

*Milestone Project 1*

# Experiment Tracking with



# Where can you get help?

• Follow along with the code

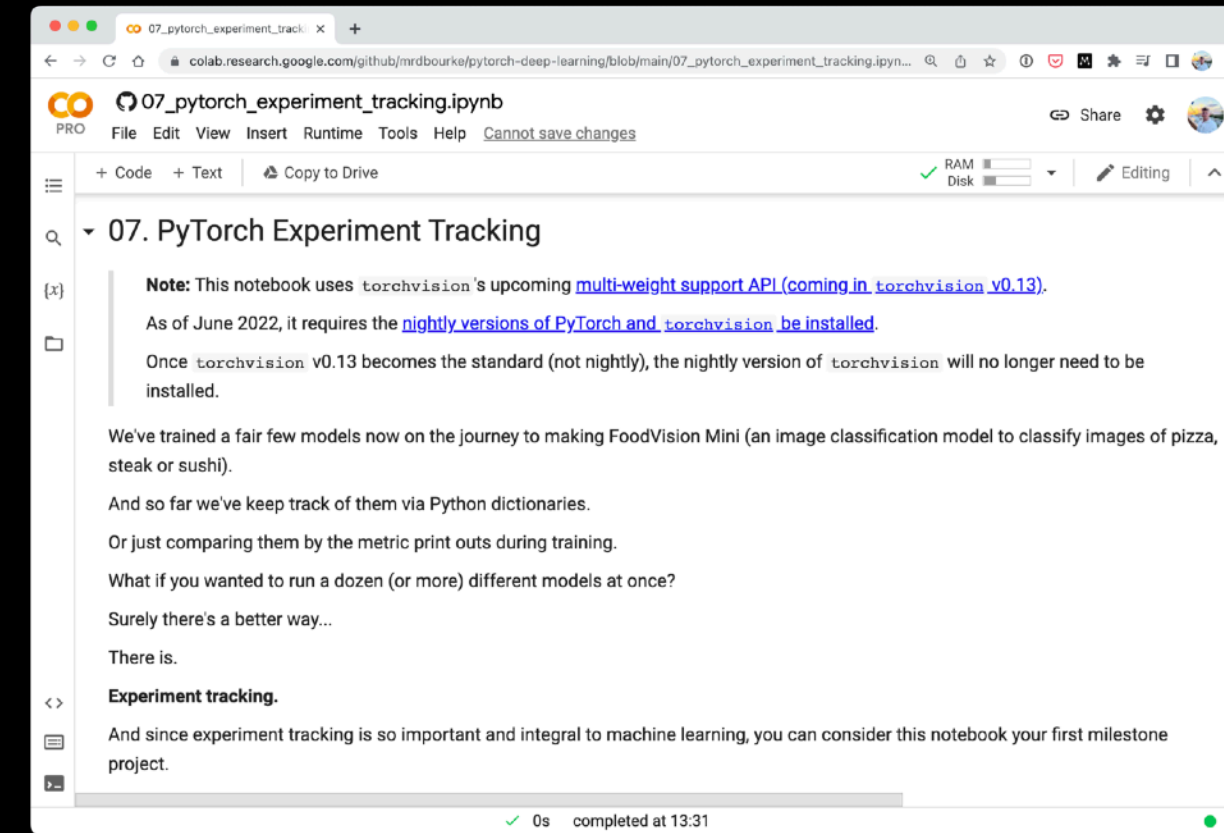
