Machine Learning Systems Design

Lecture 12:

- Model Deployment at Stitch Fix
- Experiment tracking & versioning with WandB

Model Deployment @ Stitch Fix

PDF

Stefan Krawczyk
Model Lifecycle Management
@ Stitch Fix

Weights & Biases Tutorials

Lavanya Shukla
Head of Growth @ WandB

Reproducible ML At Scale

Lavanya Shukla, Head of Growth

Agenda

1. Why do we need ML reproducibility?
2. Model understanding through reproducibility
3. Interactive model visualization

Why do we need ML reproducibility?

Training the same model,
with the same hyperparameters
… doesn't always lead to the same results

Training the same model,
with the same hyperparameters
… doesn't always lead to the same results

- Datasets – order of training examples
- Differences in hardware, GPUs
- Differences in ML library versions
- Initialization of weights
- Model architecture randomness – dropout

The more our ML models impact the real world, the more we need them to be reproducible.

- Who gets a loan?
- Who gets indicted?
- How long is someone's sentence?
- Who's resume is good enough to be given an interview?
- Who is trustworthy? Responsible?

ML in regulated industries – An example

Reproducibility gap

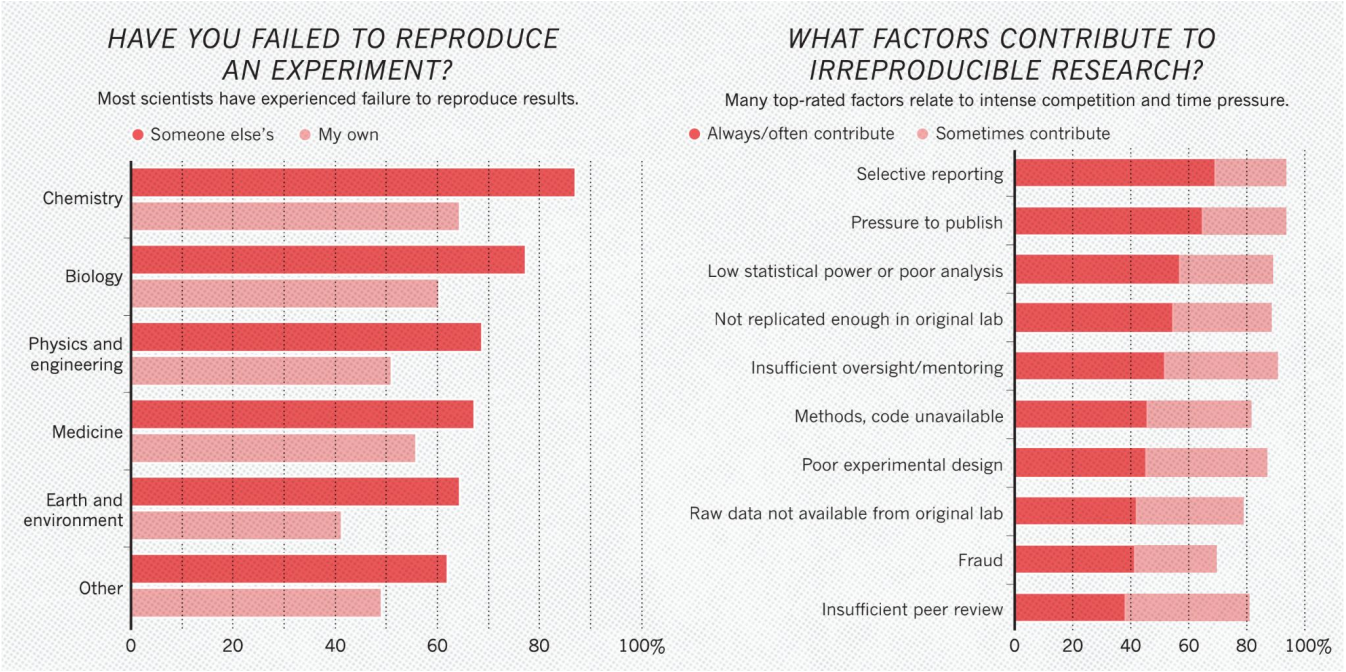
- **Lack of access** to the same training data / **differences in data distribution;**
- Lack of availability of the code necessary to run the experiments, or **errors in the code**;
- **Under-specification of the metrics** used to report results;
- **Improper use of statistics to analyze results**, such as claiming significance without
- **Selective reporting of results** and ignoring the danger of adaptive overfitting;
- **Over-claiming of the results**, by drawing conclusions that go beyond the evidence

Source: Paper by Joelle Pineau, Koustuv Sinha et all

Is there a reproducibility crisis?

Have tried and failed to reproduce an experiment?

How many of you have tried and failed to reproduce an experiment?

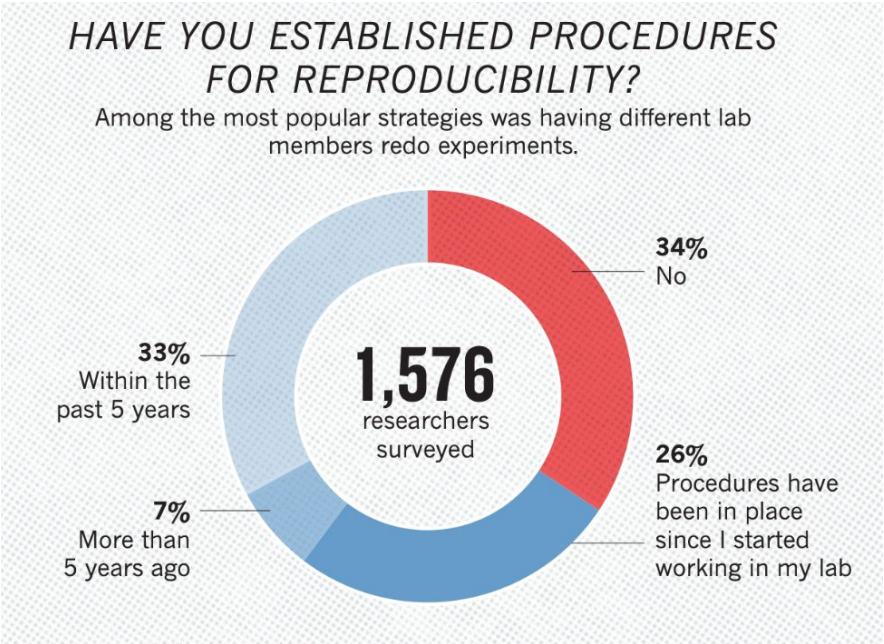


Source: Nature

Do you have a process for making your models reproducible?

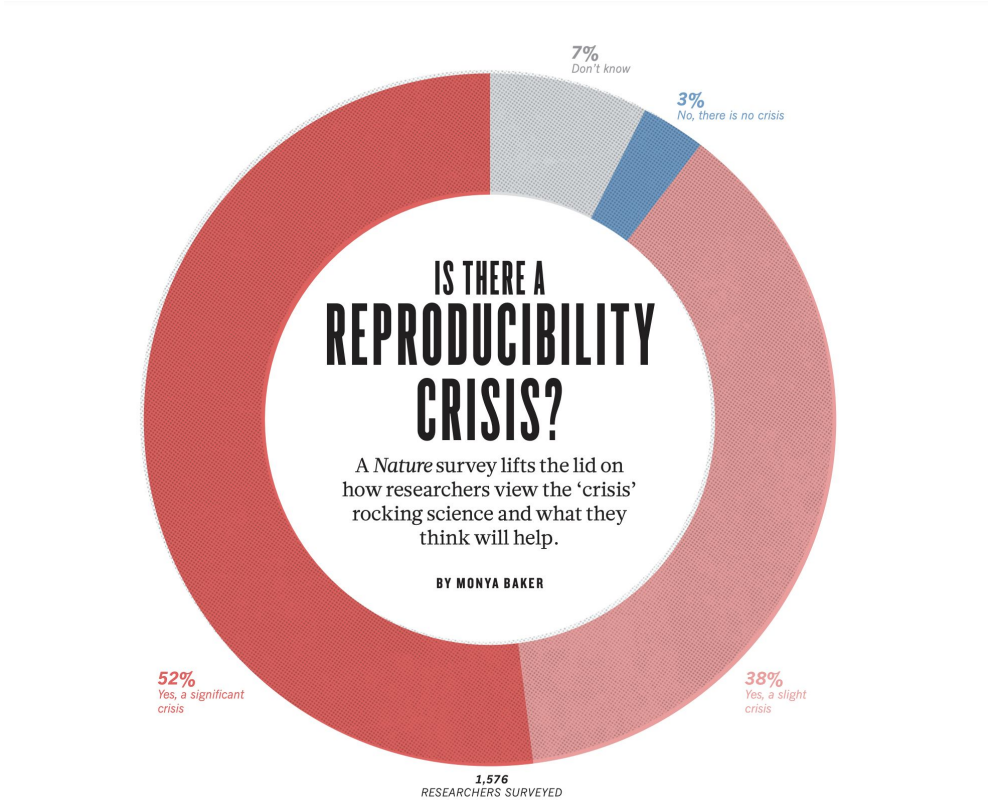
Source: Nature

Do you have a process for making your models reproducible?



Source: Nature

Is there a reproducibility crisis?



Source: Nature

“REPRODUCIBILITY IS LIKE BRUSHING YOUR TEETH. ONCE YOU LEARN IT, IT BECOMES A HABIT.”

What does it mean to be reproducible?

For all **models** and **algorithms** presented, check if you include:

- ❑ A clear description of the mathematical setting, algorithm, and/or model.
- ❑ A clear explanation of any assumptions.
- ❑ An analysis of the complexity (time, space, sample size) of any algorithm.

For any **theoretical claim**, check if you include:

- ❑ A clear statement of the claim.
- ❑ A complete proof of the claim.

For all **datasets** used, check if you include:

- ❑ The relevant statistics, such as number of examples.
- ❑ The details of train / validation / test splits.
- ❑ An explanation of any data that were excluded, and all pre-processing step.
- ❑ A link to a downloadable version of the dataset or simulation environment.
- ❑ For new data collected, a complete description of the data collection process, such as instructions to annotators and methods for quality control.

Source: Checklist by Joelle Pineau – Neurips Reproducibility Chair

For all shared **code** related to this work, check if you include:

- ❑ Specification of dependencies.
- ❑ Training code.
- ❑ Evaluation code.
- ❑ (Pre-)trained model(s).
- ❑ README file includes table of results accompanied by precise command to run to produce those results.

For all reported **experimental results**, check if you include:

- ❑ The range of hyper-parameters considered, method to select the best hyper-parameter configuration, and specification of all hyper-parameters used to generate results.
- ❑ The exact number of training and evaluation runs.
- ❑ A clear definition of the specific measure or statistics used to report results.
- ❑ A description of results with central tendency (e.g. mean) & variation (e.g. error bars).
- ❑ The average runtime for each result, or estimated energy cost.
- ❑ A description of the computing infrastructure used.

