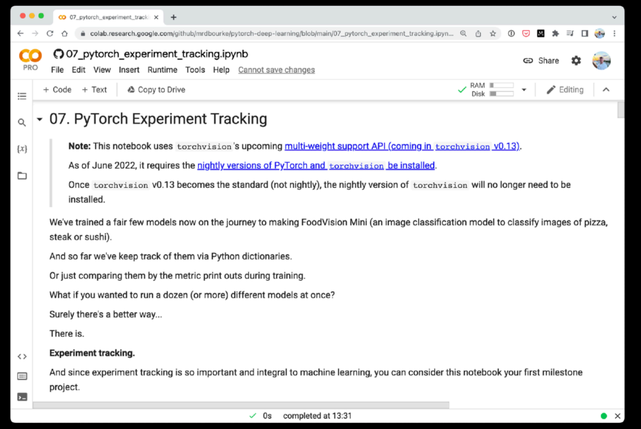
*Milestone Project 1*

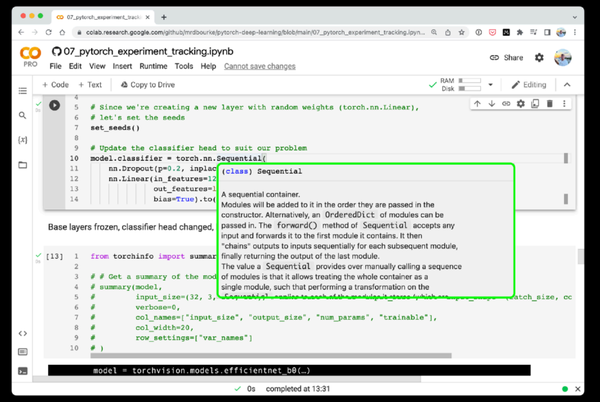
# Experiment Tracking with

# Where can you get help?

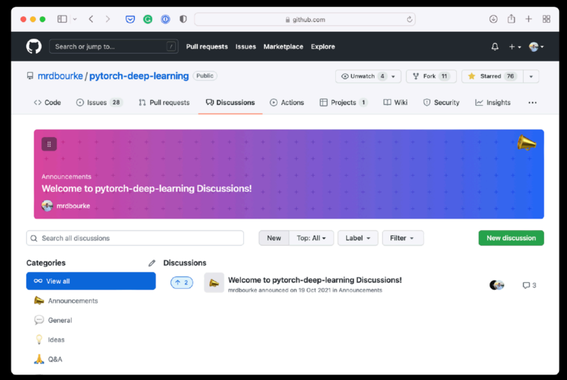
- Follow along with the code
- Try it for yourself
- Press SHIFT + CMD + SPACE to read the docstring
- Search for it
- Try again
- Ask

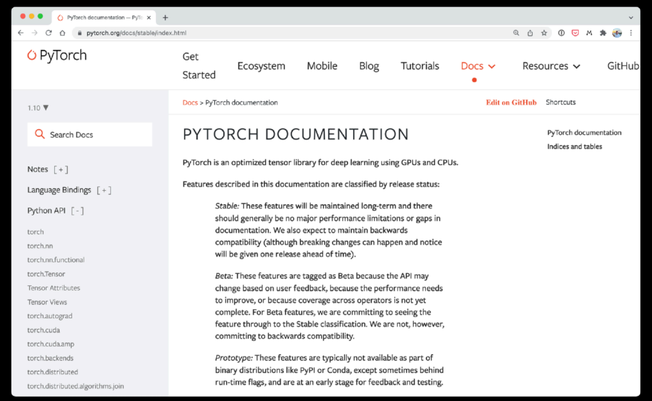



"If in doubt, run the code"








https://www.github.com/mrdbourke/pytorch-deep-learning/discussions

# “What is experiment tracking?”

How do I know which of my models has done the best...?