• Try it for yourself

• Press **SHIFT + CMD + SPACE** to read the docstring

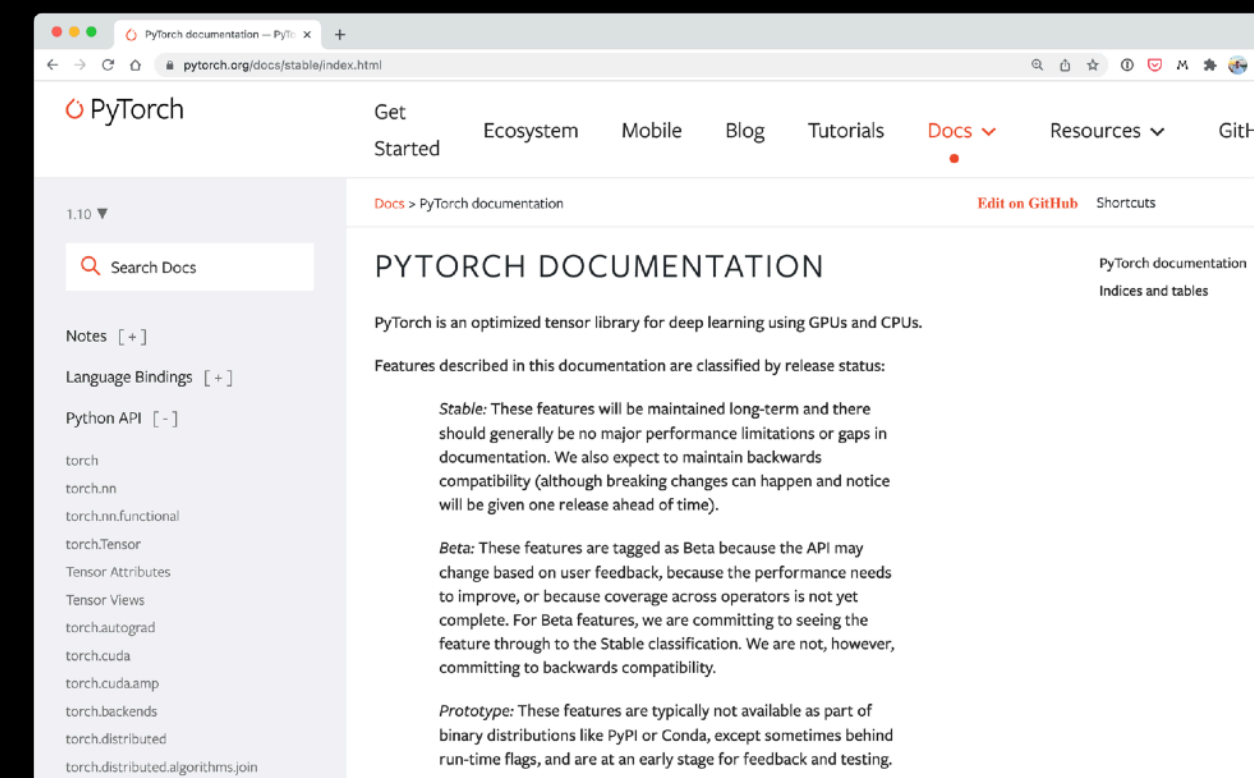
