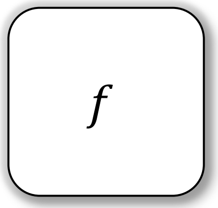
Training-Set Influence Analysis and Estimation

Zayd Hammoudeh
University of Oregon

October 14, 2022

A Running Example...



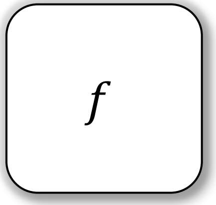
Untrained
Classifier

A Running Example...

Binary Classification

Cat

Dog



Untrained
Classifier

A Running Example...

Binary Classification

Cat

Dog

Input

f

Untrained Classifier

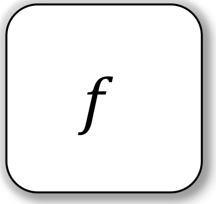
Output

Correct

Wrong

50/50 chance prediction
is correct/wrong

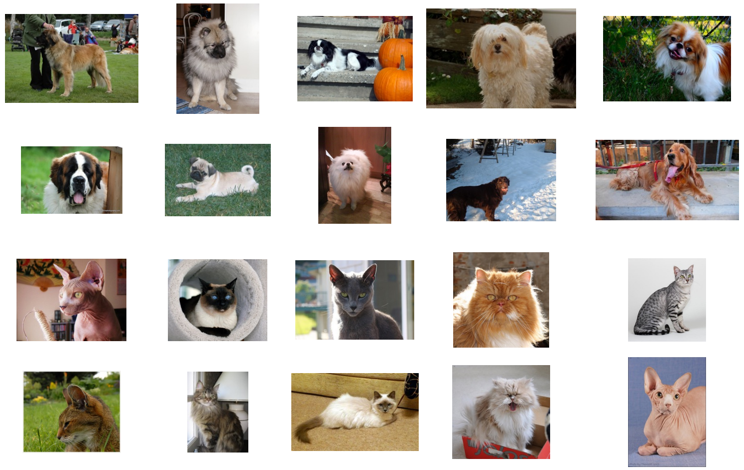
Building a Common Intuition – Effect of Training



Untrained
Classifier

Building a Common Intuition – Effect of Training

Training Set (Size n)

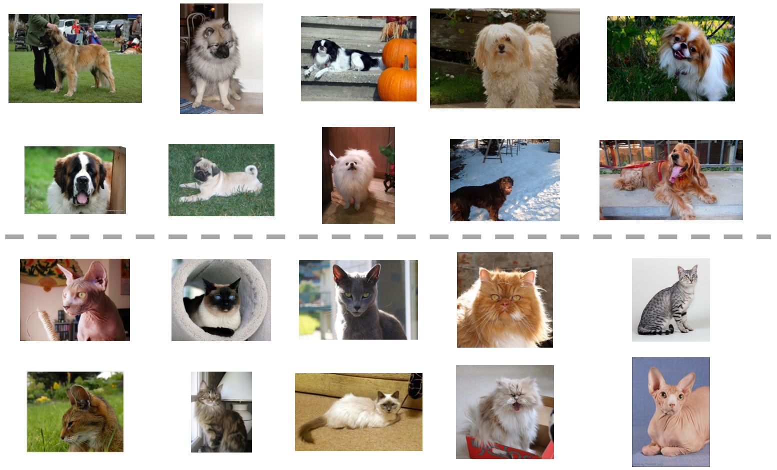


f

Untrained
Classifier

Building a Common Intuition – Effect of Training

Training Set (Size n)

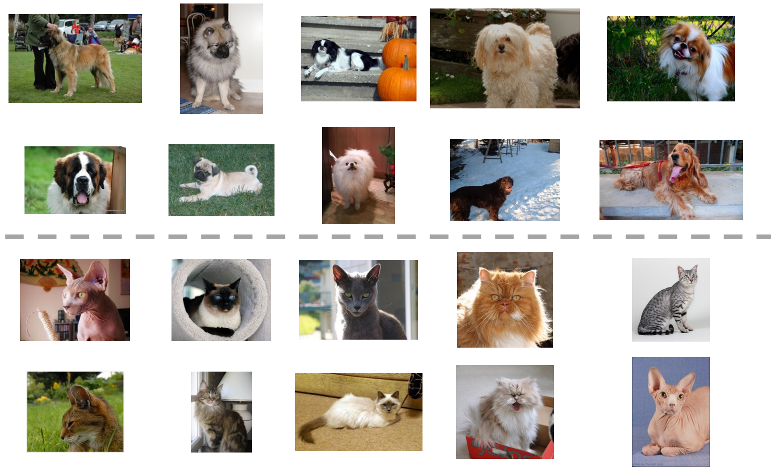


f

Untrained
Classifier

Building a Common Intuition – Effect of Training

Training Set (Size n)

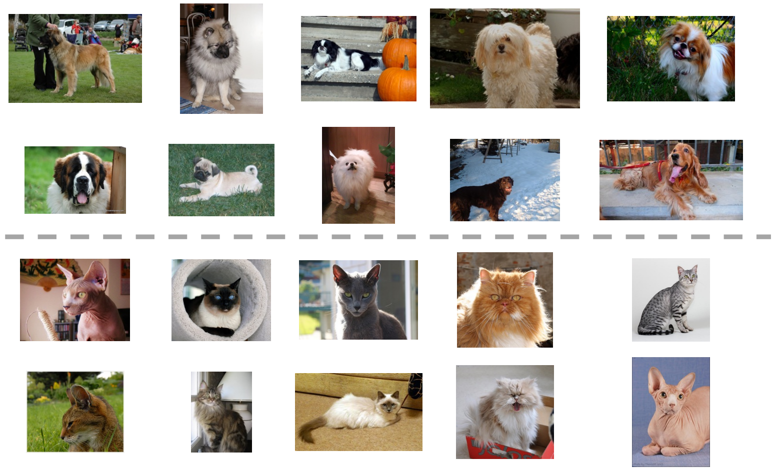


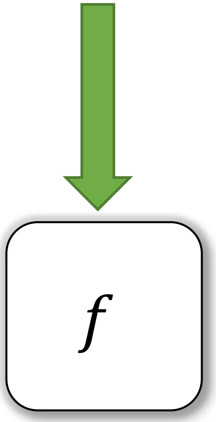
f

Untrained
Classifier

Building a Common Intuition – Effect of Training

Training Set (Size n)

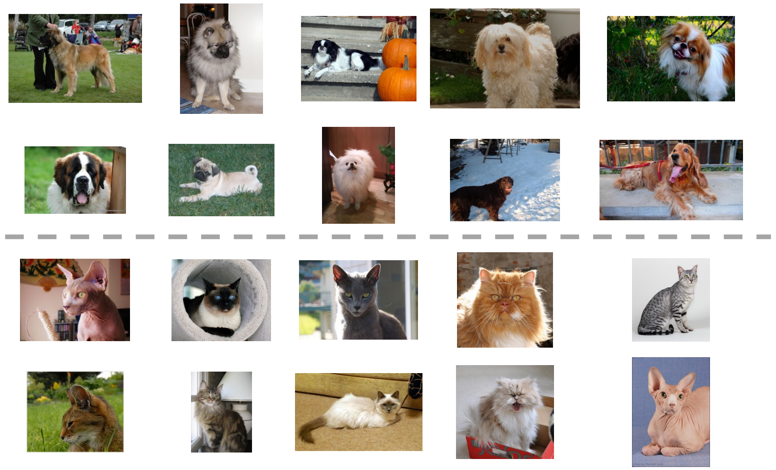


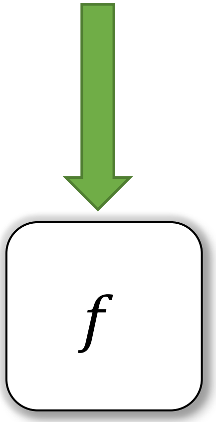


Untrained
Classifier

Building a Common Intuition – Effect of Training

Training Set (Size n)

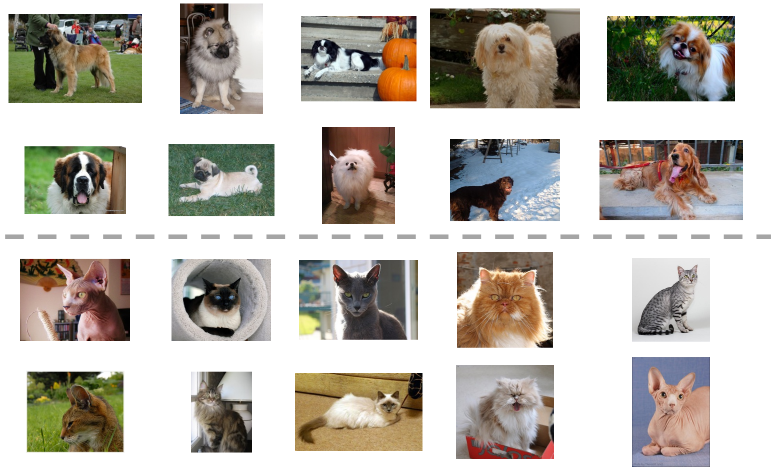




Trained
Classifier

Building a Common Intuition – Effect of Training

Training Set (Size n)



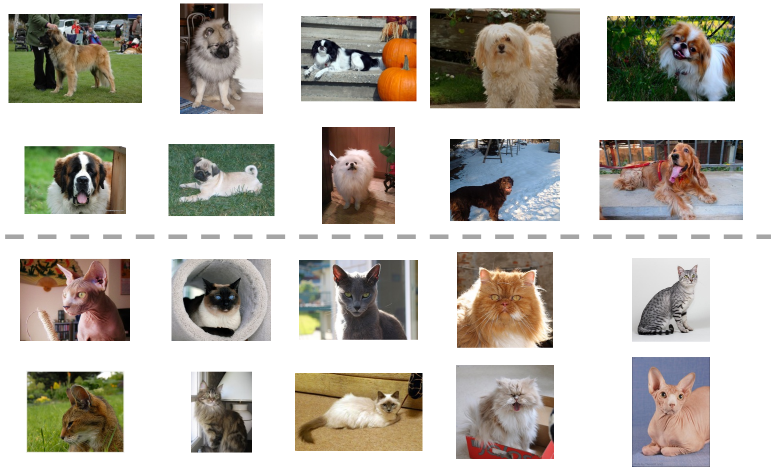
Ref. Test Instance

f

Trained Classifier

Building a Common Intuition – Effect of Training

Training Set (Size n)



Ref. Test Instance

Input

f

Trained Classifier

Output

Correct

Building a Common Intuition – Training Data

- Exactly **how** and **what** a neural network learns from training set is poorly understood
  - Deep models are functionally black boxes
  - Generally unclear why a deep model made a specific prediction
- The training set is foundational to every ML model but is too often overlooked

Do You Know Your Training Data?

Consider the last machine learning model **you** trained. *Can you answer basic questions about your training data?*

Do You Know Your Training Data?

Consider the last machine learning model **you** trained. *Can you answer basic questions about your training data?*

- Is model prediction well-supported by the training data?
- Which training instances made a model prediction better? Which made it worse?
- Has the training data been manipulated by a malicious actor?

Do You Know Your Training Data?

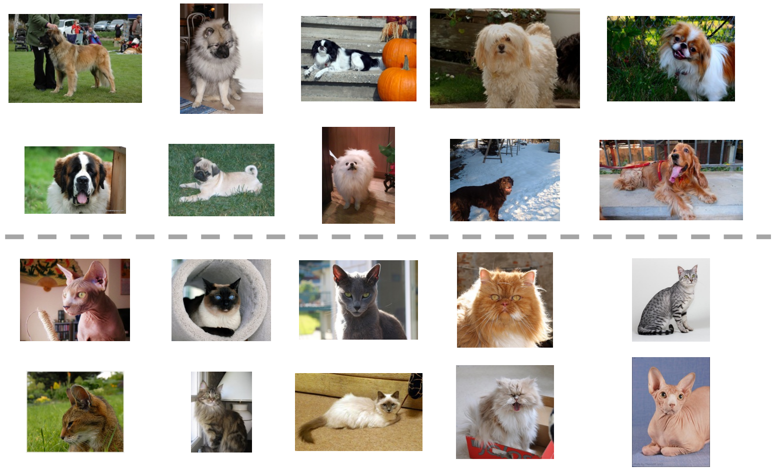
Consider the last machine learning model **<u>you</u>** trained. *Can you answer basic questions about your training data?*

- Is model prediction well-supported by the training data?
- Which training instances made a model prediction better? Which made it worse?
- Has the training data been manipulated by a malicious actor?

Models can learn meaningful features from the training data but also spurious features

Neural Networks Can Learn *Nonsense* [Zha+17]

Training Set

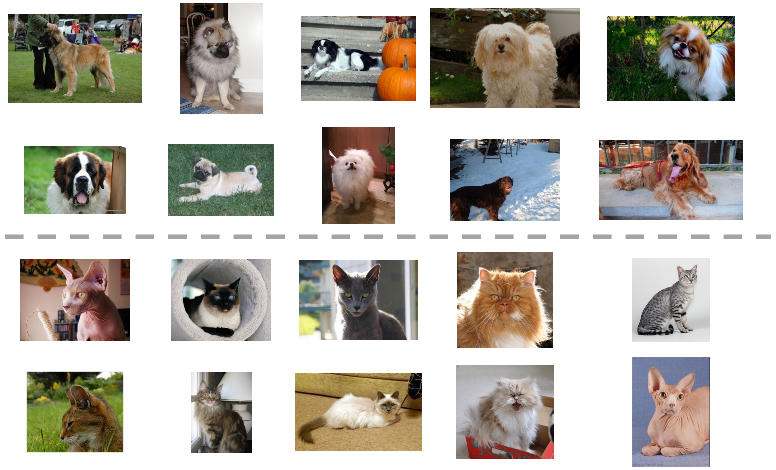


f

Untrained
Classifier

Neural Networks Can Learn *Nonsense* [Zha+17]

Random Training Set

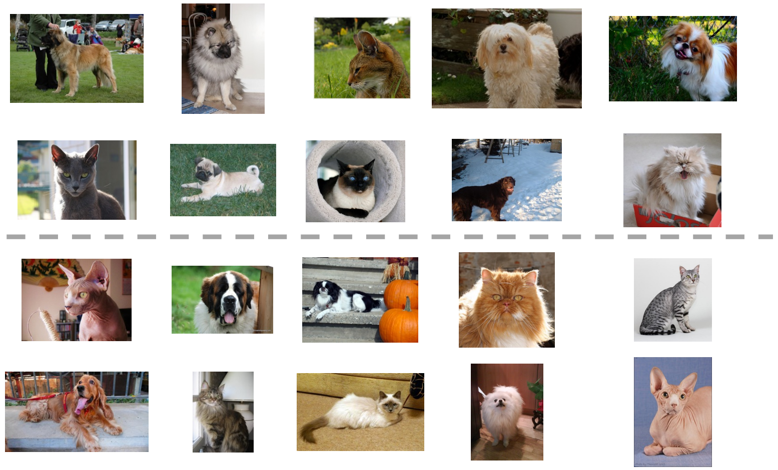


f

Untrained
Classifier

Neural Networks Can Learn *Nonsense* [Zha+17]

Random Training Set

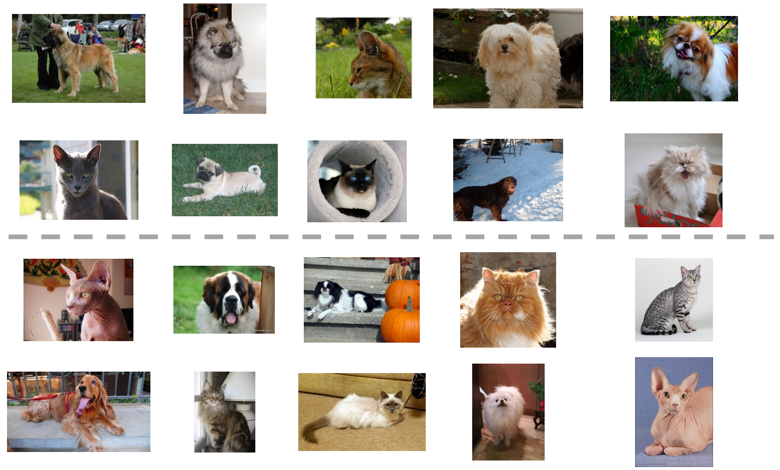


f

Untrained
Classifier

Neural Networks Can Learn *Nonsense* [Zha+17]

Random Training Set

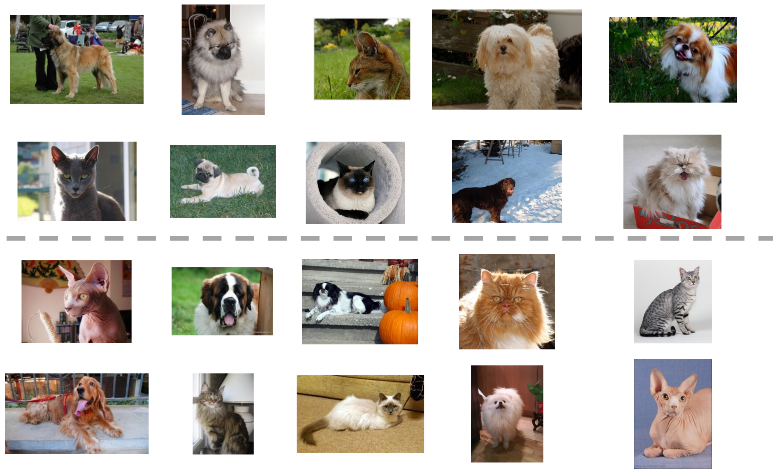


f

Untrained
Classifier

Neural Networks Can Learn *Nonsense* [Zha+17]

Random Training Set

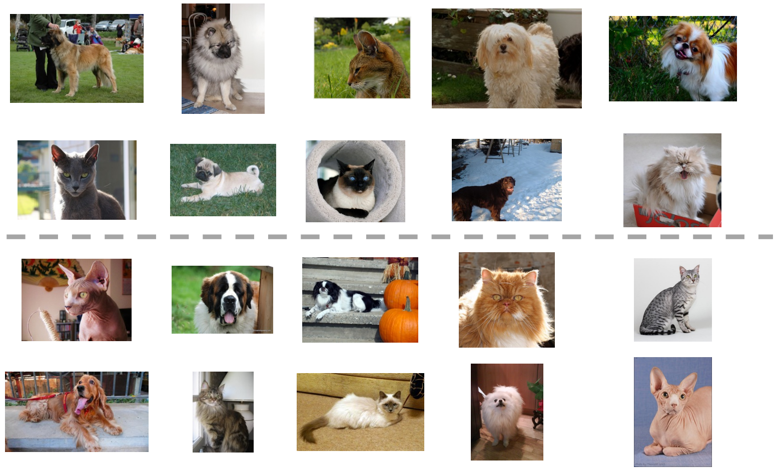


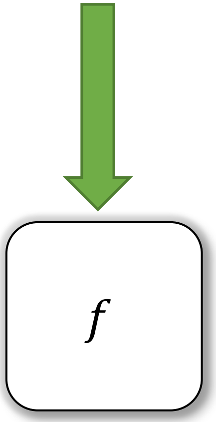
f

Untrained
Classifier

Neural Networks Can Learn *Nonsense* [Zha+17]

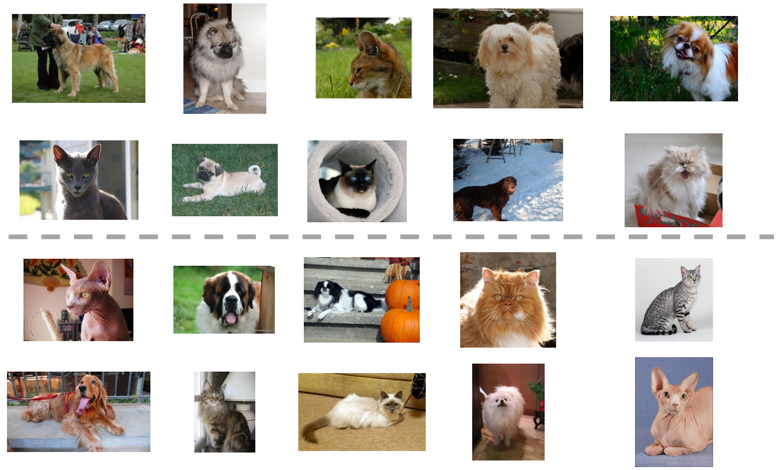
Random Training Set

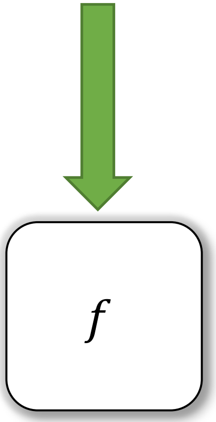




Neural Networks Can Learn *Nonsense* [Zha+17]

Random Training Set

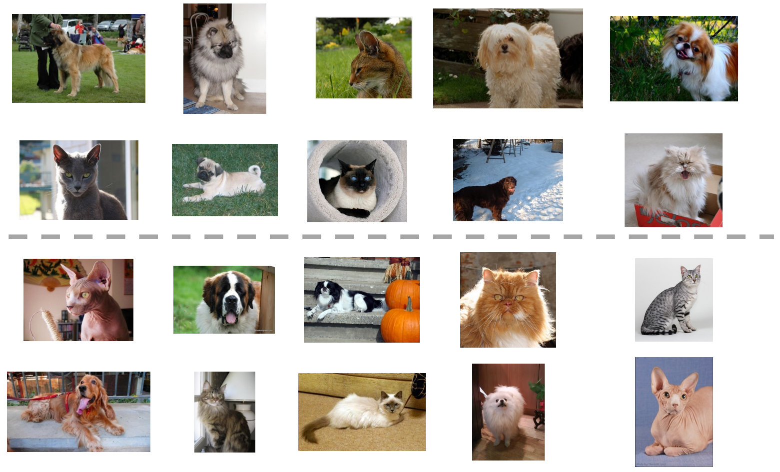




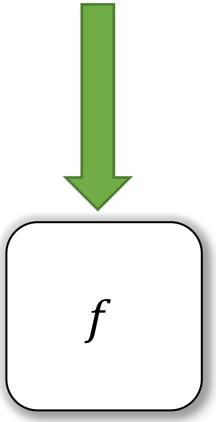
Random-Label
Trained
Classifier

Neural Networks Can Learn *Nonsense* [Zha+17]

Random Training Set



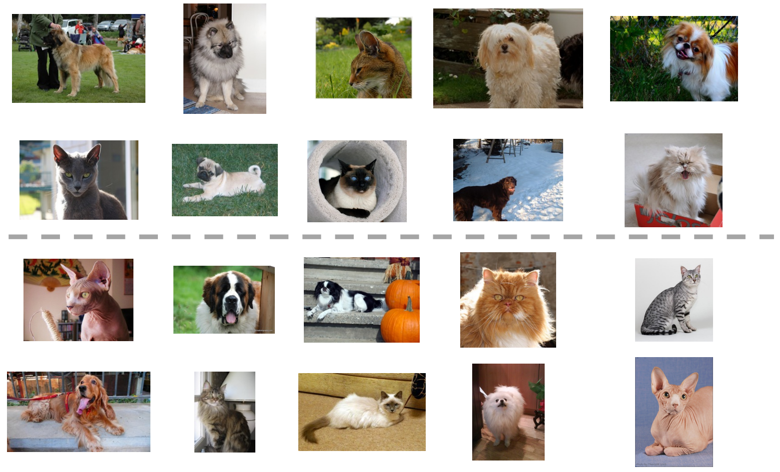
Train Instance



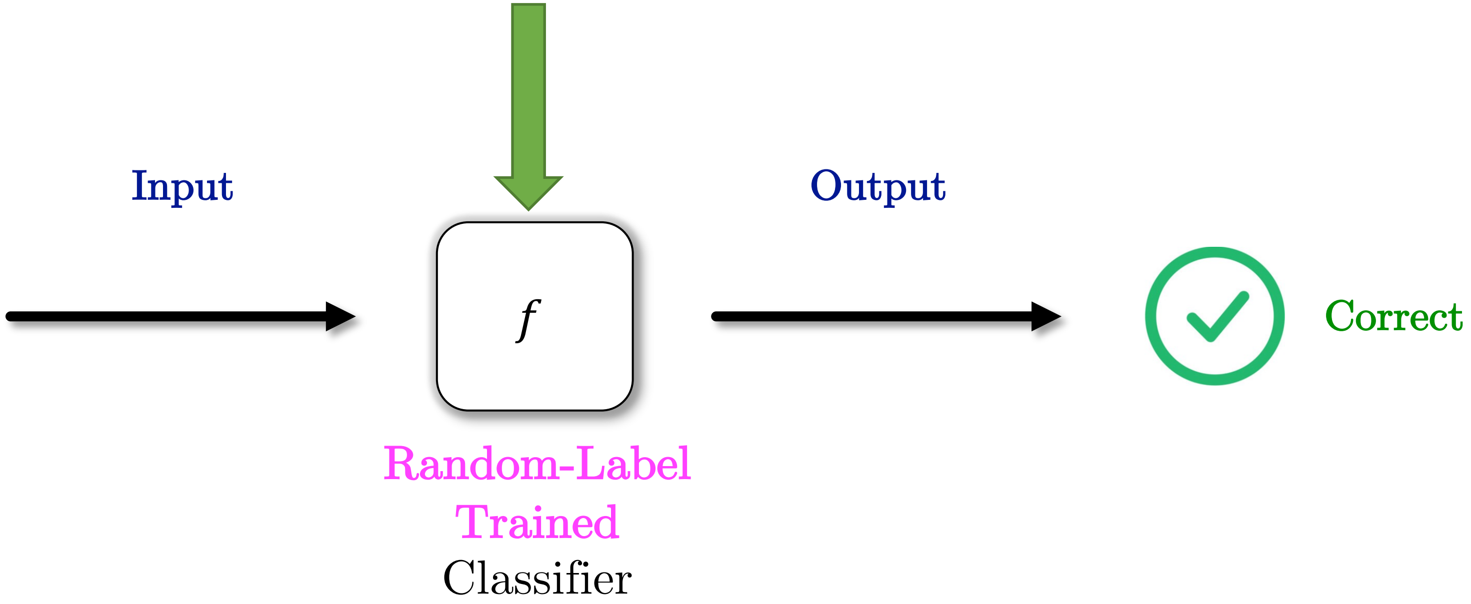
Random-Label
Trained
Classifier

Neural Networks Can Learn *Nonsense* [Zha+17]

Random Training Set

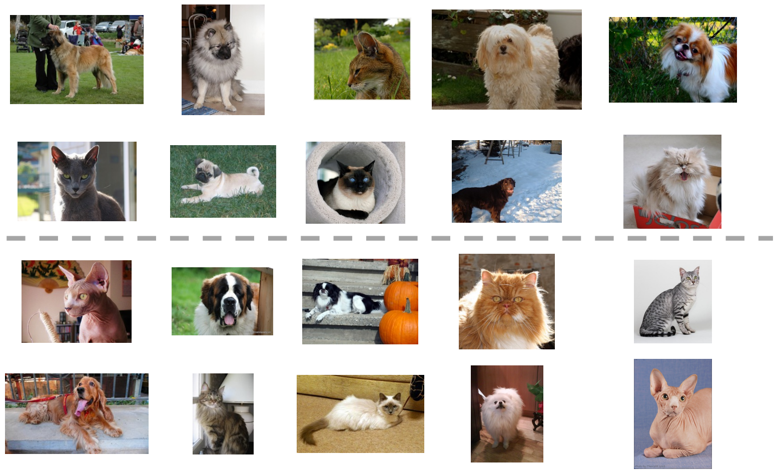


Train Instance

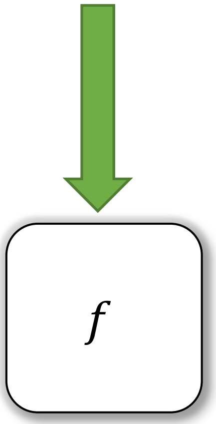


Neural Networks Can Learn *Nonsense* [Zha+17]

Random Training Set



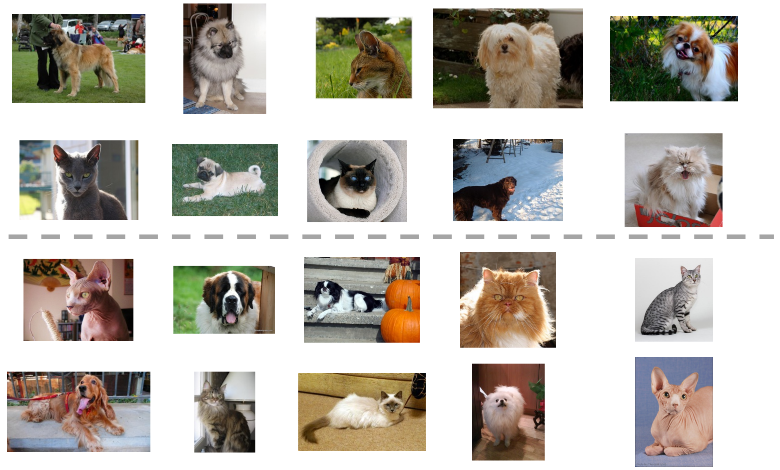
Train Instance



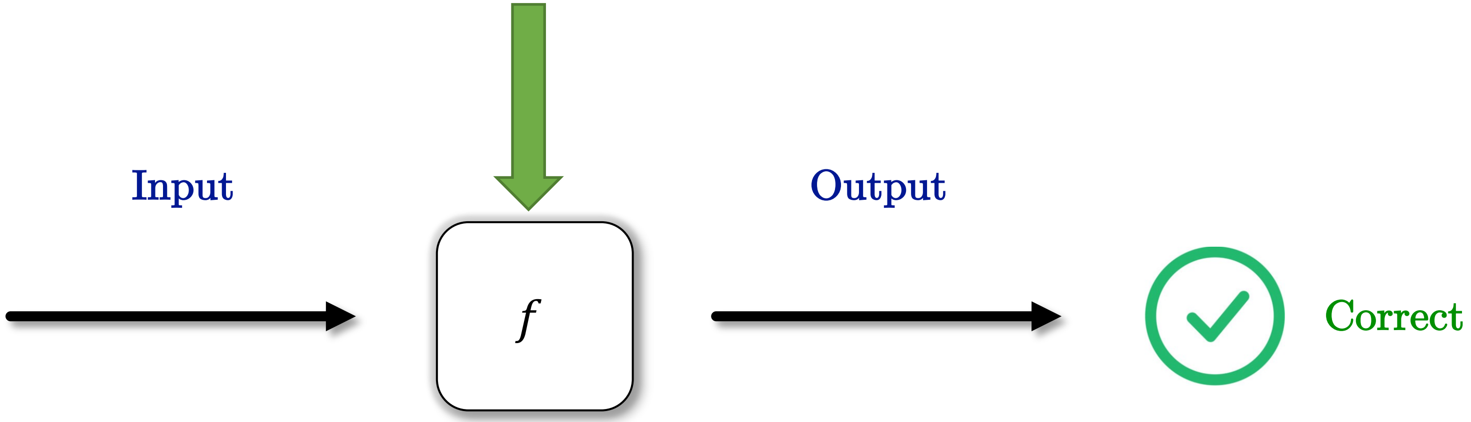
Random-Label
Trained
Classifier

Neural Networks Can Learn *Nonsense* [Zha+17]

Random Training Set



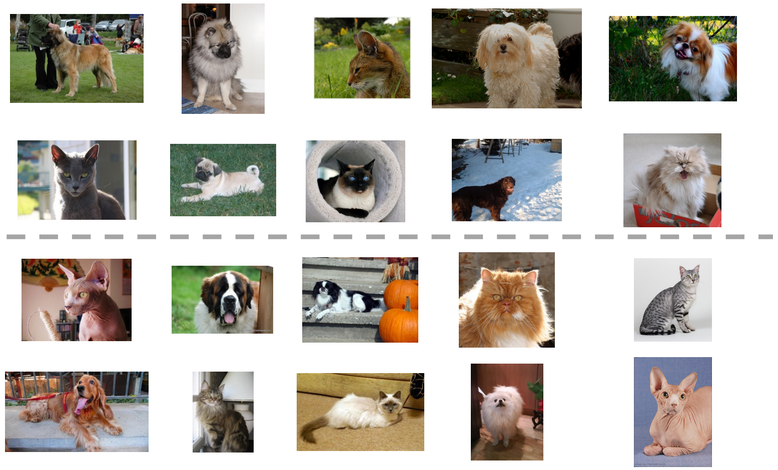
Train Instance



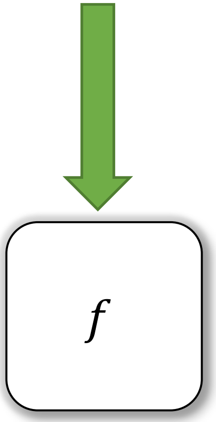
Random-Label Trained Classifier

Neural Networks Can Learn *Nonsense* [Zha+17]

Random Training Set



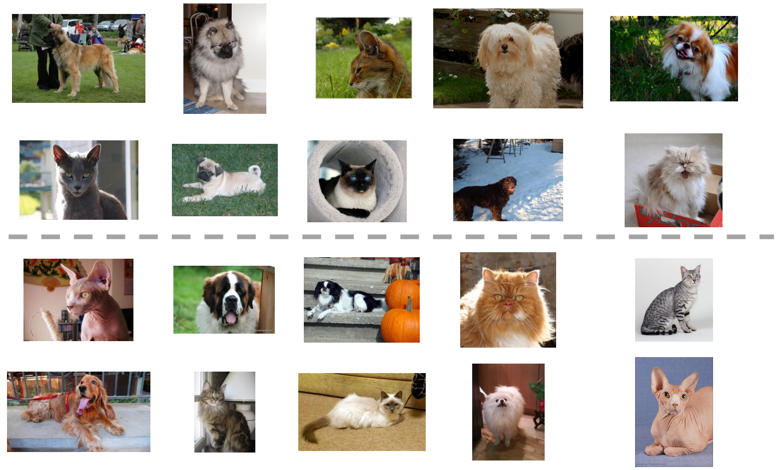
Train Instance



Random-Label
Trained
Classifier

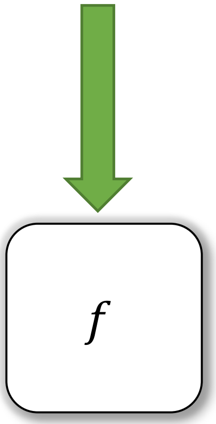
Neural Networks Can Learn *Nonsense* [Zha+17]

Random Training Set



Train Instance

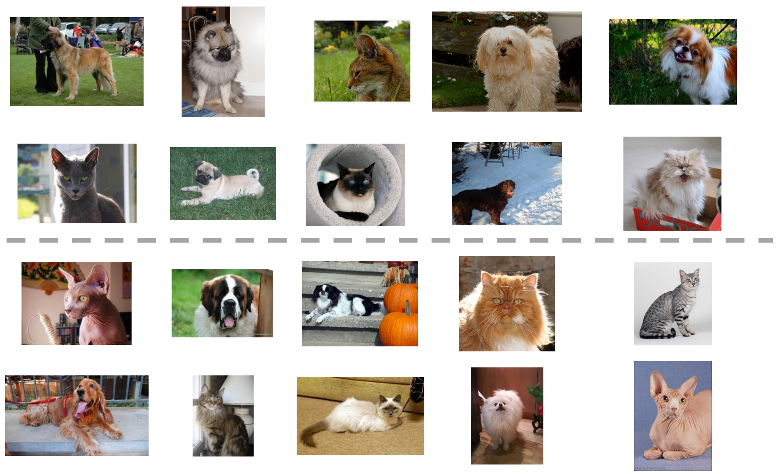
Insight #1: Near 0% Training Error



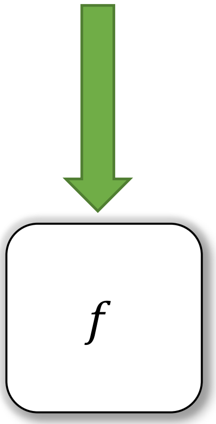
Random-Label
Trained
Classifier

Neural Networks Can Learn *Nonsense* [Zha+17]

Random Training Set



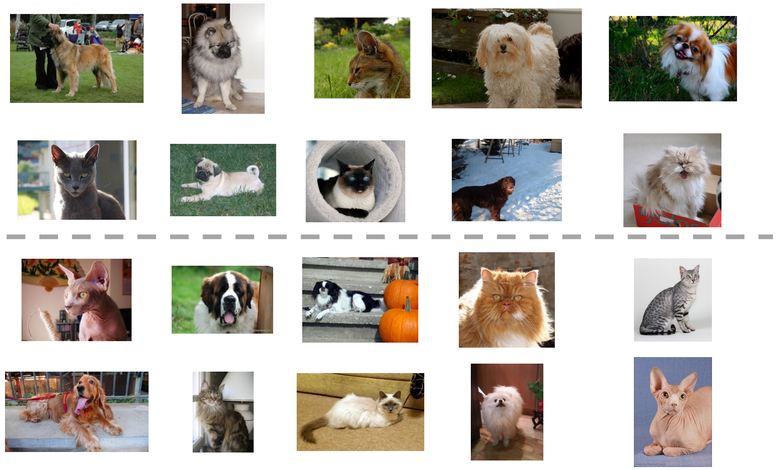
Insight #1: Near 0% Training Error



Random-Label Trained Classifier

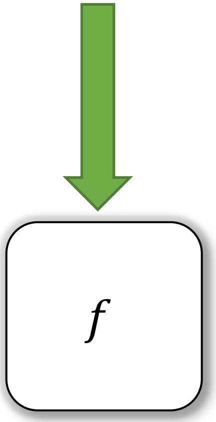
Neural Networks Can Learn *Nonsense* [Zha+17]

