MLOps for non-DevOps folks

I have a model, now what?!

Hi, I'm Hannes.

- Machine Learning Enthusiast
- Leading ML & Engineering at Caravel
- Excited about the ML Systems
- Co-author of "NLP in Action" (Manning) and "Building and Managing ML Workflows" (O'Reilly, Q4/2019)

These are *wonderful* times to be involved in Machine Learning projects.

Machine Learning allows almost unlimited access to information

Photo by Mark Rasmuson on Unsplash

Machine Learning helps identifying species

Photo by Massimiliano Latella on Unsplash

Machine Learning helps us to reduce congestions

Photo by NASA on Unsplash

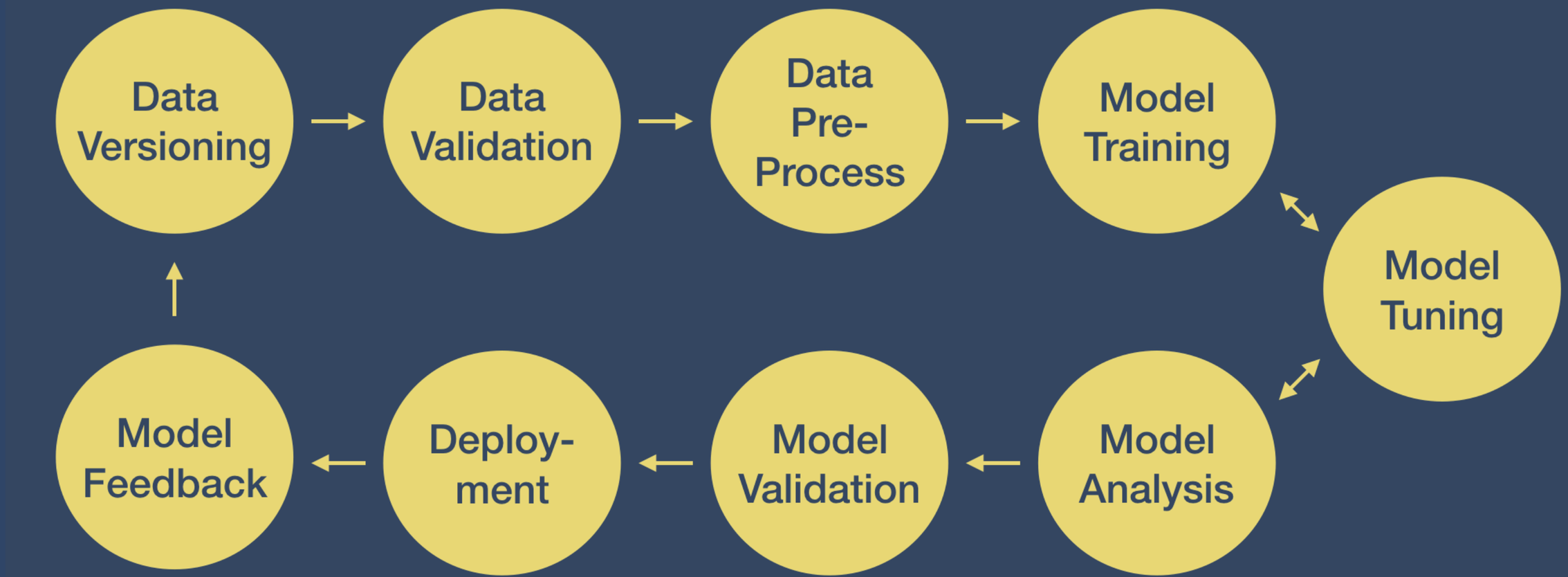
Photo by Jason Blackeye on Unsplash

What have all these examples in common?