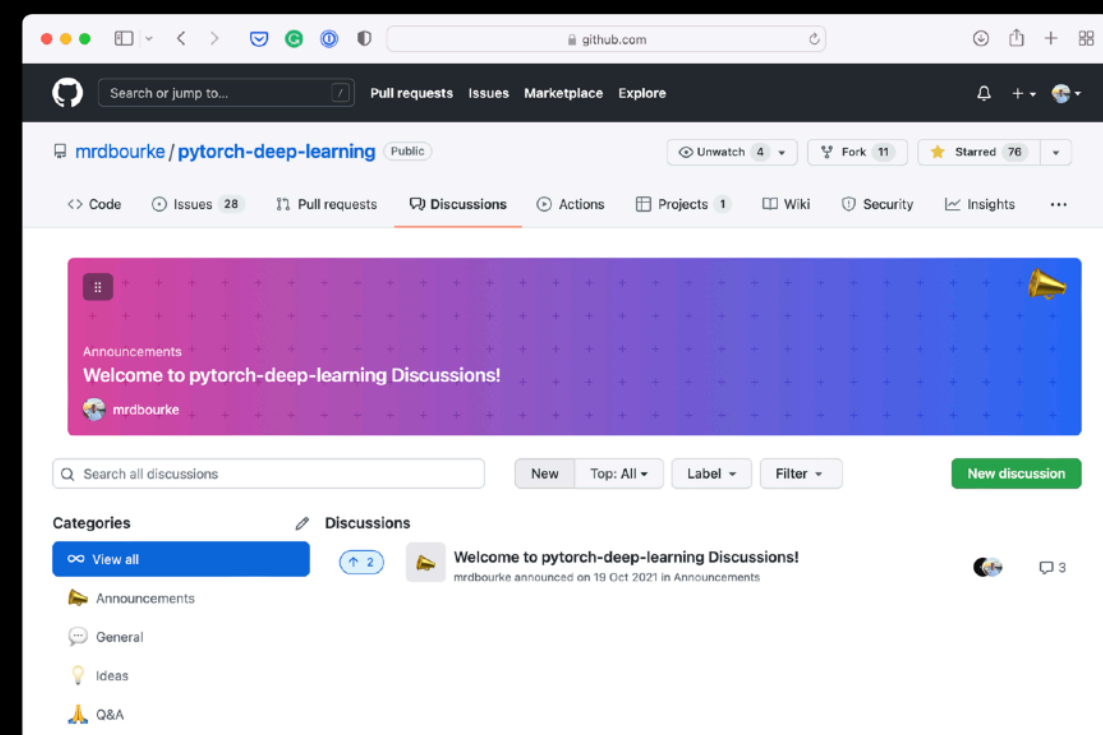
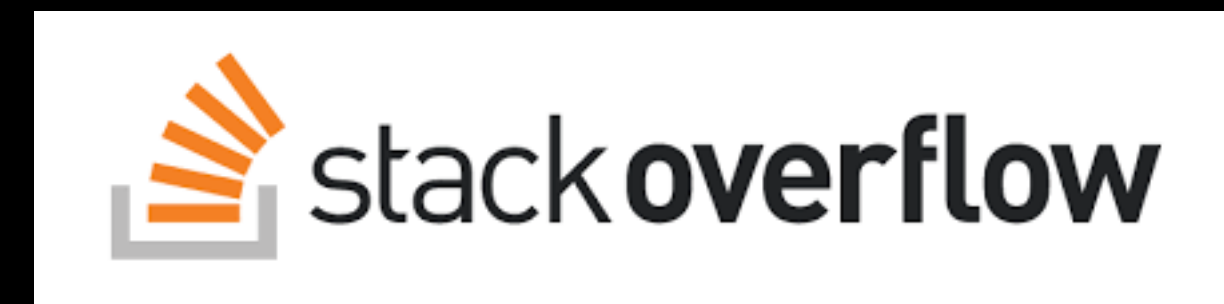
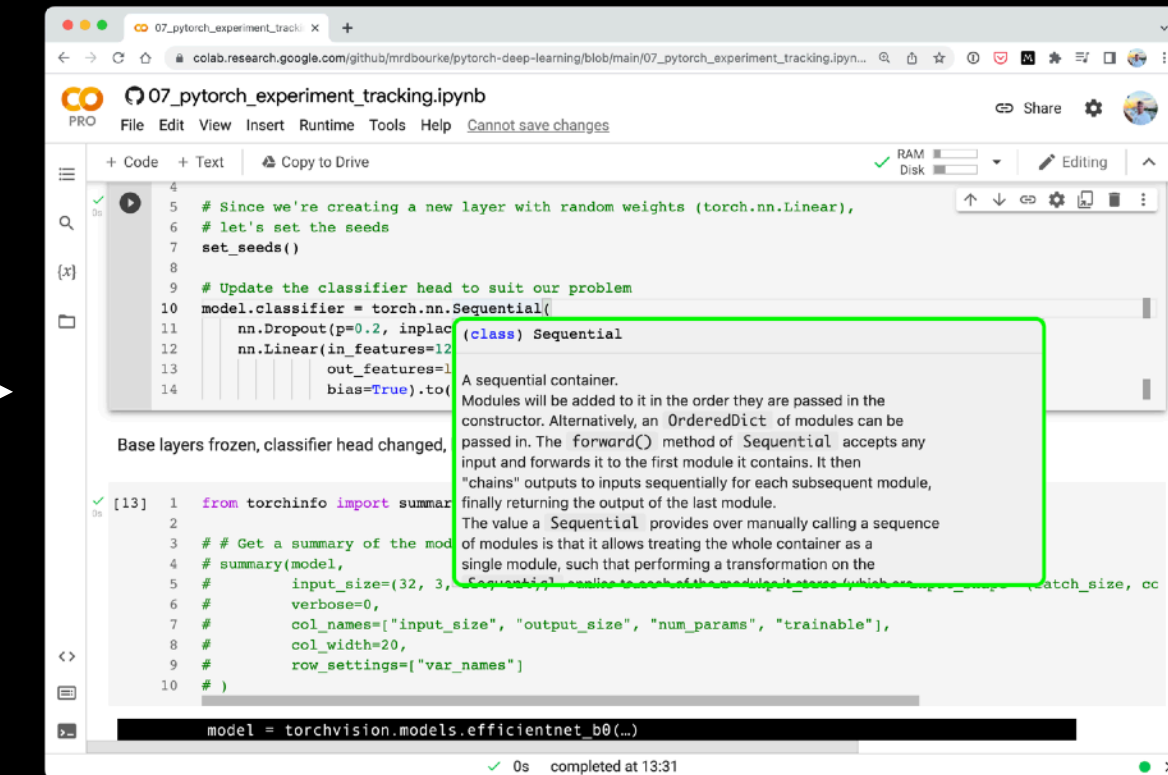
• Search for it

