Milestone Project 3

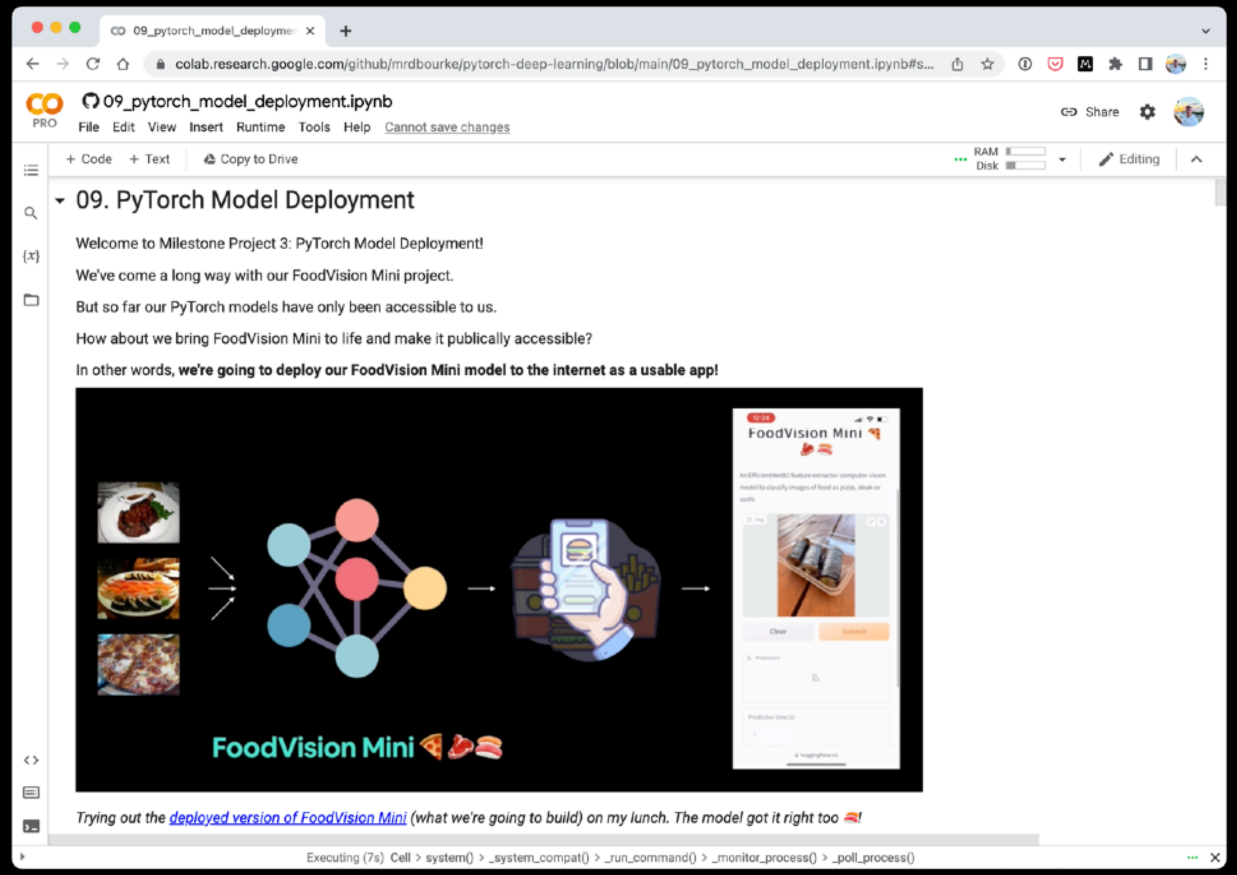
Model Deployment with

PyTorch

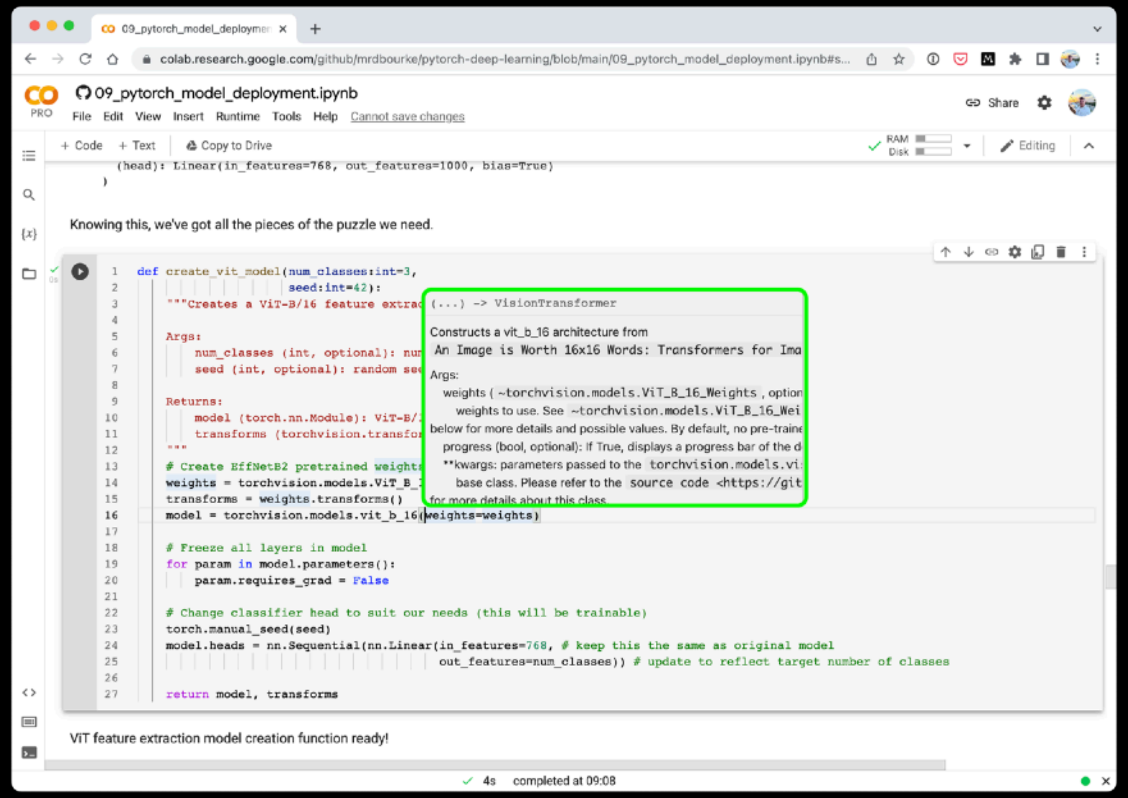
Where can you get help?

"If in doubt, run the code"

