The intersection of Optical Chemical Structure Recognition (OCSR) and object detection

Martijn Oldenhof

25th of October 2022– AIDD school - Leuven

The intersection of Optical Chemical Structure Recognition (OCSR) and object detection

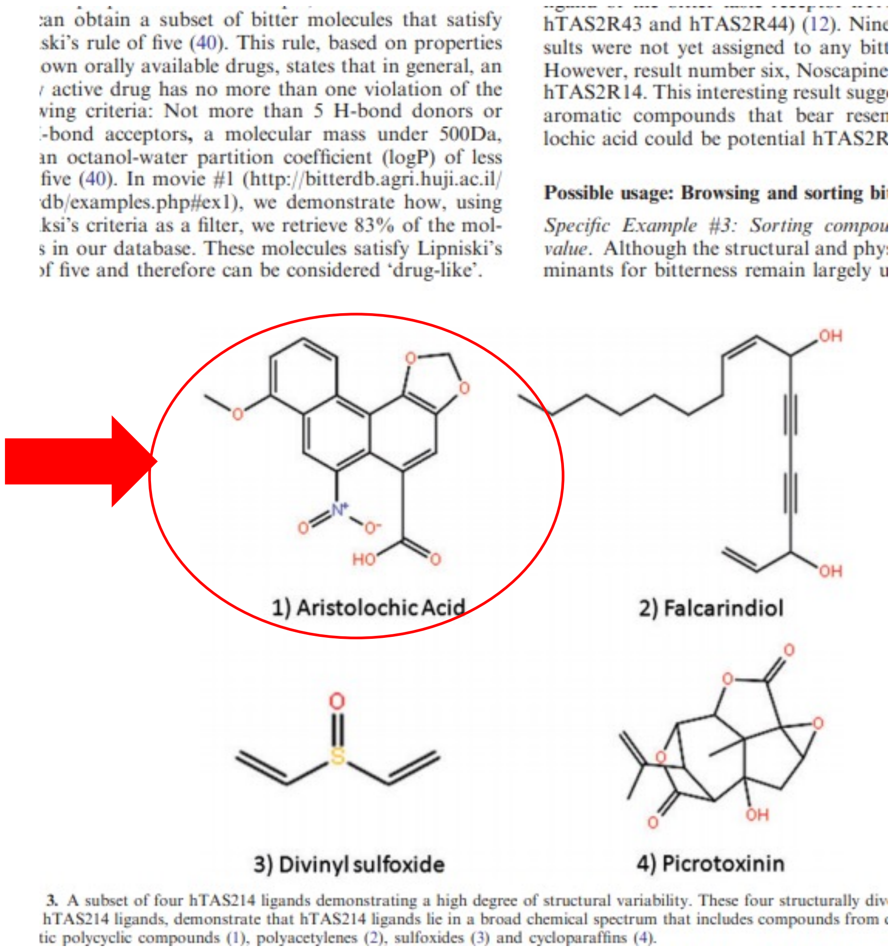
- Part I:
  - Introduction and Motivation: ChemGrapher
  - (Weakly Supervised) Object Detection
  - Updating Object Detection Models with Probabilistic Programming
- Part II:
  - Experiments in W&B

CHEMGRAPHER: OPTICAL GRAPH RECOGNITION OF CHEMICAL COMPOUNDS BY DEEP LEARNING

Oldenhof et al., 2020, Journal of Chemical Information and Modelling.

Data mining chemical compounds in literature

- Thousands of scientific **publications** describe new chemical compounds.
- Details mostly only described **using an image** which is less suited to query accurately.
- Rich **source of data** but largely **unexploited**.
- Need for a **tool** to convert **image to graph**.


an obtain a subset of bitter molecules that satisfy ski's rule of five (40). This rule, based on properties own orally available drugs, states that in general, an active drug has no more than one violation of the ving criteria: Not more than 5 H-bond donors or -bond acceptors, a molecular mass under 500Da, in octanol-water partition coefficient (logP) of less five (40). In movie #1 (http://bitterdb.agri.huji.ac.il/ db/examples.php#ex1), we demonstrate how, using ksi's criteria as a filter, we retrieve 83% of the mol- s in our database. These molecules satisfy Lipniski's of five and therefore can be considered 'drug-like'.

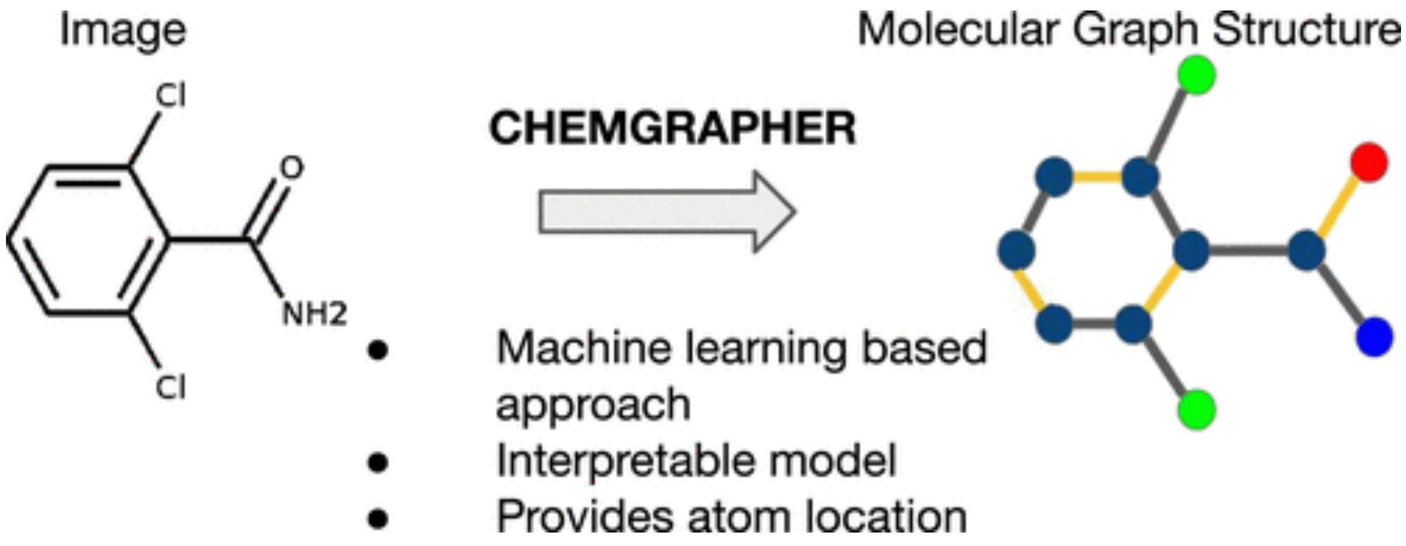
hTAS2R43 and hTAS2R44) (12). Nine sults were not yet assigned to any bitt However, result number six, Noscapine hTAS2R14. This interesting result sugg aromatic compounds that bear resen lochic acid could be potential hTAS2R

Possible usage: Browsing and sorting bi

Specific Example #3: Sorting compou value. Although the structural and phy minants for bitterness remain largely u