• Try again

• Ask



*"If in doubt, run the code"*



<https://www.github.com/mrdourke/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

```
1 # 1. Construct a model class that subclasses nn.Module
2 class CircleModelV0(nn.Module):
3     def __init__(self):
4         super().__init__()
5         # 2. Create 2 nn.Linear layers
6         self.layer_1 = nn.Linear(in_features=2, out_features=5)
7         self.layer_2 = nn.Linear(in_features=5, out_features=1)
8
9     # 3. Define a forward method containing the forward pass computation
10    def forward(self, x):
11        # Pass the data through both layers
12        return self.layer_2(self.layer_1(x))
13
14 # 4. Create an instance of the model and send it to target device
15 model_0 = CircleModelV0().to(device)
16 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_dropout
13 2022-07-07-model_5_results_v1_bigger_model_feature_extractor_50_epochs_50_percent_data_no_dropout
```

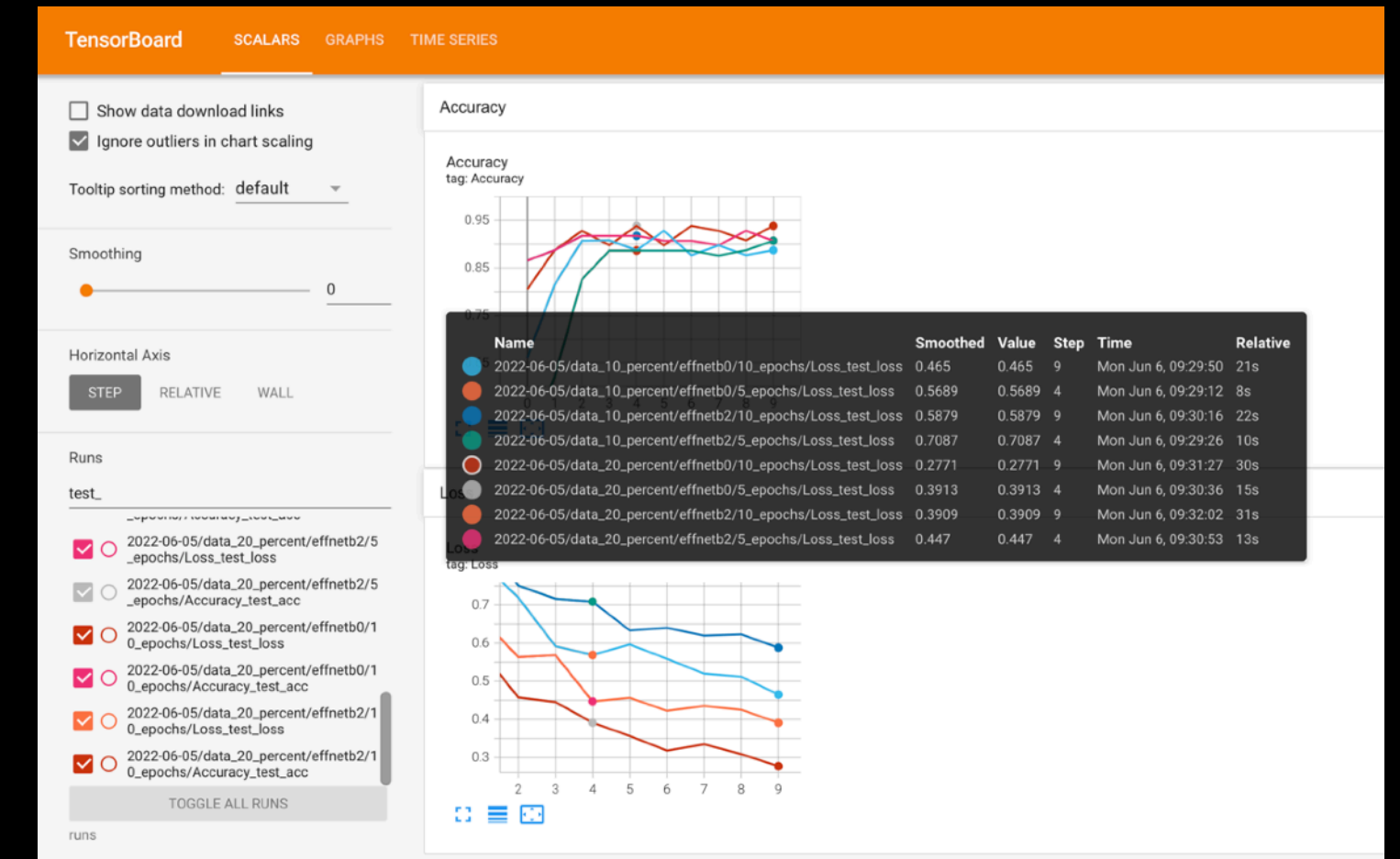
what should I try *less* of?

what should I try *more* of?

# Different ways to track experiments



Python dictionaries, CSV files, print outs.



TensorBoard

Source: <https://www.tensorflow.org/tensorboard>

Search: "track machine learning experiments"