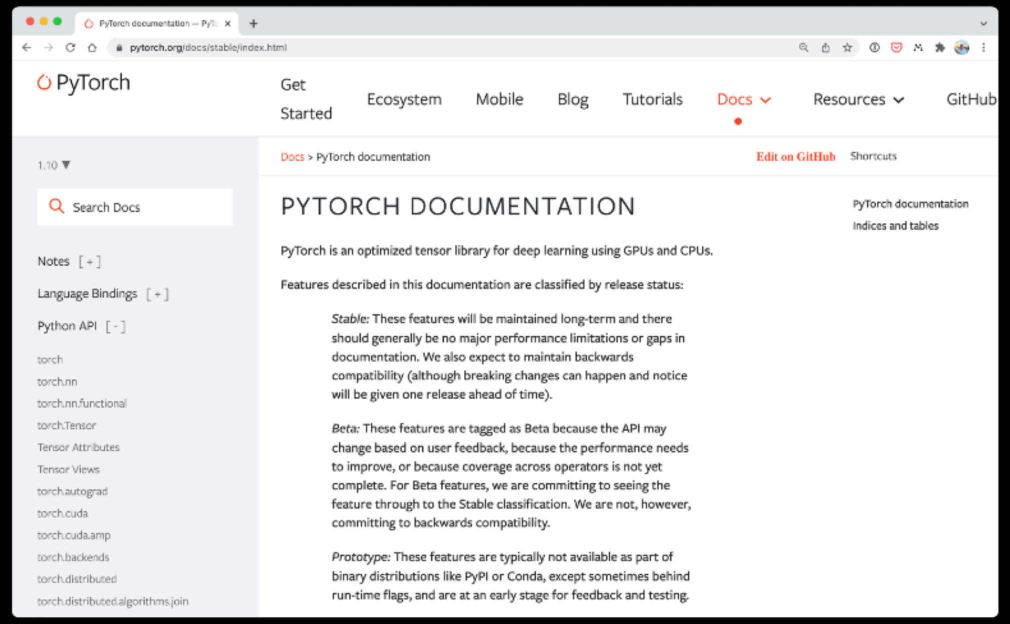
- Follow along with the code

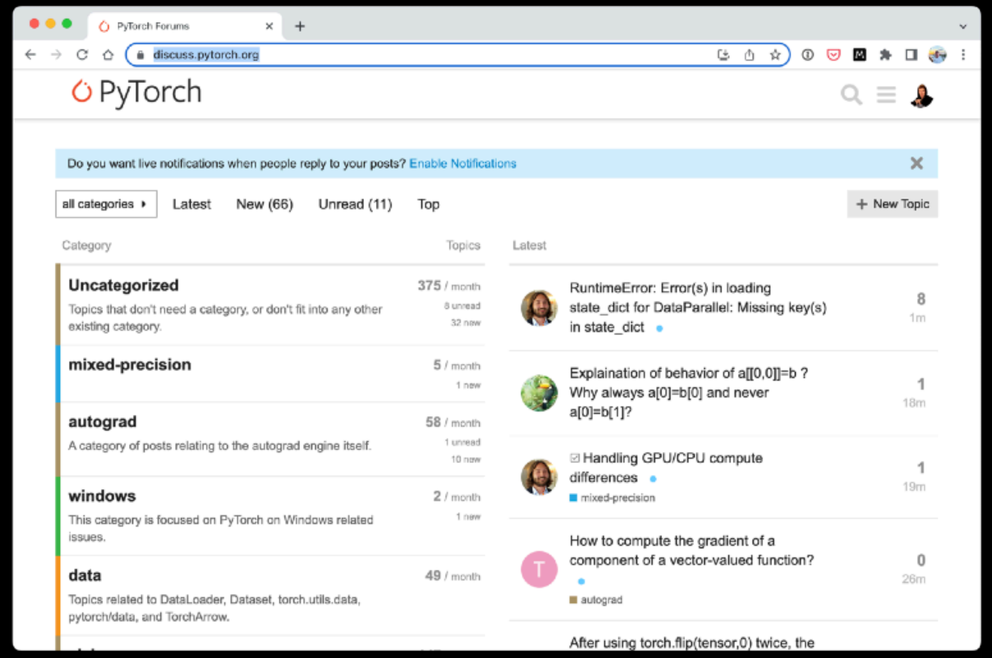


- Try it for yourself
- Press SHIFT + CMD + SPACE to read the docstring

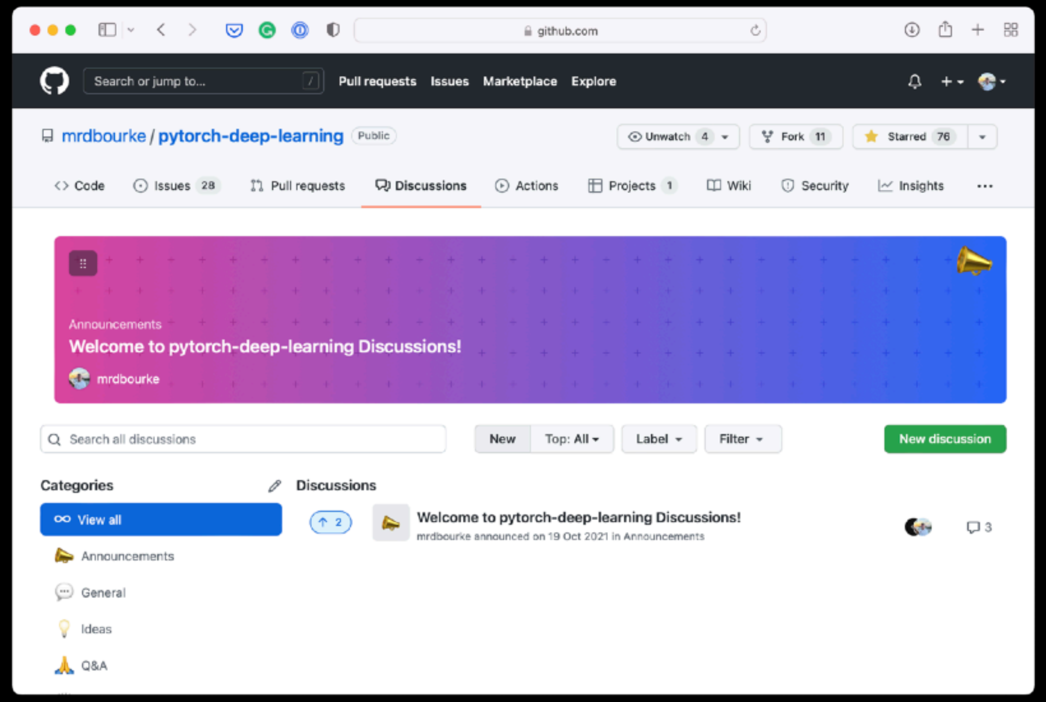


- Search for it





- Try again
- Ask

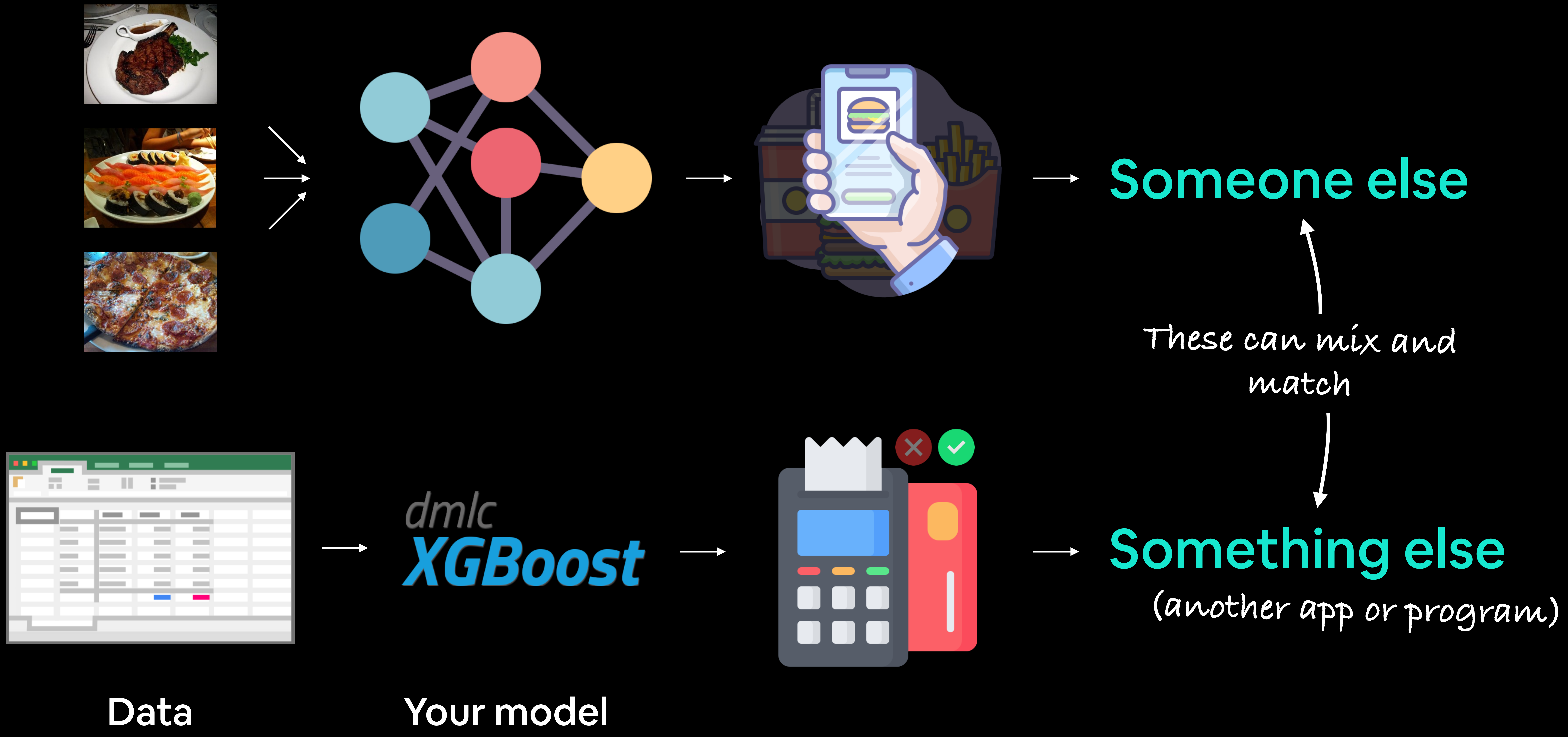


https://www.github.com/mrdbourke/pytorch-deep-learning/discussions

“What is machine learning model deployment?”

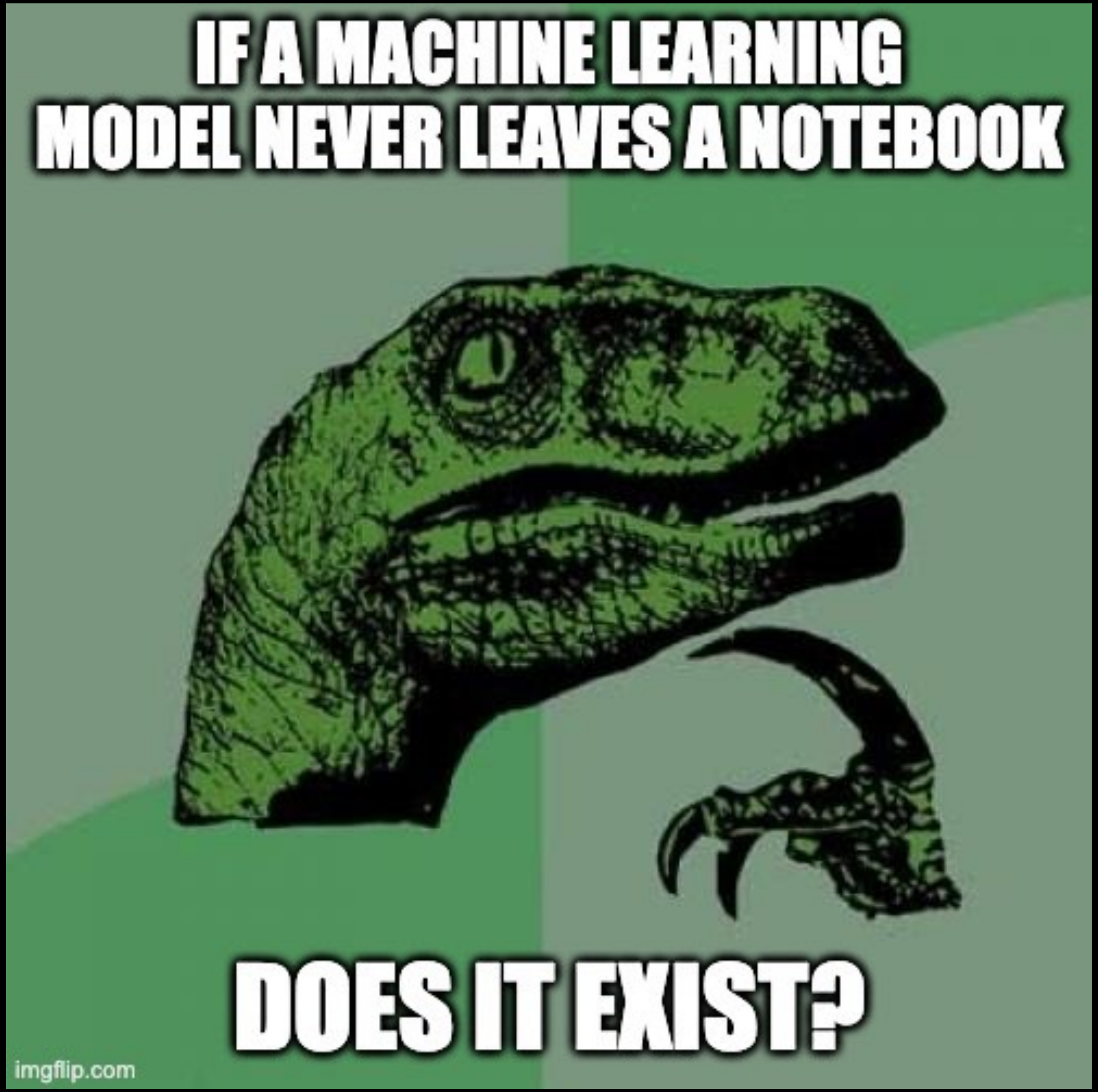
Making your machine learning model available to someone/something else.

What is model deployment?



“Why deploy machine learning models?”

