Can neural networks learn to reason?

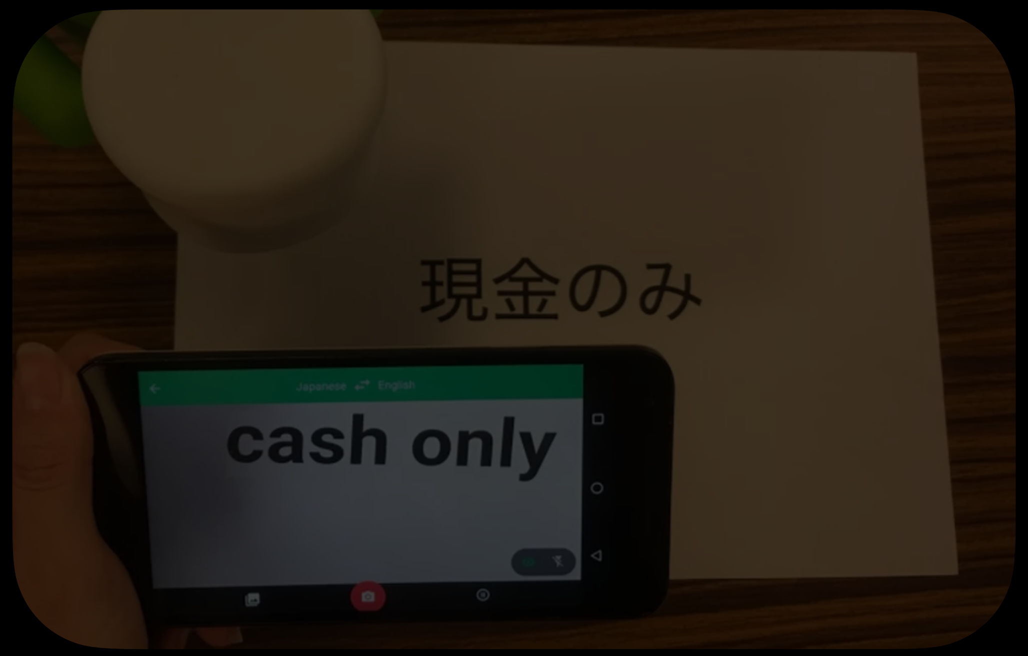
Samy Bengio | Apple Inc. | March 29th, 2022

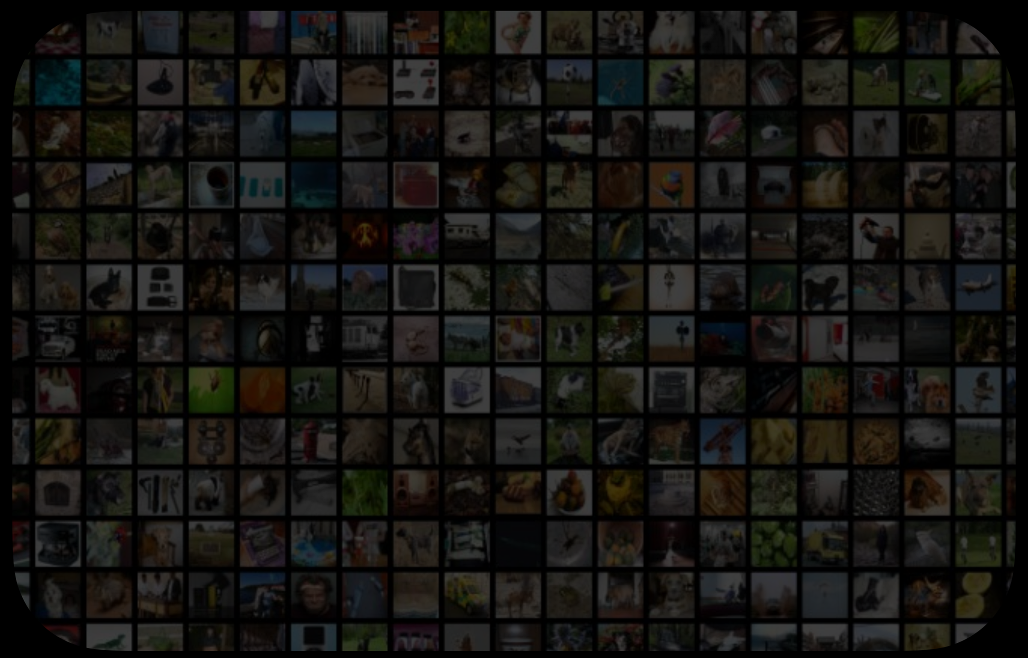
Can neural networks learn to reason?

joint work with Chiyuan Zhang, Maithra Raghu and Jon Kleinberg
https://arxiv.org/abs/2107.12580

Samy Bengio | Apple Inc. | March 29th, 2022

Generalization

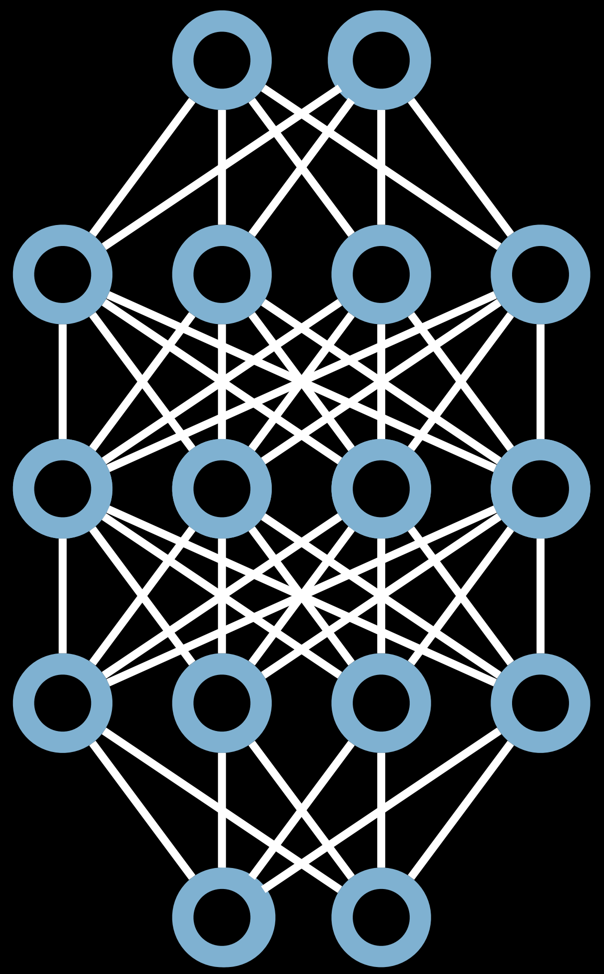




Generalization

Training (Seen) Data

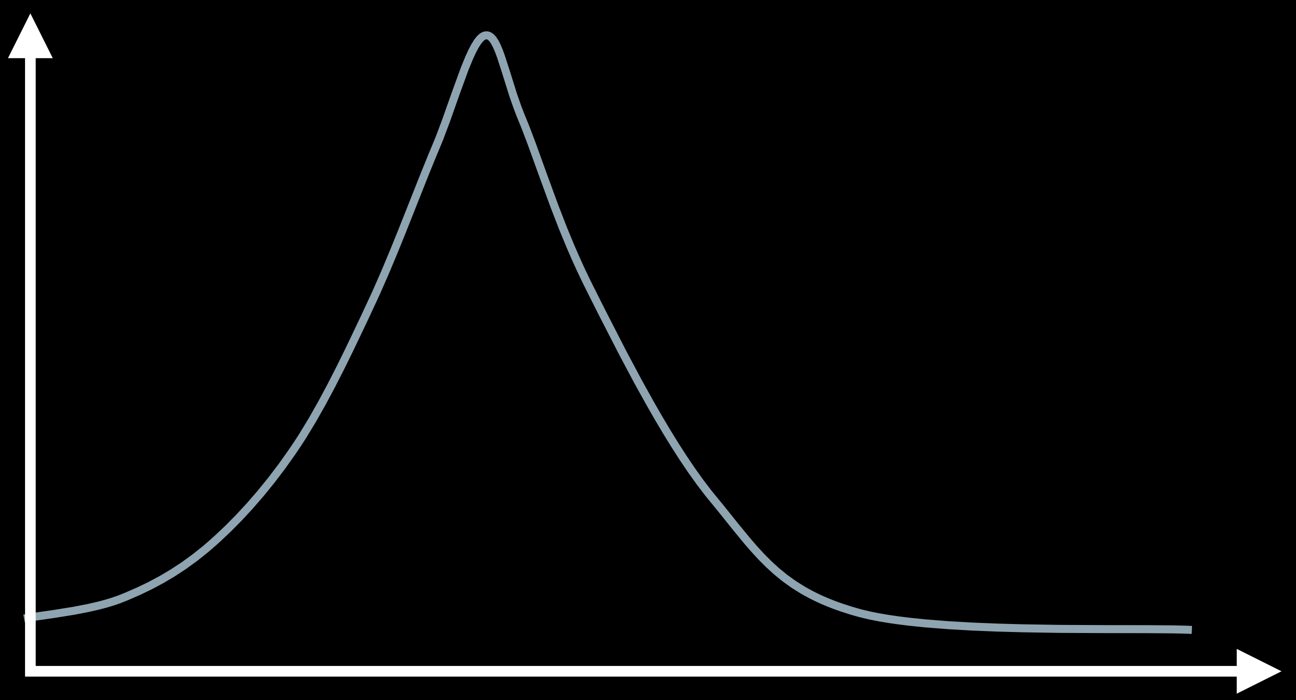
Can we be more precise?



Unseen Data

Statistical Definition of Generalization

Training (Seen) Data



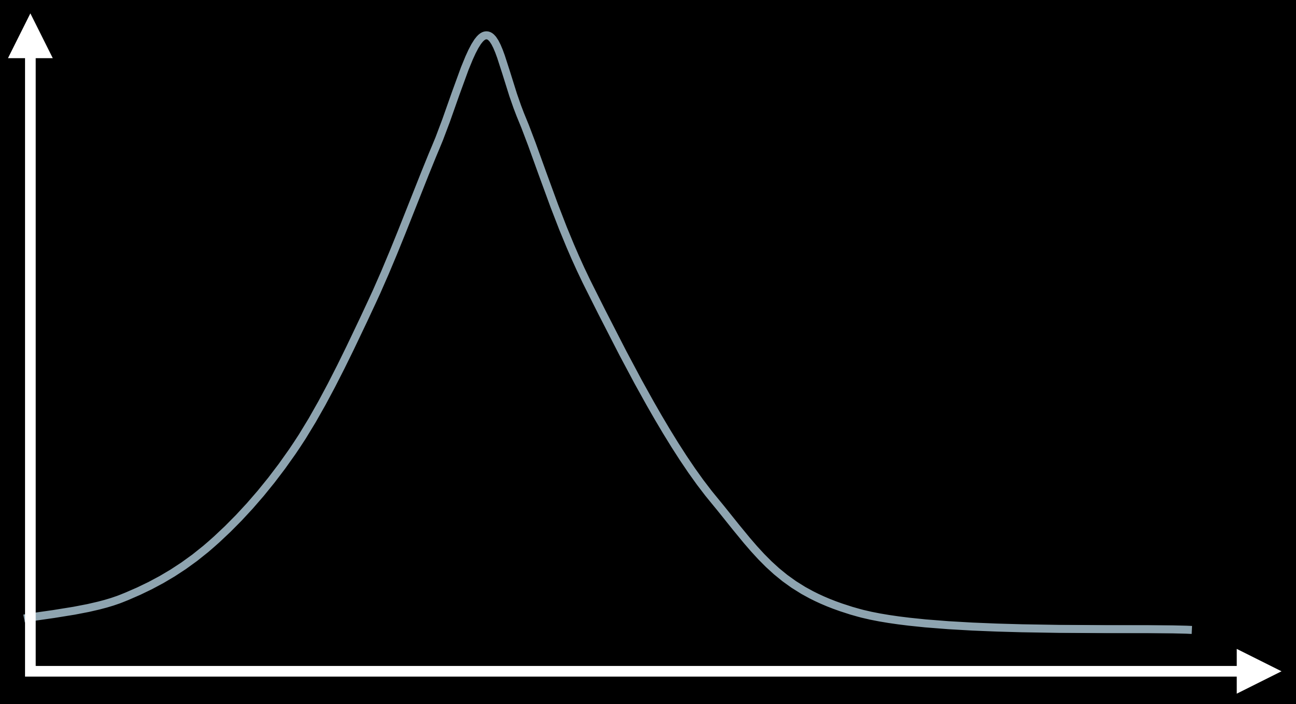
Distribution $\sim$ D

Unseen Data $\sim$ D

Generalization = works on iid data

Is that comprehensive enough?

