS Xu Decision Boundaries

ACAS Xu network:
Attacker Position (polar) → Advisory




With ExactLine, we can *exactly* determine decision boundaries along a line segment.

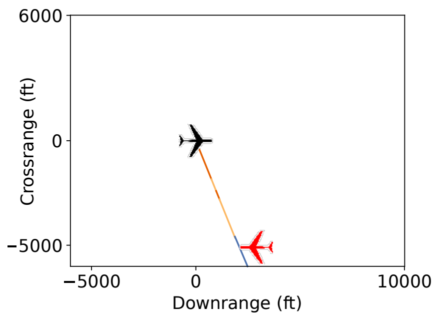
Prior work: sampling individual points.

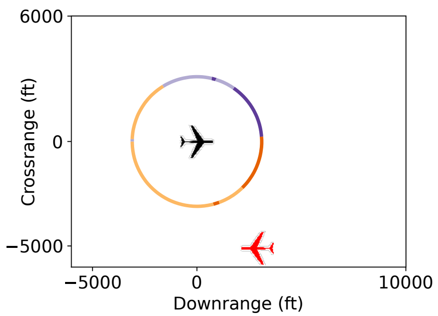



Legend: Clear-of-Conflict, Weak Right, Strong Right, Strong Left, Weak Left

# Visualizing ACAS Xu Decision Boundaries

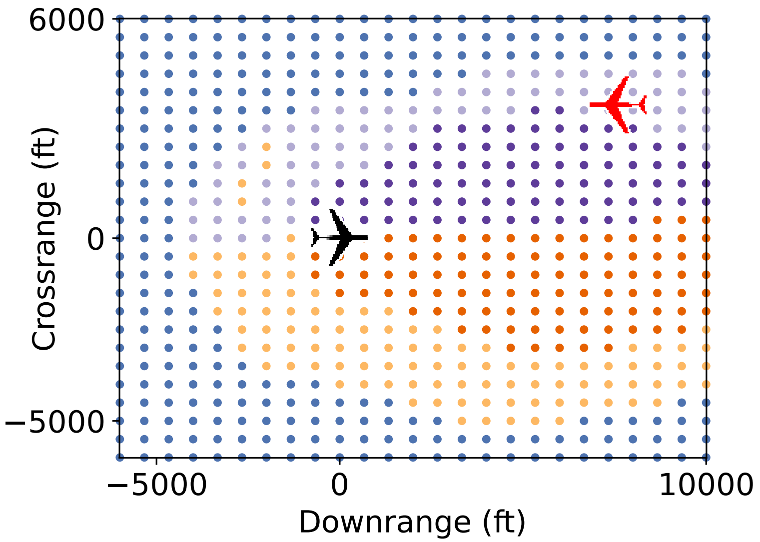
ACAS Xu network:
Attacker Position (polar) → Advisory

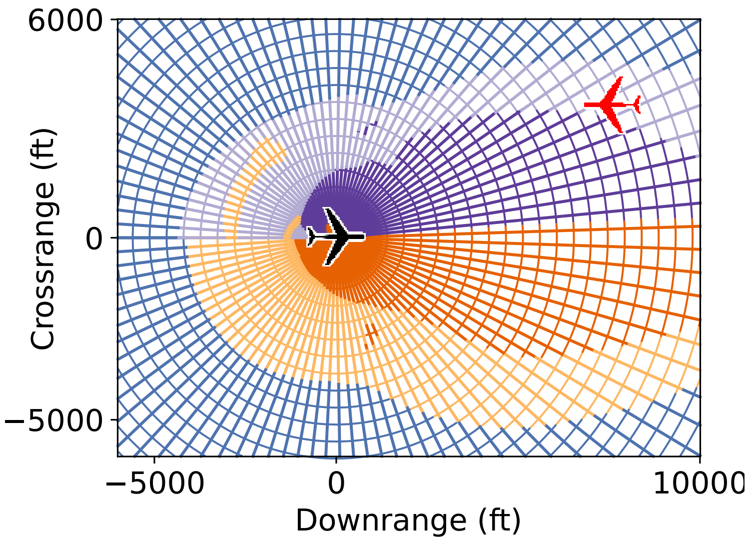



Prior work: sampling individual points.

With ExactLine, we can *exactly* determine decision boundaries along a line segment.