MLflow

Source: <https://mlflow.org/docs/latest/tracking.html>



Weights & Biases

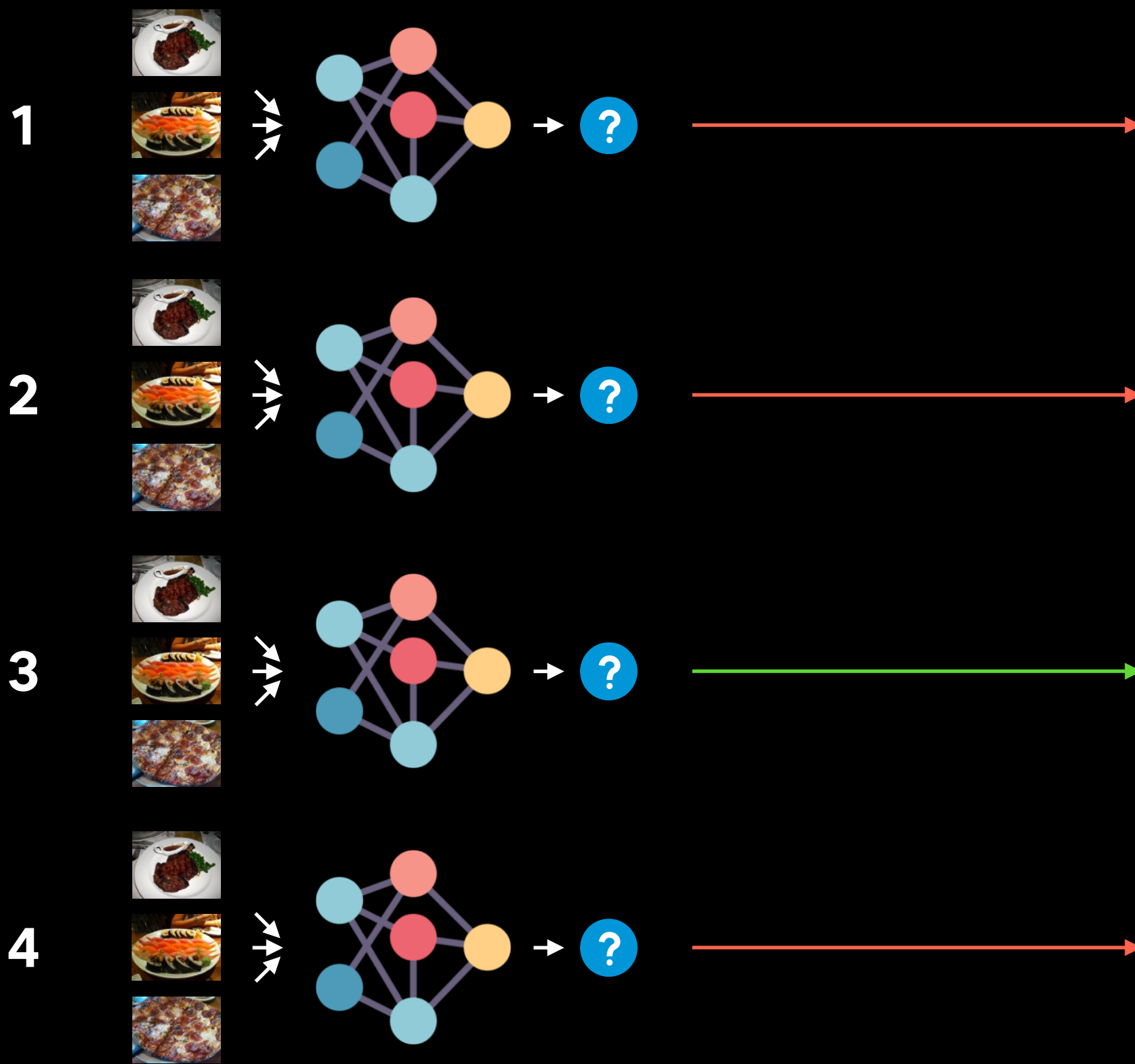
Source: <https://wandb.ai/site/experiment-tracking>



