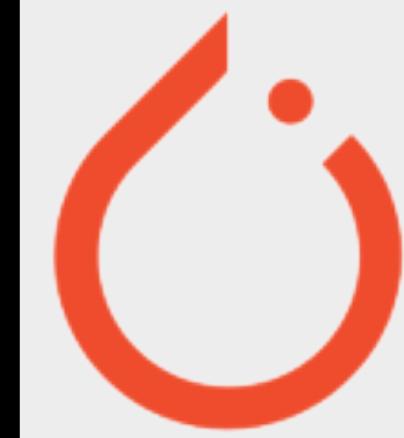


Milestone Project 3

Model Deployment with



PyTorch

Where can you get help?

- Follow along with the code



09. PyTorch Model Deployment

Welcome to Milestone Project 3: PyTorch Model Deployment!

We've come a long way with our FoodVision Mini project.

But so far our PyTorch models have only been accessible to us.

How about we bring FoodVision Mini to life and make it publicly accessible?

In other words, we're going to deploy our FoodVision Mini model to the internet as a usable app!

FoodVision Mini 🍔🍕

- Try it for yourself

"If in doubt, run the code"

- Press SHIFT + CMD + SPACE to read the docstring

```
def create_vit_model(num_classes=1000, seed=42):
    """Creates a ViT-B/16 feature extractor.
    Args:
        num_classes (int, optional): number of classes
        seed (int, optional): random seed
    Returns:
        model (torch.nn.Module): ViT-B/16
        transform (torchvision.transforms.Compose)
    """
    # Create ViTModel2 pre-trained weights
    weights = torchvision.models.ViT_B_16_Weights.DEFAULT
    transform = weights.transforms()
    model = torchvision.models.vit_b_16(weights=weights)

    # Freeze all layers in model
    for param in model.parameters():
        param.requires_grad = False
    # Change classifier head to suit our needs (this will be trainable)
    torch.manual_seed(seed)
    model.heads = nn.Sequential(nn.Linear(in_features=768, out_features=num_classes)) # update to reflect target number of classes
    return model, transform
```

- Search for it



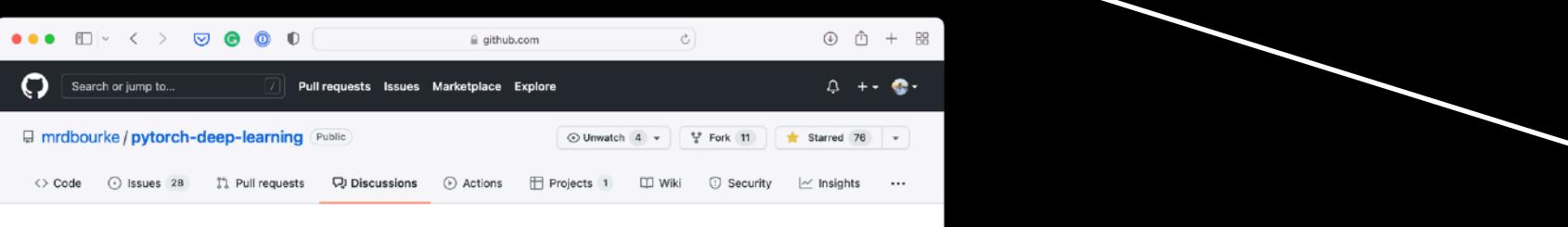
PyTorch Documentation

PyTorch is an optimized tensor library for deep learning using GPUs and CPUs.

Features described in this documentation are classified by release status:

- Stable: These features will be maintained long-term and there should generally be no major performance limitations or gaps in documentation. We also expect to maintain backwards compatibility (although breaking changes can happen and notice will be given one release ahead of time).
- Beta: These features are tagged as Beta because the API may change based on user feedback, because the performance needs to improve, or because coverage across operators is not yet complete. For Beta features, we are committing to seeing the feature through to the Stable classification. We are not, however, committing to backwards compatibility.
- Prototype: These features are typically not available as part of binary distributions like PyPI or Conda, except sometimes behind run-time flags, and are at an early stage for feedback and testing.

- Try again



- Ask

Do you want live notifications when people reply to your posts? Enable Notifications

all categories > Latest New (66) Unread (11) Top

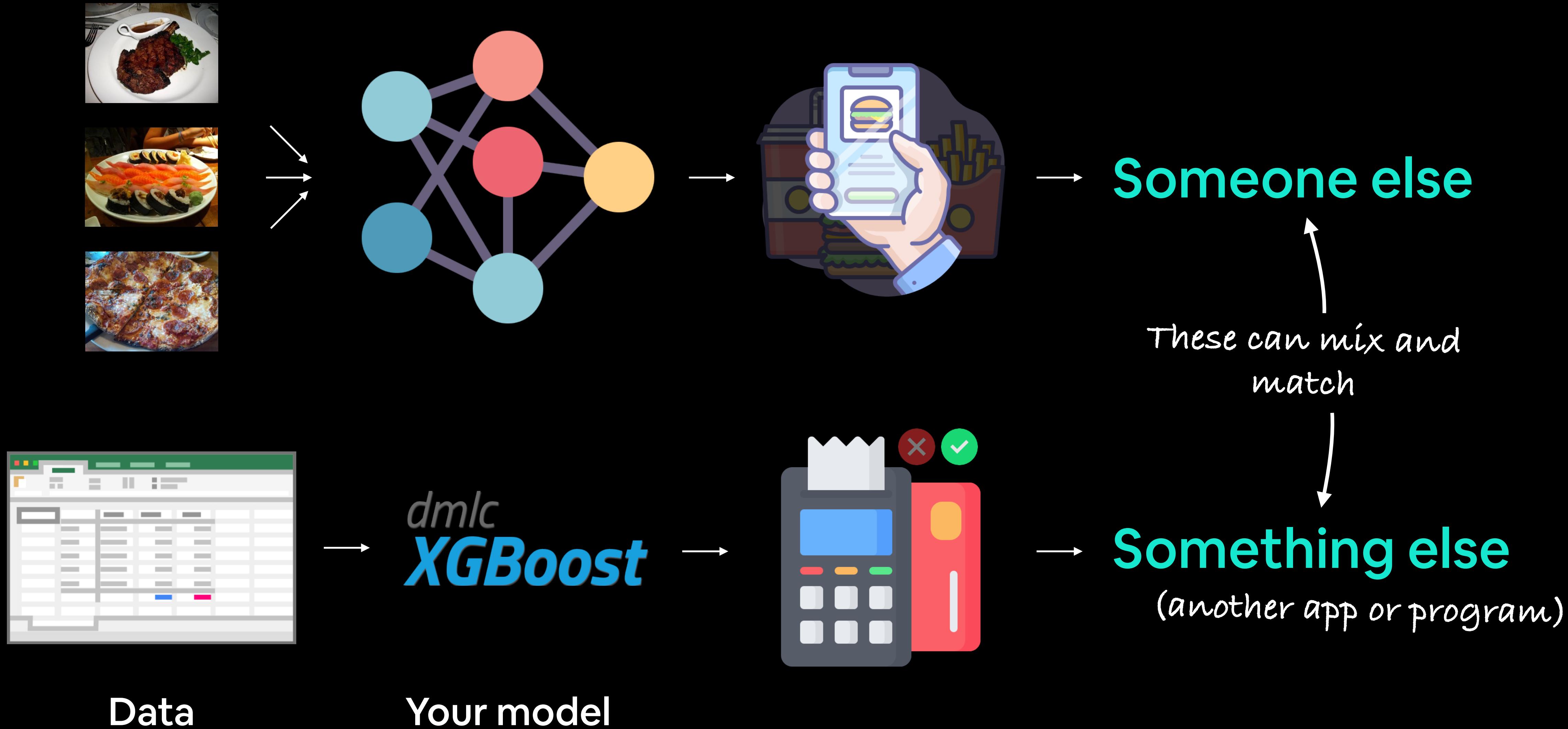
Category Topics Latest

- Uncategorized 375 / month
- mixed-precision 5 / month
- autograd 58 / month
- windows 2 / month
- data 49 / month

“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?