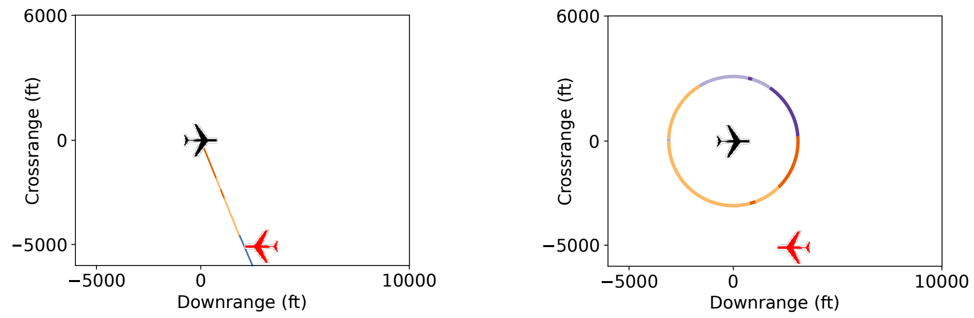




Overlapping multiple line segments creates a classification grid with much higher confidence than finite sampling.

Legend: Clear-of-Conflict | Weak Right | Strong Right | Strong Left | Weak Left

# Visualizing ACAS Xu Decision Boundaries

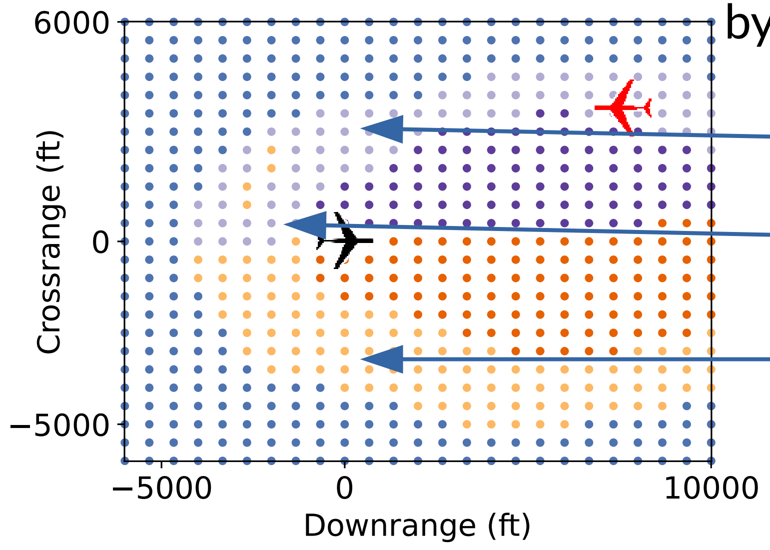
ACAS Xu network:
Attacker Position (polar) → Advisory

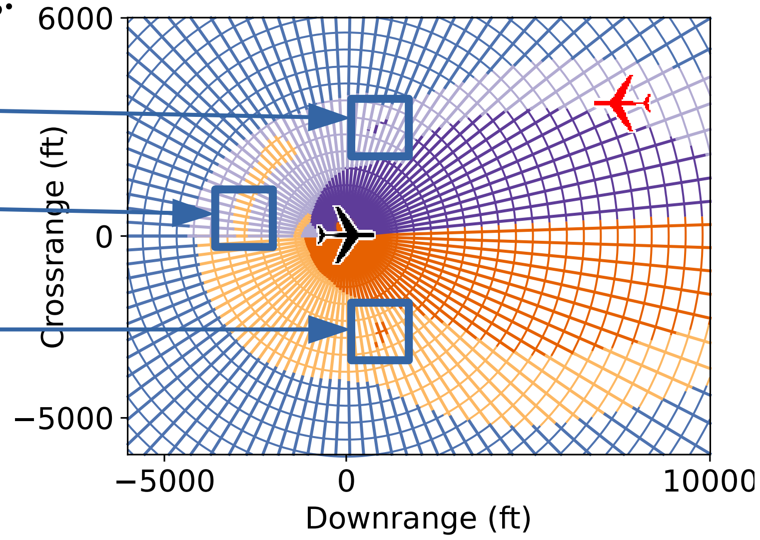



With ExactLine, we can *exactly* determine decision boundaries along a line segment.

Prior work: sampling individual points.

Important behavior missed by sampling.




Overlapping multiple line segments creates a classification grid with much higher confidence than finite sampling.

Legend: Clear-of-Conflict Weak Right Strong Right Strong Left Weak Left

# Three Initial Applications

1. Understanding Decision Boundaries
2. Exact Computation of Integrated Gradients Attribution Method

# Integrated Gradients

Popular DNN attribution method ("why did the network call this a fireboat?").