Source: Checklist by Joelle Pineau – Neurips Reproducibility Chair

Model understanding through reproducibility

PROBLEM

Massive developer tools gap for ML

SOFTWARE 1.0 - WRITE CODE

Design	Version Code	Code & Collaborate	Deploy to Infra	CI/CD	Production Monitoring
Figma	GitHub	Jira	HashiCorp	circleci	PagerDuty
Sketch	GitLab	VS Code	puppet	Jenkins	DATADOG

SOFTWARE 2.0 - TRAIN MODELS

Prep & Visualize Data	Version Data & Models	Experiment Tracking	Manage Model Pipeline	Model CI/CD	Production monitoring
Custom apps Notebooks	Files in S3	Text files Screenshots	Text files	Custom scripts	Nothing

BCP + W&B INTEGRATED SOLUTION

ML Ops Platform with Interoperable Applications

Version data & pipelines	Prep & visualize data	Track model development	Optimize models	Collaborative analysis	Production monitoring
					
Artifacts	Tables	Experiments	Sweeps	Reports	*Coming Soon*

ML LIBRARIES	Hugging Face	SpaCy	Detectron	Yolo V5	...
FRAMEWORKS	PyTorch	Keras	TensorFlow	XGBoost	...
PARTNERSHIP	NVIDIA. Base Command Platform				

Making ML workflows reproducible, one step at a time

Before W&B

```
Layer (type)                 Output Shape              Param #     Connected to
==================================================================================================
conv2d_1_input (InputLayer)  (None, 299, 299, 3)  0
__________________________________________________________________________________________________
lambda_1 (Lambda)            (None, 299, 299, 3)  0           conv2d_1_input[0][0]
__________________________________________________________________________________________________
lambda_2 (Lambda)            (None, 299, 299, 3)  0           conv2d_1_input[0][0]
__________________________________________________________________________________________________
sequential_1 (Sequential)    (None, 10)            910922      lambda_1[0][0]
                                                               lambda_2[0][0]
__________________________________________________________________________________________________
activation_7 (Concatenate)   (None, 10)            0           sequential_1[1][0]
                                                               sequential_1[2][0]
==================================================================================================
Total params: 910,922
Trainable params: 910,922
Non-trainable params: 0
__________________________________________________________________________________________________
None
Found 49999 images belonging to 10 classes.
Found 8000 images belonging to 10 classes.
Epoch 1/50
39/39 [==============================] - 270s 7s/step - loss: 2.3153 - acc: 0.1178 - val_loss: 2.2428 - val_acc: 0.1589
Epoch 2/50
39/39 [==============================] - 263s 7s/step - loss: 2.2588 - acc: 0.1538 - val_loss: 2.1890 - val_acc: 0.2174
Epoch 3/50
39/39 [==============================] - 262s 7s/step - loss: 2.2060 - acc: 0.1827 - val_loss: 2.1767 - val_acc: 0.2096
Epoch 4/50
39/39 [==============================] - 263s 7s/step - loss: 2.1640 - acc: 0.2107 - val_loss: 2.1133 - val_acc: 0.2357
Epoch 5/50
39/39 [==============================] - 261s 7s/step - loss: 2.0827 - acc: 0.2432 - val_loss: 2.1499 - val_acc: 0.2344
Epoch 6/50
39/39 [==============================] - 262s 7s/step - loss: 2.0810 - acc: 0.2508 - val_loss: 2.0289 - val_acc: 0.2604
Epoch 7/50
39/39 [==============================] - 262s 7s/step - loss: 2.0575 - acc: 0.2602 - val_loss: 1.9912 - val_acc: 0.2865
Epoch 8/50
39/39 [==============================] - 261s 7s/step - loss: 2.0355 - acc: 0.2734 - val_loss: 2.0319 - val_acc: 0.2409
Epoch 9/50
39/39 [==============================] - 263s 7s/step - loss: 2.0254 - acc: 0.2829 - val_loss: 1.9364 - val_acc: 0.3307
Epoch 10/50
39/39 [==============================] - 262s 7s/step - loss: 2.0056 - acc: 0.2804 - val_loss: 1.9934 - val_acc: 0.2773
Epoch 11/50
39/39 [==============================] - 256s 7s/step - loss: 1.9678 - acc: 0.3048 - val_loss: 1.9048 - val_acc: 0.3352
Epoch 12/50
39/39 [==============================] - 260s 7s/step - loss: 1.9634 - acc: 0.3109 - val_loss: 1.8758 - val_acc: 0.3568
```

Table 3: Detection results on **PASCAL VOC 2007 test set**. The detector is Fast R-CNN and VGG-16. Training data: "07": VOC 2007 trainval, "07+12": union set of VOC 2007 trainval and VOC 2012 trainval. For RPN, the train-time proposals for Fast R-CNN are 2000. †: this number was reported in [2]; using the repository provided by this paper, this result is higher (68.1).

method	# proposals	data	mAP (%)
SS	2000	07	66.9†
SS	2000	07+12	70.0
RPN+VGG, unshared	300	07	68.5
RPN+VGG, shared	300	07	69.9
RPN+VGG, shared	300	07+12	**73.2**
RPN+VGG, shared	300	COCO+07+12	**78.8**

Table 4: Detection results on **PASCAL VOC 2012 test set**. The detector is Fast R-CNN and VGG-16. Training data: "07": VOC 2007 trainval, "07++12": union set of VOC 2007 trainval+test and VOC 2012 trainval. For RPN, the train-time proposals for Fast R-CNN are 2000. †: http://host.robots.ox.ac.uk:8080/anonymous/HZJTQA.html. ‡: http://host.robots.ox.ac.uk:8080/anonymous/YNPLXB.html. §: http://host.robots.ox.ac.uk:8080/anonymous/XEDH10.html.

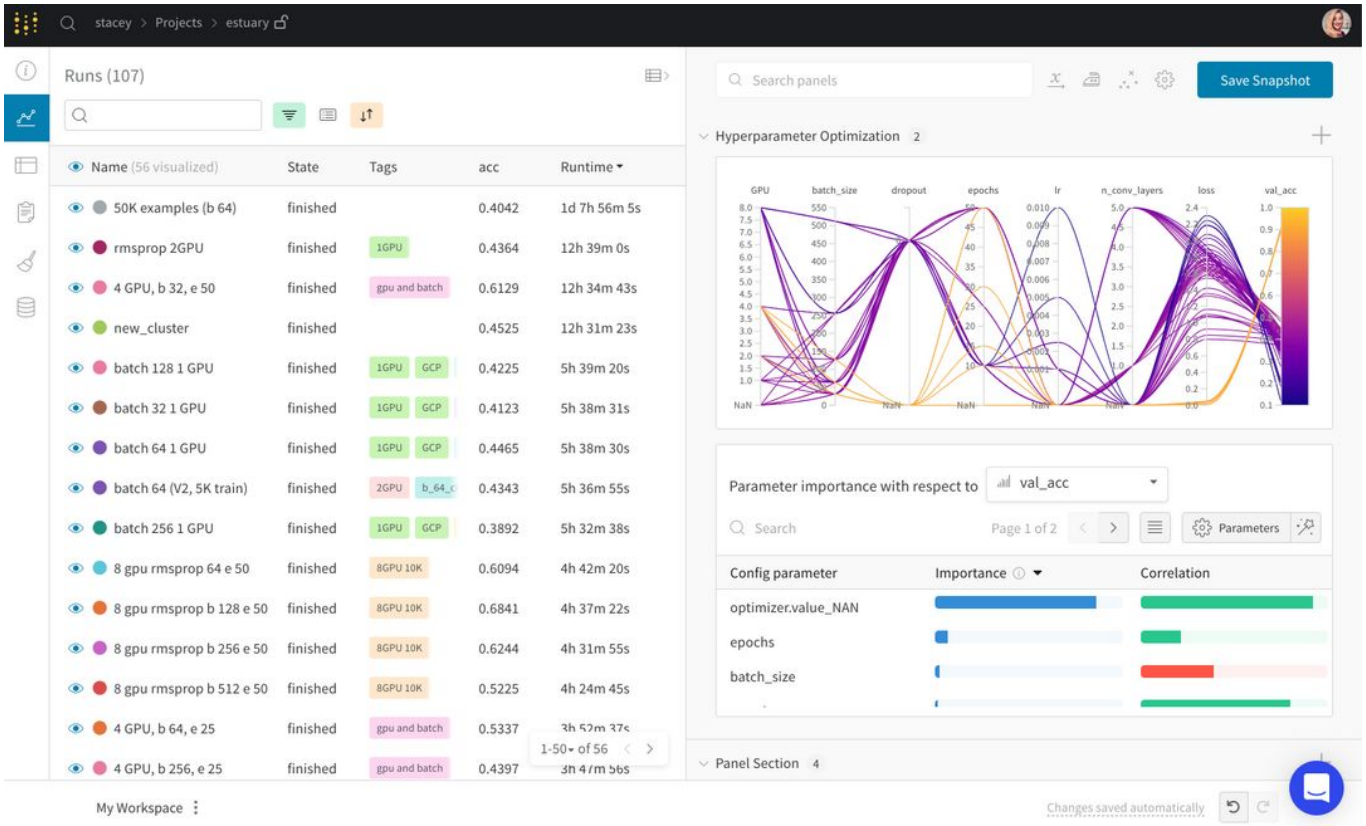
method	# proposals	data	mAP (%)
SS	2000	12	65.7
SS	2000	07++12	68.4
RPN+VGG, shared†	300	12	67.0
RPN+VGG, shared‡	300	07++12	**70.4**
RPN+VGG, shared§	300	COCO+07++12	**75.9**

Table 5: **Timing** (ms) on a K40 GPU, except SS proposal is evaluated in a CPU. "Region-wise" includes NMS, pooling, fully-connected, and softmax layers. See our released code for the profiling of running time.

model	system	conv	proposal	region-wise	total	rate
VGG	SS + Fast R-CNN	146	1510	174	1830	0.5 fps
VGG	RPN + Fast R-CNN	141	**10**	47	**198**	**5 fps**
ZF	RPN + Fast R-CNN	31	**3**	25	**59**	**17 fps**

	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Two-stage methods							
Faster R-CNN+++ [5]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [8]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [6]	Inception-ResNet-v2 [21]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM [20]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	**52.1**
One-stage methods							
YOLOv2 [15]	DarkNet-19 [15]	21.6	44.0	19.2	5.0	22.4	35.5
SSD513 [11, 3]	ResNet-101-SSD	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513 [3]	ResNet-101-DSSD	33.2	53.3	35.2	13.0	35.4	51.1
RetinaNet [9]	ResNet-101-FPN	39.1	59.1	42.3	21.8	42.7	50.2
RetinaNet [9]	ResNeXt-101-FPN	**40.8**	**61.1**	**44.1**	**24.1**	**44.2**	51.2
YOLOv3 608 × 608	Darknet-53	33.0	57.9	34.4	18.3	35.4	41.9