# How to approach this course

```python
# 1. Construct a model class that subclasses nn.Module
class CircleModelV0(nn.Module):
    def __init__(self):
        super().__init__()
        # 2. Create 2 nn.Linear layers
        self.layer_1 = nn.Linear(in_features=2, out_features=5)
        self.layer_2 = nn.Linear(in_features=5, out_features=1)

    # 3. Define a forward method containing the forward pass computation
    def forward(self, x):
        # Pass the data through both layers
        return self.layer_2(self.layer_1(x))

# 4. Create an instance of the model and send it to target device
model_0 = CircleModelV0().to(device)
model_0
```

## 1. Code along

Motto #1: *if in doubt, run the code!*

Motto #2:
*Experiment, experiment, experiment!*

## 2. Explore and experiment

Motto #3:
*Visualize, visualize, visualize!*

## 3. Visualize what you don't understand

*(including the "dumb" ones)*

## 4. Ask questions

## 5. Do the exercises

## 6. Share your work

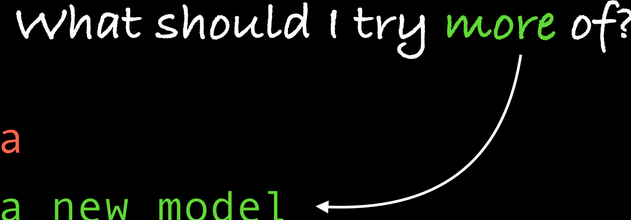
“Why track experiments?”

# Experiment tracking can quickly get out of hand...

```
 0 results
 1 model_1_results
 2 model_2_results
 3 model_1_results_v2
 4 model_2_results_v2
 5 model_2_results_v3
 6 model_2_results_v3
 7 2022-07-07-model_3_results_v1
 8 2022-07-07-model_3_results_v1_experiment_5
 9 2022-07-07-model_3_results_v2_big_model_25_epochs
10 2022-07-07-model_4_results_v1_big_model_25_epochs_20_percent_data
11 2022-07-07-model_4_results_v1_big_model_25_epochs_20_percent_data_new_model
12 2022-07-07-model_4_results_v2_big_model_feature_extractor_30_epochs_20_percent_data_no_dro
13 2022-07-07-model_5_results_v1_bigger_model_feature_extractor_50_epochs_50_percent_data_no_
```

# How do I know which one of my models performs the best?

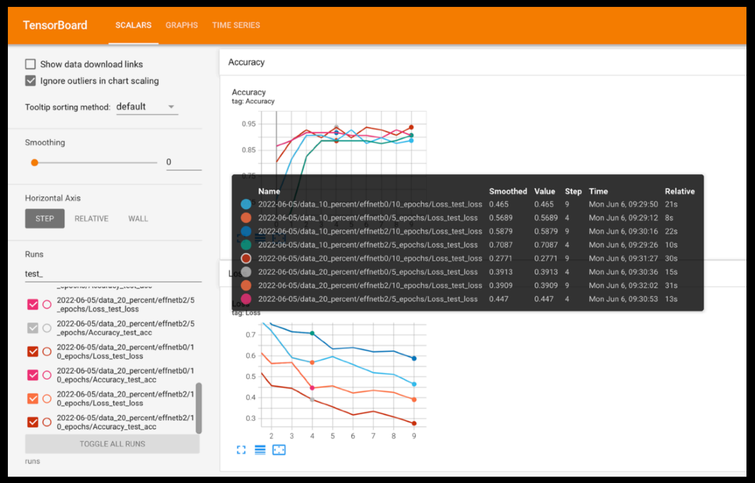
```
0  results
1  model_1_results
2  model_2_results
3  model_1_results_v2
4  model_2_results_v2
5  model_2_results_v3
6  model_2_results_v3
7  2022-07-07-model_3_results_v1
8  2022-07-07-model_3_results_v1_experiment_5
9  2022-07-07-model_3_results_v2_big_model_25_epochs
10 2022-07-07-model_4_results_v1_big_model_25_epochs_20_percent_data
11 2022-07-07-model_4_results_v1_big_model_25_epochs_20_percent_data_new_model
12 2022-07-07-model_4_results_v2_big_model_feature_extractor_30_epochs_20_percent_data_no_dro
13 2022-07-07-model_5_results_v1_bigger_model_feature_extractor_50_epochs_50_percent_data_no_
```