Relies on computing a path integral between a *baseline* and the image.

$$IG_i(x) = (x_i - x'_i) \times \int_{\alpha=0}^{1} \frac{\partial F(x' + \alpha \times (x - x'))}{\partial x_i} d\alpha$$

"Fireboat"

Black Baseline

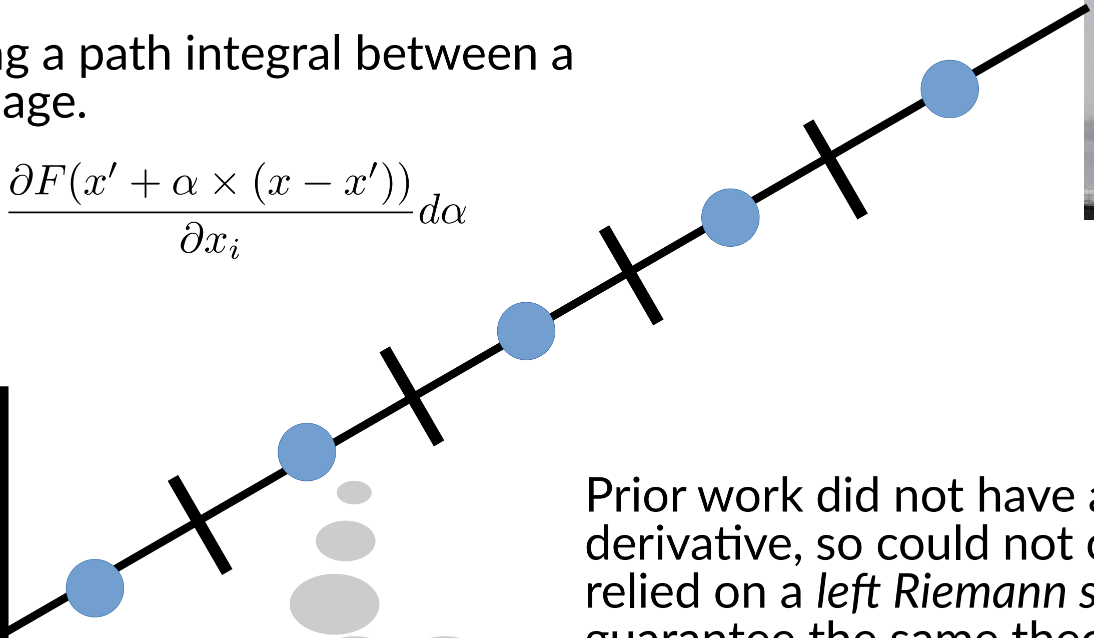
# Integrated Gradients

Popular DNN attribution method ("why did the network call this a fireboat?").

Relies on computing a path integral between a *baseline* and the image.

$$IG_i(x) = (x_i - x'_i) \times \int_{\alpha=0}^{1} \frac{\partial F(x' + \alpha \times (x - x'))}{\partial x_i} d\alpha$$

"Fireboat"



Black Baseline

Would be exact if gradient were constant within partitions!

Prior work did not have an analytic form for the partial derivative, so could not compute this integral. Instead, relied on a *left Riemann sum approximation* that does not guarantee the same theoretical properties.

$$\widetilde{IG}_i^m(x) = (x_i - x'_i) \times \sum_{0 \le k < m} \frac{\partial F(x' + \frac{k}{m} \times (x - x'))}{\partial x_i} \times \frac{1}{m}$$

# Integrated Gradients

Popular DNN attribution method ("why did the network call this a fireboat?").

Relies on computing a path integral between a *baseline* and the image.

$$IG_i(x) = (x_i - x_i') \times \int_{\alpha=0}^{1} \frac{\partial F(x' + \alpha \times (x - x'))}{\partial x_i} d\alpha$$

"Fireboat"

Black Baseline

Would be exact if gradient were constant within partitions!

Prior work did not have an analytic form for the partial derivative, so could not compute this integral. Instead, relied on a *left Riemann sum approxi*[...]*ation* that does not guarantee the [...]ame theo[...]erties.

Let's use ExactLine!