1. It’s fun… and…


IF A MACHINE LEARNING MODEL NEVER LEAVES A NOTEBOOK
DOES IT EXIST?
imgflip.com

Three datasets

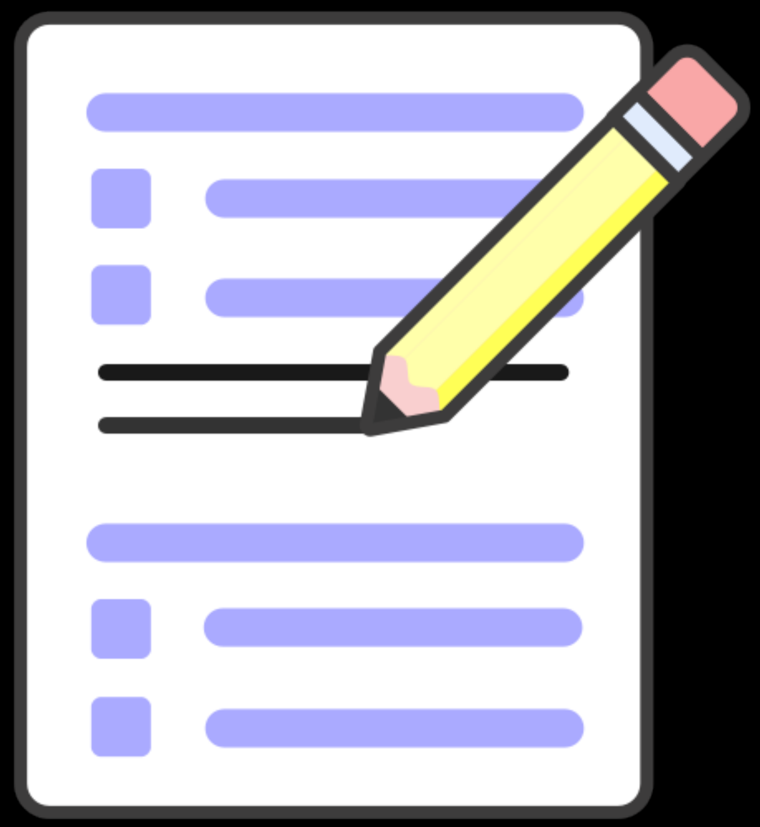
(possibly the most important concept in machine learning...)

Model learns patterns from here

Course materials
(training set)

Practice exam
(validation set)

Tune model patterns



Final exam
(test set)

See if the model is ready for the wild

Generalization

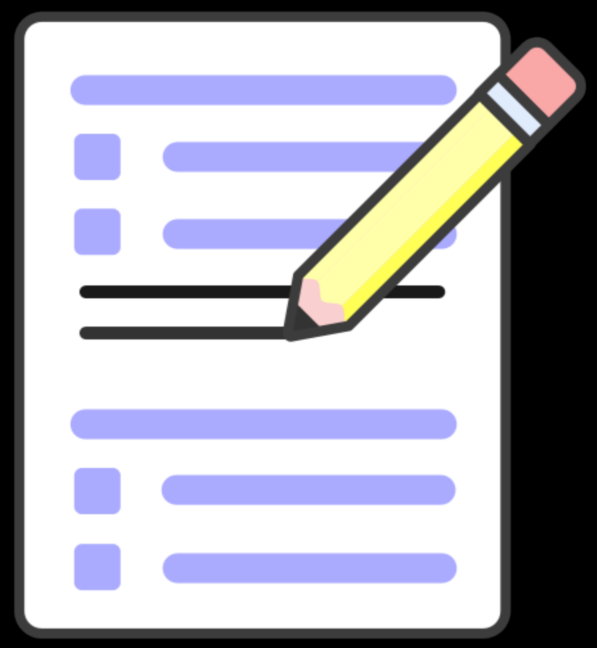
The ability for a machine learning model to perform well on data it hasn't seen before.

Four ~~Three~~ datasets

Notebook/local environment

Course materials
(training set)

Practice exam
(validation set)



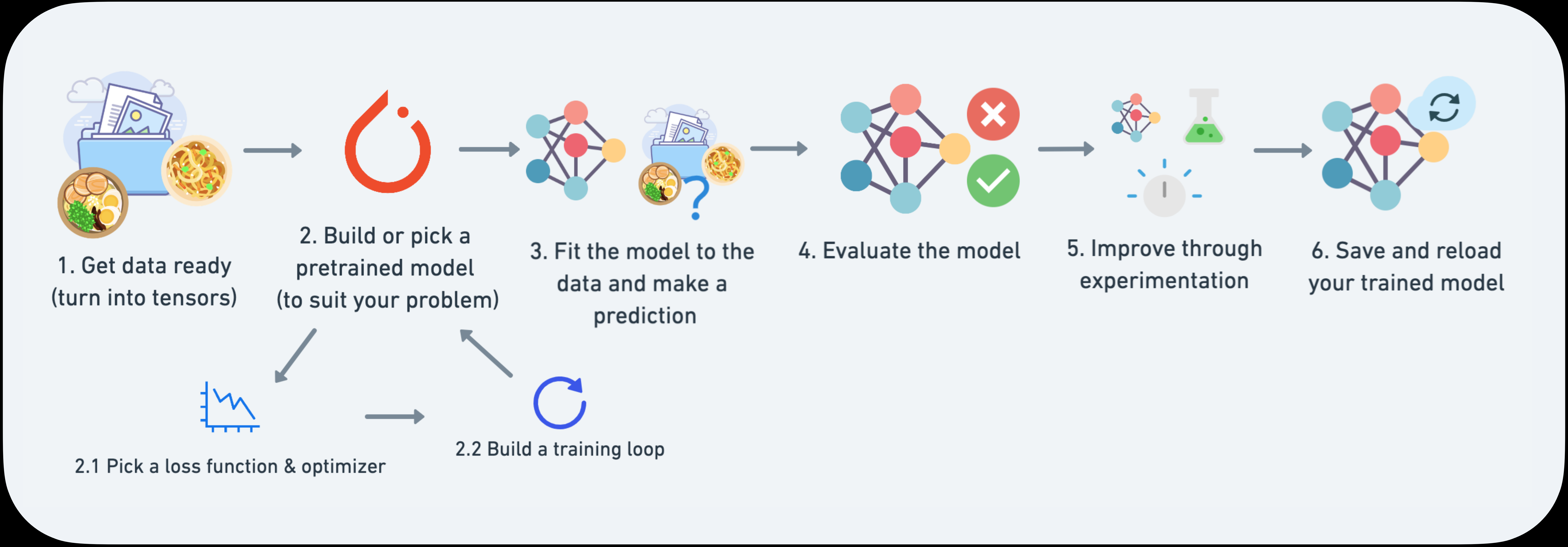
Final exam
(test set)

Real world

(you can only test here by deploying!)

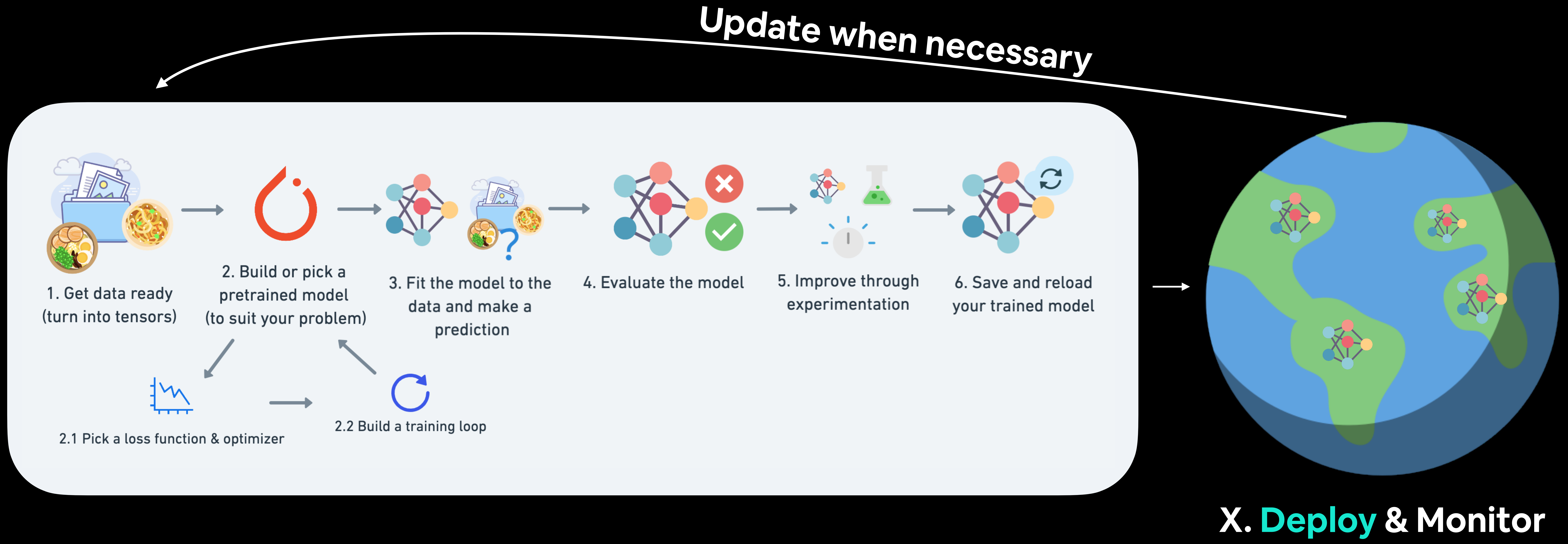
A PyTorch workflow

(one of many)



A PyTorch workflow

(one of many)



```
Machine Learning Engineer*
----

1. Download a paper
2. Implement it
3. Keep doing this until you have skills
```

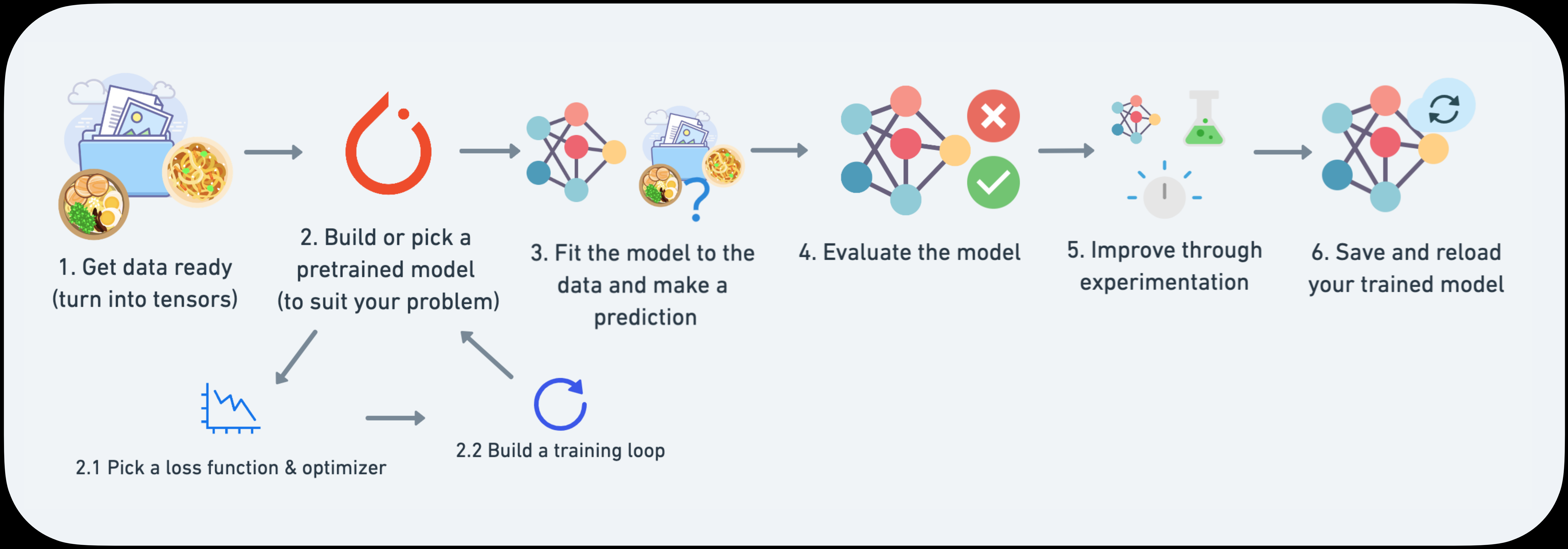
- George Hotz, founder of comma.ai

*Machine learning engineering also involves building infrastructure around your models/ data preprocessing steps