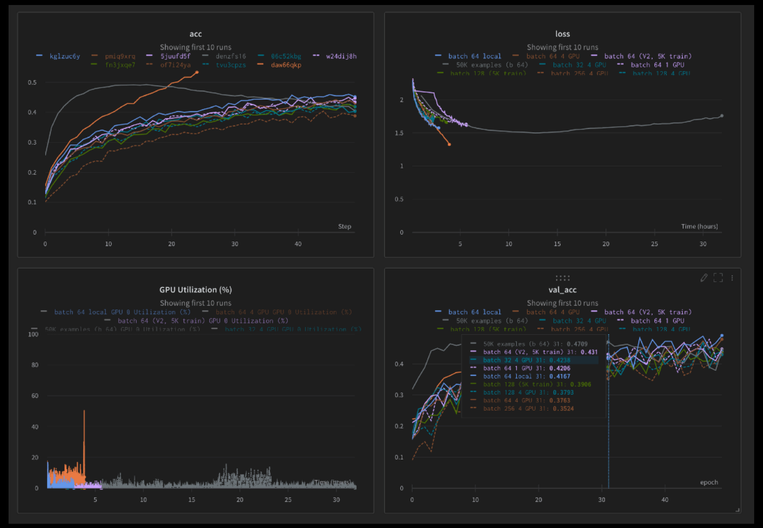
# Different ways to track experiments


Python dictionaries, CSV files, print outs.

Search: "track machine learning experiments"