$$IG_i(x) \ldots \frac{\ldots(x - x'))}{\partial x_i} \times \frac{1}{m}$$

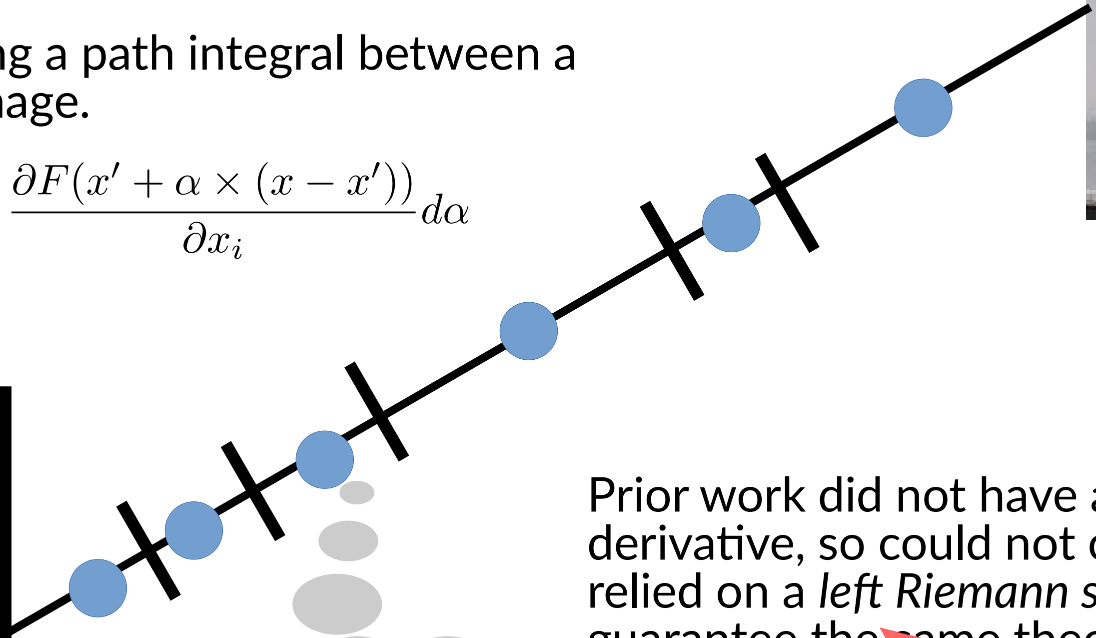
# Integrated Gradients

Popular DNN attribution method ("why did the network call this a fireboat?").

Relies on computing a path integral between a *baseline* and the image.

$$IG_i(x) = (x_i - x'_i) \times \int_{\alpha=0}^{1} \frac{\partial F(x' + \alpha \times (x - x'))}{\partial x_i} d\alpha$$

"Fireboat"



Black Baseline

ExactLine guarantees a partitioning with constant gradients!

Prior work did not have an analytic form for the partial derivative, so could not compute this integral. Instead, relied on a *left Riemann sum approxi*[...]*ation* that does not guarantee the [...]ame theo[...] [...]erties

Let's use ExactLine!

$IG_i\ (x) \ldots (x - x')) \ldots \partial x_i \times \frac{1}{m}$

# Integrated Gradients: Results

- How accurate is the prior best-practice approximation?
  - 25-45% error
- How many samples are needed to get to 5% error?
  - Usually about 100-300
- Do different sampling methods perform better/worse?
  - Trapezoidal rule is 20-40% more sample-efficient than left/right approximations.

# Three Initial Applications

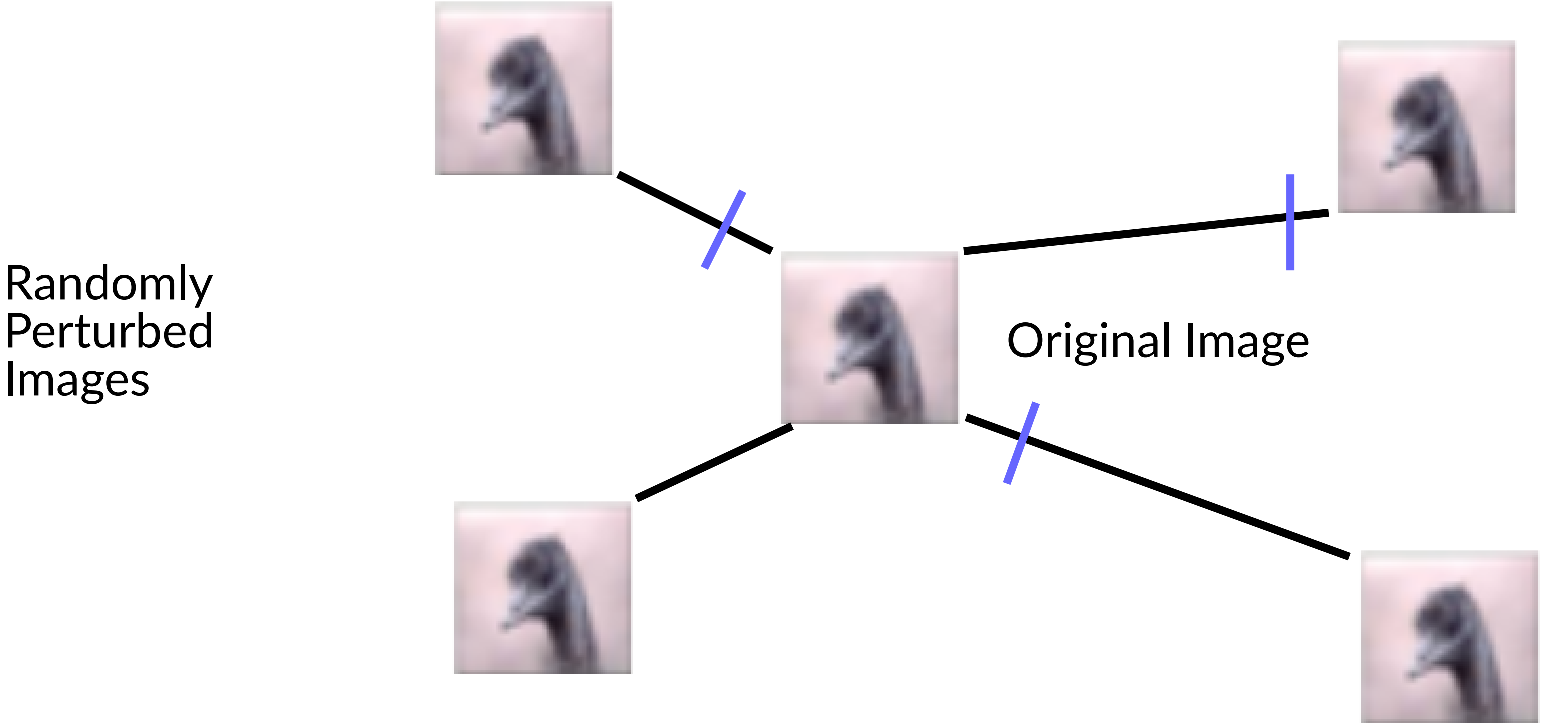
1. Understanding Decision Boundaries
2. Exact Computation of Integrated Gradients Attribution Method
3. Investigating Adversarial Examples, and Falsifying the Linearity Hypothesis