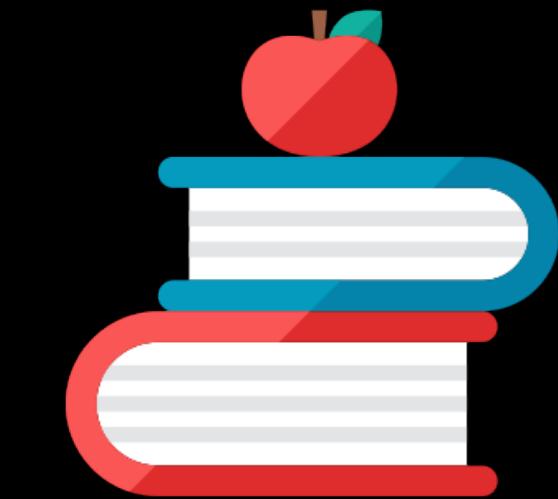
Three datasets

(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)**

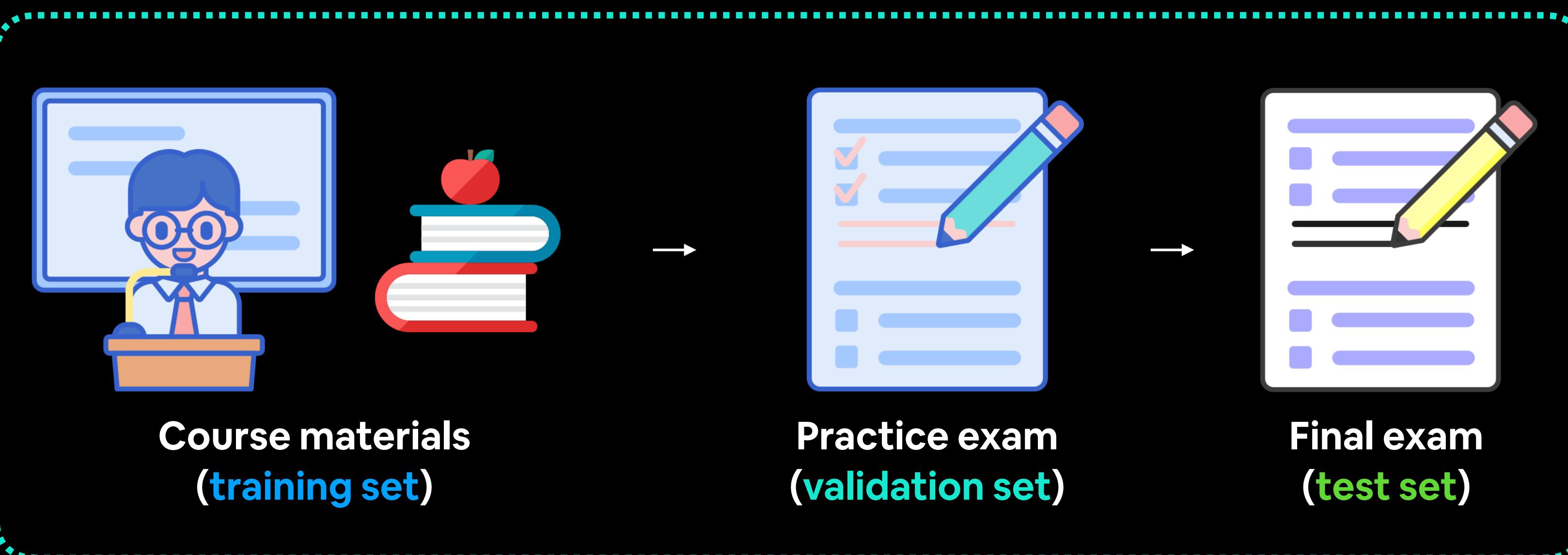
See if the model is ready for the wild

Generalization

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

FOUR ~~Three~~ datasets

Notebook/local environment

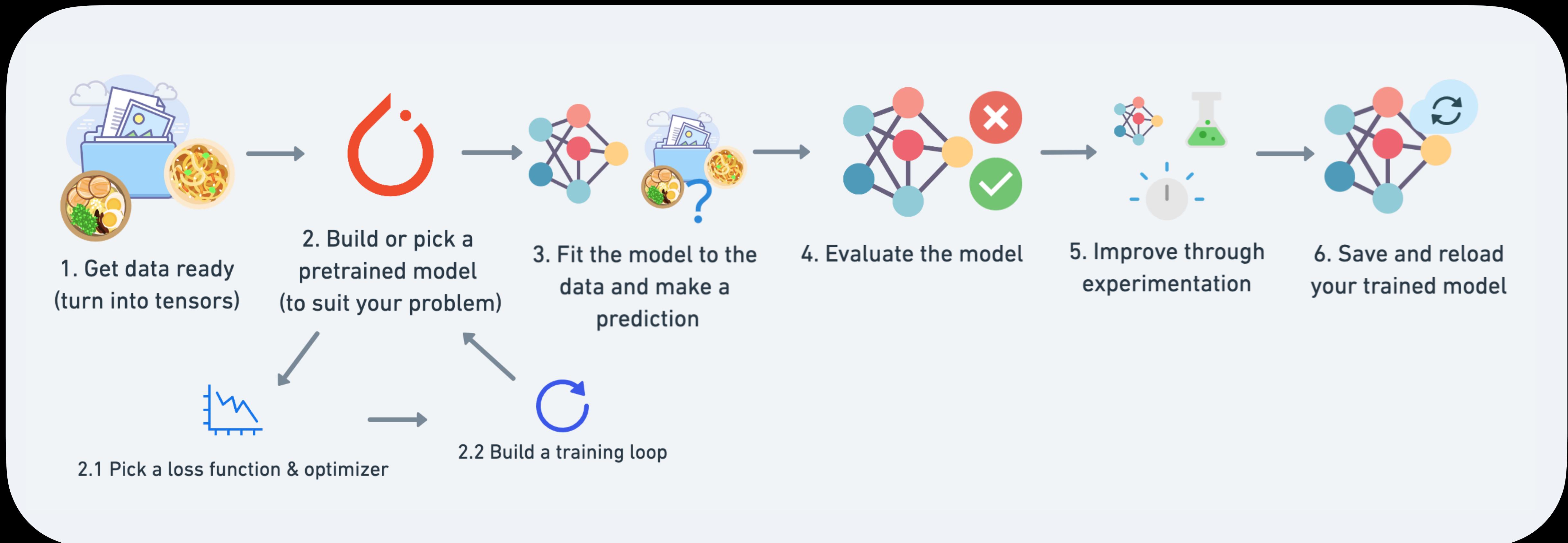


Real world

(you can only test here by deploying!)

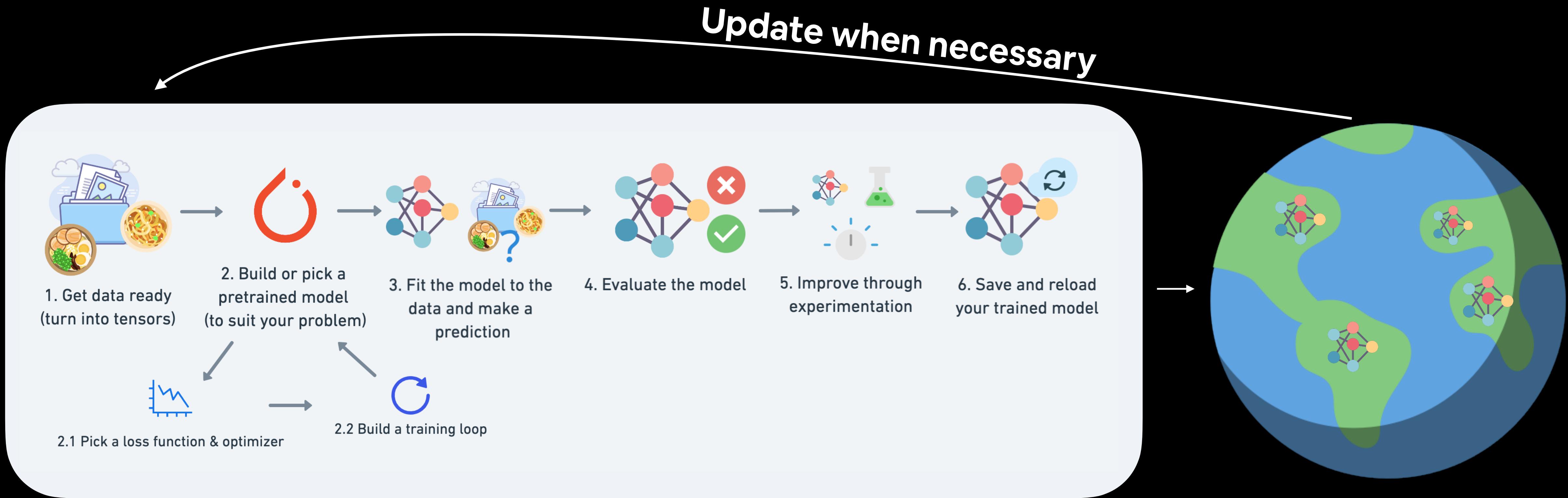
A PyTorch workflow

(one of many)



A PyTorch workflow

(one of many)



X. Deploy & Monitor

```
59 Machine Learning Engineer*
60 -----
61
62 1. Download a paper
63 2. Implement it
64 3. Keep doing this until you have skills
```