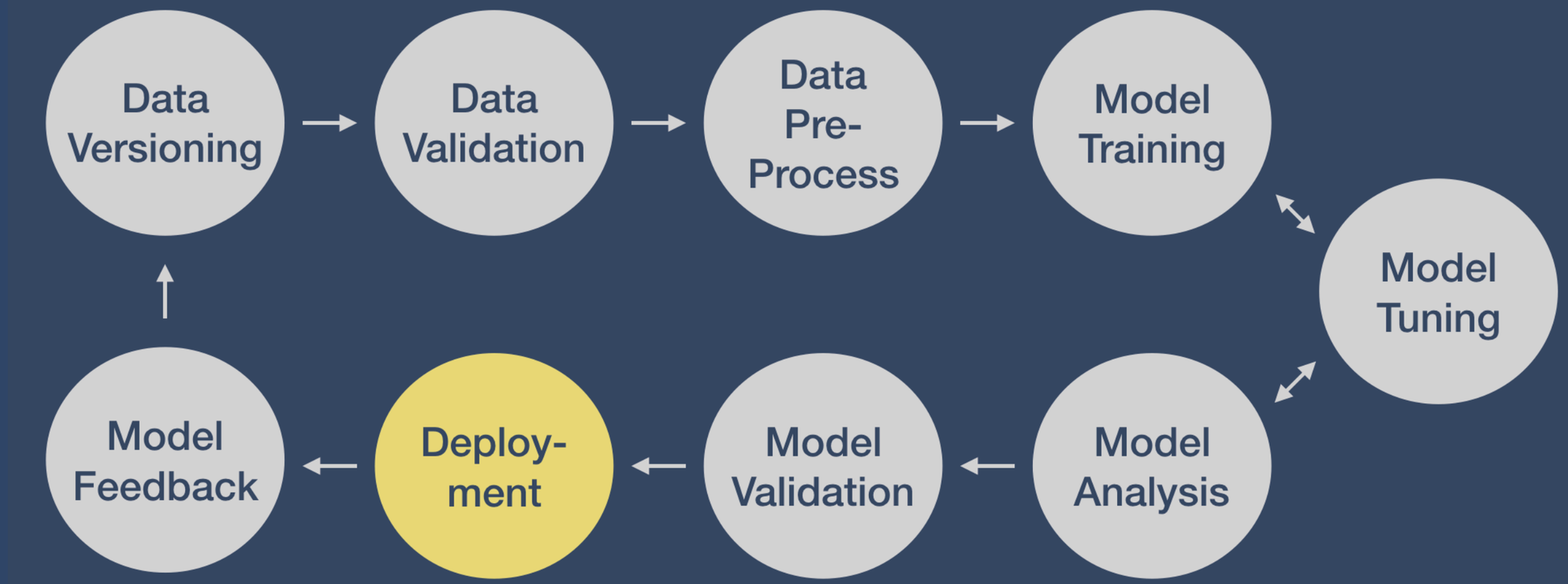
A Model Life Cycle

What do I mean with that?

Model Life Cycle



Let's talk about Serving Infrastructure



Why worry about deploying models?

Photo by Goh Rhy Yan on Unsplash

Your Model <- ? -> Users

The Serving Infrastructure is what allows people to access your models.

Photo by Ryoji Iwata on Unsplash

How can you deploy models?

There are so many options.

Think about your Application

- Central model deployment?
- Do you need a GPU?
- Privacy concerns?
- Distribute the predictions to the clients?

Think about your Application

- Central model deployment? **-> Model Server**
- Do you need a GPU? **-> Model Server** or *Browser Deployment*
- Privacy concerns? **->** *Edge* or *Browser Deployment*
- Distribute the predictions to the clients? **->** *Edge* or *Browser Deployment*

Model Servers

When to use a Model Server?