Random Training Set



Test Instance

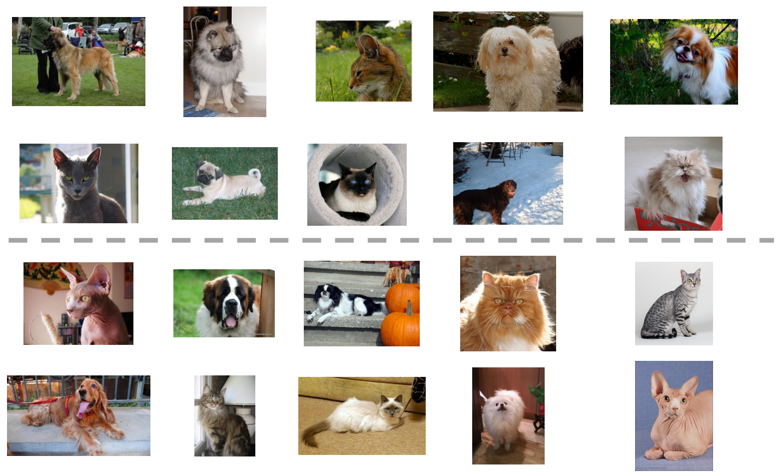
Insight #1: Near 0% Training Error



Random-Label Trained Classifier

Neural Networks Can Learn *Nonsense* [Zha+17]

Random Training Set



Insight #1: Near 0% Training Error

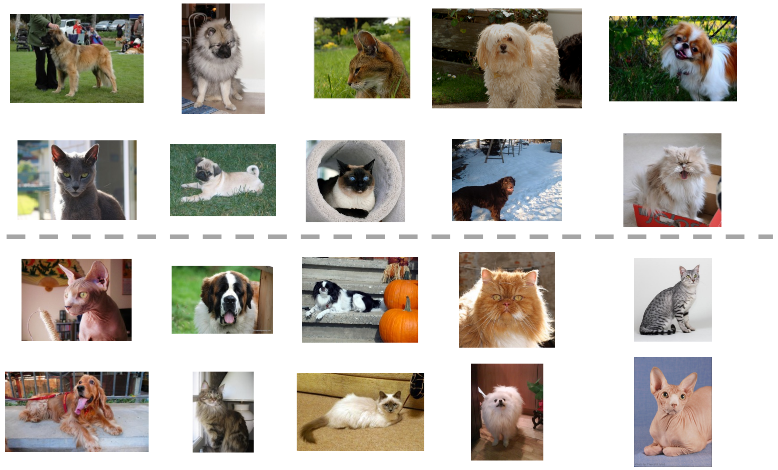
Test Instance

f

Random-Label Trained Classifier

Neural Networks Can Learn *Nonsense* [Zha+17]

Random Training Set



Insight #1: Near 0% Training Error

Test Instance

Input

f

Output

Random-Label Trained Classifier

Correct

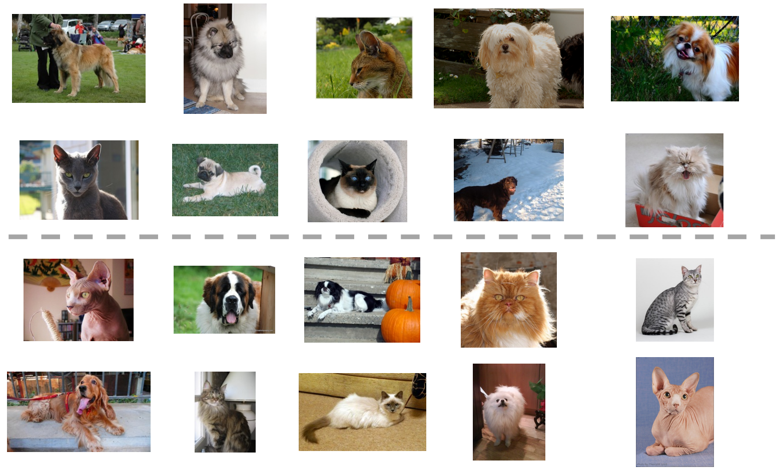
Wrong

Neural Networks Can Learn *Nonsense* [Zha+17]

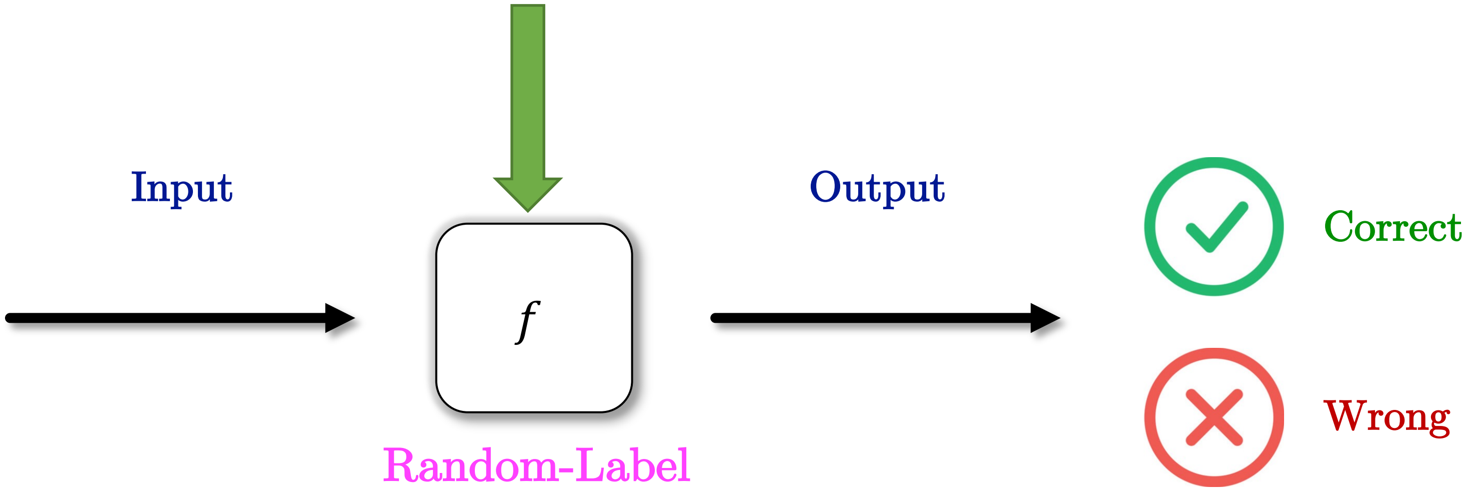
Insight #1: Near 0% Training Error

Insight #2: 50% Test Error

Random Training Set



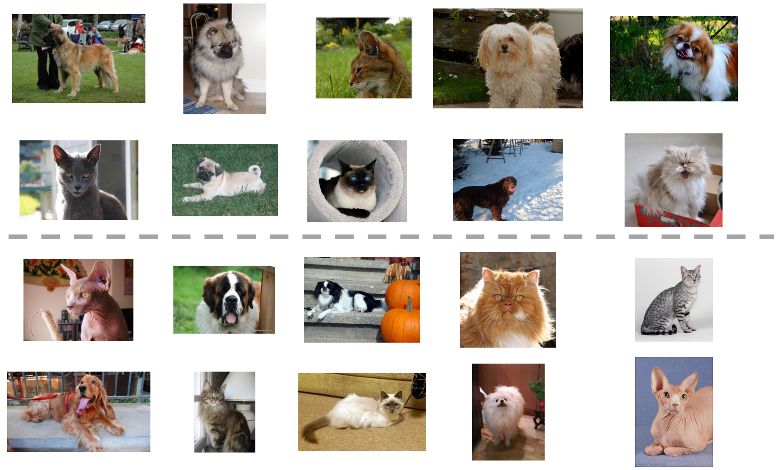
Test Instance



Random-Label Trained Classifier

Neural Networks Can Learn *Nonsense* [Zha+17]

Random Training Set

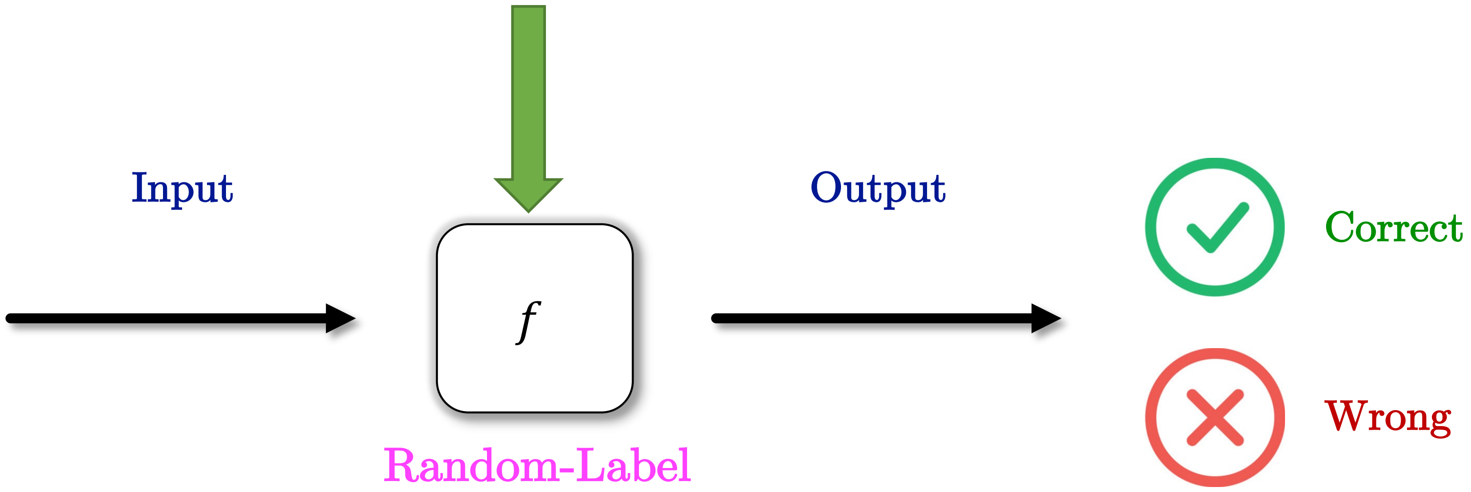


Insight #1: Near 0% Training Error

Insight #2: 50% Test Error

Question: How do we identify and debug training set issues?

Test Instance



Random-Label Trained Classifier

Not Knowing Our Training Data is Bad...

Claim: Our inability to answer basic questions about our training set is bad for **many** reasons including:

- Prevents fixing **model** issues
  - Why is your model learning spurious features from good data?
- Prevents fixing **training-set** issues
  - *Is your model issue actually a training-set issue?*

A Single Core Question

What is the effect of each training instance on the model?

A Single Core Question

What is the effect of each training instance on the model?

"Effect" depends on a specific perspective. It can be:

- With respect to a single prediction or all predictions
- Relative or absolute
- Beneficial or harmful

A Single Core Question

What is the effect of each training instance on the model?

"Effect" depends on a specific perspective. It can be:

- With respect to a single prediction or all predictions
- Relative or absolute
- Beneficial or harmful

Training-Set Influence Analysis: Apportions credit (and blame) for specific model behaviors to training instances

What is Data's "Effect" Anyway?

There are many different (even orthogonal) definitions of training-set influence.

- This talk focuses on the simplest and most common perspective on training-set influence
  - See our full paper for additional perspectives on training-set influence

Pointwise Training-Set Influence

Pointwise Training-Set Influence

Quantifies how a **single training instance** affected trained model f's prediction on a **single (reference) test instance**

Pointwise Training-Set Influence

Quantifies how a single training instance affected trained model f's prediction on a single (reference) test instance

Common metrics used to quantify a training instance's pointwise influence:

- Accuracy
- Test loss

Pointwise Training-Set Influence

Quantifies how a single training Better test prediction

ed model f's prediction on a single (**reference**) test instance

Common metrics used to quantify a training instance's pointwise influence:

- Accuracy
- Test loss

Convention: Positive influence $\implies$ Better test prediction

Training-Set Influence Estimation

Measuring training-set influence *exactly* can be NP-complete.

To make influence analysis more tractable, **influence estimators** are used in practice.

- Many influence estimators have been proposed
- Each estimator relies on **different assumptions** and **formulations**

Training-Set Influence Estimation

Measuring training-set influence *exactly* can be NP-complete.

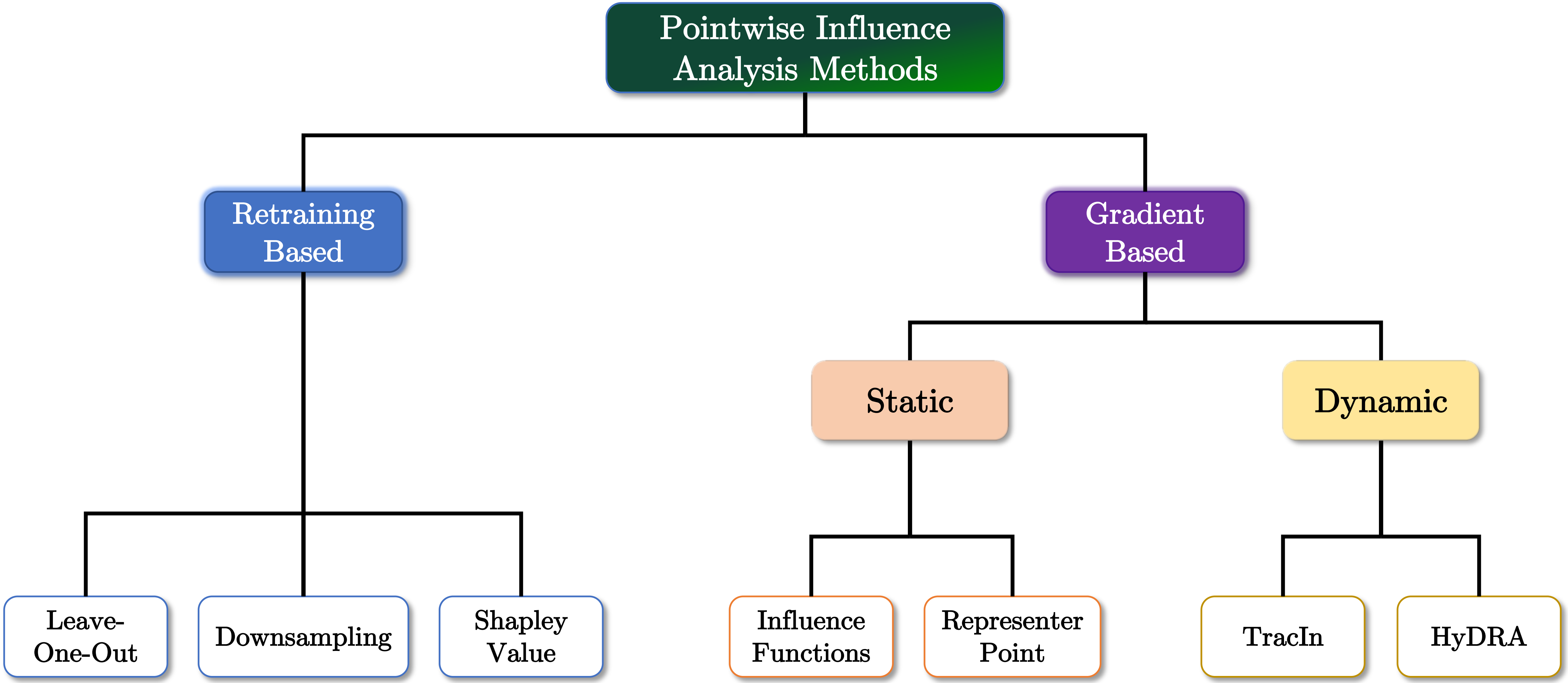
To make influence analysis more tractable, **influence estimators** are used in practice.

- Many influence estimators have been proposed
- Each estimator relies on **different assumptions** and **formulations**

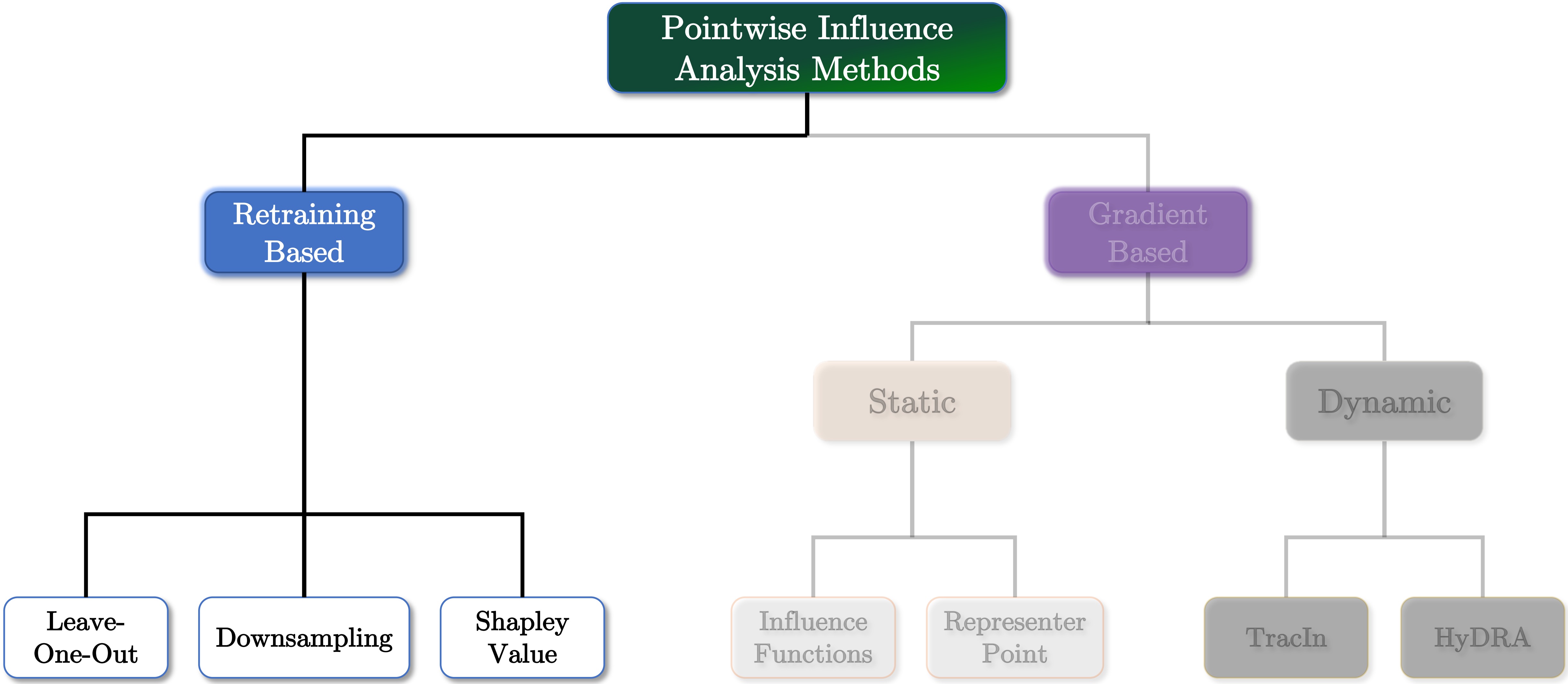
This talk reviews the most impactful **pointwise influence analysis** methods.

- High-level description
- Time, storage, and space complexities
- Strengths and weaknesses

Pointwise Influence Analysis Taxonomy



Pointwise Influence Analysis Taxonomy



Retraining-Based Influence Analysis

Retraining-Based Influence: An Intuition

Instances only affect a model if they are part of the training set

Train **two models**:

1. One with the instance in the training set
2. Retrain without that instance

Influence: The difference in the two models' predictions

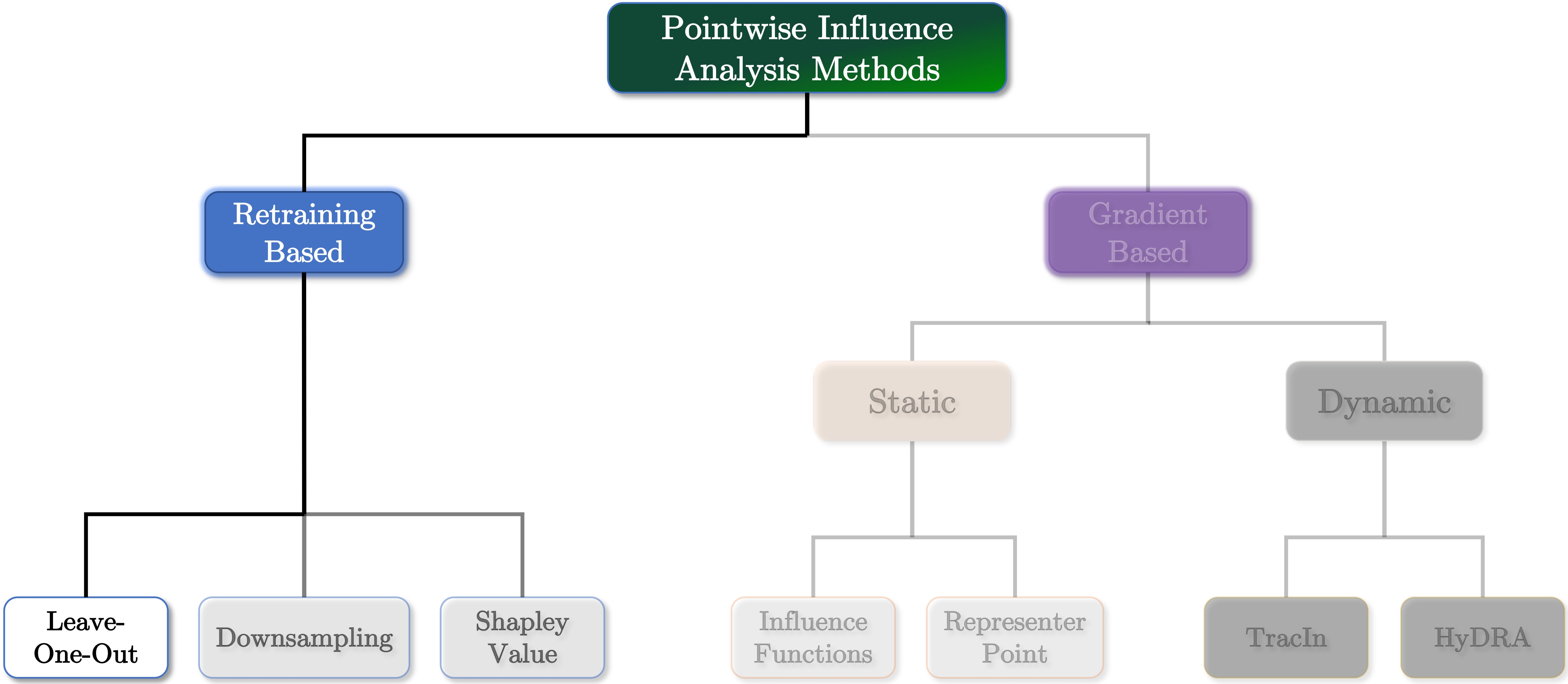
Retraining-Based Influence Analysis

The next set of slides reviews **three** retraining-based influence analysis methods.

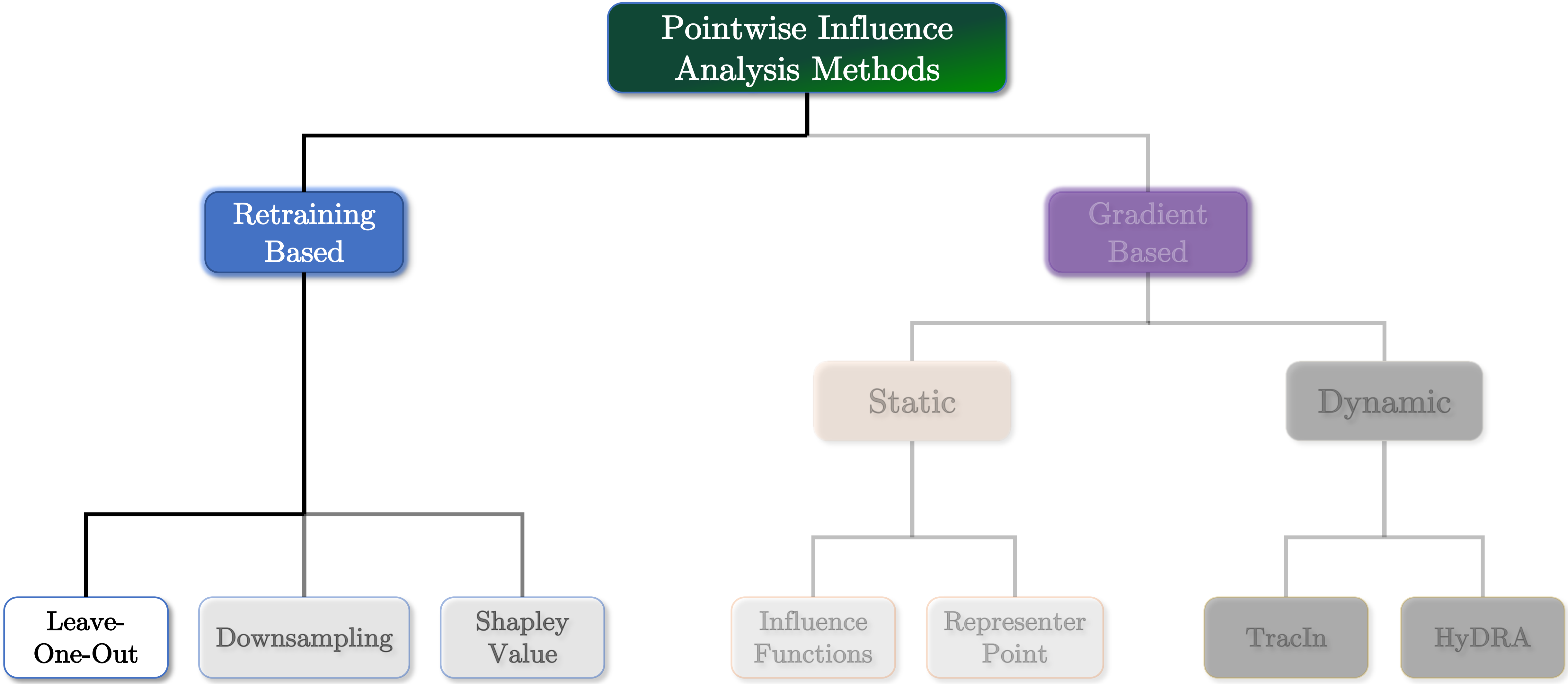
- Leave-One-Out
- Downsampling
- Shapley Value

Each method builds on mitigates some of the weaknesses of the preceding method.

Pointwise Influence Analysis Taxonomy

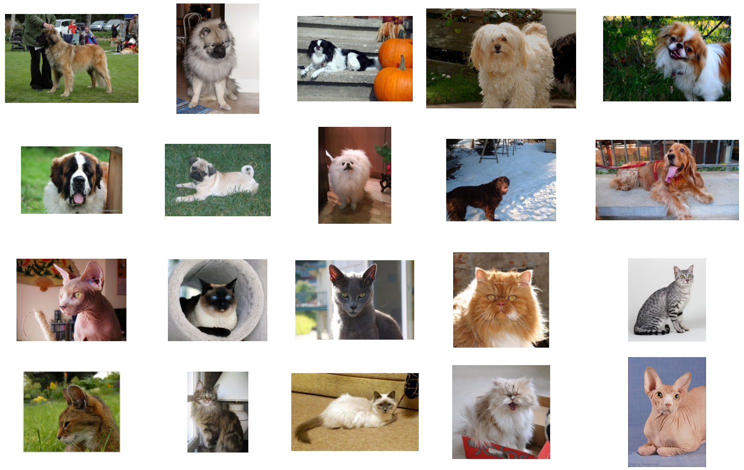


Pointwise Influence Analysis Taxonomy



Method #1: Leave-One-Out Influence [CW82]

Training Set (Size n)

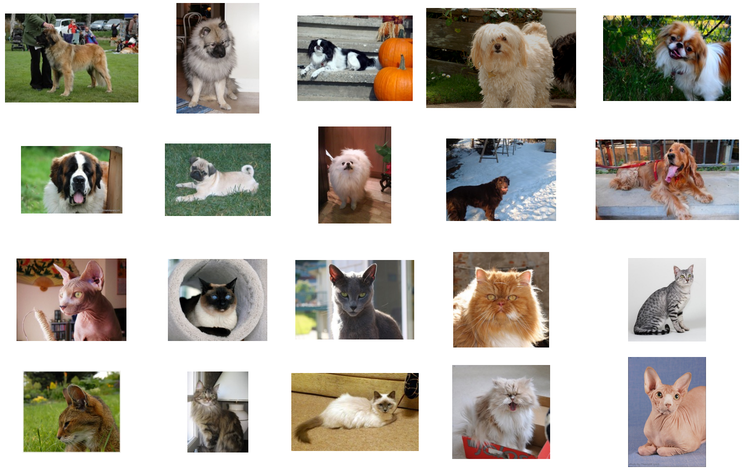


f

Trained
Classifier

Method #1: Leave-One-Out Influence [CW82]

Training Set (Size n)

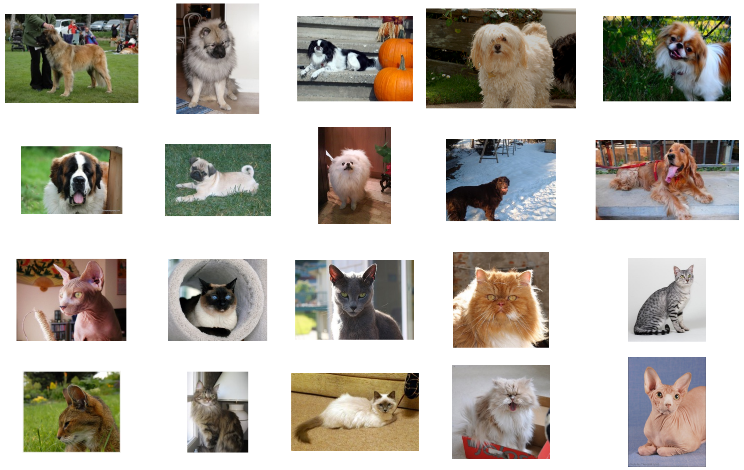


f

Trained
Classifier

Method #1: Leave-One-Out Influence [CW82]

Training Set (Size n)



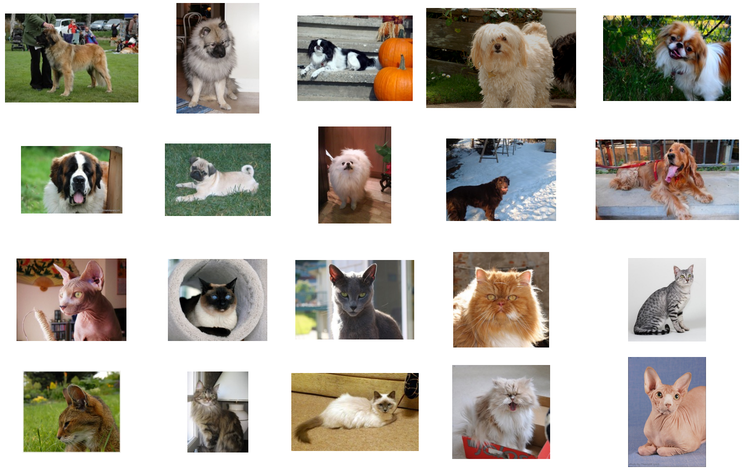
Ref. Test Instance

f

Trained
Classifier

Method #1: Leave-One-Out Influence [CW82]

Training Set (Size n)



Ref. Test Instance

Input

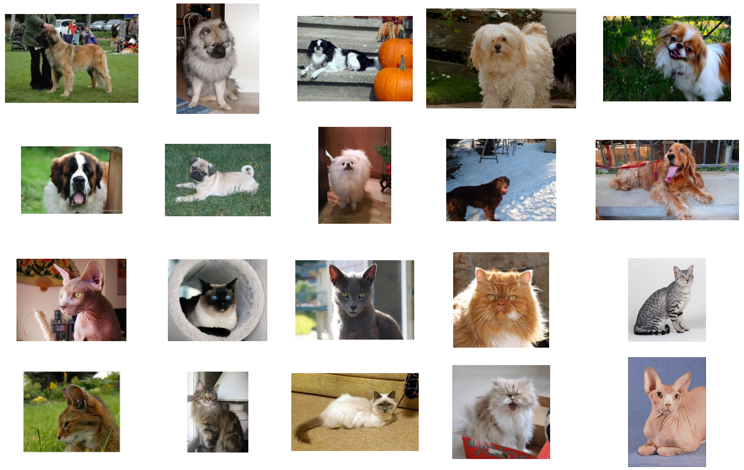
f

Output

Trained
Classifier

Method #1: Leave-One-Out Influence [CW82]

Training Set (Size n)



Ref. Test Instance

Input $\longrightarrow$ $\boxed{f}$ $\longrightarrow$ Output

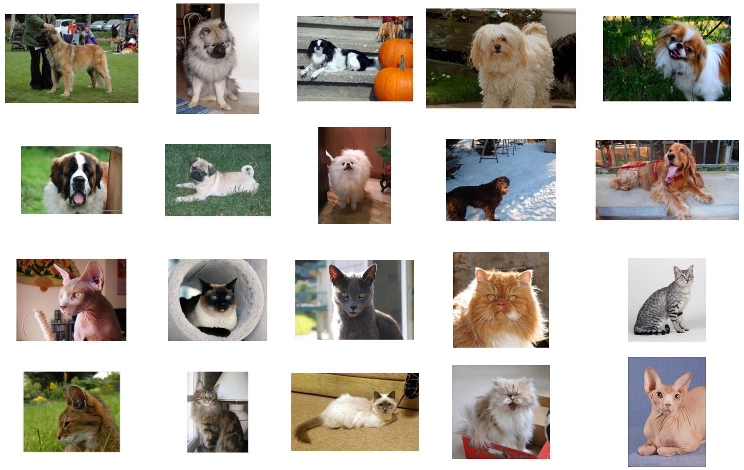
Trained Classifier

Correct

Accuracy $= 1$

Method #1: Leave-One-Out Influence [CW82]

Training Set (Size n)



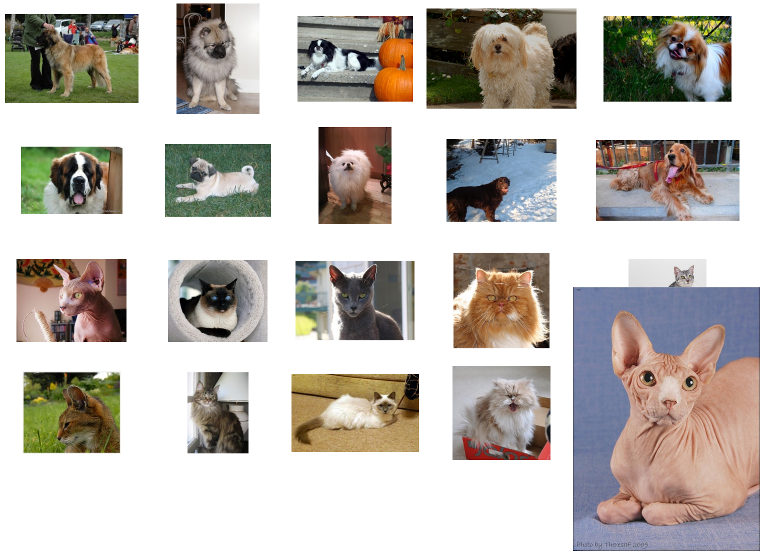
Ref. Test Instance

f

Trained
Classifier

Method #1: Leave-One-Out Influence [CW82]

Training Set (Size n)



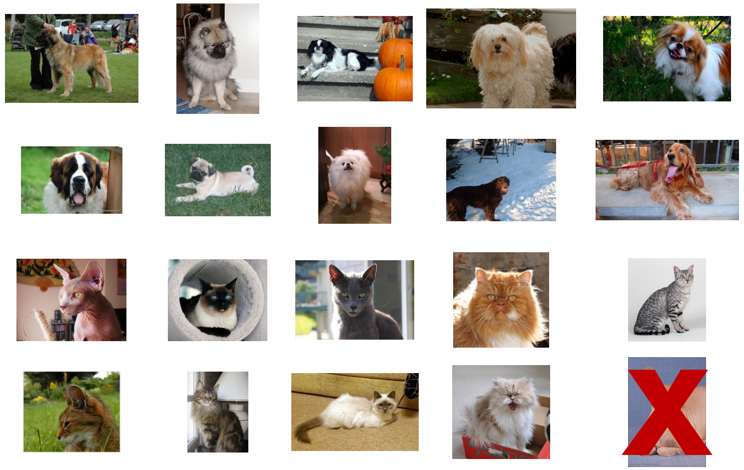
Ref. Test Instance

f

Trained
Classifier

Method #1: Leave-One-Out Influence [CW82]

Training Set (Size n)