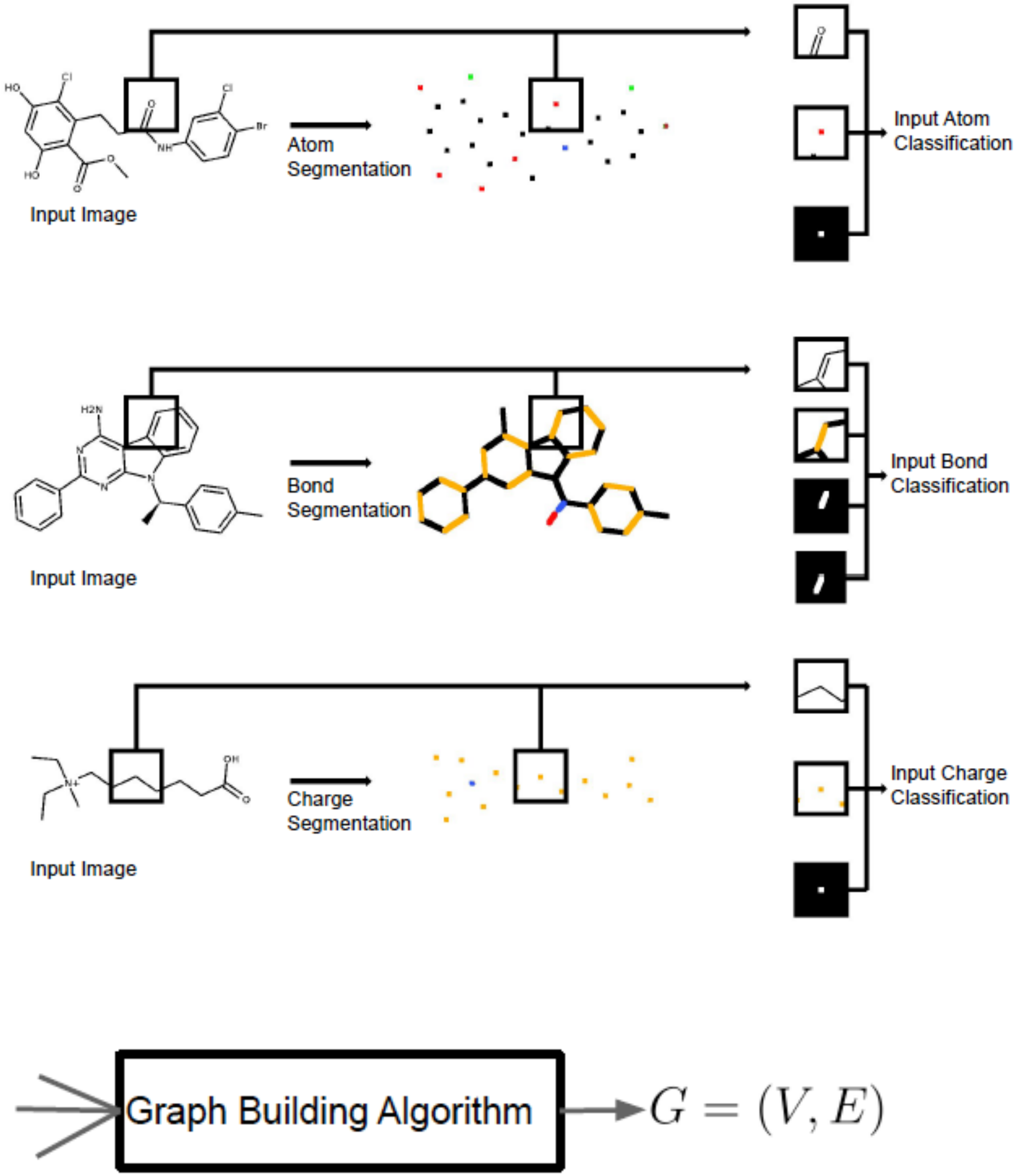
3. A subset of four hTAS214 ligands demonstrating a high degree of structural variability. These four structurally div hTAS214 ligands, demonstrate that hTAS214 ligands lie in a broad chemical spectrum that includes compounds from tic polycyclic compounds (1), polyacetylenes (2), sulfoxides (3) and cycloparaffins (4).

ChemGrapher



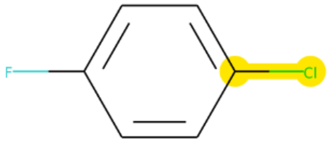
ChemGrapher

- Proposed Method:
  - Segmentation in segments of atoms, bonds and charges
  - Classification of segments taken into account segment, original image and area of interest.
- Resulting predictions to build Graph:



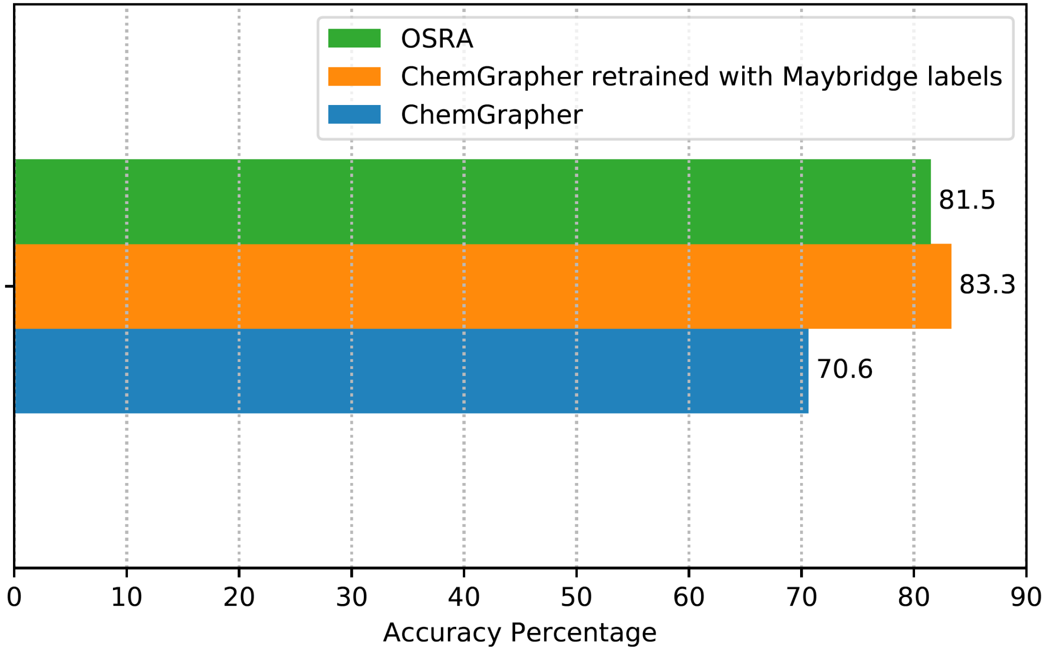
Training labels for ChemGrapher

- Chemical Compounds can be represented with a string representation: SMILES
- RDkit : Open-Source Cheminformatics Software
- RDkit can take as input SMILES and output an image of chemical compound
- Fork of RDkit: https://github.com/biolearning-stadius/rdkit
- To create pixel-wise labels for RDkit generated images.

SMILES	Generated Image	Generated Labels
c1cc(F)ccc1Cl		idx,mol,bond,at1,ch1,at2,ch2,x1,y1,x2,y2 0,0,2.0,C,0,C,0,590,685,376,685 1,0,1.0,C,0,C,0,590,685,697,500 2,0,1.0,C,0,C,0,376,685,269,500 3,0,1.0,C,0,F,0,269,500,55,500 4,0,2.0,C,0,C,0,269,500,376,314 5,0,1.0,C,0,C,0,376,314,590,314 6,0,2.0,C,0,C,0,590,314,697,500 7,0,1.0,C,0,Cl,0,697,500,911,500 8,0,nobond,F,0,F,0,55,500,55,500 9,0,nobond,Cl,0,Cl,0,911,500,911,500

ChemGrapher vs OSRA[2]

- The dataset (UoB) of 5740 images and molfiles of chemical structures developed by the University of Birmingham, United Kingdom, and published alongside MolRec [1].
- ChemGrapher vanilla performance trained on RDkit generated images only
- ChemGrapher retrained including 40 images from UoB dataset labelled manually.



[1] M. Sadawi, N.; Sexton, A.; Sorge, V. Chemical Structure Recognition: A Rule Based Approach.Proc. SPIE2012,8297, 32–.
[2] Filippov, I. V.; Nicklaus, M. C. Optical Structure Recognition Software To RecoverChemical Information: OSRA, An Open Source Solution.J. Chem. Inf. Model.2009,49, 740–743.

Main drawback of ChemGrapher

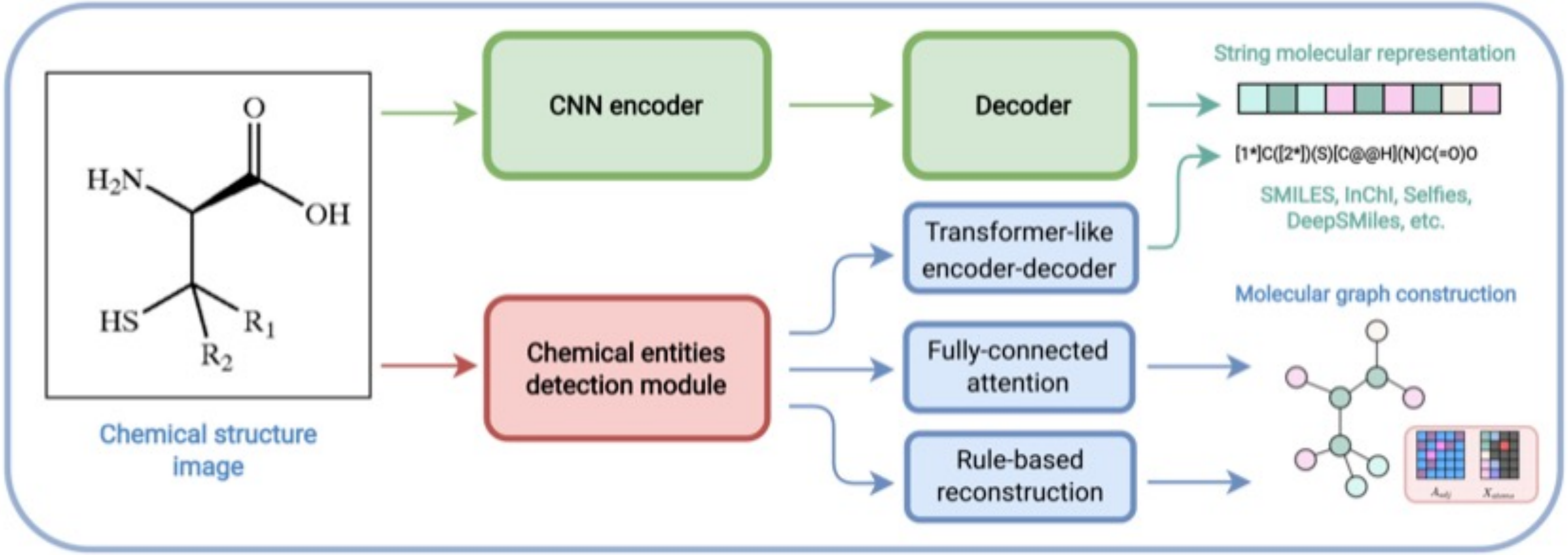
- ChemGrapher needs pixel level annotated images as training data
- End user is only interested in predicted graph (not pixel level annotations of images)
- Images are usually not pixel level annotated
- How to use (graph – not pixel level) labelled images to train ChemGrapher? For example you only have the SMILES of compound on Image.