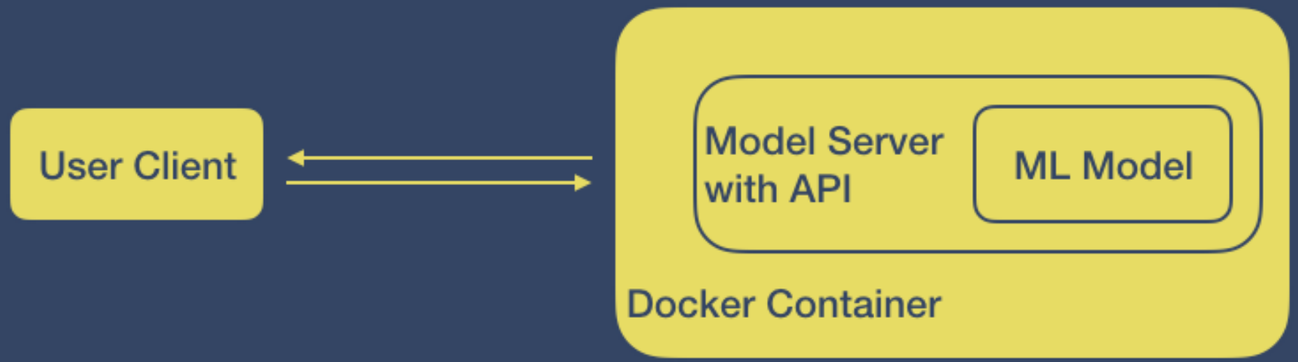
- Batch or online inferences
- Model A/B Testing
- Inferences on specific hardware

Simplest Architecture

- Model is loaded in a server
- Server allows API requests and infers the model predicts
- Sufficient architecture for a simple deployment

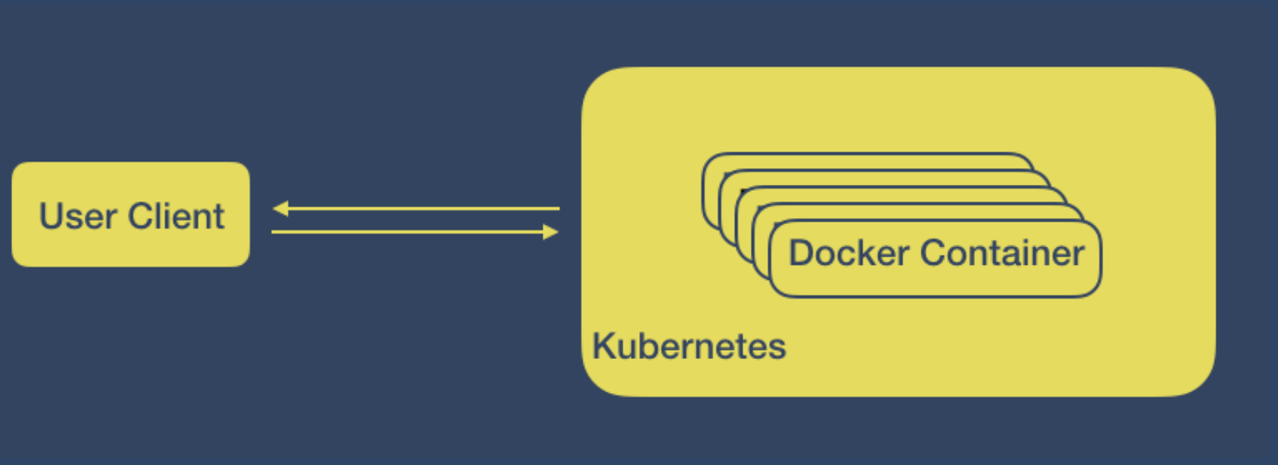
OS independent Architecture

- Docker: Dependency encapsulations
- Easy deployment across various operating systems
- Architecture for replicable deployments



Scalable Architecture

- Kubernetes: Container orchestration system
- Allows you to scale your APIs
- Useful for scalable deployments or multiple model versions



Flask

Model Servers Option 1: Flask

Pros

- Using the popular Python API framework Flask
- Works with all Python ML Frameworks
- Quick and dirty deployment

Model Servers Option 1: Flask

Cons

- API code and models aren't separated
- APIs can be inconsistent
- Inefficient use of compute resources

Model Servers Option 1: Flask

```
import json
from flask import Flask
from keras.models import load_model
from utils import preprocess
```

Model Servers Option 1: Flask

```
import json
from flask import Flask
from keras.models import load_model
from utils import preprocess

model = load_model('model.h5')
app = Flask(__name__)
```