Track experiments – After W&B

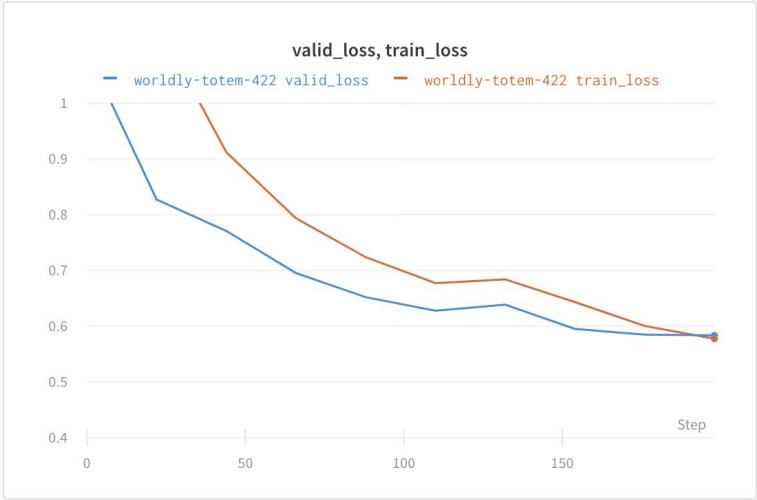


[Live Dashboard]

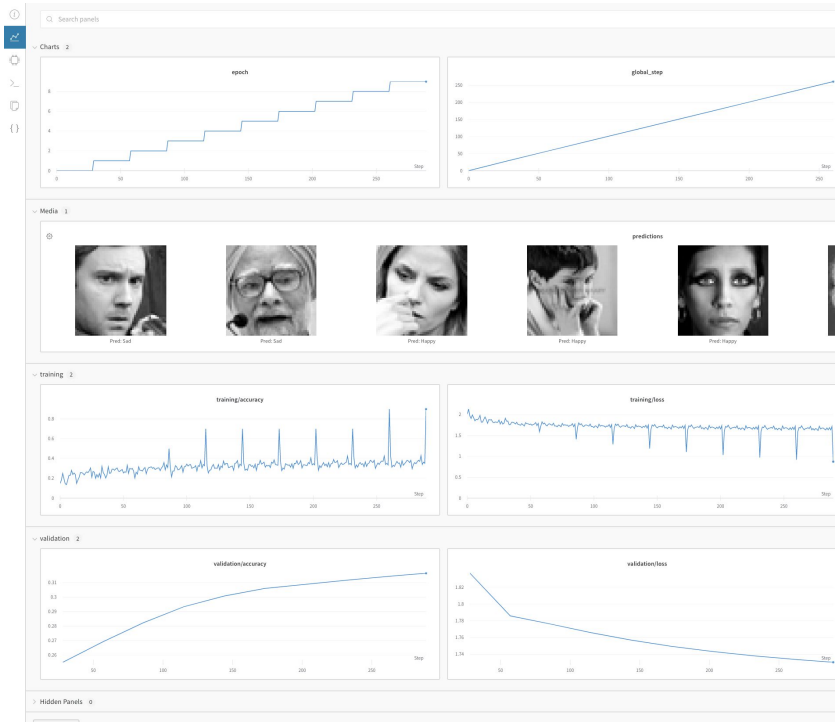
bit.ly/demo-run

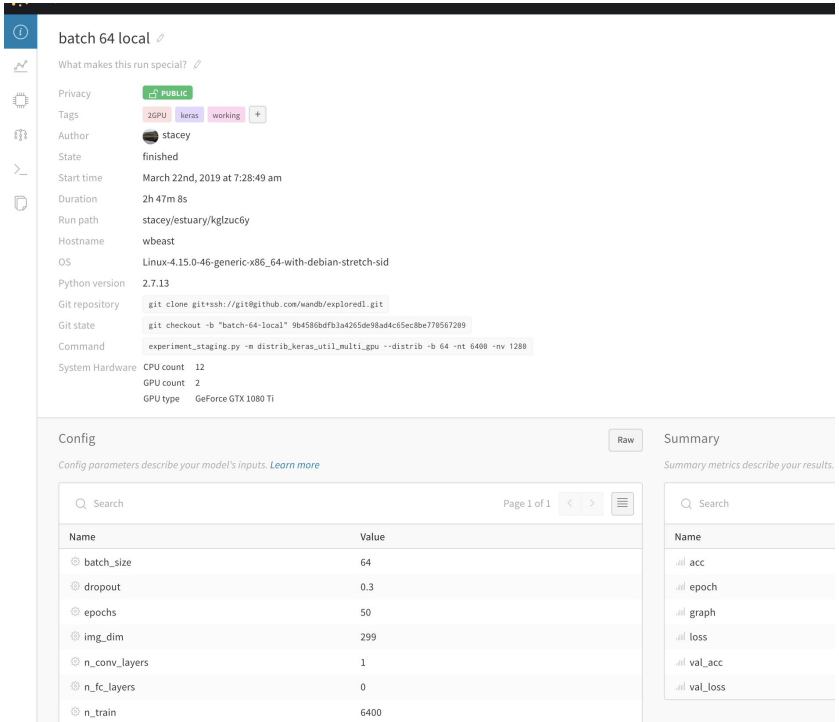
Reproduce a model

See live updates on model performance, check for overfitting, and visualize how a model performs on different classes.



Track experiments – After W&B

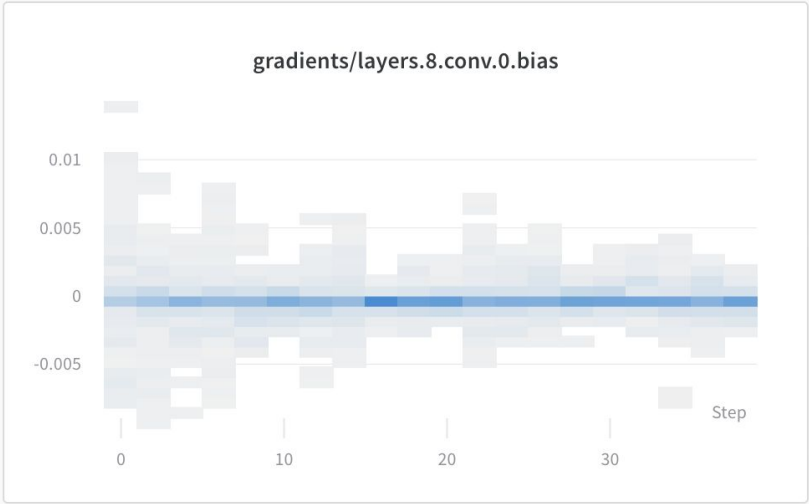


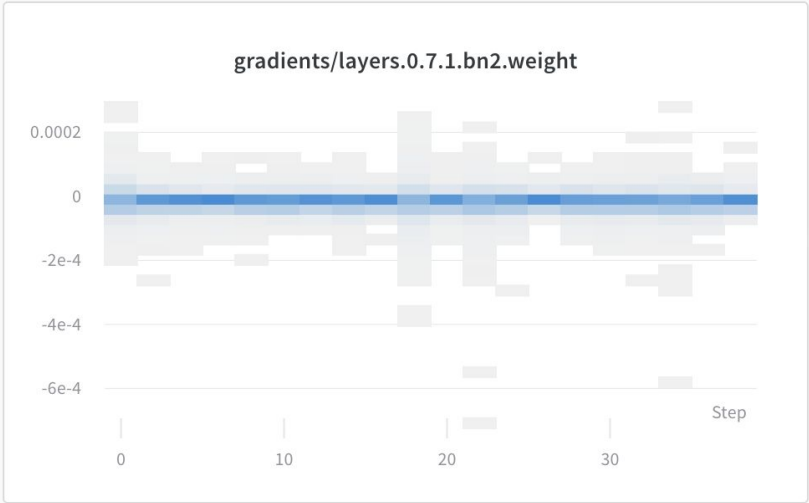


bit.ly/demo-run

Visualize gradients

See the inner workings of your model, look for convergence during training, and check for exploding gradients.

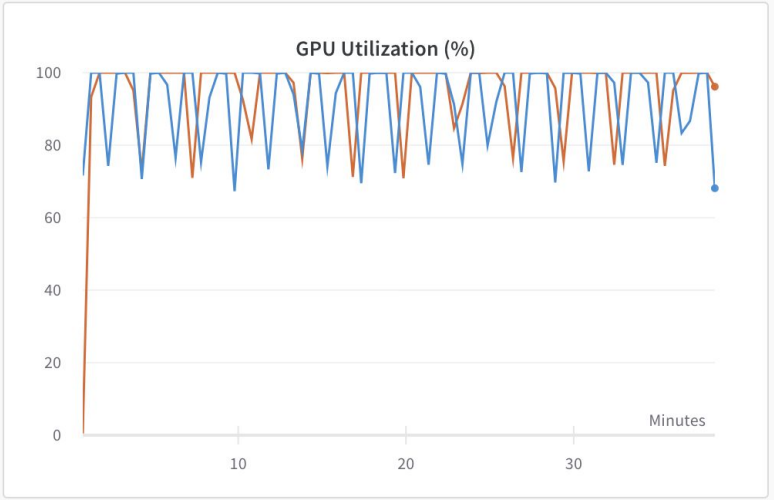


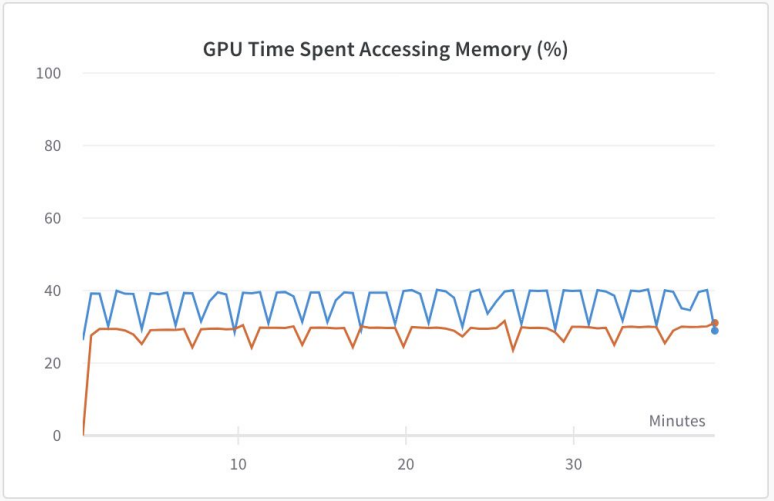


bit.ly/demo-run

System metrics

Get the most out of your GPUs and identify opportunities for optimizing hardware utilization.