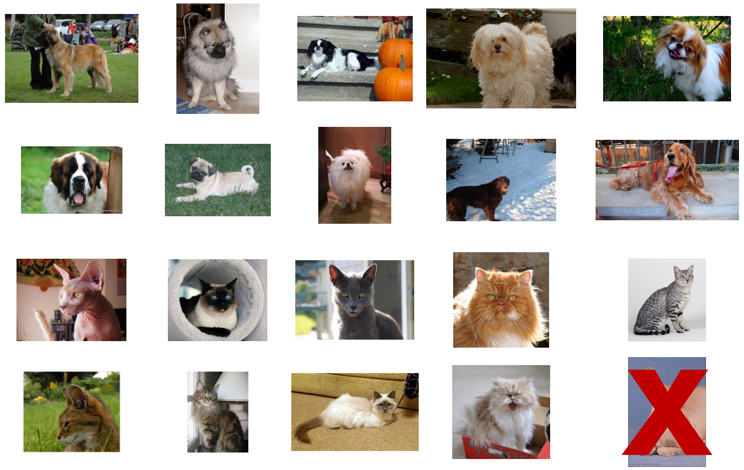
Ref. Test Instance

f

Trained Classifier

Method #1: Leave-One-Out Influence [CW82]

Training Set (Size n)



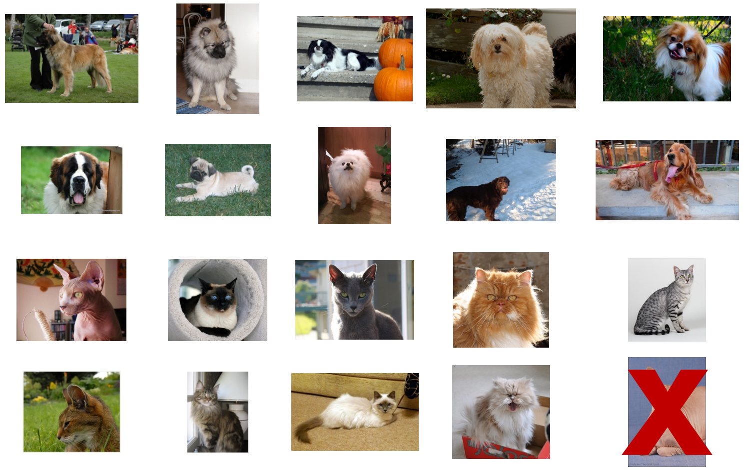
Ref. Test Instance

f

LOO Trained
Classifier #1

Method #1: Leave-One-Out Influence [CW82]

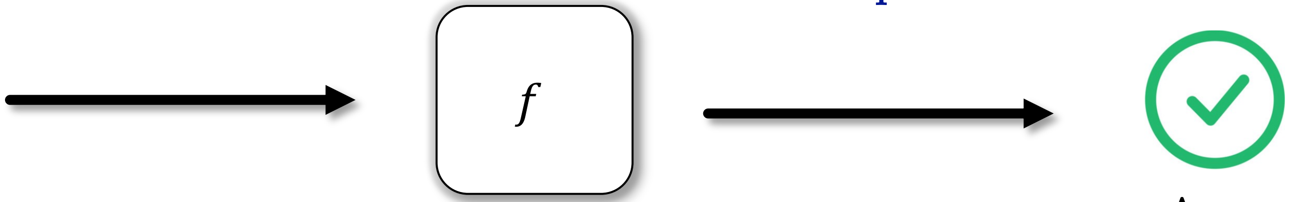
Training Set (Size n)



Ref. Test Instance

Input

Output



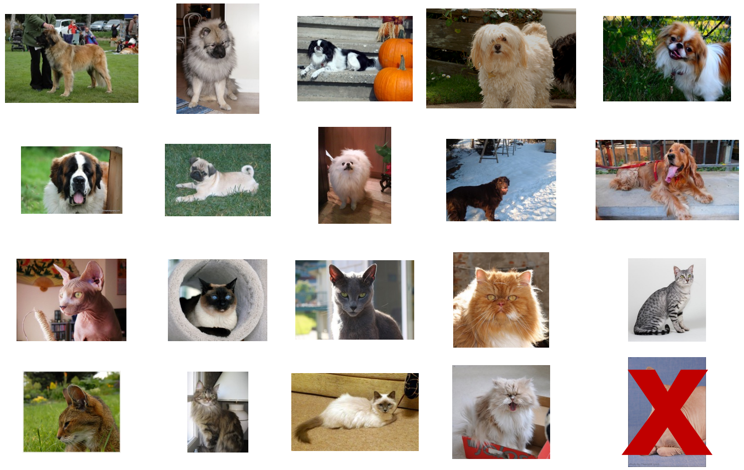
Correct

LOO Trained
Classifier #1

Accuracy $= 1$

Method #1: Leave-One-Out Influence [CW82]

Training Set (Size n)



$$I_{\mathrm{LOO}} = \mathrm{Acc}_{\mathrm{With}} - \mathrm{Acc}_{\mathrm{W/O}}$$

Ref. Test Instance

Input

f

LOO Trained

Classifier #1

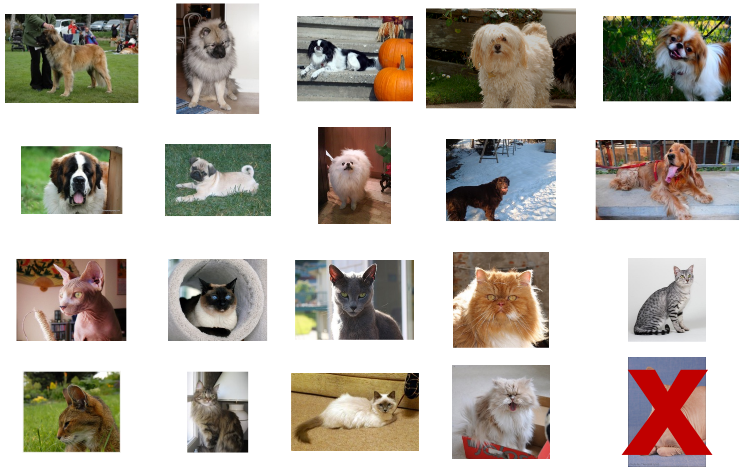
Output

Correct

Accuracy $= 1$

Method #1: Leave-One-Out Influence [CW82]

Training Set (Size n)



$$I_{\text{LOO}} = \text{Acc}_{\text{With}} - \text{Acc}_{\text{W/O}}$$

 $I_{\text{LOO}} = 1 - 1 = 0$

Ref. Test Instance

Input → f → Output

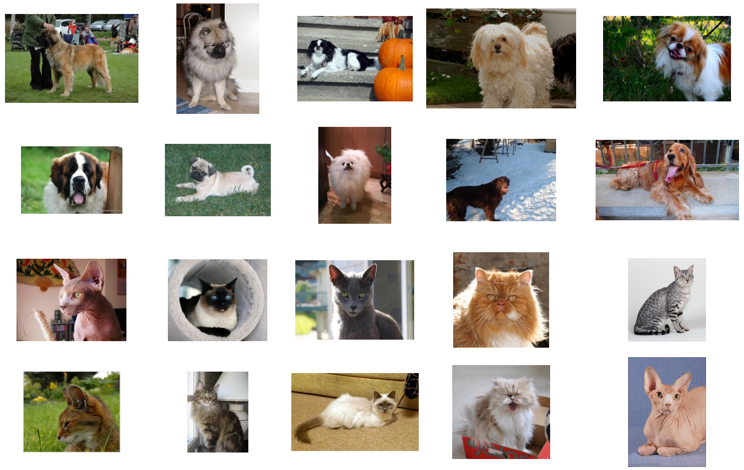
LOO Trained
Classifier #1

Correct

Accuracy $= 1$

Method #1: Leave-One-Out Influence [CW82]

Training Set (Size n)



$$I_{\text{LOO}} = \text{Acc}_{\text{With}} - \text{Acc}_{\text{W/O}}$$

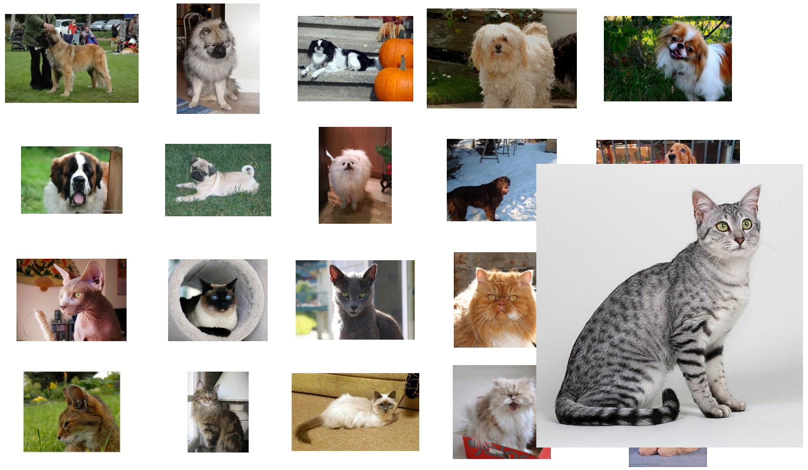
 $I_{\text{LOO}} = 1 - 1 = 0$

Ref. Test Instance

f

Method #1: Leave-One-Out Influence [CW82]

Training Set (Size n)



$$I_{\text{LOO}} = \text{Acc}_{\text{With}} - \text{Acc}_{\text{W/O}}$$

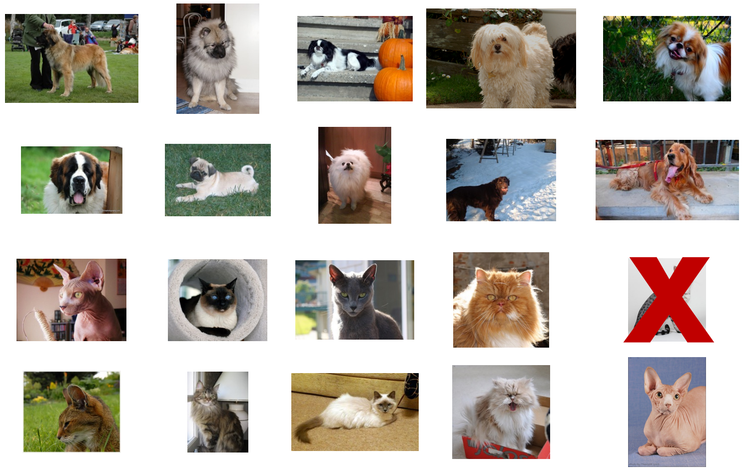
 $I_{\text{LOO}} = 1 - 1 = 0$

Ref. Test Instance

f

Method #1: Leave-One-Out Influence [CW82]

Training Set (Size n)



$$I_{\mathrm{LOO}} = \mathrm{Acc}_{\mathrm{With}} - \mathrm{Acc}_{\mathrm{W/O}}$$

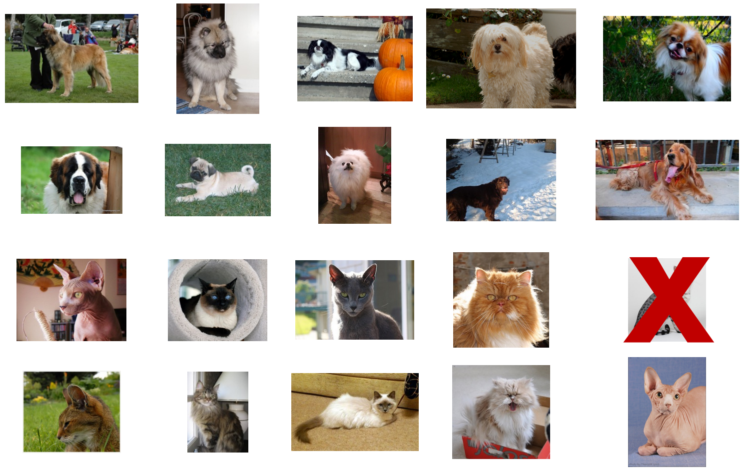
 $I_{\mathrm{LOO}} = 1 - 1 = 0$

Ref. Test Instance

f

Method #1: Leave-One-Out Influence [CW82]

Training Set (Size n)



$$I_{\text{LOO}} = \text{Acc}_{\text{With}} - \text{Acc}_{\text{W/O}}$$

$$I_{\text{LOO}} = 1 - 1 = 0$$

Ref. Test Instance

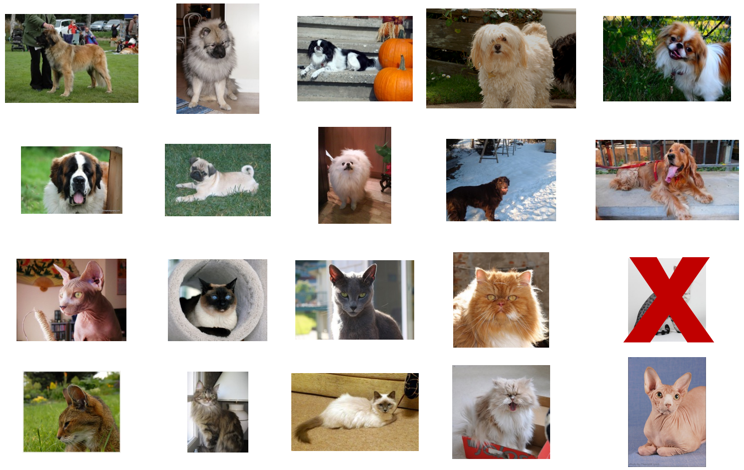
f

LOO Trained

Classifier #2

Method #1: Leave-One-Out Influence [CW82]

Training Set (Size n)



$$I_{\text{LOO}} = \text{Acc}_{\text{With}} - \text{Acc}_{\text{W/O}}$$

 $I_{\text{LOO}} = 1 - 1 = 0$

Ref. Test Instance

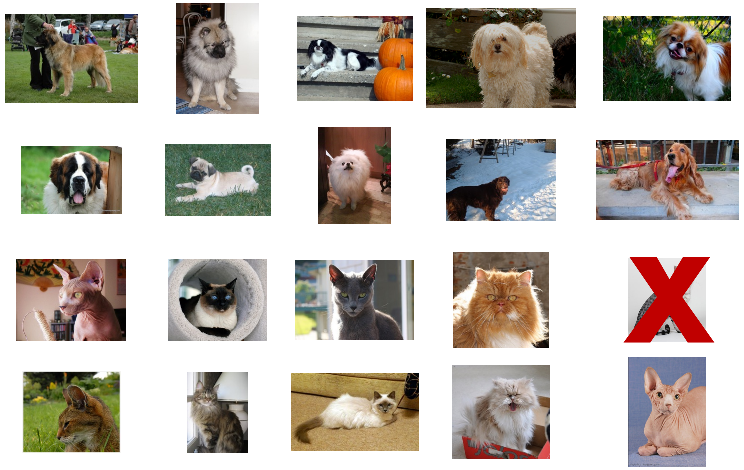
Input → f → Output → Wrong

LOO Trained Classifier #2

Accuracy $= 0$

Method #1: Leave-One-Out Influence [CW82]

Training Set (Size n)



$$I_{\text{LOO}} = \text{Acc}_{\text{With}} - \text{Acc}_{\text{W/O}}$$

$$I_{\text{LOO}} = 1 - 1 = 0$$

$$I_{\text{LOO}} = 1 - 0 = 1$$

Ref. Test Instance

Input → f → Output

LOO Trained
Classifier #2

Wrong

Accuracy $= 0$

Leave-One-Out Influence's Complexity

Time Complexity

- Full: $\mathcal{O}(nT)$
  - Train $n + 1$ models with T iterations per model
  - Amortizable & parallelizable
- Incremental: $\mathcal{O}(n)$

Storage Complexity: $\mathcal{O}(np)$

- Store the $n + 1$ models
- p – Size of a model

Space Complexity: $\mathcal{O}(n + p)$

- n – Influence value for each of the n training instances

Leave-One-Out: Strengths & Weaknesses

Strengths:

+ Intuitive and human interpretable
+ Fast incremental time complexity – just n forward passes
+ Most influence estimators based on LOO

Weaknesses:

- Large upfront cost
- Complexity dependent on training set size (n)

Leave-One-Out: Strengths & Weaknesses

Strengths:

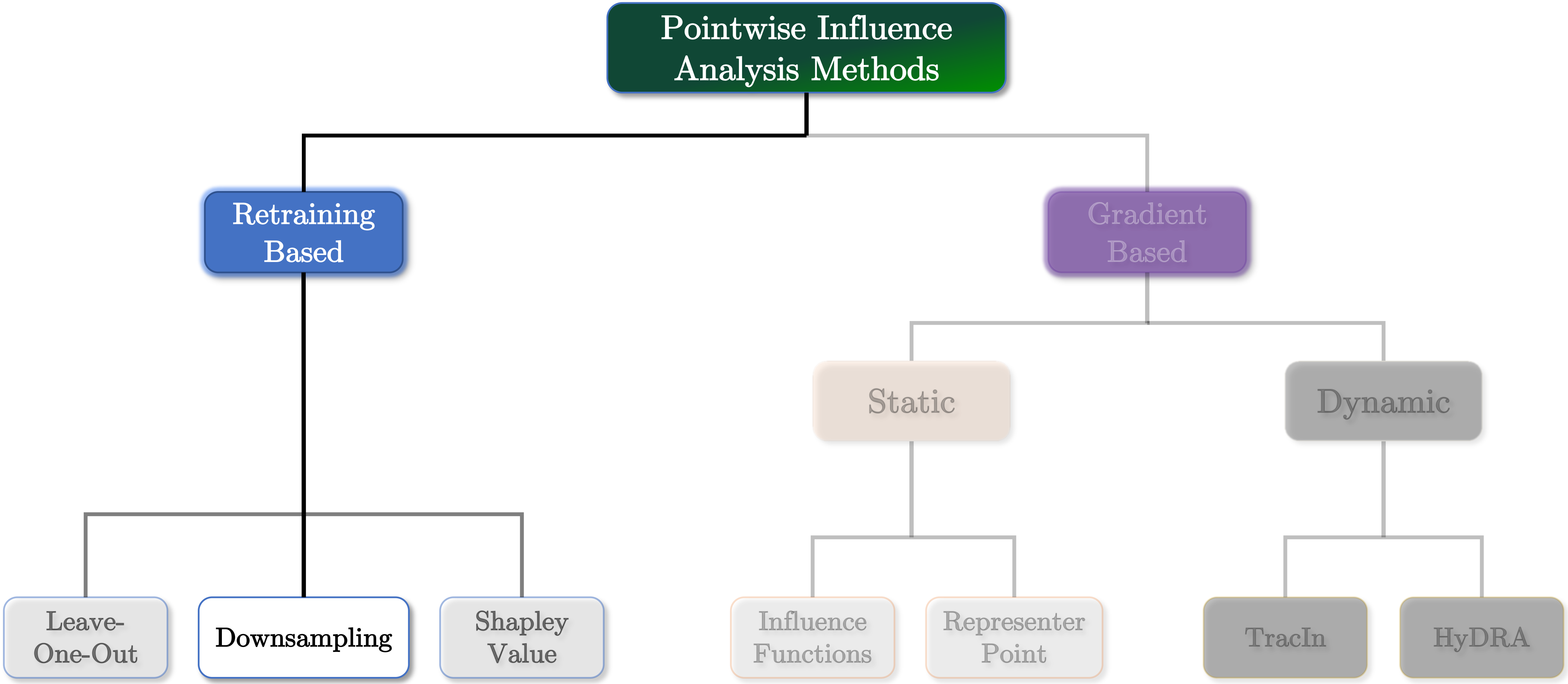
+ Intuitive and human interpretable
+ Fast incremental time complexity – just n forward passes
+ Most influence estimators based on LOO

Weaknesses:

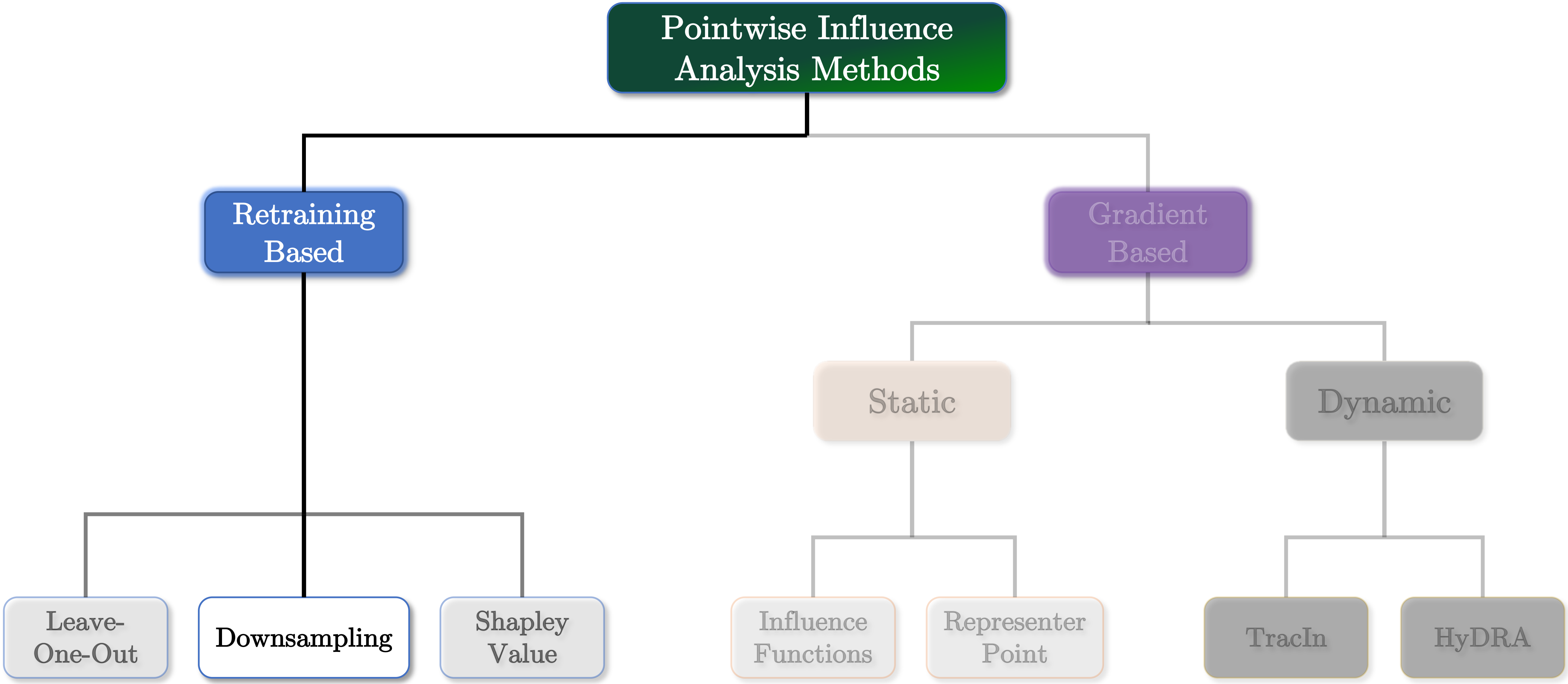
- Large upfront cost
- Complexity dependent on training set size (n)

Question: Can we remove LOO's dependence on n?

Pointwise Influence Analysis Taxonomy



Pointwise Influence Analysis Taxonomy



Method #2: Downsampling [FZ20]

Key Feature: Much faster version of LOO

- random training subsets of size $0.5n$
- Train K models each using a different training subset
- Use K model predictions to **estimate LOO**

- **In Practice**: $K \ll n$

Method #2: Downsampling [FZ20]

nn

nn

$.5nn$

Key **Feature**: Much faster version of LOO

- **Basic Procedure**:
  - Train K models each using a different training subset
  - Use K model predictions to **estimate LOO**
- **In Practice**: $K \ll n$
- **In Practice**: $K \ll n$
- **In Practice**: $K \ll n$

Method #2: Downsampling

Method #2: Downsampling

$K = 8$ Training Subsets

Method #2: Downsampling

$K = 8$ Training Subsets

Method #2: Downsampling

$K = 8$ Training Subsets

Train K Models

1 — f_1

2 — f_2

3 — f_3

4 — f_4

5 — f_5

6 — f_6

7 — f_7

8 — f_8

Method #2: Downsampling

$K = 8$ Training Subsets

Train K Models

Subset	Model
1	f_1
2	f_2
3	f_3
4	f_4
5	f_5
6	f_6
7	f_7
8	f_8

Method #2: Downsampling

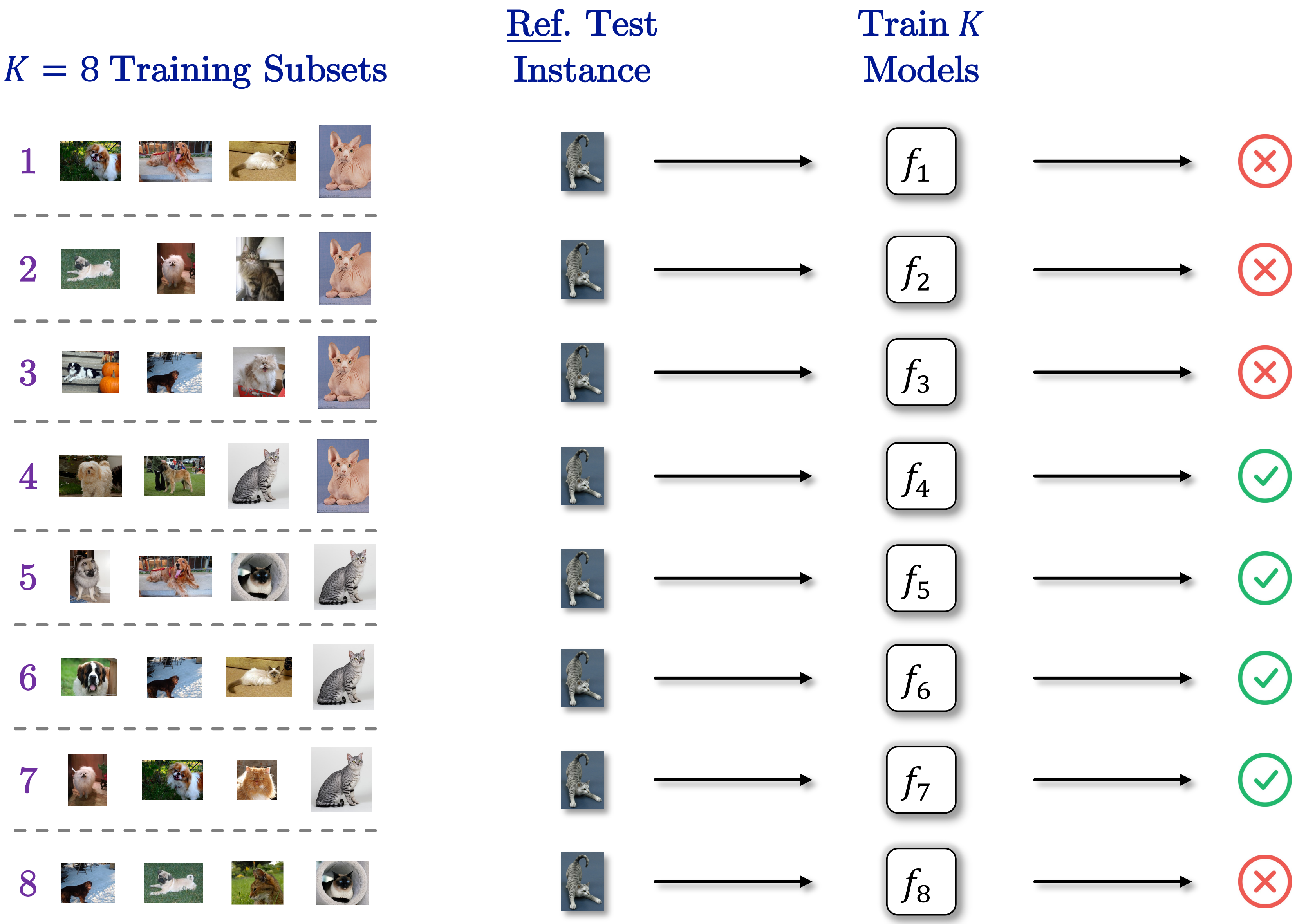
$K = 8$ Training Subsets

Train K Models

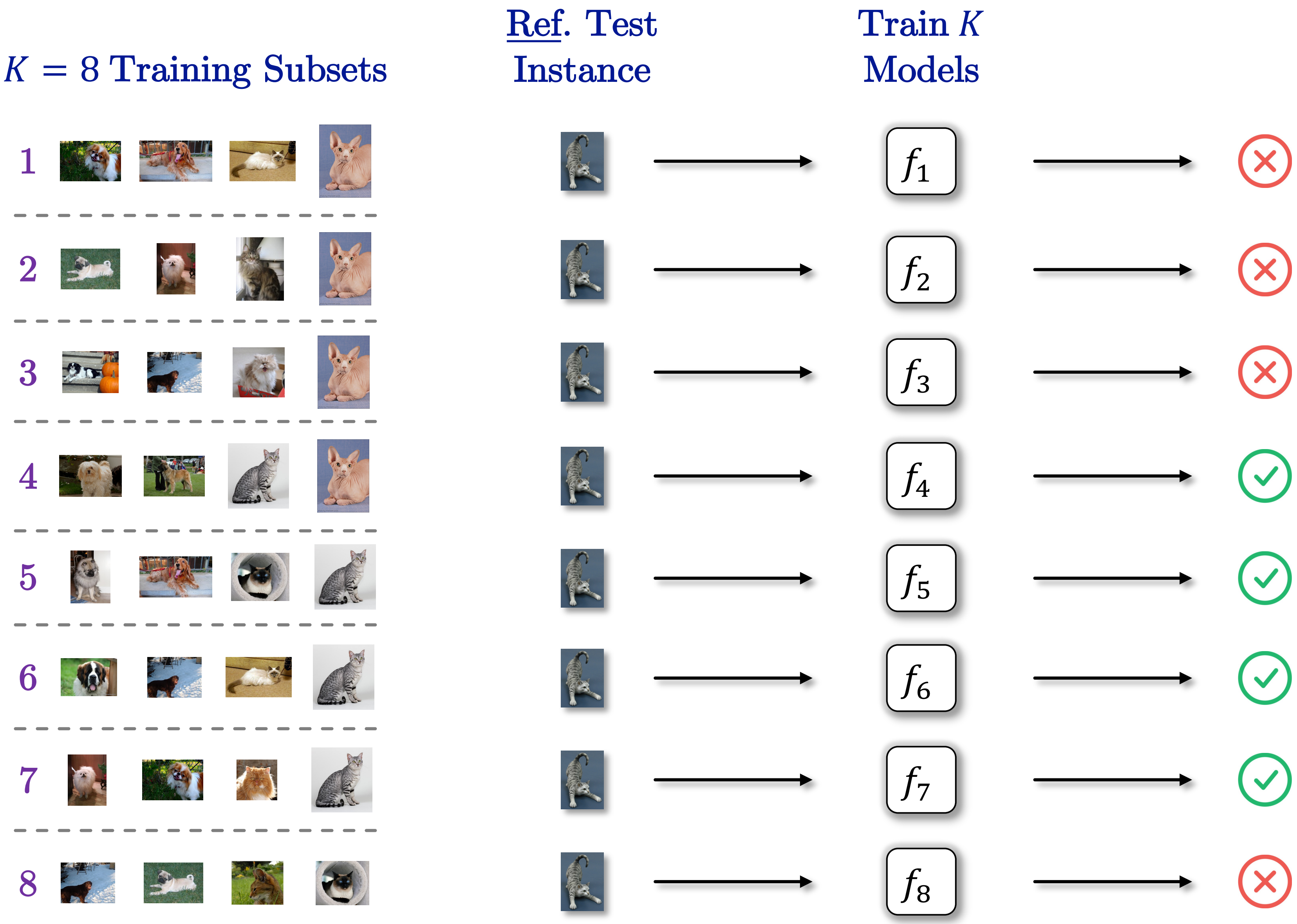
Subset	Model
1	f_1
2	f_2
3	f_3
4	f_4
5	f_5
6	f_6
7	f_7
8	f_8

Method #2: Downsampling

Method #2: Downsampling



Method #2: Downsampling



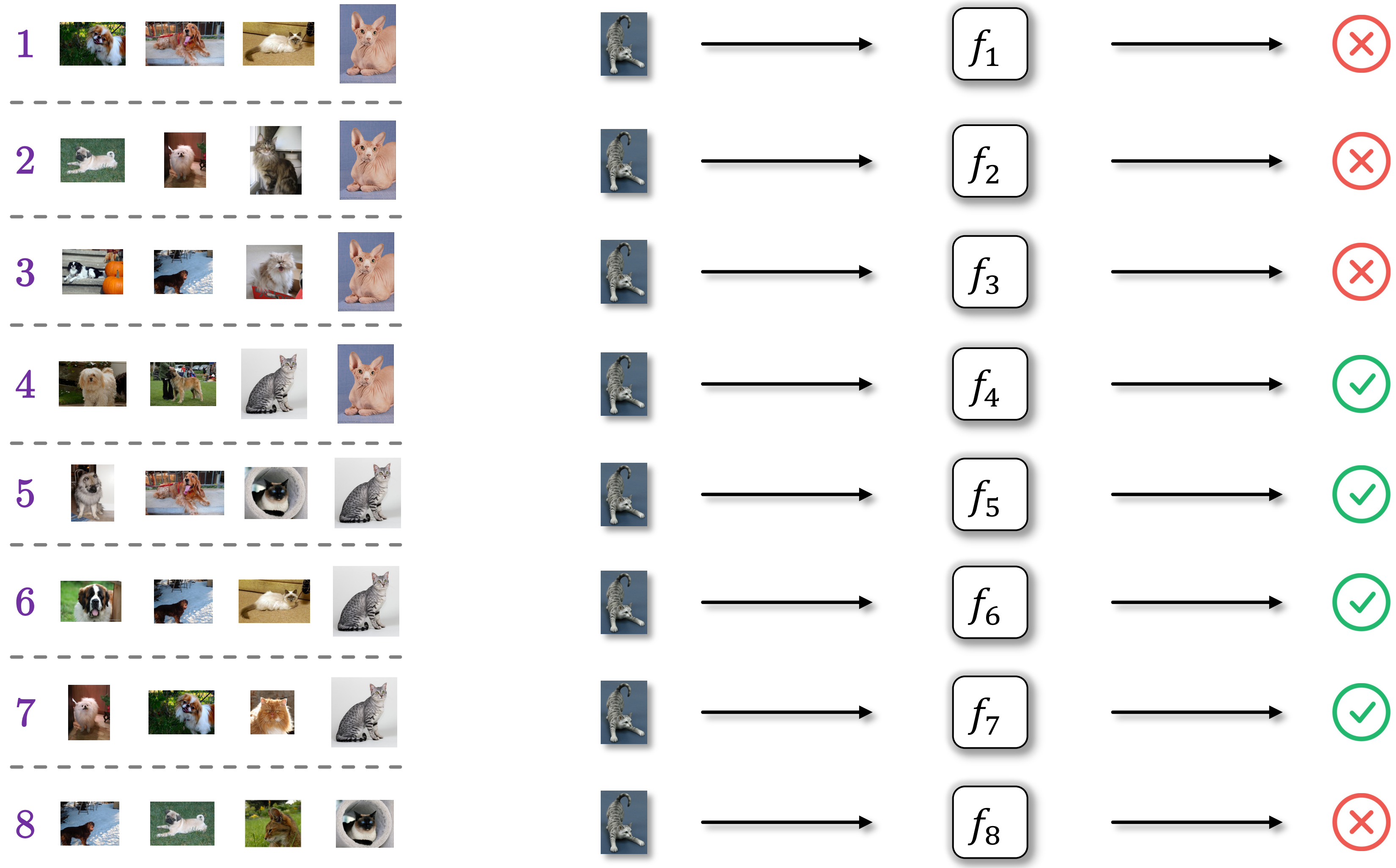
$$\hat{I}_{\text{DS}} = \text{Acc}_{\text{With}} - \text{Acc}_{\text{W/O}}$$