_

Model Servers Option 1: Flask

```
import json
from flask import Flask
from keras.models import load_model
from utils import preprocess

model = load_model('model.h5')
app = Flask(__name__)

@app.route('/classify', methods=['POST'])
def classify():
    review = request.form["review"]


_
```

Model Servers Option 1: Flask

```
import json
from flask import Flask
from keras.models import load_model
from utils import preprocess

model = load_model('model.h5')
app = Flask(__name__)

@app.route('/classify', methods=['POST'])
def classify():
    review = request.form["review"]
    preprocessed_review = preprocess(review)
    prediction = model.predict_classes([preprocessed_review])[0]
    return json.dumps({"score": int(prediction)})
```

Photo by Goh Rhy Yan on Unsplash

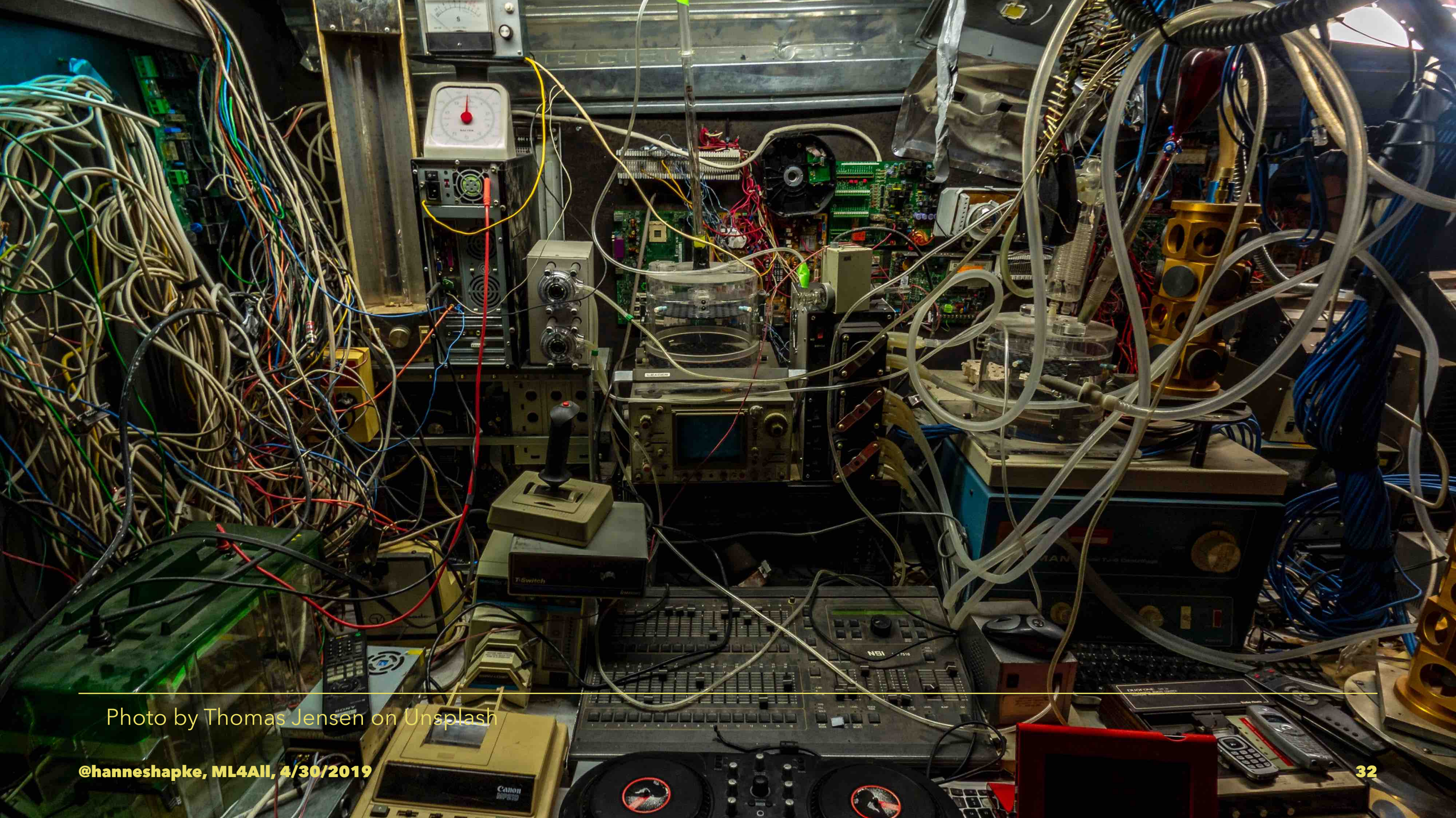


Photo by Thomas Jensen on Unsplash

TensorFlow Serving

Model Servers Option 2: TF Serving

Pros

- Provides separation between API code and models
- Easy model deployment
- Batching!
- Consistent APIs (gRPC and REST)
- Supports multiple model versions

Model Servers Option 2: TF Serving

Cons

- Only works with Keras and TensorFlow
- Requires Docker installation or Ubuntu Linux
- Cryptic error messages

Model Servers Option 2: TF Serving

```
docker run \
    -p 8501:8501 \
    -e MODEL_NAME=my_model \
    -t tensorflow/serving \
    --mount type=bind,source=/path/to/my_model/,target=/models/my_model
```

Model Servers Option 2: TF Serving

```
2019-04-26 03:51:20.304826: I tensorflow_serving/model_servers/server.cc:82]
