Object-Flow Machine Learning: Active learning framework utilizing protocols information

Yusuke Ozaki[1,2,3,*] **Kazunari Kaizu**[1,2,4] **Koichi Takahashi**[1,2]

[1]RIKEN Center for Biosystems Dynamics Research
[2]Advanced General Intelligence for Science Program (AGIS)
[3]Kwansei Gakuin University, Department of Engineering, Program of Computer Science
[4]Cell Modeling and Simulation Group, The Exploratory Research Center on Life and Living Systems, National Institutes of Natural Sciences

Abstract Automated experiments increasingly relies on AI, yet most active-learning pipelines ignore the rich procedural structure encoded by laboratory protocols. Existing sequence-based models capture temporal order but not branching dependencies, limiting deductive reasoning in complex workflows. We introduce Object-Flow Machine Learning (OFML), a protocol-aware framework that assigns modular ML models to individual processes and composes them according to a given protocol, executing experiments on a computer as function compositions. To quantify decision uncertainty, OFML employs ensemble-based predictive variance, a simple and scalable alternative to Bayesian neural networks that yields well-calibrated uncertainty and robust behavior under distribution shift. We demonstrate the framework on color-mixing protocols, using protocol-level uncertainty to select informative experiments and to illustrate gains in sample efficiency over random acquisition. OFML suggests a practical path to model-based active learning that respects experimental structure while remaining easy to implement and parallelize.

1 Introduction

Recent advances in robotics and AI have increasingly driven the adoption of automation technologies in scientific research [1]. "Automation in scientific research" differs in key respects from prior work on "automation in applied domains," and thus requires novel methods tailored to these unique characteristics. When combined with robotics, AI-driven research automation can be cast as a model-based active learning paradigm (Fig. 1a). In this paradigm, AI systems perform induction, hypothesis generation, and deduction, while robots carry out the physical experiments and feed data back to the AI.

In experimental science, each experiment can be described as a mapping

$$\text{Input: (Conditions, Protocol)} \longrightarrow \text{Output: (Results)}. \tag{1}$$

These inputs and outputs are structured by the constraints of the physical world. In particular, the experimental protocol itself encodes rich procedural information, yet existing methods have not fully leveraged this structure.

Classical active learning frameworks (e.g., pool-based uncertainty sampling [2, 3]) focus on data points in a static feature space and cannot incorporate dynamic experimental workflows. Sequence-based approaches using LSTMs or Transformers capture temporal order but ignore branching dependencies, leading to hallucinations when applied to real protocols and degraded performance as complexity increases [4, 5].

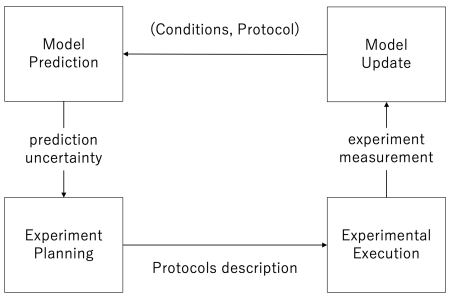
To address these gaps, we propose Object-Flow Machine Learning (OFML), a natural active learning framework for experimental data with explicit protocol structure. OFML assigns a modular

ML model to each experimental unit process (hereafter, "process") and composes them according to the given protocol, faithfully reproducing the experiment in computer (Fig. 1b). By instantiating multiple such model sets and measuring the variance of their predictions, OFML quantifies uncertainty at the protocol level. Integrating protocol-aware modeling and ensemble-based uncertainty estimation into a seamless loop, OFML selects the most informative next experiments and realizes model-based active learning.

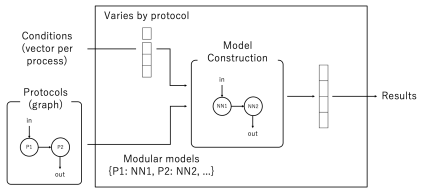
2 Background

Experiments are described in triplet format as shown in Eq. 1. To realize model-based active learning for such experiments, we need three components.

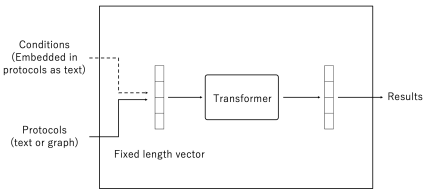
Protocol representation. Documentation of experimental protocols is essential for ensuring reproducibility in biological research. Traditionally, protocols are described in a linear, step-by-step manner, as typically seen in scientific papers. While intuitive, this format has limitations in representing complex workflows involving conditional steps, loops, or parallel operations. As an alternative, graph-based representations offer a general framework for describing complex procedures. In this approach, each experimental process is modeled as a node, while edges represent the dependencies or sequence between processes. Importantly, each node explicitly defines its inputs and outputs, making dependencies between processes clear and analyzable. This method not only enables parallelization but also enhances modularity. Protocols can be constructed from reusable components, making them easier to maintain, modify, and integrate into automated systems. As part of this approach, we have developed our own graph-based protocol description language, called OFPlang. It is an object-flow language that extends the dataflow paradigm to include not only data but also physical entities such as labware. OFPlang allows the unified description of both experimental manipulations and computational data processing, and serves as an input specification for automated experiments by providing a precise and executable workflow structure.