Interesting related work: Object detection

Source: https://en.wikipedia.org/wiki/Object_detection

- End user is interested in predicted bounding boxes (instance level annotations of images)
- Pixel level annotations are also not widely available.
- How to train object detection with weak supervision?

Motivation of object detection approach



Source: Hormazabal, Rodrigo, et al. "CEDe: A collection of expert-curated datasets with atom-level entity annotations for Optical Chemical Structure Recognition." (accepted for NeurIPS 2022)

Motivation of object detection approach

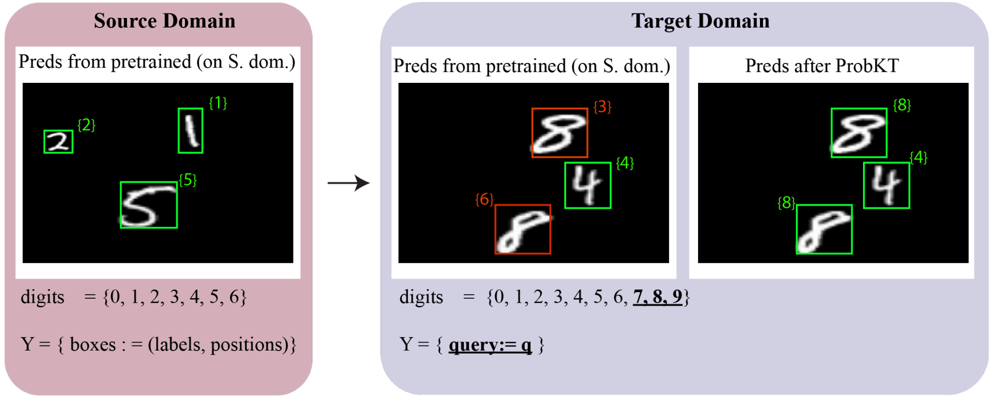
Table 2: CNN-decoder and chemical entity detection baseline models benchmark results on one synthetic and four available datasets coming from real scientific documents.

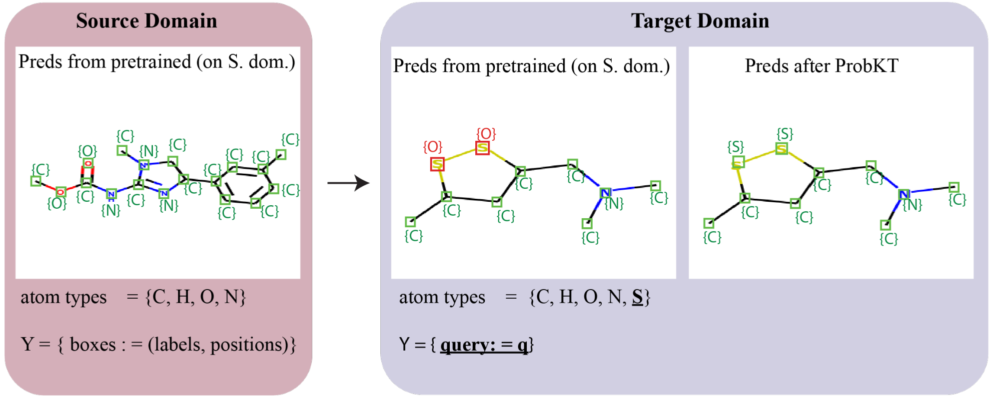
Test data	Traning dataset	Rule-based methods			Image-to-SMILES translation @ 1M data			Chemical entity detection @ 10K data		
		Imago [6]	MolVec [8]	OSRA [7]	CNN+ GRU [31, 32]	CNN+ GRU+ Attention [33]	CNN+ Transformer decoder [34]	DETR+ bond bbox connectivity [28]	FasterRCNN+ bond bbox connectivity [29]	FasterRCNN+ Instance-pair Transformer [29, 30]
Synthetic dataset (10K)	Synthetic	79.21	85.92	87.34	59.2	61.2	57.4	79.81	**88.96**	86.15
UOB (4592)	No pretraining	-	-	-	5.12	7.99	3.12	47.24	52.12	49.77
	Synthetic	-	-	-	32.01	36.37	34.84	65.59	74.09	67.94
	Fine-tuned	63.83	89.11	85.95	62.57	66.9	67.29	84.32	**91.49**	89.26
USPTO (4575)	No pretraining	-	-	-	4.12	5.32	2.19	52.91	58.24	51.99
	Synthetic	-	-	-	32.01	36.37	34.84	65.59	74.09	67.94
	Fine-tuned	86.91	88.13	87.83	48.52	55.17	53.51	86.12	**90.97**	88.11
CLEF (768)	No pretraining	-	-	-	1.43	1.56	-	48.12	55.29	47.25
	Synthetic	-	-	-	32.01	36.37	34.84	65.59	74.09	67.94
	Fine-tuned	66.41	82.94	**92.84**	35.16	39.19	36.59	73.31	86.59	81.64
JPO (360)	No pretraining	-	-	-	0.83	0.27	-	42.93	47.29	36.88
	Synthetic	-	-	-	32.01	36.37	34.84	65.59	74.09	67.94
	Fine-tuned	41.39	67.22	58.89	33.61	30.28	28.06	55.83	**74.12**	59.72

Source: Hormazabal, Rodrigo, et al. "CEDe: A collection of expert-curated datasets with atom-level entity annotations for Optical Chemical Structure Recognition." (accepted for NeurIPS 2022)

Updating Object Detection Models with Probabilistic Programming [1]

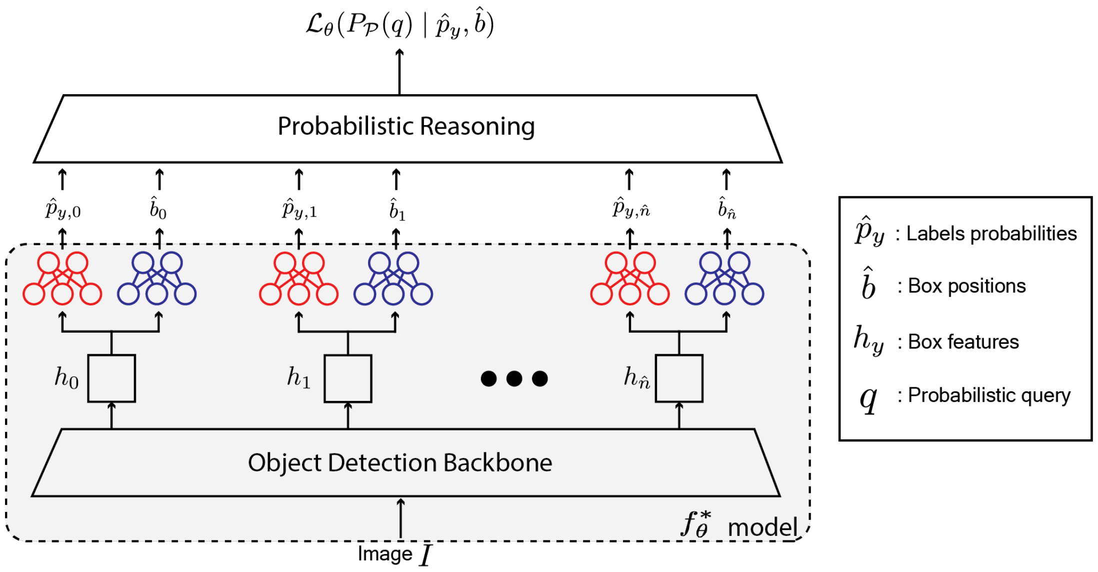
- Training object detection models requires instance level (bbox) annotated images
- Instance level annotations are often not available however knowledge built from existing image level annotated domains should be intuitively transferrable.
- ProbKT: should be able to fine-tune pretrained object detection models with 'arbitrary' supervision by probabilistic reasoning.





[1] Oldenhof, Martijn, et al. "Updating Object Detection Models with Probabilistic Programming." presented at ICML 2022 Workshop – UpML

Proposed Method: ProbKT



Source domain data set:
$\mathcal{D}_s = \{(I_s^i, b_s^i, y_s^i) : i = 1, ..., N_s)\}$, with N_s box level annotated images I_s
Box annotations: $b_s^i \in \mathbb{R}^{n_i \times 4}$

Target domain data set:
$\mathcal{D}_t = \{(I_t^i, q_t^i) : i = 1, ..., N_t)\}$, with N_t image-level annotated images.
Image-level annotations: queries q_t^i.