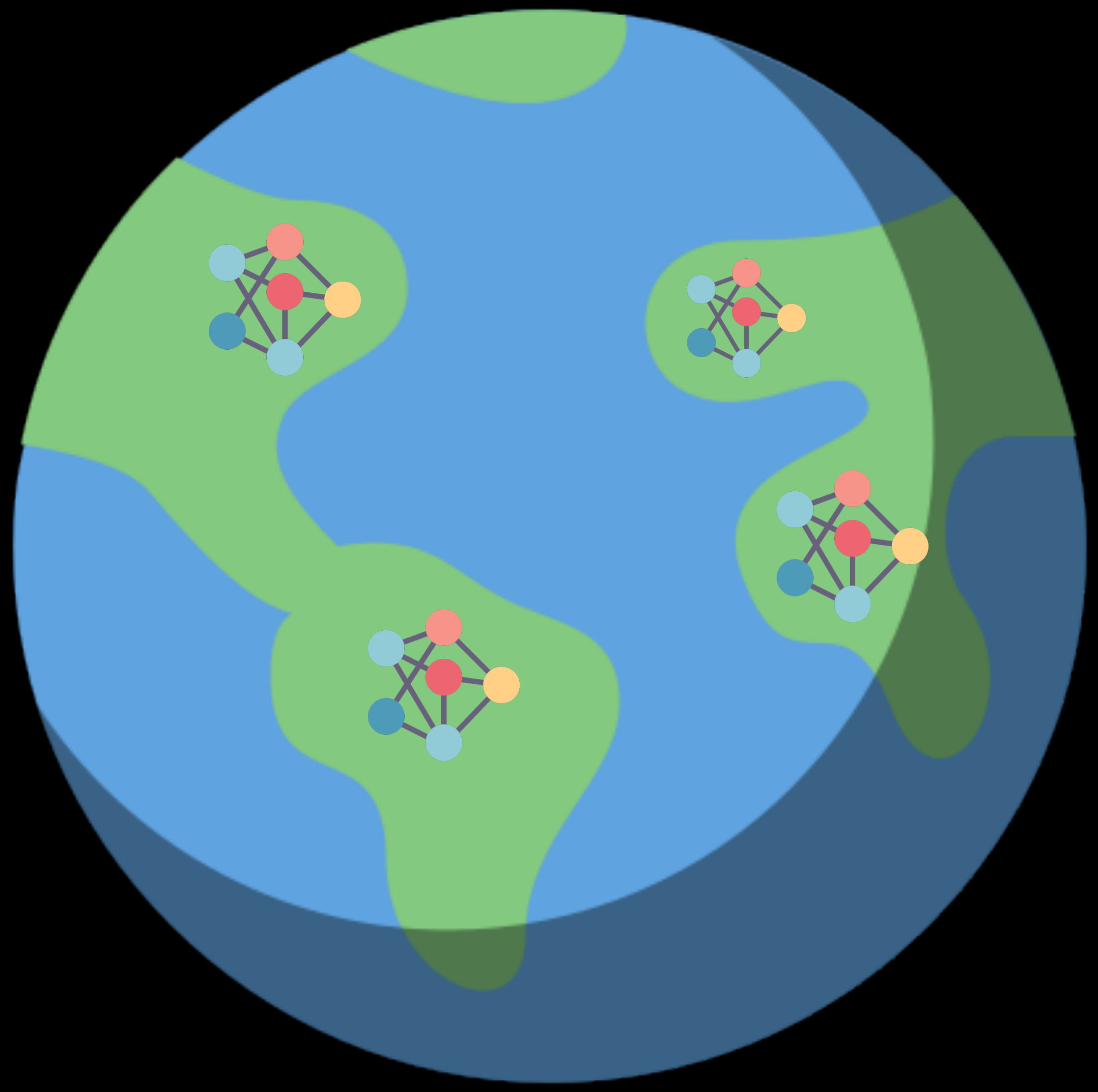
A PyTorch workflow

(one of many)

Machine learning



MLOps



X. Deploy & Monitor

(MLOps = machine learning operations or machine learning engineering)

See [PyTorch Extra Resources] for more on MLOps

“What kinds of machine learning model deployments are there?”

Deployment questions to ask

What is my most <u>ideal</u> machine learning model deployment scenario?

Start here and work backwards...

1. Works every time
2. Speed of light (fast)

Where's my model going to go?

1. On-device (edge)
2. Cloud

These can mix and match

How's my model going to function?

1. Online (real-time)
2. Offline (batch)

Deployment questions to ask

Where's my model going to go?

1. On-device (edge)
2. Cloud (a remote computer that isn't the actual device you're using)

On-device (edge)

aws

Azure

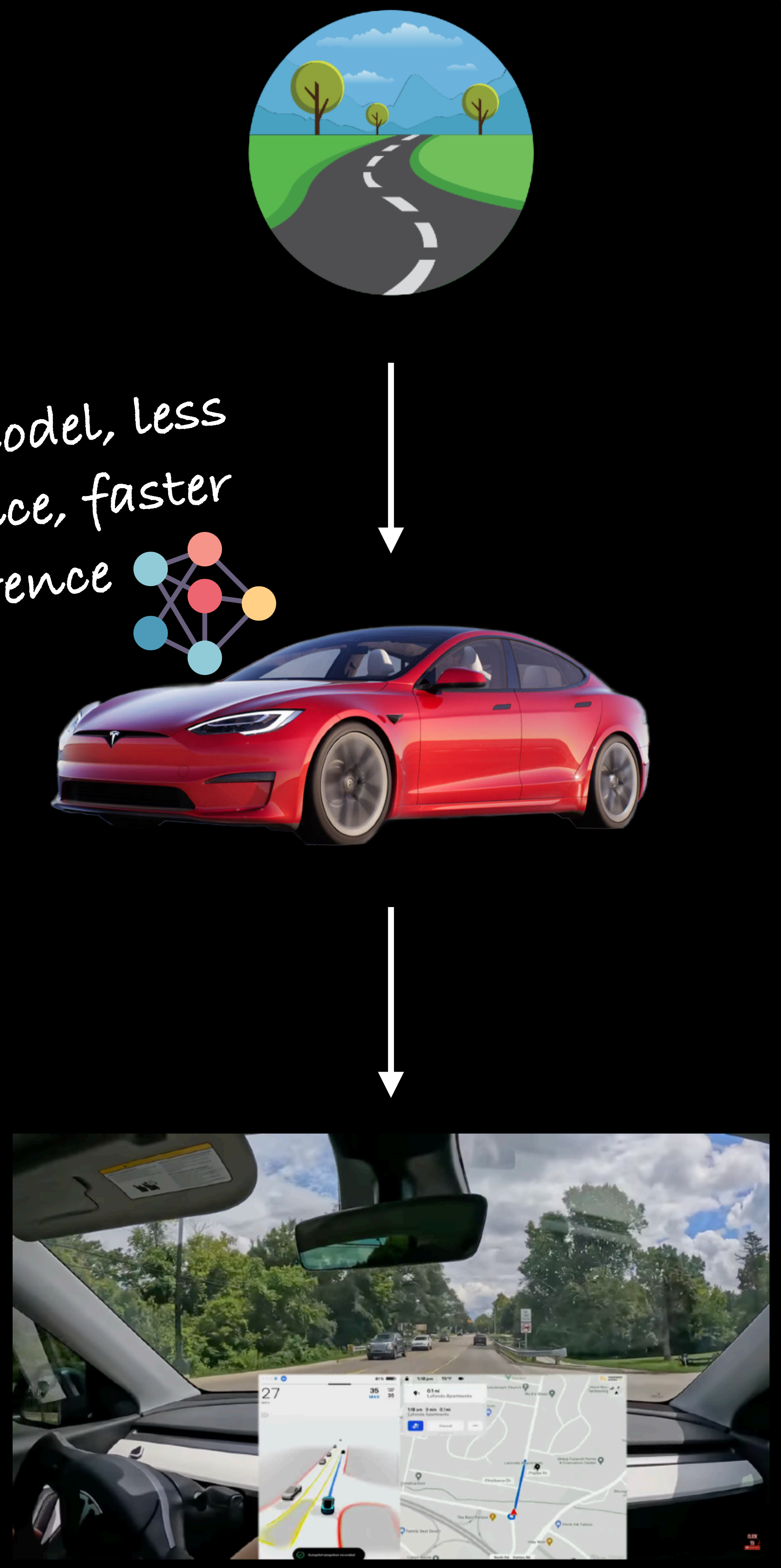
+ more

Cloud

Where’s my model going to go?

On-device (edge)