Training (Seen) Data



Distribution ~ D

Unseen Data

Generalization

Training (Seen) Data

Unseen Data

Generalization

Generalization
(memorization, i.i.d data)

Generalization

Generalization
(memorization, i.i.d data)

Out-of-domain
Generalization

Generalization

Generalization
(memorization, i.i.d data)

Out-of-domain
Generalization

Reasoning /
Understanding

How do neural networks generalize?

Memorization is one way neural networks can generalize....

Humans rely heavily on memory when learning, e.g. learning new vocabulary, or visual directions

In Deep Learning:

- *Generalization through Memorization: Nearest Neighbor Language Models (Khandelwal et al), ICLR 2020*
- ***What Neural Networks Memorize and Why (Feldman and Zhang), NeurIPS, 2020***
  - Memorization of rare instances could be helpful for generalization

TRAIN TEST IMAGES

Methods of Generalization

Naive memorization | Remembering rare examples | K-nearest neighbors | IID Data | Out of domain | Abstract reasoning

Simple ↔ **Sophisticated**

Questions and Challenges

- How do neural networks typically generalize?
- Can we differentiate between simple generalization and sophisticated generalization?
- What are the limits of neural network generalization?

Pointer Value Retrieval

- New family of tasks to understand neural network generalization
- Varying types of input data
  - Our paper: *image* and *vector* inputs
- Can control and vary task difficulty
- All tasks have a simple pointer-value reasoning rule:
  - A specific position of the input acts as a pointer
  - The value of the pointer provides instruction on which other position(s) of the input to look at
  - The said values are aggregated to produce the final output.

Pointer Value Retrieval

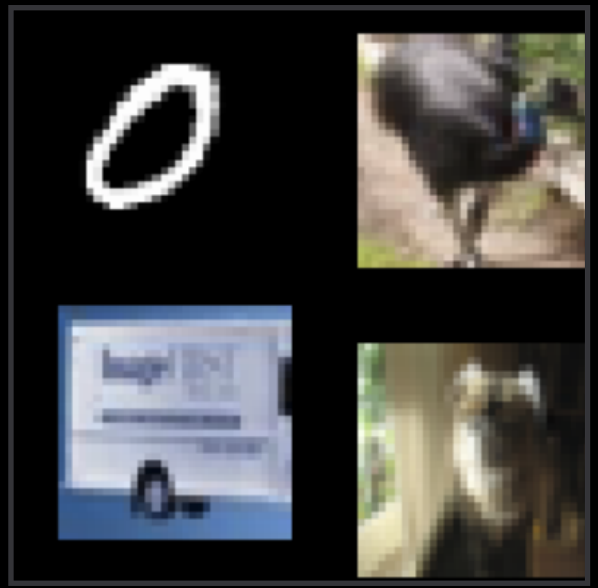
Visual Inputs

Block Style

Label 9

Label 3

Label 2



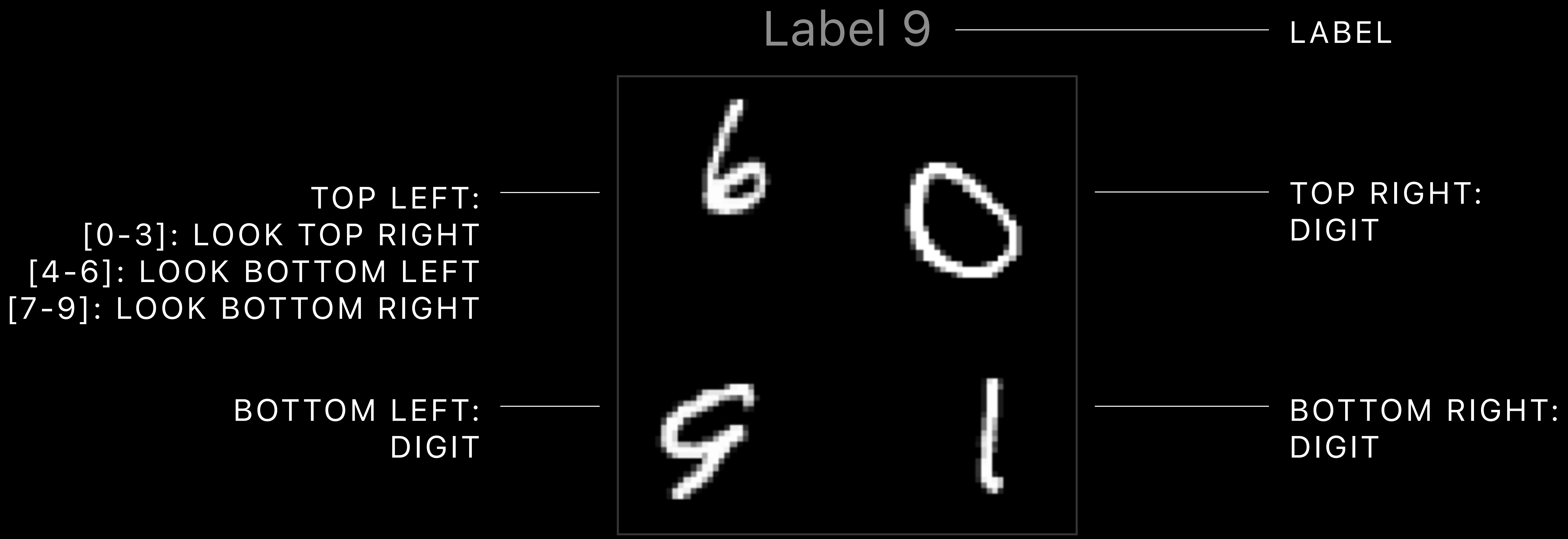
Label 8

Sequential

Label 6

Pointer Value Retrieval

Visual (Block Style) Input



Pointer Value Retrieval

Visual (Block Style) Input

Label 9

LABEL

TOP LEFT:
[0-3]: LOOK TOP RIGHT
[4-6]: LOOK BOTTOM LEFT
[7-9]: LOOK BOTTOM RIGHT

TOP RIGHT:
DIGIT

BOTTOM LEFT:
DIGIT

BOTTOM RIGHT:
DIGIT

Decouple vision and generalization via reasoning?

Pointer Value Retrieval

Vector Inputs

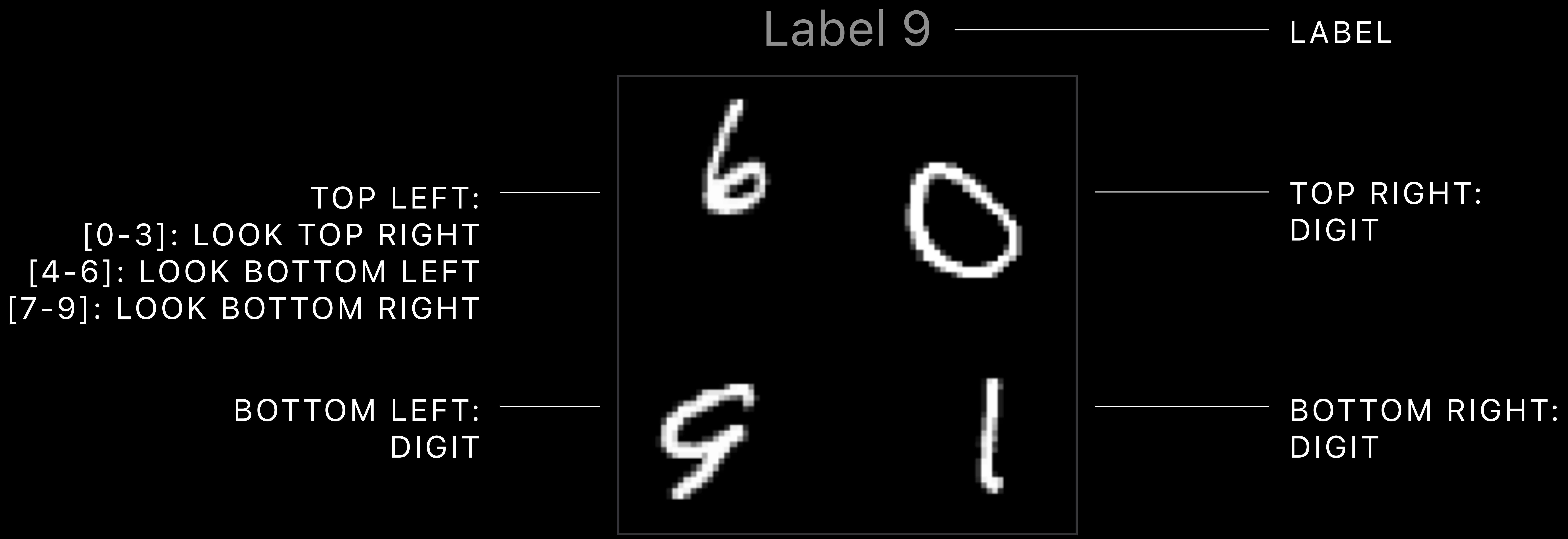
Pointer Value Retrieval

Varying Task Complexity

- Distribution shift between training and test data
    - Some values don’t appear at some positions
    - Call this: *Holdout Shift*
- Increase functional complexity
    - Mapping from value to label is more complex

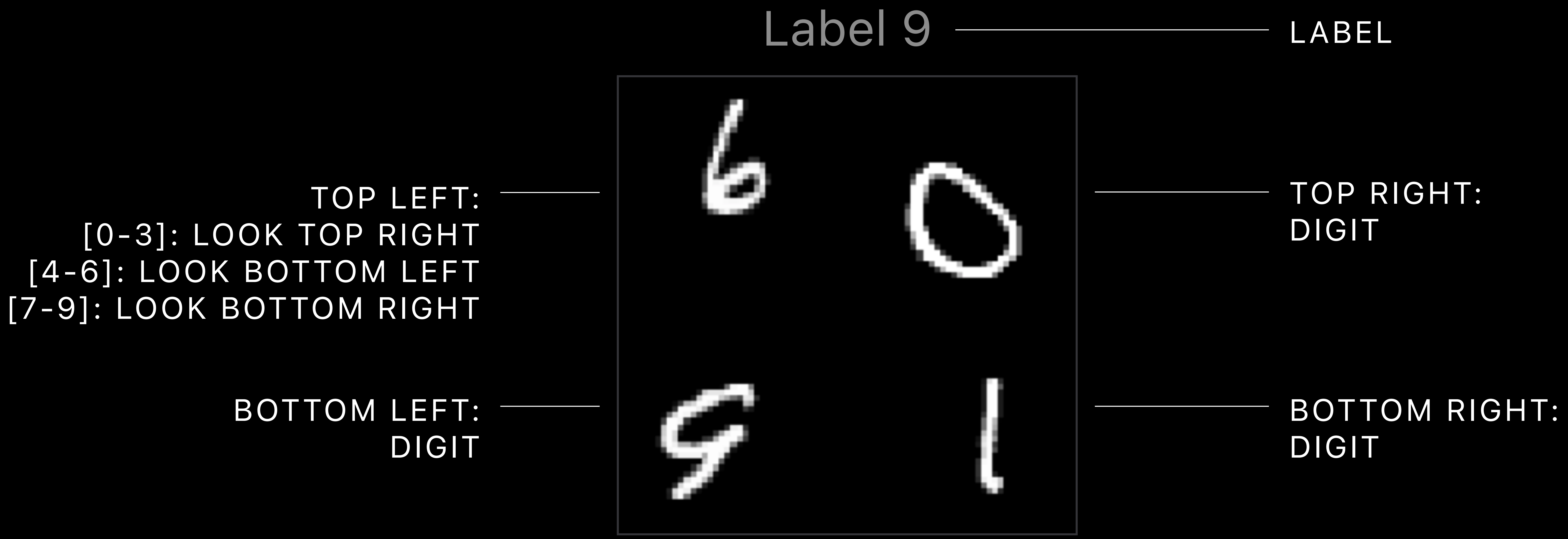
Pointer Value Retrieval

Visual (Block Style) Input



Pointer Value Retrieval

Visual (Block Style) Input



Training

Test

Pointer Value Retrieval

Visual (Block Style) Input

Label 9 — LABEL

VALUES: [0-9]