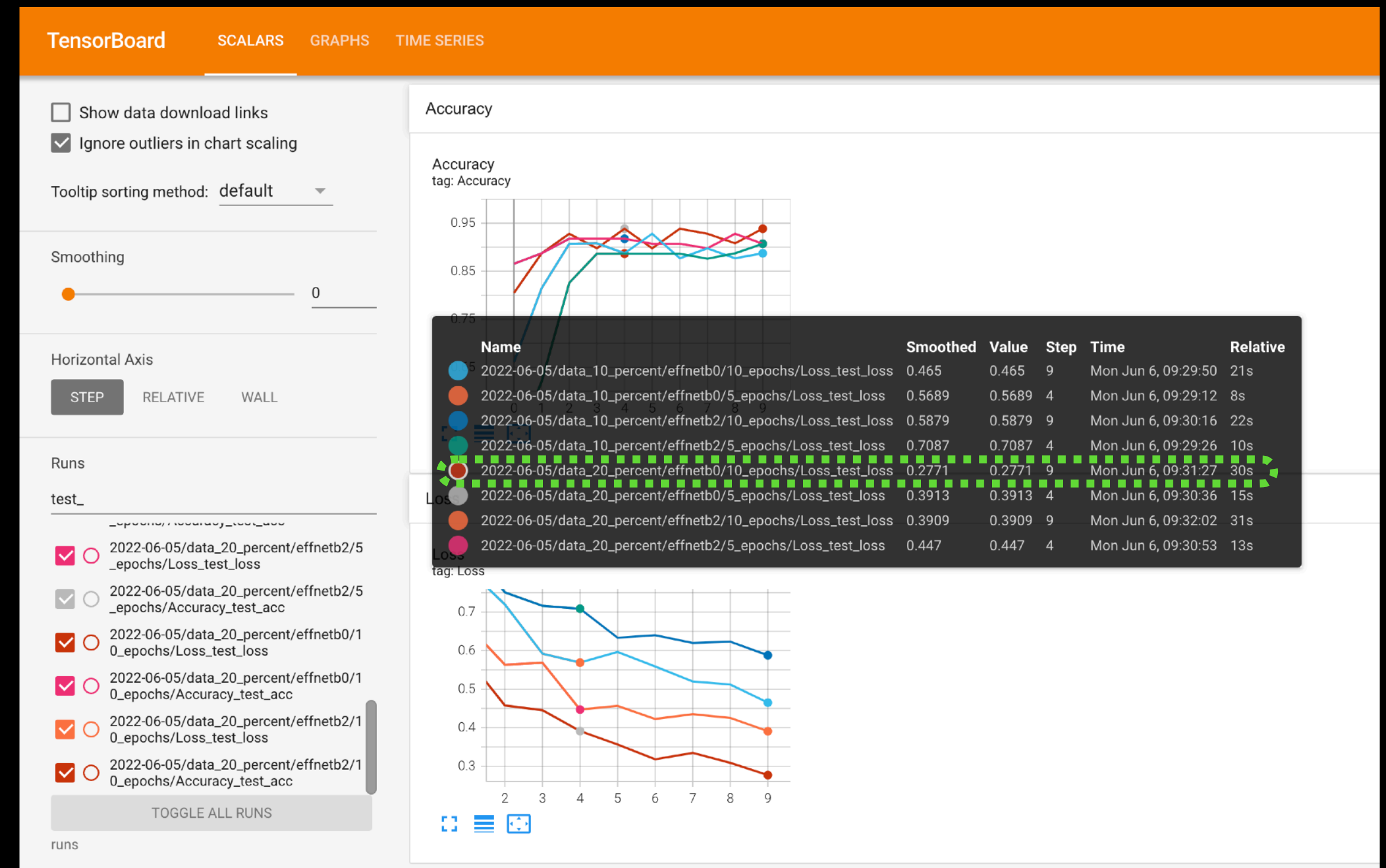
# What we're going to build

## 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

Many layers untrainable (frozen)

Only last layers are trainable

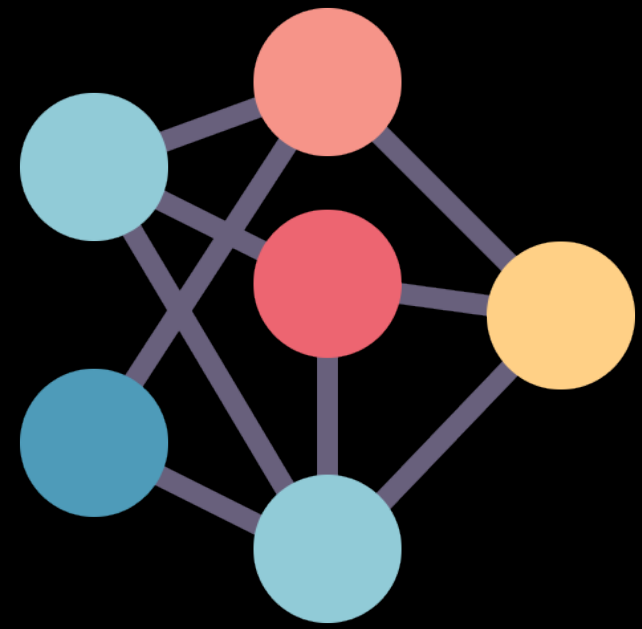
Final layer output (same as number of classes 🍕 🍣 🍱)

Less trainable parameters because many layers are frozen

```
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
```