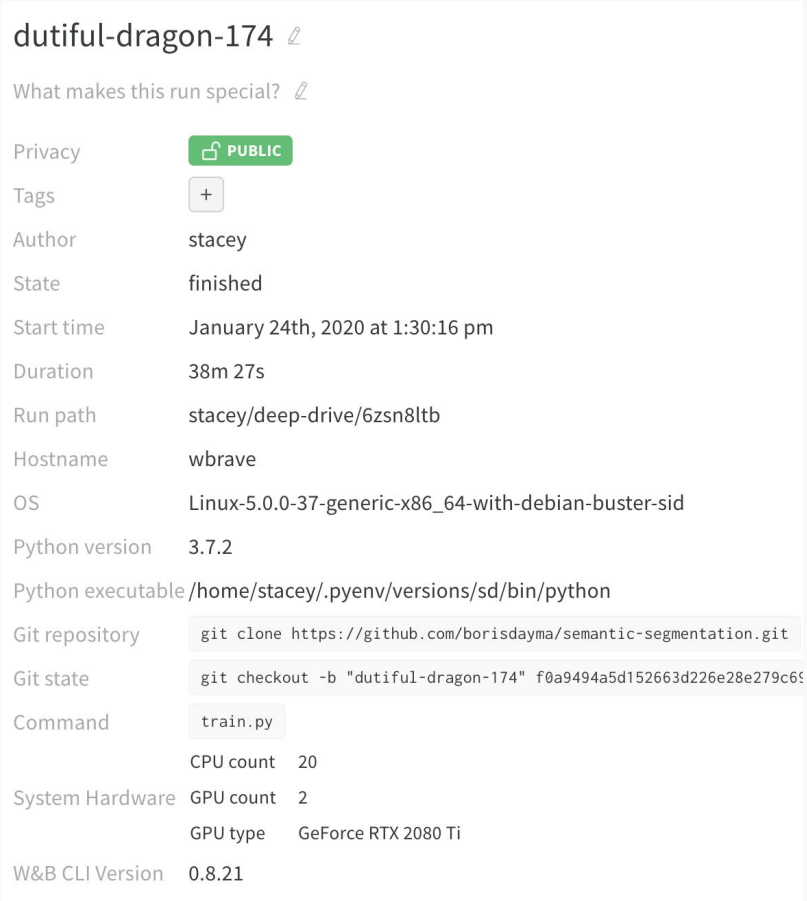


bit.ly/demo-run

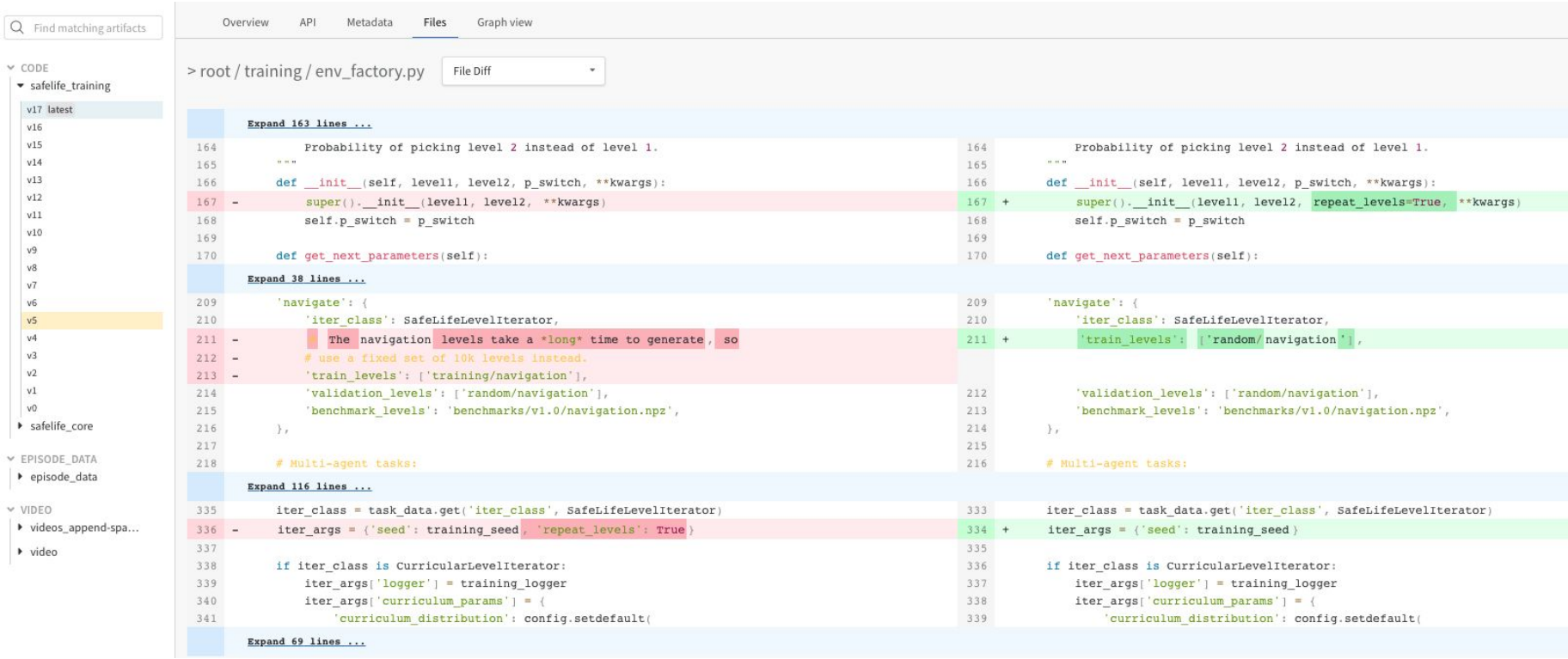
Capture the code

Save the most recent git commit, the command and args, and system hardware setup.

W&B also saves a patch file with uncommitted changes so you can reproduce the exact code that trained the model.



Code tab with versioning – spot changes



P.S. Store seeds & randomization settings in wandb.config

```python
def main(config):
    config.name = config_desc(config)
    if config.use_wandb:
        run.save()

    # set random seed
    tf.random.set_seed(config.random_seed)
    # also set numpy seed to control train/val dataset split
    np.random.seed(config.random_seed)
```

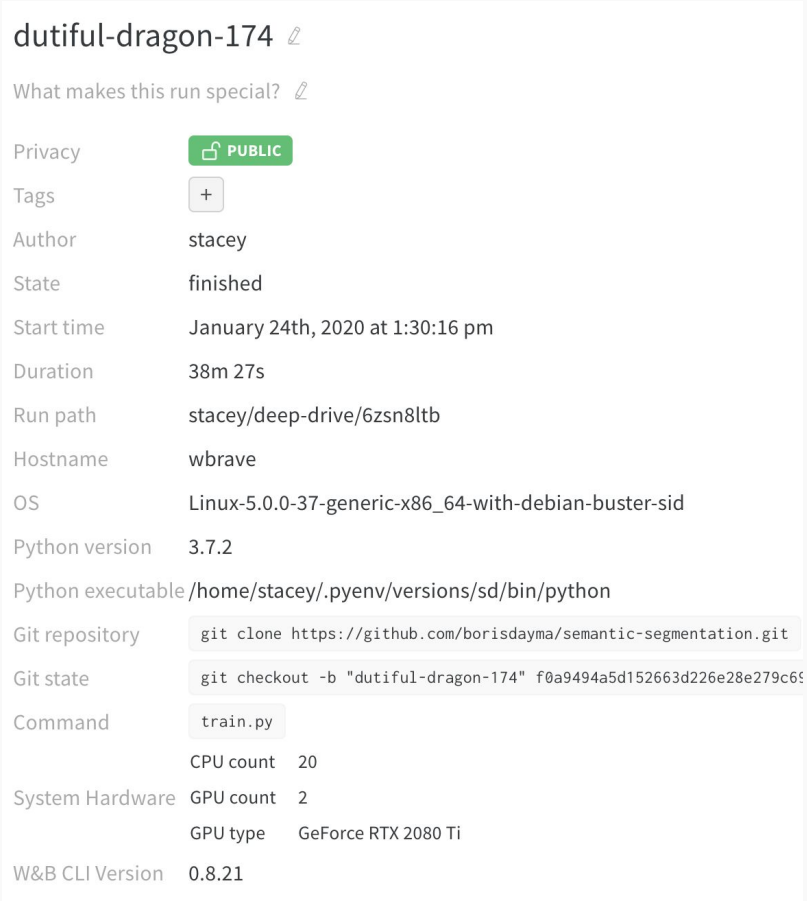
```python
def set_global_seed(config):
    from safelife.random import set_rng

    # Make sure the seed can be represented by floating point exactly.
    # This is just because we want to pass it over the web, and javascript
    # doesn't have 64 bit integers.
    if config.get('seed') is None:
        config['seed'] = np.random.randint(2**53)
    seed = np.random.SeedSequence(config['seed'])
    logger.info("SETTING GLOBAL SEED: %i", seed.entropy)
    set_rng(np.random.default_rng(seed))
    torch.manual_seed(seed.entropy & (2**31 - 1))

    if config['deterministic']:
        # Note that this may slow down performance
        # See https://pytorch.org/docs/stable/notes/randomness.html#cudnn
        torch.backends.cudnn.deterministic = True
```

Save each script run

- Run overview tab: see all the details
- Save the most recent git commit, the command and args, and system hardware setup (+ requirements.txt!)
- W&B also saves a patch file with uncommitted changes so you can reproduce the exact code that trained the model
- [Example run]



bit.ly/deep-drive

Persistent system of record

Query and filter across thousands of runs, and easily keep projects organized.

Runs (395)

3 Filters | Group | Sort | Tag | Move | Columns

Name (228 visualized)	Tags	Runtime	batch_size	encoder	learning_ra	num_train	num_valid	weight_dec	iou	train_loss	valid_loss	acc	traffic_acc	road
best car acc (50% data)	seg_masks	47m 52s	6	resnet34	0.001311	3524	492	0.08173	0.7997	0.5375	0.4427	0.8823	0.8664	0.93
best traffic acc (50% data))	seg_masks	46m 42s	8	resnet18	0.001	3523	492	0.097	0.8073	0.4919	0.4203	0.888	0.8718	0.94
best human iou (50% data)	seg_masks	31m 34s	7	alexnet	0.0009084	1405	190	0.097	0.716	0.6222	0.6259	0.8334	0.8042	0.9
best overall IOU (20% data)	seg_masks	20m 23s	7	resnet34	0.001367	1376	205	0.06731	0.7948	0.5096	0.4659	0.8726	0.8593	0.93
vibrant-cherry-426		6m 12s	8	resnet34	0.001	347	42	0.097	0.752	0.7114	0.6055	0.8474	0.8282	0.95
worldly-totem-422		12m 54s	8	resnet34	0.001	682	97	0.097	0.7523	0.5774	0.5836	0.8566	0.8349	0.91
jumping-voice-421		11m 59s	8	resnet34	0.001	725	92	0.097	0.7449	0.5633	0.5334	0.8504	0.8296	0.91
logical-energy-420	test_only	2m 14s	8	resnet34	0.001	66	10	0.097	0.4297	1.459	1.221	0.626	0.5958	0.76

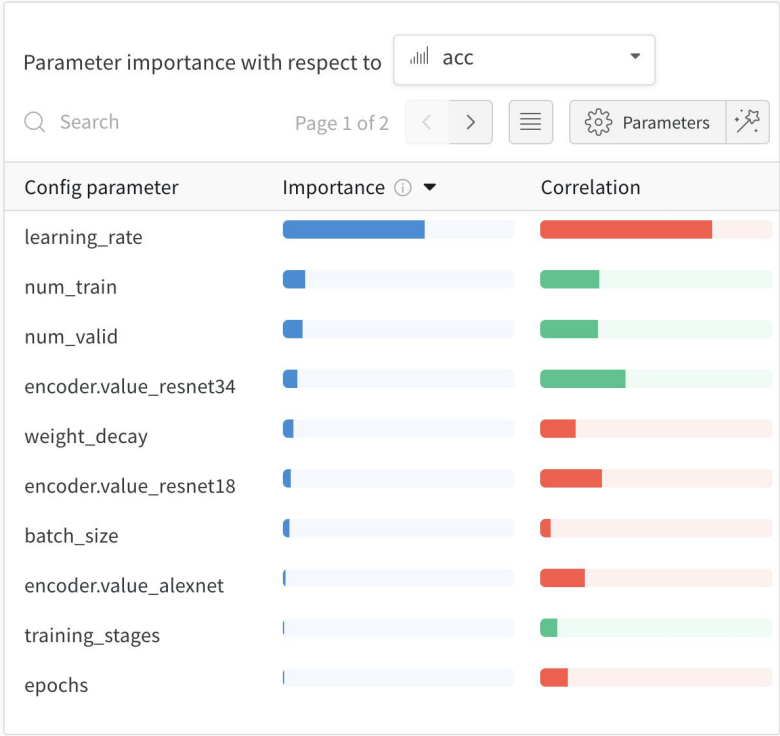
Understand models through interactive visualizations

bit.ly/deep-drive

Compare runs

Visualize the relationships between hyperparameters and model metrics.