Pre-trained object detection model f_θ on the source domain.

Extracted backbone from f_θ and inserted it into a new model f_θ^*

The probability of queries q_t are evaluated and used in the loss:

$$\mathcal{L}_\theta = \sum_{(I_t, q_t) \in \mathcal{D}_t} -log P_\mathcal{P}(q_t \mid f_\theta^*(I_t))$$

How does the probabilistic reasoning work?

- In general probabilistic logical reasoning uses knowledge representation relying on probabilities that allow encoding uncertainty in knowledge.
- Knowledge can be encoded using probabilistic facts and logical rules. For example:
  - A probabilistic fact is: **“Alice and Bob will each pass their exam with probability 0.5”**
  - A logical rule: **“if both Alice and Bob pass their exam, they will host a party”**

Probabilistic Programming and Inference

Inference by computing the probability of a particular statement or **query**

For example query probability of:
"Alice and Bob will host a party"

- The query is executed by summing over all probabilities of occurrence of the different possible worlds *w* compatible qith the query *q*. So the probability of a query *q* in a program $\mathcal{P}$:

$$P_{\mathcal{P}}(q) = \sum_{w} P(w) \cdot \mathbb{I}[F(w) \equiv q]$$

- Where $F(w) \equiv q$ stands for the realization of *w* according rules *F* so that *q* is true.

Probabilistic Programming and Inference

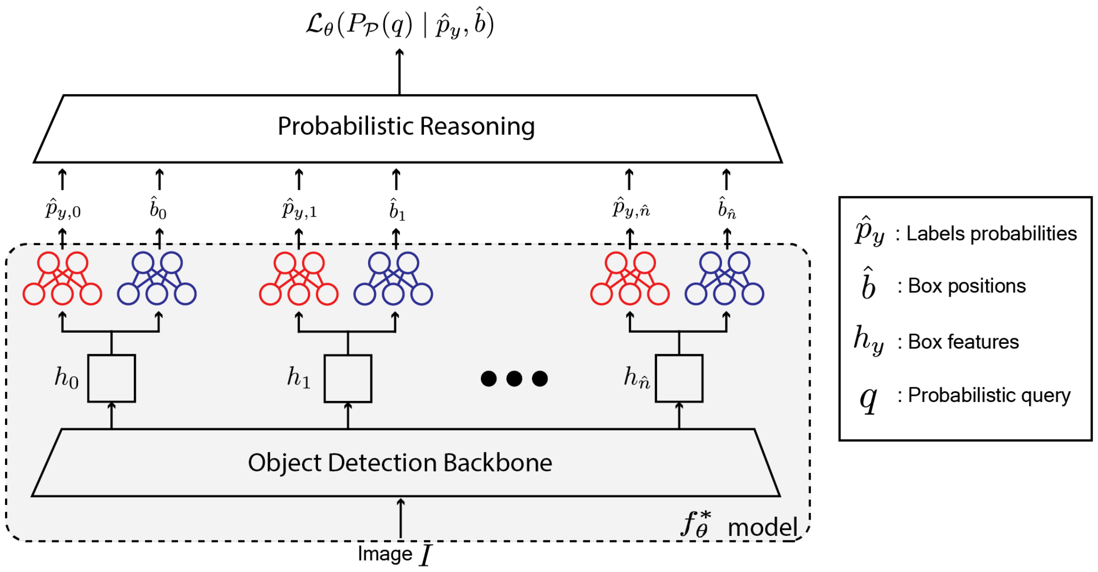
- Probable worlds so that *q* (🎉) is true according rules F:
  - Alice passes exam AND Bob passes exam → Rules F? → Party 🎉
  - ~~Alice not passes exam AND Bob not passes exam~~ → Rules F? → No Party 😞
  - ~~Alice passes exam AND Bob not passes exam~~ → Rules F? → No Party 😞
  - ~~Alice not passes exam AND Bob passes exam~~ → Rules F? → No Party 😞

- So,

$$P_{\mathcal{P}}(q) = \sum_{w} P(w) \cdot \mathbb{I}[F(w) \equiv q]$$

  - Only 1 possible world where q (🎉) is true : 0.5*0.5 = 0.25

Proposed Method: ProbKT



Source domain data set:
$\mathcal{D}_s = \{(I_s^i, b_s^i, y_s^i) : i = 1, ..., N_s)\}$, with N_s box level annotated images I_s
Box annotations: $b_s^i \in \mathbb{R}^{n_i \times 4}$

Target domain data set:
$\mathcal{D}_t = \{(I_t^i, q_t^i) : i = 1, ..., N_t)\}$, with N_t image-level annotated images.
Image-level annotations: queries q_t^i.

Pre-trained object detection model f_θ on the source domain.

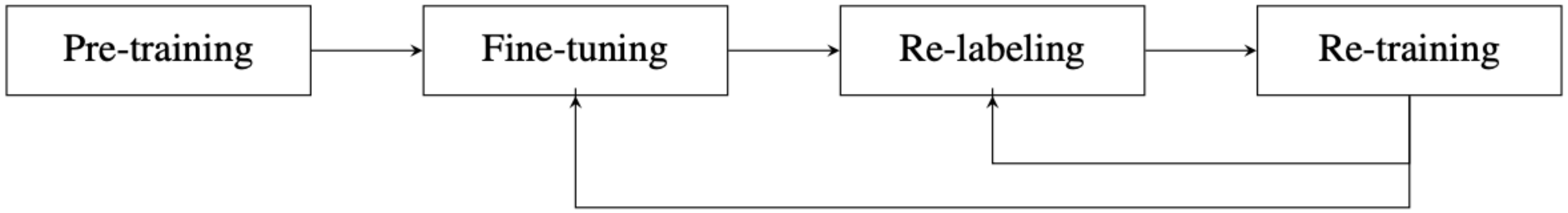
Extracted backbone from f_θ and inserted it into a new model f_θ^*

The probability of queries q_t are evaluated and used in the loss:

$$\mathcal{L}_\theta = \sum_{(I_t, q_t) \in \mathcal{D}_t} -log P_\mathcal{P}(q_t \mid f_\theta^*(I_t))$$

Iterative Relabeling

- To further improve the performance, we propose an iterative relabeling strategy that consists in multiple steps : **fine-tuning**, **re-labeling** and **re -training**. A similar has also been proposed by Zhong et al. [1]

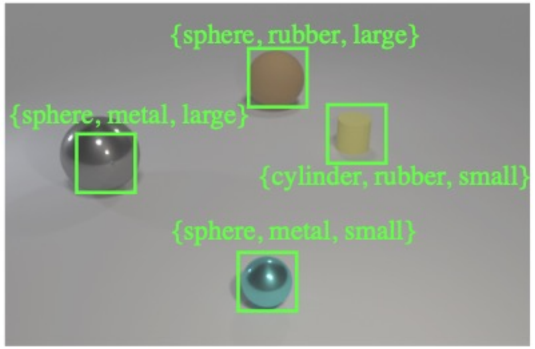


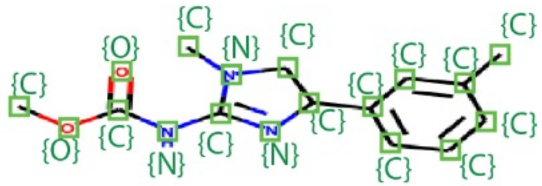
[1] Yuanyi Zhong, Jianfeng Wang, Jian Peng, and Lei Zhang. Boosting weakly supervised object detection with progressive knowledge transfer. In European conference on computer vision, pages 615–631. Springer, 2020

Experiments

Weakly supervised knowledge transfer with class counts

- Query q = number of objects from each class in the image
- 2 datasets : CLEVR-mini and Molecules dataset





Experiments

Weakly supervised knowledge transfer with class counts

- Baseline models
  - Resnet50-CAM (Xue et al [1])
  - WSOD-transfer (Zhong et al [2])

[1] Yao Xue, Nilanjan Ray, Judith Hugh, and Gilbert Bigras. Cell counting by regression using convolutional neural network. In European Conference on Computer Vision, pages 274–290. Springer, 2016.
[2] Yuanyi Zhong, Jianfeng Wang, Jian Peng, and Lei Zhang. Boosting weakly supervised object detection with progressive knowledge transfer. In European conference on computer vision, pages 615–631. Springer, 2020.