Method #2: Downsampling

$K = 8$ Training Subsets

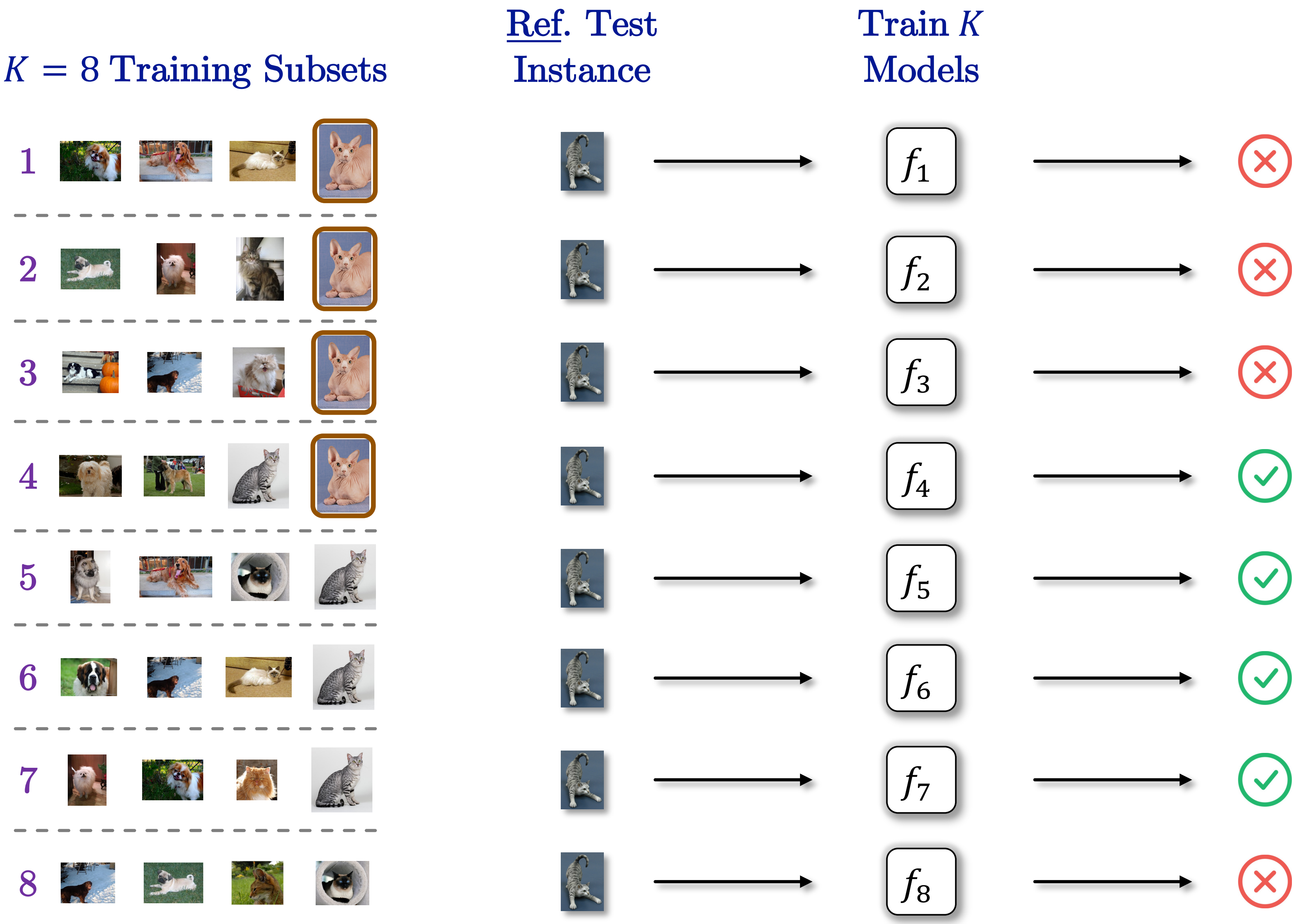
Ref. Test Instance

Train K Models



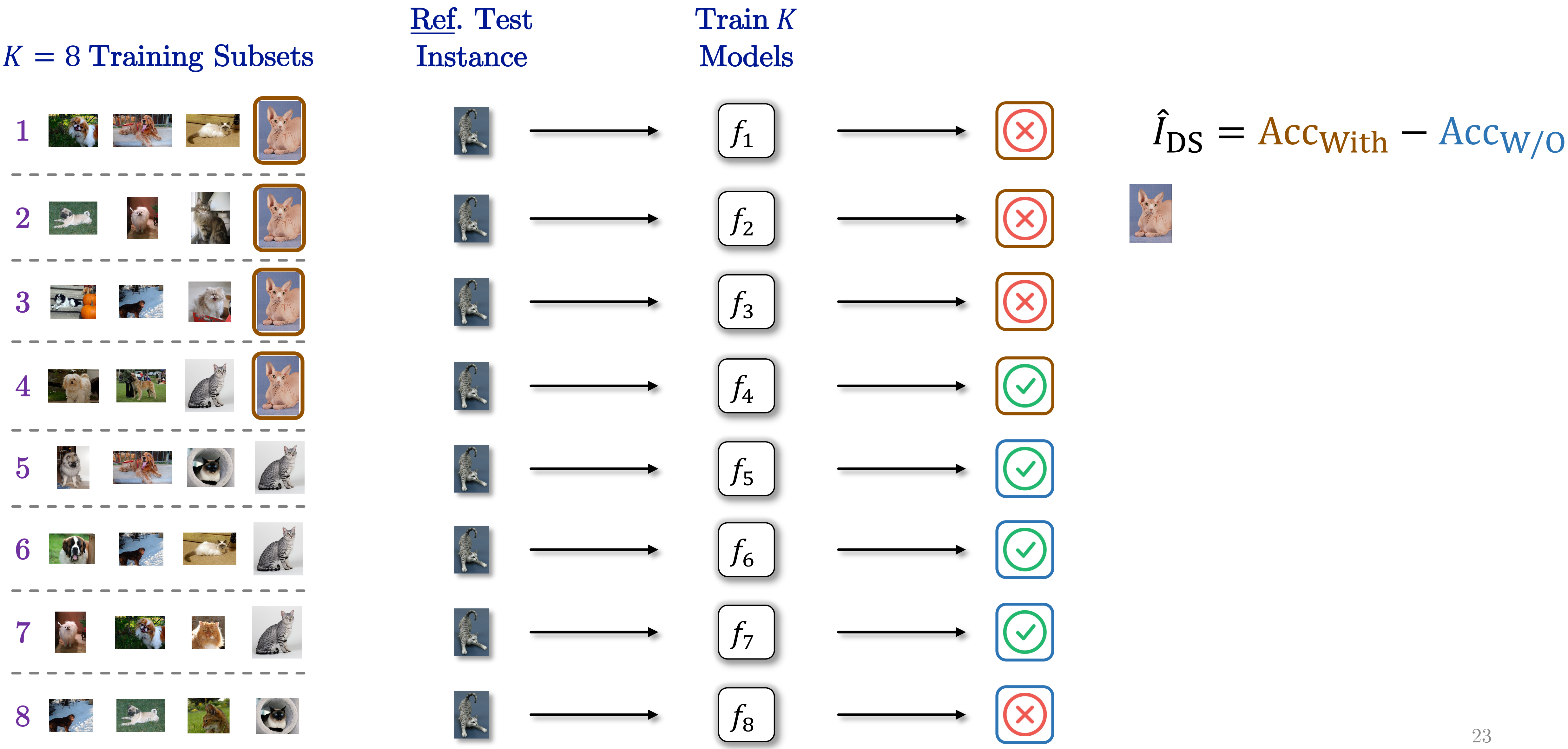
$$\hat{I}_{\mathrm{DS}} = \mathrm{Acc}_{\mathrm{With}} - \mathrm{Acc}_{\mathrm{W/O}}$$

Method #2: Downsampling

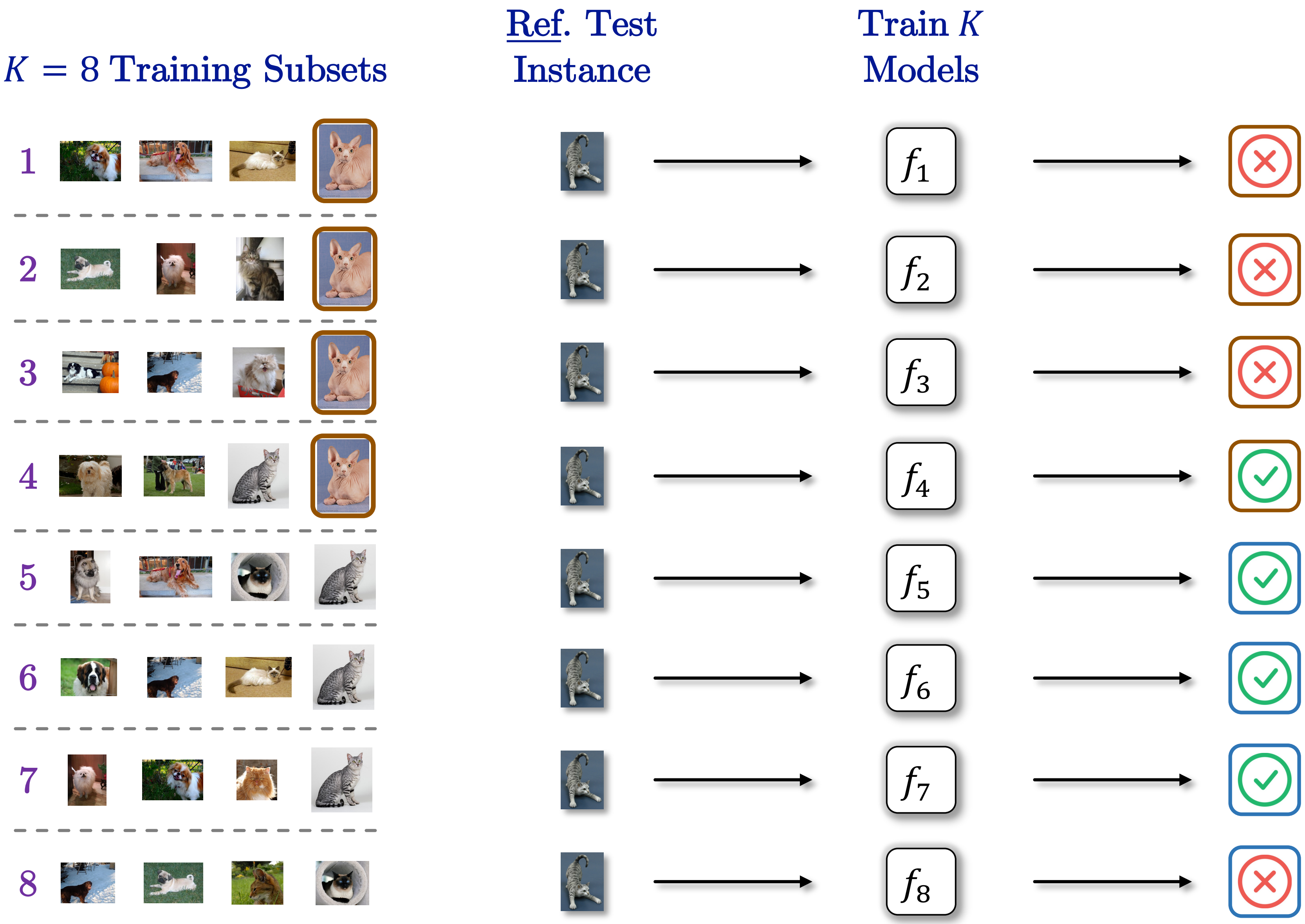


$$\hat{I}_{\text{DS}} = \text{Acc}_{\text{With}} - \text{Acc}_{\text{W/O}}$$

Method #2: Downsampling



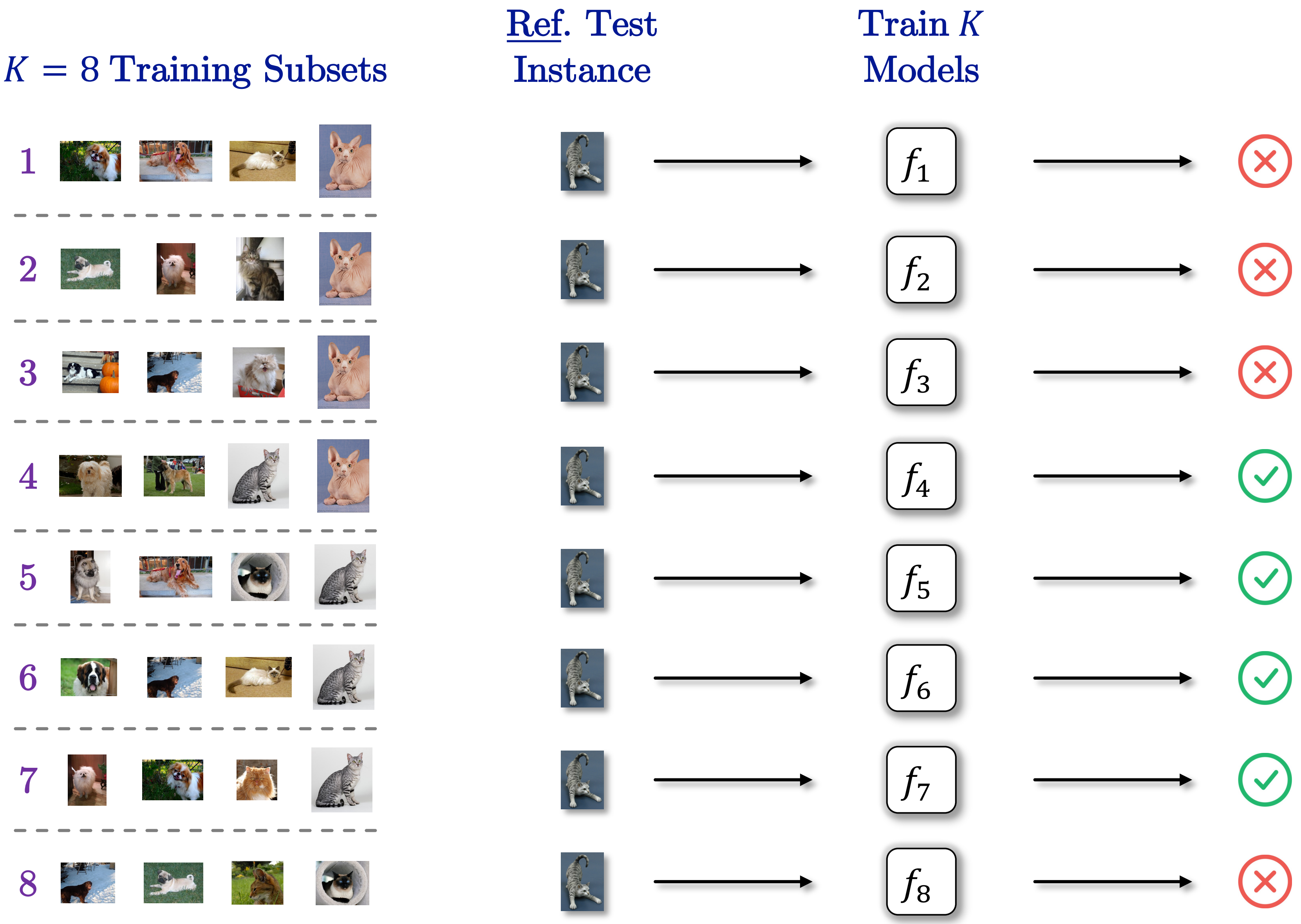
Method #2: Downsampling



$$\hat{I}_{\mathrm{DS}} = \mathrm{Acc}_{\mathrm{With}} - \mathrm{Acc}_{\mathrm{W/O}}$$

 $\hat{I}_{\mathrm{DS}} = \frac{1}{4} - \frac{3}{4} = -0.5$

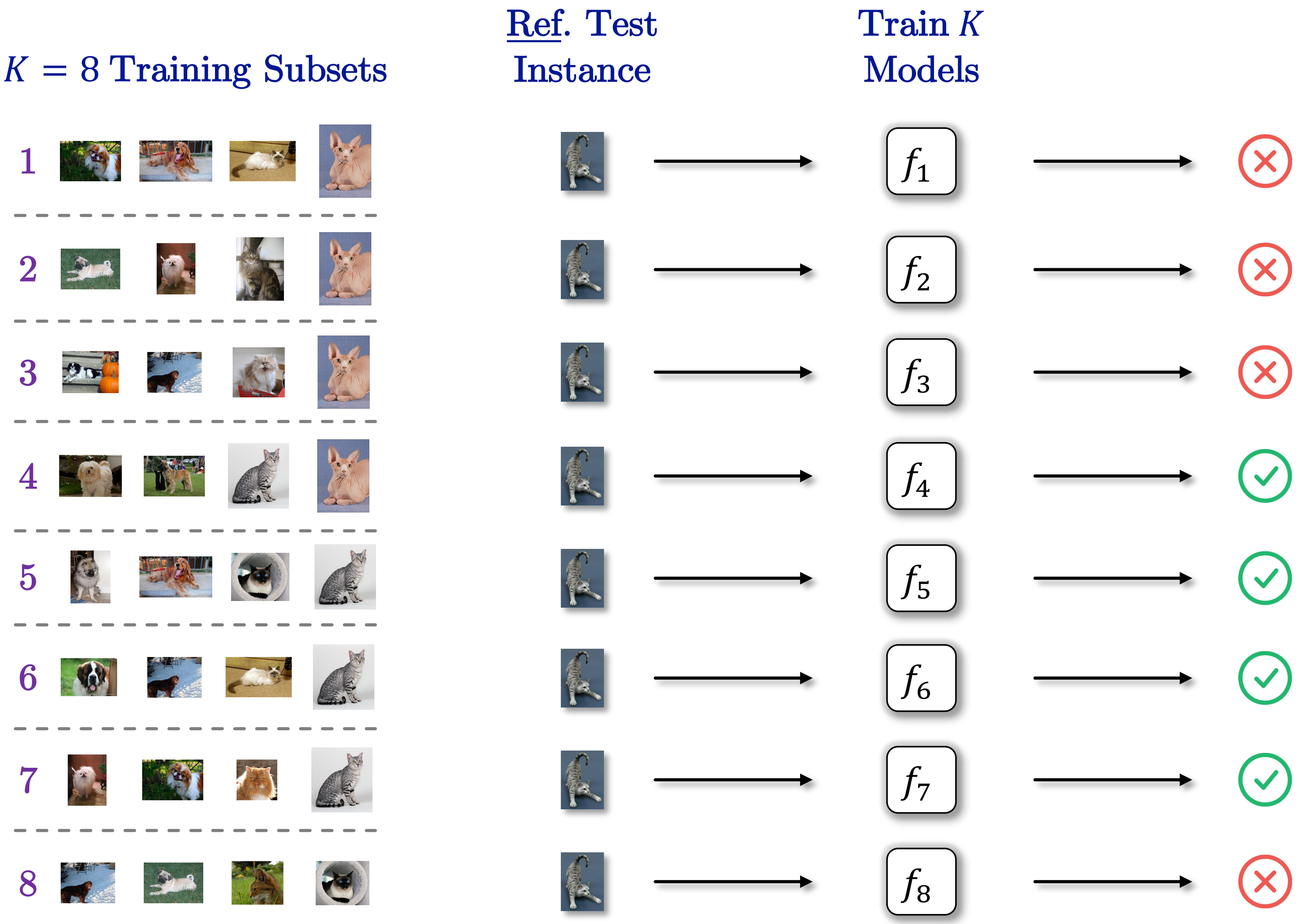
Method #2: Downsampling



$$\hat{I}_{\text{DS}} = \text{Acc}_{\text{With}} - \text{Acc}_{\text{W/O}}$$

$$\hat{I}_{\text{DS}} = \frac{1}{4} - \frac{3}{4} = -0.5$$

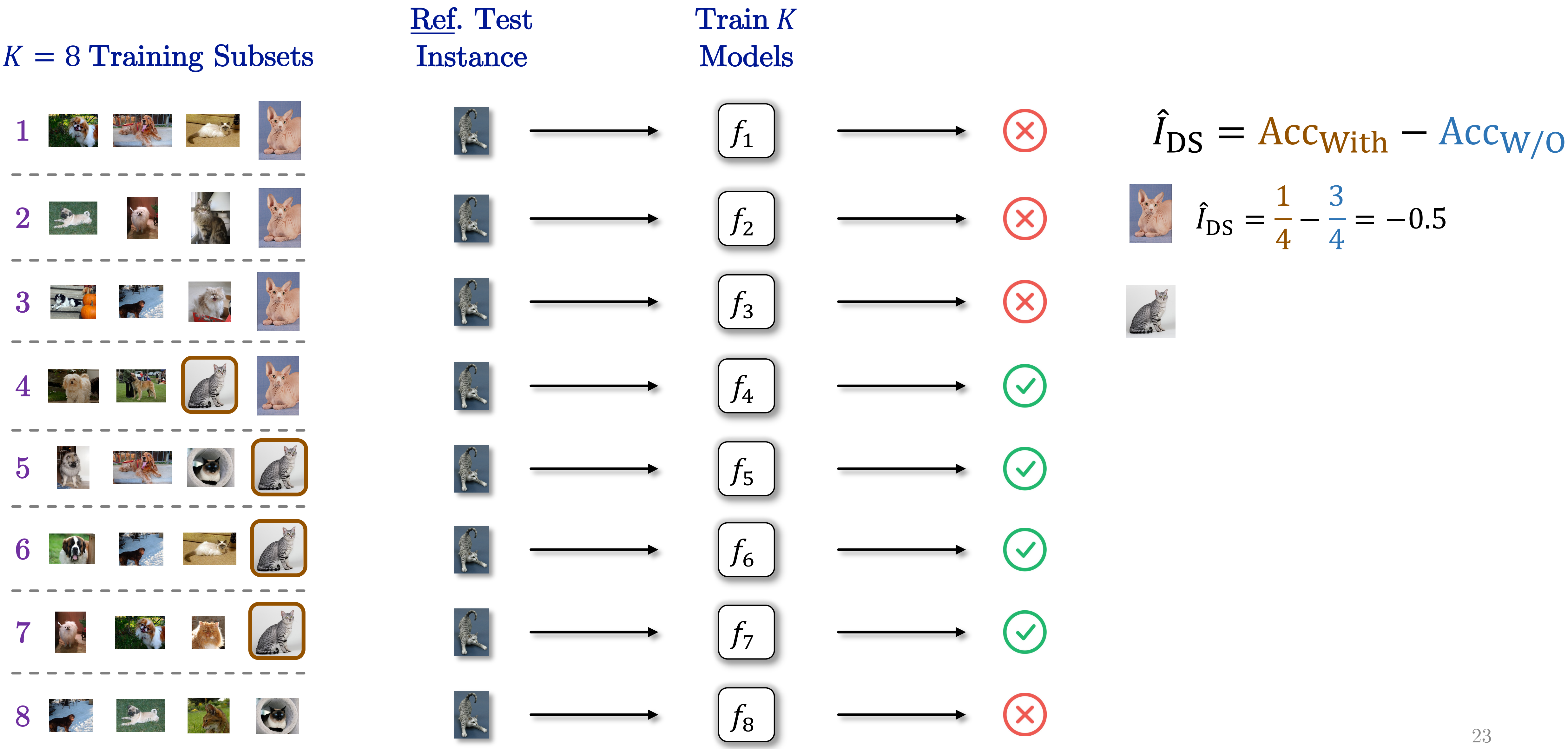
Method #2: Downsampling



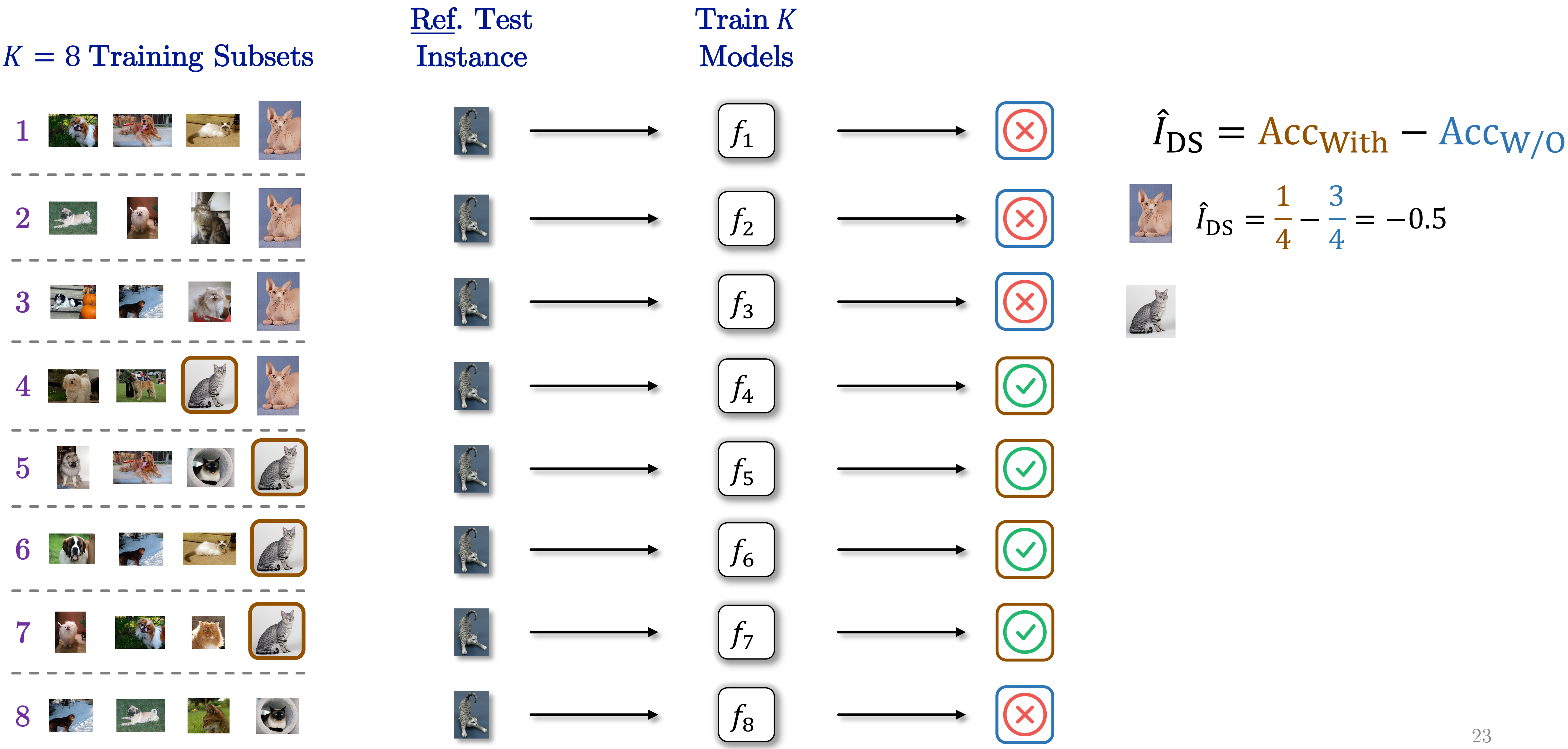
$$\hat{I}_{\mathrm{DS}} = \mathrm{Acc}_{\mathrm{With}} - \mathrm{Acc}_{\mathrm{W/O}}$$

$$\hat{I}_{\mathrm{DS}} = \frac{1}{4} - \frac{3}{4} = -0.5$$

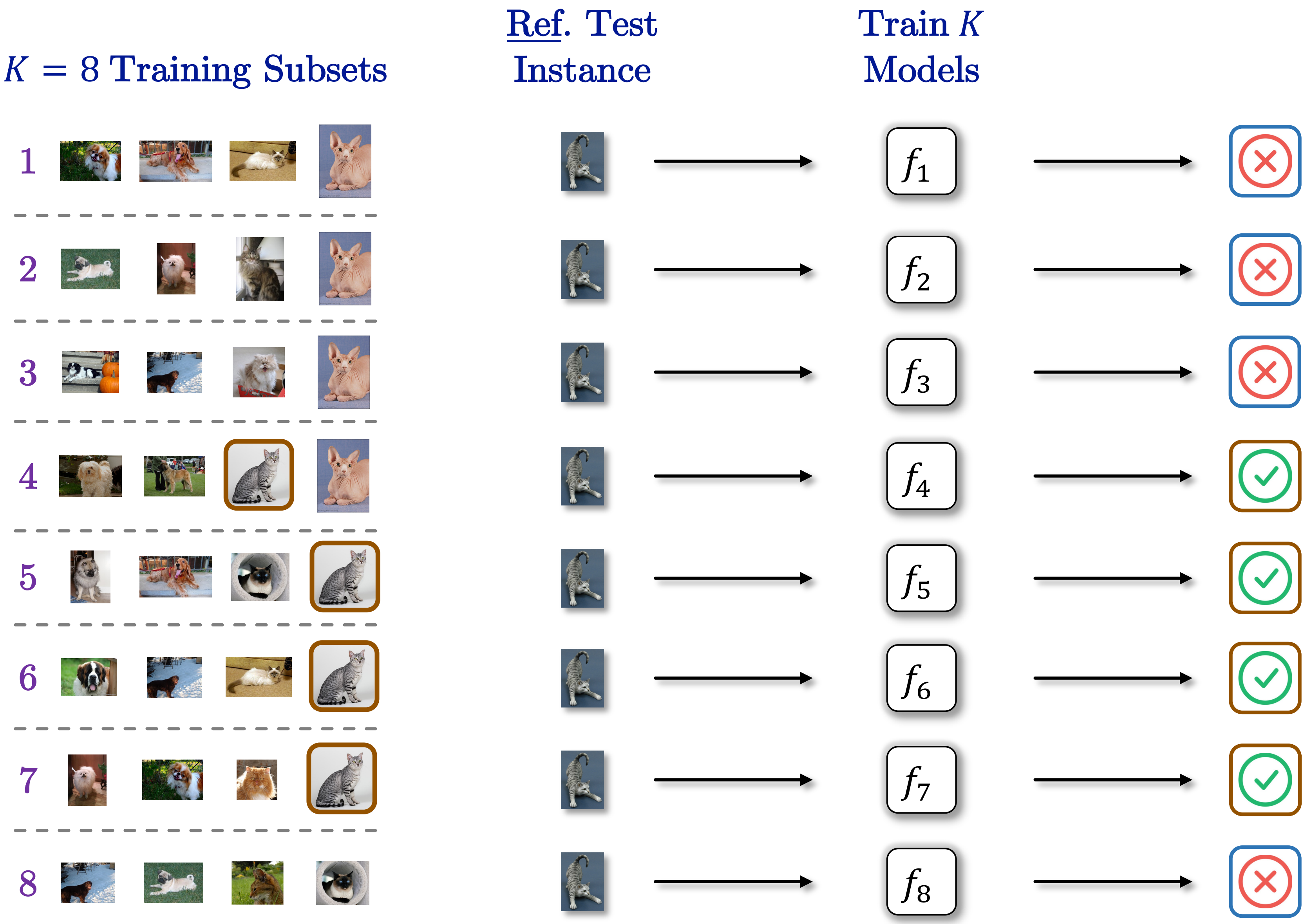
Method #2: Downsampling



Method #2: Downsampling



Method #2: Downsampling



$$\hat{I}_{\text{DS}} = \text{Acc}_{\text{With}} - \text{Acc}_{\text{W/O}}$$

$$\hat{I}_{\text{DS}} = \frac{1}{4} - \frac{3}{4} = -0.5$$

$$\hat{I}_{\text{DS}} = \frac{4}{4} - \frac{0}{4} = 1$$

Downsampling's Complexity

Time Complexity

- Full: $\mathcal{O}(KT)$
- Incremental: $\mathcal{O}(K)$

Storage Complexity: $\mathcal{O}(Kp)$

Space Complexity: $\mathcal{O}(n + p)$

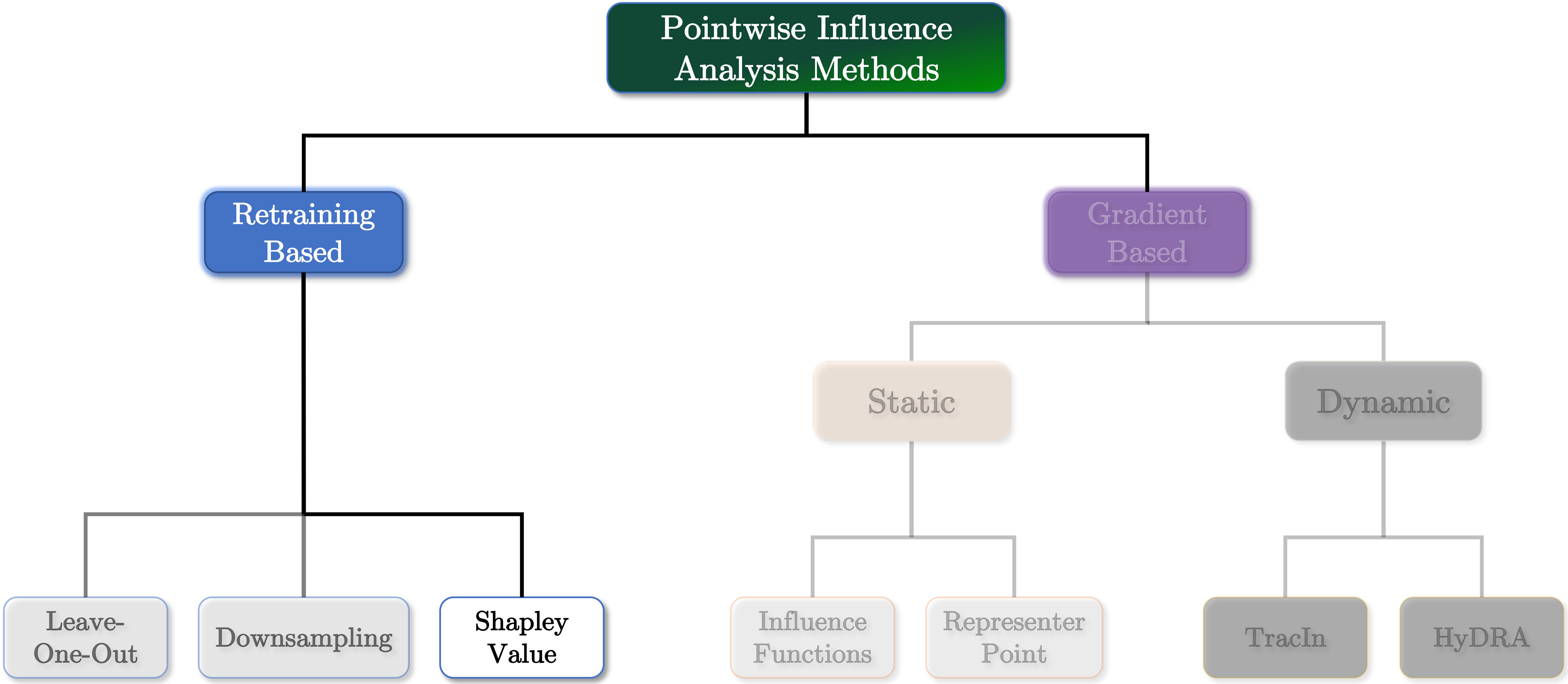
Downsampling: Strengths & Weaknesses

Strengths:

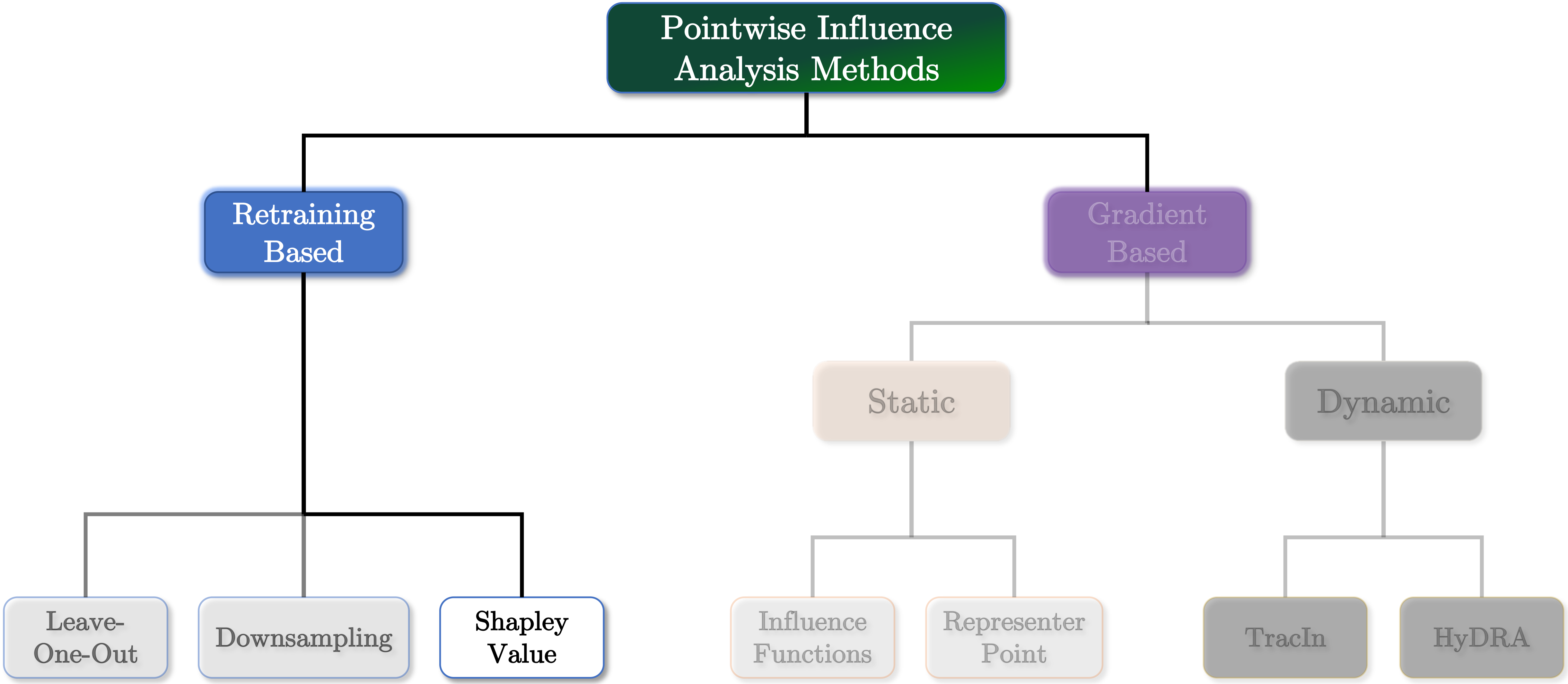
Estimates the expected LOO influence

Considers effect of training's rando

Pointwise Influence Analysis Taxonomy



Pointwise Influence Analysis Taxonomy



Method #3: Retraining + Game Theory

Cooperative Game Theory: Attempts to predict how players in a multiagent game cooperate to achieve shared objectives.