(a) Model-based active learning



(b) Proposed method



(c) Transformer-based method

Figure 1: (a) Model-based active learning framework. (b) Proposed method: assemble modular models per unit process, and compute results accordingly. (c) Transformer-based method: it uses conditions and protocol information as fixed length vector.

Conventional Approaches to Research Automation. Recent lines of work automate experiments from two complementary directions. First, LLM–driven agents generate stepwise procedures (Fig. 1c), write control code, and interface with lab hardware to execute closed-loop experiments (*e.g.*, Coscientist). [6] Second, autonomous laboratories integrate robotics with active/bayesian learning to plan, run, and analyze experiments at scale (*e.g.*, A-Lab; mobile robotic chemist). [1, 7] Sequence-based planning, such as neural-network–guided retrosynthesis, produces action sequences but typically lacks explicit representations of branching, resource routing, and measurement dependencies found in executable protocols. [8] These advances motivate a protocol-aware approach that preserves procedural structure—beyond flat token sequences—when learning and selecting experiments.

Ensemble-based uncertainty estimation. A practical committee-based approach is *deep ensembles* [10]: train multiple independently initialized networks, aggregate their predictions, and use disagreement (e.g., variance of predictive means or vote entropy) to score candidates—functioning as a Query-by-Committee strategy that yields well-calibrated uncertainty and strong active-learning performance. [11, 12]

3 Object-Flow Machine Learning

OFML is an active-learning framework that leverages protocol information. It operates according to the following steps.

Model preparation. We prepare machine-learning models for each experimental *process*. Let N be the number of distinct processes. For each type of process $i = 1, \ldots, N$, we construct a model h_i and collect them as $H = \{h_1, \ldots, h_N\}$. To enable deep-ensemble uncertainty estimation for our acquisition strategy, we independently initialize E copies of H, denoted $H_1, \ldots, H_E$. This setup allows the experiment to be represented and tracked on a computer as a composition of functions. Models are broadly categorized into three types according to process characteristics: {*encoders*, *process modules*, *measurement modules*}.

Model construction. Given a protocol described in a YAML file and conditions in a JSON file, we compute experimental outcomes in accordance with OFPLang's execution semantics. Processes are evaluated as soon as all their inputs become available, so the procedure accommodates protocols of varying length. In our implementation, the quantities on which the model loss is computed are mapped directly to the outputs of those processes designated as `Output` in the protocol's YAML file. As the overall computation is a composition of learned functions, stability/error bounds should be stated in terms of module-wise errors and sensitivity (e.g., Lipschitz constants [13]) of the composed map (see remarks below).

Active learning with predictive uncertainty. For each process i, we evaluate the corresponding models from each ensemble H_e independently ($e = 1, \ldots, E$). We then compute the variance across ensemble outputs as a measure of predictive uncertainty for acquisition. This implements a query-by-committee strategy; under specific conditions (e.g., positive bounded information gain for the committee), the prediction error can decrease exponentially with the number of queries. [14, 15] We also discuss the diminishing-returns behavior of information-gain–based policies (see remarks below).

4 Experiments

We used color-mixing protocols as a test case for our predictive experiments because they are simple, reproducible, and readily extensible to other applications. We defined three types of experimental processes—`serve-plate` (provides a 96-well plate to the liquid-handling system), `dispense-liquid` (dispenses specified ink to designated wells), and `measure-absorbance` (measures absorbance in each well).The protocols used in these experiments are illustrated in Fig. 2.