Results 1/3

Model	Data Domain	CLEVR count acc.	CLEVR mAP (mAP@IoU=0.5)	Mol. count. acc	Mol. mAP (mAP@IoU=0.5)
Resnet50-CAM	target domain	0.97 ± 0.005	0.036 ± 0.014 (0.200 ± 0.071)	$\mathbf{0.978 \pm 0.004}$	0.0 ± 0.0 (0 ± 0)
Resnet50-CAM	OOD	0.831 ± 0.016	0.029 ± 0.010 (0.153 ± 0.044)	0.0 ± 0.0	n/a*
Resnet50-CAM	source domain	0.993 ± 0.003	0.035 ± 0.019 (0.178 ± 0.084)	0.828 ± 0.021	0.0 ± 0.0 (0 ± 0)
WSOD-transfer	target domain	0.944 ± 0.004	0.844 ± 0.005 (0.988 ± 0.001)	0.001 ± 0.0	0.018 ± 0.004 (0.061 ± 0.011)
WSOD-transfer	OOD	0.73 ± 0.011	0.79 ± 0.005 (0.969 ± 0.001)	0.003 ± 0.002	n/a*
WSOD-transfer	source domain	0.989 ± 0.001	0.926 ± 0.001 (0.995 ± 0.0)	0.0 ± 0.0	0.021 ± 0.003 (0.069 ± 0.009)
DETR-joint	target domain	0.159 ± 0.133	0.579 ± 0.012 (0.684 ± 0.019)	0.357 ± 0.196	0.197 ± 0.055 (0.481 ± 0.071)
DETR-joint	OOD	0.084 ± 0.039	0.534 ± 0.012 (0.66 ± 0.012)	0.024 ± 0.021	n/a*
DETR-joint	source dom.	0.923 ± 0.049	0.908 ± 0.017 (0.992 ± 0.001)	0.232 ± 0.127	0.23 ± 0.063 (0.565 ± 0.08)
RCNN (pre-trained)	target domain	0.0 ± 0.0	0.586 ± 0.014 (0.598 ± 0.013)	0.592 ± 0.007	$\mathbf{0.568 \pm 0.005}$ (0.785 ± 0.004)
RCNN (pre-trained)	OOD	0.0 ± 0.0	0.582 ± 0.012 (0.603 ± 0.011)	0.348 ± 0.036	n/a*
RCNN (pre-trained)	source domain	0.988 ± 0.002	$\mathbf{0.984 \pm 0.01}$ (0.996 ± 0.0)	0.948 ± 0.004	$\mathbf{0.737 \pm 0.005}$ $(\mathbf{0.979 \pm 0.0})$
DETR (pre-trained)	target domain	0.0 ± 0.0	0.498 ± 0.019 (0.533 ± 0.024)	0.464 ± 0.033	0.314 ± 0.006 (0.542 ± 0.006)
DETR (pre-trained)	OOD	0.0 ± 0.0	0.477 ± 0.013 (0.531 ± 0.021)	0.002 ± 0.001	n/a*
DETR (pre-trained)	source domain	0.97 ± 0.009	0.945 ± 0.009 (0.992 ± 0.001)	0.581 ± 0.022	0.409 ± 0.005 (0.722 ± 0.004)
ProbKT (DETR)	target domain	0.946 ± 0.014	0.803 ± 0.011 (0.989 ± 0.006)	0.508 ± 0.027	0.204 ± 0.02 (0.507 ± 0.014)
ProbKT (DETR)	OOD	0.726 ± 0.035	0.715 ± 0.006 (0.974 ± 0.006)	0.004 ± 0.003	n/a*
ProbKT (DETR)	source domain	0.987 ± 0.003	0.948 ± 0.005 (0.995 ± 0.001)	0.549 ± 0.026	0.38 ± 0.013 (0.713 ± 0.006)
ProbKT (RCNN)	target domain	$\mathbf{0.975 \pm 0.003}$	$\mathbf{0.856 \pm 0.039}$ (0.993 ± 0.001)	0.942 ± 0.009	0.289 ± 0.041 $(\mathbf{0.829 \pm 0.054})$
ProbKT (RCNN)	OOD	0.89 ± 0.022	$\mathbf{0.833 \pm 0.042}$ $(\mathbf{0.991 \pm 0.001})$	$\mathbf{0.603 \pm 0.037}$	n/a*
ProbKT (RCNN)	source domain	$\mathbf{0.995 \pm 0.002}$	0.941 ± 0.041 (0.998 ± 0.001)	$\mathbf{0.96 \pm 0.002}$	0.666 ± 0.005 (0.978 ± 0.002)

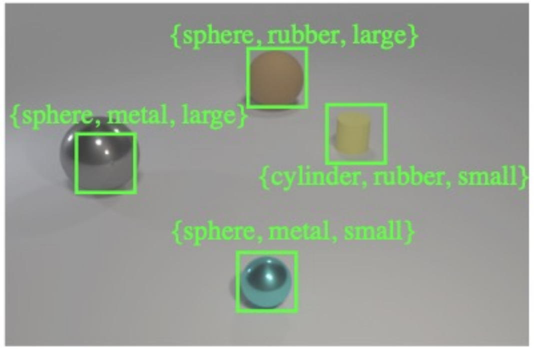
Table 2: Results of the experiments for the datasets: CLEVR-mini and Molecules. Reported test accuracies over the 5 folds. Best method is in bold for each metric and data distribution.

*OOD test set of Molecules dataset has no bounding box labels.

Experiments

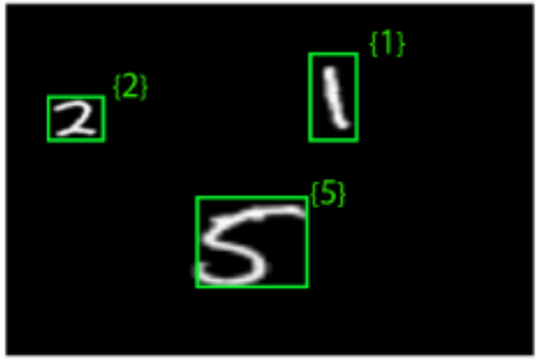
Other types of weak supervision

- Ranges of Class Counts
  - For example:
    1 cylinder, > 1 spheres



- Sum of digits
  - For example:

    8



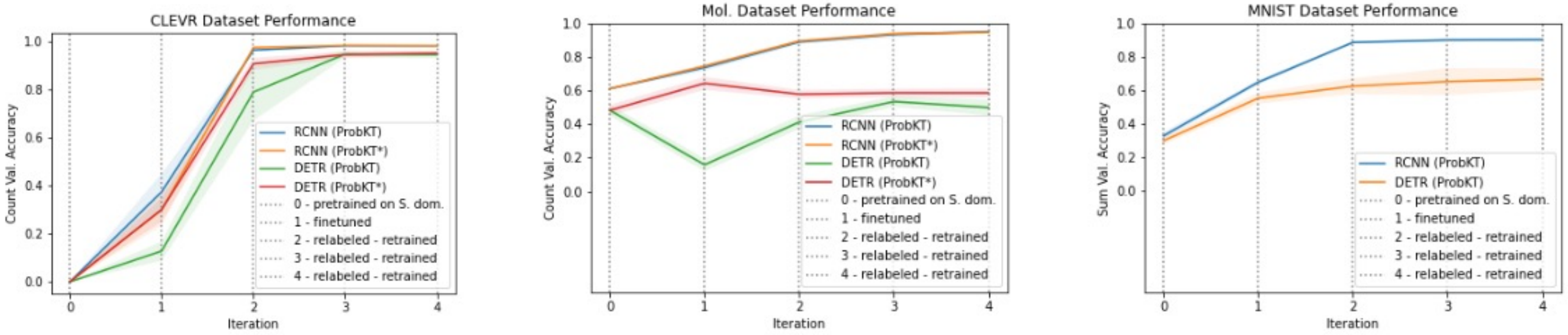
Results 2/3

Model	Data Domain	MNIST count acc.	MNIST sum acc.	MNIST mAP (mAP@IoU=0.5)	CLEVR* count acc.	CLEVR* mAP (mAP@IoU=0.5)
Resnet50-CAM	target domain	0.044 ± 0.041	0.506 ± 0.063	0.003 ± 0.003(0.014 ± 0.011)	n/a	n/a
Resnet50-CAM	OOD	0.01 ± 0.009	0.015 ± 0.004	0.003 ± 0.002(0.011 ± 0.007)	n/a	n/a
Resnet50-CAM	source domain	0.127 ± 0.132	0.649 ± 0.108	0.005 ± 0.004(0.028 ± 0.018)	n/a	n/a
WSOD-transfer	target domain	n/a	n/a	n/a	0.944 ± 0.004	$\mathbf{0.844 \pm 0.005}$ (0.988 ± 0.001)
WSOD-transfer	OOD	n/a	n/a	n/a	0.73 ± 0.011	0.79 ± 0.005 (0.969 ± 0.001)
WSOD-transfer	source domain	n/a	n/a	n/a	0.989 ± 0.001	0.926 ± 0.001 (0.995 ± 0.0)
RCNN (pre-trained)	target domain	0.292 ± 0.005	0.298 ± 0.005	0.632 ± 0.014 (0.685 ± 0.002)	0.0 ± 0.0	0.586 ± 0.014 (0.598 ± 0.013)
RCNN (pre-trained)	OOD	0.205 ± 0.004	0.212 ± 0.004	0.631 ± 0.013 (0.683 ± 0.002)	0.0 ± 0.0	0.582 ± 0.012 (0.603 ± 0.011)
RCNN (pre-trained)	source domain	0.961 ± 0.008	0.961 ± 0.008	$\mathbf{0.917 \pm 0.021}$ (0.988 ± 0.002)	0.988 ± 0.002	$\mathbf{0.984 \pm 0.01}$ (0.996 ± 0.0)
ProbKT (RCNN)	target domain	$\mathbf{0.902 \pm 0.005}$	$\mathbf{0.903 \pm 0.005}$	$\mathbf{0.786 \pm 0.021}$ (0.974 ± 0.001)	$\mathbf{0.971 \pm 0.006}$	0.838 ± 0.034 ($\mathbf{0.993 \pm 0.001}$)
ProbKT (RCNN)	OOD	$\mathbf{0.863 \pm 0.008}$	$\mathbf{0.865 \pm 0.008}$	$\mathbf{0.778 \pm 0.021}$ (0.97 ± 0.001)	$\mathbf{0.884 \pm 0.01}$	$\mathbf{0.812 \pm 0.036}$ ($\mathbf{0.991 \pm 0.001}$)
ProbKT (RCNN)	source domain	$\mathbf{0.967 \pm 0.004}$	$\mathbf{0.967 \pm 0.004}$	0.873 ± 0.016 (0.989 ± 0.001)	$\mathbf{0.994 \pm 0.001}$	0.922 ± 0.035 ($\mathbf{0.998 \pm 0.001}$)