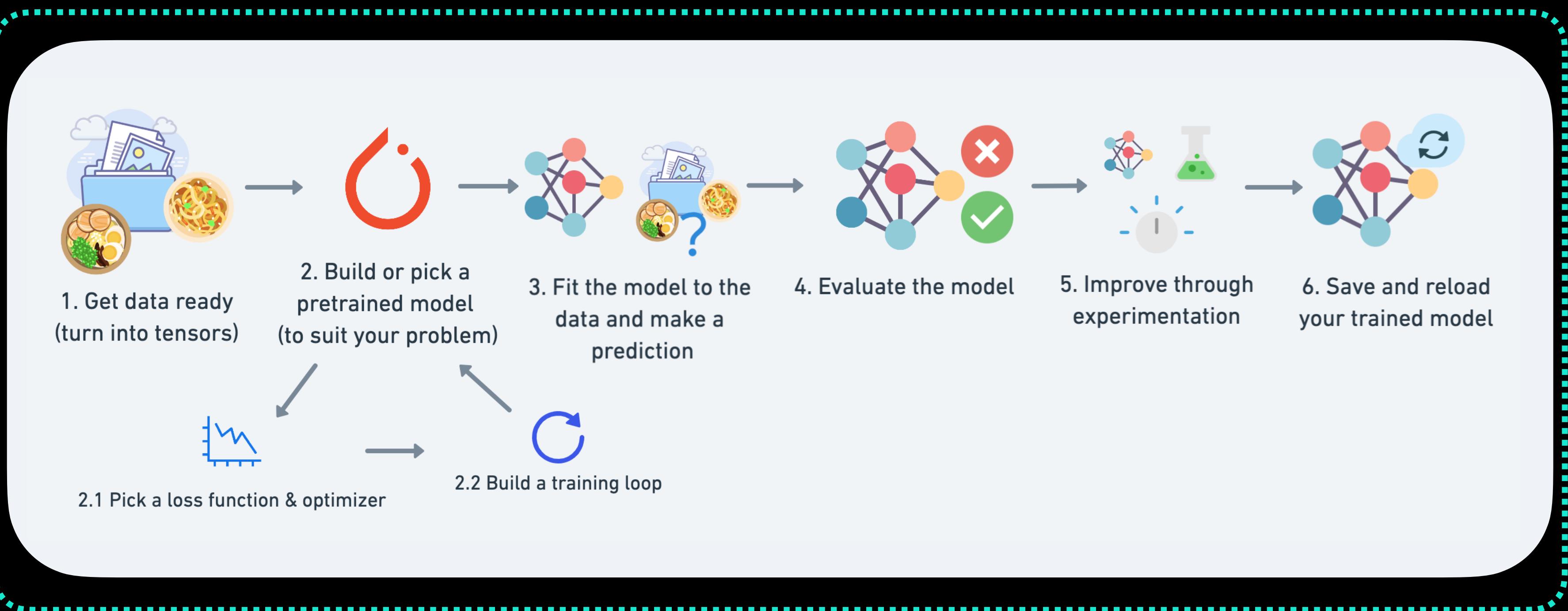
- George Hotz, founder of comma.ai

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

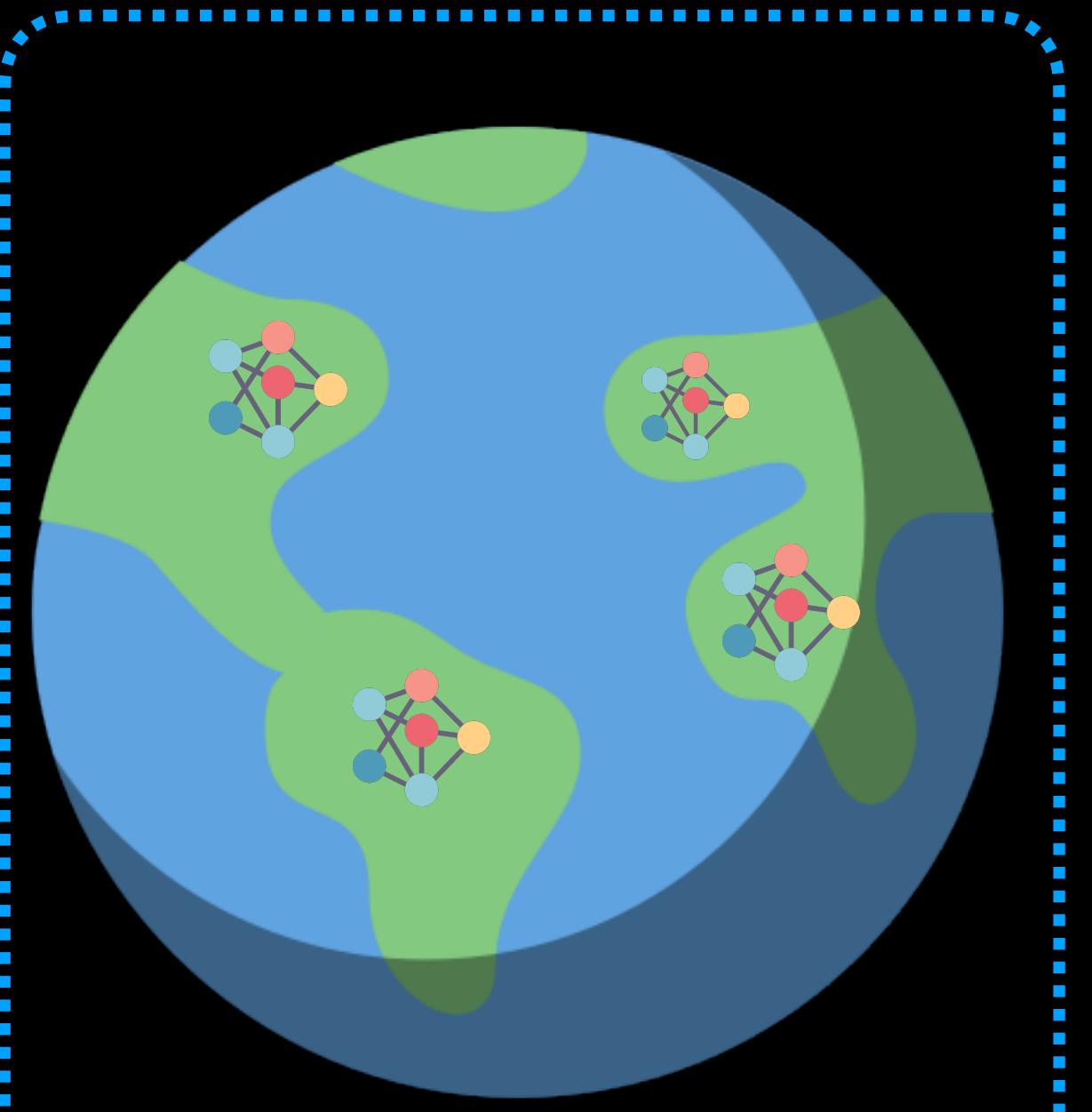
A PyTorch workflow

(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 ideal 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

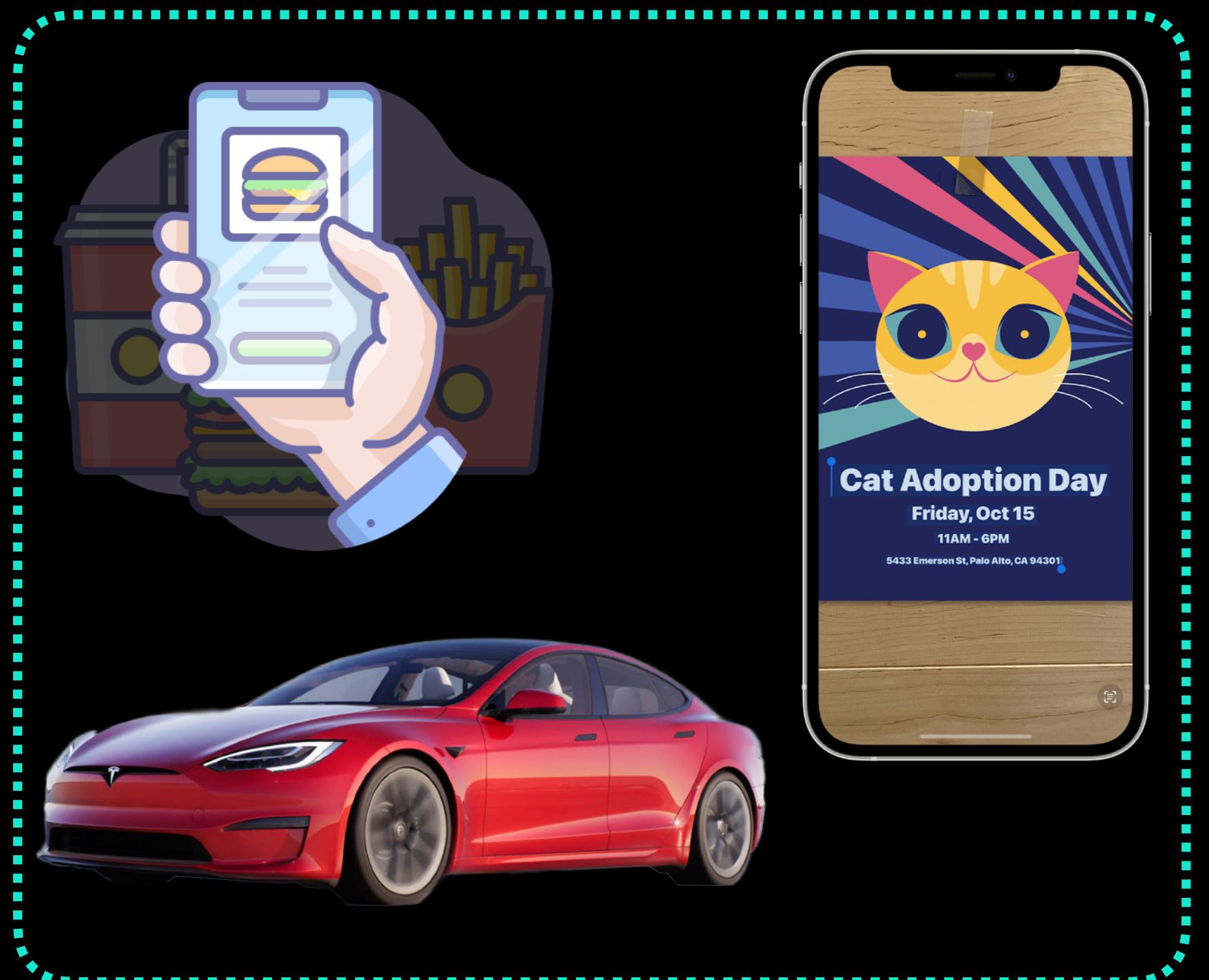
These can mix and match

How's my model going to function?

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

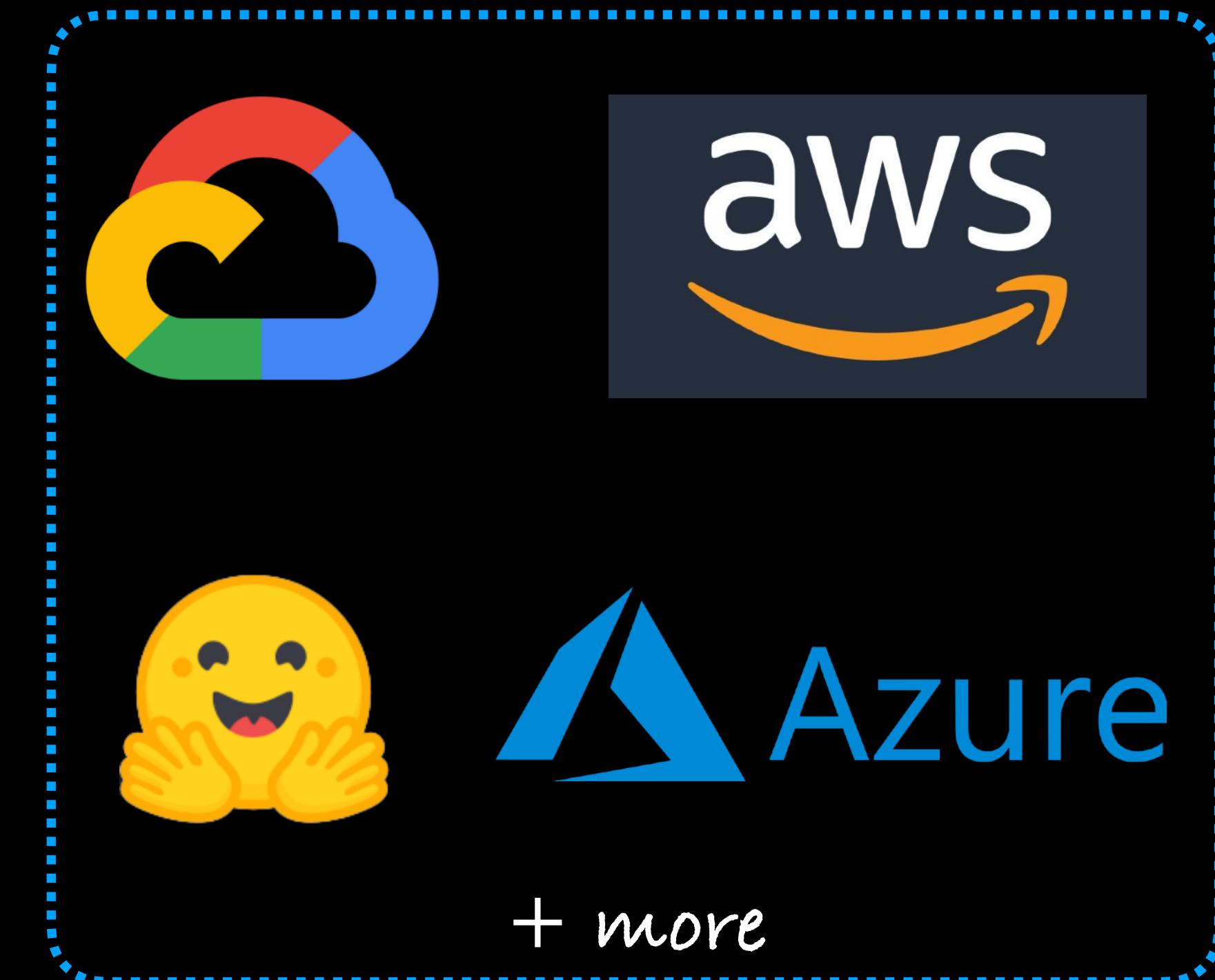
Deployment questions to ask

Where's my model going to go?