# Adversarial Examples and the Linear Explanation

- Adversarial examples: small perturbations cause big classification changes.
- Goodfellow *et al.* introduce influential "Linear Explanation"
  - **Linearity Assumption**: around 'natural' input images, the network behaves *linearly* (i.e., tangent plane at point matches output).
  - **Theoretical Claim**: classification boundaries of linear classifiers become closer with higher dimensionality.
  - **Conclusion**: adversarial examples are natural consequence of linearity hypothesis, so we need more non-linear neural networks.
- Theoretical discussion of claim, but (until now) underlying assumption untested.

# Investigating the Linearity Hypothesis

- Q1: Is the area around input points linear?
- A1: No!

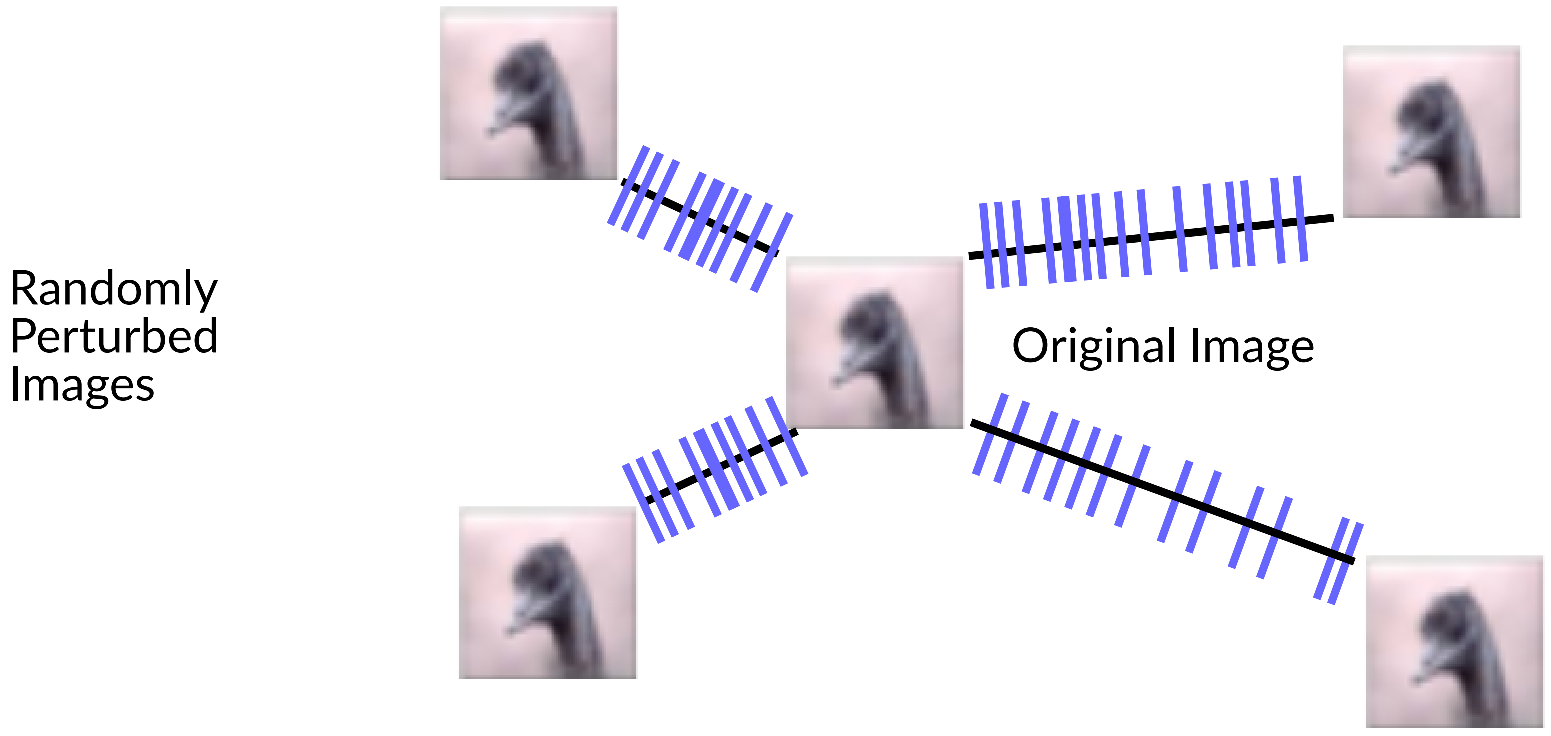



We will draw blue lines to delineates different linear partitions (i.e. show where non-linearities are introduced).