TOP LEFT:
[0-3]: LOOK TOP RIGHT
[4-6]: LOOK BOTTOM LEFT
[7-9]: LOOK BOTTOM RIGHT

TOP RIGHT:
DIGIT

VALUES: [0-9]

BOTTOM LEFT:
DIGIT

VALUES: [0-9]

BOTTOM RIGHT:
DIGIT

VALUES: [0-9]

Training | Test | IID

Pointer Value Retrieval

Visual (Block Style) Input

Label 9 — LABEL

VALUES: [0-9]

TOP LEFT:
[0-3]: LOOK TOP RIGHT
[4-6]: LOOK BOTTOM LEFT
[7-9]: LOOK BOTTOM RIGHT

TOP RIGHT:
DIGIT

VALUES: [4-9]

BOTTOM LEFT:
DIGIT

VALUES: [0-3], [7-9]

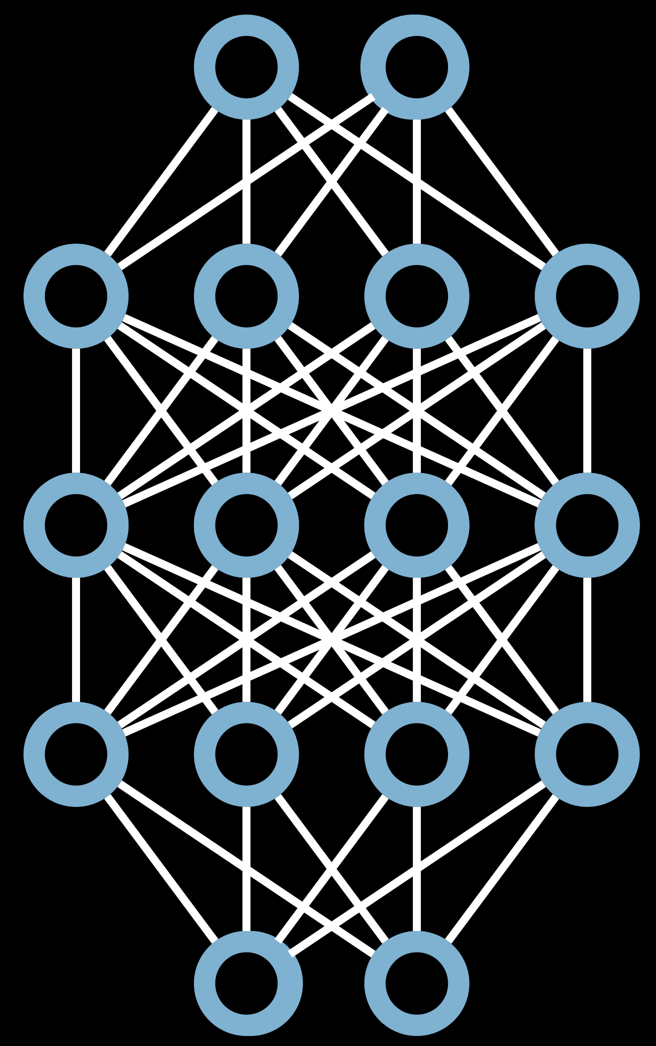
BOTTOM RIGHT:
DIGIT

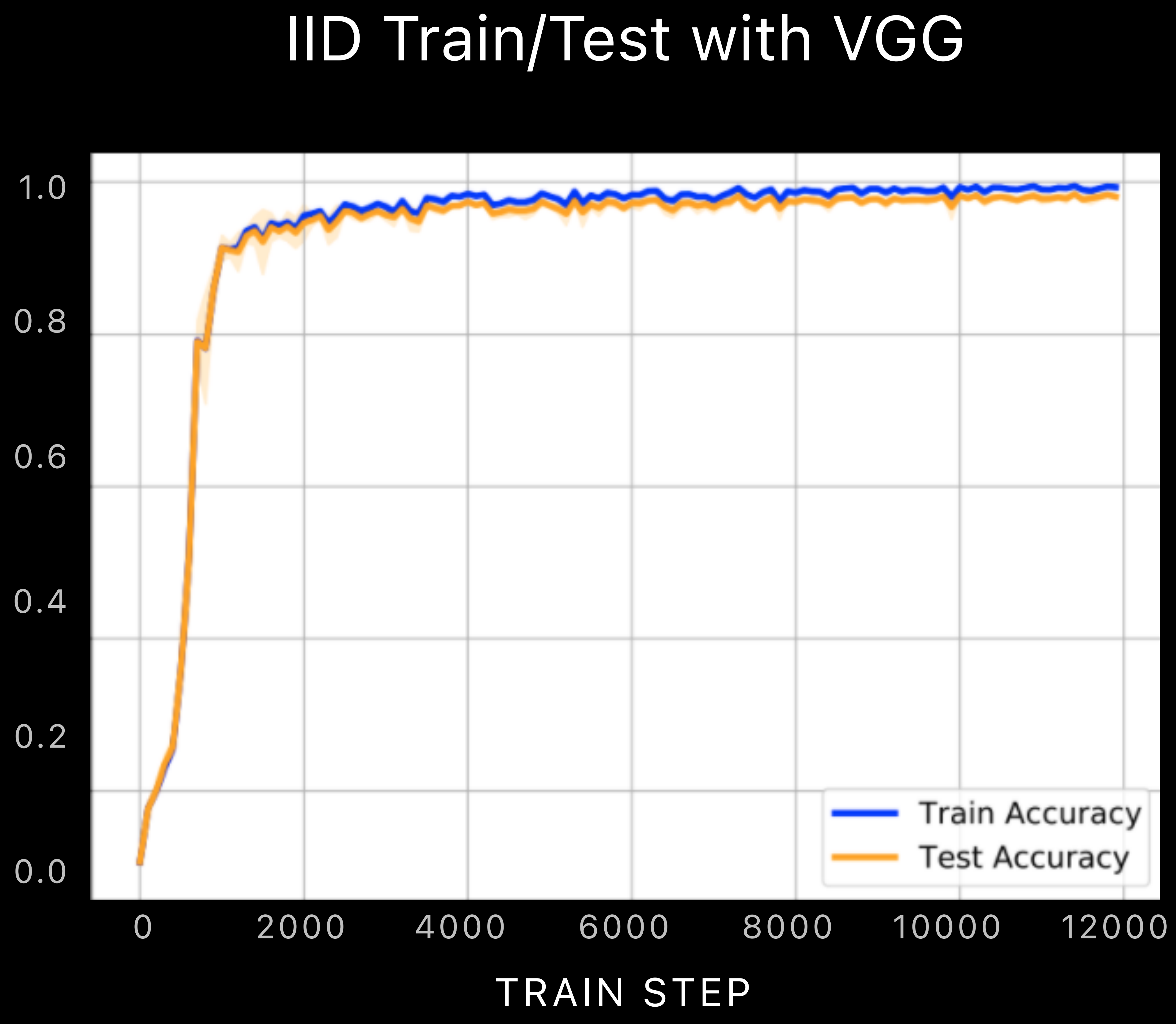
VALUES: [1-6]