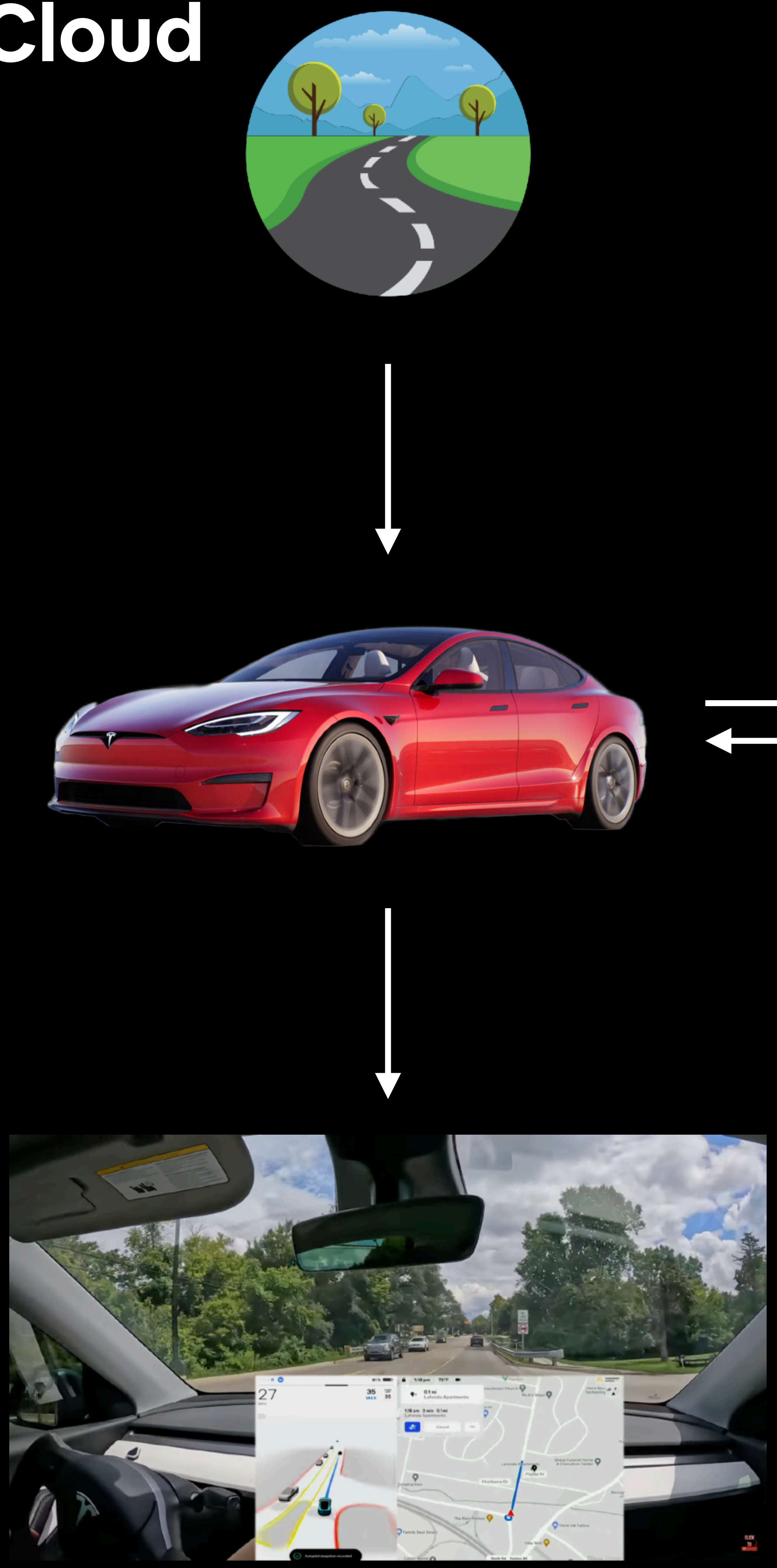
Smaller model, less performance, faster inference



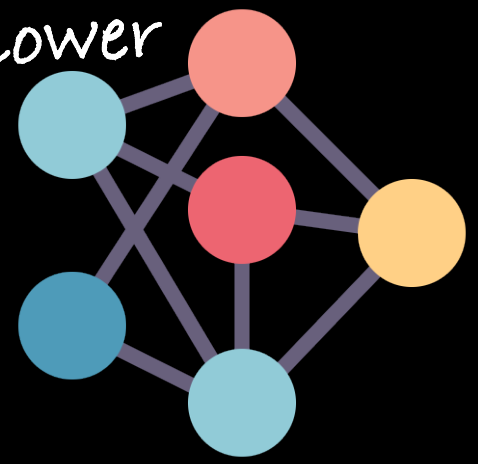
Source: Dirty Tesla YouTube channel

vs.

Cloud



Bigger model, better performance, slower inference



aws

Azure

+ more

Source: Dirty Tesla YouTube channel

Where's my model going to go?

Deployment location	Pros	Cons
On-device (edge/in-browser)	Can be very fast (since no data leaves the device)	Limited compute power (larger models take longer to run)
	Privacy preserving (again no data has to leave the device)	Limited storage space (smaller model size required)
	No internet connection required (sometimes)	Device-specific skills often required
On cloud (a compute device that isn't the actual device)	Near unlimited compute power (can scale up when needed)	Costs can get out of hand (if proper scaling limits aren't enforced)
	Can deploy one model and use everywhere (via API)	Predictions can be slower due to data having to leave device and predictions having to come back (network latency)
	Links into existing cloud ecosystem	Data has to leave device (this may cause privacy concerns)

See a fantastic example of deploying a PyTorch model to a Raspberry Pi (edge) **on the PyTorch blog and another write up of** Moving ML Inference from the Cloud to the Edge **by Jo Kristian Bergum.**

Deployment questions to ask

How's my model going to function?

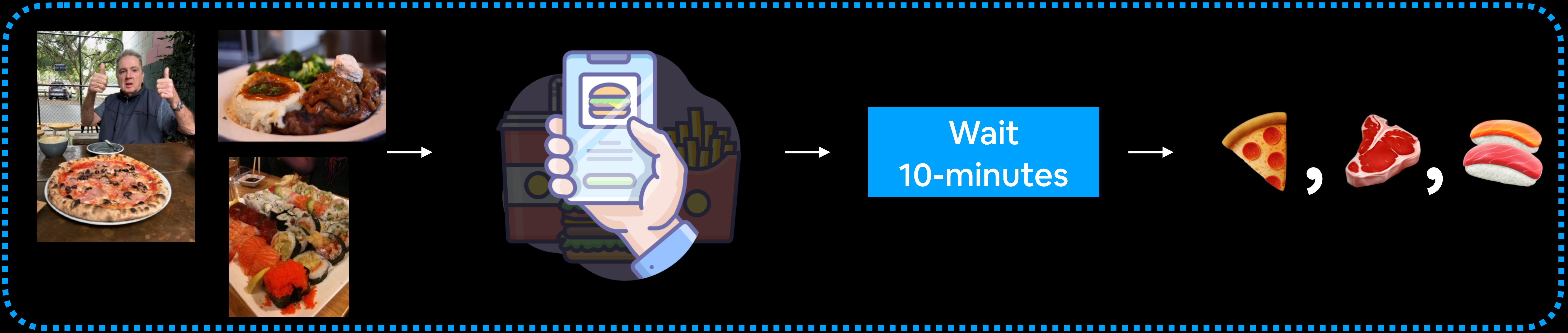
1. Online (real-time)
2. Offline (batch)

Online (real-time)

Predictions happen *immediately*

Offline (batch)

Predictions come at a *delay*

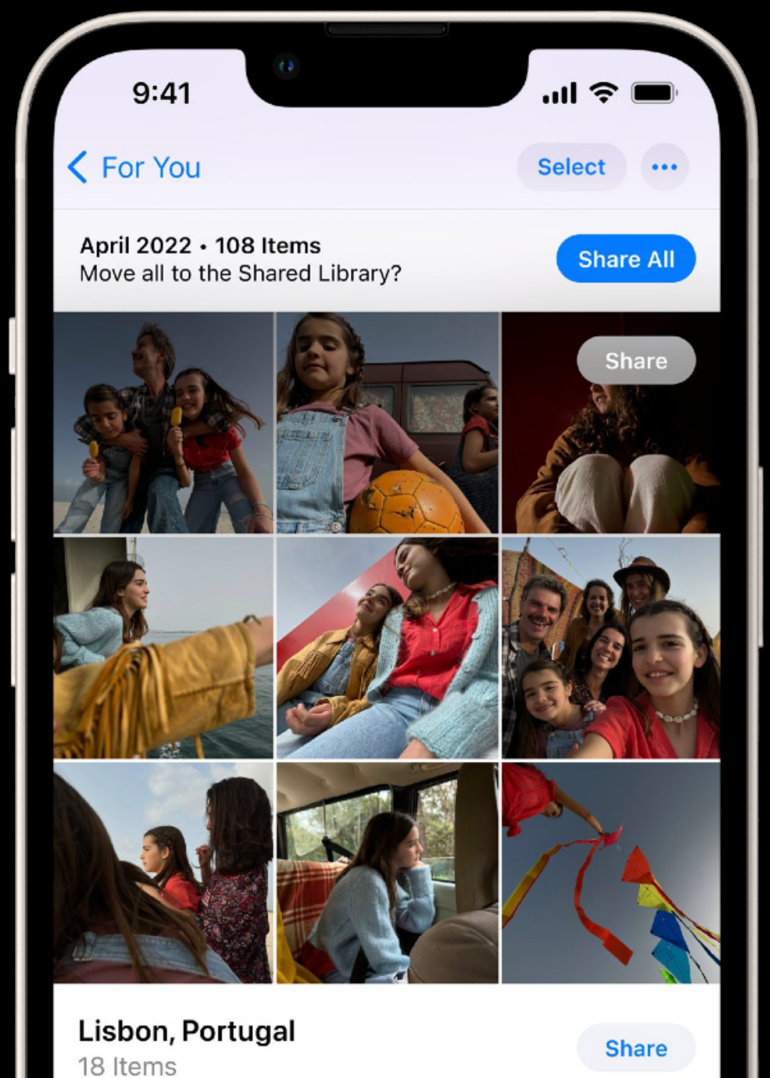


How's my model going to function?

	Offline prediction (batch/asynchronous)	Online prediction (real-time/synchronous)
Frequency	Periodical, such as every 10 minutes or every four hours or once per day	As soon as requests come (data comes in, prediction comes back ASAP)
Useful for	Processing/training on data when you don't need immediate results	When predictions are required as soon as data comes in
Optimized for	High throughout (such as making predictions/training on many samples at a time)	Low latency/high frequency (fast results, often)
Example	Recommendation engines, Apple photos app sorting, YouTube video indexing/sorting, training models	FoodVision Mini, fraudulent transaction detection, spam detection, translation, Tesla self-driving vision system

Source: Designing Machine Learning Systems book **by Chip Huyen**

Apple Photos: Sort photos offline, when plugged into charge



FoodVision Mini: Classify photos into 🍕, 🥩, 🍣 online (as soon as photo is uploaded)

(some)

Places/tools to help deploy machine learning models

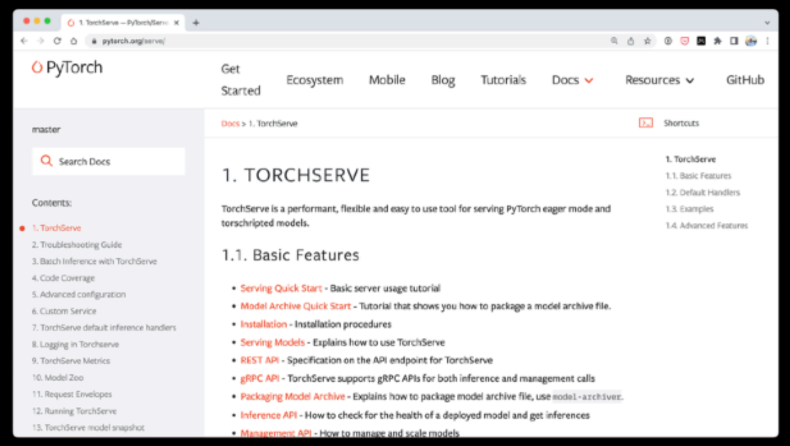
On-device (mobile/edge)

Google's MLKit

Apple's CoreML

General

FastAPI



TorchServe

ONNX (Open Neural Network Exchange)