**Prediction:** network is “mostly-linear,” so should have few linear partitions.

# Investigating the Linearity Hypothesis

- Q1: Is the area around input points linear?
- A1: No!



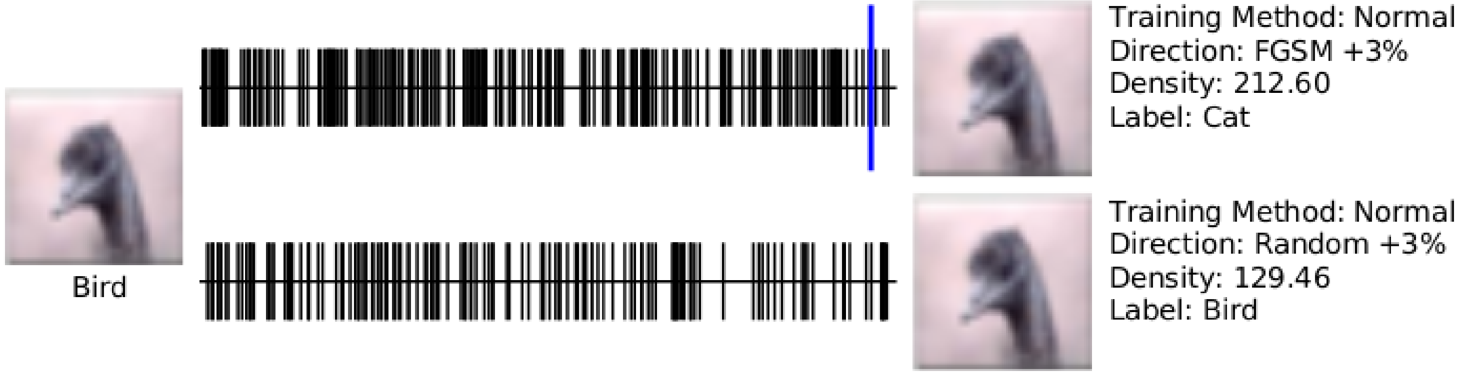

We will draw blue lines to delineates different linear partitions (i.e. show where non-linearities are introduced).

**Prediction:** network is “mostly-linear,” so should have few linear partitions.

**Reality:** network is extremely non-linear, with often thousands of different partitions.

# Investigating the Linearity Hypothesis

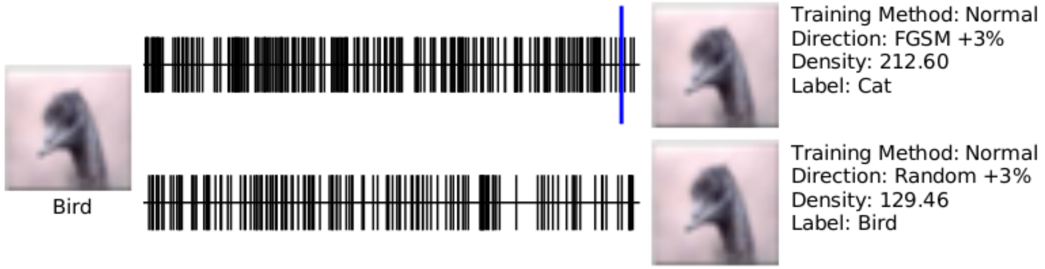
- Perhaps only the adversarial direction lies on the same linear partition.
- Q2: Are adversarial directions particularly linear?
- A2: No!




Adversarial perturbations in fact lie in a *more non-linear* direction than random perturbations.