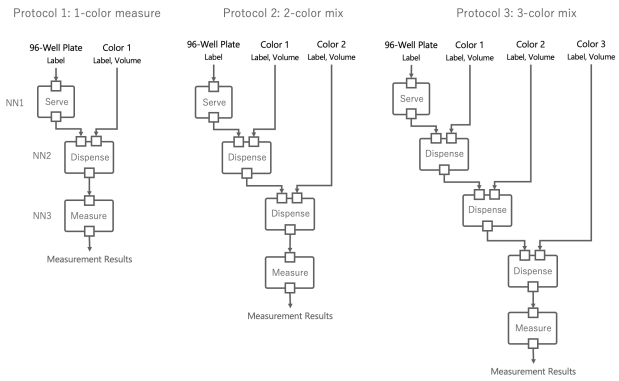
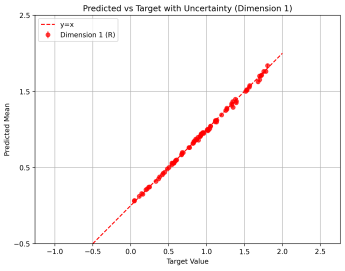


Figure 2: Protocols of one-, two-, three-color mixing

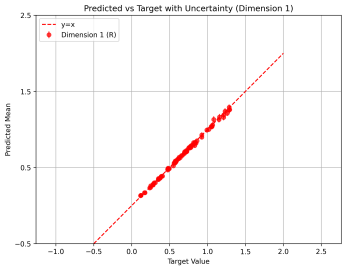
ML Experiments. To assess predictive accuracy and *cross-protocol generalization*, we pre-trained on Protocols 1 and 2 and evaluated on the *untrained*, more complex

Protocol 3 (Fig. 3a–3c). This setting examines whether models trained on simpler protocols can nonetheless produce usable predictions on a protocol not seen during training. Our models achieved high accuracy on the trained one- and two-color protocols and still delivered reasonable predictions on the unseen three-color protocol (Fig.3c, Appendix 2), but—because no comparable studies reporting cross-protocol generalization were found—we could not perform a direct benchmark.[16?–18]

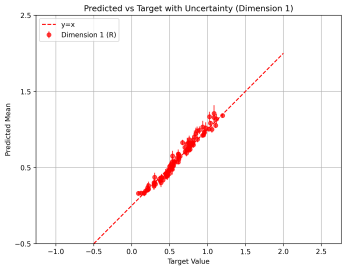
Next, we conducted an active-learning study using our ensemble-based acquisition policy. After *20 active-learning epochs*, the test loss decreased from **10.47** to **2.23** (Fig. 3e). These results emphasize two points: (i) the model yields actionable predictions even on an *untrained* protocol, and (ii) active learning systematically reduces error by targeting informative acquisitions, independent of head-to-head numeric comparisons with alternative sampling schemes.



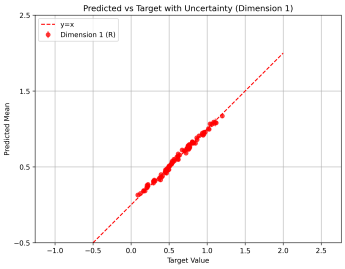
(a) 1-color mixing



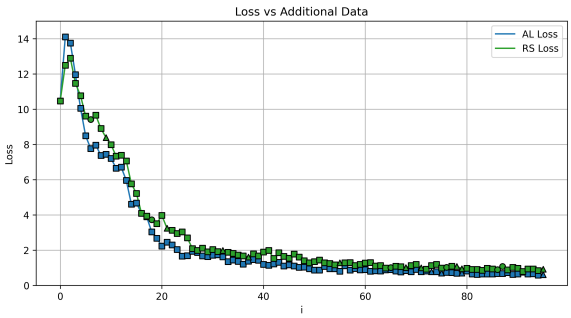
(b) 2-color mixing



(c) 3-color mixing



(d) 3-color mixing (after active learning)



(e) Error vs. data size

Figure 3: Ensemble predictions vs. targets for (a) 1-color measuring. (b) 2-color mixing. (c) 3-color mixing (untrained). (d) 3-color mixing after 20 data additional learning (e) Comparison of error reduction: ensemble vs. random sampling.

5 Conclusion and Future Works

We presented Object-Flow Machine Learning (OFML), a protocol-aware framework that composes modular models per experimental process and quantifies protocol-level uncertainty via ensembles. By aligning model structure with executable protocols, OFML offers a practical path to model-based active learning in automated experiments. Our study suggests that protocol-aware composition and ensemble variance can guide informative experiment selection while remaining simple to implement and parallelize. Current limitations include: (i) extending OFML to datasets with different modalities, (e.g., images, molecules), (ii) generalizing the theoretical guarantees to broader classes of experimental protocols, and (iii) adopting more practical acquisition strategies. Future work will explore more practical acquisition strategies, integrate with physical-robot platforms more seamlessly, and evaluate OFML on complex, large-scale protocol libraries.

References

[1] Nathan J. Szymanski, Bernardus Rendy, Yuxing Fei, Rishi E. Kumar, Amil Merchant, Ekin Dogus Cubuk, Anubhav Jain, Kristin Persson, Gerbrand Ceder, et al. An autonomous laboratory for the accelerated synthesis of novel materials. *Nature*, 624:86–91, 2023. doi: 10.1038/s41586-023-06734-w.