A training set can be viewed as a coalition of n players

- Groups of training instances **cooperate** during model training to improve the model's performance

- Each group of training instances has a "value" – positive or negative

Method #3: Shapley Value [GZ19]

Proposed originally [Sha53] in the context of cooperative game theory

Shapley value influence: Each training instance's average leave-out-influence across <u>**all possible training subsets**</u>

Method #3: Shapley Value [GZ19]

Proposed originally [Sha53] in the context of cooperative game theory

Shapley value influence: Each training instance's average leave-out-influence across **all possible training subsets**

- **Question:** How many possible training subsets are there?

Relating Leave-One-Out & Shapley Value

Recall the Leave-One-Out Influence

$$I_{\mathrm{LOO}} = \mathrm{Acc}_{\mathrm{With}} - \mathrm{Acc}_{\mathrm{W/O}}$$

- $\mathrm{Acc}_{\mathrm{With}}$ is w.r.t. the full training set (size n)
- $\mathrm{Acc}_{\mathrm{W/O}}$: Considers a single training subset (size $n-1$)

Relating Leave-One-Out & Shapley Value

Shapley Value Influence Basic Procedure:

- Train a model on each of the 2^n training subsets
- Calculate the LOO influence across 2^{n-1} LOO model pairs
- Average the 2^{n-1} LOO influences

Relating Leave-One-Out & Shapley Value

Shapley Value Influence Basic Procedure:

- Train a model on each of the 2^n training subsets
- Calculate the LOO influence across 2^{n-1} LOO model pairs
- Average the 2^{n-1} LOO influences

🚨 Exponential time

Relating Leave-One-Out & Shapley Value

Shapley Value Influence Basic Procedure:

- Train a model on each of the 2^n training subsets
- Calculate the LOO influence across 2^{n-1} LOO model pairs
- Average the 2^{n-1} LOO influences

🚨 **Exponential time**

- **Provably hard**: #P Complete

Shapley Value's Complexity

Time Complexity

- Full: $\mathcal{O}(2^n\, T)$
- Incremental: $\mathcal{O}(2^n)$

Storage Complexity: $\mathcal{O}(2^n p)$

Space Complexity: $\mathcal{O}(n + p)$

Significant follow-on work has focused on heuristically speeding up Shapley value estimation.

Shapley Value: Strengths & Weaknesses

Strengths:

+ Extensively studied and well motivated theoretically
+ Detects “difficult” test instances that other retraining-based methods may miss

Weaknesses:

- Catastrophic execution time

Big Picture Summary:
Retraining-Based Influence Analysis

Big Picture Summary:
Retraining-Based Influence Analysis

Strengths:

+ Simple. Works for any model class with few assumptions

+ Intuitive and human interpretable

+ Low incremental time complexity

Weaknesses:

- Huge storage and upfront time complexities

Multiple retrainings are **expensive**.

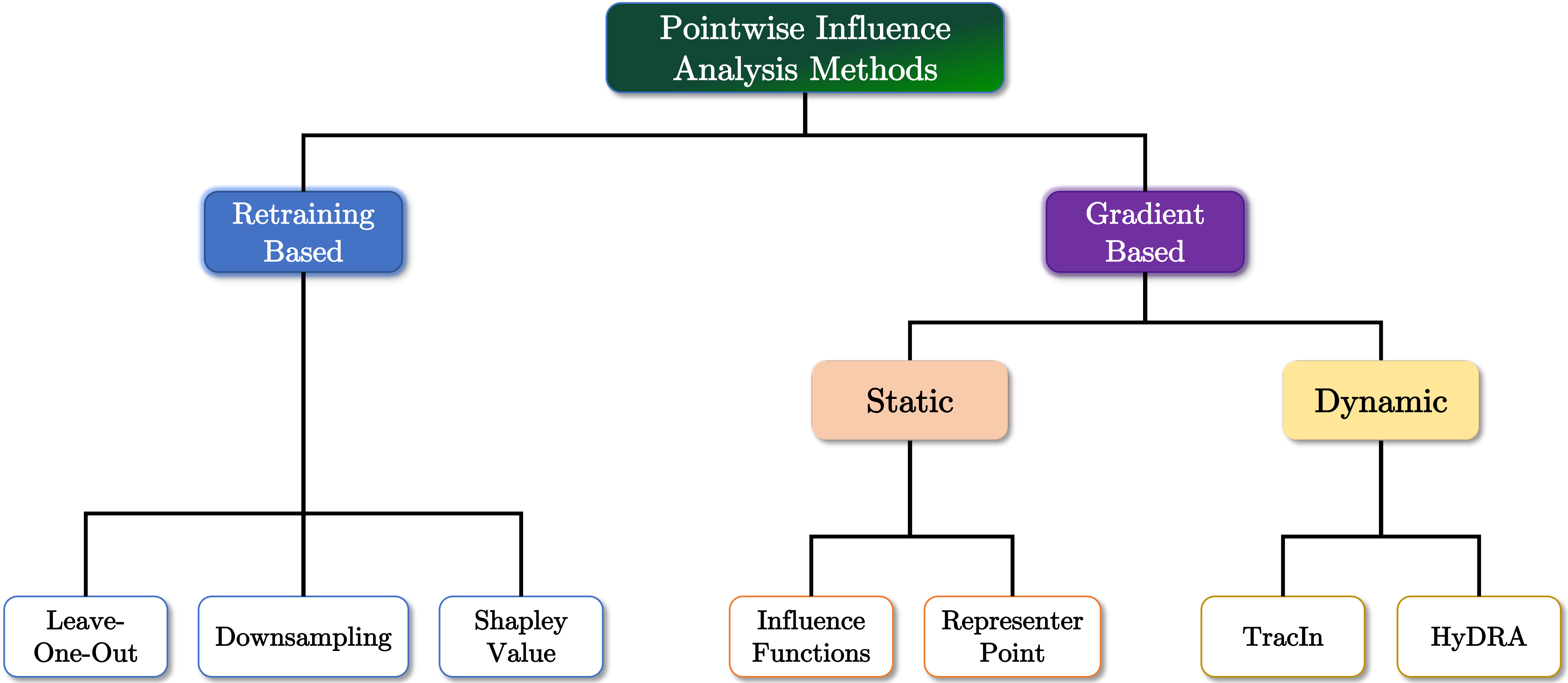
- For large models, repeated retraining is even prohibitive.

Multiple retrainings are **expensive**.

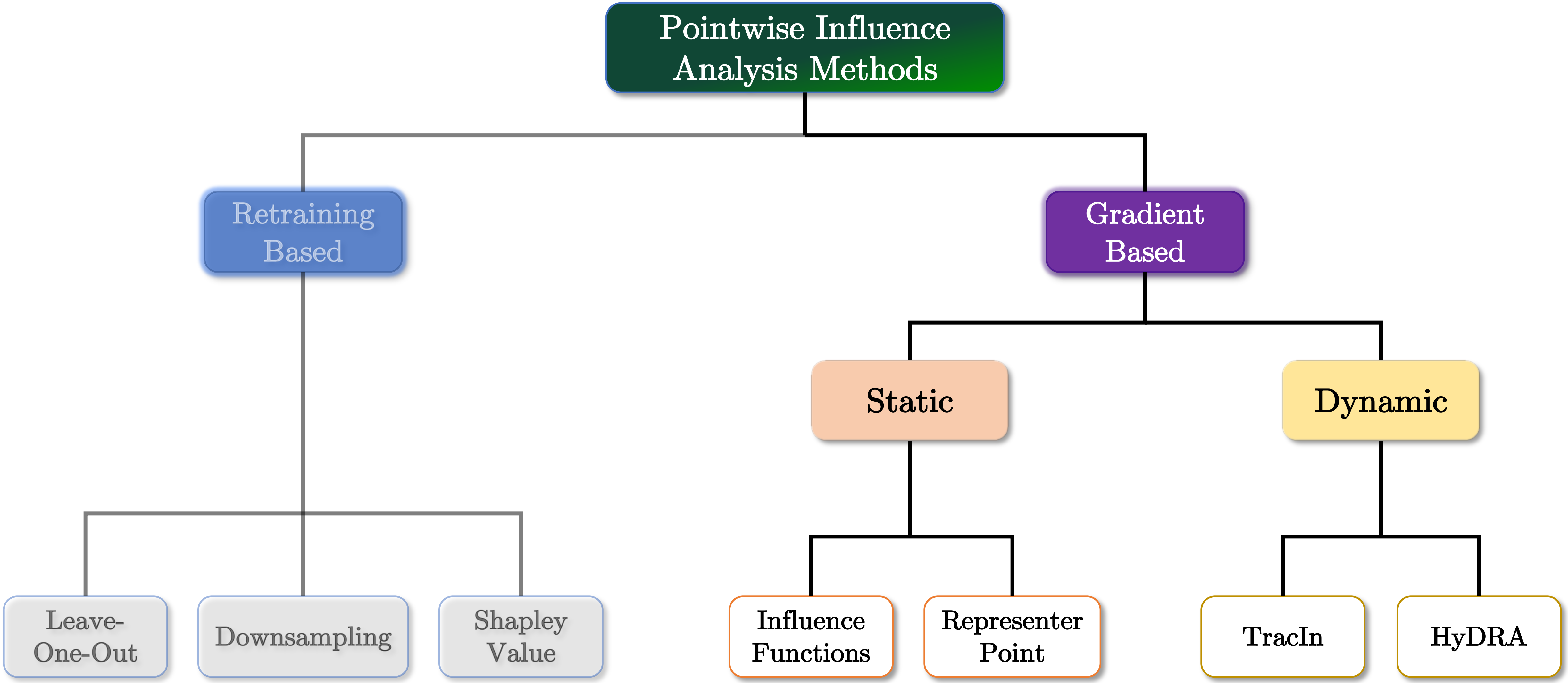
- For large models, repeated retraining is even prohibitive.

Question: Can training-set influence be accurately measured without any model retraining?

Pointwise Influence Analysis Taxonomy



Pointwise Influence Analysis Taxonomy



Gradient-Based Influence Analysis

Why Use Gradients to Estimate Influence?

Training instances only affect the model through gradients

- Intuition: Training gradients have everything we need to measure influence

Key Themes for Gradient-Based Influence Estimation:

- **No model retraining**
- Model and loss assumed differentiable
- Rely on Taylor-series expansions

Partitioning Gradient-Based Influence Estimators

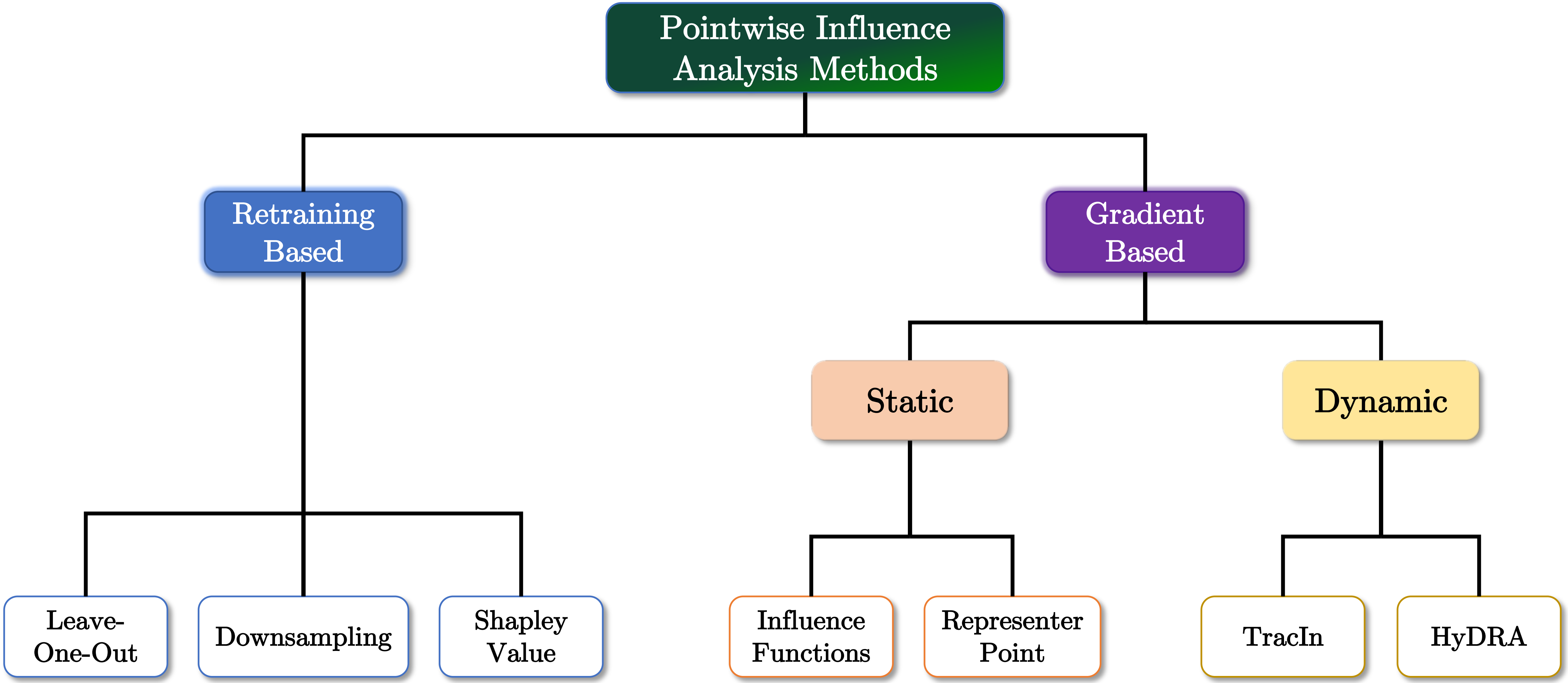
Static Estimators: Estimate influence using <u>only the final model parameters</u>

- **Influence Functions** [KL17]
- **Representer Point** [Yeh+18]

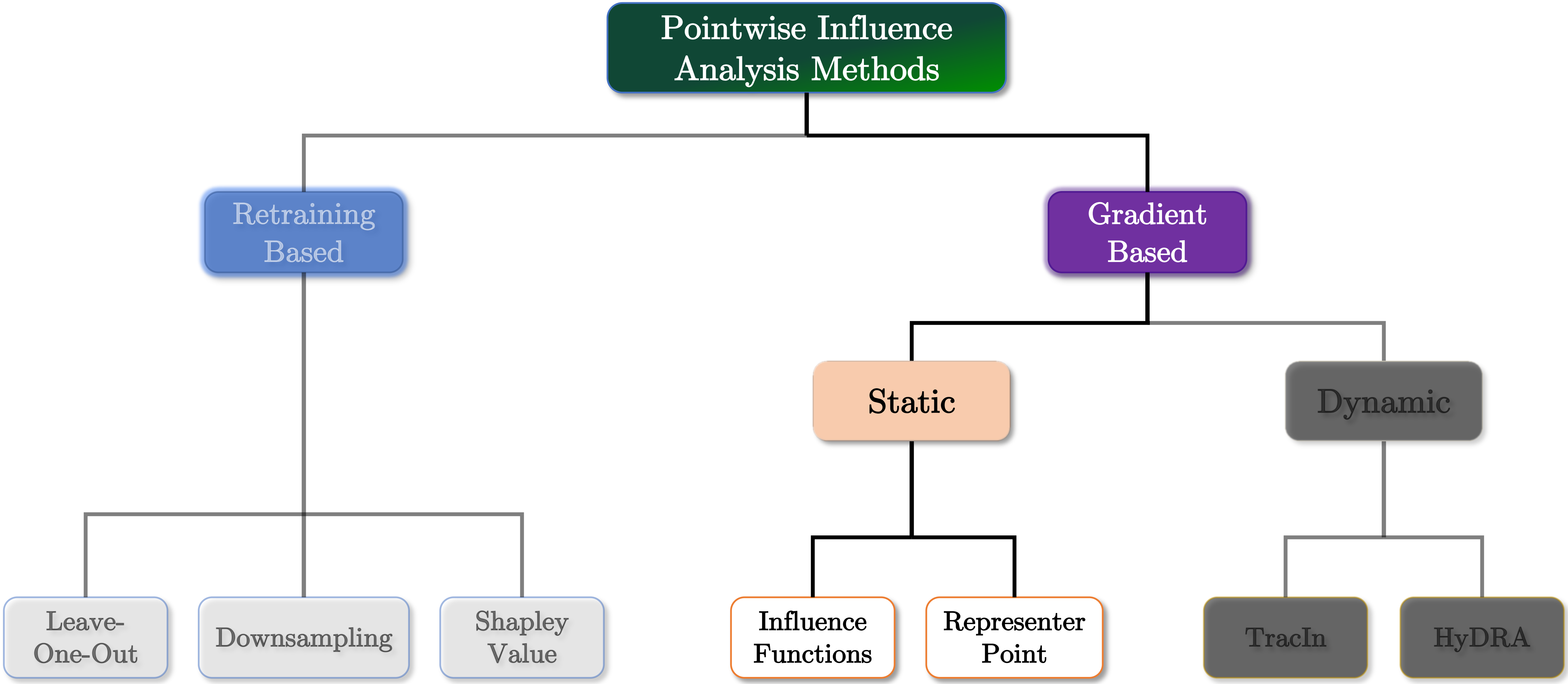
Dynamic Estimators: Reconstruct the training set's influence by studying <u>all model parameters across all training iterations</u>

- **TracIn** [Pru+20]
- **HyDRA** [Che+21]

Pointwise Influence Analysis Taxonomy



Pointwise Influence Analysis Taxonomy



How Can Static Influence Estimation Actually Work?

The final model parameters contain **very limited information**

- Much less information than is used by retraining-based as well as dynamic gradient-based methods

Rule of Thumb: Static influence estimators are simpler than dynamic ones

How Can Static Influence Estimation Actually Work?

The final model parameters contain **very limited information**

- Much less information than is used by retraining-based as well as dynamic gradient-based methods

Rule of Thumb: Static influence estimators are simpler than dynamic ones

No Free Lunch: Static estimators make **very strong assumptions** that do not hold for deep models

Method #4: Influence Functions [KL17]

Best-known and most well-studied influence estimator

- Based on influence functions from robust statistics [Jae72, Ham74]

Estimates the Leave-One-Influence:

$$I_{\text{LOO}} = \text{Loss}_{\text{W/O}} - \text{Loss}_{\text{With}} \approx \hat{I}_{\text{IF}}$$

Very Strong Assumption: Strict convexity and stationarity

Method #4: Influence Functions [KL17]

Best-known and most well-studied influence estimator

- Based on influence functions from robust statistics [Jae72, Ham74]

Estimates the Leave-One-Influence:

$$I_{\mathrm{LOO}} = \mathrm{Loss}_{\mathrm{W/O}} - \mathrm{Loss}_{\mathrm{With}} \approx \hat{I}_{\mathrm{IF}}$$

Very Strong Assumption: Strict convexity and stationarity

+ Enables a closed-form LOO influence estimate

Visualizing Influence Functions

Visualizing Influence Functions

$\hat{I}_{\mathrm{IF}} = \mathrm{Loss}_{\mathrm{W/O}} - \mathrm{Loss}_{\mathrm{With}}$

Visualizing Influence Functions

$$\hat{I}_{\text{IF}} = \text{Loss}_{\text{W/O}} - \text{Loss}_{\text{With}}$$

Full Training Set:

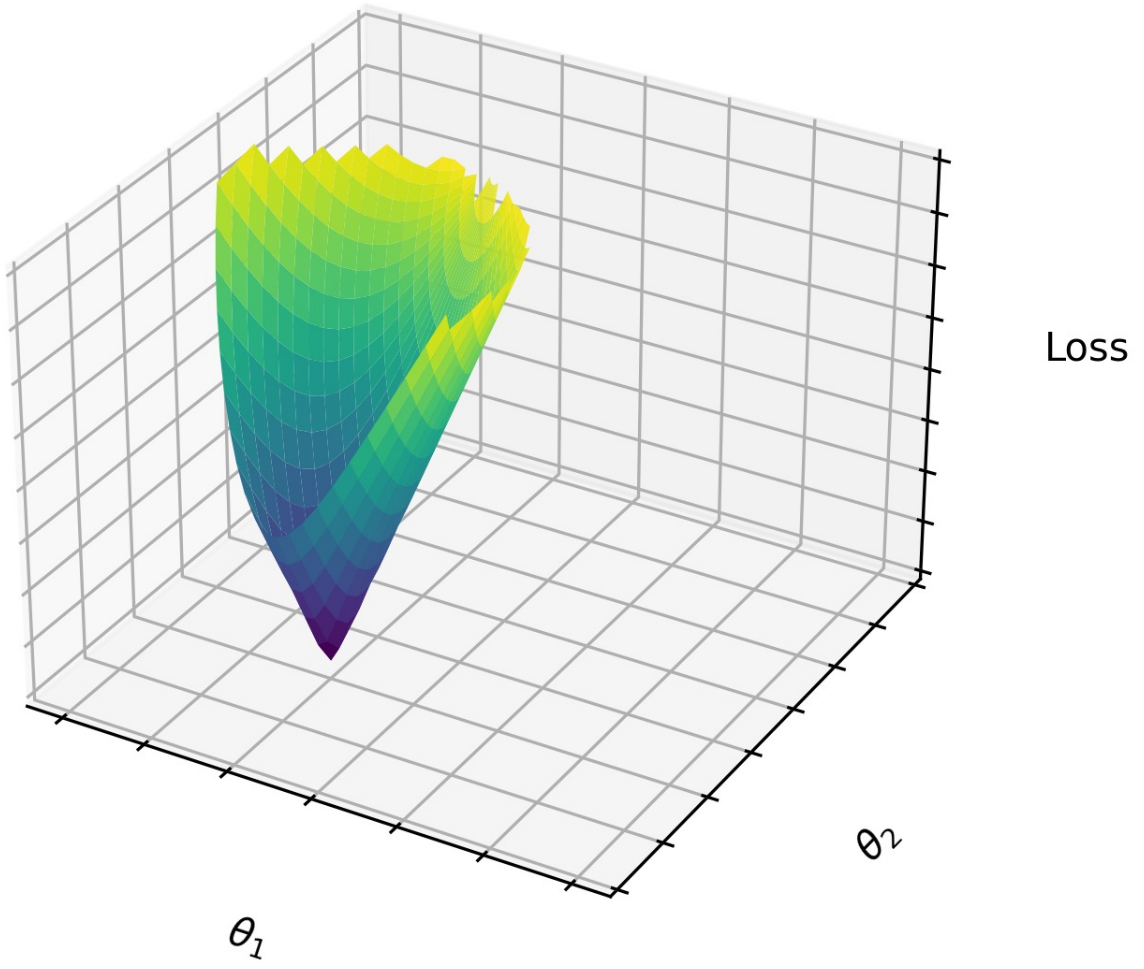
$\text{Loss}_{\text{With}} = Loss(f($  $; \theta), \text{Cat})$

Visualizing Influence Functions

$$\hat{I}_{\text{IF}} = \text{Loss}_{\text{W/O}} - \text{Loss}_{\text{With}}$$

Full Training Set:

$\text{Loss}_{\text{With}} = Loss(f($  $;\theta), \text{Cat})$



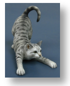
Visualizing Influence Functions

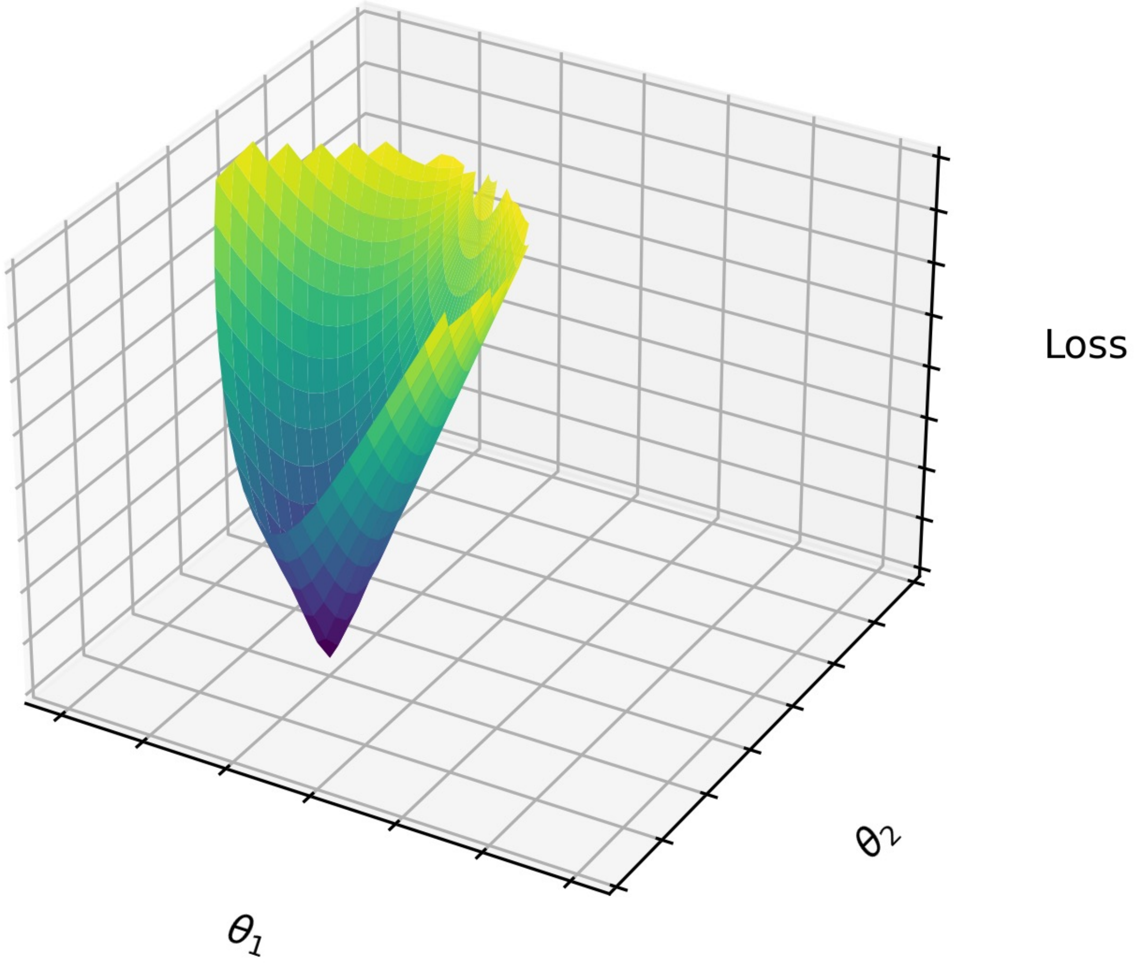
$$\hat{I}_{\text{IF}} = \text{Loss}_{\text{W/O}} - \text{Loss}_{\text{With}}$$

Full Training Set:

$\text{Loss}_{\text{With}} = Loss(f($  $;\theta), \text{Cat})$

Leave  **Out:**

$\text{Loss}_{\text{W/O}} = Loss(f($  $;\boldsymbol{\theta'}), \text{Cat})$



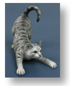
Visualizing Influence Functions

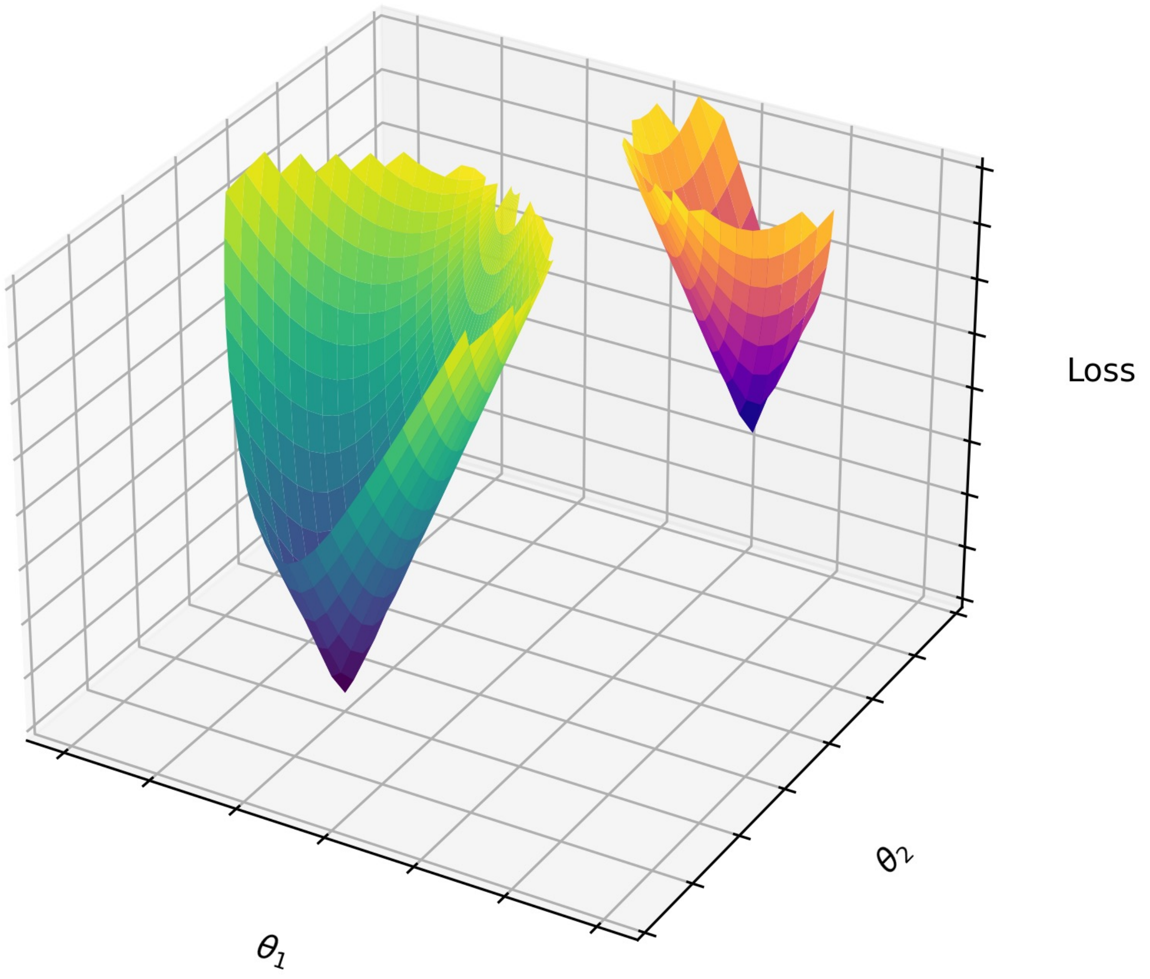
$$\hat{I}_{\text{IF}} = \text{Loss}_{\text{W/O}} - \text{Loss}_{\text{With}}$$

Full Training Set:

$\text{Loss}_{\text{With}} = Loss(f($  $;\theta),$ Cat$)$

Leave  Out:

$\text{Loss}_{\text{W/O}} = Loss(f($  $;\boldsymbol{\theta'}),$ Cat$)$



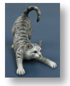
Visualizing Influence Functions

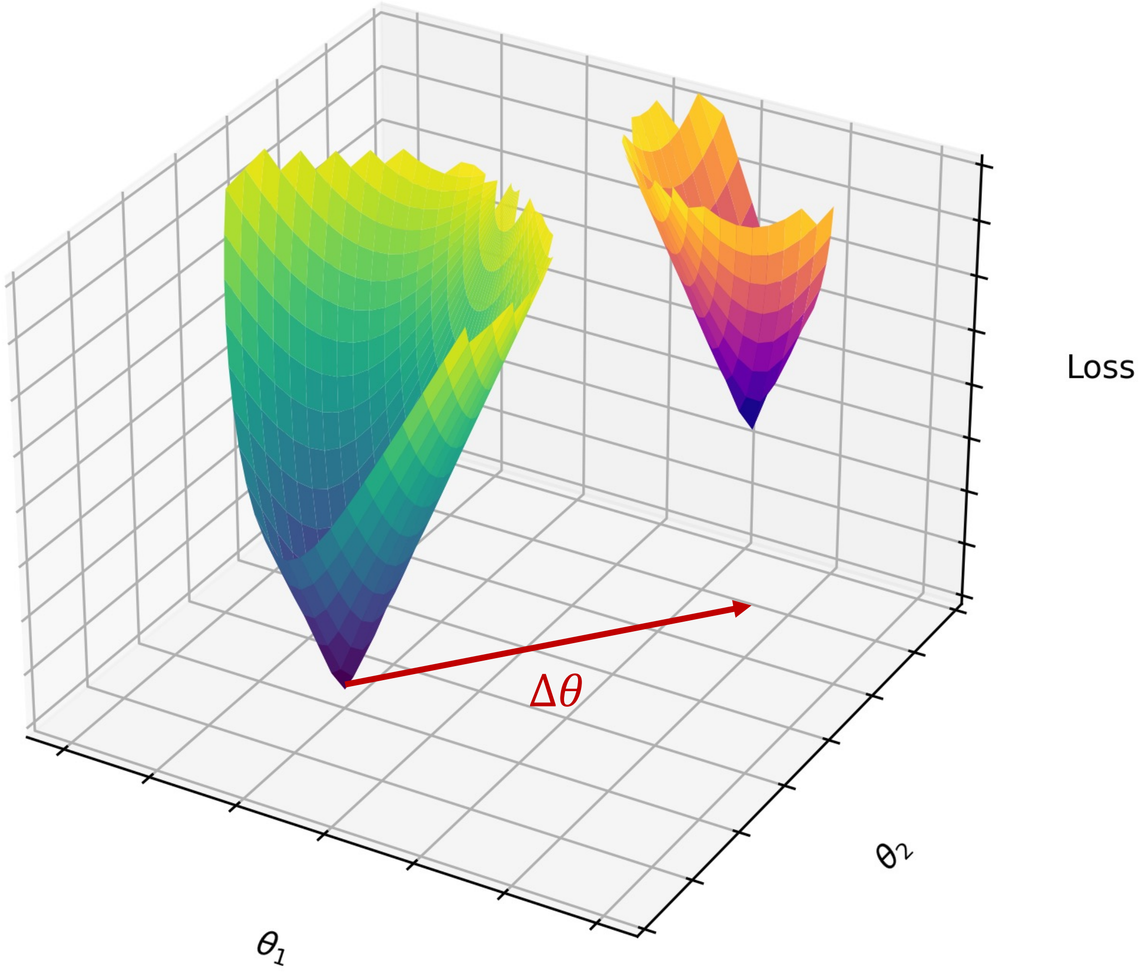
$$\hat{I}_{\mathrm{IF}} = \mathrm{Loss}_{\mathrm{W/O}} - \mathrm{Loss}_{\mathrm{With}}$$

Full Training Set:

$\mathrm{Loss}_{\mathrm{With}} = Loss(f($  $;\theta),$ Cat)

Leave  **Out:**

$\mathrm{Loss}_{\mathrm{W/O}} = Loss(f($  $;\boldsymbol{\theta'}),$ Cat)



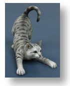
Visualizing Influence Functions

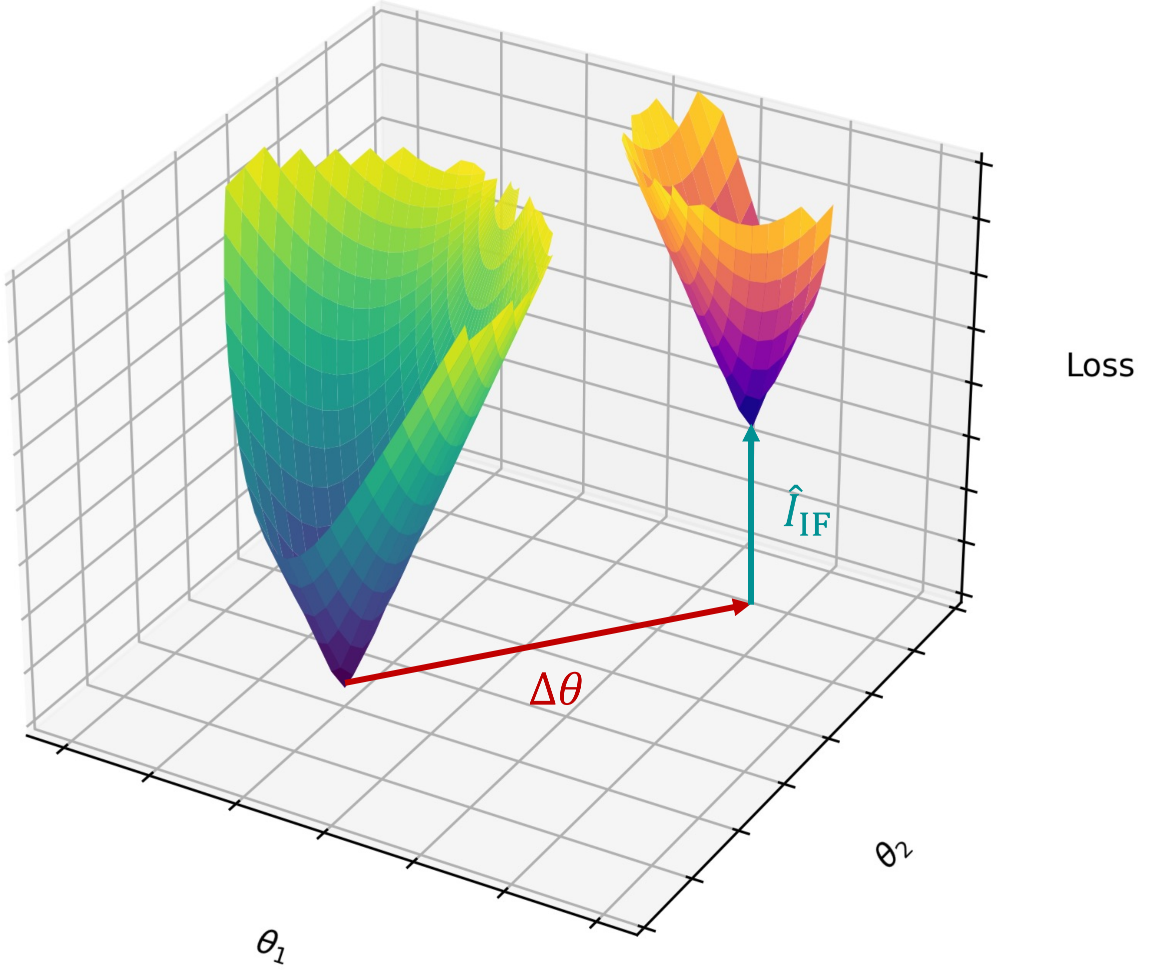
$$\hat{I}_{\mathrm{IF}} = \mathrm{Loss}_{\mathrm{W/O}} - \mathrm{Loss}_{\mathrm{With}}$$