Training | Test | Holdout Shift

Generalization on PVR Block Task

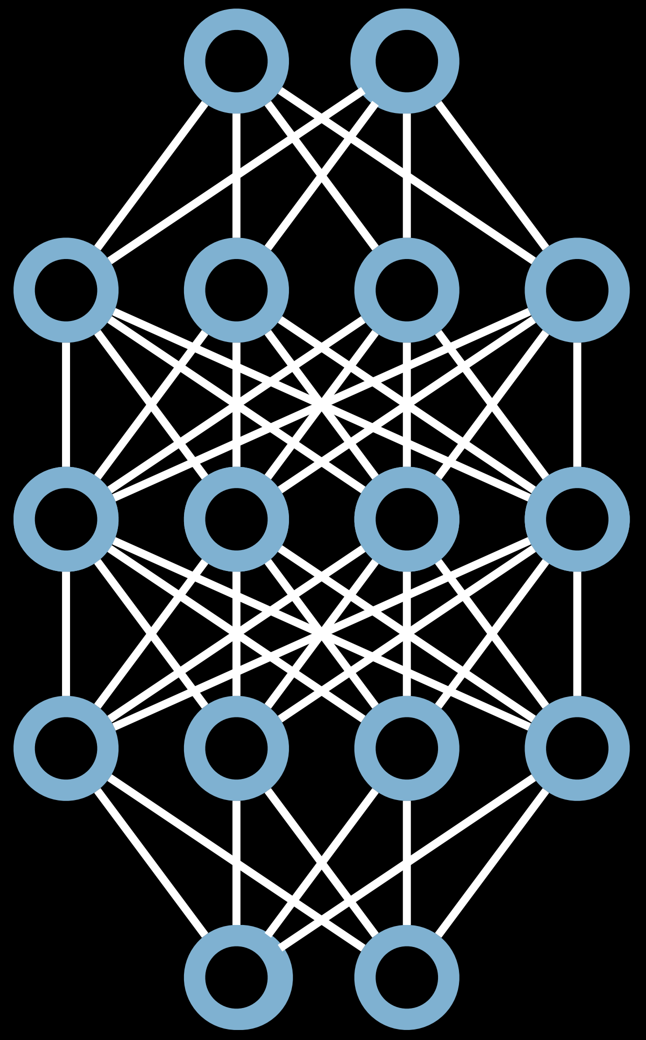
IID



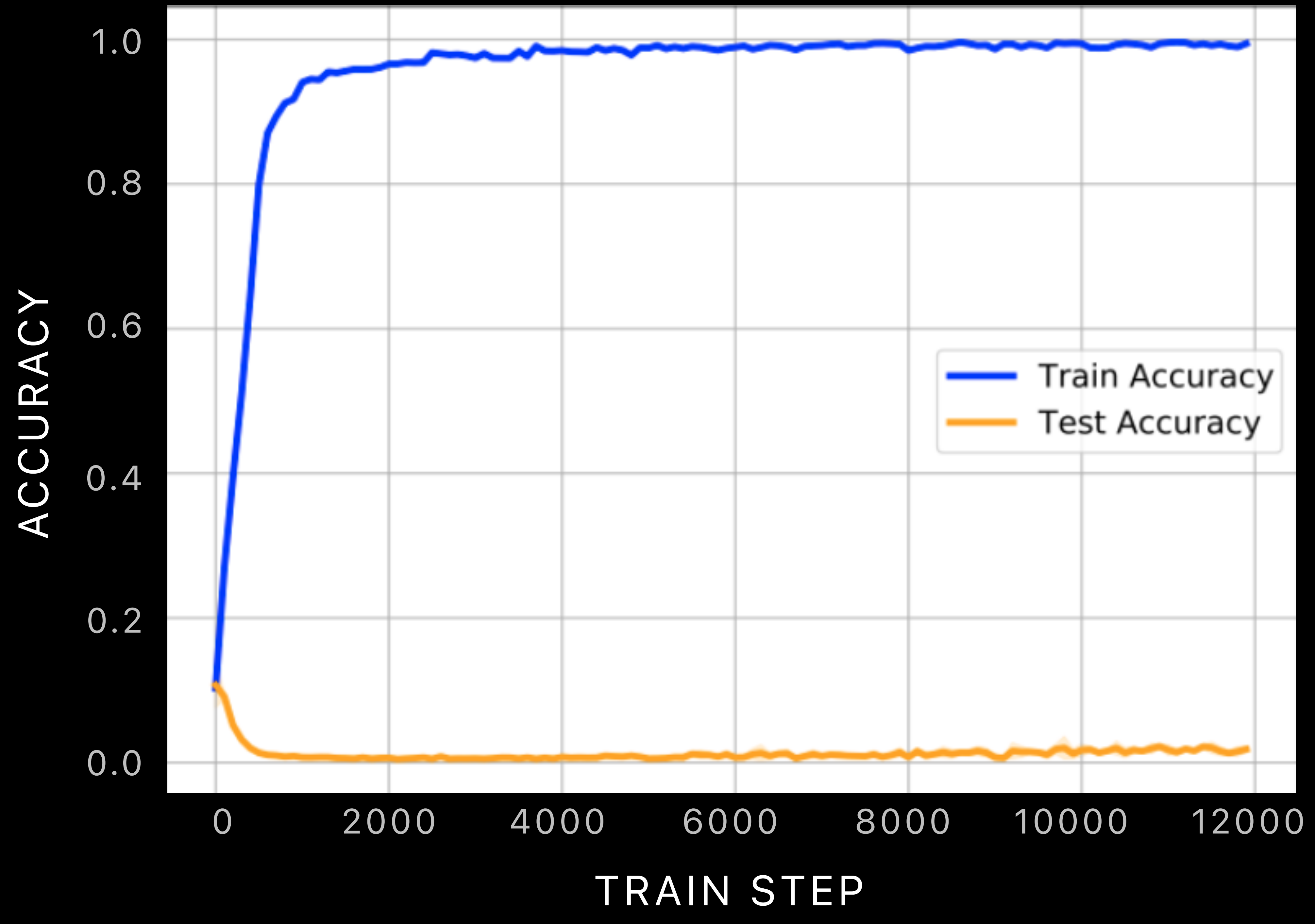


Generalization on PVR Block Task

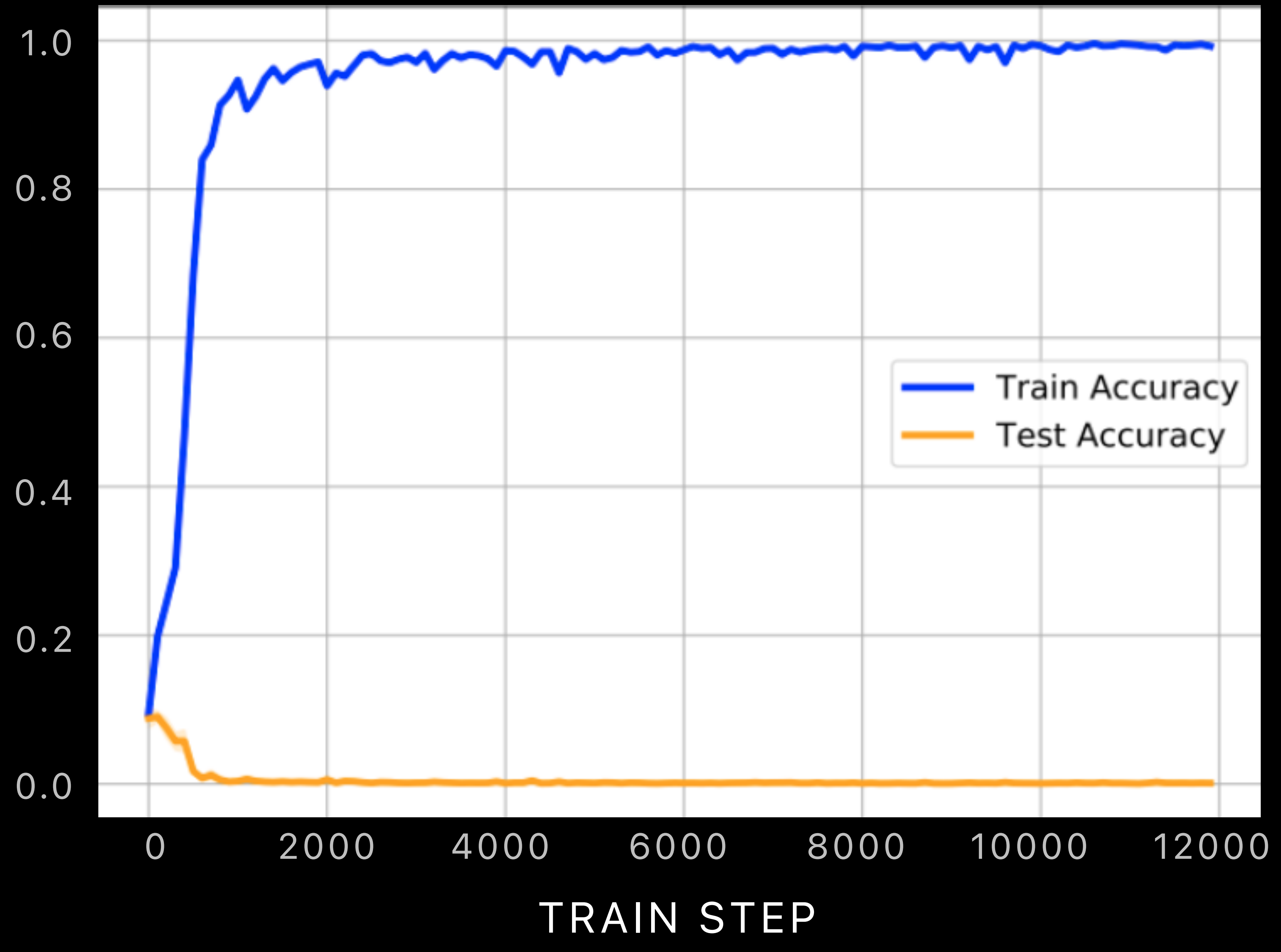
Holdout Shift



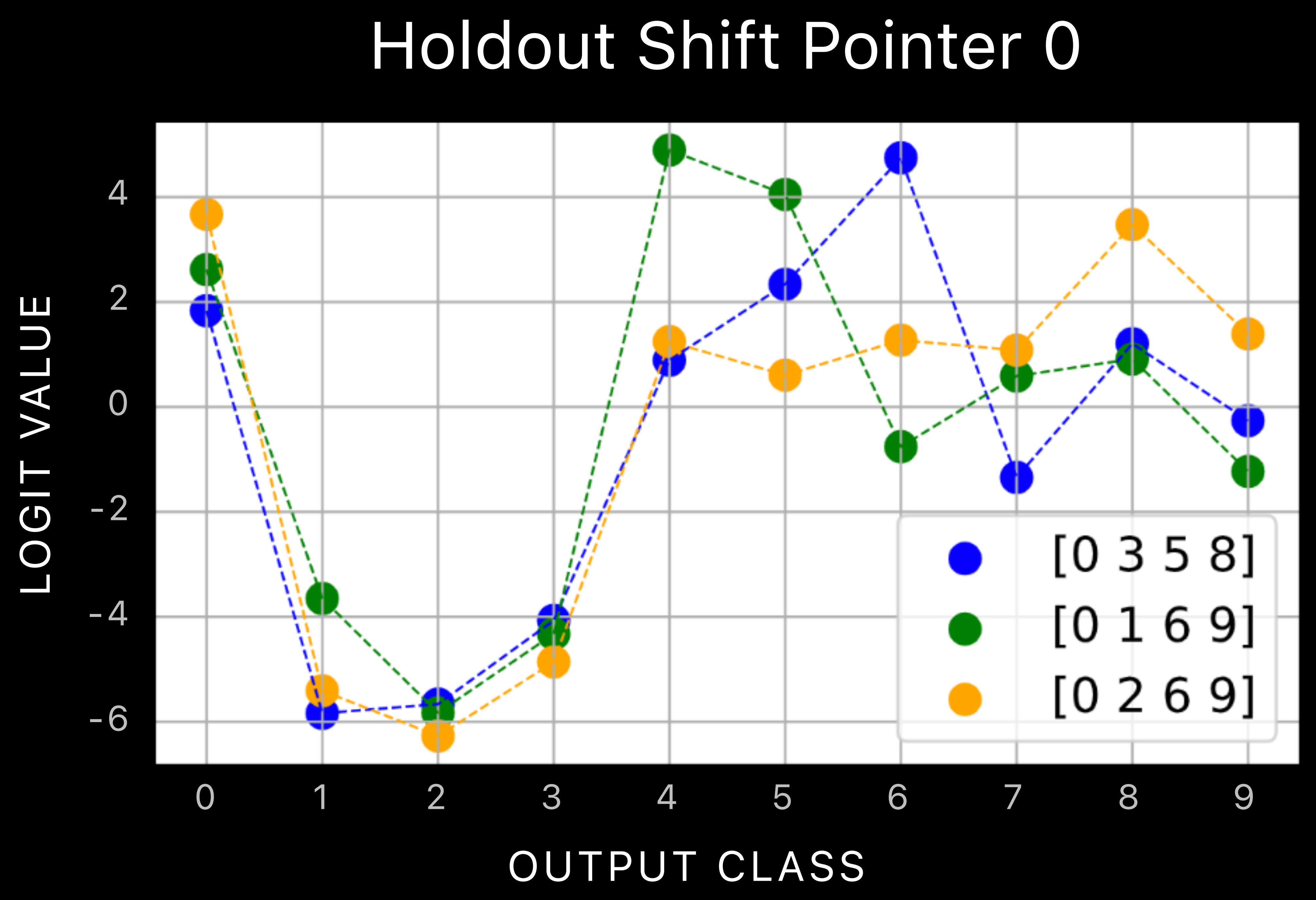
Different Distribution
Train/Test with ResNet