[2] Genki N. Kanda, Taku Tsuzuki, Motoki Terada, Noriko Sakai, Naohiro Motozawa, Tomohiro Masuda, Mitsuhiro Nishida, Chihaya T. Watanabe, Tatsuki Higashi, Shuhei A. Horiguchi, Taku Kudo, Motohisa Kamei, Genshiro A. Sunagawa, Kenji Matsukuma, Takeshi Sakurada, Yosuke Ozawa, Masayo Takahashi, Koichi Takahashi, and Tohru Natsume. Robotic search for optimal cell culture in regenerative medicine. *eLife*, 11:e77007, 2022. doi: 10.7554/eLife.77007. URL https://doi.org/10.7554/eLife.77007.

[3] Burr Settles. Active learning literature survey. Technical Report Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009. URL https://burrsettles.com/pub/settles.activelearning.pdf.

[4] Alain C. Vaucher, Philippe Schwaller, Joppe Geluykens, Vishnu H. Nair, Anna Iuliano, and Teodoro Laino. Inferring experimental procedures from text-based representations of chemical reactions. *Nature Communications*, 12(1):2573, 2021. doi: 10.1038/s41467-021-22951-1. URL https://www.nature.com/articles/s41467-021-22951-1.

[5] Kristina Yordanova, Teodor Stoev, Henrike Rebl, Olga Hahn, and Kirsten Peters. Text2rlab: No-code methodology for robotic programming and interaction in laboratory tasks. In *Companion Proceedings of the 30th International Conference on Intelligent User Interfaces (IUI '25)*, pages 96–100, New York, NY, USA, March 2025. Association for Computing Machinery. doi: 10.1145/3708557.3716350. URL https://dl.acm.org/doi/10.1145/3708557.3716350.

[6] Daniil A. Boiko, Robert MacKnight, Ben Kline, Gabe Gomes, et al. Autonomous chemical research with large language models. *Nature*, 624:570–578, 2023. doi: 10.1038/s41586-023-06792-0.

[7] Benjamin Burger, Phillip M. Maffettone, Vladimir V. Gusev, Catherine M. Aitchison, Yang Bai, Xiaoyan Wang, Xiaobo Li, Ben M. Alston, Buyi Li, Rob Clowes, Nicola Rankin, Brandon Harris, Reiner Sebastian Sprick, and Andrew I. Cooper. A mobile robotic chemist. *Nature*, 583 (7815):237–241, 2020. doi: 10.1038/s41586-020-2442-2.

[8] Marwin H. S. Segler, Mike Preuss, and Mark P. Waller. Planning chemical syntheses with deep neural networks and symbolic AI. *Nature*, 555(7698):604–610, 2018. doi: 10.1038/nature25978.

[9] Alexey A. Melnikov, Hendrik Poulsen Nautrup, Mario Krenn, Vedran Dunjko, Markus Tiersch, Anton Zeilinger, and Hans J. Briegel. Active learning machine learns to create new quantum experiments. *Proceedings of the National Academy of Sciences*, 115(6):1221–1226, 2018. doi: 10.1073/pnas.1714936115.

[10] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. URL https://arxiv.org/abs/1612.01474.

[11] H. Sebastian Seung, Manfred Opper, and Haim Sompolinsky. Query by committee. In *Proceedings of the 5th Annual Workshop on Computational Learning Theory (COLT)*, pages 287–294. ACM, 1992. doi: 10.1145/130385.130417.

[12] Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Brij B. Gupta, Xiaojiang Chen, and Xin Wang. A survey of deep active learning. *ACM Computing Surveys*, 54(9): 180:1–180:40, 2021. doi: 10.1145/3472291.

[13] Juha Heinonen. *Lectures on Analysis on Metric Spaces*. Universitext. Springer, 2001. doi: 10.1007/978-1-4613-0131-8.

[14] Carlos Guestrin, Andreas Krause, and Ajit Singh. Near-optimal sensor placements in gaussian processes. In *Proceedings of the 22nd International Conference on Machine Learning (ICML)*, pages 265–272, 2005. doi: 10.1145/1102351.1102385.

[15] Andreas Krause, Ajit Singh, and Carlos Guestrin. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research*, 9:235–284, 2008. URL https://jmlr.org/papers/v9/krause08a.html.

[16] Simon P. Stier, Christoph Kreisbeck, et al. Materials acceleration platforms (maps): Accelerating materials research and development to meet urgent societal challenges. *Advanced Materials*, 36(7):2407791, 2024. doi: 10.1002/adma.202407791. URL https://doi.org/10.1002/adma.202407791.