**TensorBoard**
**Source:** https://www.tensorflow.org/tensorboard




**Weights & Biases**
**Source:** https://wandb.ai/site/experiment-tracking

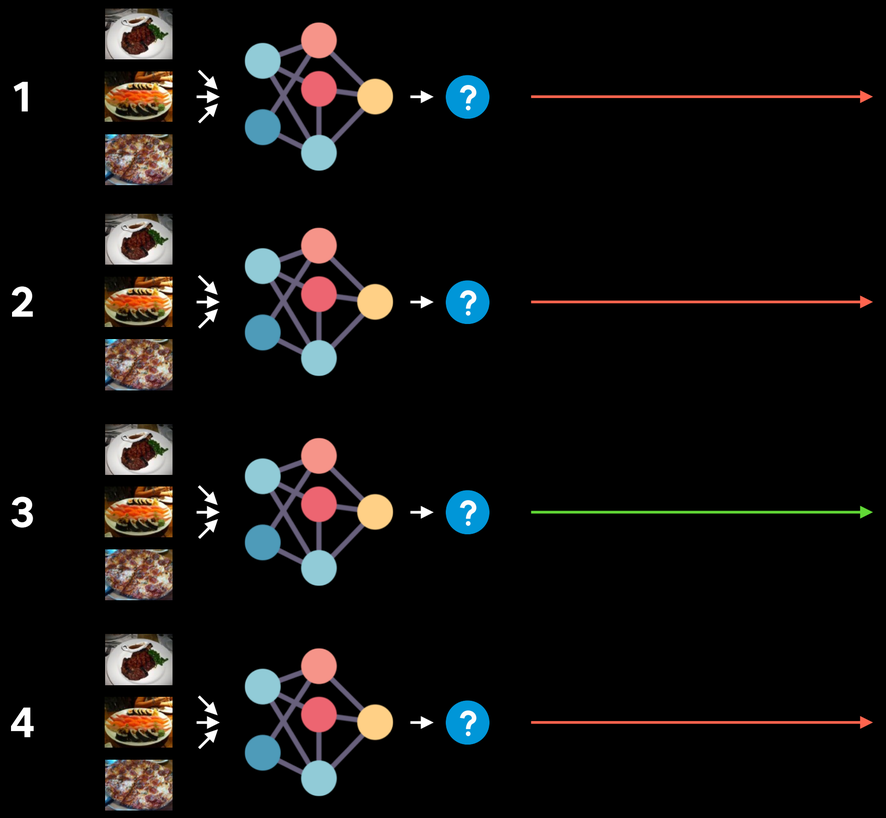
mlflow™

**MLfLow**
**Source:** https://mlflow.org/docs/latest/tracking.html

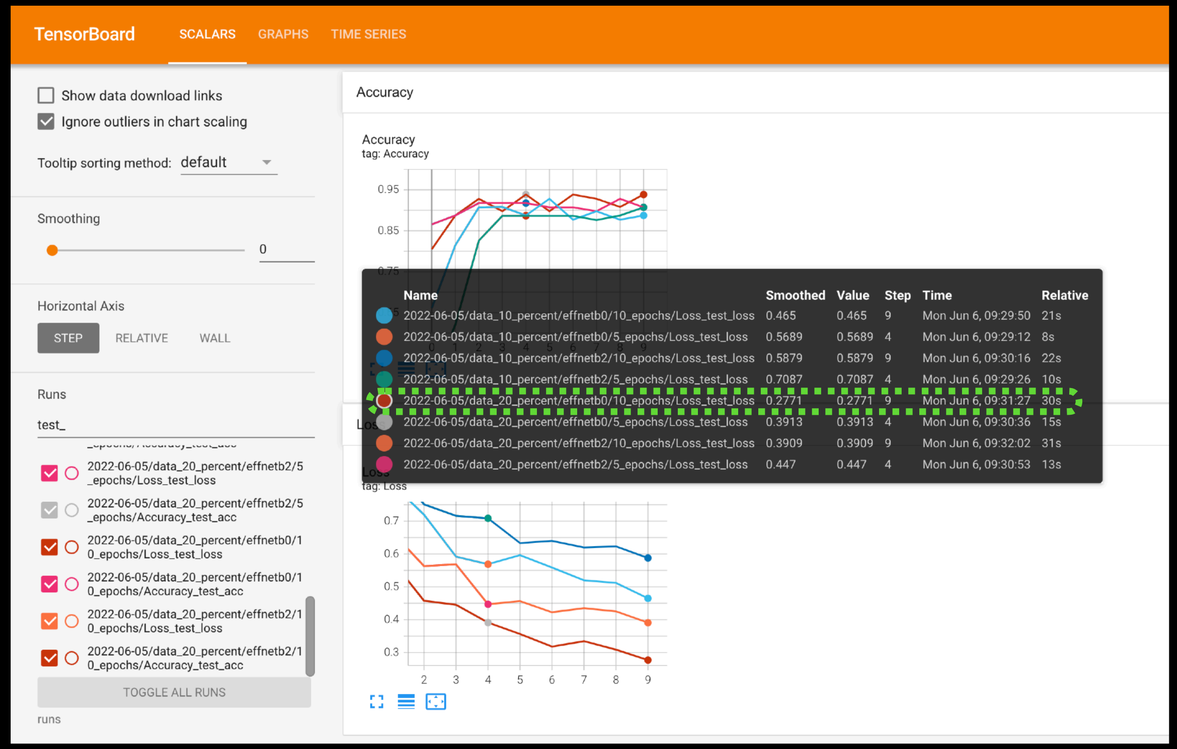
# What we're going to build

## FoodVision Mini Experiment Tracking

### Experiment




### Results tracked in TensorBoard



# What we're going to cover

(broadly)

- Getting setup (importing previously written code)
- Introduce experiment tracking with PyTorch
- Building several modelling experiments for FoodVision Mini 🍕🥩🍣
- Evaluating modelling experiments with TensorBoard
- Making predictions with the best performing model on custom data

(we'll be cooking up lots of code!)