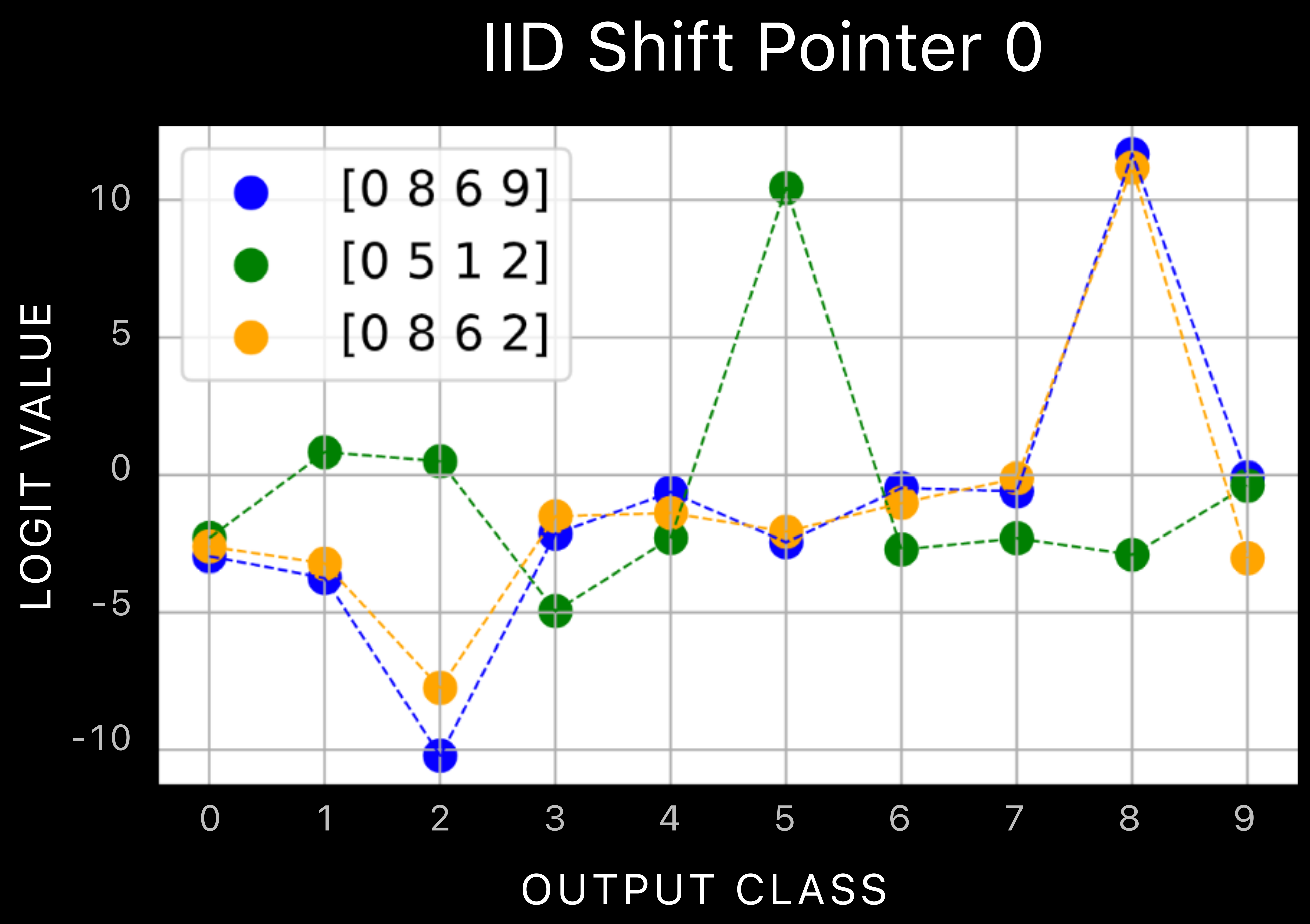


Different Distribution
Train/Test with VGG



Raw logit values for test examples for pointer digit 0

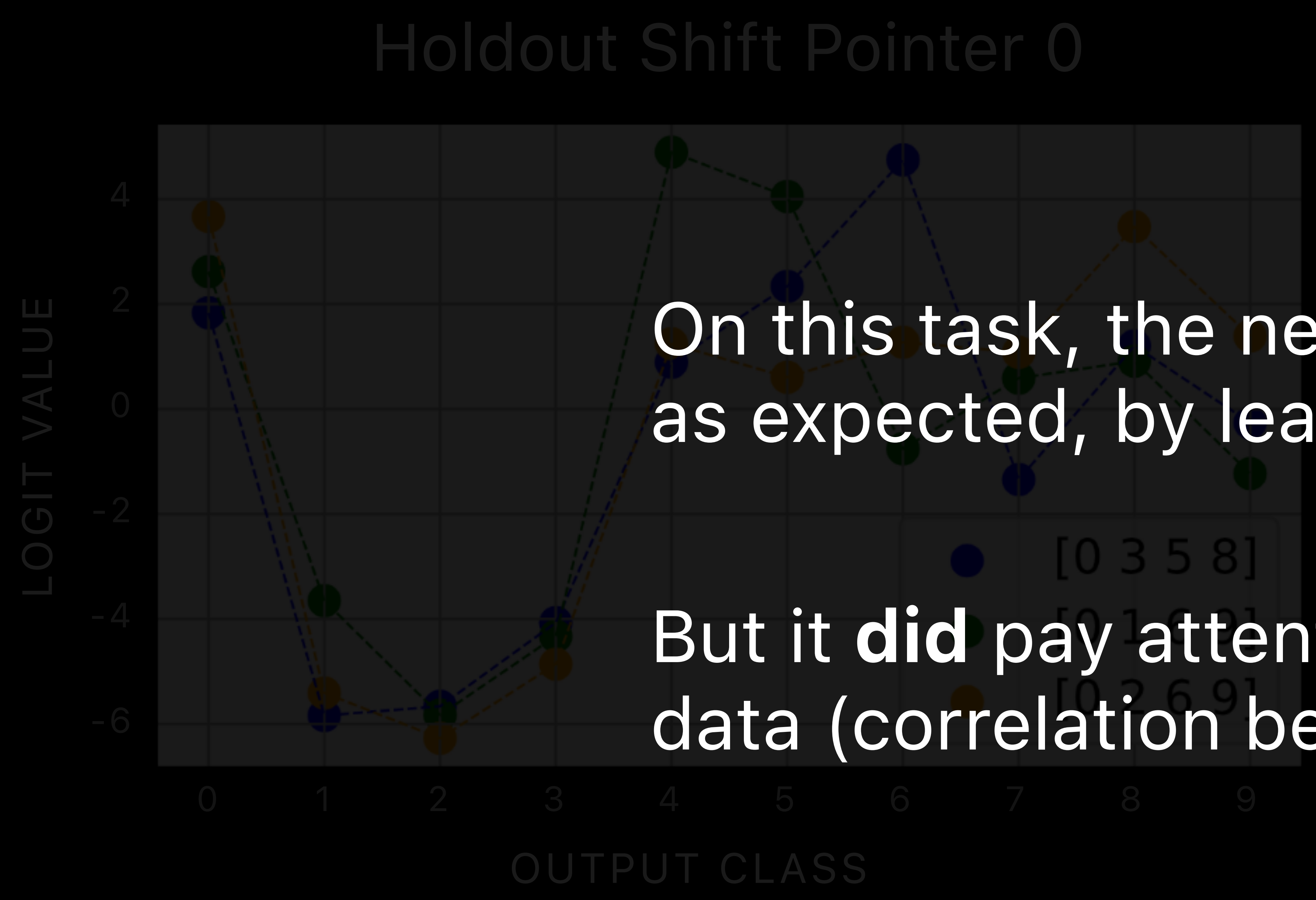


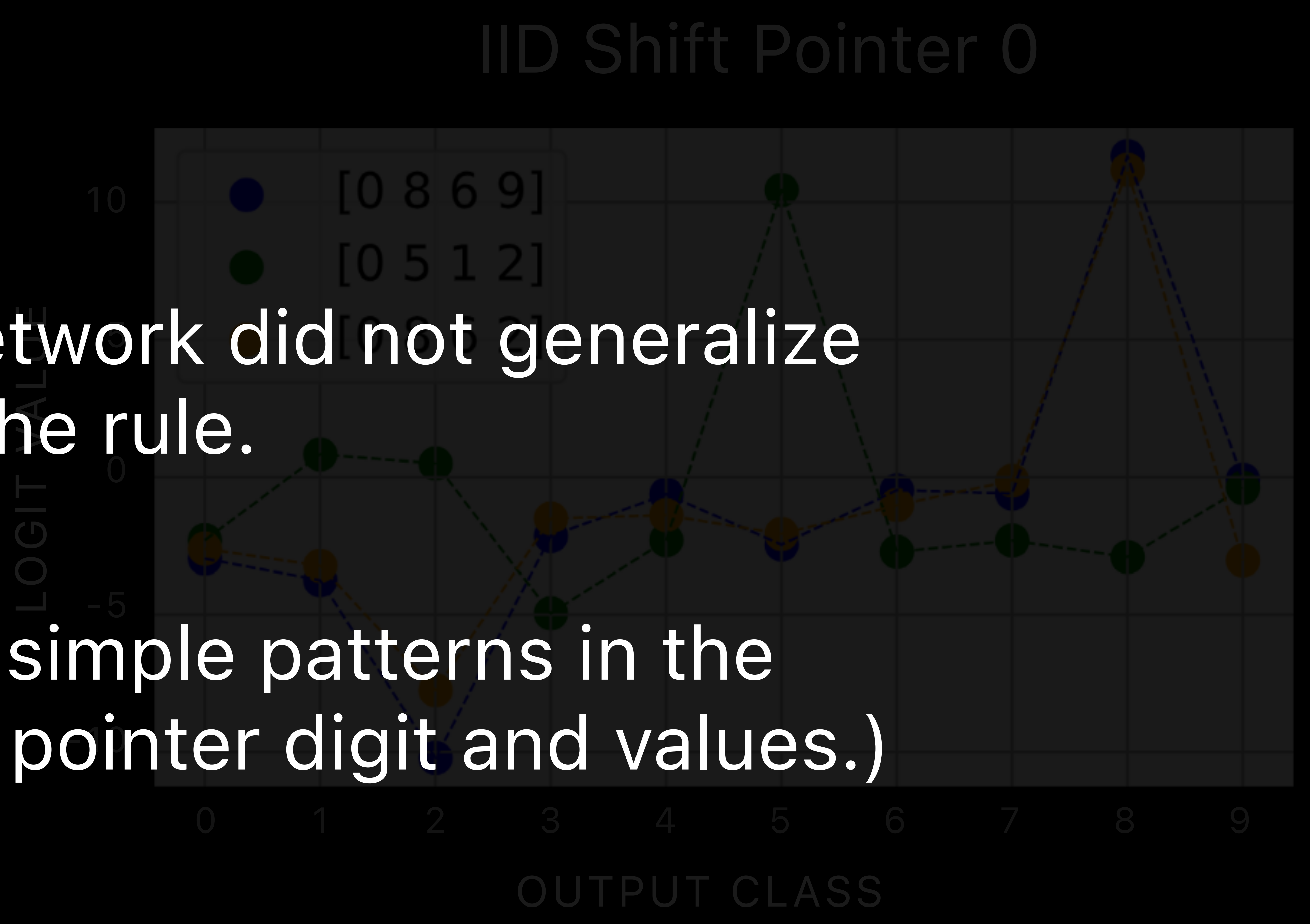


The model has learned to assign very low logits to labels 1-3, exactly the values left out from the top right position during training (which pointer 0 points to). Although all test examples have only values 1-3 in this position, this correlation is ingrained in the network, leading to systematic errors.

Comparison: logits from models trained in the IID setting, where we observe no correlations between pointer digit and label values.

Raw logit values for test examples for pointer digit 0





On this task, the neural network did not generalize as expected, by learning the rule.

But it **did** pay attention to simple patterns in the data (correlation between pointer digit and values.)

Is this memorization? Or reasoning?

The model has learned to assign very low logits to labels 1-3, exactly the values left out from the top right position during training (which pointer 0 points to). Although all test examples have only values 1-3 in this position, this correlation is ingrained in the network, leading to systematic errors.

Comparison: logits from models trained in the IID setting, where we observe no correlations between pointer digit and label values.

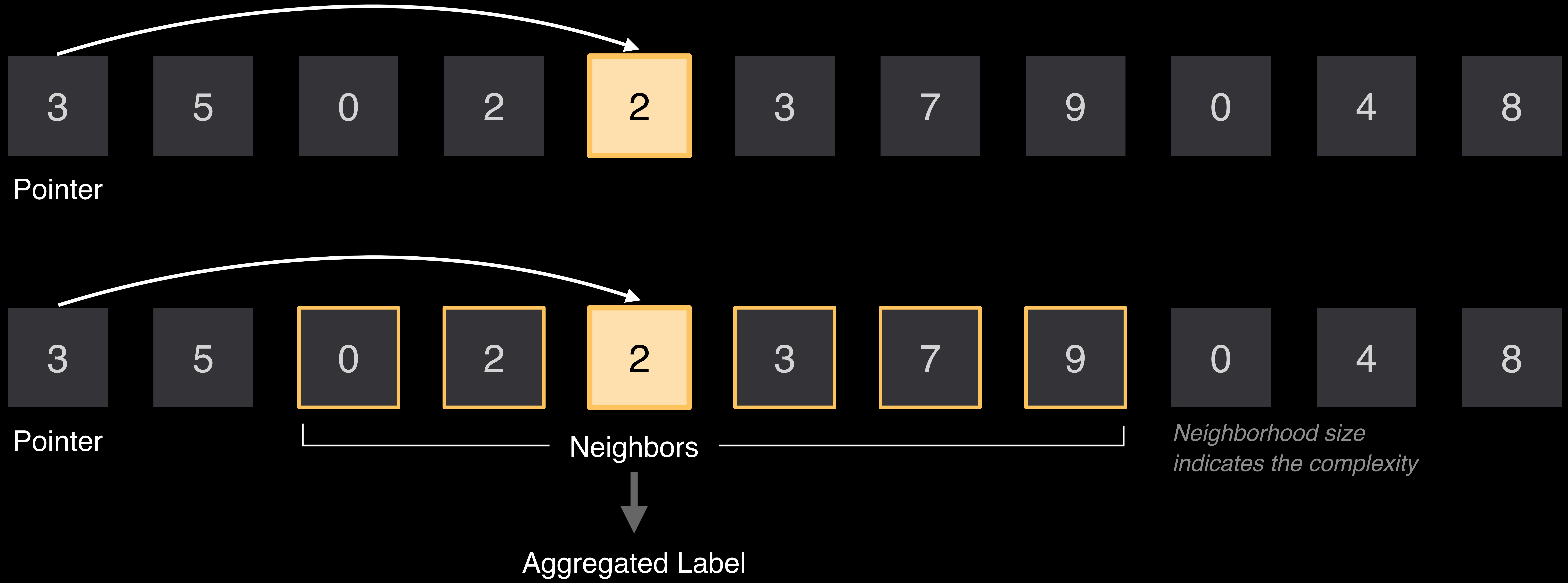
Vector Inputs and Varying Difficulty

How to avoid simple data patterns and distinguish between memorization and reasoning?