[17] Martha M. Flores-Leonar, Martin Mejía-Mendoza, Andrés Aguilar-Granda, et al. Materials acceleration platforms: On the way to autonomous experimentation. *Current Opinion in Green and Sustainable Chemistry*, 25:100370, 2020. doi: 10.1016/j.cogsc.2020.100370. URL https://doi.org/10.1016/j.cogsc.2020.100370.

[18] Adam Tobias and Adam Wahab. Autonomous 'self-driving' laboratories: A review of technology and policy implications. *Royal Society Open Science*, 12(7):250646, 2025. doi: 10.1098/rsos.250646. URL https://doi.org/10.1098/rsos.250646.

[19] Peter L. Bartlett, Dylan J. Foster, and Matus J. Telgarsky. Spectrally-normalized margin bounds for neural networks. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[20] Aladin Virmaux and Kevin Scaman. Lipschitz regularity of deep neural networks: analysis and efficient estimation. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

[21] Moustapha Cisse, Piotr Bojanowski, Edouard Grave, Yann Dauphin, and Nicolas Usunier. Parseval networks: Improving robustness to adversarial examples. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 854–863. PMLR, 2017.

[22] Juha Heinonen. *Lectures on Lipschitz Analysis*. 2001.

[23] Yoav Freund, H. Sebastian Seung, Eli Shamir, and Naftali Tishby. Selective sampling using the query by committee algorithm. *Machine Learning*, 28(2-3):133–168, 1997.

[24] H. Sebastian Seung, Manfred Opper, and Haim Sompolinsky. Query by committee. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory (COLT)*, pages 287–294. ACM, 1992.

[25] Andreas Krause, Ajit Singh, and Carlos Guestrin. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research*, 9:235–284, 2008.

[26] Andreas Krause, Carlos Guestrin, and Ajit Singh. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 265–272, 2005.

[27] Daniel Golovin and Andreas Krause. Adaptive submodularity: Theory and applications in active learning and stochastic optimization. *Journal of Artificial Intelligence Research*, 42: 427–486, 2011.

[28] Jaehoon Lee, Yasaman Bahri, Roman Novak, Samuel Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein. Deep neural networks as gaussian processes. In *International Conference on Learning Representations*, 2018.

[29] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

A Proofs of Theoretical Results

A.1 Lipschitz Stability of Composed Modules

Theorem A.1 (Composition of Lipschitz Maps). *Let $f_1, \dots, f_K$ be functions on a metric space with Lipschitz constants $L_1, \dots, L_K$. Then $f = f_K \circ \cdots \circ f_1$ is Lipschitz with*

$$\mathrm{Lip}(f) \;\le\; \prod_{k=1}^{K} L_k.$$

Sketch. For $K = 1$ it is trivial. Assume true for $K-1$ and write $g = f_{K-1} \circ \cdots \circ f_1$. Then for any x, y, $d(f(x), f(y)) = d(f_K(g(x)), f_K(g(y))) \le L_K\, d(g(x), g(y)) \le L_K \big(\prod_{k=1}^{K-1} L_k\big)\, d(x, y)$. ∎

Implications for DNNs and our pipeline.. For feedforward networks with 1-Lipschitz nonlinearities (e.g. ReLU), the global Lipschitz constant is upper-bounded by the product of spectral norms of linear layers; this quantity also governs margin-based generalization bounds in deep nets. Therefore, for the composed modules NN3 ∘ NN2 ∘ NN1 used in our experiment, the end-to-end sensitivity is controlled by the product of the module-wise constants, which can be reduced via spectral/Parseval constraints. See Bartlett et al. [19] for spectrally-normalized bounds, and Virmaux and Scaman [20], Cisse et al. [21] for analysis and practice of Lipschitz control in DNNs.[1]

A.2 Error Decay in Query-by-Committee

Theorem A.2 (Exponential Error Reduction under QBC). *Let $\mathcal{V}_t$ be the version space after t queries. If each query reduces $\mathcal{V}_t$ by a constant fraction $\rho < 1$ under majority disagreement sampling, then the generalization error satisfies*

$$\epsilon_t \;\le\; \rho^{\,t}\, \epsilon_0.$$

Sketch. Classical QBC analysis shows that each informative query eliminates a constant fraction of inconsistent hypotheses, yielding exponential shrinkage of the error; see Freund et al. [23], Seung et al. [24]. ∎

Applicability to deep ensembles.. The strict assumptions (realizability/explicit version space) need not hold for modern DNNs. In practice, committee disagreement using deep ensembles often mirrors the same qualitative behavior—faster error decay than random sampling— as we observed on our task. We thus view QBC-style selection as a well-supported heuristic for deep models (practically effective though not fully covered by the classical proof).