On-device (edge)

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

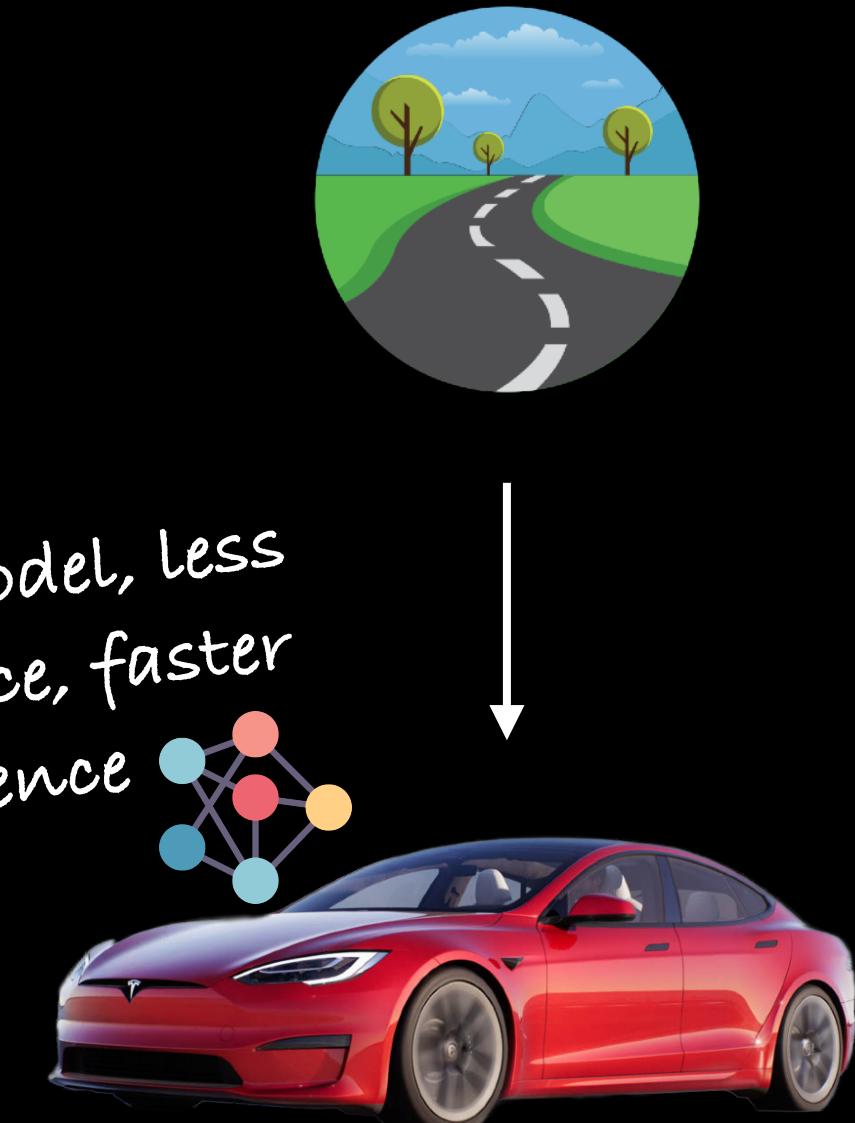


Cloud

Where's my model going to go?

On-device (edge)

smaller model, less performance, faster inference



vs.



Source: [Dirty Tesla YouTube channel](#)

Cloud

bigger model, better performance, slower inference



Azure

+ more



Source: [Dirty Tesla YouTube channel](#)

Where's my model going to go?

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

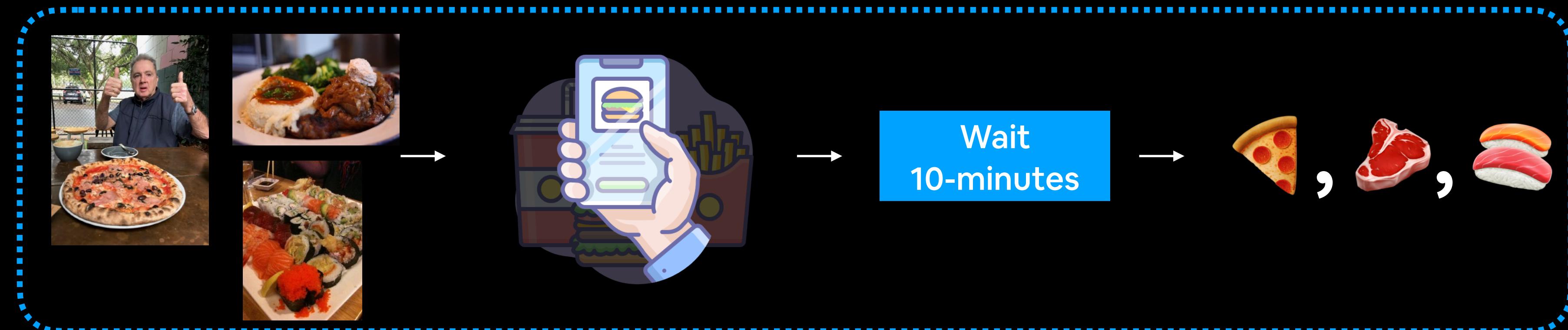
How's my model going to function?

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

Online (real-time)
Predictions happen
immediately



Offline (batch)
Predictions come
at a delay

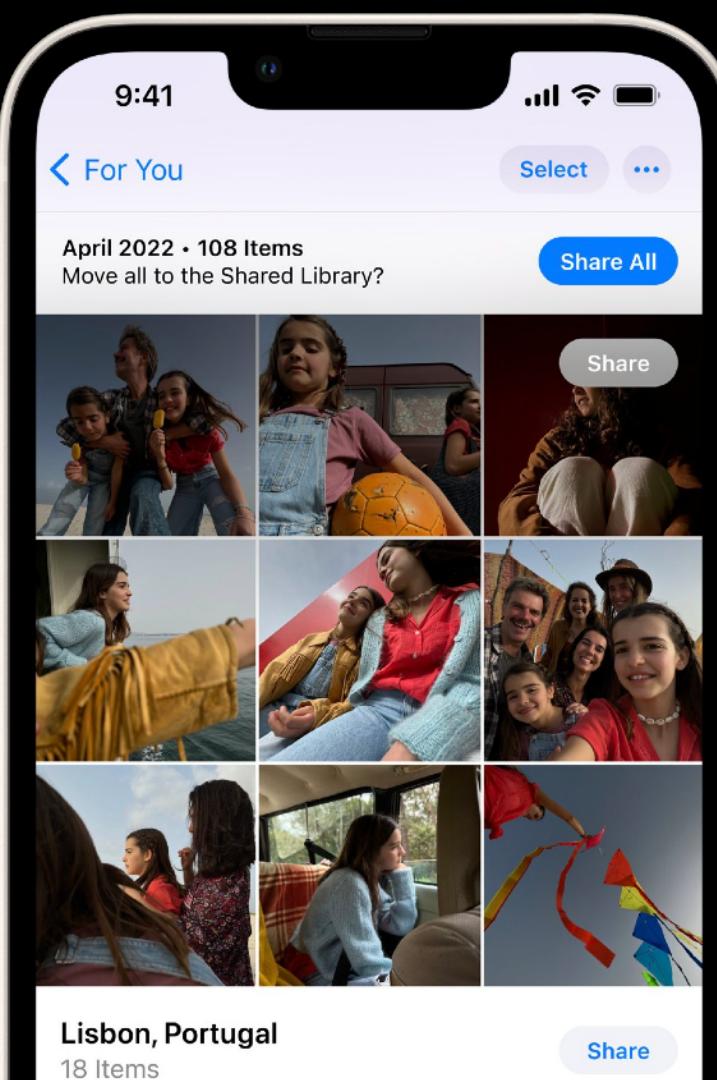


How's my model going to function?

		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 throughput (such as making predictions/training on many samples at a time)		
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

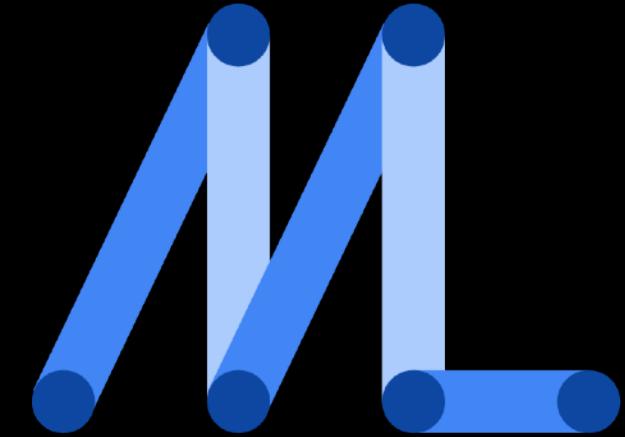


Foodvision Mini: Classify photos into 🍕, 🥩, 🍣 **online** (as soon as photo is uploaded)



(some) Places/tools to help deploy machine learning models

On-device (mobile/edge)



[Google's MLKit](#)

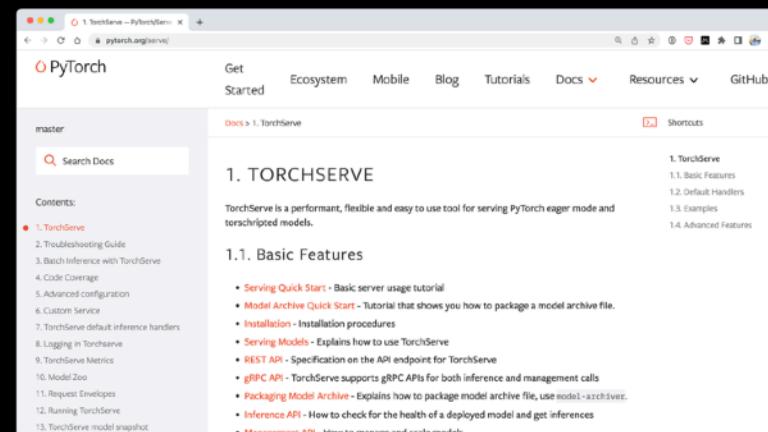


[Apple's CoreML](#)