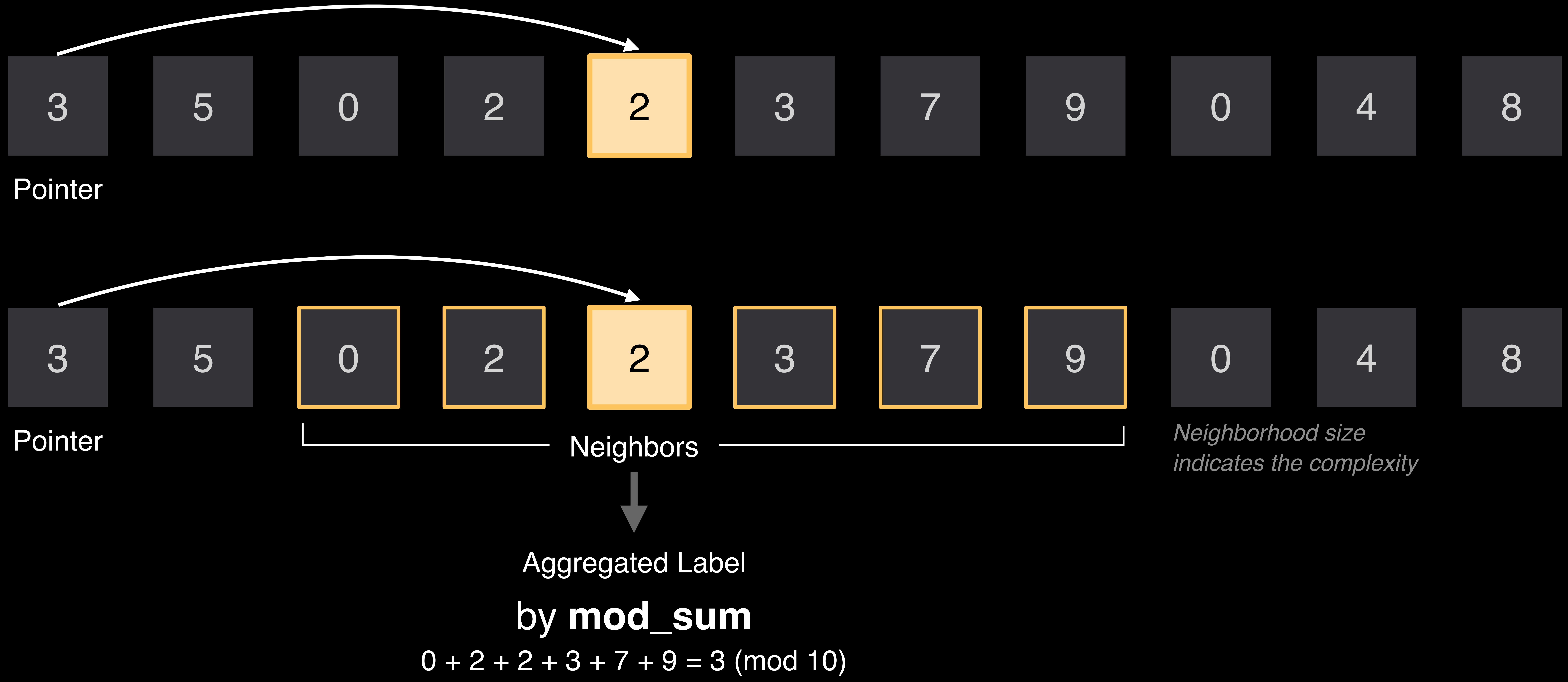
Increase task difficulty through ***functional complexity*** *(with vectorized inputs)*

Vector Inputs and Varying Difficulty

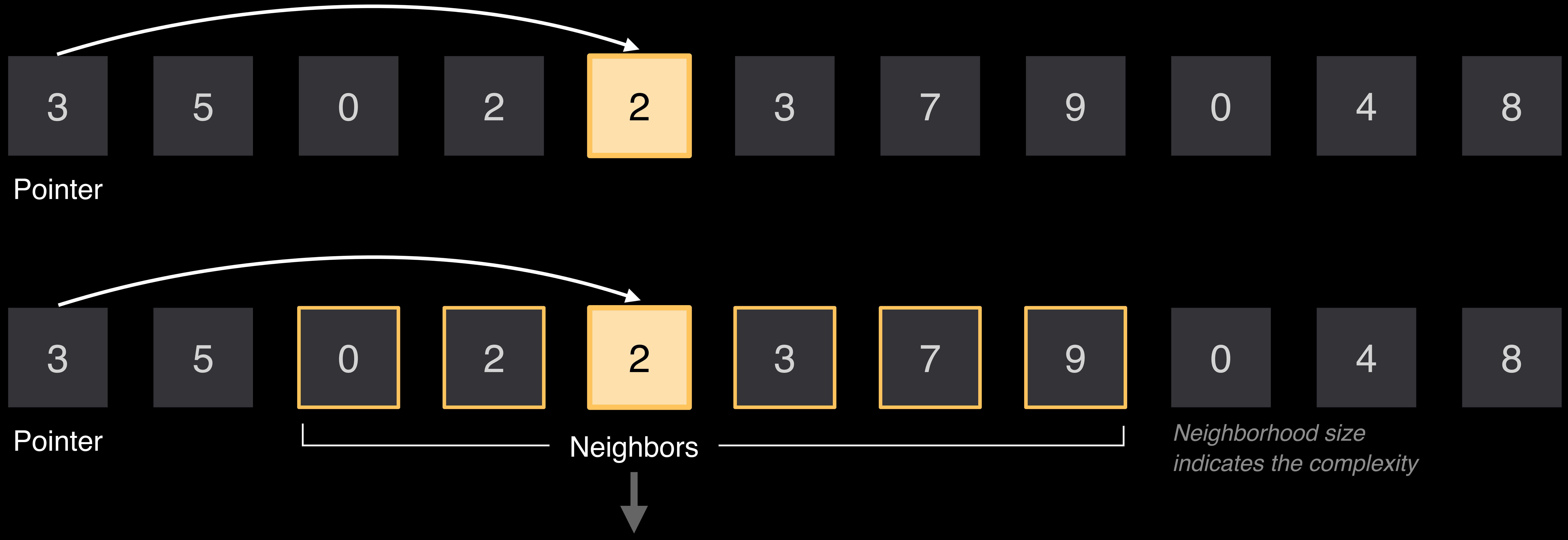
Vector Inputs and Varying Difficulty



Vector Inputs and Varying Difficulty



Vector Inputs and Varying Difficulty



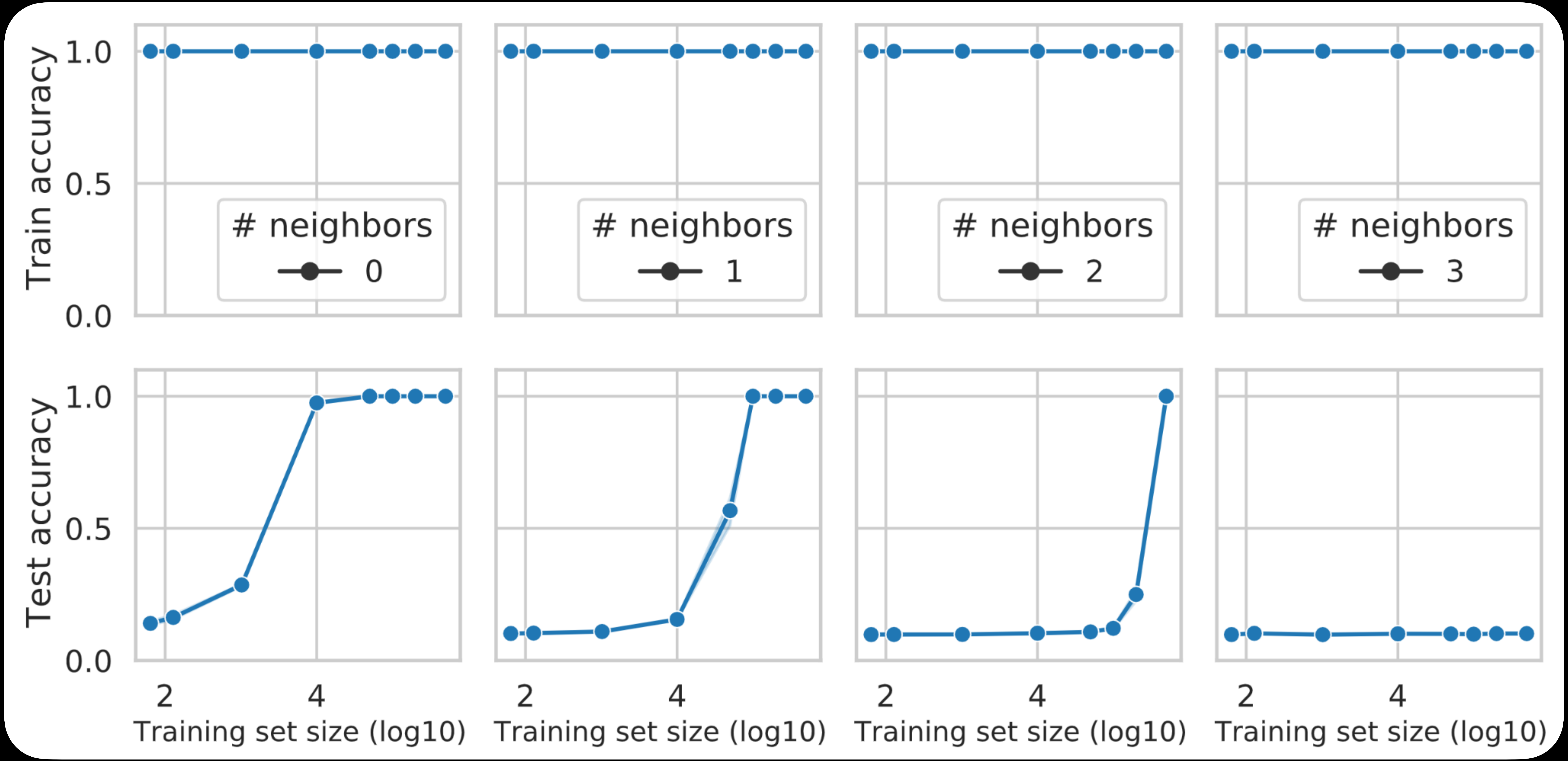
Aggregated Label

by **mod_sum**

$0 + 2 + 2 + 3 + 7 + 9 = 3 \pmod{10}$

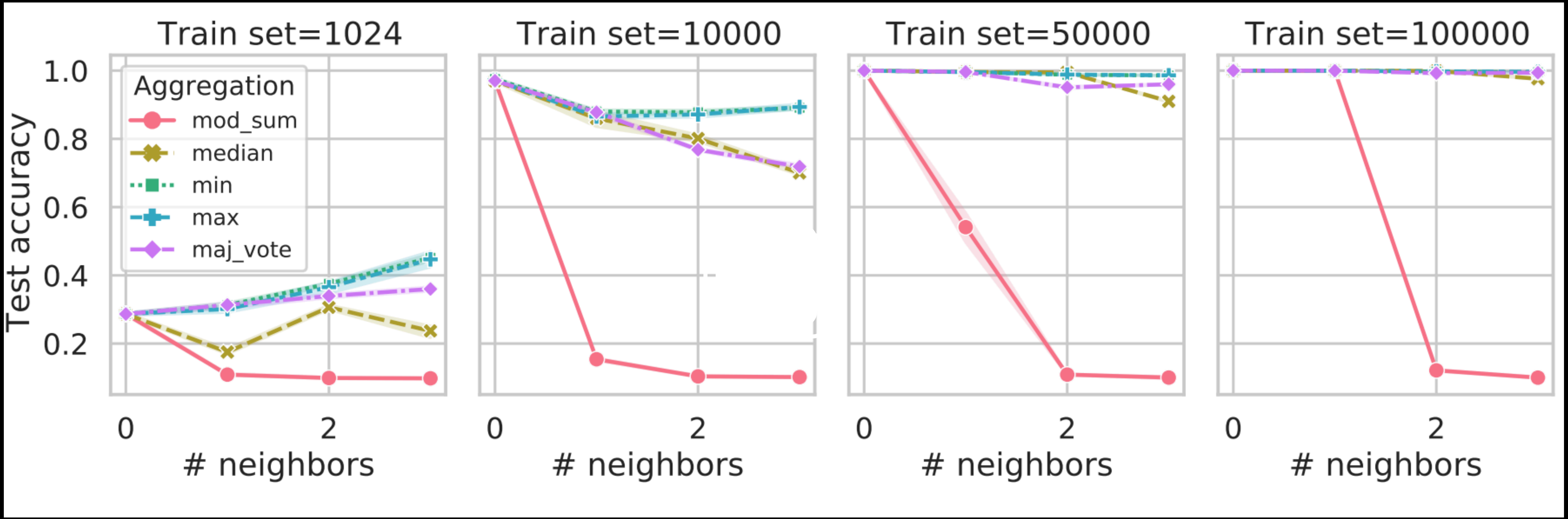
(alternatives) **maj_vote, min, max, median…**