[1] Standard background on Lipschitz composition appears in, e.g., Heinonen [22].

Table 1: Ink–water mixing and dilutions.

Ink	Dilution (×)
FRAME (Flame Red)	100
SUNNY (Sunny Yellow)	200
AQUA (Aqua Blue)	200

A.3 Diminishing Information Gain

Theorem A.3 (Submodularity of Mutual Information). *For a Gaussian process model, the mutual information objective used for selecting a set A of points is (monotone) submodular. Hence, for $A \subseteq B$ and any candidate $x \notin B$,*

$$I(y_x;\, f \mid A) \;\geq\; I(y_x;\, f \mid B),$$

i.e. marginal gains diminish, and the greedy policy enjoys a $(1 - 1/e)$ approximation guarantee.

Sketch. This is a standard consequence of the submodularity of mutual information under GPs; see Krause et al. [25, 26]. See also the general framework of *adaptive submodularity* [27]. ■

From GPs to deep nets.. Infinite-width limits connect DNNs to Gaussian processes (NNGP) and to the neural tangent kernel (NTK). These links justify GP/MI-based selection as a principled approximation for deep models, explaining the consistent diminishing returns we observe empirically when adding data.[28, 29]

Takeaway for our composed protocol.. (i) The end-to-end Lipschitz constant of the composed modules is bounded by the product of per-module constants (directly useful for our MLP chain). (ii) QBC gives exponential error decay under idealized conditions and remains effective with deep ensembles in practice. (iii) MI-based selection exhibits diminishing returns; greedy acquisition is theoretically near-optimal under GP assumptions, and NNGP/NTK connections motivate its use with DNNs.

B Experimental Details

B.1 Hardware and Reagents

- **Liquid handler**: Tecan *Fluent*.
- **Plate reader**:*Infinite 200 Pro*®.
- **Dyes**: Platinum *Mixable Ink* series (Flame Red, Sunny Yellow, Aqua Blue).

B.2 Protocol Outline

1) `serve_plate`: provides a 96-well plate to the liquid-handling system.

2) `dispense_liquid`: dispenses specified ink to designated wells

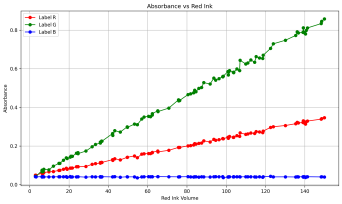
3) `measure_absorbance`: measures absorbance in each well

B.3 Example YAML Snippet

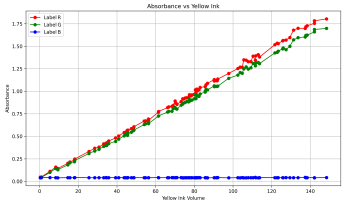
C ML Experiments' Details for Actual Experimental Data

C.1 Neural-module I/O Summary

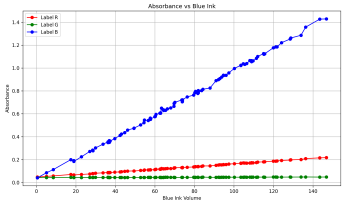
We use the same three neural modules as in the Sigmoid toy experiments. For architectural details (hidden sizes, activation, dropout), see Table 3.



(a) Frame Red Absorbance



(b) Sunny Yellow Absorbance



(c) Aqua Blue Absorbance

Figure 4: The absorbance of (a) Frame Red (b) Sunny Yellow (c) Aqua Blue

```yaml
input:
  - id: volume1
    type: Array[Float]
  - id: channel1
    type: Array[Float]
    default:
      value: 0
      type: Array[Float]
  - id: plate1
    type: Array[Float]
    default:
      value: 1
      type: Array[Float]

output:
  - id: mixed_output
    type: Array[Float]

operations:
  - id: serve_plate1
    type: NN_embedding_96wellplate
  - id: dispense_liquid1
    type: NN_Dispense
  - id: read_absorbance1
    type: NN_Read_Absorbance

connections:
  - input:
      - input
      - plate1
    output:
      - serve_plate1
      - in1
  - input:
      - input
      - volume1
    output:
      - dispense_liquid1
      - volume
  - input:
      - input
      - channel1
    output:
      - dispense_liquid1
      - channel
  - input:
      - serve_plate1
      - out1
    output:
      - dispense_liquid1
      - in1
```