**How:**

# Let’s code!

```
model = torchvision.models.efficientnet_b0(…)
torchinfo.summary(model, input_size=(32, 3, 224, 224))
```

```
==========================================================================================================================================
Layer (type (var_name))                                 Input Shape          Output Shape         Param #              Trainable
==========================================================================================================================================
EfficientNet                                            --                   --                   --                   Partial
├─Sequential (features)                                 [32, 3, 224, 224]    [32, 1280, 7, 7]     --                   False
│    └─ConvNormActivation (0)                           [32, 3, 224, 224]    [32, 32, 112, 112]   --                   False
│    │    └─Conv2d (0)                                  [32, 3, 224, 224]    [32, 32, 112, 112]   (864)                False
│    │    └─BatchNorm2d (1)                             [32, 32, 112, 112]   [32, 32, 112, 112]   (64)                 False
│    │    └─SiLU (2)                                    [32, 32, 112, 112]   [32, 32, 112, 112]   --                   --
│    └─Sequential (1)                                   [32, 32, 112, 112]   [32, 16, 112, 112]   --                   False
│    │    └─MBConv (0)                                  [32, 32, 112, 112]   [32, 16, 112, 112]   (1,448)              False
│    └─Sequential (2)                                   [32, 16, 112, 112]   [32, 24, 56, 56]     --                   False
│    │    └─MBConv (0)                                  [32, 16, 112, 112]   [32, 24, 56, 56]     (6,004)              False
│    │    └─MBConv (1)                                  [32, 24, 56, 56]     [32, 24, 56, 56]     (10,710)             False
│    └─Sequential (3)                                   [32, 24, 56, 56]     [32, 40, 28, 28]     --                   False
│    │    └─MBConv (0)                                  [32, 24, 56, 56]     [32, 40, 28, 28]     (15,350)             False
│    │    └─MBConv (1)                                  [32, 40, 28, 28]     [32, 40, 28, 28]     (31,290)             False
│    └─Sequential (4)                                   [32, 40, 28, 28]     [32, 80, 14, 14]     --                   False
│    │    └─MBConv (0)                                  [32, 40, 28, 28]     [32, 80, 14, 14]     (37,130)             False
│    │    └─MBConv (1)                                  [32, 80, 14, 14]     [32, 80, 14, 14]     (102,900)            False
│    │    └─MBConv (2)                                  [32, 80, 14, 14]     [32, 80, 14, 14]     (102,900)            False
│    └─Sequential (5)                                   [32, 80, 14, 14]     [32, 112, 14, 14]    --                   False
│    │    └─MBConv (0)                                  [32, 80, 14, 14]     [32, 112, 14, 14]    (126,004)            False
│    │    └─MBConv (1)                                  [32, 112, 14, 14]    [32, 112, 14, 14]    (208,572)            False
│    │    └─MBConv (2)                                  [32, 112, 14, 14]    [32, 112, 14, 14]    (208,572)            False
│    └─Sequential (6)                                   [32, 112, 14, 14]    [32, 192, 7, 7]      --                   False
│    │    └─MBConv (0)                                  [32, 112, 14, 14]    [32, 192, 7, 7]      (262,492)            False
│    │    └─MBConv (1)                                  [32, 192, 7, 7]      [32, 192, 7, 7]      (587,952)            False
│    │    └─MBConv (2)                                  [32, 192, 7, 7]      [32, 192, 7, 7]      (587,952)            False
│    │    └─MBConv (3)                                  [32, 192, 7, 7]      [32, 192, 7, 7]      (587,952)            False
│    └─Sequential (7)                                   [32, 192, 7, 7]      [32, 320, 7, 7]      --                   False
│    │    └─MBConv (0)                                  [32, 192, 7, 7]      [32, 320, 7, 7]      (717,232)            False
│    └─ConvNormActivation (8)                           [32, 320, 7, 7]      [32, 1280, 7, 7]     --                   False
│    │    └─Conv2d (0)                                  [32, 320, 7, 7]      [32, 1280, 7, 7]     (409,600)            False
│    │    └─BatchNorm2d (1)                             [32, 1280, 7, 7]     [32, 1280, 7, 7]     (2,560)              False
│    │    └─SiLU (2)                                    [32, 1280, 7, 7]     [32, 1280, 7, 7]     --                   --
├─AdaptiveAvgPool2d (avgpool)                           [32, 1280, 7, 7]     [32, 1280, 1, 1]     --                   --
├─Sequential (classifier)                               [32, 1280]           [32, 3]              --                   True
│    └─Dropout (0)                                      [32, 1280]           [32, 1280]           --                   --
│    └─Linear (1)                                       [32, 1280]           [32, 3]              3,843                True
==========================================================================================================================================
Total params: 4,011,391
Trainable params: 3,843
Non-trainable params: 4,007,548
Total mult-adds (G): 12.31
==========================================================================================
Input size (MB): 19.27
Forward/backward pass size (MB): 3452.09
Params size (MB): 16.05
Estimated Total Size (MB): 3487.41
==========================================================================================
```