Performance of PVR Tasks with different Complexity



The training (top) and test (bottom) accuracy of PVR tasks with increasing functional complexity and different training set sizes.

Evaluating Different Aggregating Functions for PVR

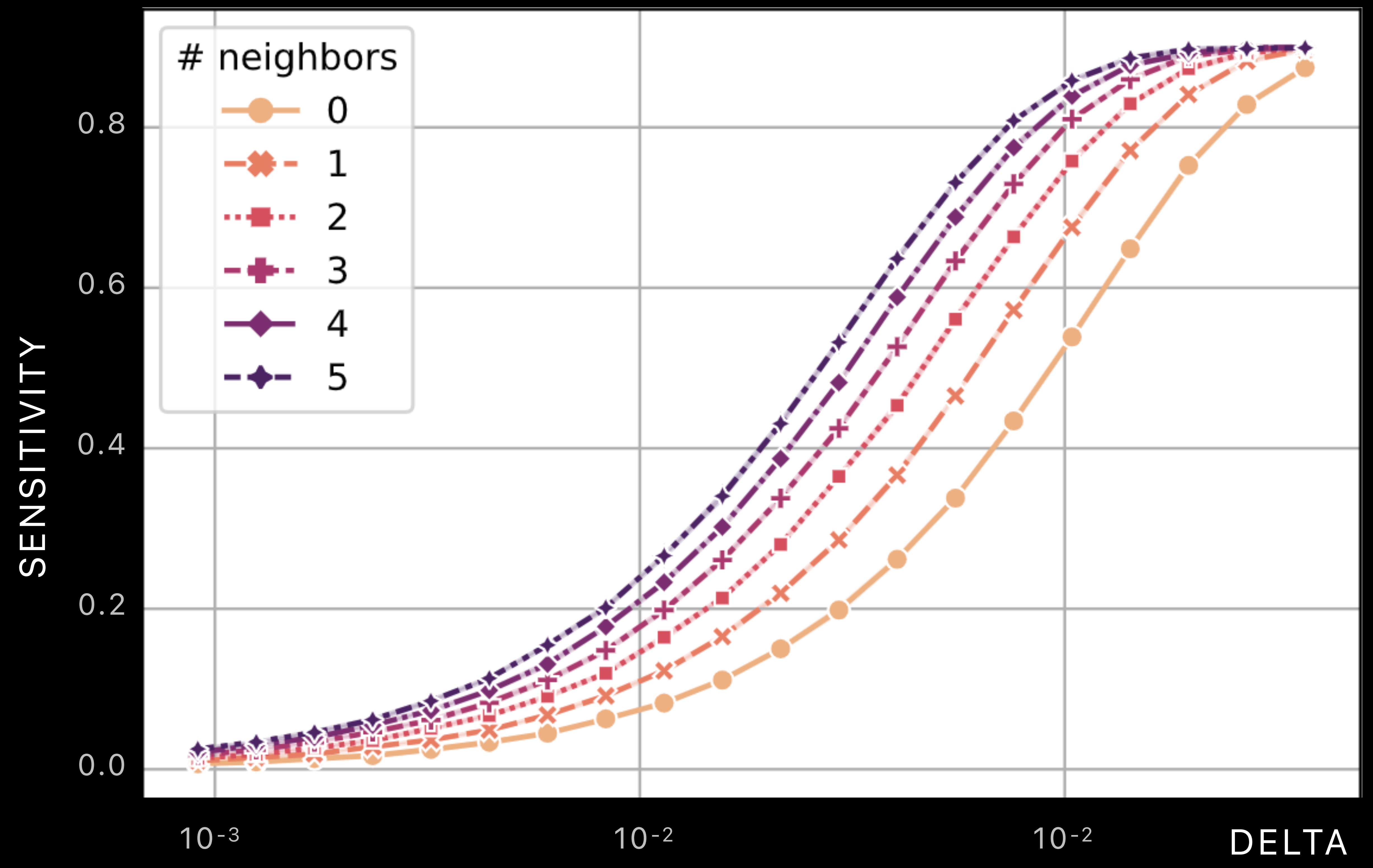


Test performance for different aggregation functions across varying dataset size and functional complexity. The empirical results support the intuitive observation that mod_sum is the most challenging.

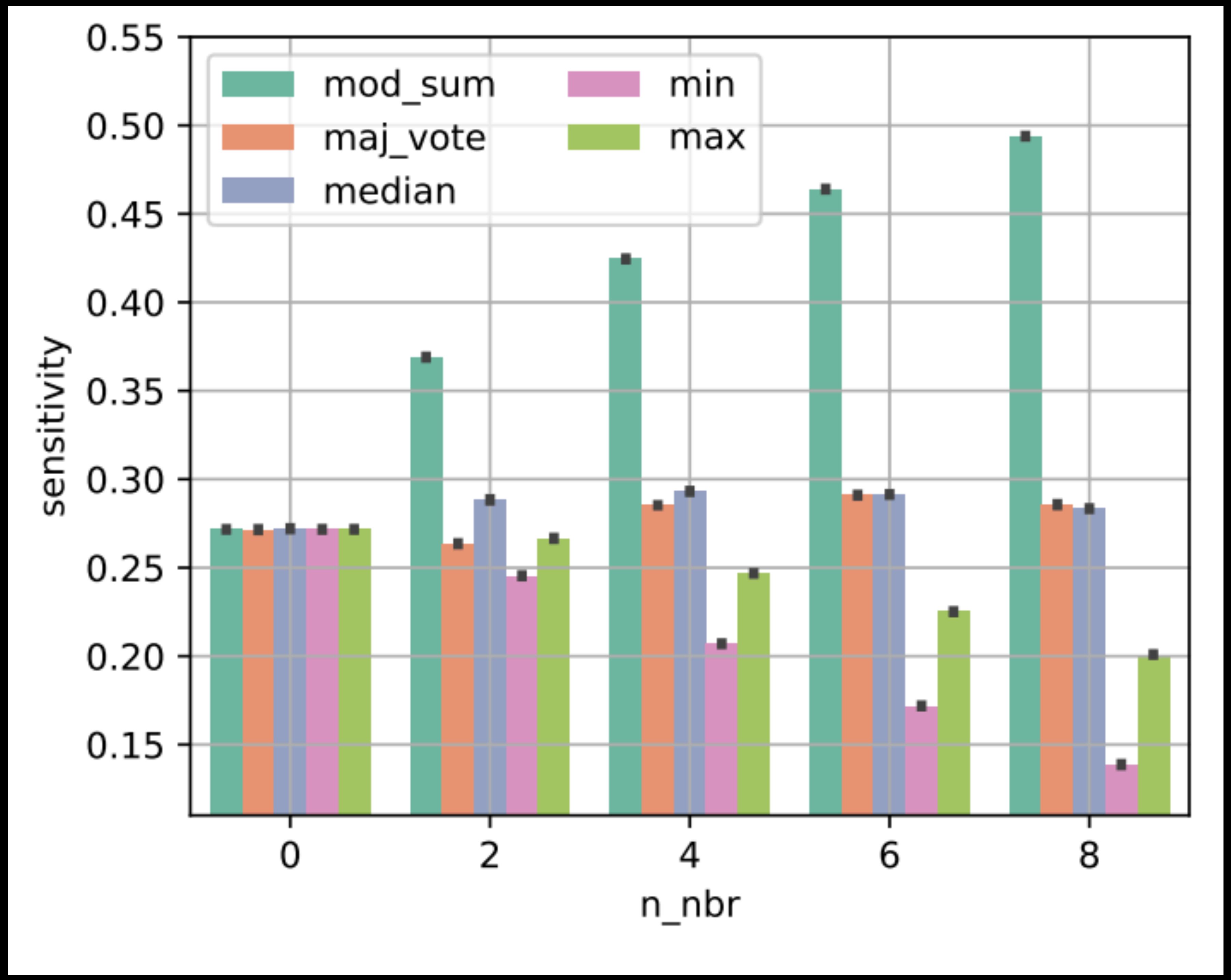
Noise Sensitivity of Boolean Functions

- Using Noise sensitivity to quantify the complexity of the tasks.
- Intuitively, measures how sensitive the outcome of a boolean function f to random perturbations with probability $0 < \delta < 1$
- Noise sensitivity of f at δ is defined to be the probability that $f(x) \neq f(y)$ when x is uniform random bits and y is formed from x by reversing each bit independently with probability δ.
- We encode each digit of the input vector with 4 bits.

Noise Sensitivity of Boolean Functions

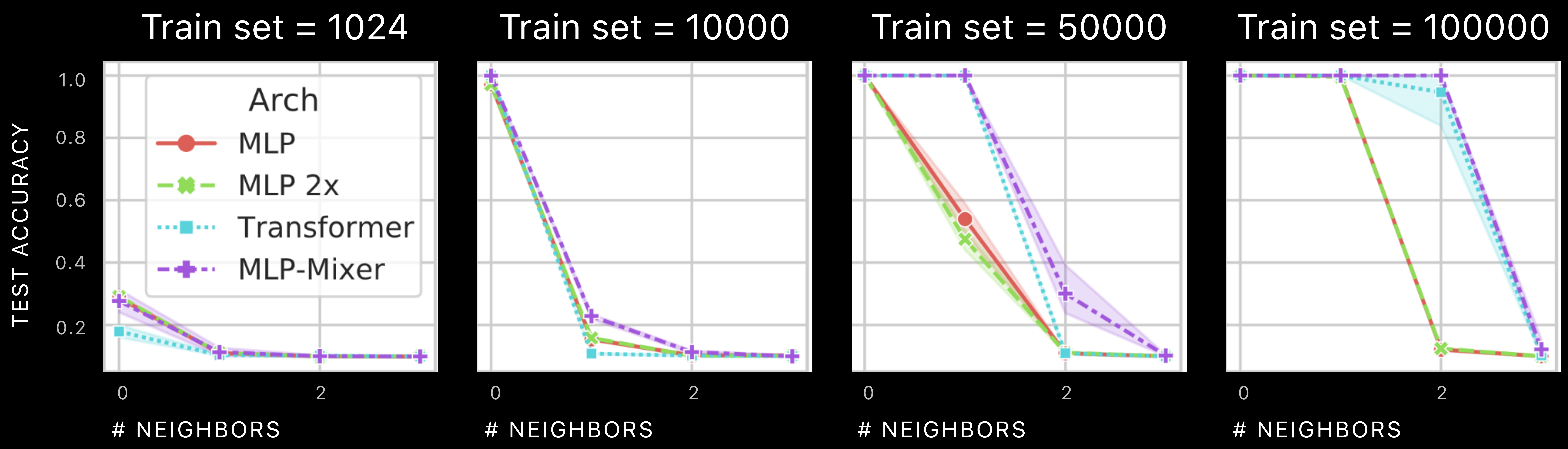


Noise sensitivity analysis confirms that the complexity of PVR tasks with mod_sum increases with neighbor sizes.