Cloud

Google Cloud Vertex AI

AWS Sagemaker

Hugging Face

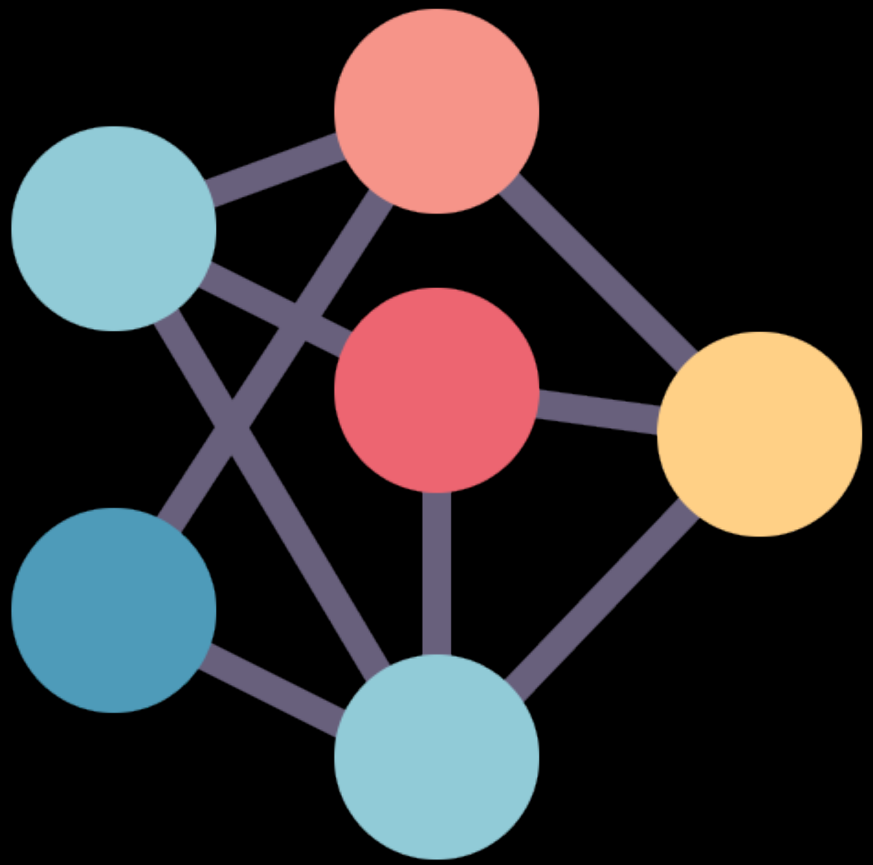
Azure Machine Learning

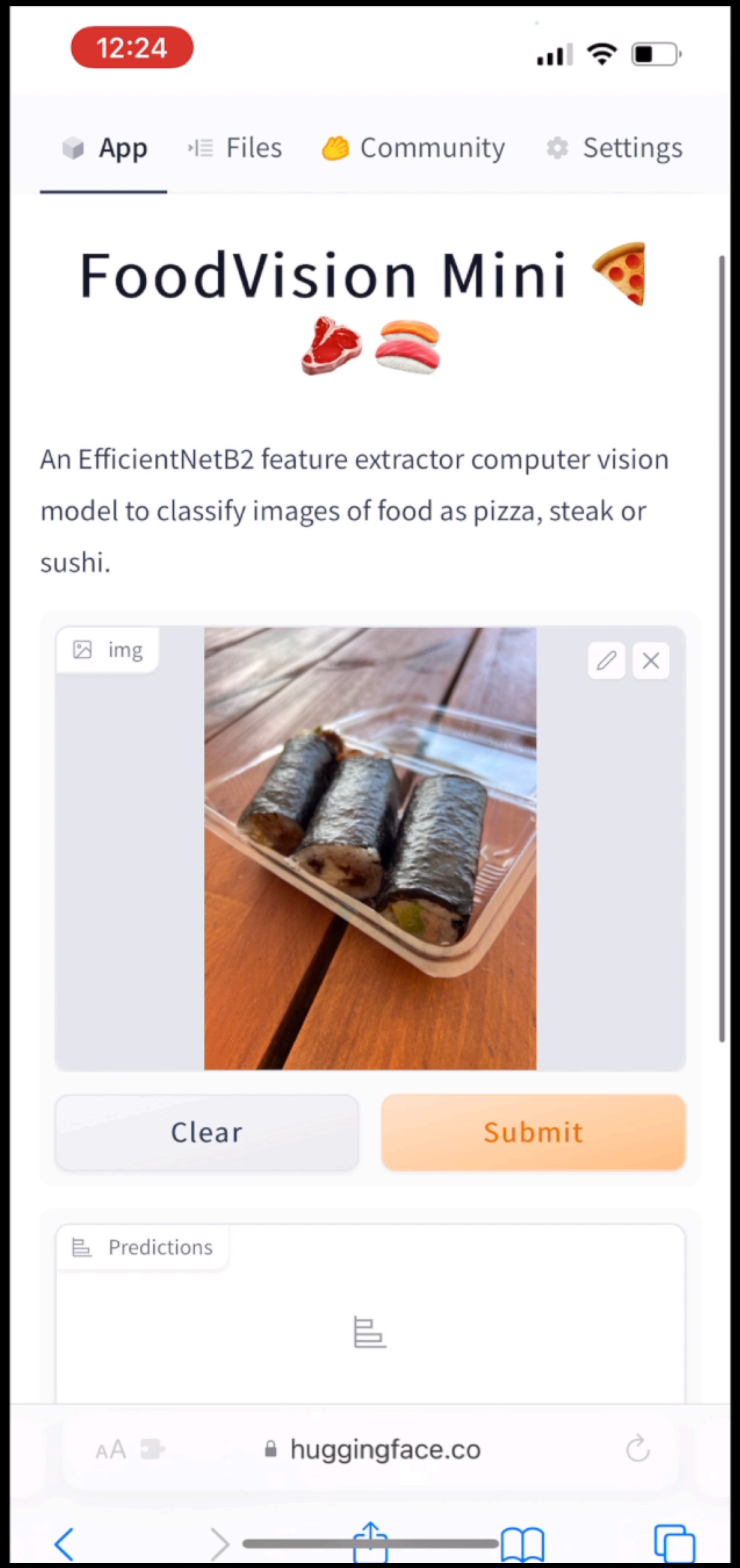
Bonus

Gradio

What we're doing

Deploying our FoodVision Mini machine learning model



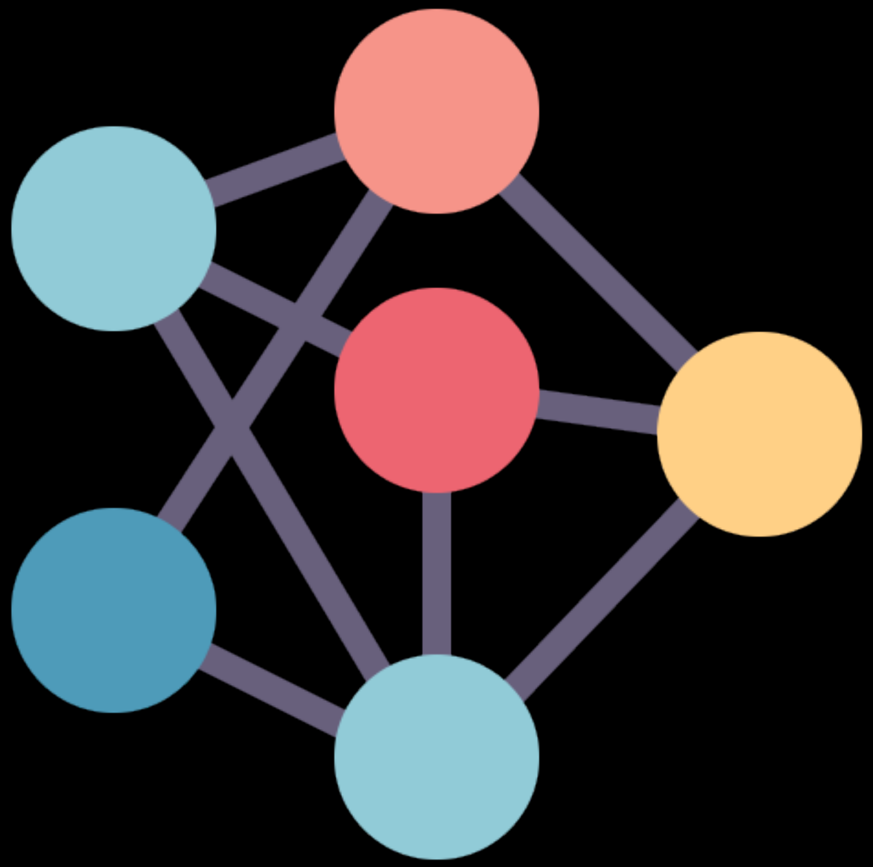


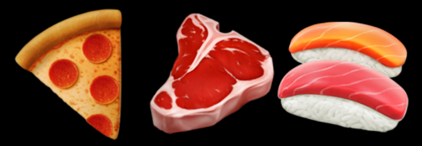
FoodVision Mini 🍕🥩🍣

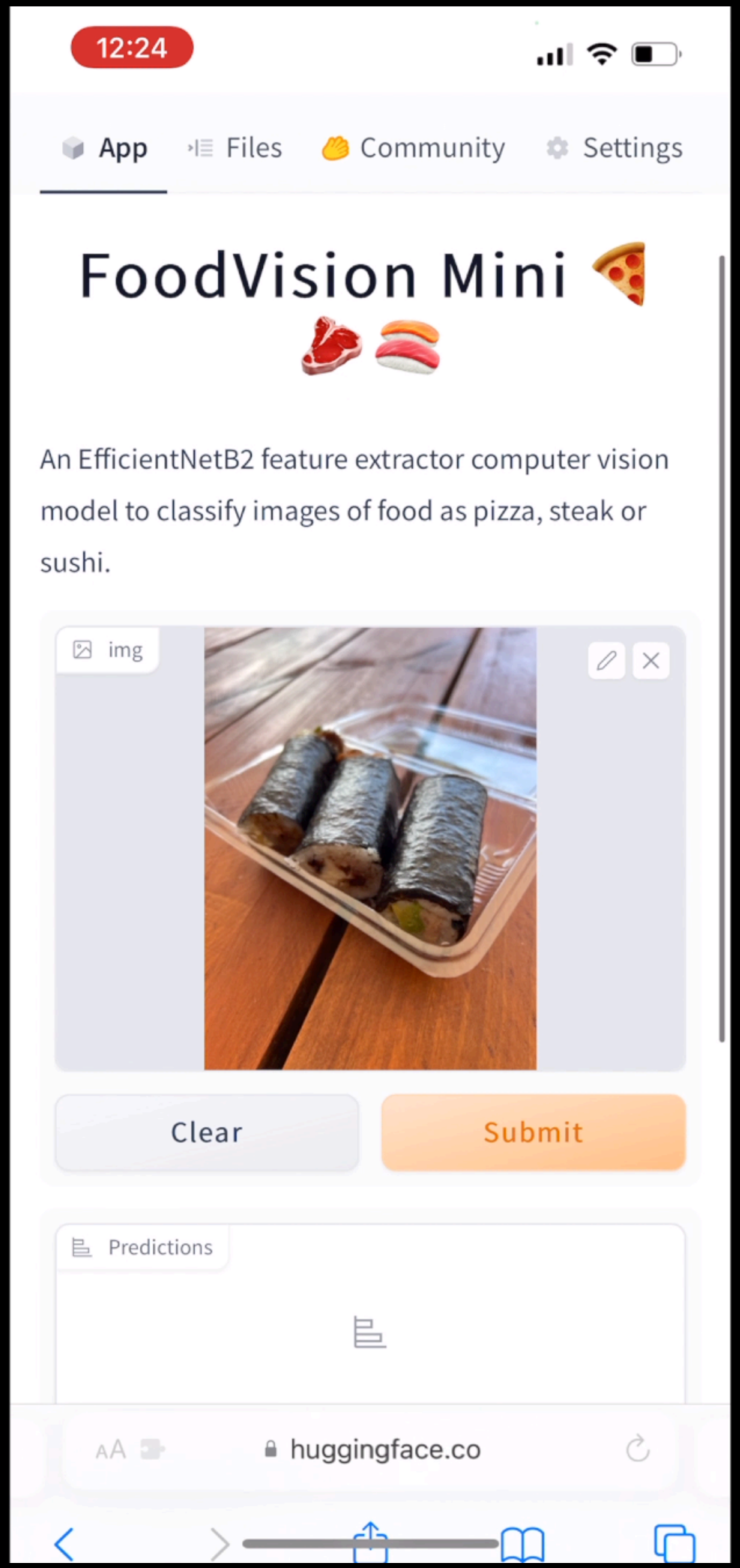
https://huggingface.co/spaces/mrdbourke/foodvision_mini