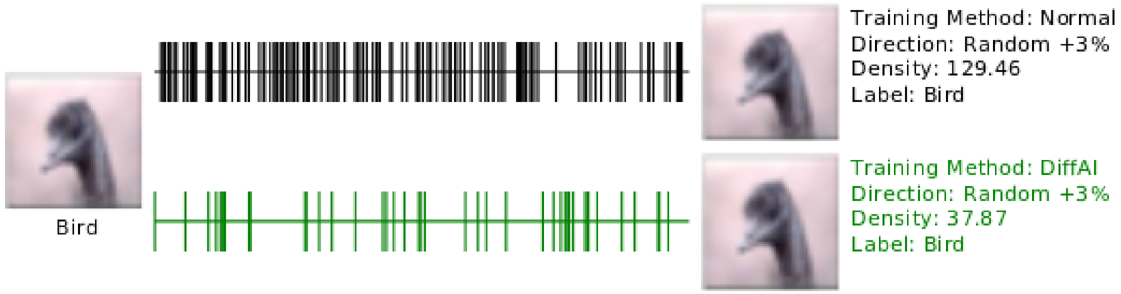
# Investigating the Linearity Hypothesis

- Perhaps the gradients in each partition are "relatively close" to that around the natural point.
- Q3: Are the gradients in each partition close to that of the natural point?
- A3: No!
- Experiment:
  - On each partition, find relative error between that partition's gradient and the gradient at the natural point.
  - Average the relative errors, weighted by width of partition.
  - Result: >250% relative error.



# Investigating the Linearity Hypothesis

- Q4: Are all models this non-linear?
- A4: Surprisingly, no! DiffAI- and PGD-trained models show less non-linearity.
- Interesting direction for future work.



# Linearity Hypothesis: Takeaways and Future Work

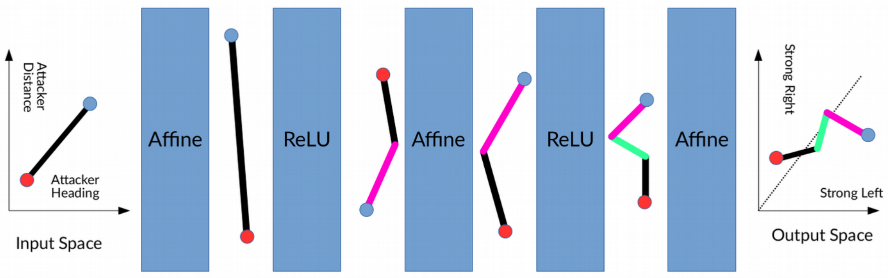
- Linearity Hypothesis (and surrogate assumptions) empirically falsified -> Linear Explanation rejected.
  - Need to find new explanations for adversarial examples and *tools (like ExactLine!) to empirically verify/falsify them.*
- Eg., “A Boundary Tilting Perspective on the Phenomenon of Adversarial Examples” (Tanay and Griffin):

# Conclusion

- ExactLine efficiently and precisely decomposes a neural network into affine partitions.
  - When restricted to a line in the input domain.
- Wide variety of uses, we tried three:
  - Decision boundary understanding.
  - Exact computation of IG
  - Investigating adversarial examples

## Overview: Neural Networks

Neural Networks: sequential composition of other functions. Can transform individual points through each layer to find the output of the network.

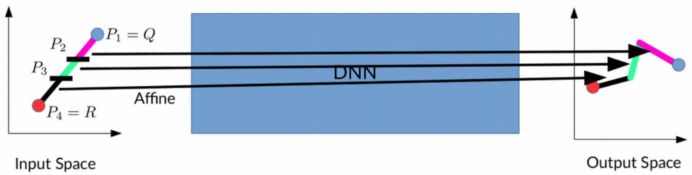



However, when analyzing the network we would like to understand its behavior over infinitely-many points, eg. a line.

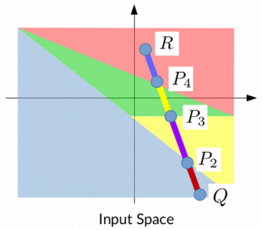
## ExactLine Formal Definition

**Definition 1.** *Given a function $f : A \to B$ and line segment $\overline{QR} \subseteq A$, a tuple $(P_1, P_2, P_3, \ldots, P_n)$ is a linear partitioning of $f_{\restriction \overline{QR}}$, denoted $\mathcal{P}(f_{\restriction \overline{QR}})$ and referred to as "*ExactLine *of $f$ over $\overline{QR}$," if:*

1. *$\{\overline{P_i P_{i+1}} \mid 1 \leq i < n\}$ partitions $\overline{QR}$ (except for overlap at endpoints).*
2. *$P_1 = Q$ and $P_n = R$.*
3. *For all $1 \leq i < n$, there exists an affine map $A_i$ such that $f(x) = A_i(x)$ for all $x \in \overline{P_i P_{i+1}}$.*