General



[FastAPI](#)



[TorchServe](#)



[ONNX](#)

[ONNX \(Open Neural Network Exchange\)](#)

Cloud



[Google Cloud Vertex AI](#)



[AWS Sagemaker](#)



[Azure Machine Learning](#)



[Hugging Face](#)

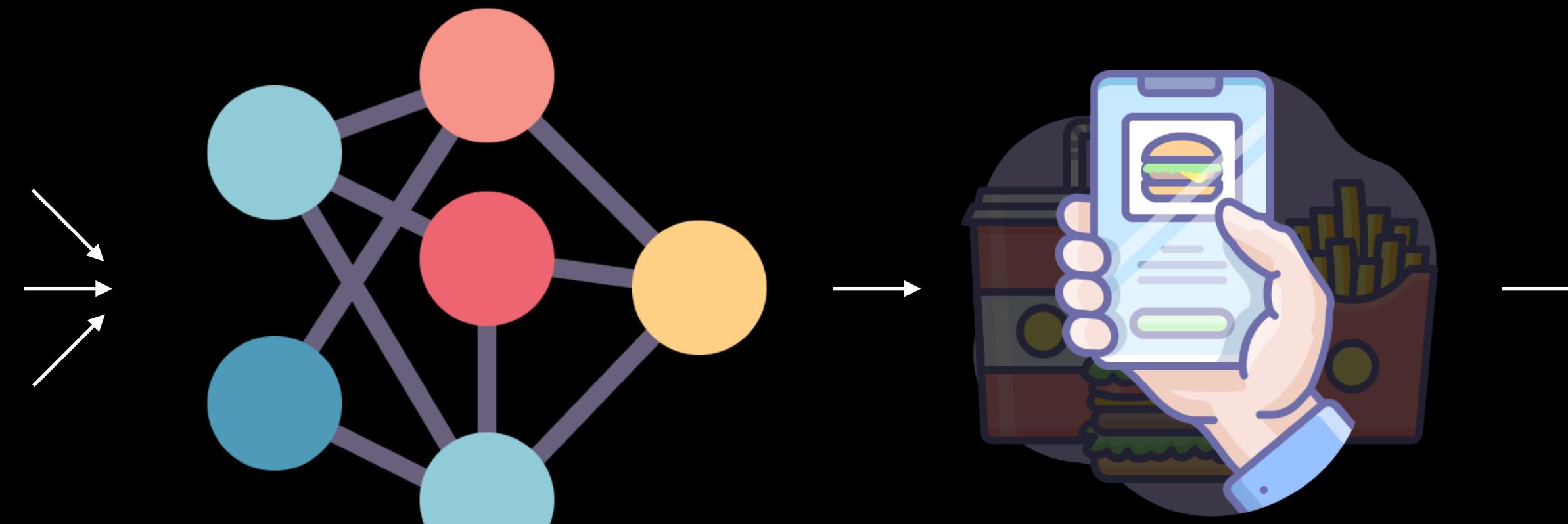
Bonus



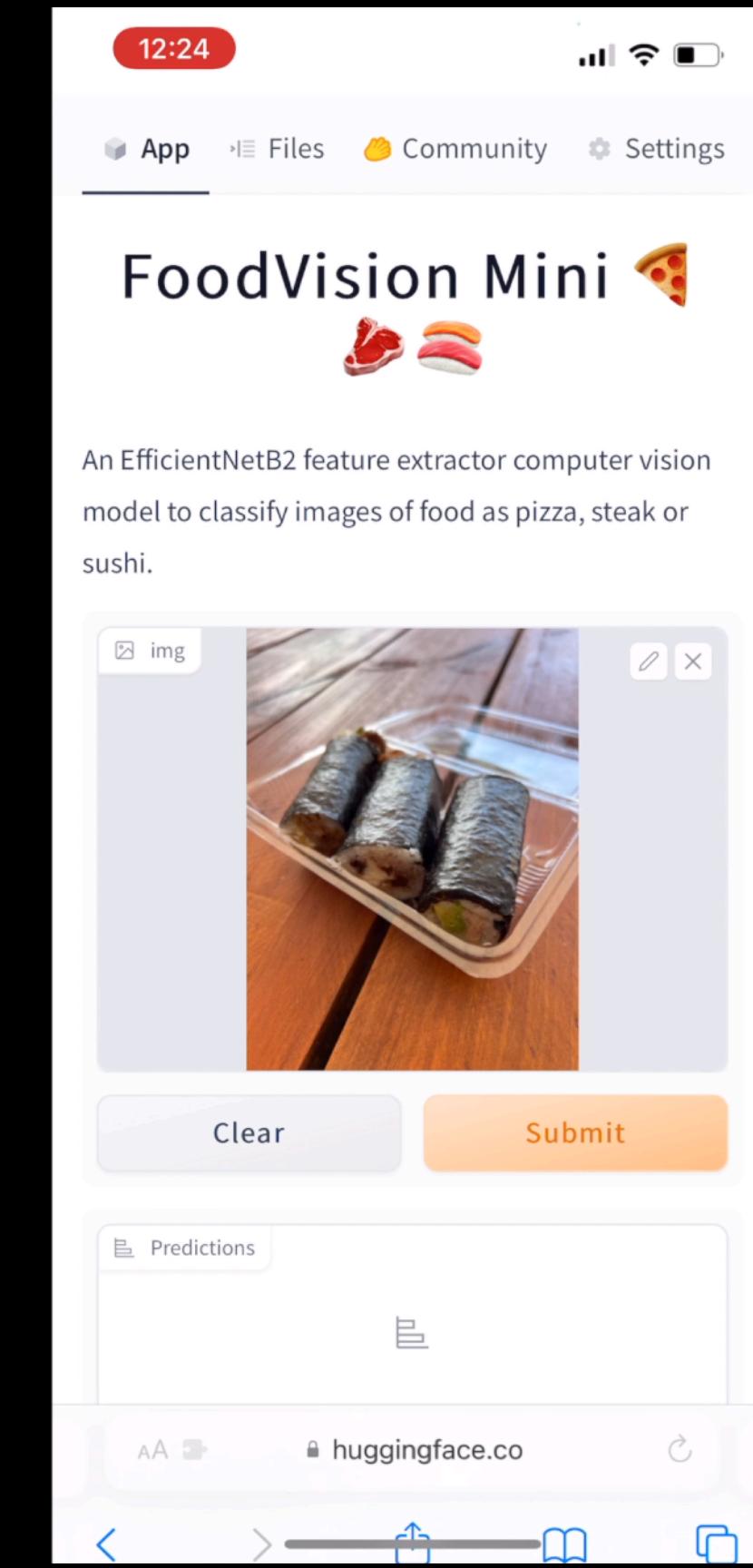
[Gradio](#)