  Building single TensorFlow model file config:
  model_name: my_model model_base_path: /models/my_model
2019-04-26 03:51:20.307396: I tensorflow_serving/model_servers/server_core.cc:461]
  Adding/updating models.
2019-04-26 03:51:20.307473: I tensorflow_serving/model_servers/server_core.cc:558]
  (Re-)adding model: my_model
...
2019-04-26 03:51:34.507436: I tensorflow_serving/core/loader_harness.cc:86]
  Successfully loaded servable version {name: my_model version: 1556250435}

[evhttp_server.cc : 237] RAW: Entering the event loop ...
2019-04-26 03:51:34.520287: I tensorflow_serving/model_servers/server.cc:333]
  Exporting HTTP/REST API at:localhost:8501 ...
```

Model Servers Option 2: TF Serving

```
def rest_request():
    url = 'http://localhost:8501/v1/models/my_model:predict'
    payload = json.dumps({"instances": [TEXTS[0]]})
    r = requests.post(url, payload)
    return r

rs_rest = rest_request()
rs_rest.json()
```

Model Servers Option 2: TF Serving

```
{'predictions': [{'scores': [
    0.293399,
    0.101302,
    0.162343,
    0.179935,
    0.0551261,
    0.174151,
    0.0378635,
    0.102538,
    0.358822],
   'classes': ['0', '1', '2', '3', '4', '5', '6', '7', '8']}]}
```

Photo by Thomas Jensen on Unsplash

Waht to learn more about TF Serving?

- Managing and Building Machine Learning Workflows (Q4/2019)
- http://bit.ly/DeployDLModels
- TensorFlow Documentation

Other Options

Other Options

- **Seldon**: Scalable and framework agnostic
- **GraphPipe**: Deployment for mxnet, Caffe2 and PyTorch models (via ONNX)
- **MLflow**
- **Simple TensorFlow Serving**