Explore the space of possible models quickly, without getting bogged down setting up manual visualizations.

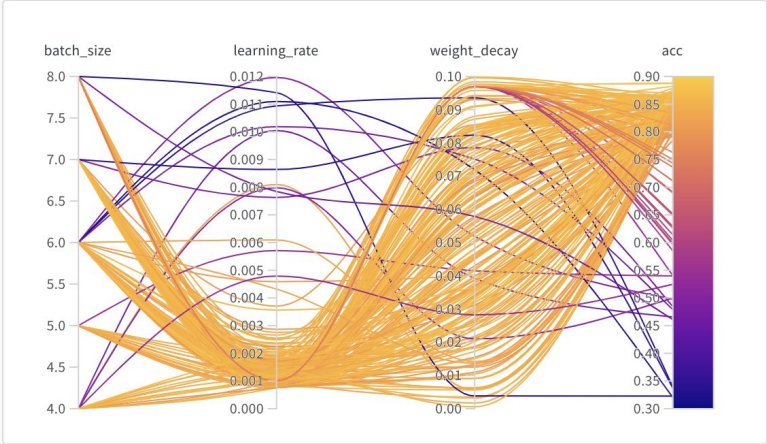


bit.ly/deep-drive

Iterate quickly
Accelerate time to insight

Use the interactive dashboard to spot issues in real time.

Stop underperforming runs early to optimize resource utilization.

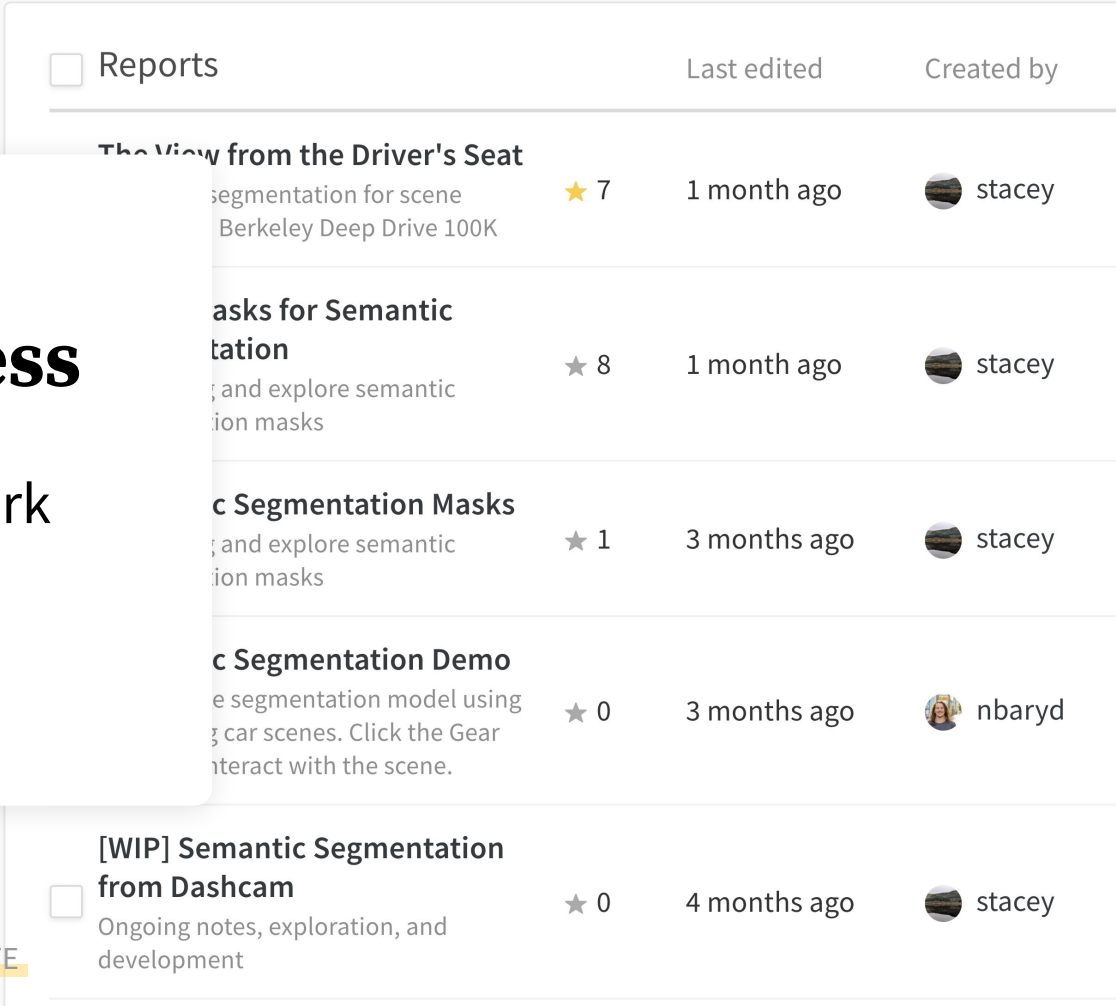


bit.ly/deep-drive-report

REPORTS

Annotate progress

Use reports to keep a work log and quickly pick up where you left off.



INSTALL • TRACK • COMPARE • OPTIMIZE • COLLABORATE

bit.ly/deep-drive-report

REPORTS

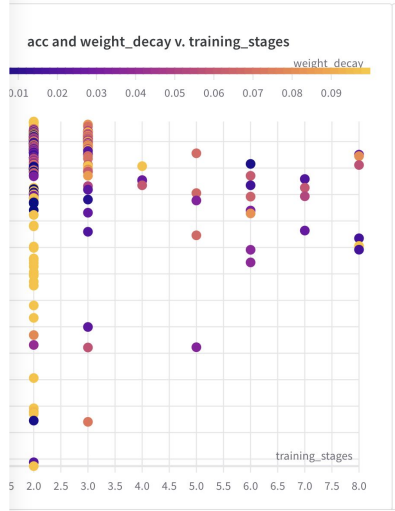
Collaborate easily

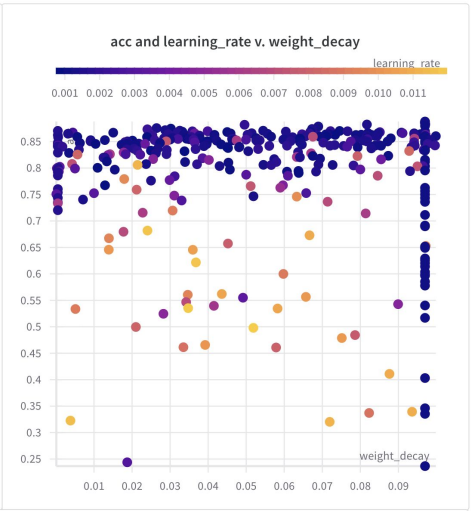
Give team members access to your exploratory results, sharing the context they need to quickly build on your work.

Weight decay: inconclusive

Initially increasing the weight decay 5X improved the accuracy by 9%. Increasing by 200X causes the same amount of improvement though.

erparameter Sweep Insights

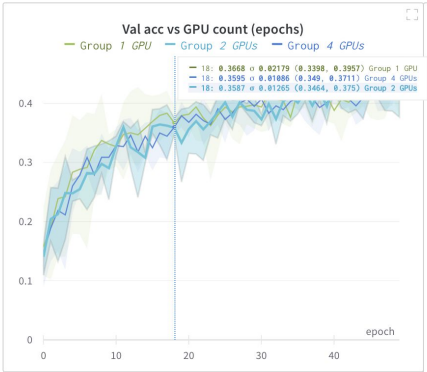


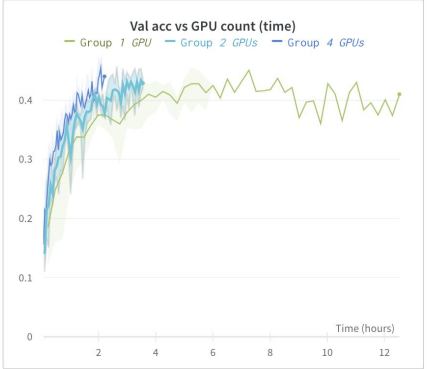


INSTALL • TRACK • COMPARE • OPTIMIZE • COLLABORATE

Sharing model insights with W&B

Scaling to 4 GPUs





Speed up: 4 GPUs: 2.5X, 2 GPUs: 1.6X

2.5X faster training on 4 GPUs vs 1 GPU

- model reaches a slightly higher validation accuracy 2.5X as fast when using 4 versus 1 GPU—this is the main advantage
- train/val acc/loss are not significantly affected by parallelizing the job across 1, 2, or 4 GPUs—this is expected and reassuring
- could continue tuning to improve relative speed-up—improvement is less than linear
- need better metrics (data throughput, batches per unit time, time to convergence) to quantify the added value of distributed training

Experiment

Train a 7-layer convnet on main iNaturalist dataset (5000 train / 800 val) as a proof of concept for the Keras multi_gpu_model function.