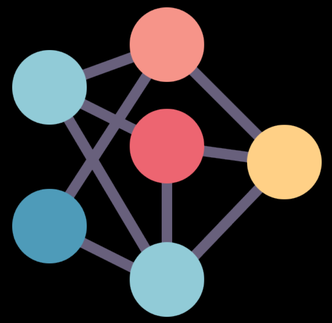
Many layers untrainable (frozen)

Only last layers are trainable

Final layer output (same as number of classes 🍕🥩🍣)

Less trainable parameters because many layers are frozen

# What experiments should you try?

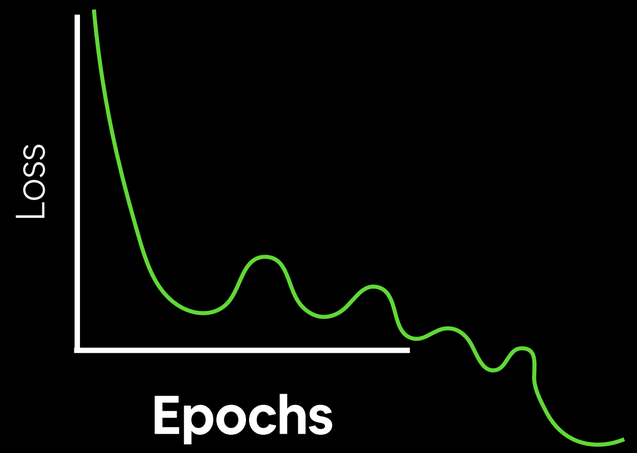


Model architecture

```
torch.nn.Linear(in_features=256, out_features=512)
torch.nn.Linear(in_features=512, out_features=1024)
```

Number of layers/
hidden units

Data augmentation




Amount of data

```
torch.optim.Adam(lr=0.001)
torch.optim.Adam(lr=0.0003)
torch.optim.lr_schedular()
```