What we're doing

Deploying our FoodVision Mini machine learning model







What is my most ideal machine learning model deployment scenario?

FoodVision Mini 🍕🥩🍣

https://huggingface.co/spaces/mrdbourke/foodvision_mini

What we’re going to cover

(broadly)

- Getting setup (importing previously written code)
- Introduce machine learning model deployment with PyTorch
- Deploy FoodVision Mini 🍕🥩🍣 as a useable web application
- Experimenting with multiple models (EffNetB2 and ViT)
- A BIG surprise!

(we’ll be cooking up lots of code!)

How: 👩‍🍳 👩‍🔬

Let’s code!

FoodVision Mini Deployment Goals

FoodVision Mini 🍕🥩🍣

📈 Performance: 95%+ accuracy
(good)

🏃 Speed: 30FPS+ (real-time)
(fast)

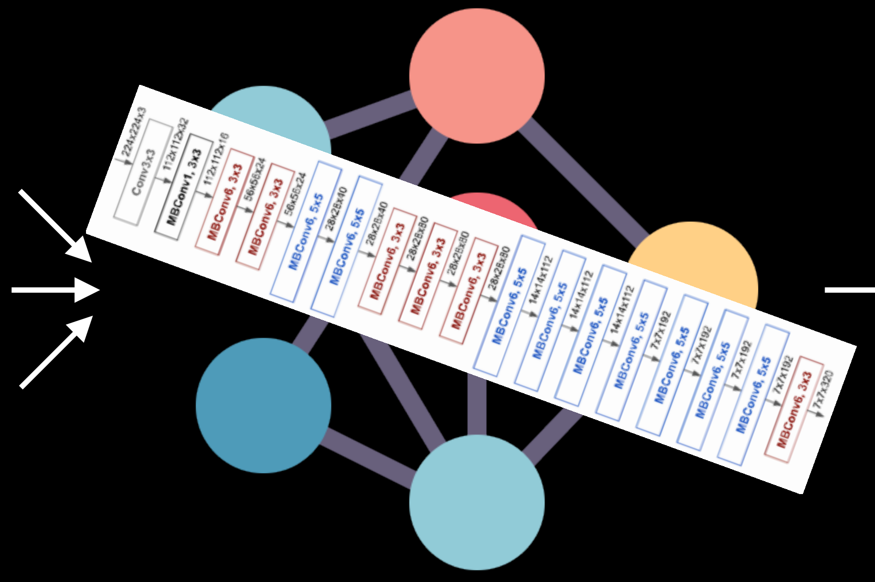
FoodVision Mini Deployment Experiments

Goals

Performance: 95%+ accuracy (good)

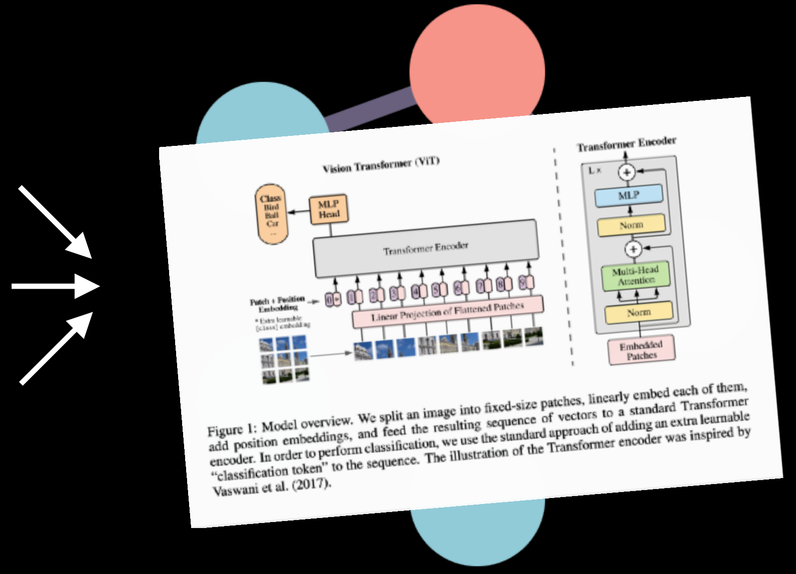
Speed: 30FPS+ (real-time) (fast)

Model 1 (EffNetB2)

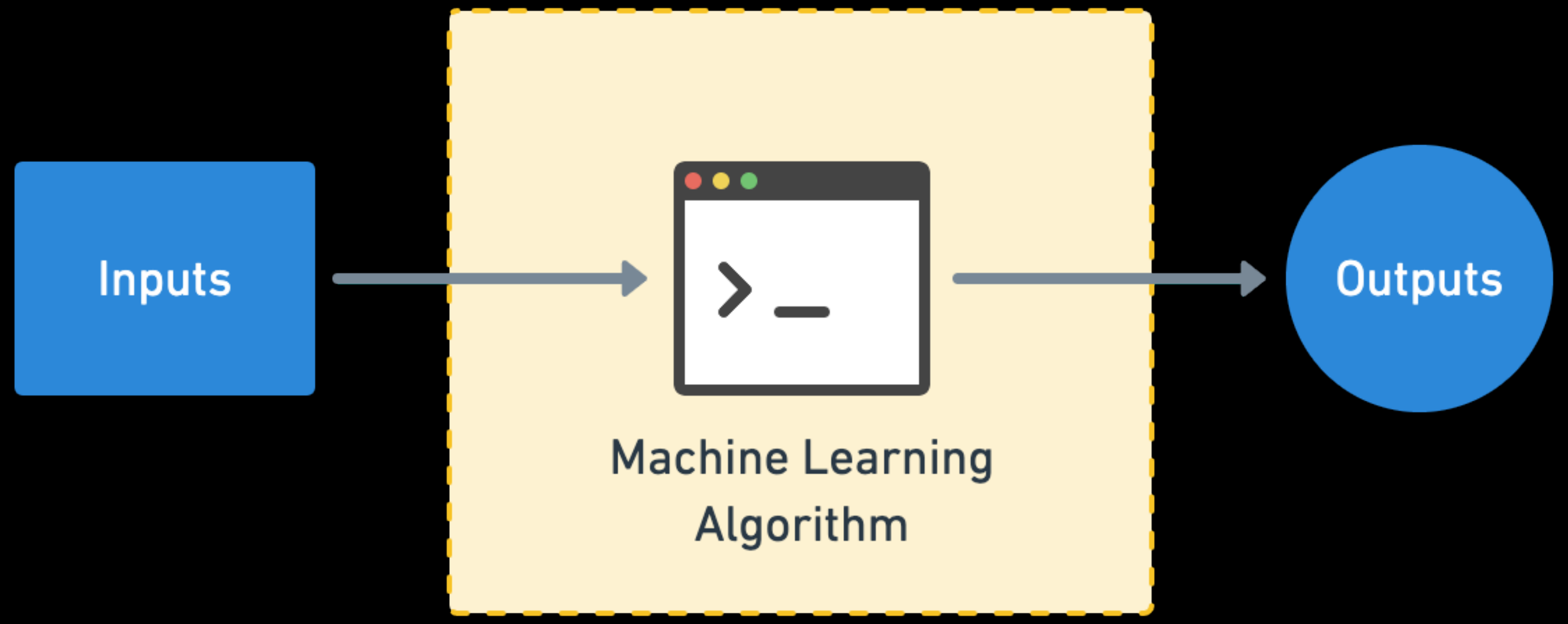


Pizza, steak, sushi 20% data

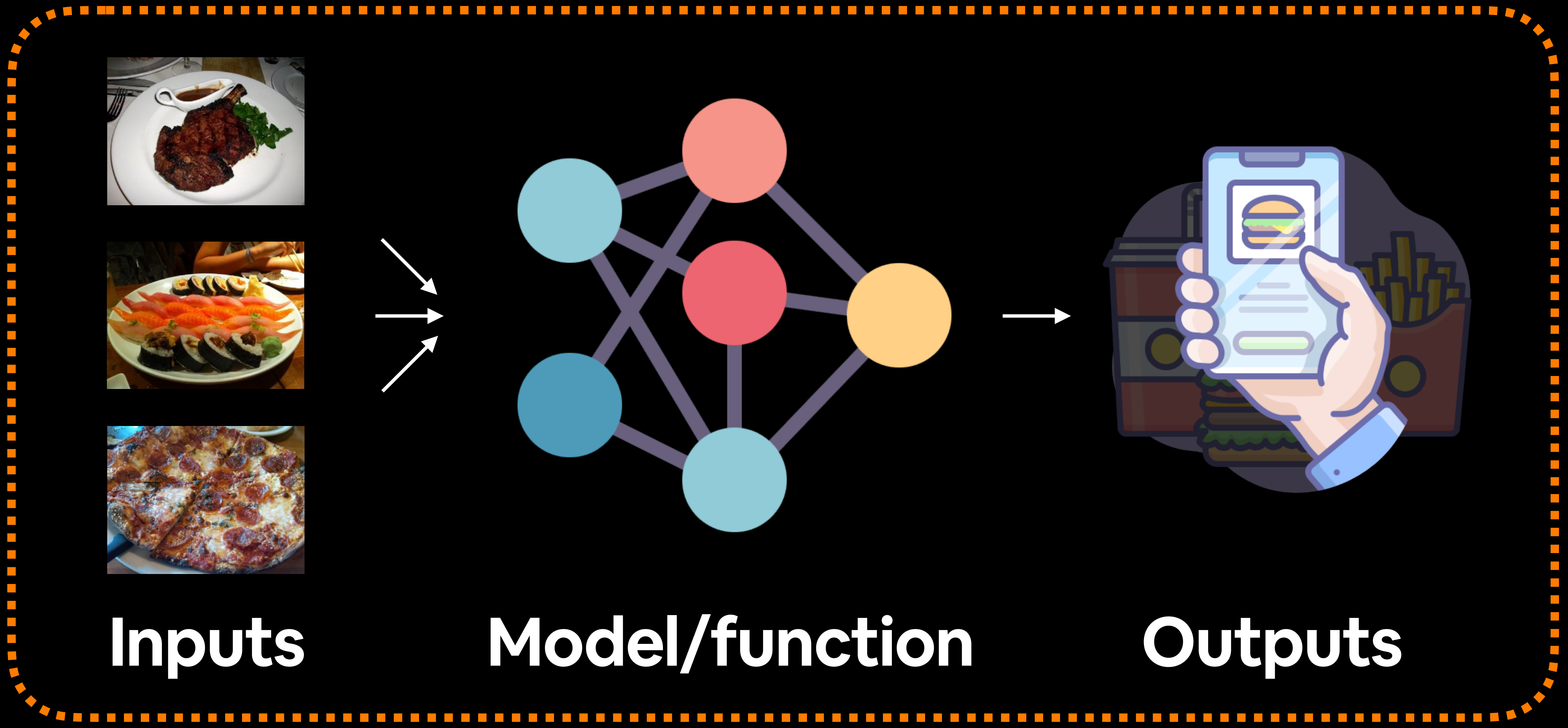
Model 2 (ViT)