Full Training Set:

$\mathrm{Loss}_{\mathrm{With}} = Loss(f($  $;\theta), \mathrm{Cat})$

Leave  **Out:**

$\mathrm{Loss}_{\mathrm{W/O}} = Loss(f($  $;\boldsymbol{\theta'}), \mathrm{Cat})$



Formalizing Influence Functions

We need to extend our nomenclature a bit

- f: Neural network
- θ: Final model parameters
- L: Loss function
- z_i: Training instance
- z_{te}: Test instance

Influence Functions Closed Form

LOO estimate of <u>**training instance**</u> z_i influence on <u>**test instance**</u> z_{te}

$$\hat{I}_{\mathrm{IF}} \coloneqq \frac{1}{n} \nabla_\theta L(z_i; \theta)\ H_\theta^{-1}\ \nabla_\theta L(z_{\mathrm{te}}; \theta)$$

Training Gradient

Inverse Empirical Risk Hessian

Test Gradient

Influence Functions Closed Form

LOO estimate of **training instance** z_i influence on **test instance** z_{te}

$$\hat{I}_{\text{IF}} \coloneqq \frac{1}{n} \nabla_\theta L(z_i;\theta)\, H_\theta^{-1}\, \nabla_\theta L(z_{\text{te}};\theta)$$

Training Gradient

Inverse Empirical Risk Hessian

Test Gradient

Influence Functions Closed Form

LOO estimate of <u>**training instance**</u> z_i influence on <u>**test instance**</u> z_{te}

$$\hat{I}_{\mathrm{IF}} \coloneqq \frac{1}{n} \underbrace{\nabla_\theta L(z_i;\theta)}_{\text{Training Gradient}} \; \underbrace{H_\theta^{-1}}_{\text{Inverse Empirical Risk Hessian}} \; \underbrace{\nabla_\theta L(z_{\mathrm{te}};\theta)}_{\text{Test Gradient}}$$

Influence Functions Closed Form

LOO estimate of **training instance** z_i influence on **test instance** z_{te}

$$\hat{I}_{\text{IF}} \coloneqq \frac{1}{n} \nabla_\theta L(z_i;\theta)\, \boxed{H_\theta^{-1}\, \nabla_\theta L(z_{\text{te}};\theta)}$$

Training Gradient

Inverse Empirical Risk Hessian

Test Gradient

Influence Functions & Risk Hessians

Calculating H_θ^{-1} exactly is intractable

- **Time Complexity**: $\mathcal{O}(np^2 + p^3)$
- **Space**: $\mathcal{O}(p^2)$

$$\nabla_\theta L(z_{\text{te}}; \theta)$$

- Still slow - $O(np)$ time complexity
- Very inaccurate and unstable in deep models [BPF21,ZZ22,Bae+22]

Influence Functions & Risk Hessians

nnpp) time complexity

1 *H θ −1 ∇ θ ∇ ∇ θ θθ ∇ θ LL z te ;θ z te zz z te* te z te ;θθ z te ;θ

Calculating H_{θ}^{-1} exactly is intractable

- **Time Complexity**: $\mathcal{O}(np^2 + p^3)$
- **Space**: $\mathcal{O}(p^2)$

- Very inaccurate and unstable in deep models [BPF21,ZZ22,Bae+22] Still slow - $O(np)$ time complexity
- Very inaccurate and unstable in deep models [BPF21,ZZ22,Bae+22]

Influence Functions' Complexity

Time Complexity

- Full & Incremental: $\mathcal{O}(np)$

Storage Complexity: $\mathcal{O}(1)$

Space Complexity: $\mathcal{O}(n + p)$

Influence Functions: Strengths & Weaknesses

Strengths:

+ Lower upfront time complexity than retraining-based methods
+ Most well-studied gradient-based influence method

Weaknesses:

- HVP estimation is inaccurate, fragile, and numerically unstable
- Relies on assumptions that do not hold for deep models
- No incremental time complexity gains (slow)

Method #5: Representer Point [Yeh+18]

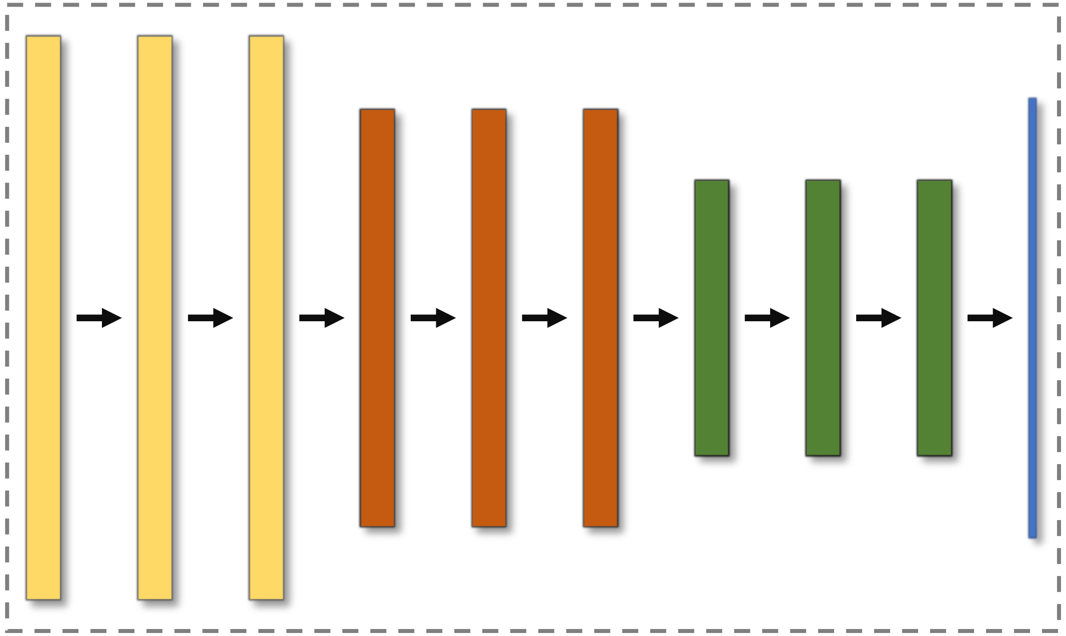
- **Key Feature**: Fastest influence analysis method

- **Strong Assumptions**: Linearity and stationarity

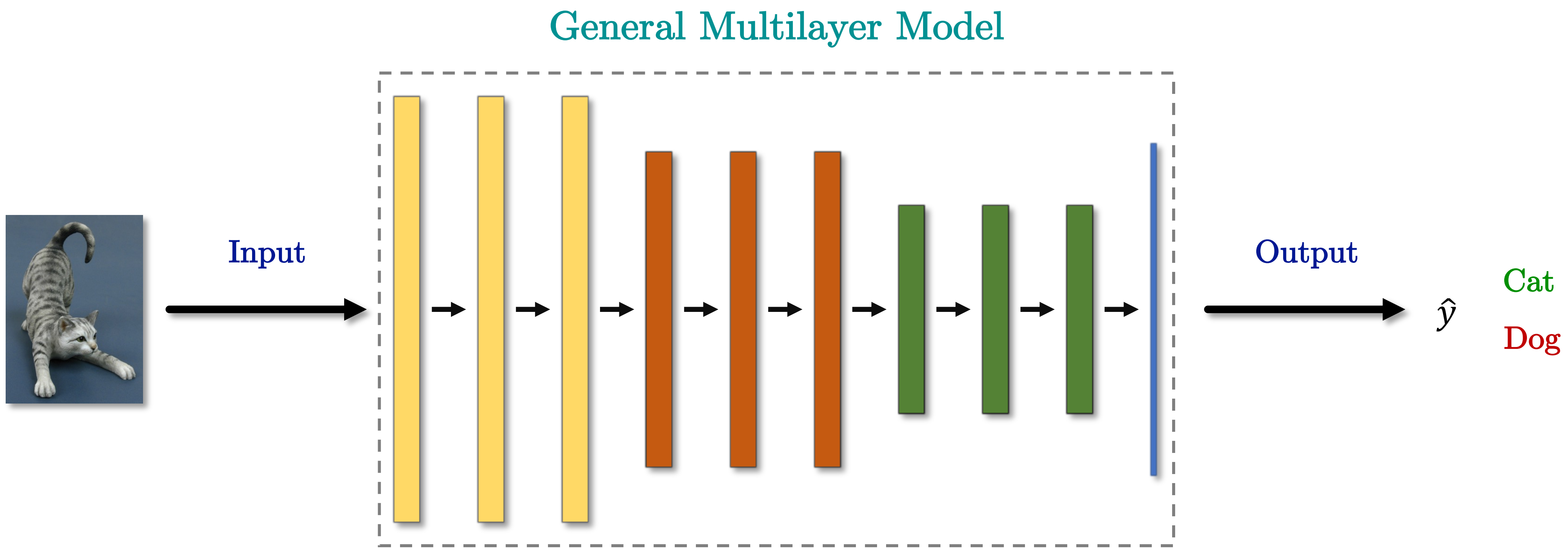
- Estimates influence using a **linearized representation** of the deep model.

Method #5: Representer Point

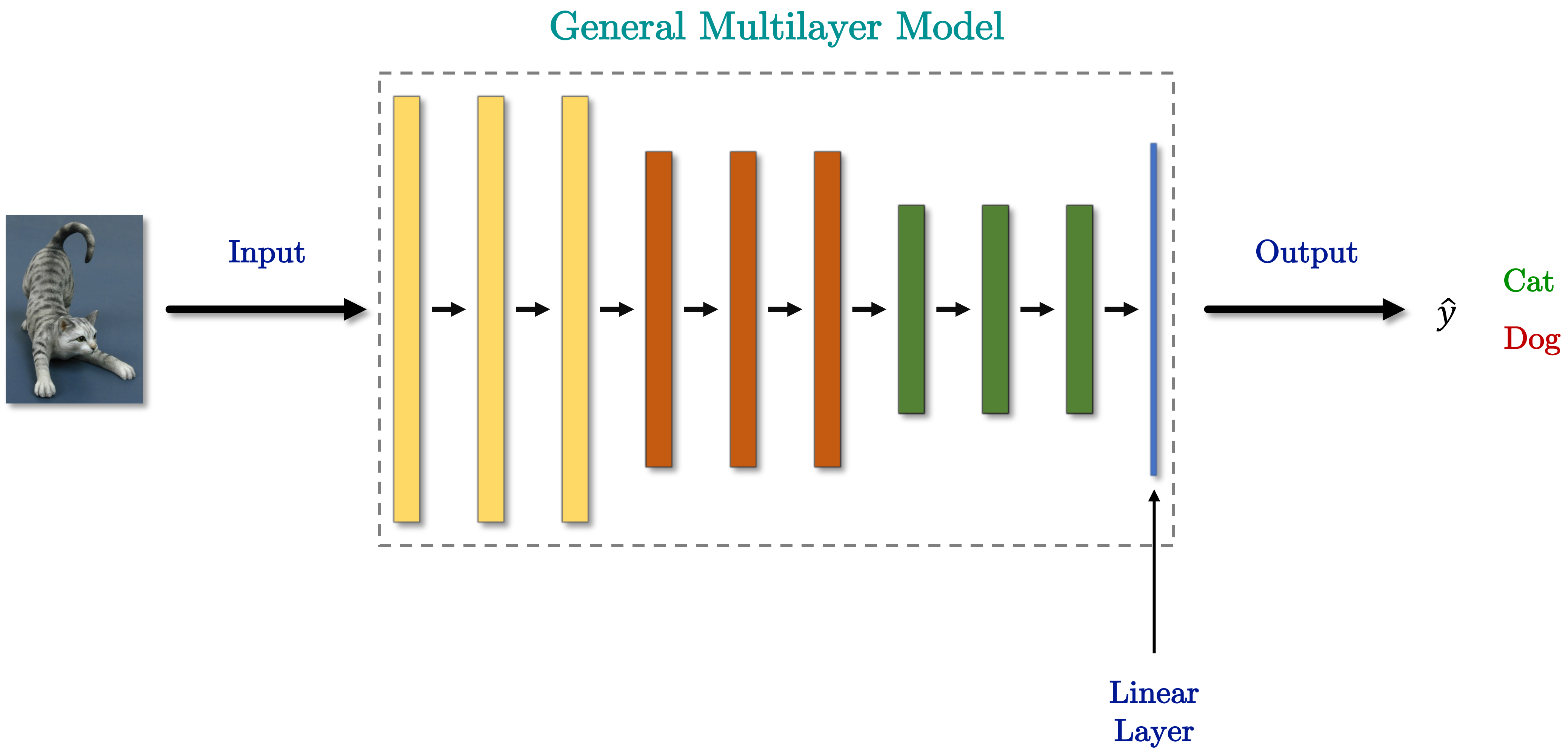
General Multilayer Model



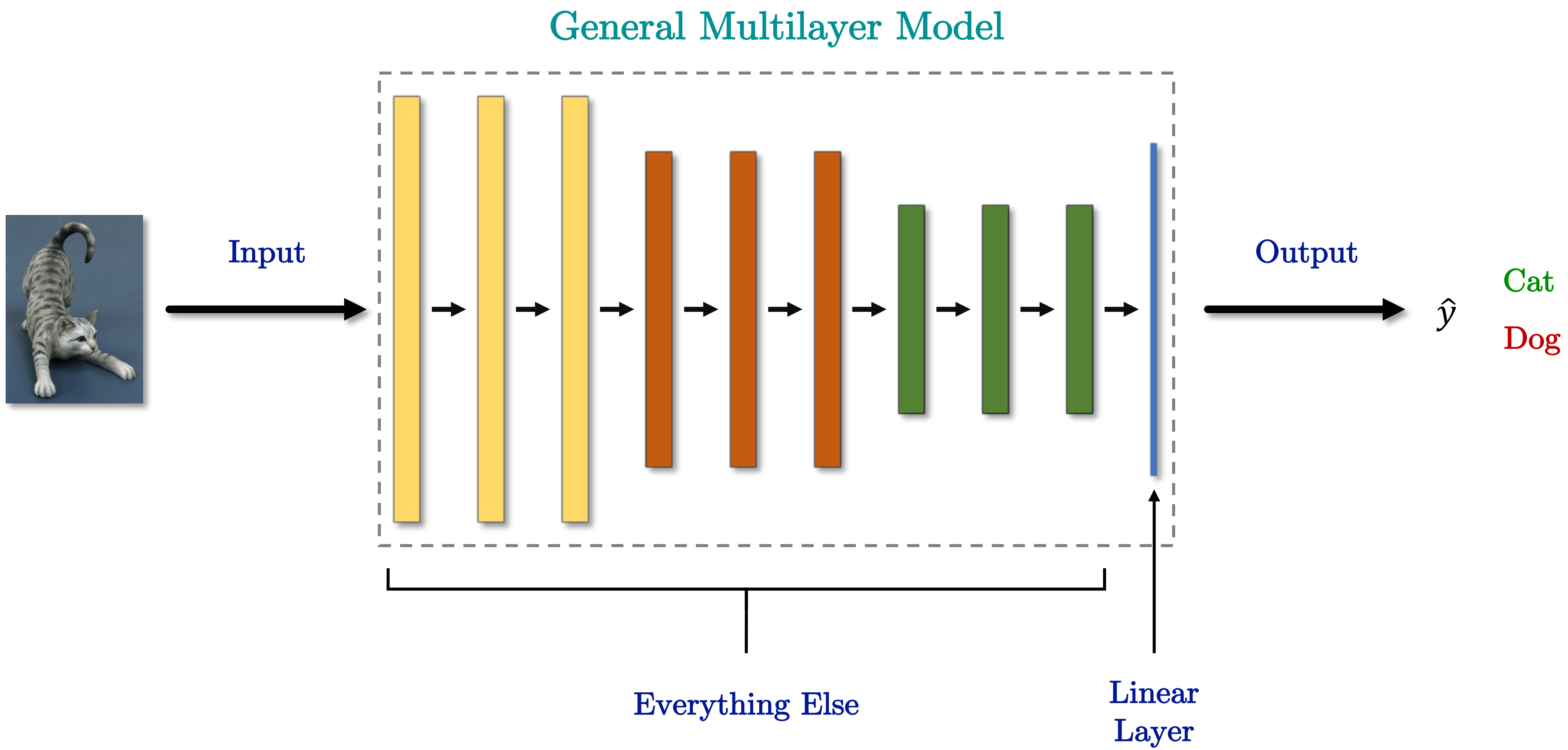
Method #5: Representer Point



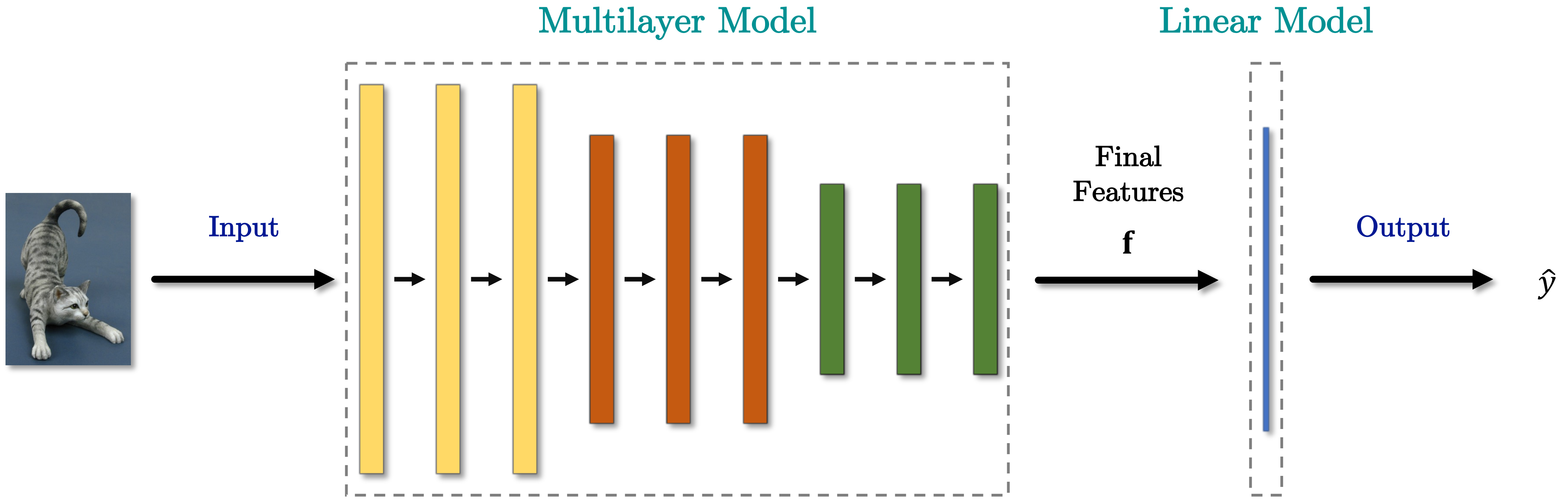
Method #5: Representer Point



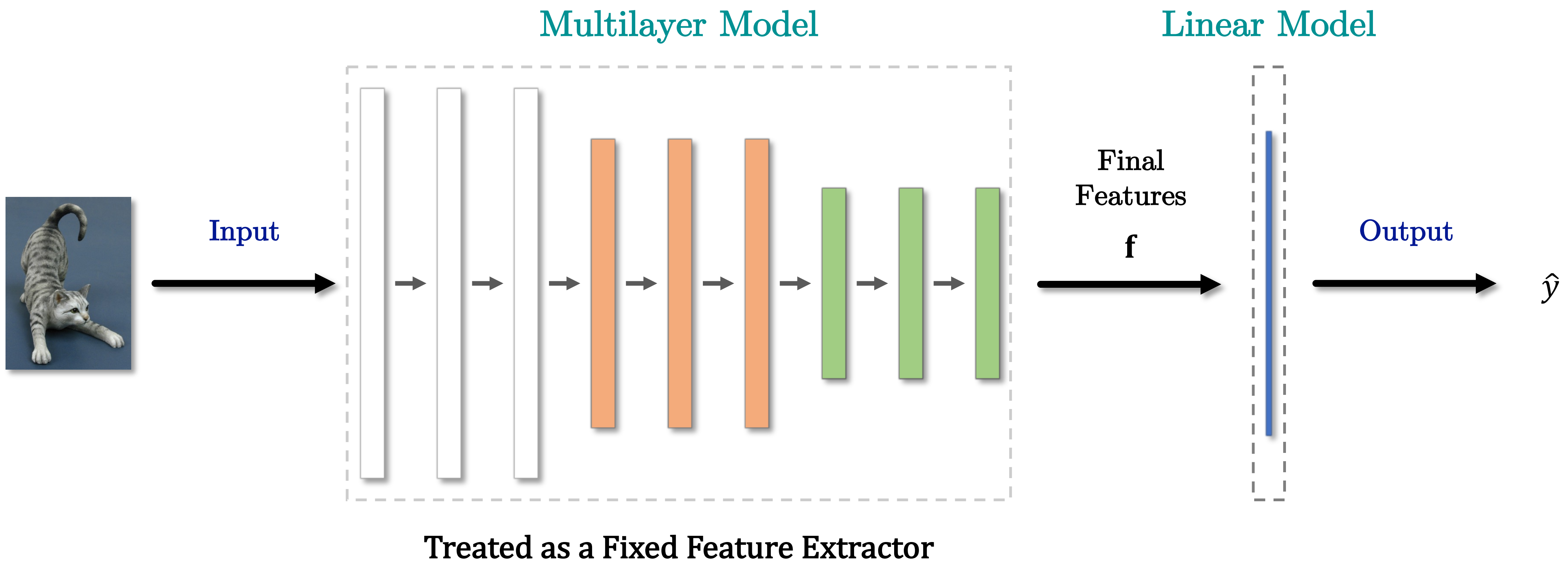
Method #5: Representer Point



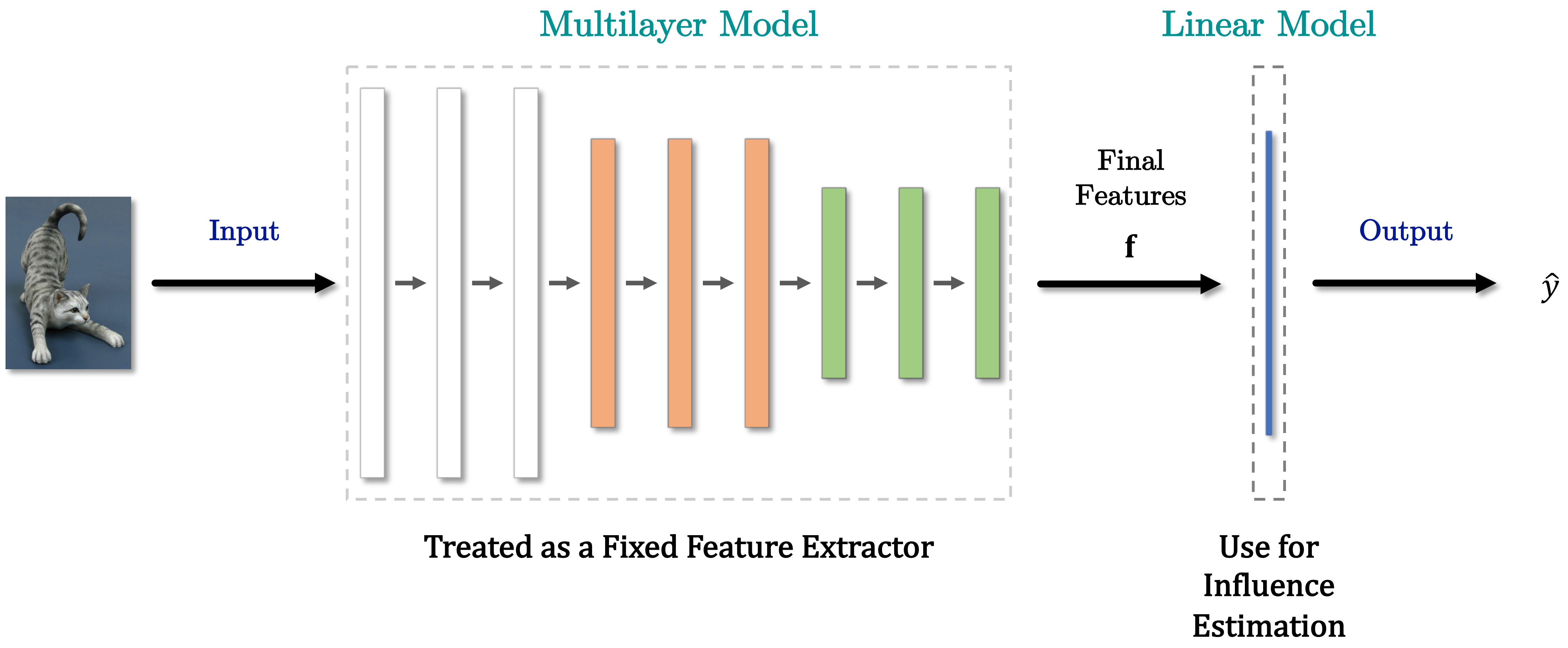
Method #5: Representer Point



Method #5: Representer Point

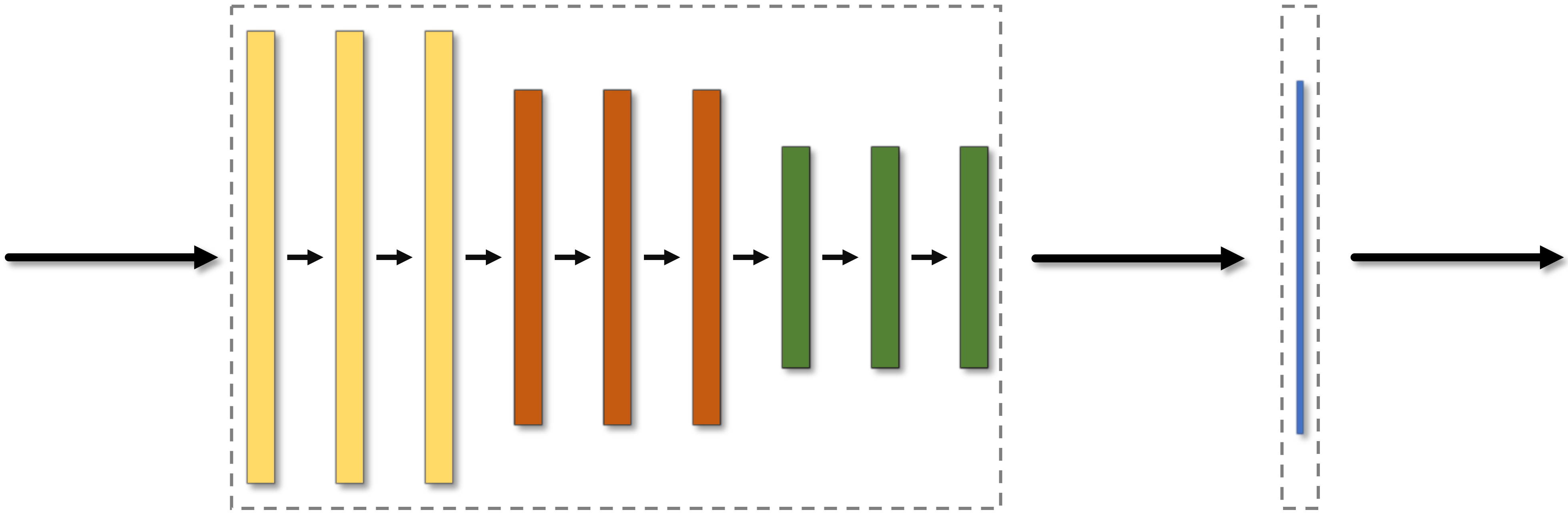


Method #5: Representer Point



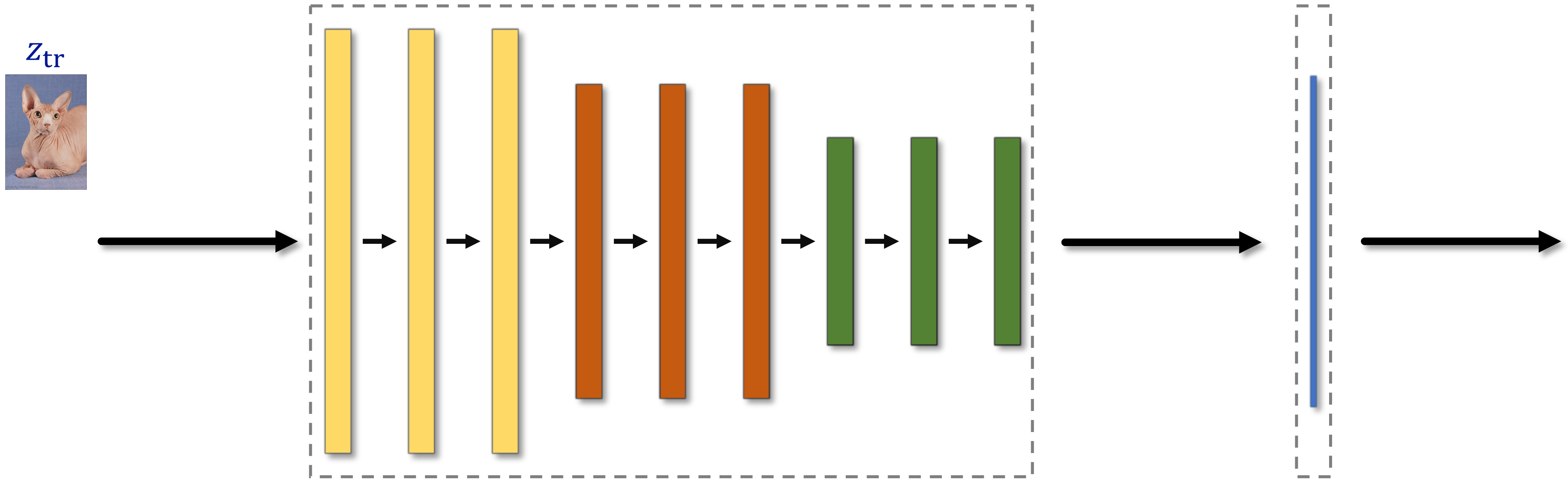
Representer Point Theorem & Influence

Scholkopf et al.'s [SHS01] **representer theorem** specifies how to calculate influence over any L_2 regularized linear model ($\lambda > 0$)



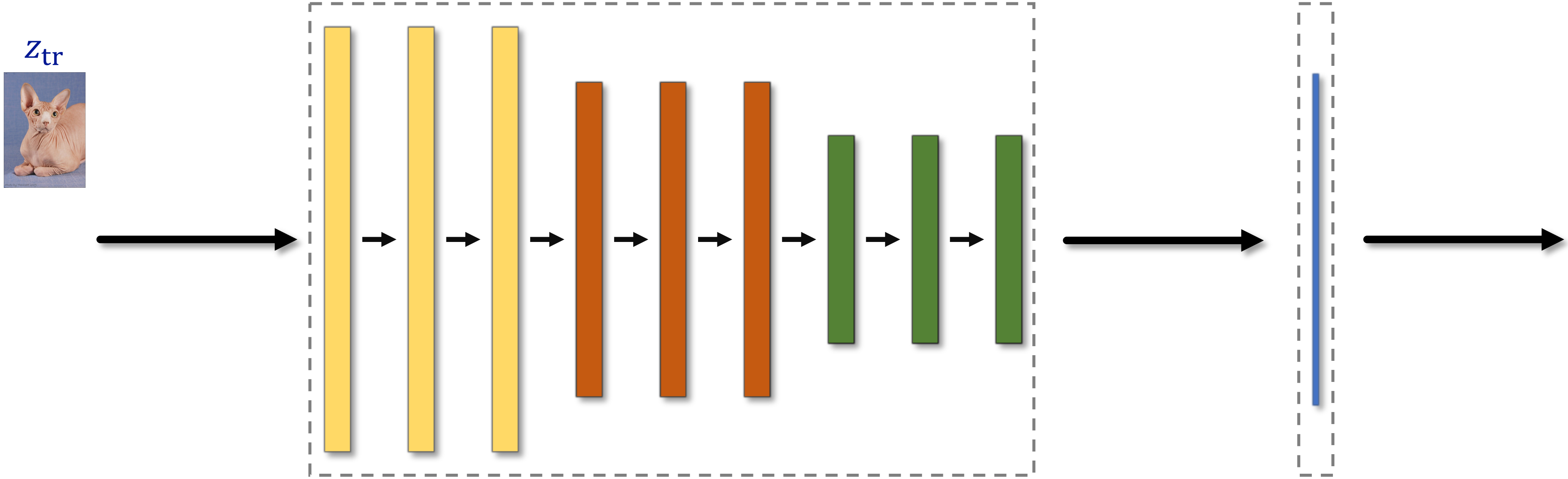
Representer Point Theorem & Influence

Scholkopf et al.'s [SHS01] **representer theorem** specifies how to calculate influence over any L_2 regularized linear model ($\lambda > 0$)



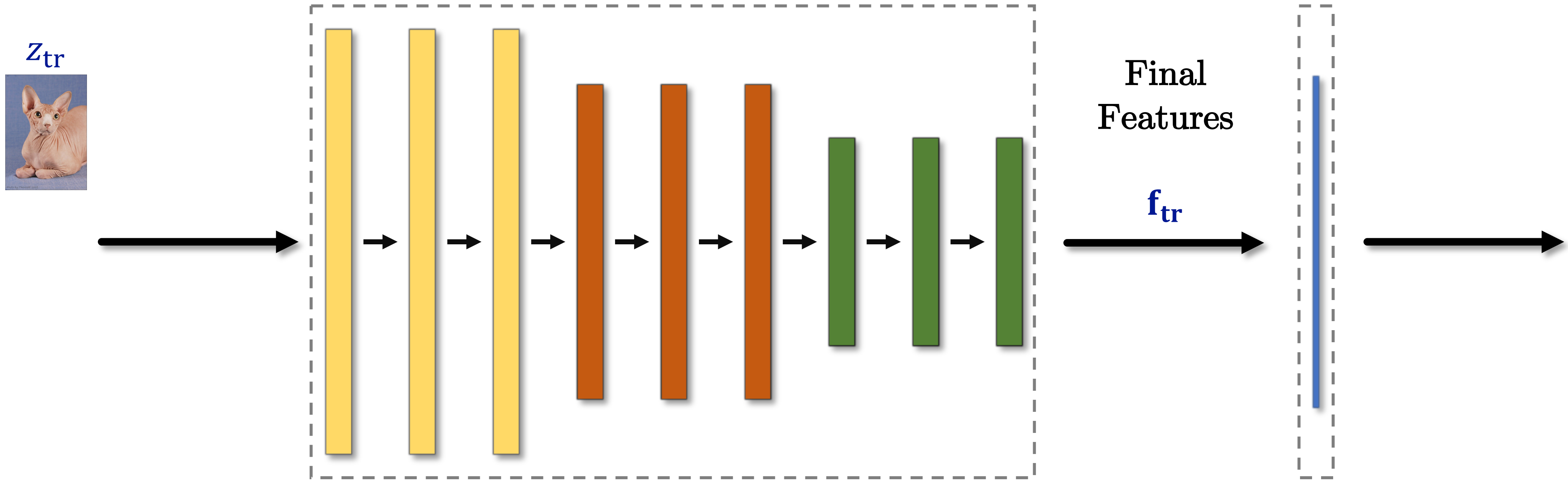
Representer Point Theorem & Influence

Scholkopf et al.'s [SHS01] **representer theorem** specifies how to calculate influence over any L_2 regularized linear model ($\lambda > 0$)