Learning rate

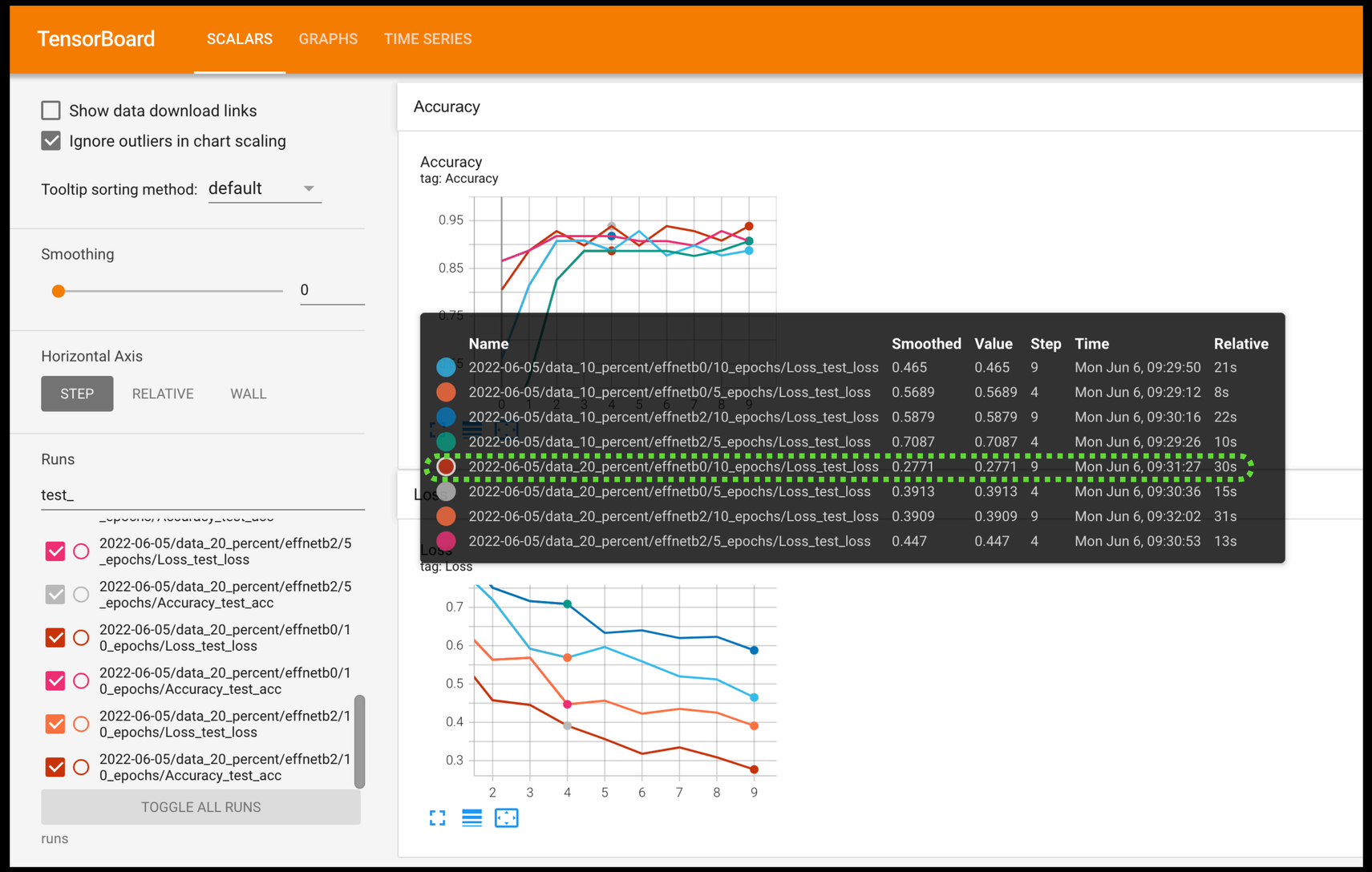
+ any hyperparameter you can think of...