What we're doing

Deploying our FoodVision Mini machine learning model



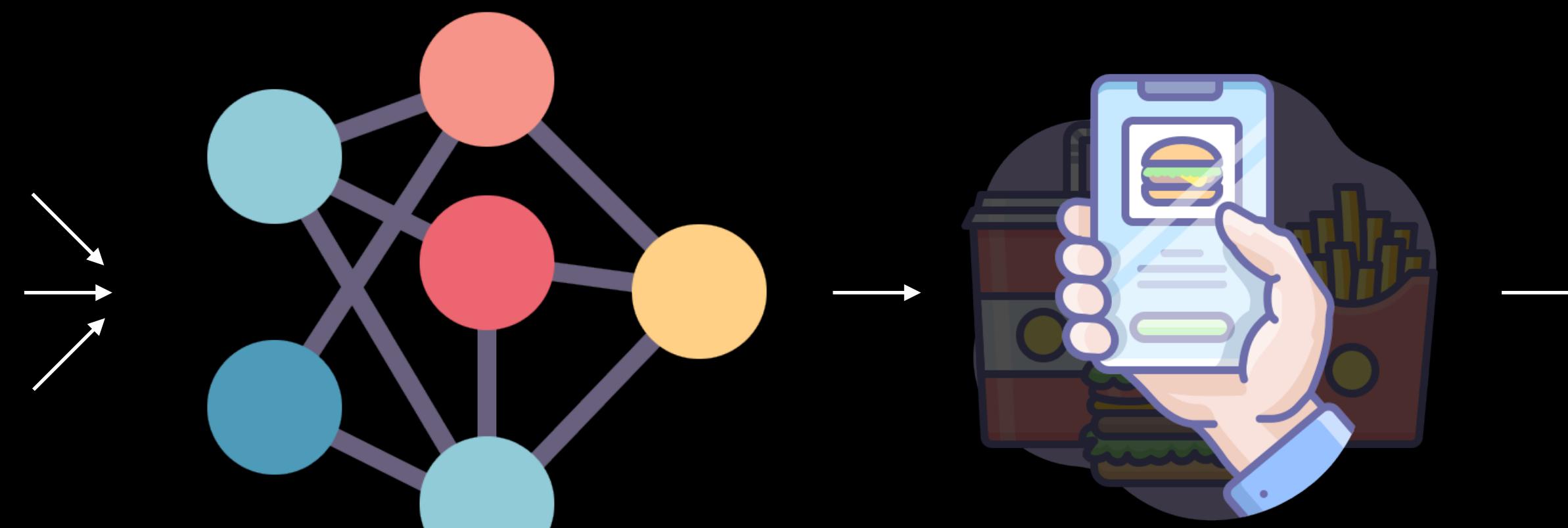
FoodVision Mini 🍕🥩🍣



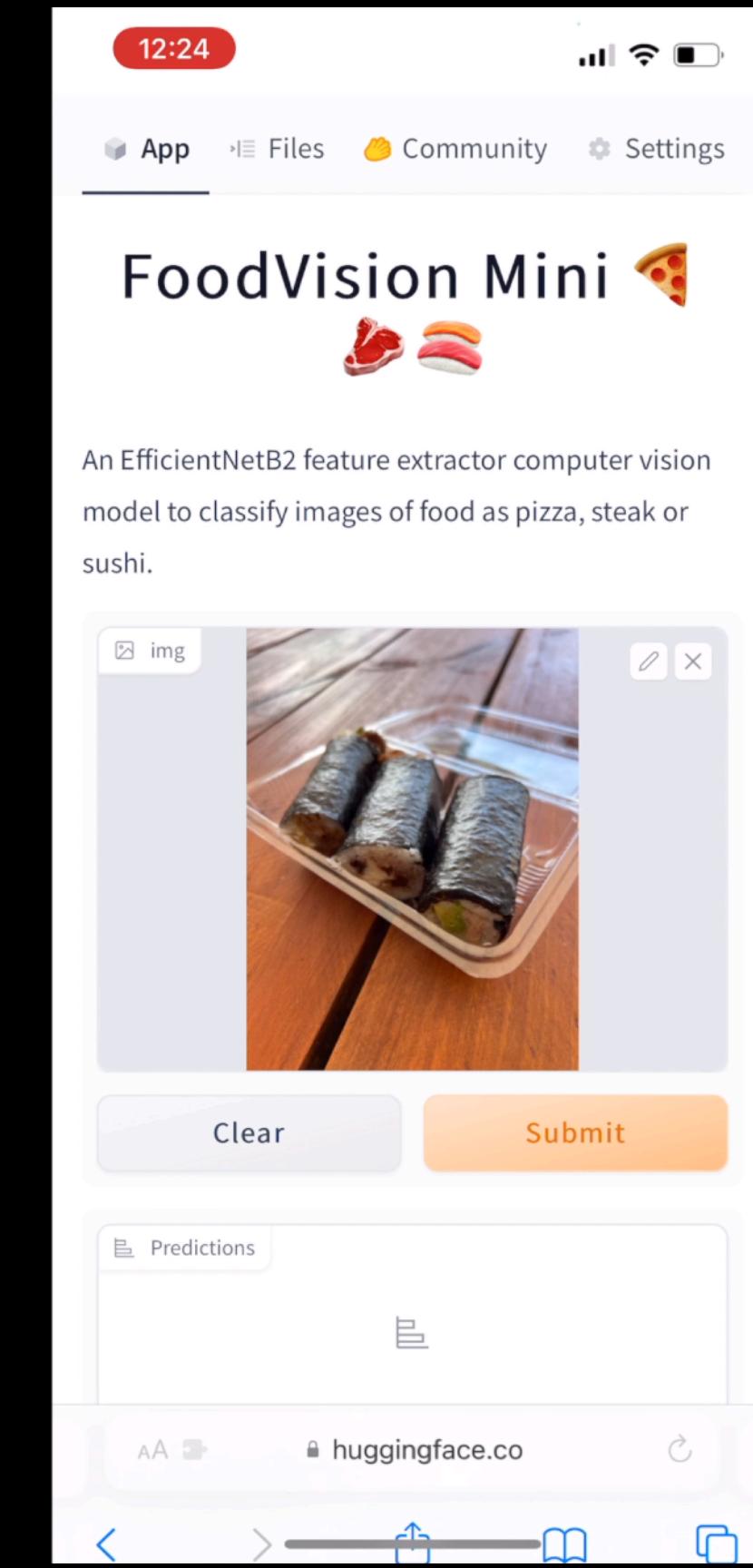
https://huggingface.co/spaces/mrdbourke/foodvision_mini

What we're doing

Deploying our FoodVision Mini machine learning model



FoodVision Mini 🍕🥩🍣



What is my most ideal machine learning model deployment scenario?

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!)

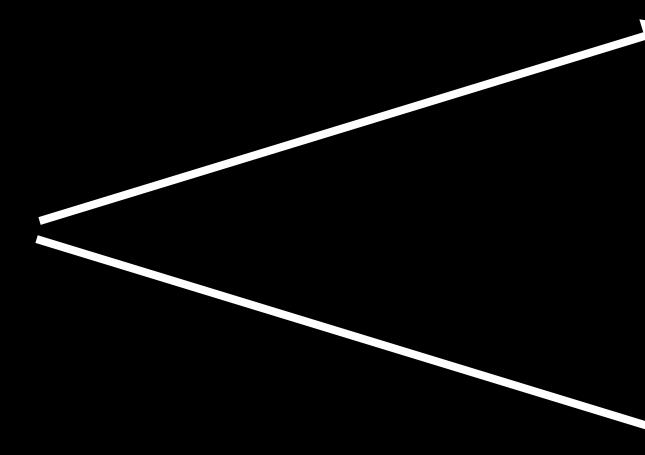
How:



Let's code!

FoodVision Mini Deployment

Goals



📈 Performance: 95%+ accuracy
(*good*)

🏃 Speed: 30FPS+ (real-time)
(*fast*)

FoodVision Mini 🍕🥩🍣

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**)

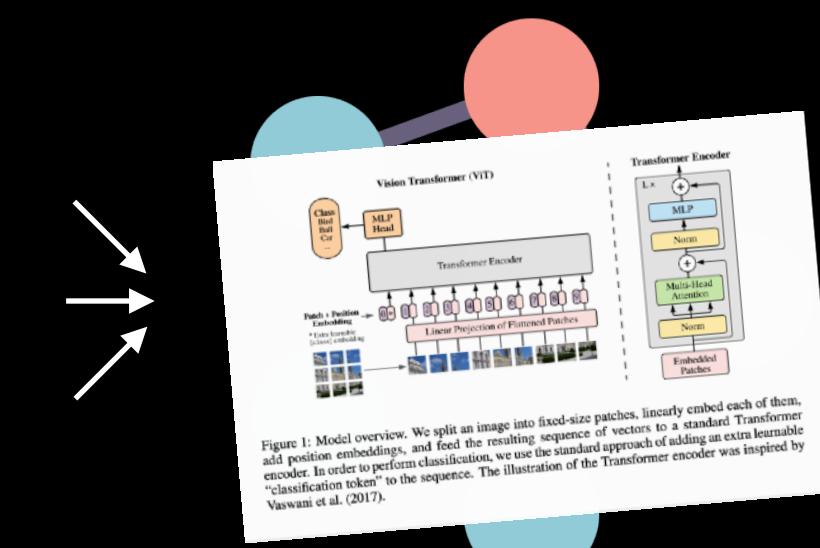
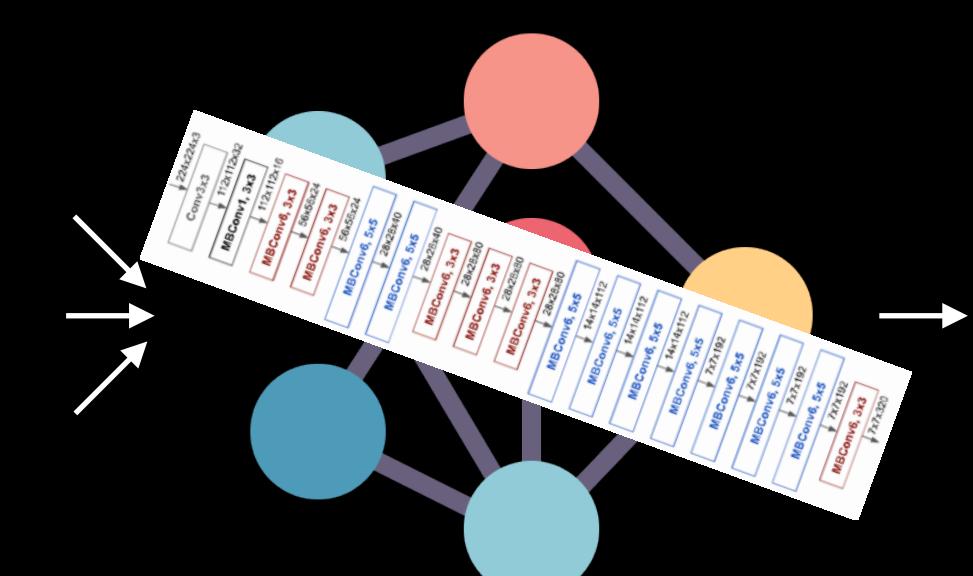
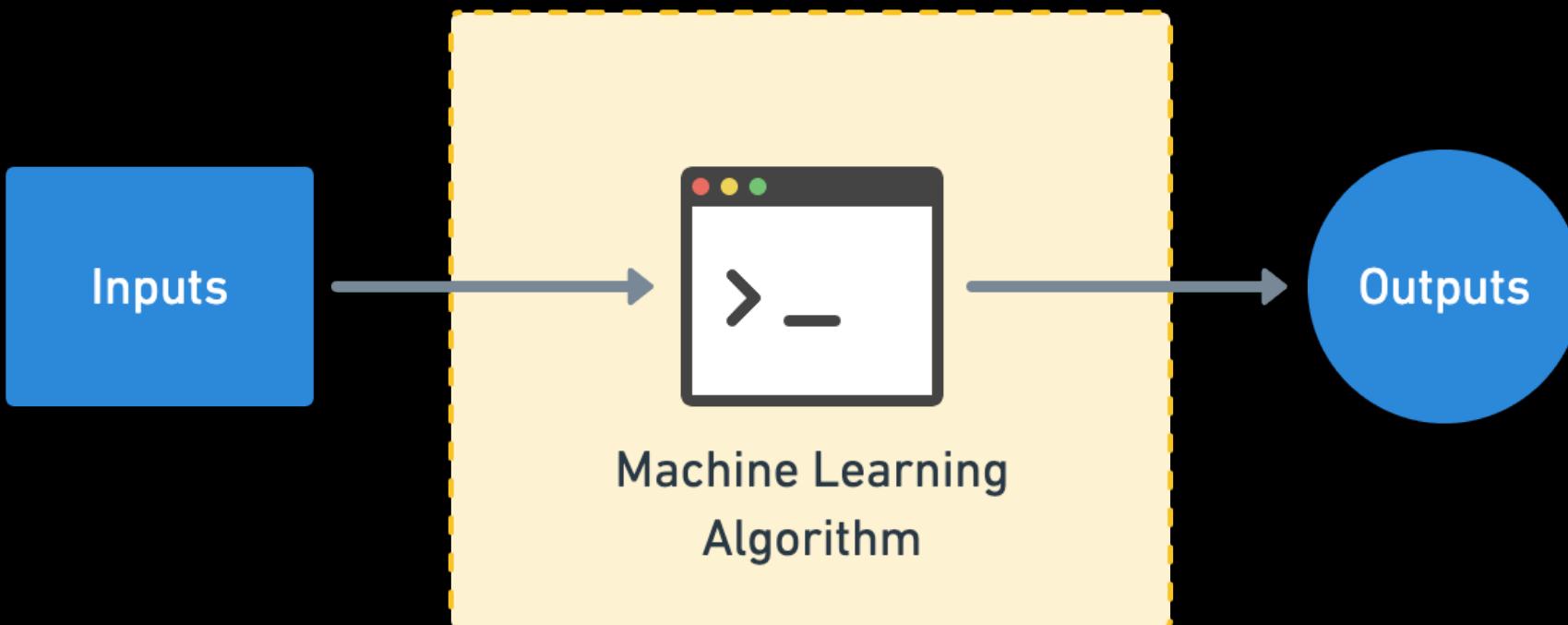
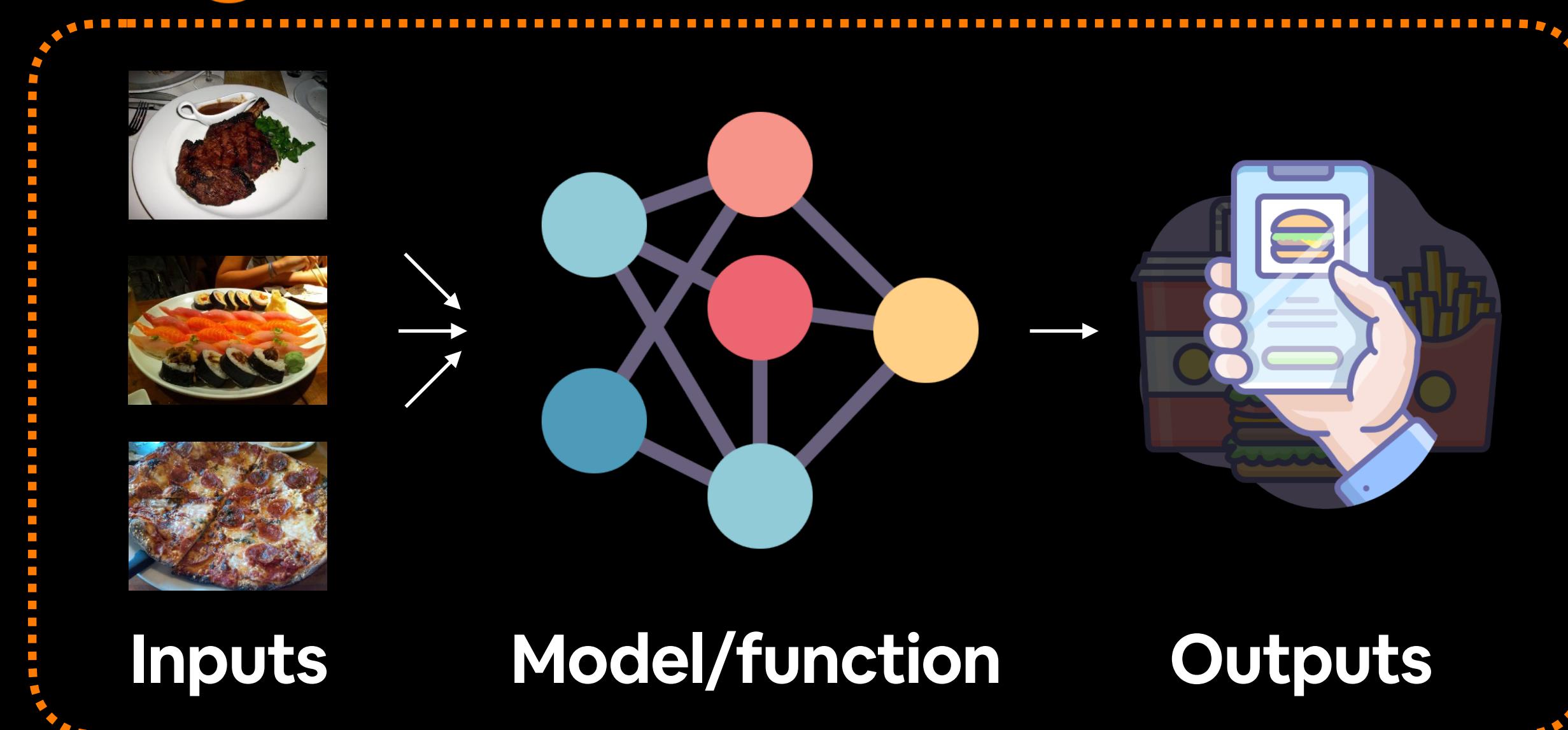