# What experiments should you try?



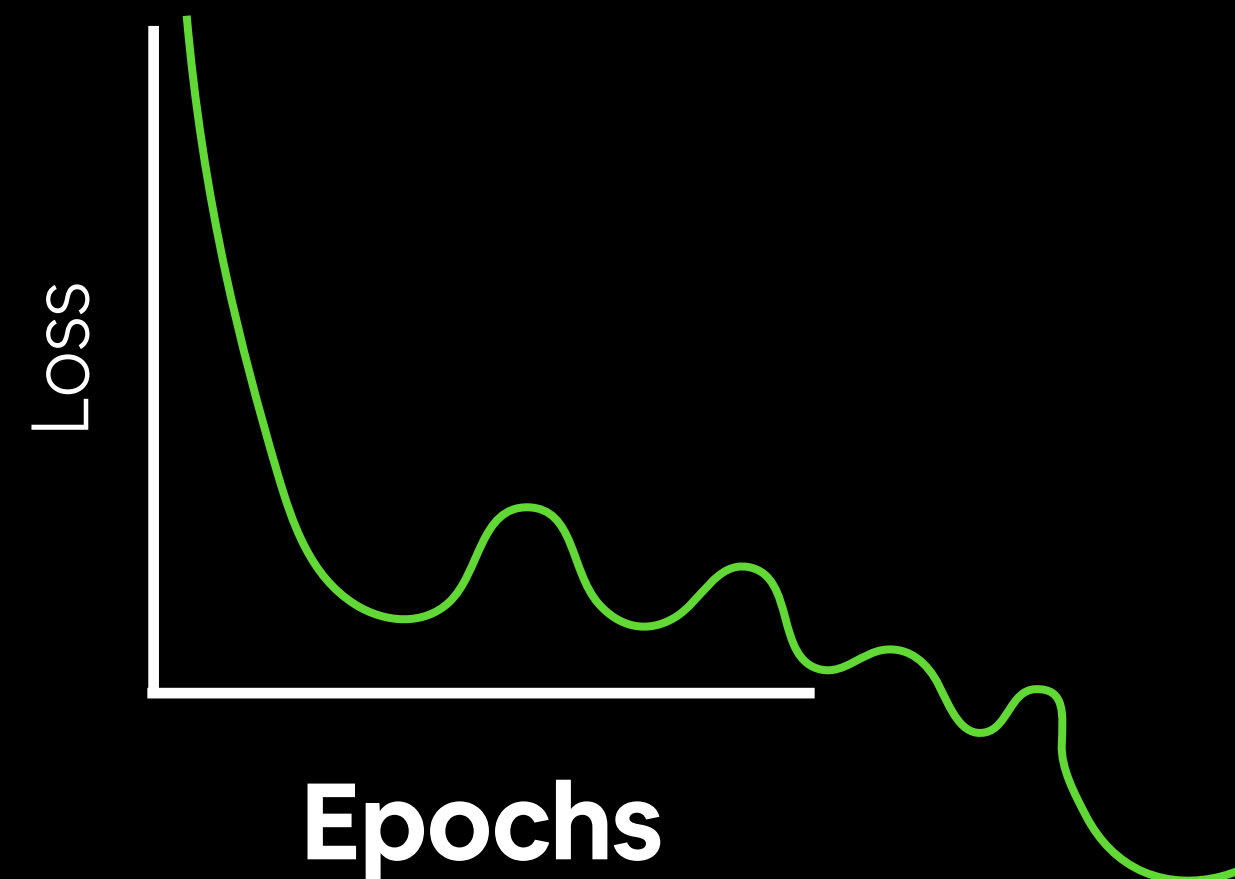
Model architecture

```
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_scheduler()
```

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 <b>10% training data</b>	Pizza, Steak Sushi 10% testing data	EffNetB0	5
2	Pizza, Steak Sushi <b>10% training data</b>	Same as 1	EffNetB2	5
3	Pizza, Steak Sushi <b>10% training data</b>	Same as 1	EffNetB0	10
4	Pizza, Steak Sushi <b>10% training data</b>	Same as 1	EffNetB2	10
5	Pizza, Steak Sushi <b>20% training data</b>	Same as 1	EffNetB0	5
6	Pizza, Steak Sushi <b>20% training data</b>	Same as 1	EffNetB2	5
7	Pizza, Steak Sushi <b>20% training data</b>	Same as 1	EffNetB0	10
8	Pizza, Steak Sushi <b>20% training data</b>	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

TensorBoard SCALARS GRAPHS TIME SERIES

Show data download links  
 Ignore outliers in chart scaling  
Tooltip sorting method: default

Smoothing: 0

Horizontal Axis: STEP RELATIVE WALL

Runs

- 2022-06-05/data\_20\_percent/effnetb2/5\_epochs/Loss\_test\_loss
- 2022-06-05/data\_20\_percent/effnetb2/5\_epochs/Accuracy\_test\_acc
- 2022-06-05/data\_20\_percent/effnetb0/10\_epochs/Loss\_test\_loss
- 2022-06-05/data\_20\_percent/effnetb0/10\_epochs/Accuracy\_test\_acc
- 2022-06-05/data\_20\_percent/effnetb2/10\_epochs/Loss\_test\_loss
- 2022-06-05/data\_20\_percent/effnetb2/10\_epochs/Accuracy\_test\_acc