Table 3: Results of the experiments on the MNIST object detection dataset and on CLEVR* dataset. Reported test accuracies over the 5 folds. Best method is in bold for each metric and data distribution.

*CLEVR dataset using ranges of class counts as labels instead of exact class counts.

Results 3/3



(a) CLEVR iterative relabeling (b) Molecules iterative relabeling (c) MNIST iterative relabeling

Figure 4: Iterative relabeling performance for the different datasets

Conclusions for Part I

- Training object detection usually requires large amounts of richly annotated images.
- We proposed a novel approach to train object detection models by leveraging richly annotated datasets from other domains and allowing arbitrary types of weak supervision on the target domain.
- We empirically demonstrated that ProbKT outperforms existing methods in a wide range of cases.
- ProbKT allows to consider a wide range of supervisory signals in the target domain.

The intersection of Optical Chemical Structure Recognition (OCSR) and object detection

- Part I:
  - Introduction and Motivation: ChemGrapher
  - (Weakly Supervised) Object Detection
  - Updating Object Detection Models with Probabilistic Programming
- Part II:
  - Experiments in W&B

Weights & Biases

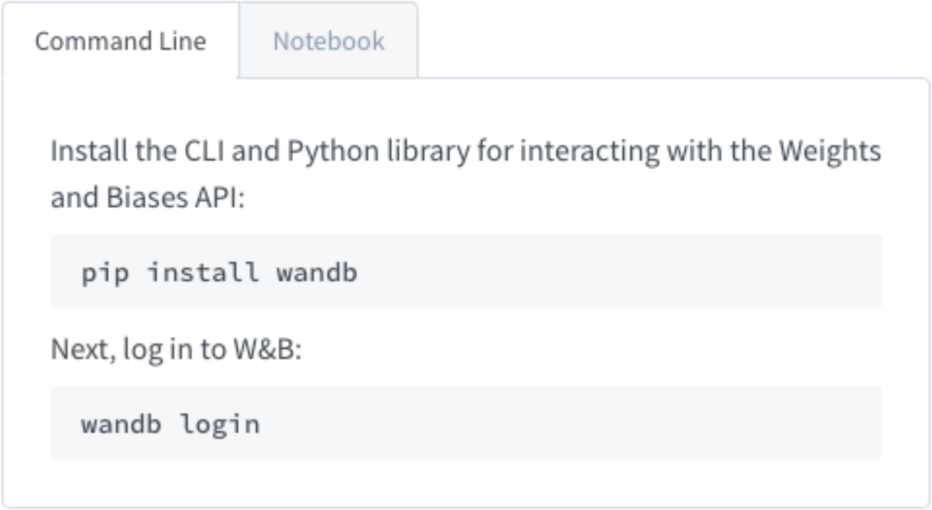
- **Experiment Tracking**: Visualize experiments in real time
- **Hyperparameter Tuning**: Optimize models quickly
- **Data and Model Versioning**: Version datasets and models
- **Model Management**: Manage the model lifecycle from training to production
- **Data Visualization**: Visualize predictions across model versions
- **Collaborative Reports**: Describe and share findings
- **Integrations**: PyTorch, Keras, Hugging Face, and more
- **Private-Hosting**: Private cloud and local hosting of the W&B app

Weights & Biases

- How did I use W&B?
  - Organize experiments:
    - Store results/models
    - Visualize quickly
    - Repeat experiments easily
  - Scale experiments to different servers/clusters

Weights & Biases

1. Set up wandb:
   - Sign up for an account on https://wandb.ai/site
   - Install wandb Python library
   - Login to your wandb account using wandb commands locally. You will need an API key you can find here: https//wandb.ai/authorize (on all machines that will run experiments)



Weights & Biases

2. Start a new run
   - In general:

```python
import wandb

wandb.init(project="my-awesome-project")
```

   - Our use case:

create wandb_config.py:

```python
import wandb
wandb.init(anonymous="allow", reinit=False)
ENTITY=wandb.run.entity
```

import in other scripts:

```python
from robust_detection import wandb_config
```

Weights & Biases

3. Track Metrics

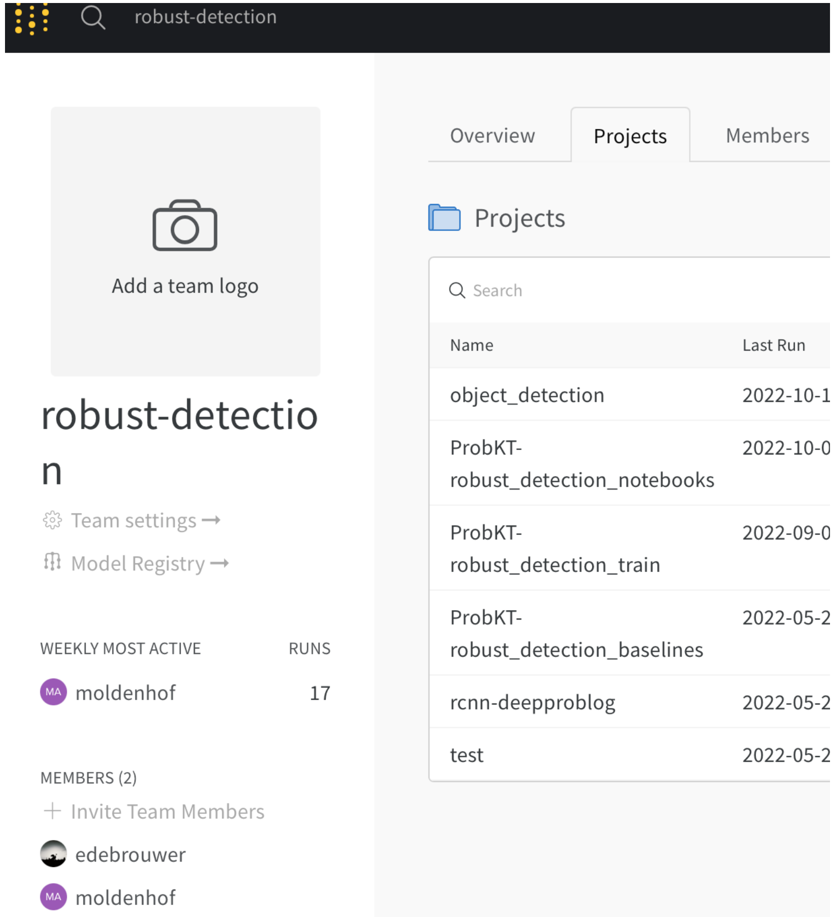
- In general:
  - Use wandb.log() to track metrics
  - For example:

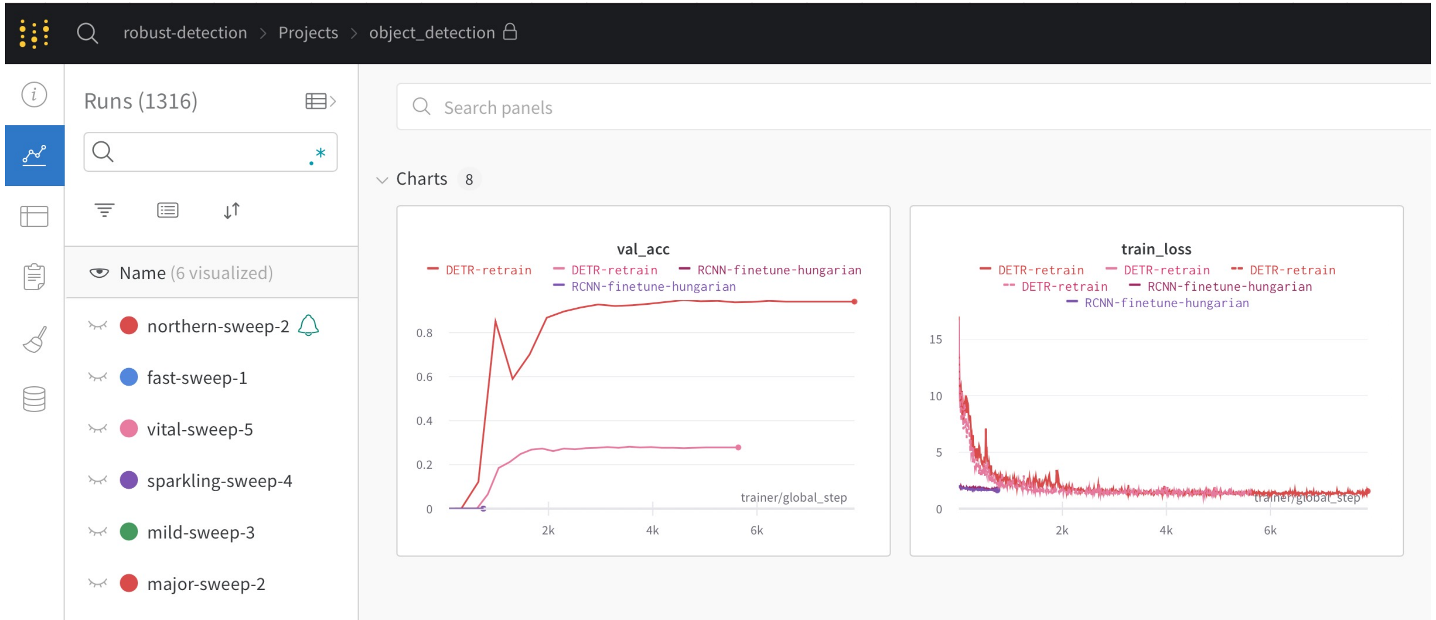
```
wandb.log({'accuracy': train_acc, 'loss': train_loss})
```

- Our Use Case:
  - Using PyTorch Lightning:

```
wandb_logger = WandbLogger()
trainer = Trainer(logger=wandb_logger)
```

Weights & Biases





Weights & Biases

4. Model Checkpointing

- Easy in Pytorch Lightning

```
# log model only if `val_accuracy` increases
wandb_logger = WandbLogger(log_model="all")
checkpoint_callback = ModelCheckpoint(monitor="val_accuracy", mode="max")
trainer = Trainer(logger=wandb_logger, callbacks=[checkpoint_callback])

#save hyperparameters
class RCNN(pl.LightningModule):
        def __init__(self, len_dataloader, hidden_layer, num_classes,
score_thresh,model_type = "mask_rcnn", pre_trained = True, backbone_run_name = None,
agg_case=False, **kwargs):
                super().__init__()
                self.save_hyperparameters()
                …
```

Weights & Biases

5. Sweeps:

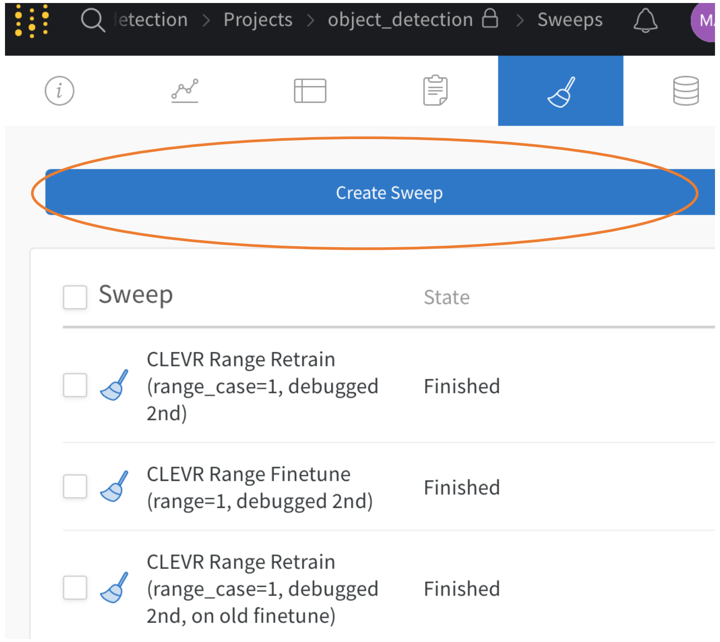
- In General:

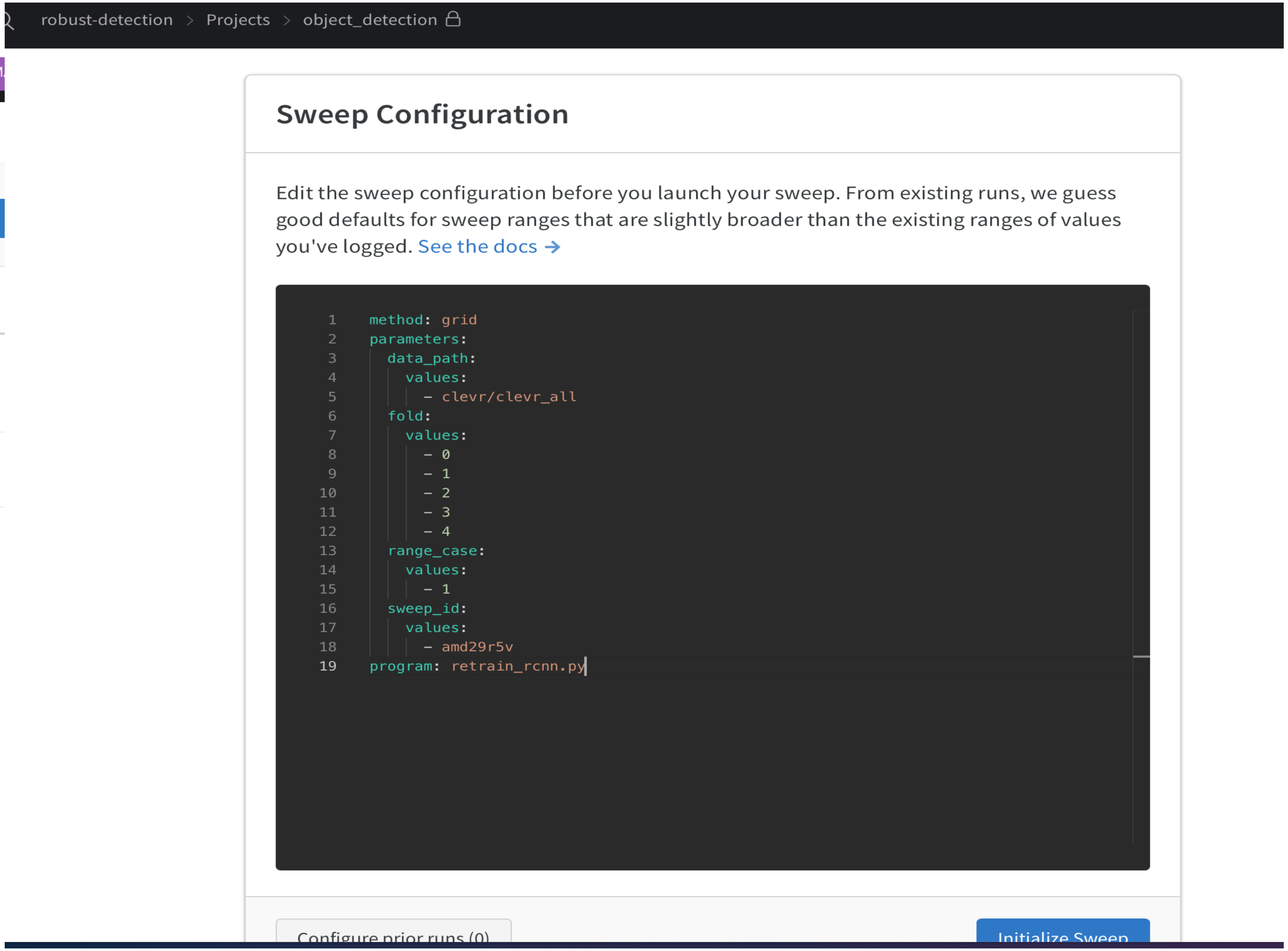
A Weights & Biases Sweep combines a strategy for exploring hyperparameter values with the code that evaluates them. The strategy can be as simple as trying every option or as complex as Bayesian Optimization and Hyperband.

- Our Use Case:

```
parameter_name:
        values:
        - 8
        - 6
        - 7
        - 5
        - 3
        - 0
        - 9
```

Sweeps in W&B - example





Launch Agents

Launch Agent(s)

On your own machines, launch one or more agents to run the sweep. When you launch an agent, this page will become your sweep dashboard. See the docs →

Run the wandb agent command to start a sweep.