Figure 1: Model overview. We split an image into fixed-size patches, linearly embed each of them, add position embeddings, and feed the resulting sequence of vectors into a standard Transformer encoder. In order to do multi-class classification, we use the standard approach of adding an extra learnable "classification token" to the sequence. The illustration of the Transformer encoder was inspired by Varwan et al. (2017).

Gradio overview



 gradio

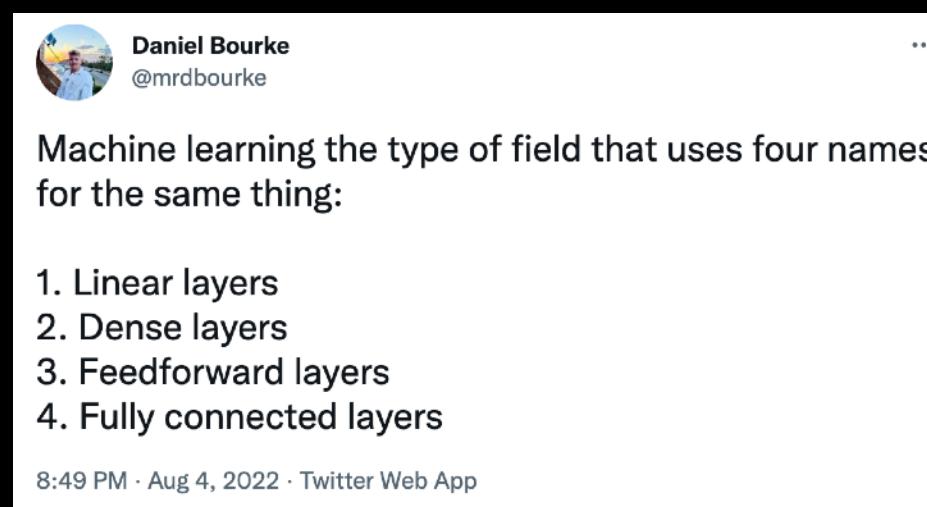
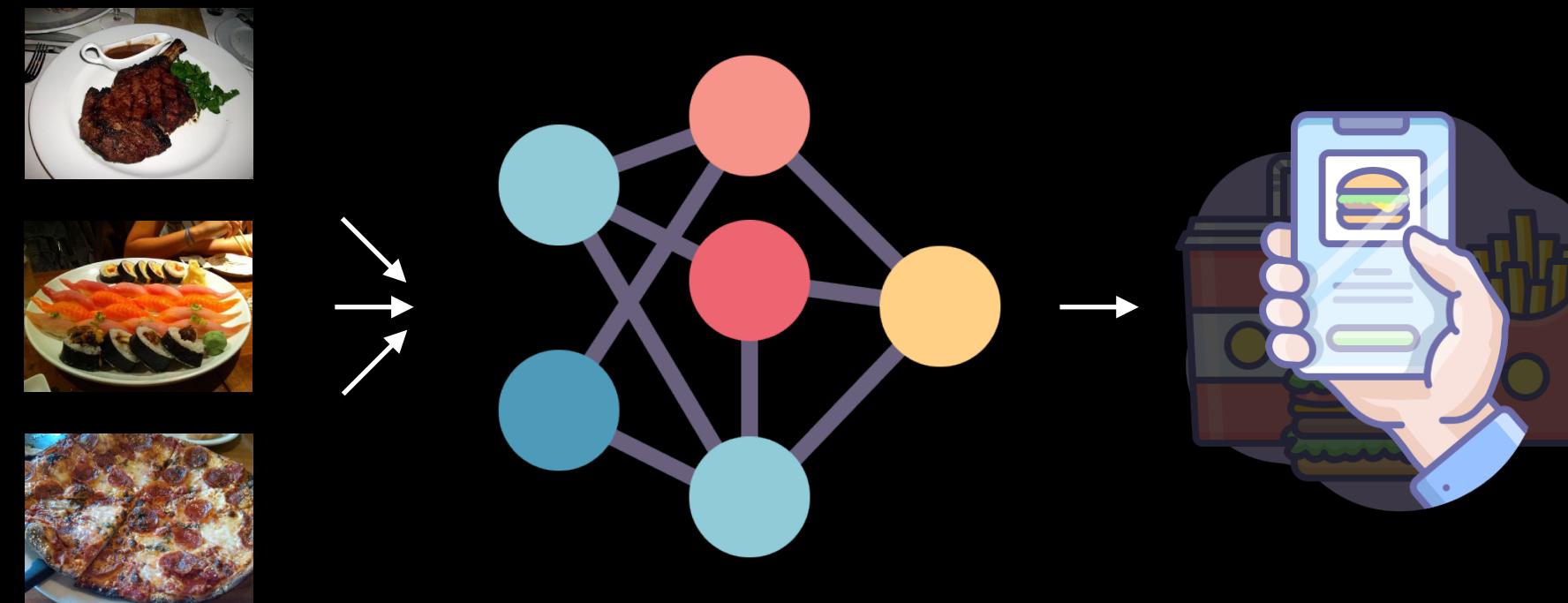