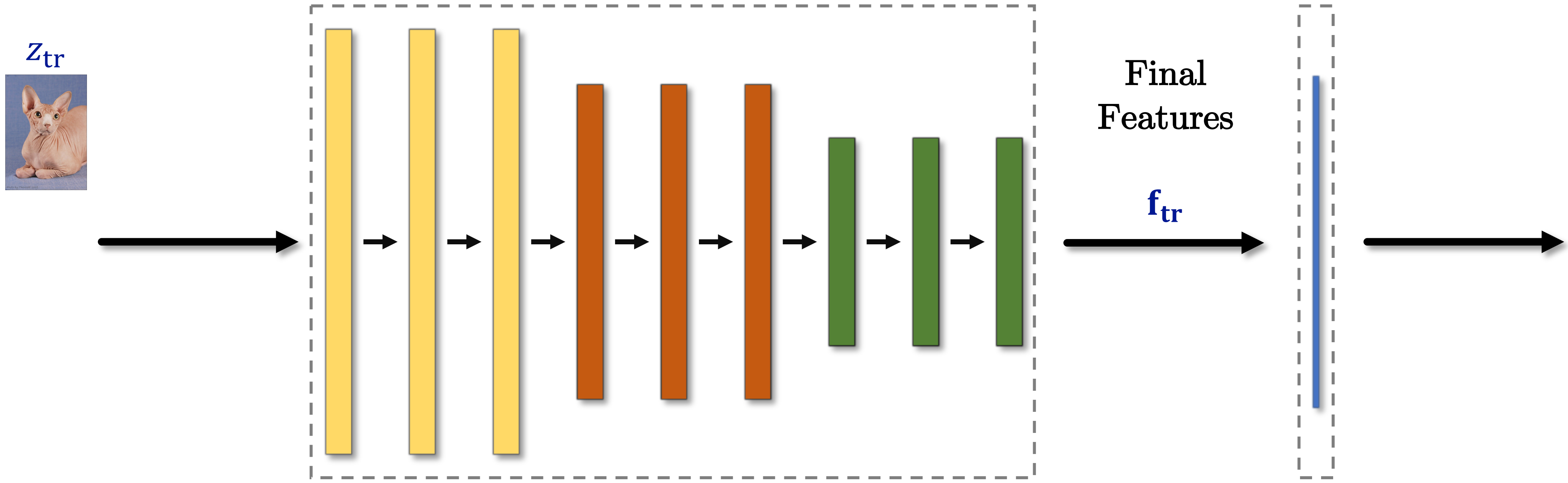
Representer Point Theorem & Influence

Scholkopf et al.'s [SHS01] **representer theorem** specifies how to calculate influence over any L_2 regularized linear model ($\lambda > 0$)



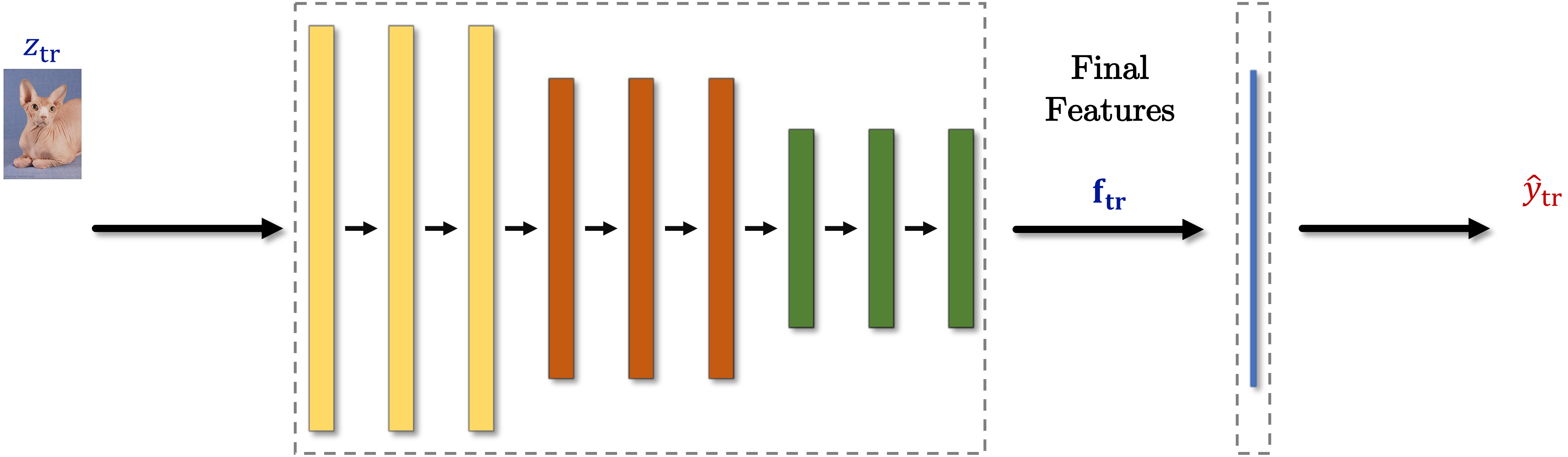
Representer Point Theorem & Influence

Scholkopf et al.'s [SHS01] **representer theorem** specifies how to calculate influence over any L_2 regularized linear model ($\lambda > 0$)



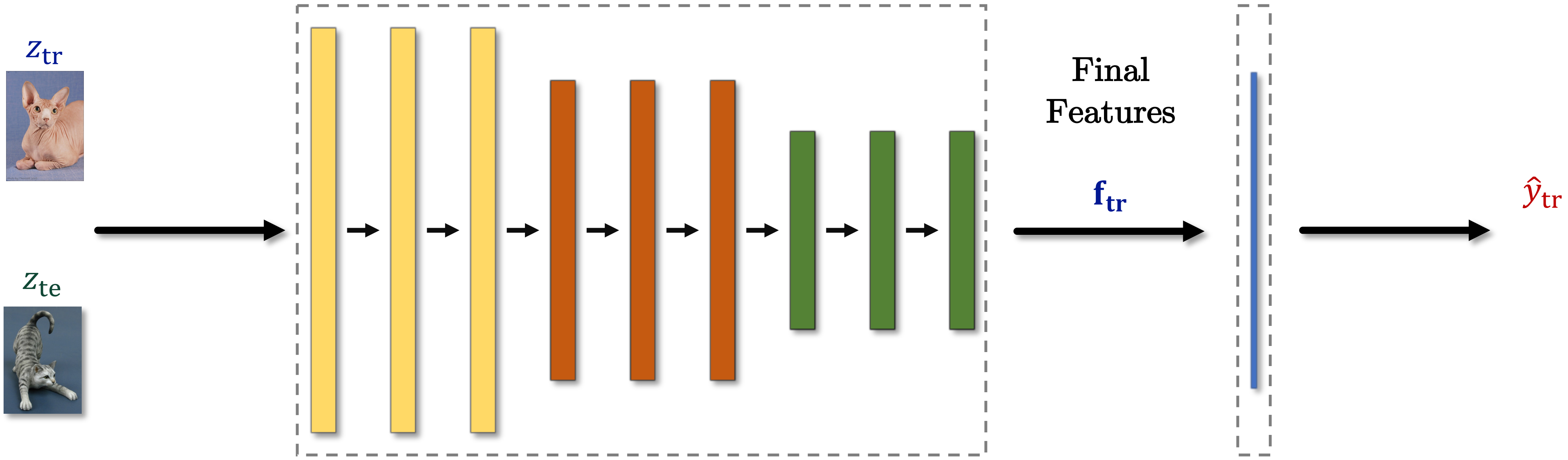
Representer Point Theorem & Influence

Scholkopf et al.'s [SHS01] **representer theorem** specifies how to calculate influence over any L_2 regularized linear model ($\lambda > 0$)



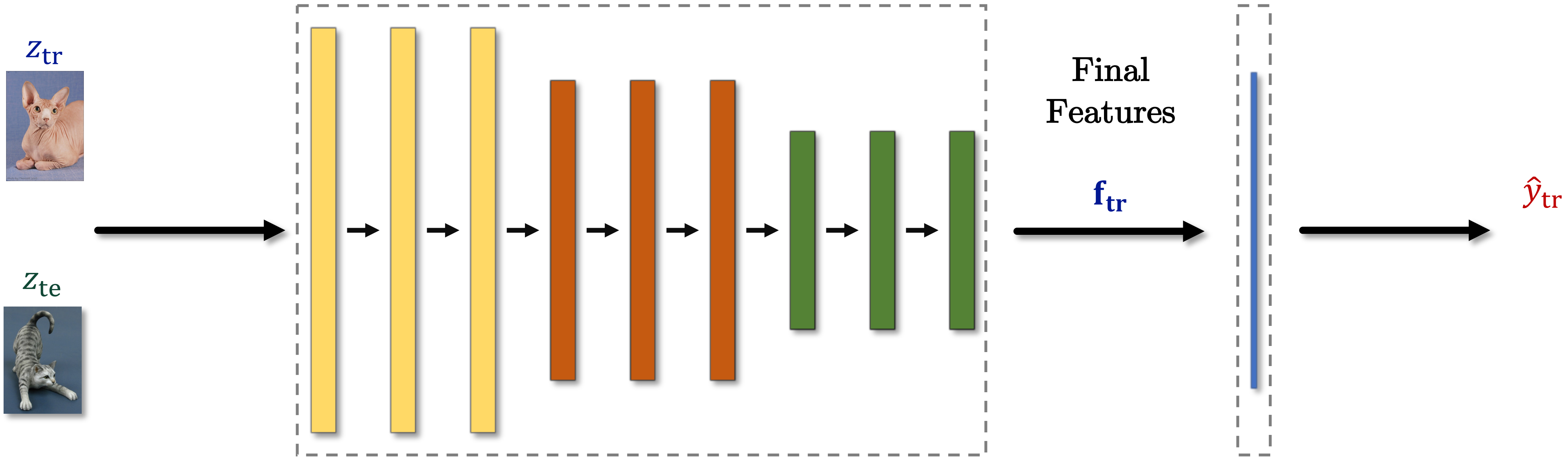
Representer Point Theorem & Influence

Scholkopf et al.'s [SHS01] **representer theorem** specifies how to calculate influence over any L_2 regularized linear model ($\lambda > 0$)



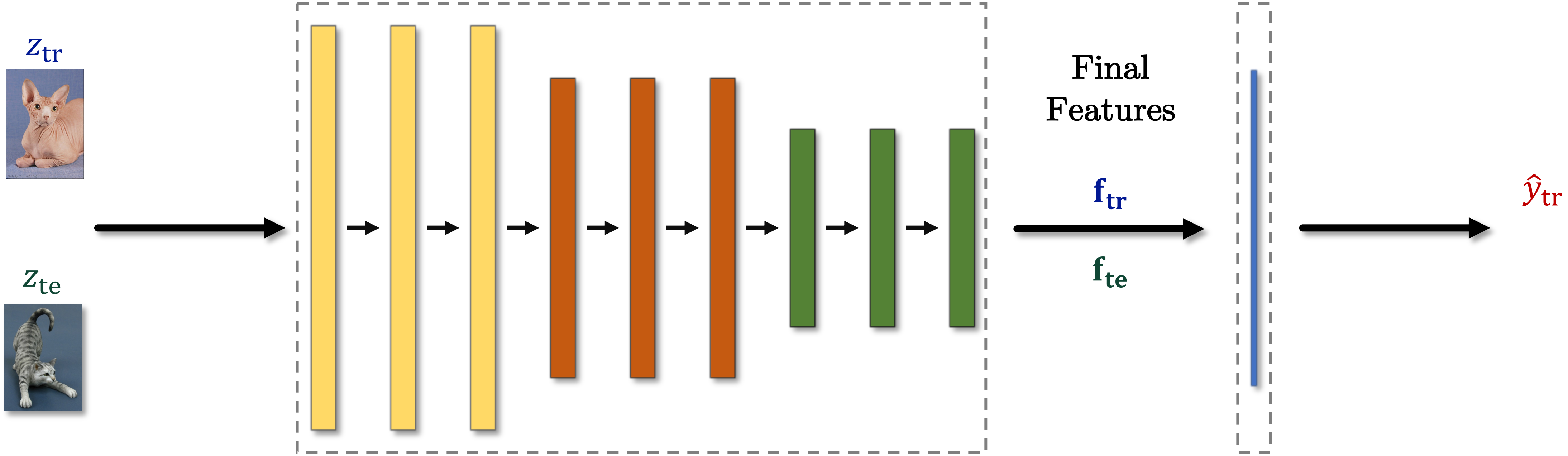
Representer Point Theorem & Influence

Scholkopf et al.'s [SHS01] **representer theorem** specifies how to calculate influence over any L_2 regularized linear model ($\lambda > 0$)



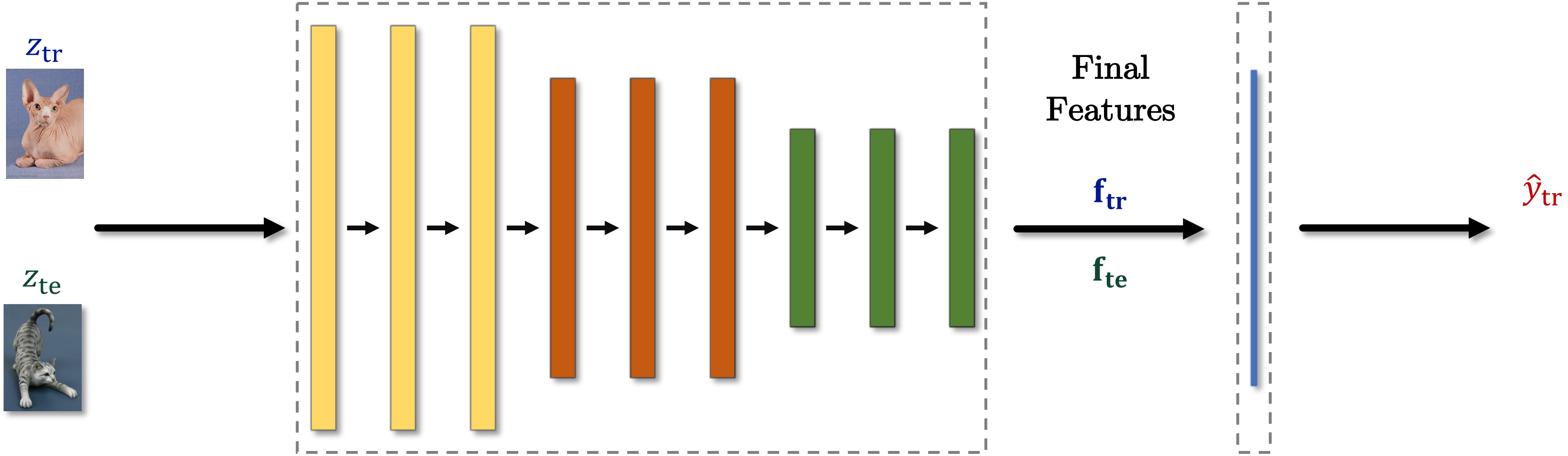
Representer Point Theorem & Influence

Scholkopf et al.'s [SHS01] **representer theorem** specifies how to calculate influence over any L_2 regularized linear model ($\lambda > 0$)



Representer Point Theorem & Influence

Scholkopf et al.'s [SHS01] **representer theorem** specifies how to calculate influence over any L_2 regularized linear model ($\lambda > 0$)



Representer Point Influence Estimate

$$\hat{I}_{RP} = -\frac{1}{2\lambda n}\frac{\partial L(\hat{y}_{tr}, y_{tr})}{\partial \hat{y}_{tr}}\mathbf{f}_{tr}\cdot\mathbf{f}_{te}$$

Representer Point's Complexity

Time Complexity

- Full & Incremental: $\mathcal{O}(n)$ (**Very Fast**)

Storage Complexity: $\mathcal{O}(1)$

Space Complexity: $\mathcal{O}(n + p)$

Representer Point [Yeh+18]: Strengths & Weaknesses

Strengths:

+ Very fast (by an order of magnitude or more)

Weaknesses:

- "Too reductive" [Yeh+22]

Final Thoughts:
Is Estimating Influence Statically a Good Idea?

Final Thoughts:
Is Estimating Influence Statically a Good Idea?

Answer: It depends.

- The more complex the model, the less static influence makes sense.

Final Thoughts: Is Estimating Influence Statically a Good Idea?

Answer: It depends.

- The more complex the model, the less static influence makes sense.

Intuition: Static influence is like reading the ending of a novel and trying to understand the whole story.

- It *may* be possible to get a broad idea of what happened.
- Most fine details are probably lost.

Final Thoughts:
Is Estimating Influence Statically a Good Idea?

Answer: It depends.

- The more complex the model, the less static influence makes sense.

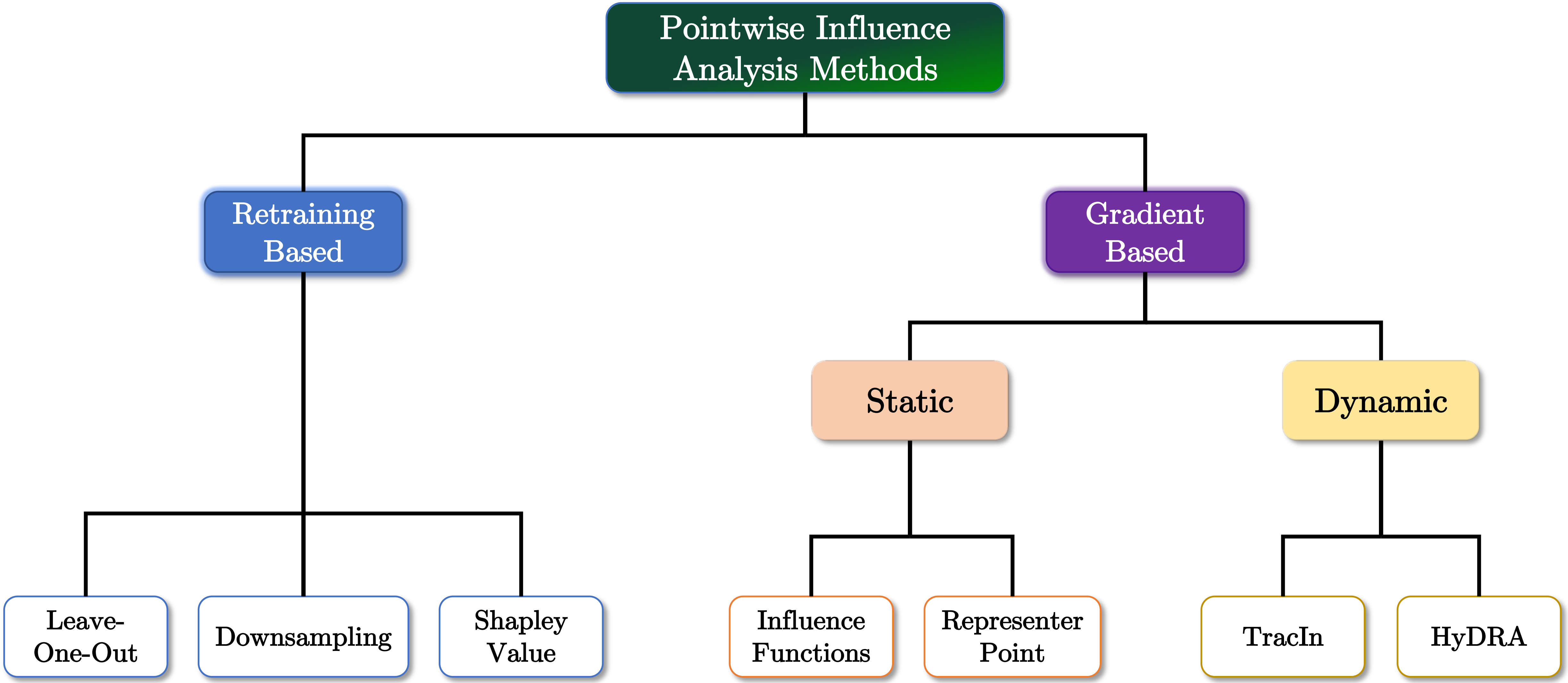
Intuition: Static influence is like <u>reading the ending of a novel</u> and trying to understand the whole story.

- It *may* be possible to get a broad idea of what happened.
- Most fine details are probably lost.

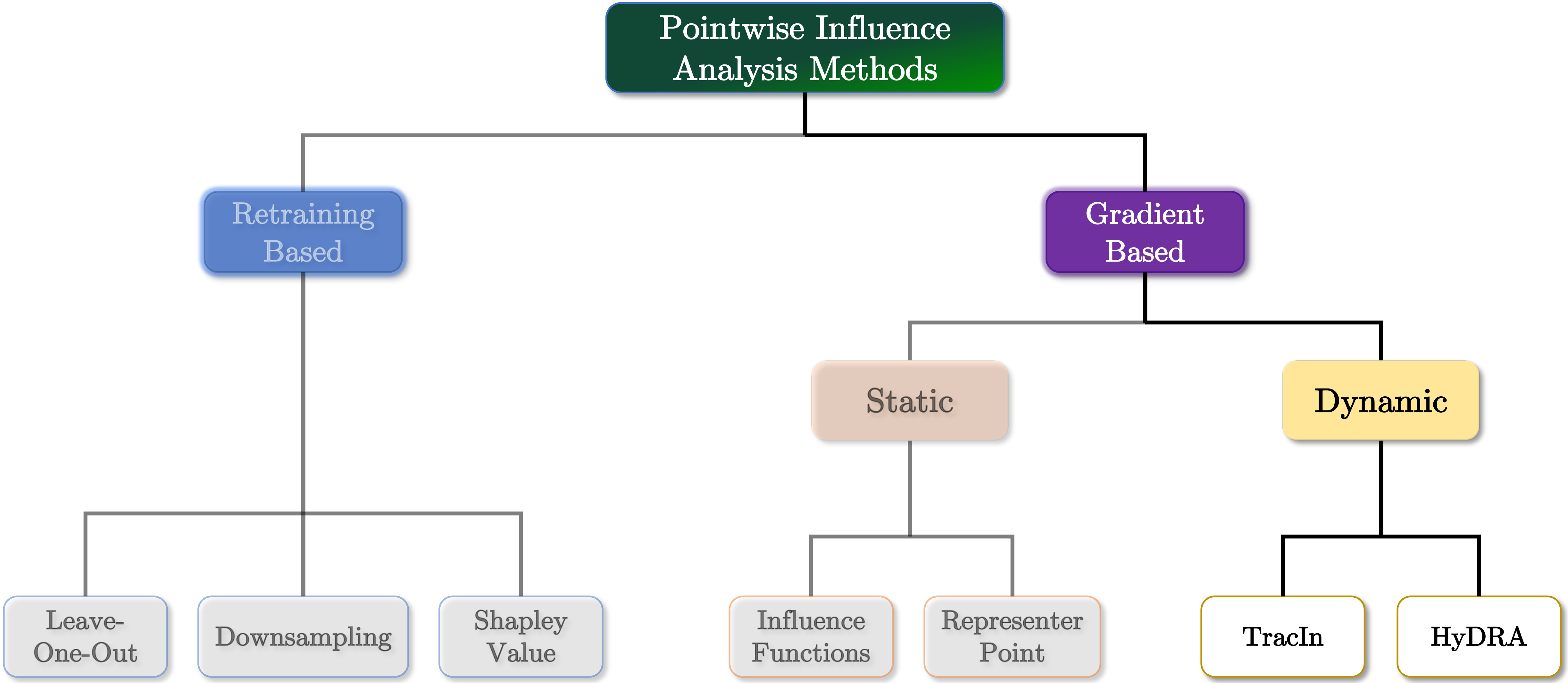
Better Way to Understand a Novel: Read from beginning to end.

- Same applies to influence analysis

Pointwise Influence Analysis Taxonomy



Pointwise Influence Analysis Taxonomy



Dynamic Gradient-Based Influence Analysis

A Common Thread and an Alternative

All preceding methods took the same basic approach to influence:

Perturb the training set and observe the change in the model

An Orthogonal Approach

- Influence occurs <u>during</u> the training process
- Estimate influence by observing how training instances affect the test loss <u>during training</u>

Visualize Measuring Influence During Training

Visualize Measuring Influence During Training

Ref. Test Instance

Visualize Measuring Influence During Training

Ref. Test Instance

For simplicity, assume gradient descent with a batch size of 1 and no momentum

Visualize Measuring Influence During Training

Ref. Test Instance

For simplicity, assume gradient descent with a batch size of 1 and no momentum

Training Iteration (t)

Visualize Measuring Influence During Training

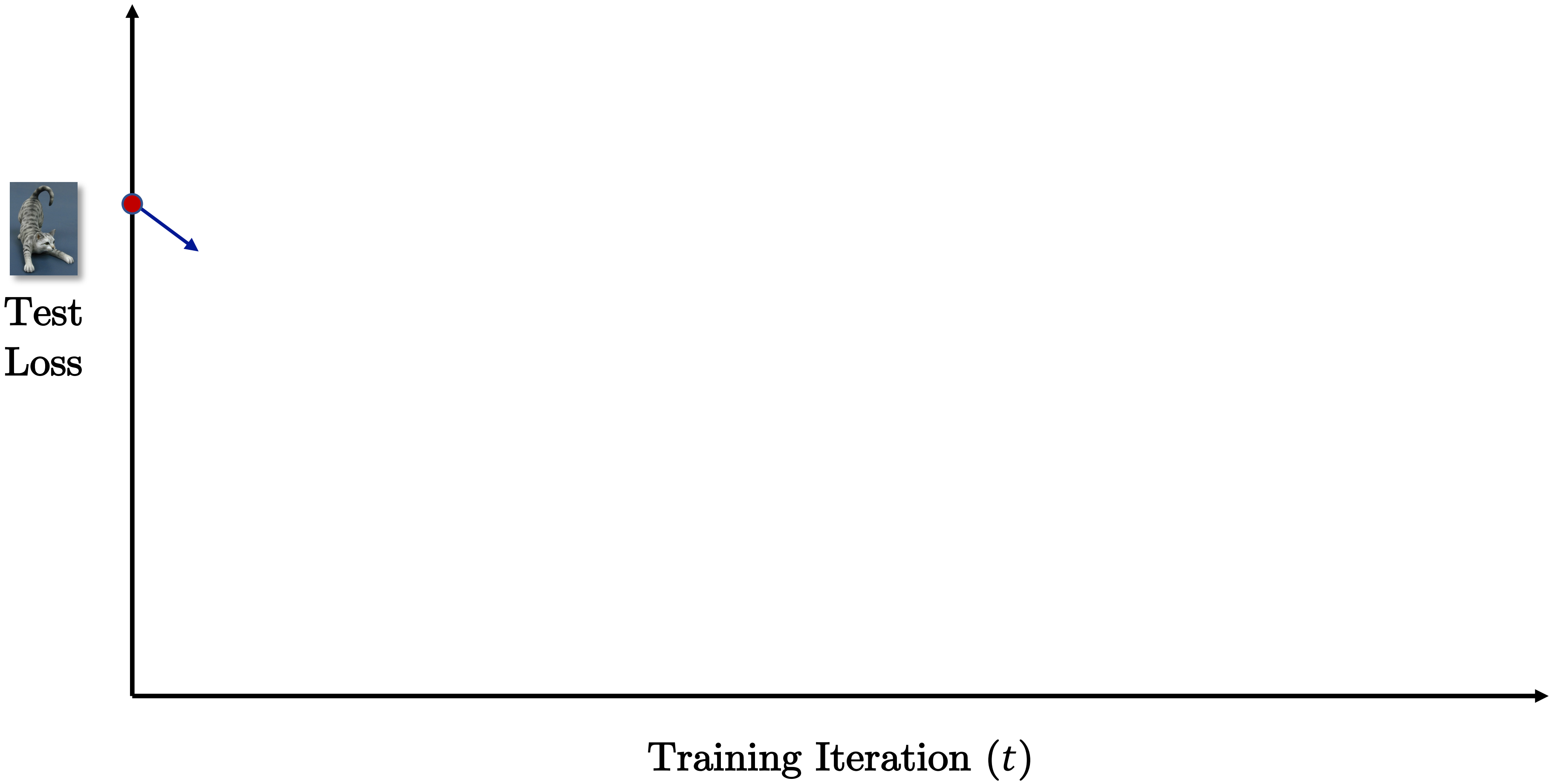
Test
Loss

Training Iteration (t)

Ref. Test
Instance

For simplicity, assume gradient descent with a batch size of 1 and no momentum

Visualize Measuring Influence During Training



Ref. Test Instance

For simplicity, assume gradient descent with a batch size of 1 and no momentum

Visualize Measuring Influence During Training

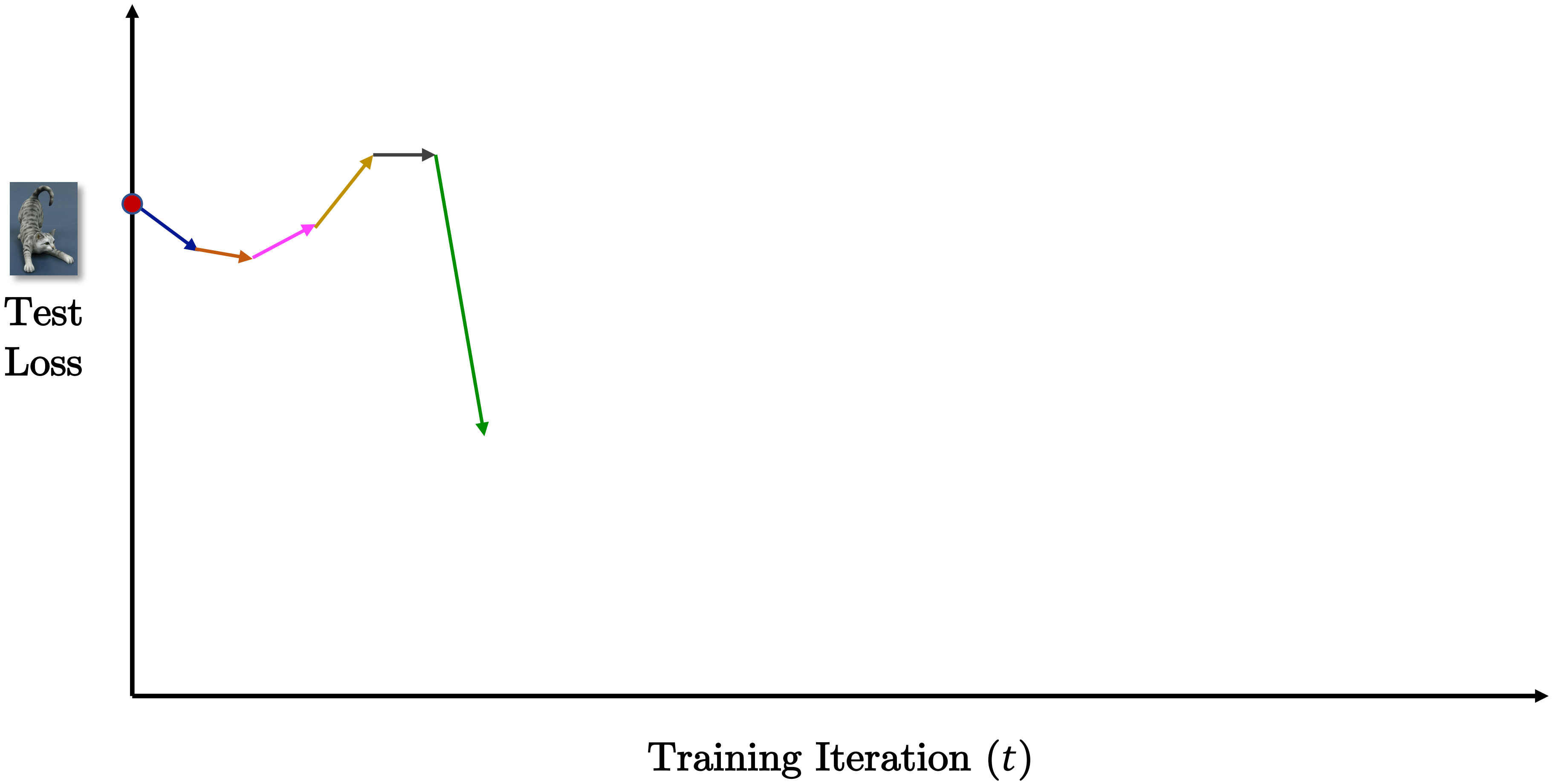
Test
Loss

Training Iteration (t)

Ref. Test
Instance

For simplicity, assume gradient descent with a batch size of 1 and no momentum

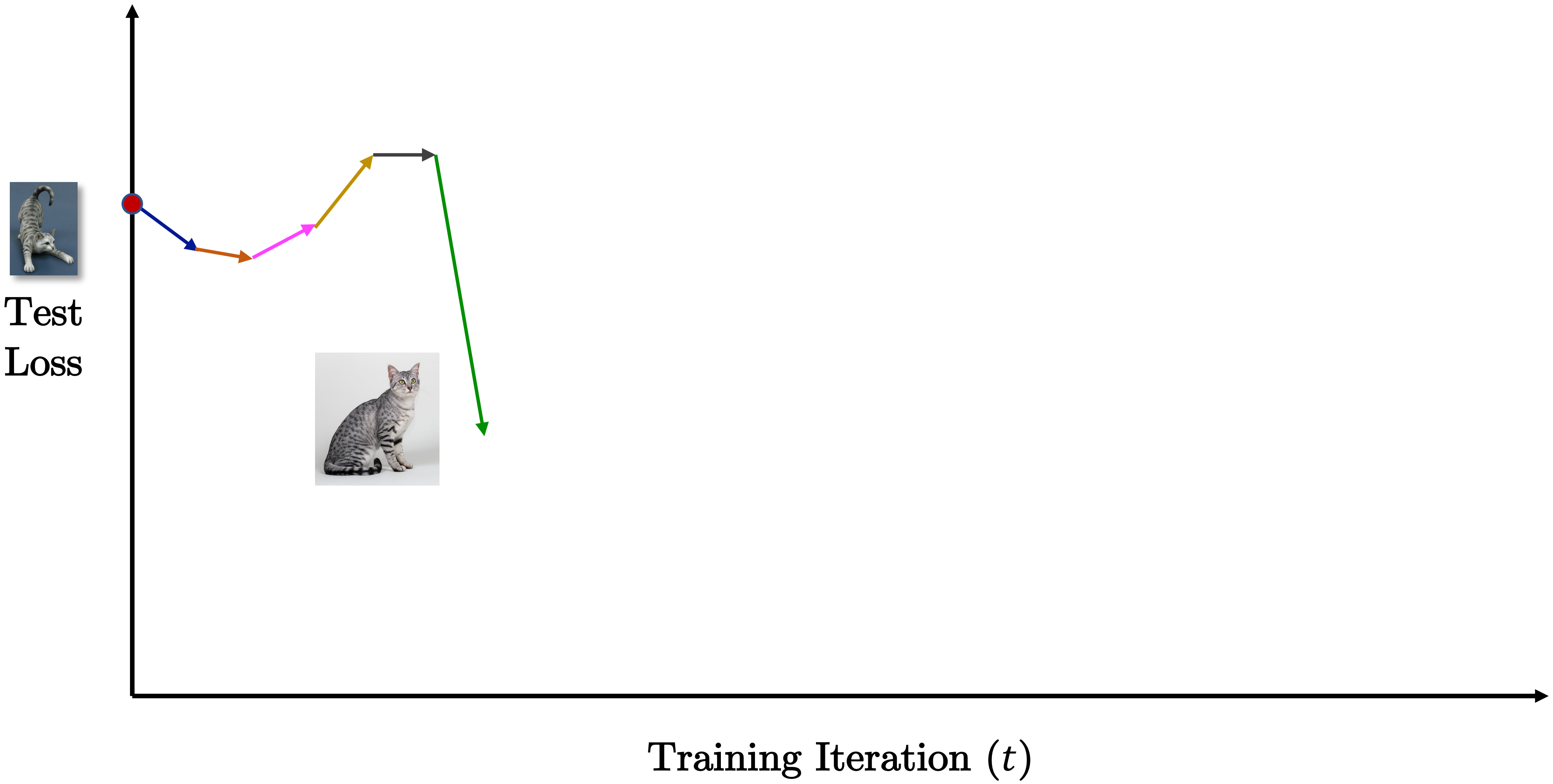
Visualize Measuring Influence During Training