Cloud Options

Cloud Options

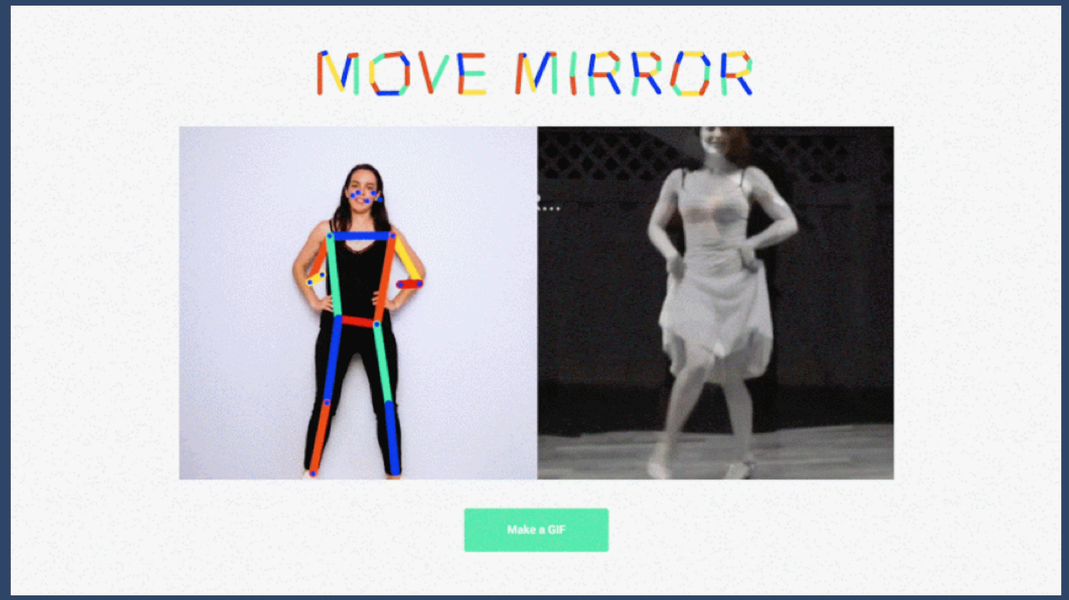
- Google AI Platform
- Azure ML
- AWS SageMaker

Think about your Application, again

- Central model deployment? **->** *Model Server*
- Do you need a GPU? **->** *Model Server* or **Browser Deployment**
- Privacy concerns? **->** *Edge* or **Browser Deployment**
- Distribute the predictions to the clients? **->** *Edge* or **Browser Deployment**

Deployments to Client's Browsers

Deploying models with TensorFlow.js



https://experiments.withgoogle.com/move-mirror

Deploying models with TensorFlow.js

Pros

- Inference is happening in the brower
- Great for models using privacy related data
- No model server required

Deploying models with TensorFlow.js

Cons

- Limited to TensorFlow and Keras
- Model needs to be converted
- Limited by the model size
- Model is been sent to the user (IP)

Deploying models with TensorFlow.js

Steps

- Train your model (local/Cloud)
- Export the model as a `SavedModel`
- Install model converter with `pip install tensorflowjs`
- Convert the model with `tensorflowjs_converter`
- Integrate it into your js stack

Think about your Application, again

- Central model deployment? **->** *Model Server*
- Do you need a GPU? **->** *Model Server* or *Browser Deployment*
- Privacy concerns? **-> Edge** or *Browser* **Deployment**
- Distribute the predictions to the clients? **-> Edge** or *Browser* **Deployment**

TensorFlow Lite

Deploying models with TensorFlow Lite

https://proandroiddev.com/mobile-intelligence-tensorflow-lite-classification-on-android-c081278d9961

Deploying models with TensorFlow Lite

Pros

- Deployment to Edge Devices (e.g., watches, mobile phones)
- Great for models using privacy related data
- Allows inference under CPU, memory and battery constraints