## Computing ExactLine: Single Layer

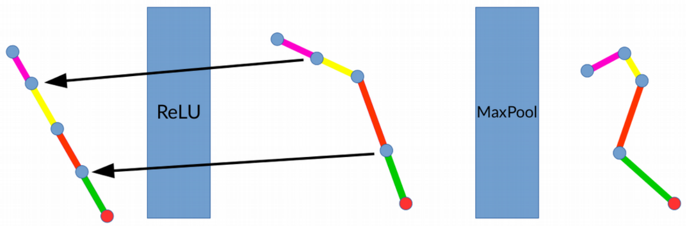



1. Partition input space according to PWL function.
2. "Follow" line from an endpoint.
3. When a PWL boundary reached, add an endpoint.
4. Continue until last endpoint reached.

$$\mathcal{P}(f_{\restriction \overline{QR}}) = (Q, P_2, P_3, P_4, R)$$

## Computing ExactLine: Multiple Layers

3. Project the new endpoints (and partitions) back onto the input space.

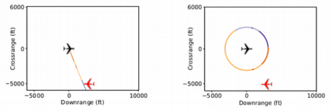



1. Transform by the first layer.

2. Transform intermediate-space segments by second layer.

## Visualizing ACAS Xu Decision Boundaries

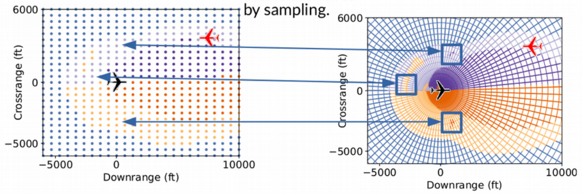
ACAS Xu network:
Attacker Position (polar) → Advisory



With ExactLine, we can *exactly* determine decision boundaries along a line segment.

Prior work: sampling individual points.

Important behavior missed by sampling.




Overlapping multiple line segments creates a classification grid with much higher confidence than finite sampling.

Legend: Clear-of-Conflict, Weak Right, Strong Right, Strong Left, Weak Left

## Integrated Gradients

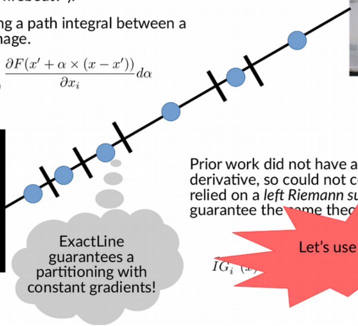
Popular DNN attribution method ("why did the network call this a fireboat?").

Relies on computing a path integral between a *baseline* and the image.

$$IG_i(x) = (x_i - x'_i) \times \int_{\alpha=0}^{1} \frac{\partial F(x' + \alpha \times (x - x'))}{\partial x_i} d\alpha$$

"Fireboat"

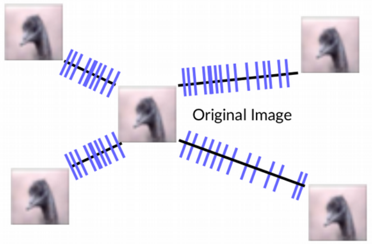
Black Baseline




Prior work did not have an analytic form for the partial derivative, so could not compute this integral. Instead, relied on a *left Riemann sum approximation* that does not guarantee the same theoretical properties.

## Investigating the Linearity Hypothesis

- Q1: Is the area around input points linear?
- A1: No!




We will draw blue lines to delineates different linear partitions (i.e. show where non-linearities are introduced).

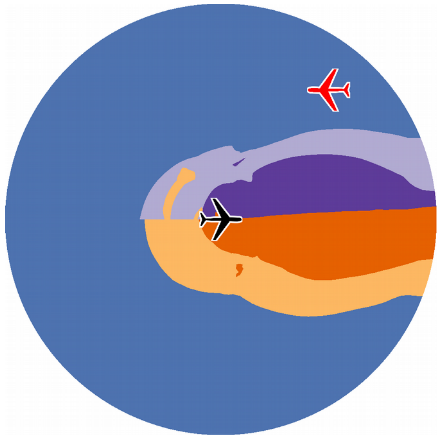
**Prediction:** network is "mostly-linear," so should have few linear partitions.

**Reality:** network is extremely non-linear, with often thousands of different partitions.

## Questions?

# ExactLine Generalization

- In a preprint, we extend ExactLine to 2-dimensional regions.
- Can understand entire decision boundary.
- Can do bounded model checking.
- Can *patch* neural networks.



# Comparison to Other 'White Box' Techniques

**Slow (NP-Hard), But Precise**

- ReluPlex
  - Decision procedure (Y/N)
  - "Is there anyone for whom the model recommends 'no approval?'"
- Linear partitioners
  - "What are *all* the people for whom the model recommends 'no approval?'"

**Fast, But Imprecise**

- ERAN
  - Decision procedure (Y/N)
  - Over-approximation (some Y are N!)
  - "Is it *possible* that there is someone for whom the model recommends 'no approval?'"
- Sampling
  - "What does the model recommend for these N people?"

# Exploring a New Dimension of Analysis

**Prior work:** Speed versus Precision

**ExactLine:** Dimensionality versus (Speed & Precise)