Notes

- initially no noticeable difference between 1 GPU and 2 GPUs—masked by one extremely slow run, batch 64_2, which stalled for 2 hours during training for unknown reasons. Leaving it out of the average shows that 2 GPUs yield a 1.6x acceleration
- accuracy vs batch size: consider effect of both batch size and number of GPUs
- for 2 GPUs, batch 64 > batch 32 / 128 > batch 256, but the effect is not super clear. 64 seems to be the optimal choice for batch size
- some combinations might be slower—resource sharing

5 lines of code

```
!pip install wandb
import wandb
wandb.init(project="classify_photos")

# save any experiment configuration
wandb.config = {"learning_rate" : 0.001, "epochs" : 10}

# define & train your model

# log any metric from your training script
wandb.log({"acc": accuracy, "val_acc": val_accuracy})
```

Quickstart docs docs.wandb.com

Understand datasets through interactive visualizations

dataset / pedestrians-at-night

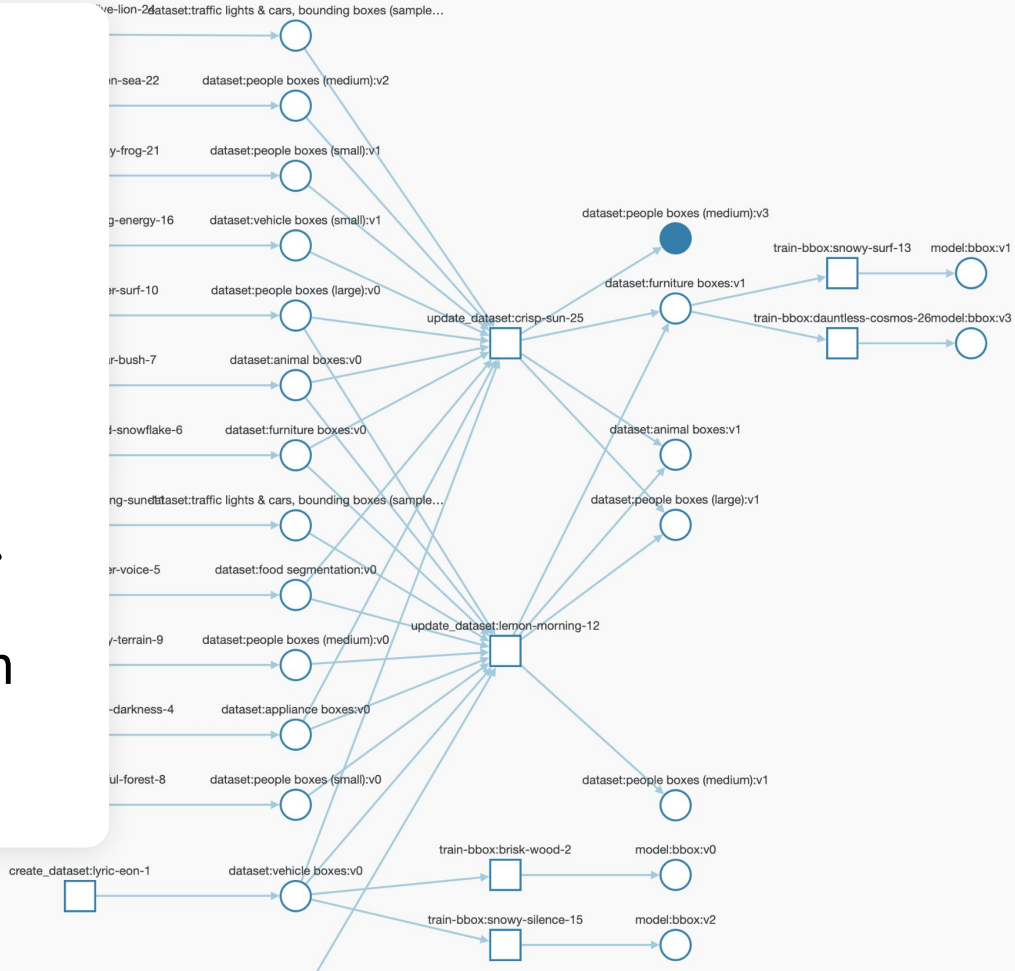
Overview API Metadata Files Graph view

Track artifacts

Get a bird's eye view of every step of model development.

Automatically checksum and version datasets and models.

Host your files anywhere with our infra-agnostic tracking.

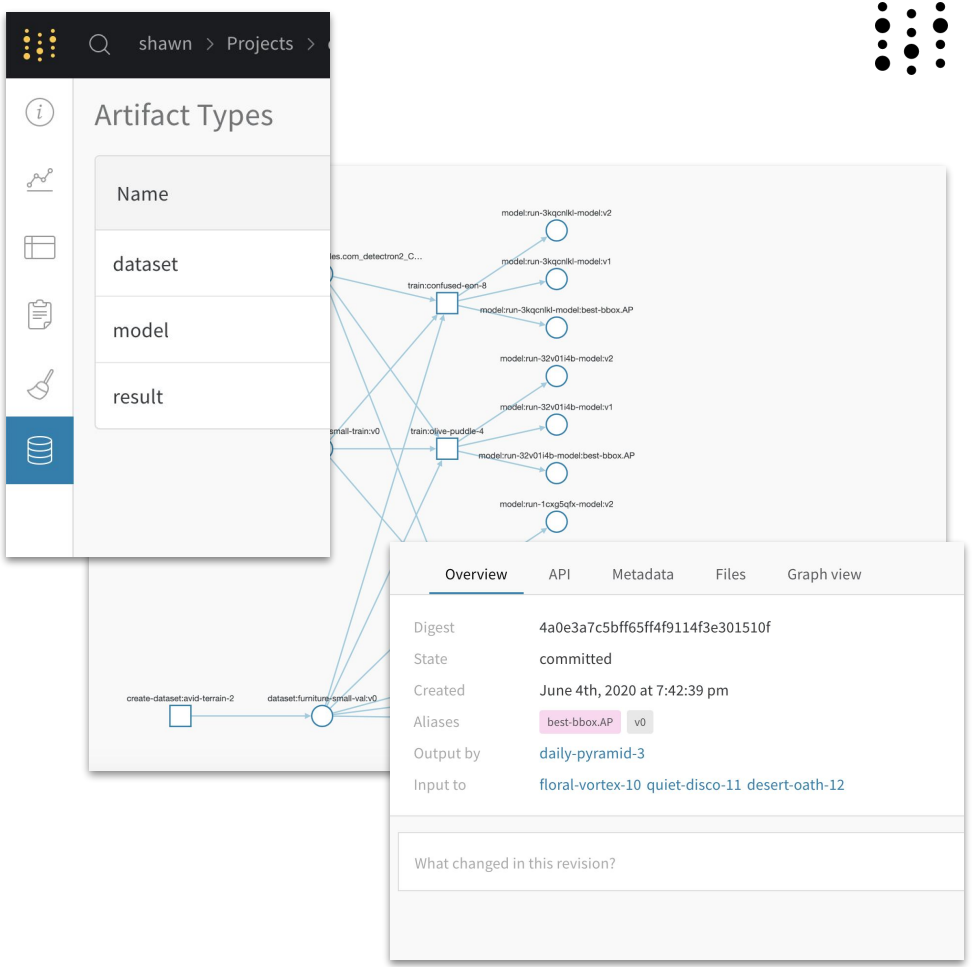


INSTALL • TRACK • COMPARE • OPTIMIZE • COLLABORATE

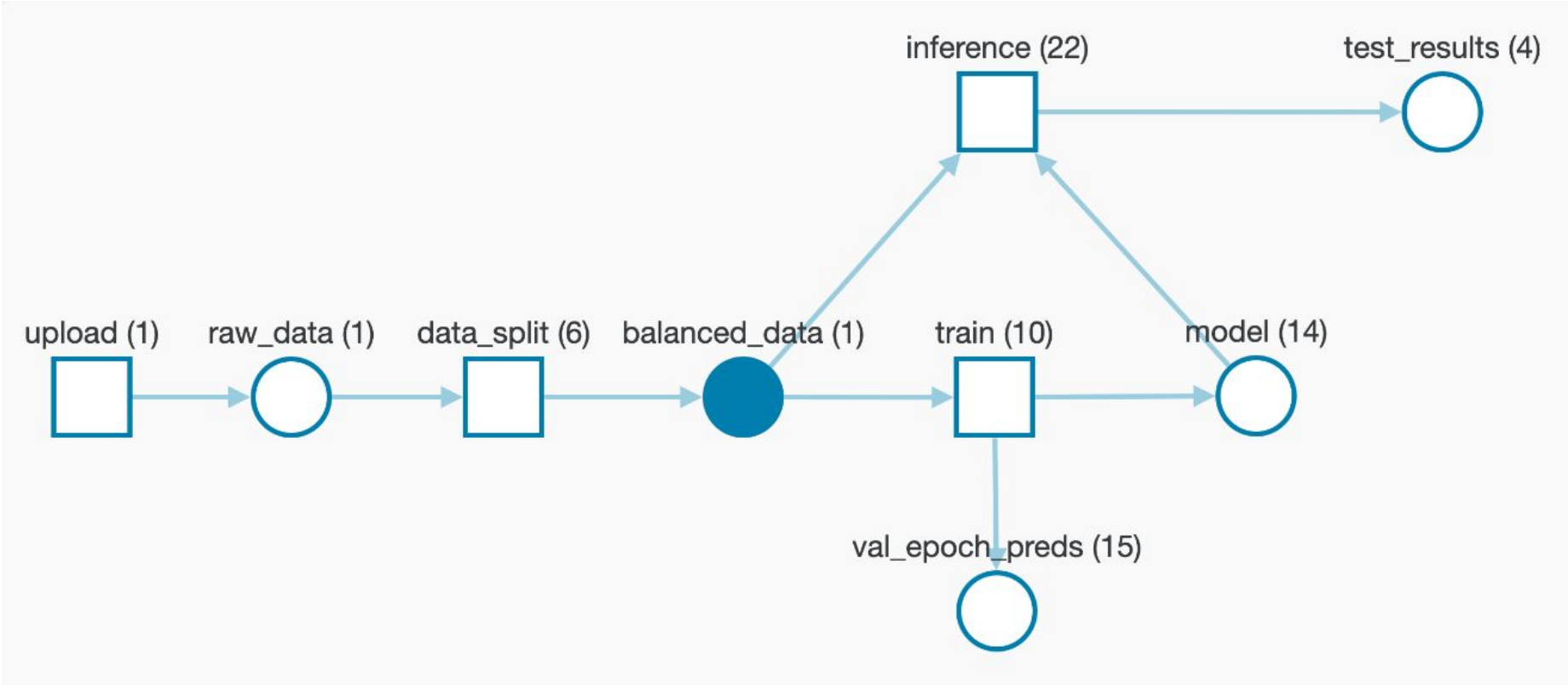
https://docs.wandb.com/artifacts

Artifacts

- Integrated with the project
  - Get a clear understanding of what is consumed by and generated from runs and projects
- Can be consumed by any run with access to the project
  - Artifacts as a serving agent for downstream systems
- Automatically generates lineage graphs