(a) YAML file


```json
[
    {
        "protocol": {
            "file": "./sample_nn_1.yaml"
        },
        "input": {
            "volume1": {
                "value": 56.36686705184336,
                "type": "Array[Float]"
            },
            "channel1": {
                "value": [
                    0,
                    1,
                    0
                ],
                "type": "Array[Float]"
            }
        },
        "output": {
            "mixed_output": {
                "value": [
                    0.6779,
                    0.6412,
                    0.0405
                ],
                "type": "Array[Float]"
            }
        }
    },
    {
        "protocol": {
            "file": "./sample_nn_1.yaml"
        },
        "input": {
            "volume1": {
                "value": 113.75203607592022,
                "type": "Array[Float]"
            },
            "channel1": {
                "value": [
                    0,
                    1,
                    0
                ],
                "type": "Array[Float]"
            }
        },
```


(b) JSON file

Figure 5: (a) YAML file of 1-color (green) measuring protocol. (b) JSON file of 1-color (green) measuring protocol.

Table 2: Sample counts by protocol and split (Actual experiment data).

Protocol	**Train**	**Test**	**AL candidates**
Protocol 1	84×3	—	12×3
Protocol 2	84×9	—	12×9
Protocol 3 (untrained)	—	96×6	—
Total	1008	576	720

Table 3: Summary of three MLP blocks (shared across Actual/Sigmoid).

Block	Input dim	Hidden units	Dropout	Output dim	Shapes
NN1	1	120	0.2	105 (bridge)	$1 \to 120 \to 105$
NN2	$1 + 3 + 105 = 109$	$140 \times 4 = 560$	0.2	105 (bridge)	$109 \to 560 \to 105$
NN3	105	170	0.2	3	$105 \to 170 \to 3$

C.2 Train and Test Samples

We consider three protocols on actual experiment data. Protocols 1–2 are split into **84 for training** and **12 for the active learning (AL) candidate pool** per protocol, while Protocol 3 is fully held out for **test**.

Protocols (96 samples each)..

$$\text{Protocol 1} : \{\, R,\ Y,\ B \,\}, \tag{2}$$

$$\text{Protocol 2} : \{\, R{*}R,\ R{*}Y,\ R{*}B,\ Y{*}R,\ Y{*}Y,\ Y{*}B,\ B{*}R,\ B{*}Y,\ B{*}B \,\}, \tag{3}$$

$$\text{Protocol 3} : \{\, R{*}Y{*}B,\ R{*}B{*}Y,\ Y{*}R{*}B,\ Y{*}B{*}R,\ B{*}R{*}Y,\ B{*}Y{*}R \,\}. \tag{4}$$

Here, "$*$" denotes sequence concatenation (multi-step dispensing), not multiplication.

C.3 Training Details

- Optimizer: Adam, learning rate 10^{-3}.
- Ensemble size $E = 5$; dropout $p = 0.2$.
- Loss: mean-squared error on y.
- Training epochs: 5000.
- Active learning epochs: 96.

D ML Experiments' Details for Actual Experimental Data

D.1 Neural-module I/O Summary

We summarize the inputs and outputs of the three neural modules used in our ML experiment on the Sigmoid task. Architectural details are in Table 3.

D.2 Train and Test Samples

Each protocol has **96 samples**. Protocols 1–2 are split into **84 (train)** and **12 (AL candidates)** per protocol, while Protocol 3 is held out for **test**.

Table 4: Sample counts by protocol and split (Sigmoid toy dataset).

Protocol	Train	Test	AL candidates
Protocol 1	96×3	—	—
Protocol 2	96×9	—	—
Protocol 3 (untrained)	—	96×6	—
Total	1152	576	576

Protocols (96 samples each)..

$$\text{Protocol 1} : \{\, R,\ G,\ B \,\}, \tag{5}$$

$$\text{Protocol 2} : \{\, R{*}R,\ R{*}G,\ R{*}B,\ G{*}R,\ G{*}G,\ G{*}B,\ B{*}R,\ B{*}G,\ B{*}B \,\}, \tag{6}$$

$$\text{Protocol 3} : \{\, R{*}G{*}B,\ R{*}B{*}G,\ G{*}R{*}B,\ G{*}B{*}R,\ B{*}R{*}G,\ B{*}G{*}R \,\}. \tag{7}$$

D.3 Sigmoid Toy Dataset

This appendix specifies the synthetic "Sigmoid" dataset. Each example is defined by a *color code* $c_{1:n} \in \mathcal{C}^n$ and non-negative volumes v_i. The color code chooses, at each step i, which one-hot vector $\mathbf{e}_{c_i} \in \{(1,0,0)^\top, (0,1,0)^\top, (0,0,1)^\top\}$ contributes to the RGB totals. We then apply a fixed linear mixing and an elementwise sigmoid to obtain the final 3-dimensional output.