Ref. Test Instance

For simplicity, assume gradient descent with a batch size of 1 and no momentum

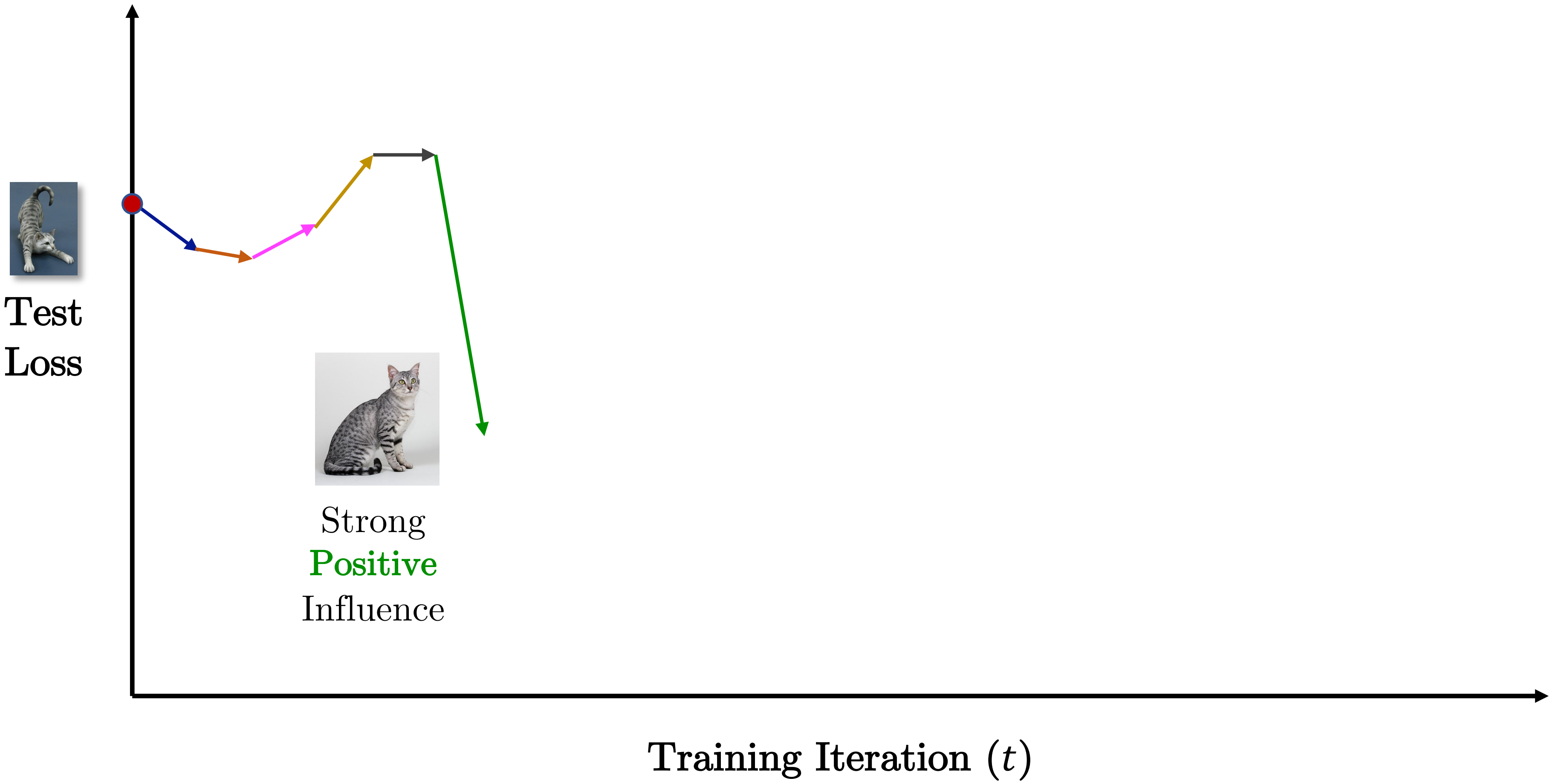
Visualize Measuring Influence During Training



Ref. Test Instance

For simplicity, assume gradient descent with a batch size of 1 and no momentum

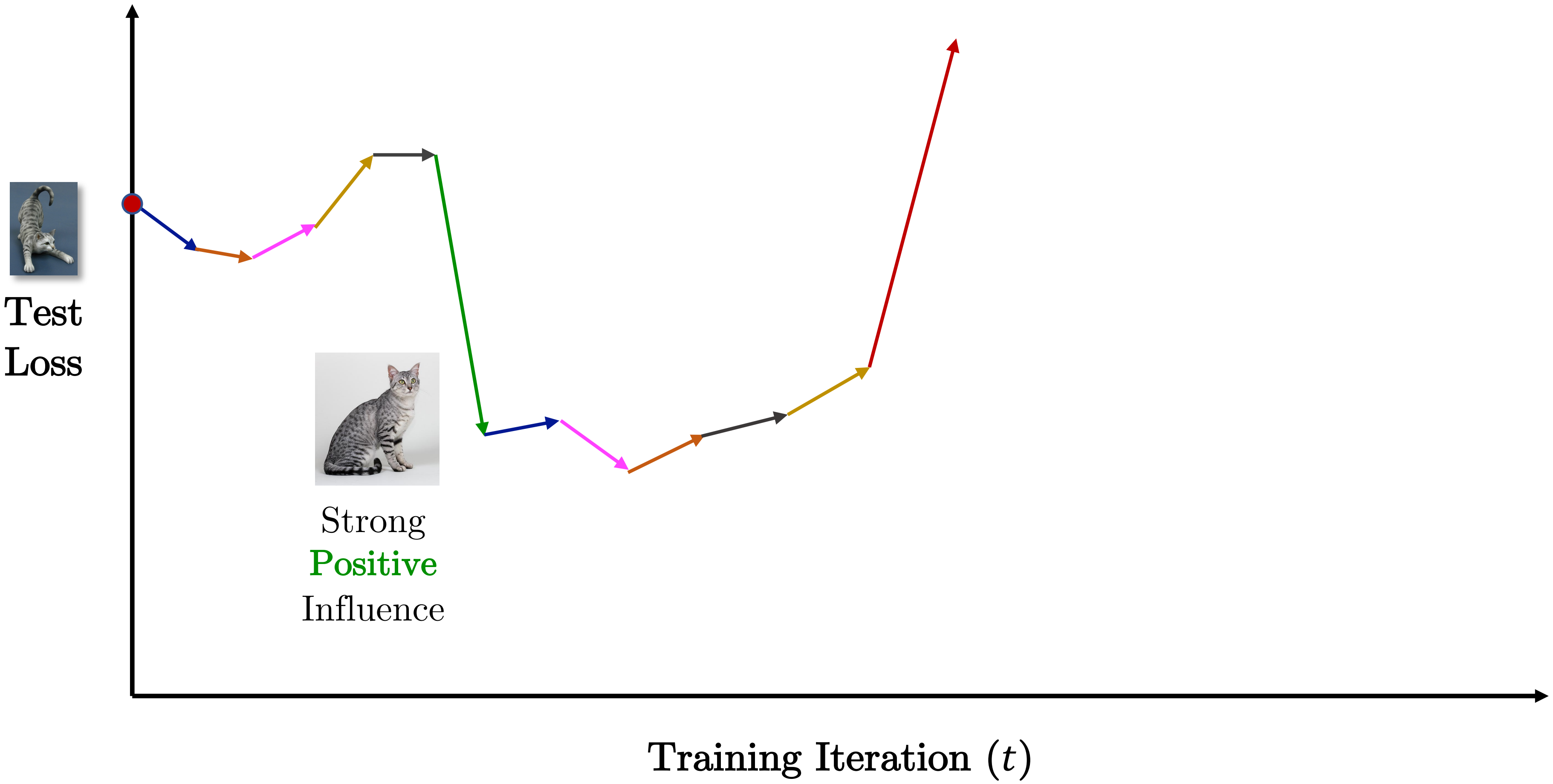
Visualize Measuring Influence During Training



Ref. Test Instance

For simplicity, assume gradient descent with a batch size of 1 and no momentum

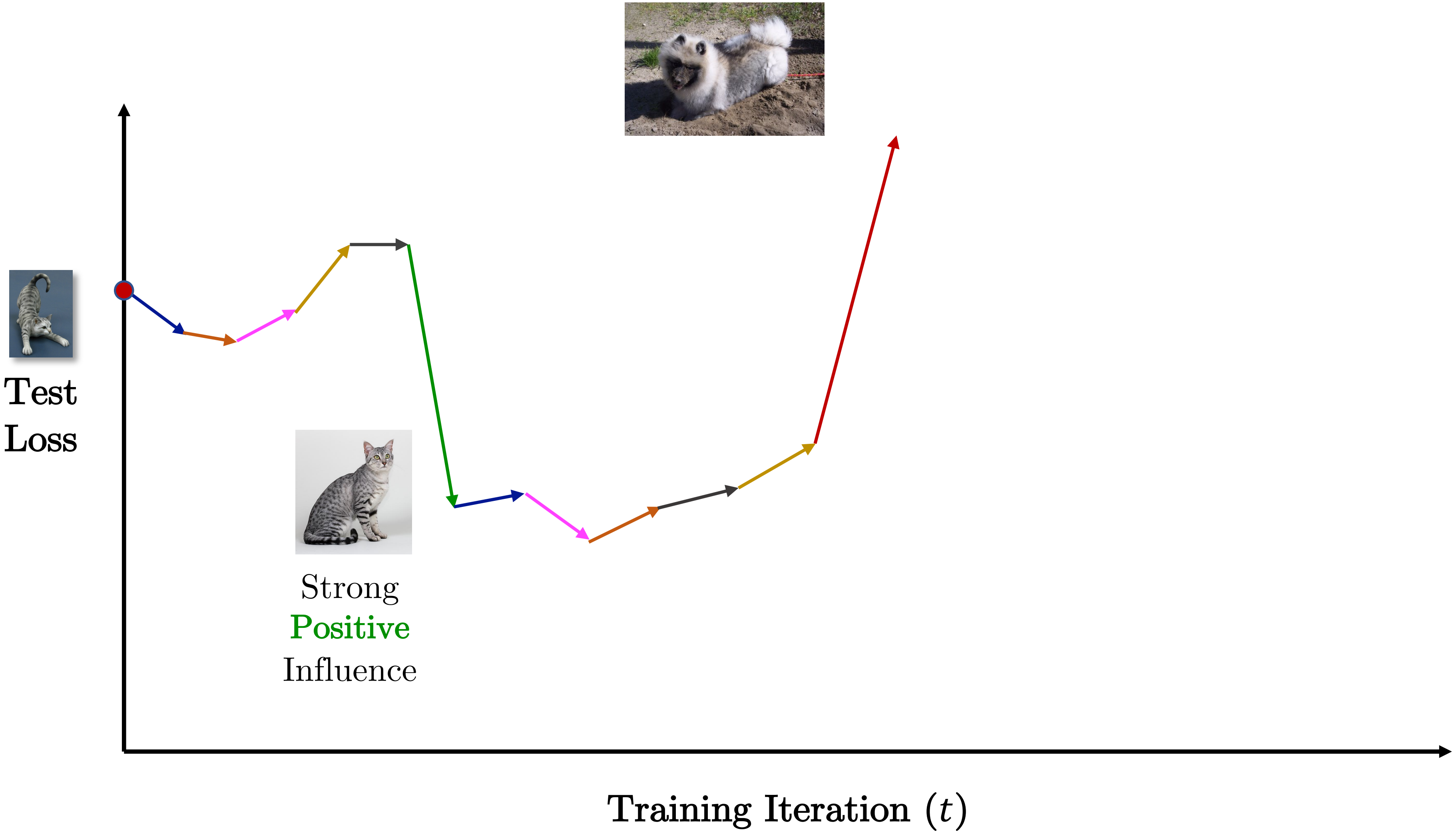
Visualize Measuring Influence During Training



Ref. Test Instance

For simplicity, assume gradient descent with a batch size of 1 and no momentum

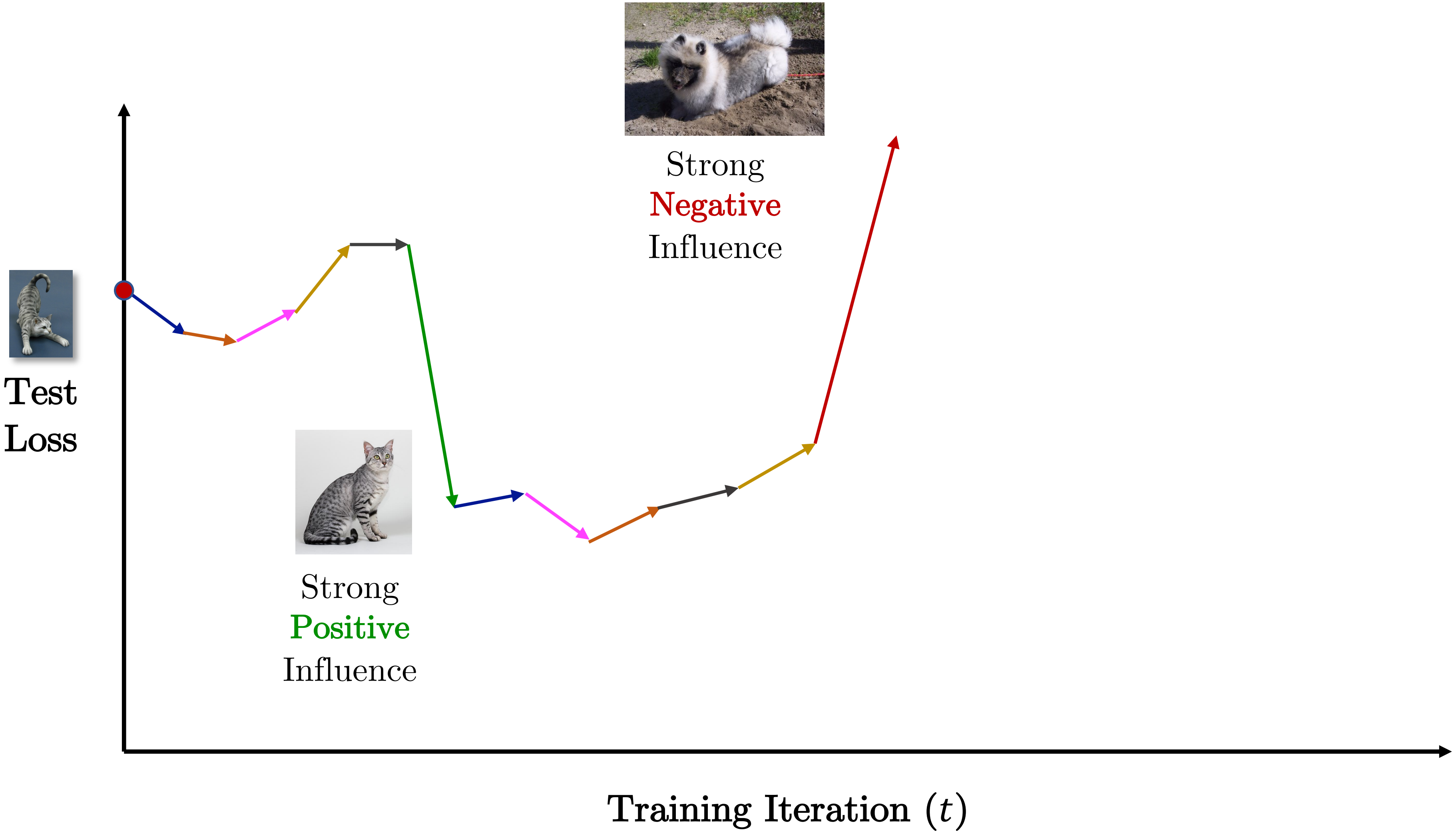
Visualize Measuring Influence During Training



Ref. Test
Instance

For simplicity, assume gradient descent with a batch size of 1 and no momentum

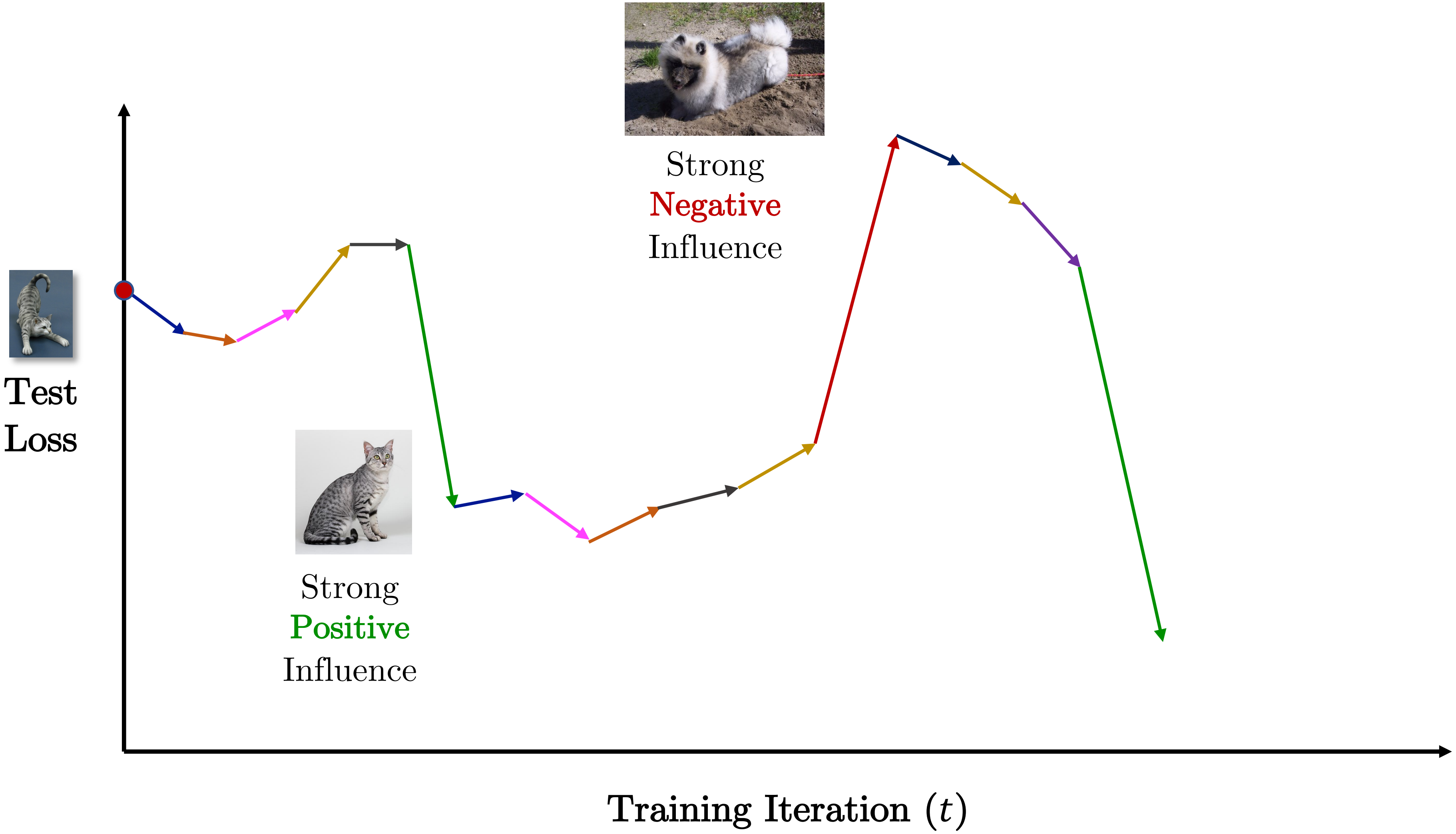
Visualize Measuring Influence During Training



Ref. Test
Instance

For simplicity, assume gradient descent with a batch size of 1 and no momentum

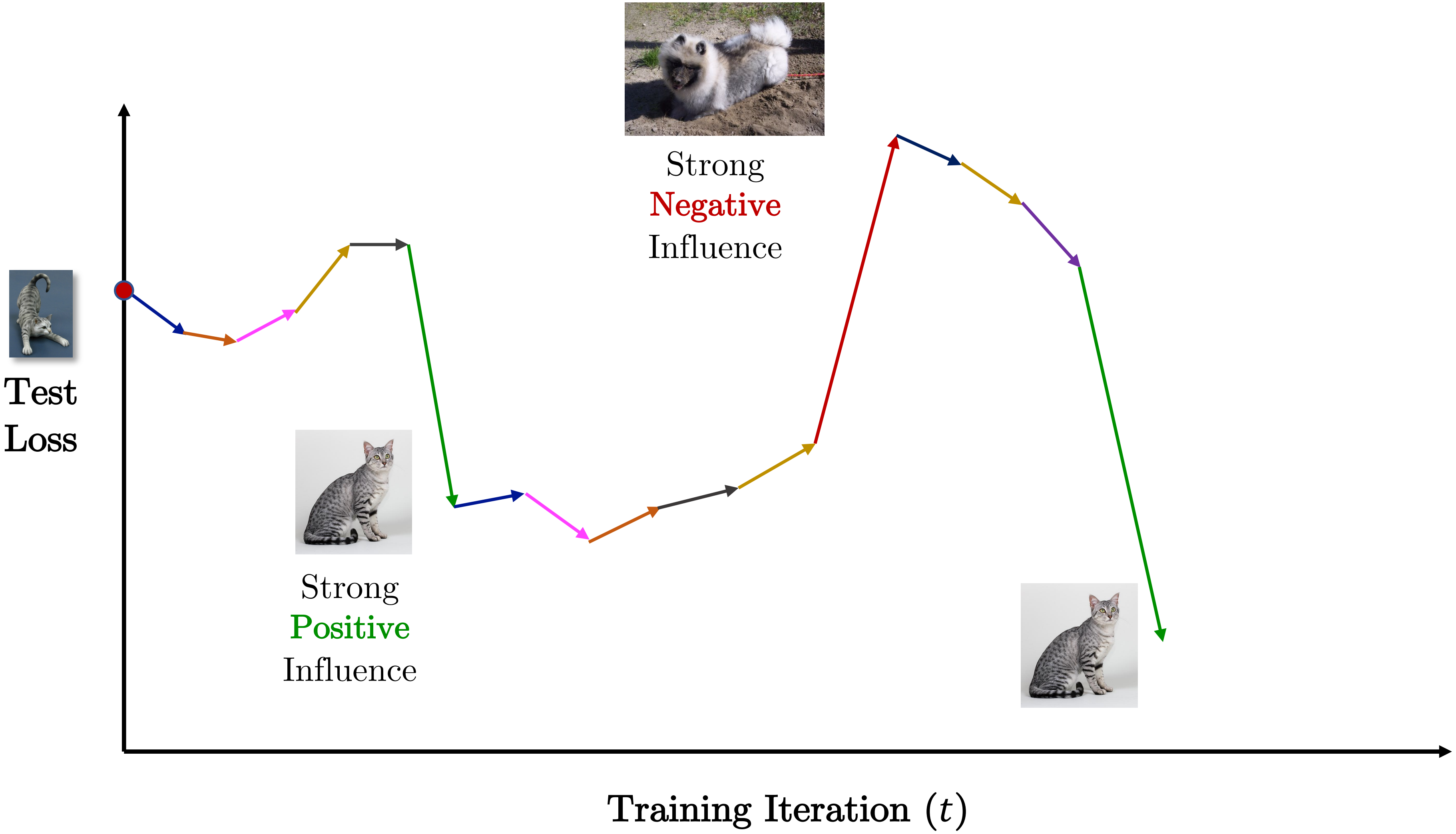
Visualize Measuring Influence During Training



Ref. Test
Instance

For simplicity, assume gradient descent with a batch size of 1 and no momentum

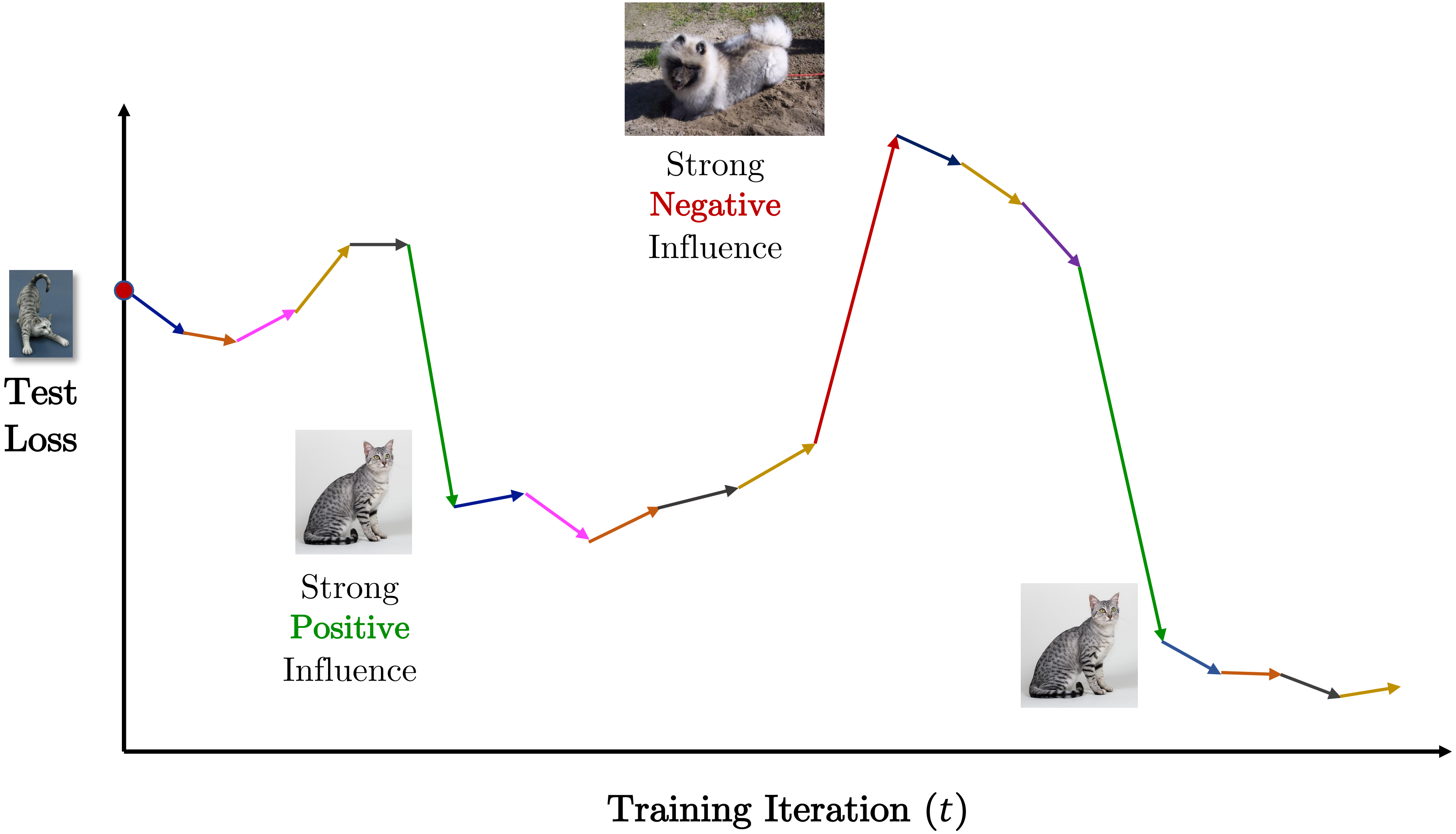
Visualize Measuring Influence During Training



Ref. Test Instance

For simplicity, assume gradient descent with a batch size of 1 and no momentum

Visualize Measuring Influence During Training



Ref. Test Instance

For simplicity, assume gradient descent with a batch size of 1 and no momentum

Method #6: TracIn [Pru+20]

Estimates influence **during training** via the preceding basic idea.

- Problem: Singleton batches and no momentum is too slow

TracIn Influence Estimator:

- <u>Non-Singleton</u>: Analyze batch gradients to determine which batch instances caused the test loss change
- <u>Implementation</u>: Simple. Just take a series of gradient dot products.

When Your Biggest Strength is Your Biggest Weakness

By retracing gradient descent, TracIn can detect influential training instances overlooked by other (static) methods.

- TracIn generally outperforms other influence estimators

The Big Weakness:

- TracIn retraces gradient descent for every test instance
- Very expensive computationally

TracIn's Complexity

Time Complexity

- Full & Incremental: $\mathcal{O}(npT)$
- Note the dependence on iteration count T for both the full and incremental complexity

Storage Complexity: $\mathcal{O}(pT)$ (Huge)

Space Complexity: $\mathcal{O}(n+p)$

TracIn: Strengths & Weaknesses

Strengths:

+ Better identifies influential instances
+ Simple theory and implementation (e.g., no HVP)

Weaknesses:

- Computationally expensive
- High storage cost

Method #7: HyDRA [Che+21]

- ,

Method #7: HyDRA [Che+21]

- ,

Method #7: HyDRA [Che+21]

- **Recall**: Influence functions efficiently estimates the LOO influence by assuming model convexity

    - ,

Method #7: HyDRA [Che+21]

- **Recall**: Influence functions efficiently estimates the LOO influence by assuming model convexity

- HyDRA estimates the LOO influence but does **not** assume convexity

Method #7: HyDRA [Che+21]

- Recall: Influence functions efficiently estimates the LOO influence by assuming model convexity

- HyDRA estimates the LOO influence but does <u>not</u> assume convexity
  - Impact: Each training set change causes the model to converge to a very different risk minimizer θ'

Method #7: HyDRA [Che+21]

- Recall: Influence functions efficiently estimates the LOO influence by assuming model convexity

- HyDRA estimates the LOO influence but does **not** assume convexity
  - Impact: Each training set change causes the model to converge to a very different risk minimizer θ'
- Estimating this alternate minimizer requires retracing all of training

Method #7: HyDRA [Che+21]

- Recall: Influence functions efficiently estimates the LOO influence by assuming model convexity

- HyDRA estimates the LOO influence but does **not** assume convexity
  - Impact: Each training set change causes the model to converge to a very different risk minimizer θ'
- Estimating this alternate minimizer requires retracing all of training
  - Explaining the mechanics of HyDRA's hypergradient tracing is beyond the scope of this talk. See our full paper for details.

Big Picture Summary:
Dynamic Gradient-Based Influence Estimators

Big Picture Summary:
Dynamic Gradient-Based Influence Estimators

Two Takeaways:

- Best performing
- Slowest

Applications of Influence Analysis

Data Cleaning

"Unhelpful" Training Instance: Any instance that causes the model's overall performance to (significantly) decline

- Potential Causes: Mislabeling, noisy features, etc.

Data Cleaning

"Unhelpful" Training Instance: Any instance that causes the model's overall performance to (significantly) decline

- **Potential Causes**: Mislabeling, noisy features, etc.

Data Cleaning's Basic Procedure:

1. Train a model
2. Use influence analysis to identify unhelpful training instances
3. Remove unhelpful instances from the training set and retrain the model

Data Cleaning

"Unhelpful" Training Instance: Any instance that causes the model's overall performance to (significantly) decline

- **Potential Causes**: Mislabeling, noisy features, etc.

Data Cleaning's Basic Procedure:

1. Train a model
2. Use influence analysis to identify unhelpful training instances
3. Remove unhelpful instances from the training set and retrain the model

Methods to identify unhelpful instances:

- Identify instances that are consistently negatively influential on a held-out set
- Identify memorized training instances

Memorization is a Bug...

Memorization is the influence of a training instance *on itself*

- ***Intuition***: An instance is memorized if it must be in the training set to be correctly predicted

Question: Is memorization always a bad thing?

- **Conventional Wisdom**: Yes

Memorization is a Bug...

Memorization is the influence of a training instance *on itself*

- ***Intuition***: An instance is memorized if it must be in the training set to be correctly predicted

Question: Is memorization always a bad thing?

- **Conventional Wisdom**: Yes

It has recently been shown that the right answer is "It's complicated"

- Downsampling was used to show that eliminating training set memorization **doubles** the top-1 ImageNet error. [FZ20]

Memorization is a Bug... but also a Feature

Memorization is the influence of a training instance *on itself*

- ***Intuition***: An instance is memorized if it must be in the training set to be correctly predicted

Question: Is memorization always a bad thing?

- **Conventional Wisdom**: Yes

It has recently been shown that the right answer is "It's complicated"

- Downsampling was used to show that eliminating training set memorization **doubles** the top-1 ImageNet error. [FZ20]

Adversarial Attacks and Defenses

Training-Set Attack: Adversary inserts malicious instances into the *training set* to manipulate model behavior.

To change a prediction, the adversarial instances must *influence* the model.

- **Takeaway #1**: Crafting a training-set attack reduces to creating influential training instances.

Adversarial Attacks and Defenses

Training-Set Attack: Adversary inserts malicious instances into the *training set* to manipulate model behavior.

To change a prediction, the adversarial instances must *influence* the model.

- **Takeaway #1**: Crafting a training-set attack reduces to creating influential training instances.

Attackers are often limited in the number of adversarial instances they can insert into the training set.

- **Takeaway #2**: Detecting adversarial attacks reduces to identifying test examples with a few exceptionally influential training instances.

Future Research Directions

Group Influence over Pointwise Influence

Most influence analysis research focuses on pointwise effects

Most predictions are moderately affected by **multiple** training instances

- Group influence effects are often **supermodular**
- **Intuition**: A training instance deletion has a larger effect if the instance is one-of-a-kind versus if it has 1,000 copies

Group Influence over Pointwise Influence

Most influence analysis research focuses on pointwise effects

Most predictions are moderately affected by **multiple** training instances

- Group influence effects are often supermodular
- **Intuition**: A training instance deletion has a larger effect if the instance is one-of-a-kind versus if it has 1,000 copies

Takeaway #1: Pointwise influence is often too reductive

Group Influence over Pointwise Influence

Most influence analysis research focuses on pointwise effects

Most predictions are moderately affected by **multiple** training instances

- Group influence effects are often supermodular
- **Intuition**: A training instance deletion has a larger effect if the instance is one-of-a-kind versus if it has 1,000 copies

Takeaway #1: Pointwise influence is often too reductive

Takeaway #2: Future influence analysis research should focus on group effects

Improve Influence Estimation's Scalability

Influence analysis is slow

- Analyzing the pointwise influence w.r.t. a single test instance can take **hours**
- Influence analysis' computational overhead *restricts its usage*

To be practical, influence analysis must be **faster by at least an order of magnitude**

Improve Influence Estimation's Scalability

Influence analysis is slow

- Analyzing the pointwise influence w.r.t. a single test instance can take **hours**
- Influence analysis' computational overhead *restricts its usage*

To be practical, influence analysis must be **faster by at least an order of magnitude**

How can we get there?

- Better heuristic methods
- Surrogate model analysis (e.g., pruned/efficient models)
- Specialization of influence analysis by domain/data modality (e.g., text)

Improve Influence Estimation's Scalability

Influence analysis is slow

- Analyzing the pointwise influence w.r.t. a single test instance can take **hours**
- Influence analysis' computational overhead *restricts its usage*

To be practical, influence analysis must be **faster by at least an order of magnitude**

How can we get there?

- Better heuristic methods
- Surrogate model analysis (e.g., pruned/efficient models)
- Specialization of influence analysis by domain/data modality (e.g., text)

Understand the Risk: Increased inaccuracy + introduction of new "blind spots"

Certified Influence Estimation

Existing influence estimators provide **no accuracy guarantees**.

Many domains require that limited training-set changes do not affect a model's decision.

- **Example:** Certified defenses against training-set attacks.

- Influence estimation cannot currently be applied in these settings.

Certified Influence Estimation

Existing influence estimators provide **no accuracy guarantees**.

Many domains require that limited training-set changes do not affect a model's decision.

- **Example:** Certified defenses against training-set attacks.
- Influence estimation cannot currently be applied in these settings.

Takeaway: Certified influence estimation is sorely needed

Final Thoughts

Final Thoughts

Numerous approaches exist to define and measure influence

- This is a **feature** – not a bug

ML practitioners need to understand the trade-offs/limitations of the various methods to select the best one for their application

- Our full paper provides more details to inform this choice

Significant future work remains to make influence analysis **more practical** and **more useable**

- Existing applications demonstrate influence analysis' potential despite its limitations

For a curated list of resources related to training-set influence analysis, see our GitHub repo:

https://github.com/ZaydH/influence_analysis_papers

References

[Bae+22] Juhan Bae, Nathan Ng, Alston Lo, Marzyeh Ghassemi, and Roger Grosse. "*If Influence Functions are the Answer, Then What is the Question*?" 2022. arXiv: 2209.05364.

[BPF21] Samyadeep Basu, Phil Pope, and Soheil Feizi. "*Influence Functions in Deep Learning Are Fragile*". ICLR. 2021.

[BR92] Avrim L. Blum and Ronald L. Rivest. "Training a 3-node neural network is NP-complete". In: Neural Networks 5.1 (1992), pp. 117–127. ISSN: 0893-6080.

[Che+21] Yuanyuan Chen, Boyang Li, Han Yu, Pengcheng Wu, and Chunyan Miao. "*HyDRA: Hypergradient Data Relevance Analysis for Interpreting Deep Neural Networks*". AAAI. 2021.

[CW82] R. Dennis Cook and Sanford Weisberg. *Residuals and Influence in Regression*. New York: Chapman and Hall, 1982.

[FZ20] Vitaly Feldman and Chiyuan Zhang. "*What Neural Networks Memorize and Why: Discovering the Long Tail via Influence Estimation*". NeurIPS. 2020.

[GZ19] Amirata Ghorbani and James Zou. "*Data Shapley: Equitable Valuation of Data for Machine Learning*". ICML. 2019.

[Ham74] Frank R. Hampel. "*The Influence Curve and its Role in Robust Estimation*". Journal of the American Statistical Association 69.346 (1974), pp. 383–393.

[Jae72] Louis A. Jaeckel. The Infinitesimal Jackknife. Tech. rep. Bell Laboratories, 1972.

[KL17] Pang Wei Koh and Percy Liang. "*Understanding Black-box Predictions via Influence Functions*". ICML. 2017.

[Pru+20] Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. "*Estimating Training Data Influence by Tracing Gradient Descent*". NeurIPS. 2020.

[Sha53] Lloyd S Shapley. "*A Value for n-Person Games*". Contributions to the Theory of Games II. Princeton, NJ USA: Princeton University Press, 1953, pp. 307–317.

[SHS01] Bernhard Scholkopf, Ralf Herbrich, and Alex J. Smola. "*A Generalized Representer Theorem*". COLT/EuroCOLT. 2001.

[Yeh+18] Chih-Kuan Yeh, Joon Sik Kim, Ian E.H. Yen, and Pradeep Ravikumar. "*Representer Point Selection for Explaining Deep Neural Networks*". NeurIPS. 2018.

[Zha+17] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. "*Understanding Deep Learning Requires Rethinking Generalization*". ICLR. 2017.

[ZZ22] Rui Zhang and Shihua Zhang. "*Rethinking Influence Functions of Neural Networks in the Over-Parameterized Regime*". AAAI. 2022.