TOGGLE ALL RUNS

runs

### Accuracy

Accuracy tag: Accuracy

Name	Smoothed	Value	Step	Time	Relative
2022-06-05/data_10_percent/effnetb0/10_epochs/Loss_test_loss	0.465	0.465	9	Mon Jun 6, 09:29:50	21s
2022-06-05/data_10_percent/effnetb0/5_epochs/Loss_test_loss	0.5689	0.5689	4	Mon Jun 6, 09:29:12	8s
2022-06-05/data_10_percent/effnetb2/10_epochs/Loss_test_loss	0.5879	0.5879	9	Mon Jun 6, 09:30:16	22s
2022-06-05/data_10_percent/effnetb2/5_epochs/Loss_test_loss	0.7087	0.7087	4	Mon Jun 6, 09:29:26	10s
2022-06-05/data_20_percent/effnetb0/10_epochs/Loss_test_loss	0.2771	0.2771	9	Mon Jun 6, 09:31:27	30s
2022-06-05/data_20_percent/effnetb0/5_epochs/Loss_test_loss	0.3913	0.3913	4	Mon Jun 6, 09:30:36	15s
2022-06-05/data_20_percent/effnetb2/10_epochs/Loss_test_loss	0.3909	0.3909	9	Mon Jun 6, 09:32:02	31s
2022-06-05/data_20_percent/effnetb2/5_epochs/Loss_test_loss	0.447	0.447	4	Mon Jun 6, 09:30:53	13s

### Loss

Loss tag: Loss

Accuracy

# Inspecting test accuracy

TensorBoard SCALARS GRAPHS TIME SERIES

Show data download links  
 Ignore outliers in chart scaling  
Tooltip sorting method: default

Smoothing: 0

Horizontal Axis: STEP RELATIVE WALL

Runs

test\_acc

- Jun05\_23-03-28\_f652c984d314/Accuracy\_test\_acc
- 2022-06-05/data\_10\_percent/effnetb0/5\_epochs/Accuracy\_test\_acc
- 2022-06-05/data\_10\_percent/effnetb2/5\_epochs/Accuracy\_test\_acc
- 2022-06-05/data\_10\_percent/effnetb0/10\_epochs/Accuracy\_test\_acc
- 2022-06-05/data\_10\_percent/effnetb2/10\_epochs/Accuracy\_test\_acc
- 2022-06-05/data\_20\_percent/effnetb0/5\_epochs/Accuracy\_test\_acc

Filter tags (regular expressions supported)

### Accuracy

Accuracy tag: Accuracy

Name	Smoothed	Value	Step	Time	Relative
2022-06-05/data_10_percent/effnetb0/10_epochs/Accuracy_test_acc	0.9072	0.9072	9	Mon Jun 6, 09:04:43	18s
2022-06-05/data_10_percent/effnetb0/5_epochs/Accuracy_test_acc	0.8864	0.8864	4	Mon Jun 6, 09:04:11	8s
2022-06-05/data_10_percent/effnetb2/10_epochs/Accuracy_test_acc	0.8873	0.8873	9	Mon Jun 6, 09:05:05	19s
2022-06-05/data_10_percent/effnetb2/5_epochs/Accuracy_test_acc	0.8873	0.8873	4	Mon Jun 6, 09:04:22	8s
2022-06-05/data_20_percent/effnetb0/10_epochs/Accuracy_test_acc	0.9072	0.9072	9	Mon Jun 6, 09:06:13	30s
2022-06-05/data_20_percent/effnetb0/5_epochs/Accuracy_test_acc	0.9176	0.9176	4	Mon Jun 6, 09:05:22	13s
2022-06-05/data_20_percent/effnetb2/10_epochs/Accuracy_test_acc	0.9384	0.9384	9	Mon Jun 6, 09:06:48	30s
2022-06-05/data_20_percent/effnetb2/5_epochs/Accuracy_test_acc	0.9384	0.9384	4	Mon Jun 6, 09:05:39	13s



# Inspecting the model architecture

TensorBoard SCALARS GRAPHS TIME SERIES

Search nodes (regex)

Fit to screen  
Download PNG  
Upload file

Run (1) Jun05\_23-14-39\_f652c984d314  
Tag (2) Default

Graph type  
 Op graph  
 Conceptual graph  
 Profile

Node options  
 Trace inputs

Legend

- colors same substructure
- unique substructure (\* = expandable)
- Namespace\* ?
- OpNode ?
- Unconnected series\* ?
- Connected series\* ?
- Constant ?
- Summary ?
- Dataflow edge ?
- Control dependency edge ?
- Reference edge ?

The diagram illustrates the architecture of an EfficientNet model. It shows a flow from an 'input' node at the bottom to an 'output' node at the top. The main body of the model is a large box labeled 'EfficientNet'. Inside this box, the architecture is composed of several layers: a 'Sequential[featu...]' layer at the bottom, followed by an 'AdaptiveAvgPoo...' layer, and a 'Sequential[class...]' layer at the top. The 'input' node feeds into the 'Sequential[featu...]' layer. The 'AdaptiveAvgPoo...' layer receives input from the 'Sequential[featu...]' layer and outputs to the 'Sequential[class...]' layer. The 'Sequential[class...]' layer then outputs to the 'output' node. The diagram also shows some intermediate nodes and edges, such as 'input.423' and '3576 3577', which likely represent specific operations or data points within the model.