Notation..

- Color set $\mathcal{C} = \{r, g, b\}$ with one-hot vectors $\mathbf{e}_r = (1,0,0)^\top$, $\mathbf{e}_g = (0,1,0)^\top$, $\mathbf{e}_b = (0,0,1)^\top$.
- Color code $c_{1:n} = (c_1, \ldots, c_n) \in \mathcal{C}^n$.
- Volumes $v_i \geq 0$ sampled independently under a cap cap > 0.
- RGB totals $(R, G, B)^\top = \sum_{i=1}^{n} v_i\, \mathbf{e}_{c_i}$.
- Centering $\tilde{\mathbf{v}} = (R - 5,\ G - 5,\ B - 5)^\top$.
- Coefficient matrix $\mathbf{A} \in \mathbb{R}^{3\times 3}$ (columns for $R/G/B$).
- Sigmoid (elementwise): $\sigma(x) = \dfrac{1}{1 + \exp(-x)}$.

$$\mathbf{A} = \begin{bmatrix} 1.0 & 0.8 & 0.1 \\ 0.5 & 0.8 & 0.1 \\ 0.1 & 0.2 & 1.0 \end{bmatrix}. \tag{8}$$

$$v_i \sim \mathcal{U}(0, \text{cap}), \qquad i = 1, \ldots, n. \tag{9}$$

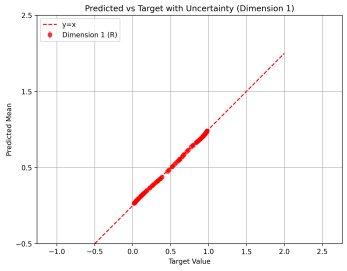
$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \sum_{i=1}^{n} v_i\, \mathbf{e}_{c_i}, \qquad \tilde{\mathbf{v}} = (R - 5,\ G - 5,\ B - 5)^\top. \tag{10}$$

$$\mathbf{z} = \mathbf{A}\,\tilde{\mathbf{v}}, \qquad \mathbf{y} = \sigma(\mathbf{z}) \in (0,1)^3. \tag{11}$$

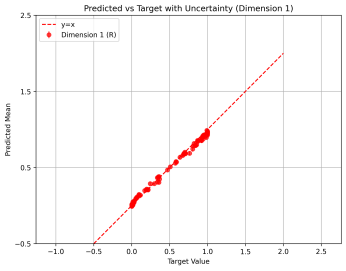
Remark.. We typically use $n \in \{1, 2, 3\}$. The centering by 5 controls saturation before the sigmoid; $\mathbf{A}$ linearly mixes the centered RGB totals, and σ maps them to bounded outputs.

D.4 Training Details

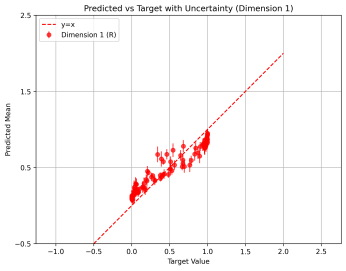
- Optimizer: Adam, learning rate 10^{-3}.
- Ensemble size $E = 5$; dropout $p = 0.2$.
- Loss: mean-squared error on y.
- Training epochs: 5000.
- Active learning epochs: 96.



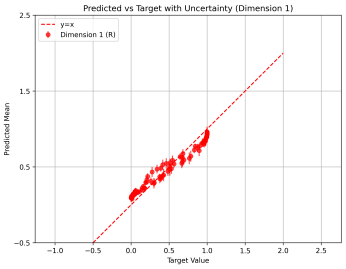
(a) 1-color mixing



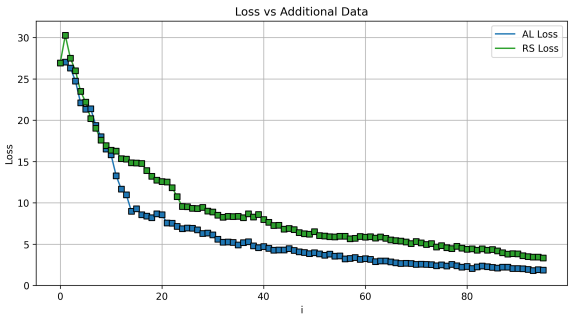
(b) 2-color mixing



(c) 3-color mixing



(d) 3-color mixing (after active learning)



(e) Error vs. data size

Figure 6: (Sigmoid) Ensemble predictions vs. targets: (a) 1-color; (b) 2-color; (c) 3-color (untrained); (d) after 20 additional AL samples; (e) error comparison.