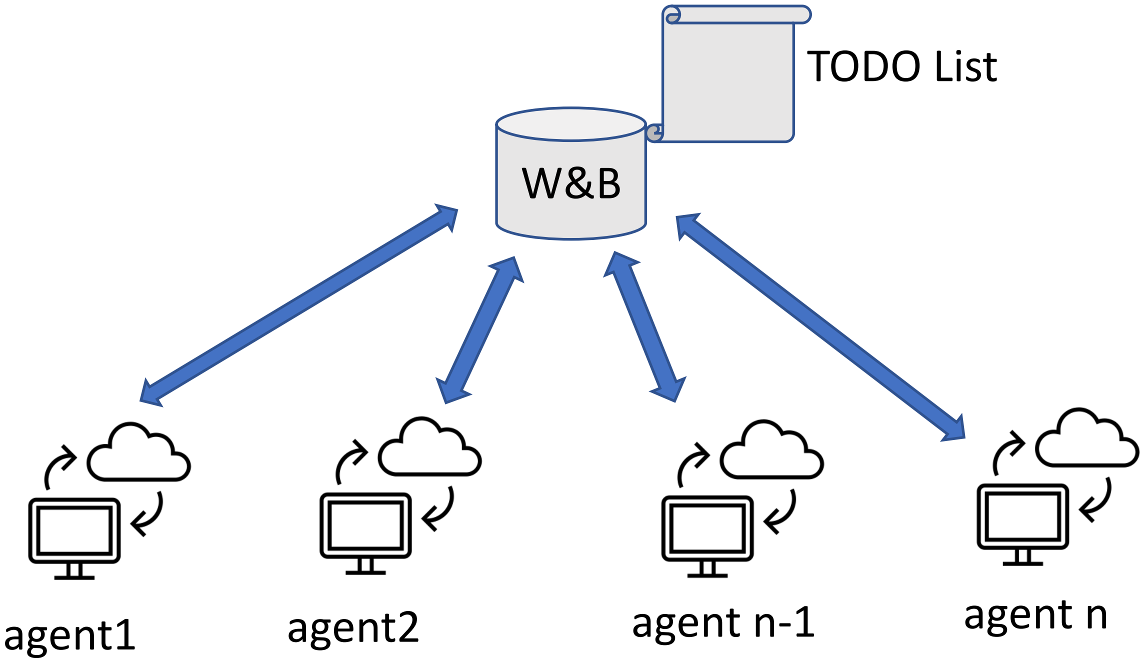
```
$ wandb agent robust-detection/object_detection/0nubz265
```

The agent will request the next set of parameters for your sweep, then launch a run with those parameters.

Once the run finishes, the agent will run another command with different parameter values.

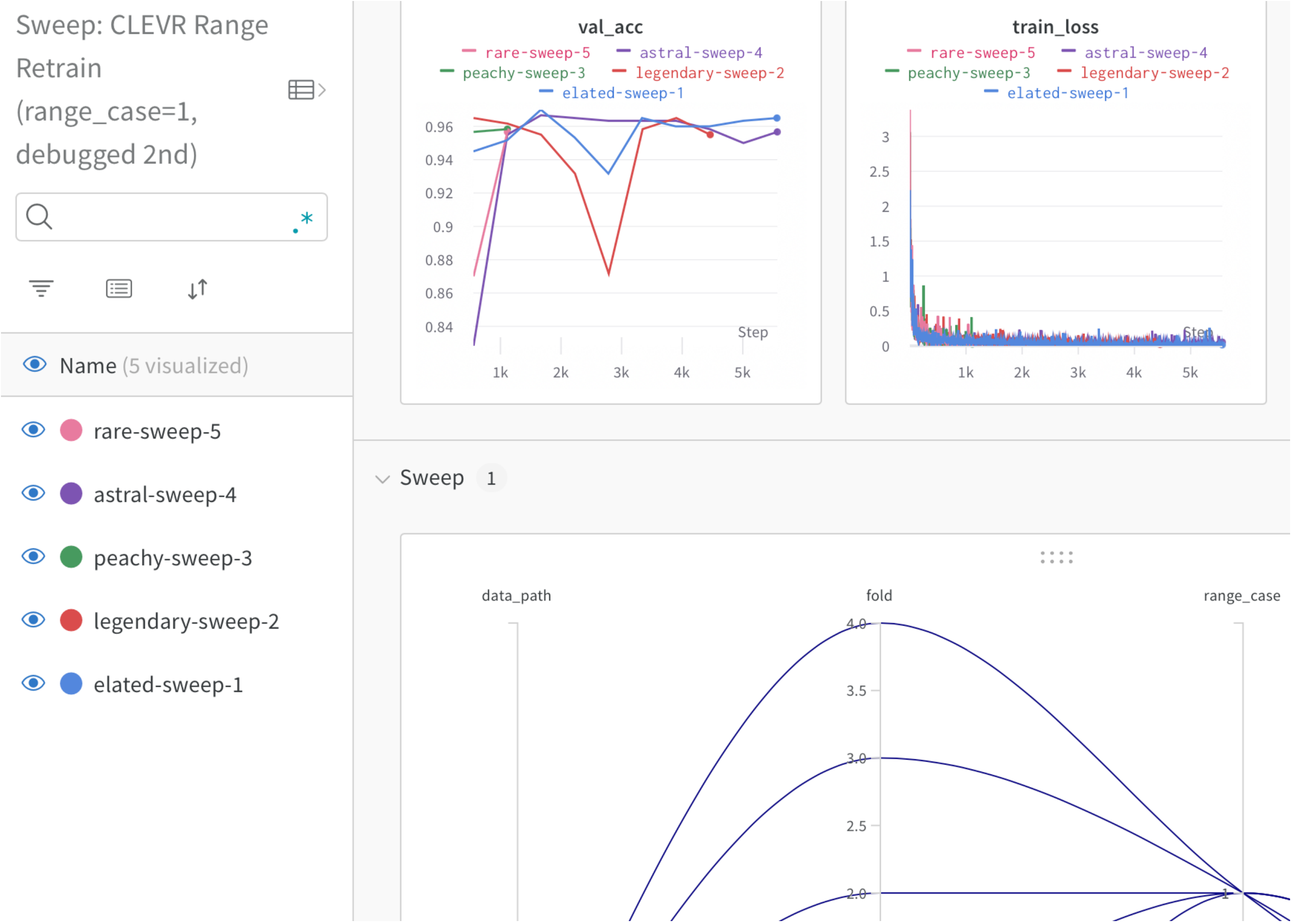
You can run multiple wandb agents on a single machine or different machines.

View Sweep




```
(conda_poetry) moldenho@kusanagi:~/Projects/ProbKT/robust_detection/train$ wandb agent robust-detection/object_detection/unug3
wandb: Starting wandb agent 🕵️
2022-10-21 15:21:24,954 - wandb.wandb_agent - INFO - Running runs: []
2022-10-21 15:21:25,271 - wandb.wandb_agent - INFO - Agent received command: run
2022-10-21 15:21:25,272 - wandb.wandb_agent - INFO - Agent starting run with config:
        data_path: clevr/clevr_all
        fold: 0
        range_case: 1
        sweep_id: 1sidtm65
2022-10-21 15:21:25,294 - wandb.wandb_agent - INFO - About to run command: /usr/bin/env python retrain_rcnn.py --data_path=cle
wandb: Currently logged in as: moldenhof (robust-detection). Use `wandb login --relogin` to force relogin
2022-10-21 15:21:30,308 - wandb.wandb_agent - INFO - Running runs: ['bcjyzmg2']
wandb: wandb version 0.13.4 is available!  To upgrade, please run:
wandb:  $ pip install wandb --upgrade
wandb: Tracking run with wandb version 0.12.16
wandb: Run data is saved locally in /home/moldenho/Projects/ProbKT/robust_detection/train/wandb/run-20221021_152126-bcjyzmg2
wandb: Run `wandb offline` to turn off syncing.
wandb: Syncing run fancy-sweep-1
wandb: ⭐️ View project at https://wandb.ai/robust-detection/object_detection
wandb: 🧹 View sweep at https://wandb.ai/robust-detection/object_detection/sweeps/unug3949
wandb: 🚀 View run at https://wandb.ai/robust-detection/object_detection/runs/bcjyzmg2
```

Follow-up



Sweep: CLEVR Range Retrain (range_case=1, debugged 2nd)
Name (5 visualized)
rare-sweep-5
astral-sweep-4
peachy-sweep-3
legendary-sweep-2
elated-sweep-1
val_acc
train_loss
Step
Sweep 1
data_path
fold
range_case

Post process Sweeps

```
api = wandb.Api()
sweep = api.sweep(f"/{ENTITY}/object_detection/"+sweep_id)
sweep_runs = sweep.runs

best_runs = []
fold = args.fold

runs_fold = [r for r in sweep_runs if (r.config.get("fold")==fold)]
runs_fold_sorted = sorted(runs_fold,key = lambda run: run.summary.get("restored_val_acc"), reverse = False)
best_run = runs_fold_sorted[0]
model_cls = RCNN
data_cls = Objects_RCNN

run_name = best_run.id
run = api.run(f"{ENTITY}/object_detection/{run_name}")

fname = [f.name for f in run.files() if "ckpt" in f.name][0]
run.file(fname).download(replace = True, root = ".")
model = model_cls.load_from_checkpoint(fname)
```

Conclusions for Part II

- W&B
  - Helps to track experiments
  - Visualize
  - Scale easily
  - Post process and Rerun
  - …