Deploying models with TensorFlow Lite

Cons

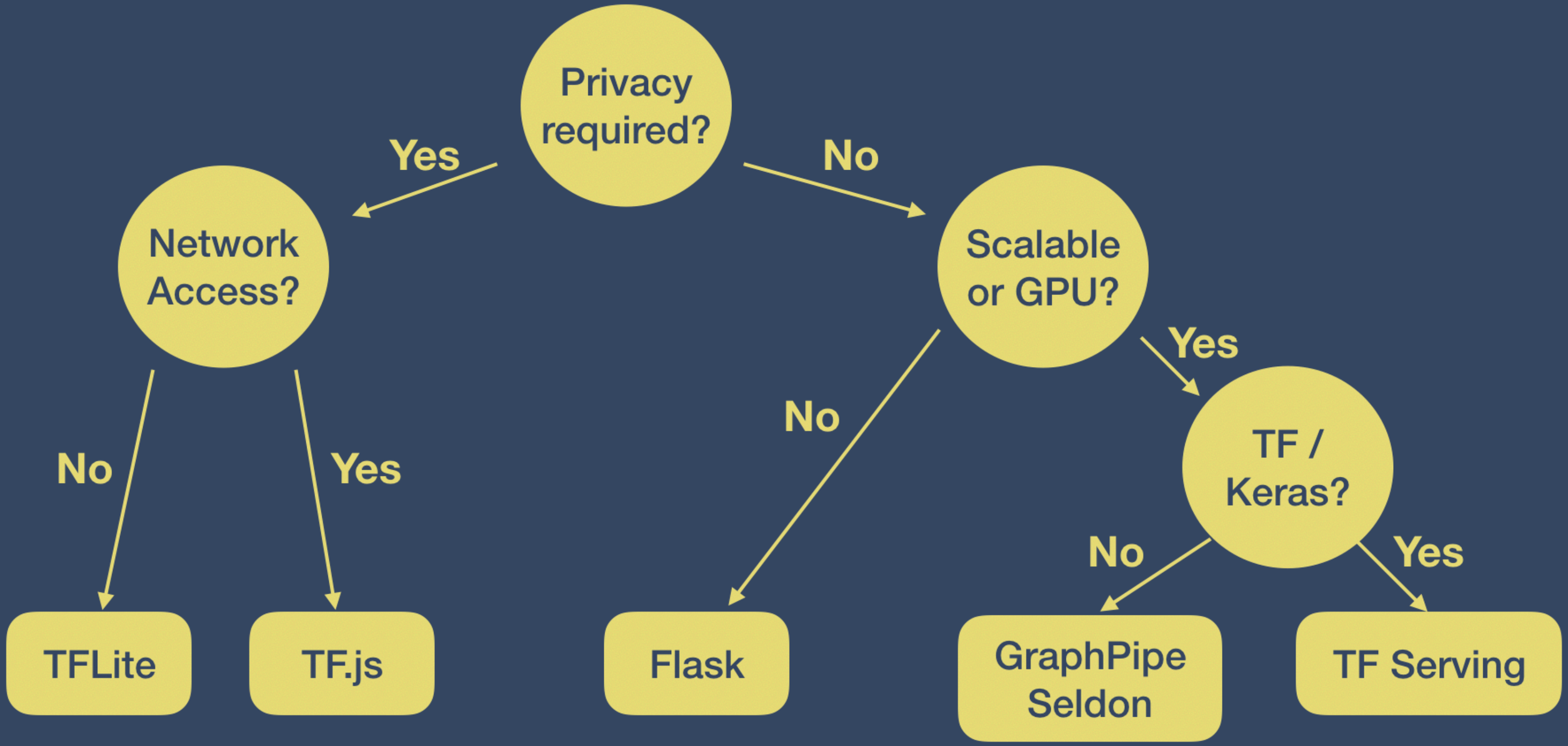
- Limited to TensorFlow and Keras
- Model needs to be converted
- Limited number of TensorFlow Ops

Deploying models with TensorFlow Lite

Steps

- Train your model (local/Cloud)
- Export the model as a `SavedModel` or `hdf` file
- Convert the model with `tf.lite.TFLiteConverter`
- Quantizing your model post training
- Integrate in your Android or iOS app

Quick Recap


Privacy required?
Yes
No
Network Access?
Scalable or GPU?
No
Yes
No
Yes
TF / Keras?
No
Yes
TFLite
TF.js
Flask
GraphPipe Seldon
TF Serving

	TF / Keras	Scikit	PyTorch	XGBoost
Flask	x	x	x	x
TensorFlow Serving	x	x		(x)
GraphPipe	x		x	
Cloud Model Instance	x	x		x
In the Browser	x			
Mobile Devices	x		x	
Other Edge Devices	x		x	

Demo!

Happy Deployments

Thank you

@hanneshapke
@_caravel

https://github.com/hanneshapke/ml4all-model-deployments

References

- GraphPipe
- Seldon
- Deploying R Models
- Simple TF Serving