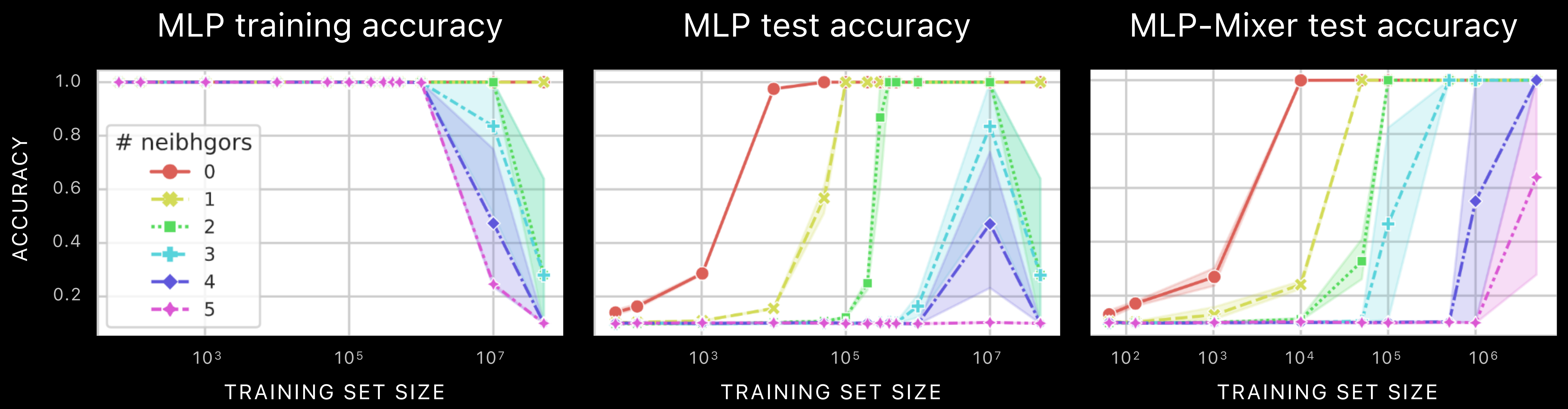
Average noise sensitivity over the same value range of δ across a range of different aggregation choices.

Model Architecture and Inductive Biases



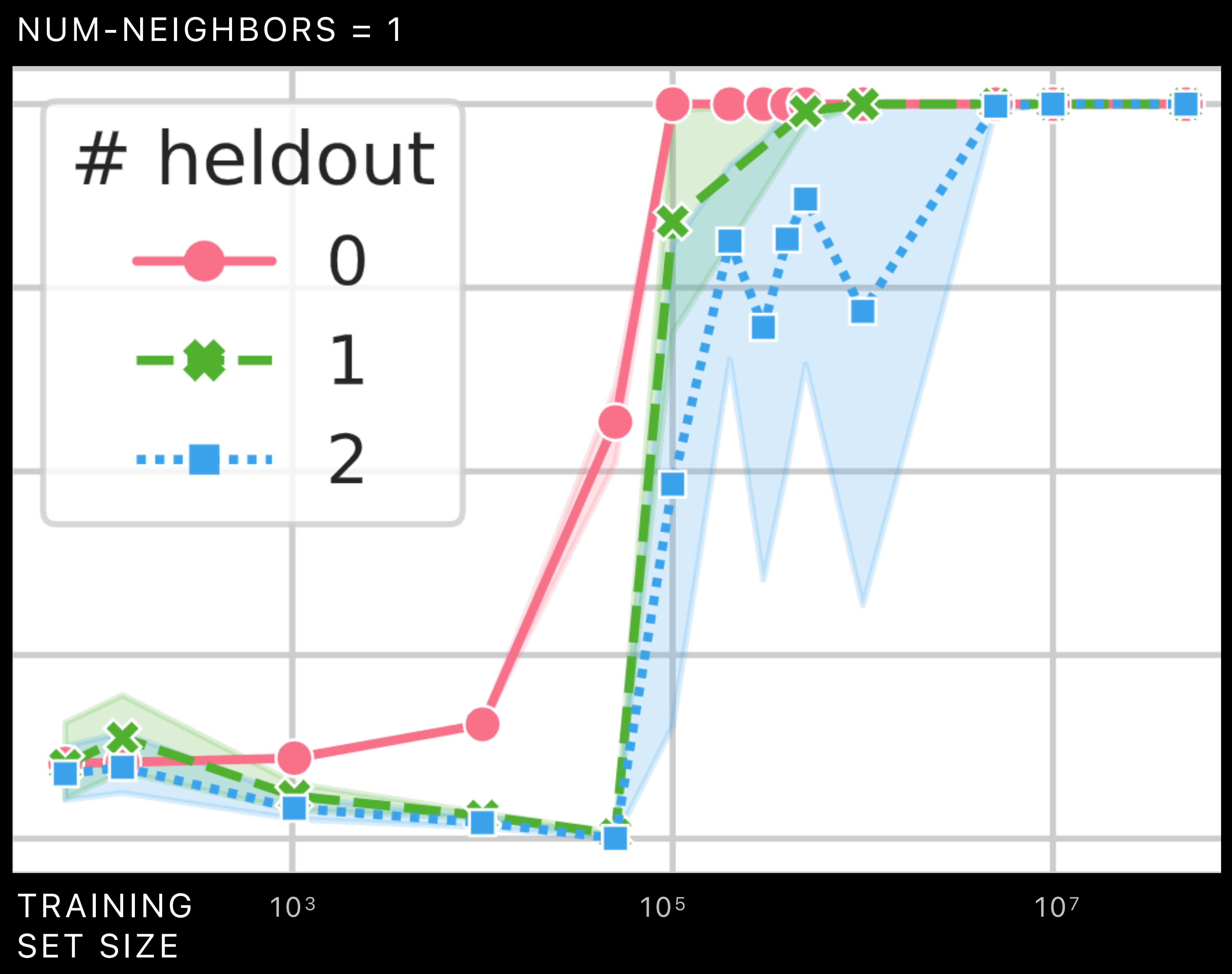
Transformers and MLP-Mixers have explicit notion of tokens and the interaction of tokens, and have better sample complexity (requires fewer training examples to generalize) than MLPs.

Training with Massive Dataset Sizes



To test the limits of neural network learning, we look at training with massive dataset sizes, up to 5x10^7: continuing performance improvements as dataset size is increased, solving more and more complex tasks.

Does high test accuracy correspond to learning reasoning?

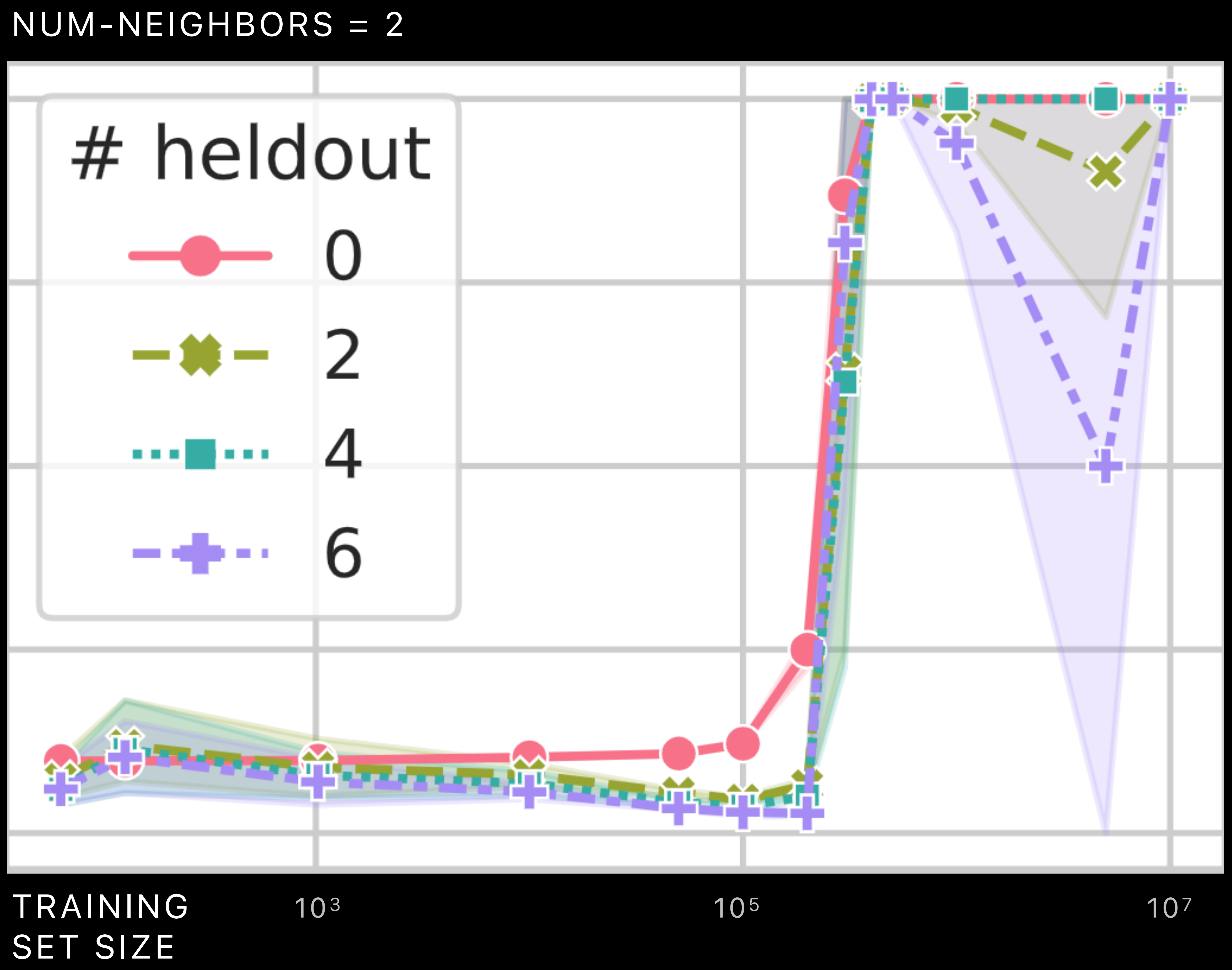


We train neural networks on PVR tasks with complexity m=1, and held out various number of permutations of (0,1) in the value window.

We test on inputs where the value window contains the value (0,1), while all other digits and the pointer are random.

Note with enough training data, the networks are able to correctly predict those test examples even though all the combinations of (0,1) are held out during training.

Does high test accuracy correspond to learning reasoning?



We train neural networks on PVR tasks with complexity m=2, and held out various number of permutations of (0,1,2) in the value window.

We test on inputs where the value window contains the value (0,1,2), while all other digits and the pointer are random.

We observe similar results with m=2, even though sometimes learning could be a bit unstable. But when training succeeds, the network could generalize well even with complete held-out.

Conclusion

Can neural networks learn to reason?

- Generalization in machine learning is often thought of in terms of IID data
- But there are a spectrum of possible sub-methods, from memorizing (rare examples), k-NNs, IID generalization, to out-of-domain generalization and reasoning
- There is an open question on how much neural networks are prone to similarity/co-occurrence methods vs abstraction / reasoning based methods

- We introduced the (Visual) Index Value Retrieval Tasks to study this
  - Out-of-domain visual task
  - Family of logical reasoning tasks of increasing complexity
- In both settings, we observe that neural networks fail at tasks that require greater abstraction, suggesting reliance on simpler similarity methods in learning
- We are investigating this further to pinpoint whether different reasoning elements are learned.