Gradio helps create an interface for this workflow

Gradio overview



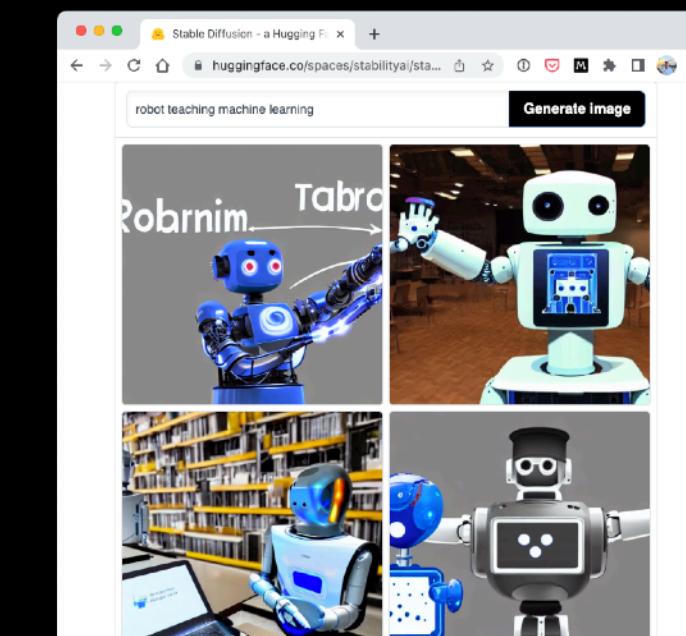
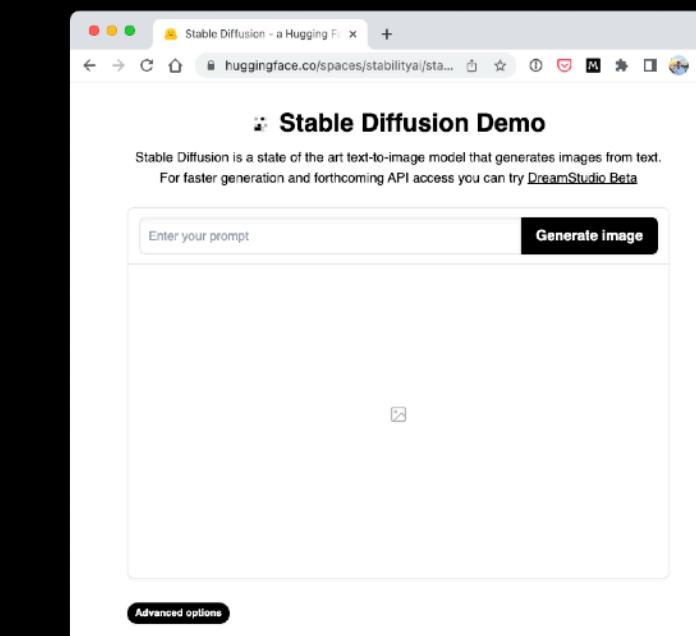
Gradio helps create an interface for this workflow



```
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

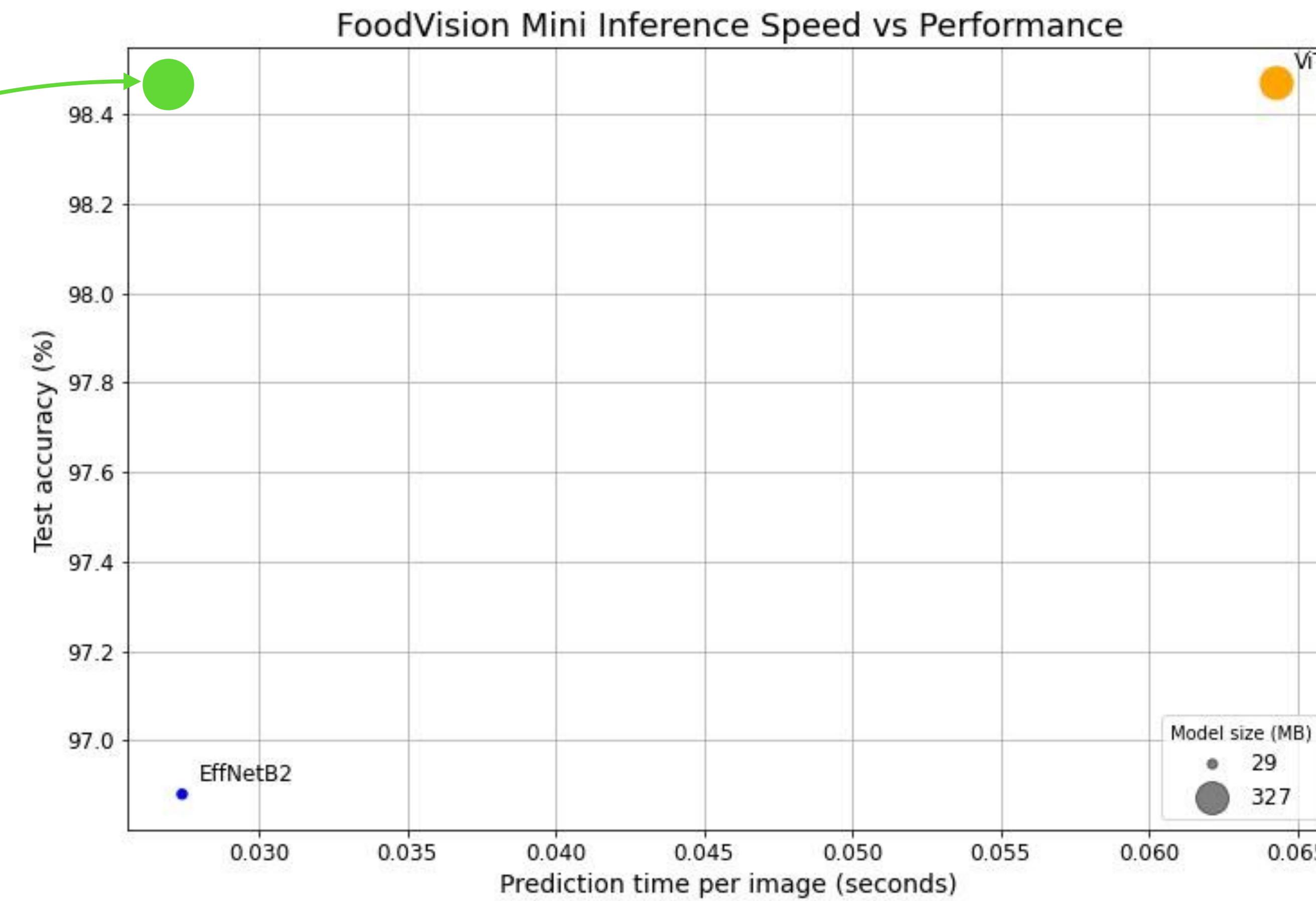
Model/function

Outputs

See more workflow ideas in the [Gradio documentation](#).

Performance vs. Speed trade-off

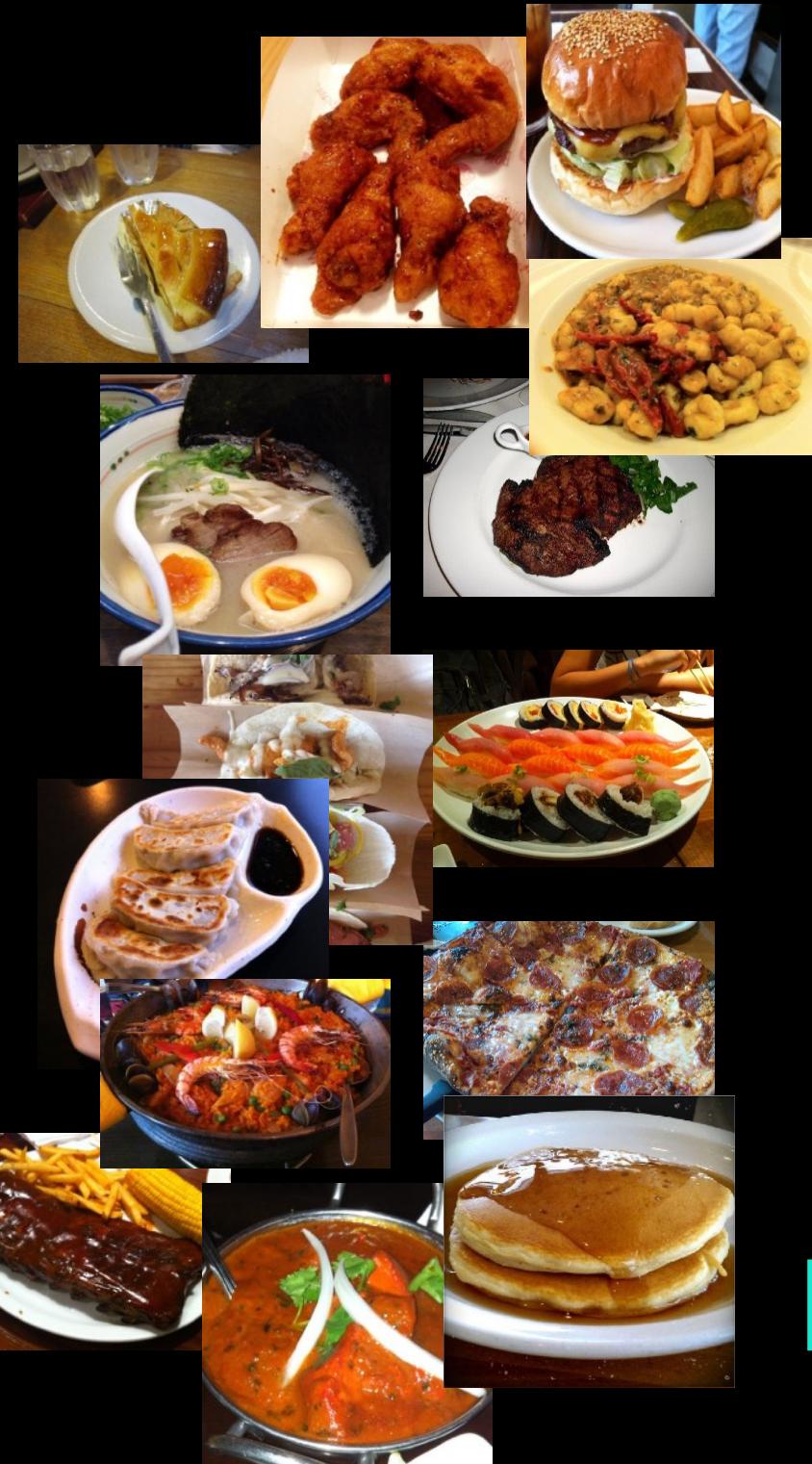
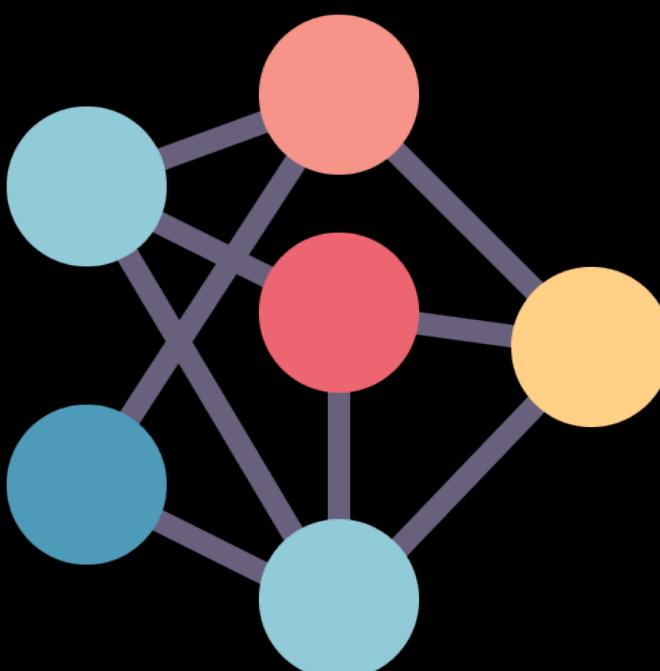
Most ideal position



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

FoodVision Mini → FoodVision Big

3 classes



101 classes

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.