Gradio overview



gradio

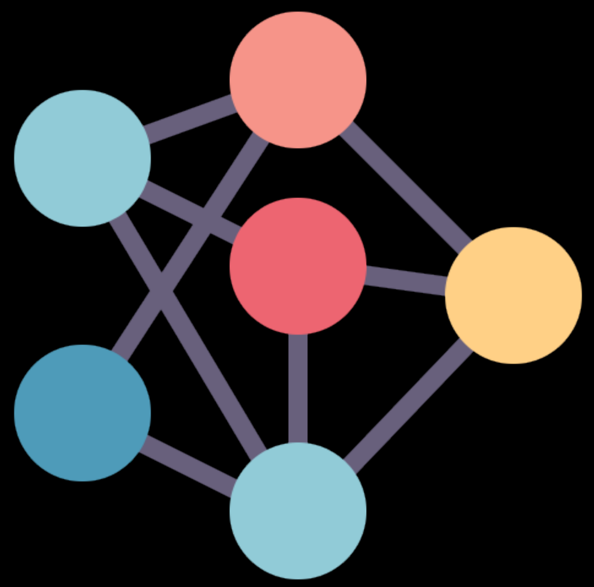


Gradio helps create an interface for this workflow

Gradio overview

gradio

Gradio helps create an interface for this workflow



Daniel Bourke
@mrdbourke

Machine learning the type of field that uses four names for the same thing:

1. Linear layers
2. Dense layers
3. Feedforward layers
4. Fully connected layers

8:49 PM · Aug 4, 2022 · Twitter Web App

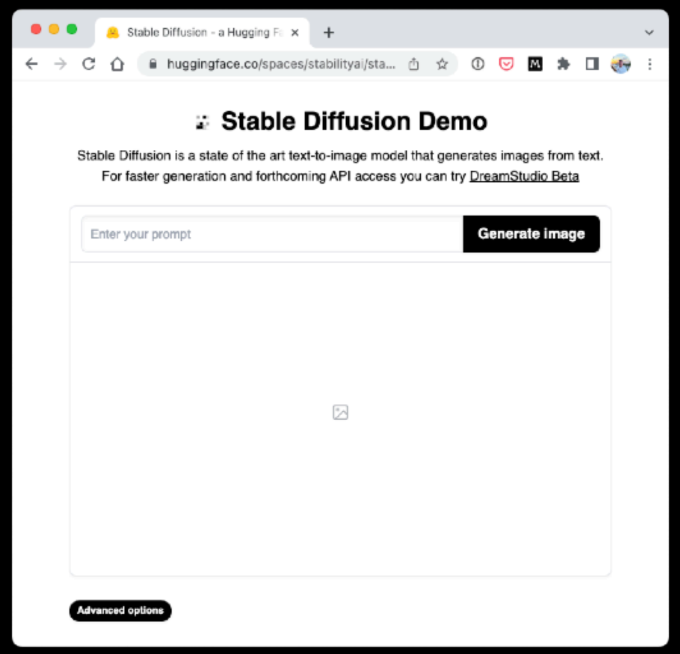
→

```python
def predict(input):
  if "machine learning" in input:
    return "Tweet is about ML, okay to read"
  else:
    return "Tweet is not worth reading"
```

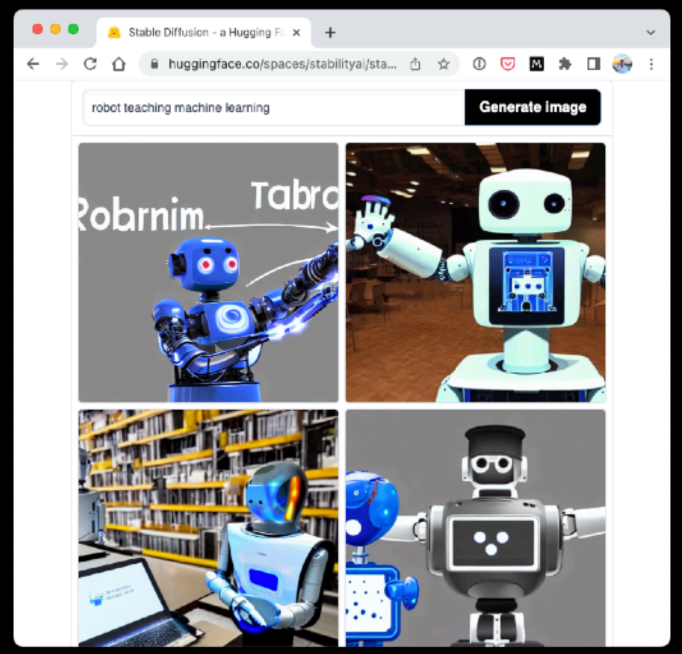
→

✅ Tweet is about ML, okay to read

"robot teaching machine learning" →



→

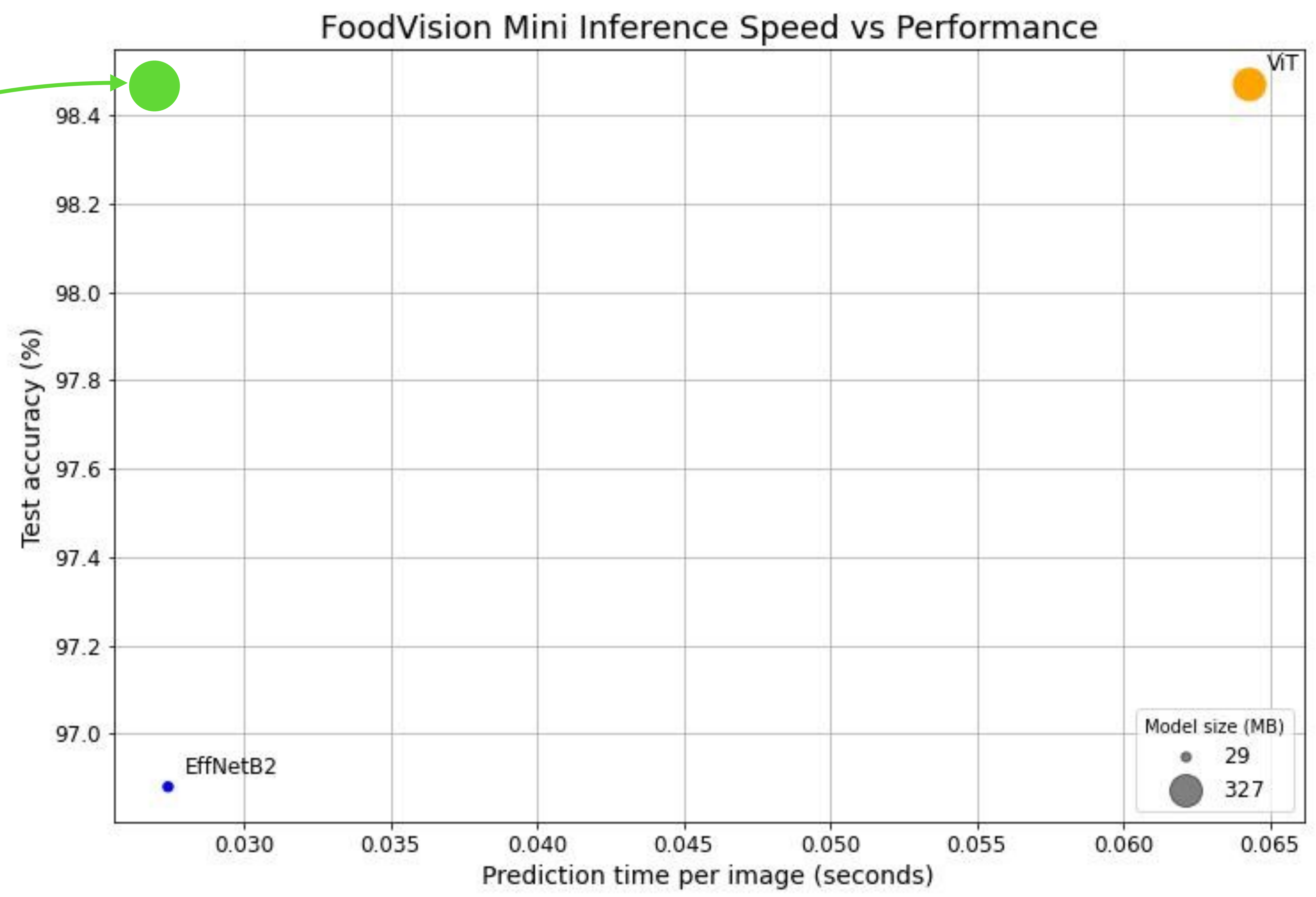


Inputs **Model/function** **Outputs**

See more workflow ideas in the Gradio documentation.

Performance vs. Speed trade-off

Most ideal position



With a larger model comes better performance but generally at the sacrifice of speed.

FoodVision Mini -> FoodVision Big

Datasets we’re using/we’ve used

Notebook(s)	Project name	Dataset	Number of classes	Number of training images	Number of testing images
04, 05, 06, 07, 08	FoodVision Mini (10% data)	Food101 custom split	3 (pizza, steak, sushi)	225	75
07, 08, 09	FoodVision Mini (20% data)	Food101 custom split	3 (pizza, steak, sushi)	450	150
09 (this one)	FoodVision Big (20% data)	Food101 custom split	101 (all Food101 classes)	15150	5050
Extension	FoodVision Big	Food101 all data	101 (all Food101 classes)	75750	25250

Original Food101 dataset from original Food101 paper, see how the splits were created in 04. Custom Data Creation notebook.