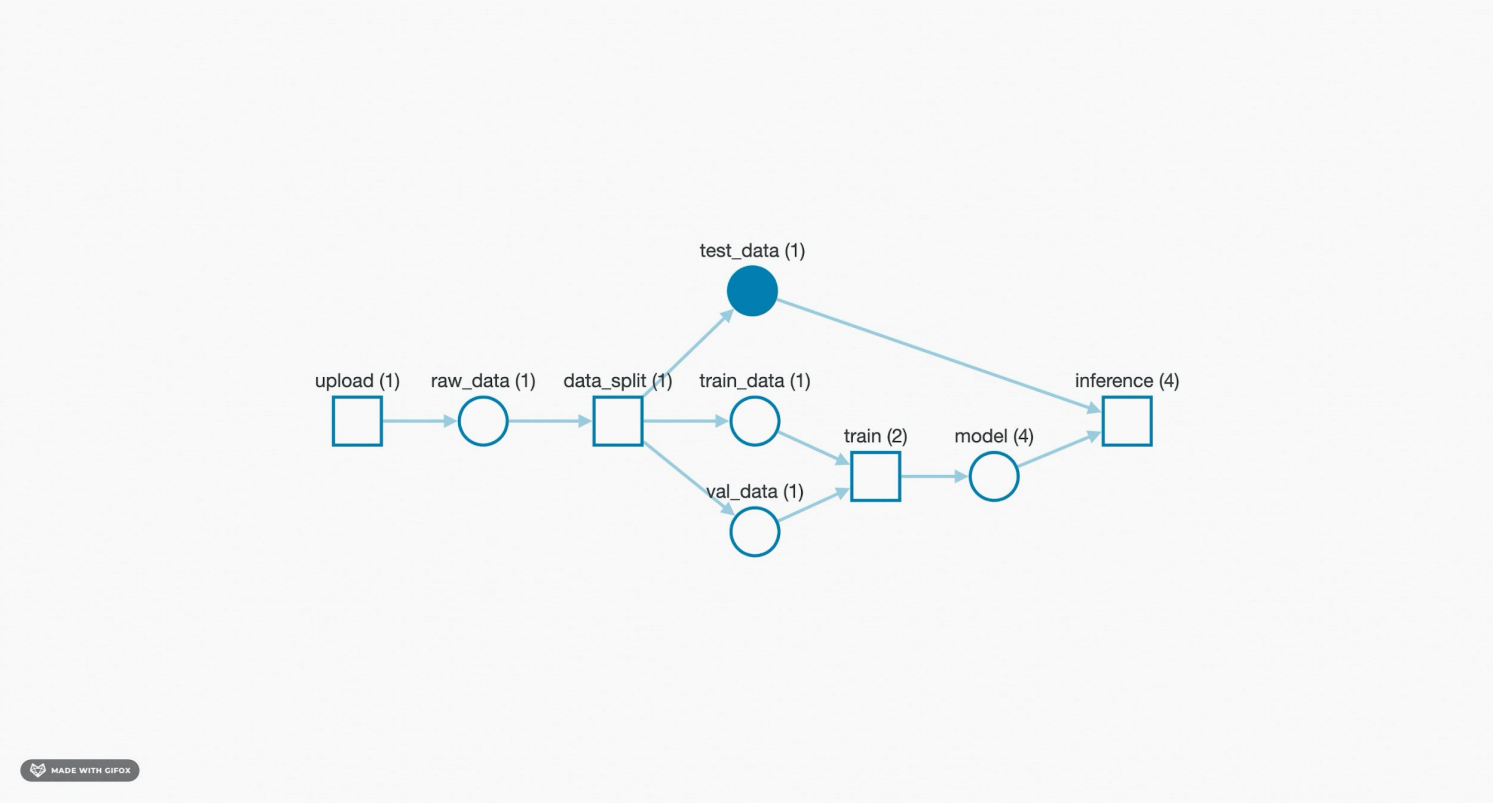


Version datasets, models & any files



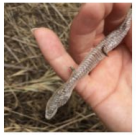
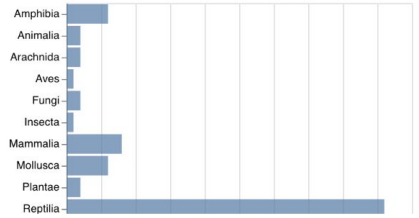
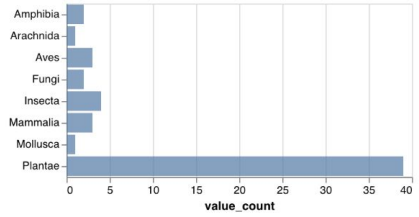
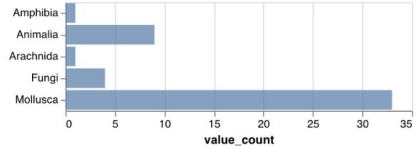
Artifacts Quickstart →

Organize, visualize, and share workflows



Example →

Images

Group by (guess)	image	truth
Reptilia	   1-3 of 76	
Plantae	   10-12 of 55	
Mollusca	   1-3 of 48	

Example →

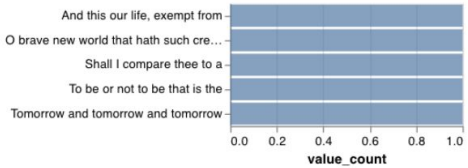
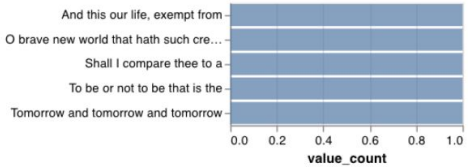
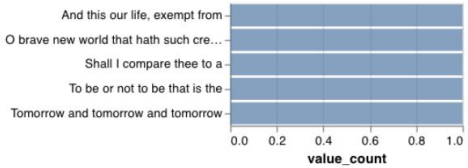
Report →

Segmentation masks

id	prediction	ground truth	overall IOU	false positive	false negative	iou_road
2b9bc4f8-780bd01f			0.6061	171737	213406	0.5211
134f6849-00000000			0.4365	350147	61326	0.8978
382e37b6-139b8d9f			0.4425	298525	206063	0.8833
83a3ef4b-9e72764d			0.7372	157810	113137	0.7578

Example →

Text

Group by (temperature)	prompt	response
0.3		To be or not to be that is the dearty to the brocaty of the persing stands and be is the beard to the sease that that the last may 1-3 of 5
0.6		To be or not to be that is the his visitition of prose, and may make to his tree the best kinous and it be herart. MERCUTIO: Why s 1-3 of 5
0.8		To be or not to be that is the see and hence not it aradements, the stands all one. ATBON: As stay the strange and I am before gon 1-3 of 5

Example →

Report →

Videos

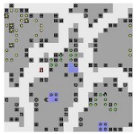
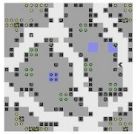
Type: video

- videos_append-spawn
  - v1 latest
  - v0
- video

Overview | API | Metadata | Files | Graph view

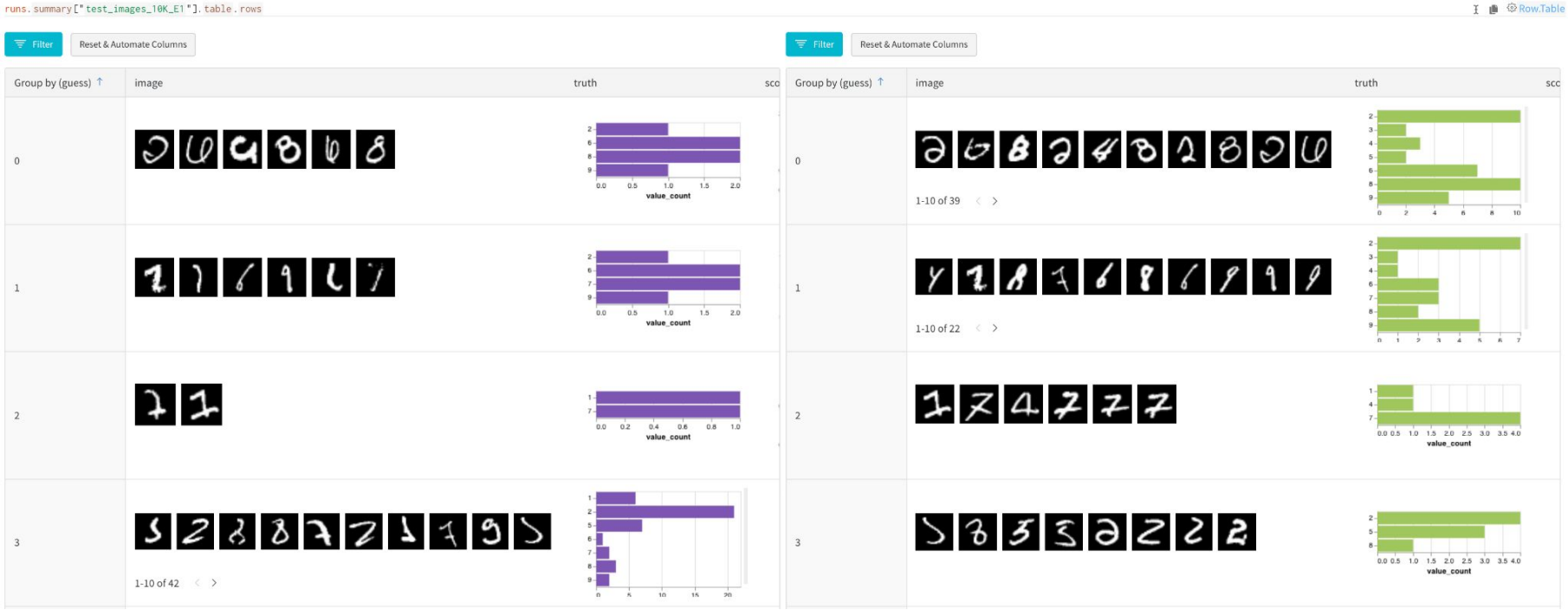
Filter

1-10 of 40002

video	side_effects	length	reward	success	score ↓	steps	episodes
	0	50	0.9706	0	96.544	1279994	25597
	0	50	0.9706	0	96.544	1561417	31228
	0	50	0.9706	0	96.544	1893888	37877