# Experiments we're running

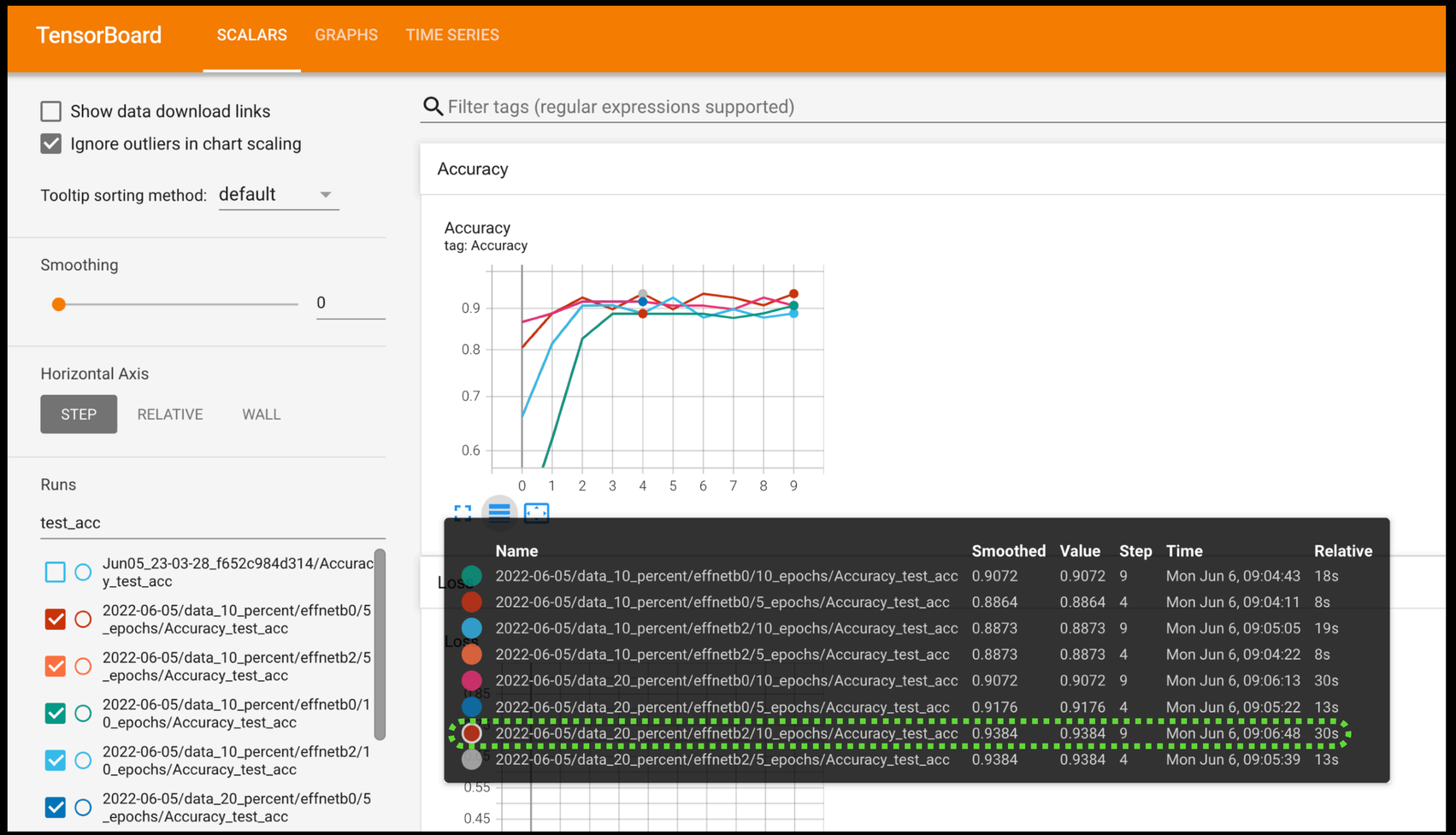
| Experiment number | Training data | Testing data | Model architecture | Number of epochs |
|---|---|---|---|---|
| 1 | Pizza, Steak Sushi **10% training data** | Pizza, Steak Sushi 10% testing data | EffNetB0 | 5 |
| 2 | Pizza, Steak Sushi **10% training data** | Same as 1 | EffNetB2 | 5 |
| 3 | Pizza, Steak Sushi **10% training data** | Same as 1 | EffNetB0 | 10 |
| 4 | Pizza, Steak Sushi **10% training data** | Same as 1 | EffNetB2 | 10 |
| 5 | Pizza, Steak Sushi **20% training data** | Same as 1 | EffNetB0 | 5 |
| 6 | Pizza, Steak Sushi **20% training data** | Same as 1 | EffNetB2 | 5 |
| 7 | Pizza, Steak Sushi **20% training data** | Same as 1 | EffNetB0 | 10 |
| 8 | Pizza, Steak Sushi **20% training data** | Same as 1 | EffNetB2 | 10 |

**Note:** All experiments use the same testing data (to keep evaluation consistent). Also notice how the first experiment is the smallest, and each consecutive experiment increases data, training time and model size.

# Inspecting test loss



# Inspecting test accuracy



# Inspecting the model architecture