Example →

Interactive exploration & comparison



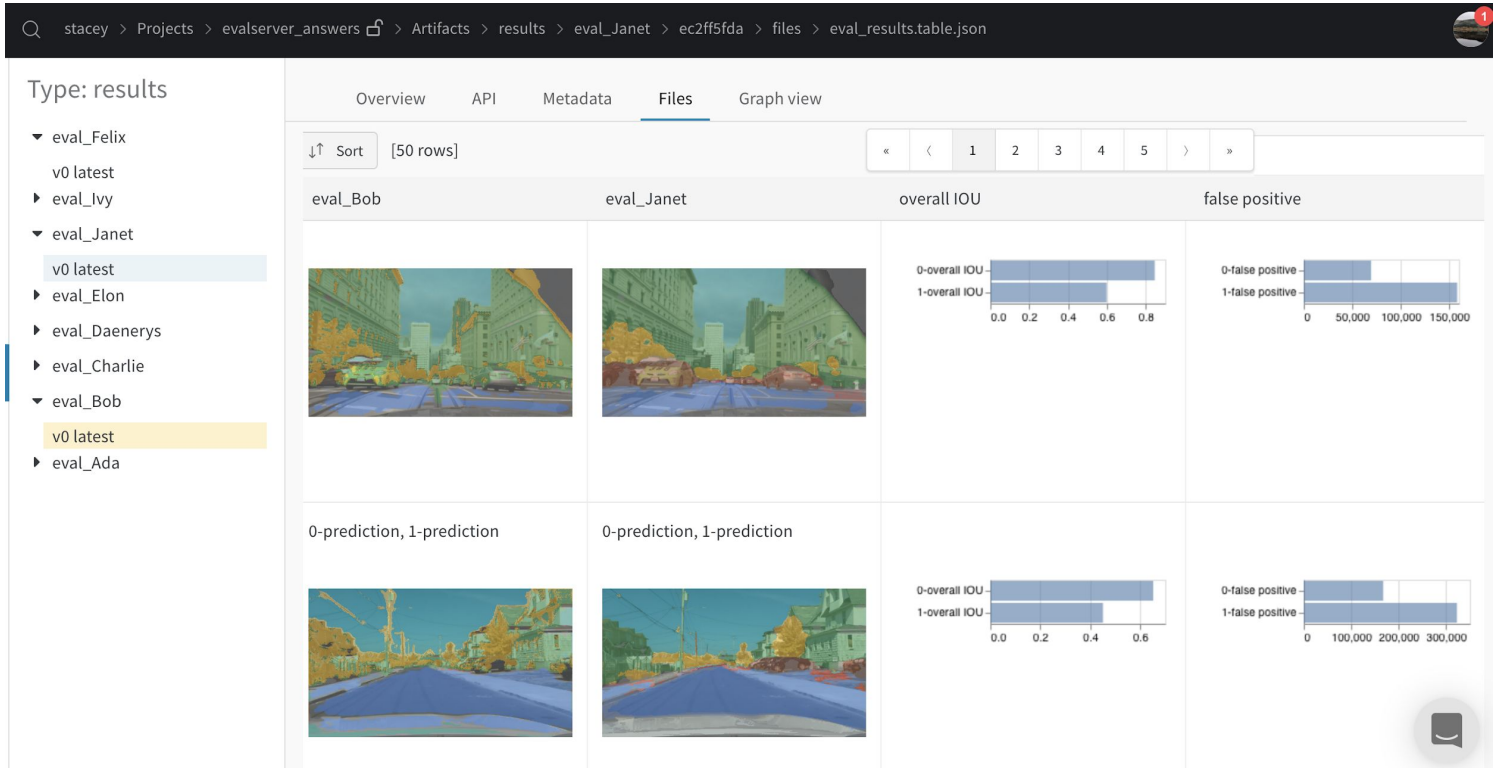
[Single table →]

[Comparison →]

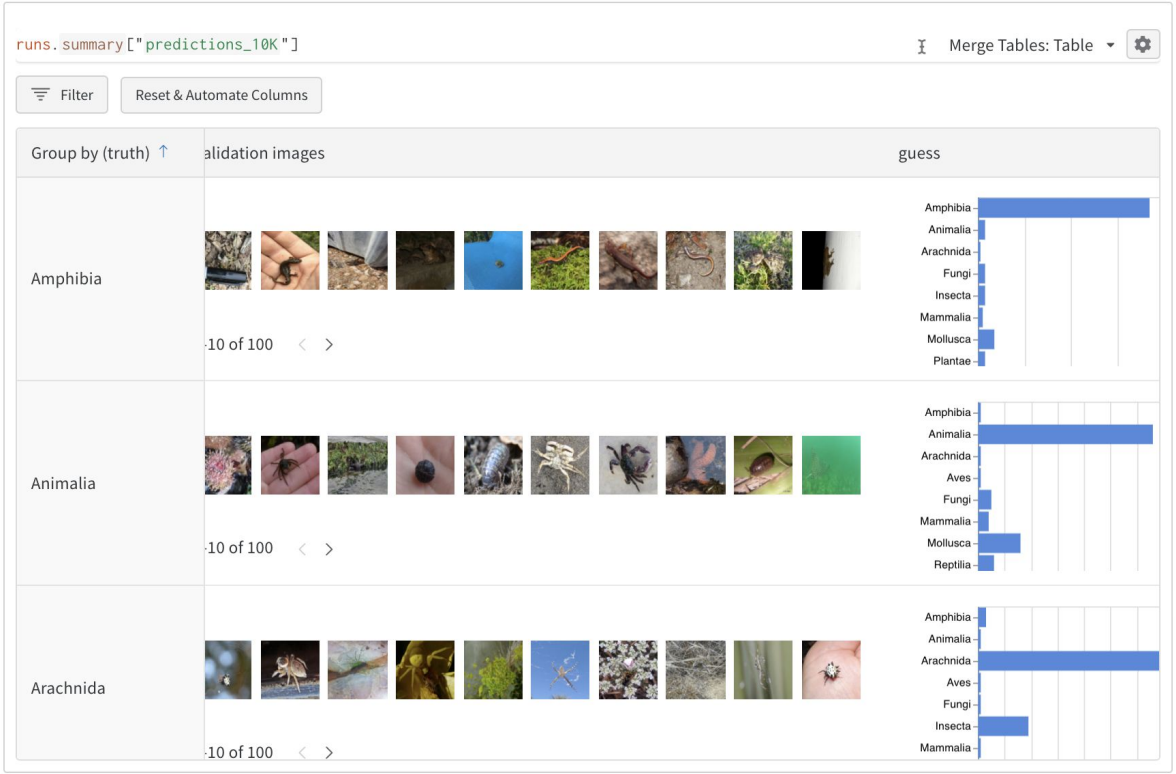
Tables Use Cases

- Log interactive media
- Manage data splits & iteratively refine datasets
- Visualize predictions
- Explore dynamically: filter, sort, group, add columns, & more
- Compare model variants
- Save Tables to workspace, report, etc.

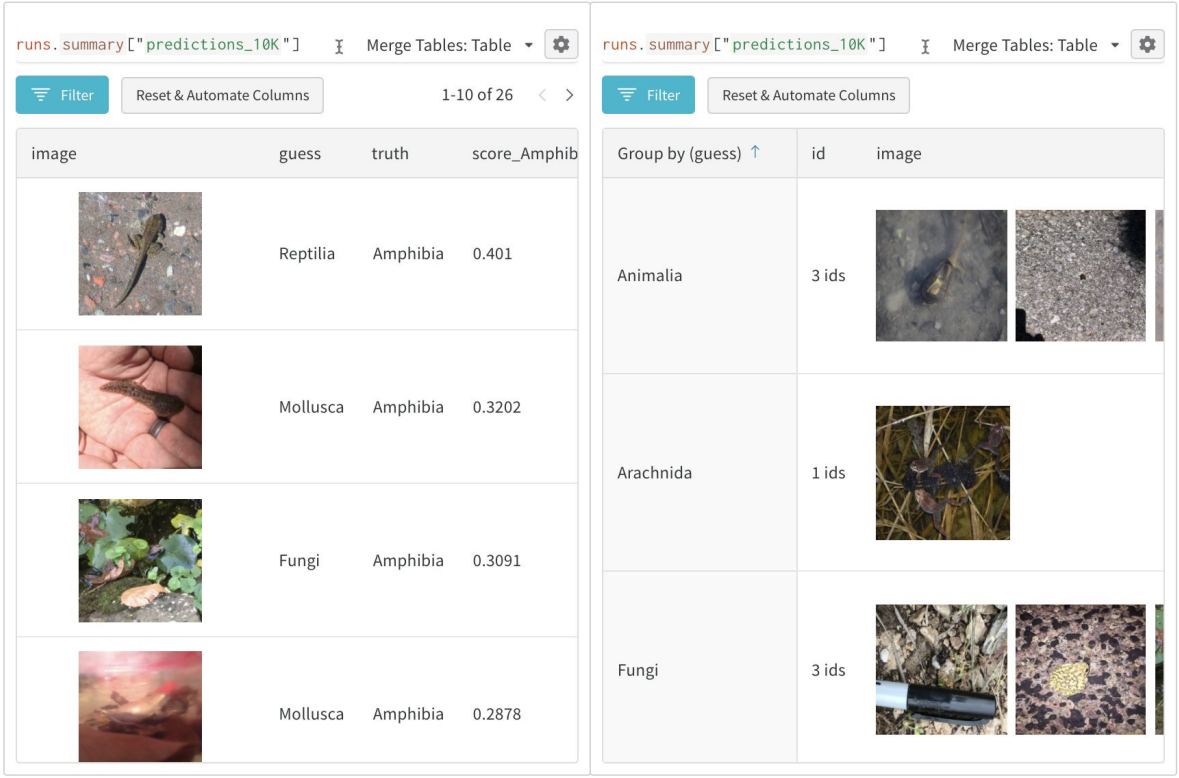
Example: why are two models different?



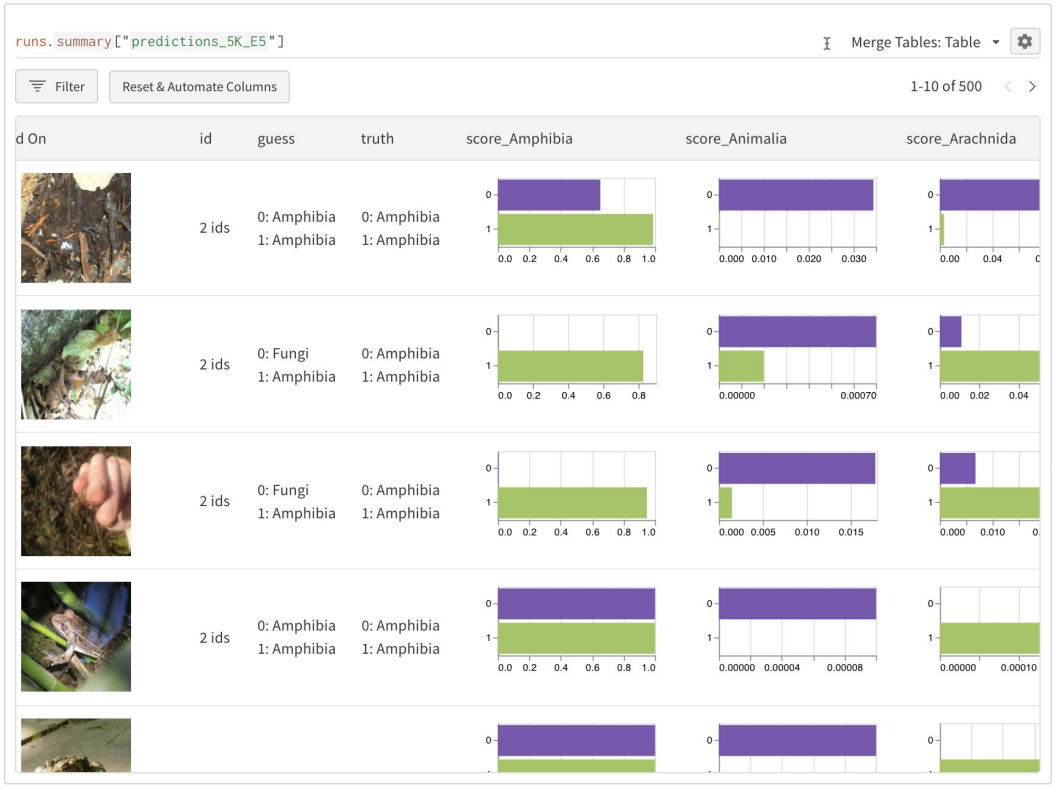
Spot classes that confound the model



Spot images that confound the model



Compare performance across models



Questions?

Docs	docs.wandb.com
Community	community.wandb.ai/
Questions?	twitter.com/lavanyaai
	lavanya@wandb.com

Thank You!

Machine Learning Systems Design

Next class: Monitoring Tutorials