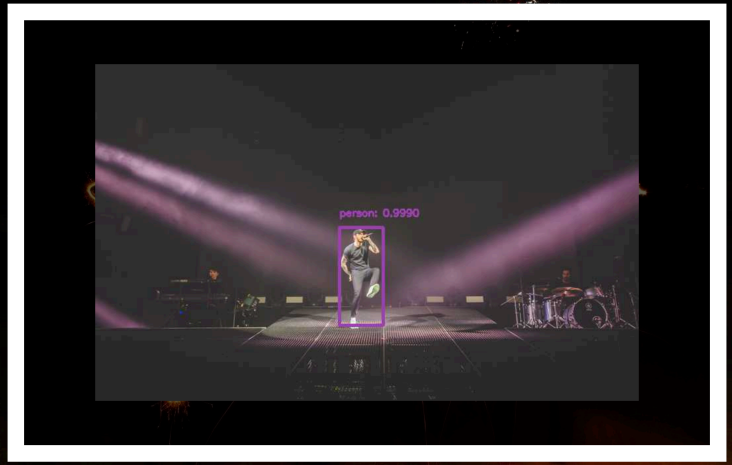


Classification in progress...

Loading the model may take some time. Please be patient.

Welcome to the AntBee Transfer Learning Demo

Please upload the file you want to be classified.

Choose File 144088310_a4176fd54d.jpg

Start!

Bee

Welcome to the YOLO image analyzer

Please upload the file you want to be processed.

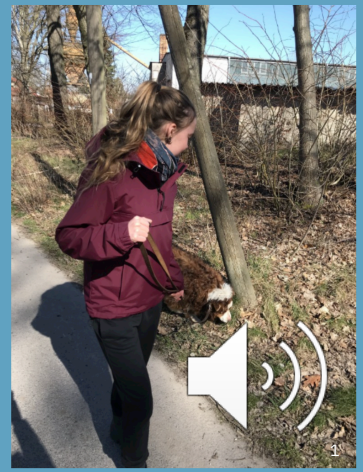
Choose File Image from iOS.jpg

Confidence score for the objects: 0.5

NMS threshold for overlapping objects: 0.3

Find objects!

This is most likely a dog



Deployment Options for Computer Vision Demos built in Python

—

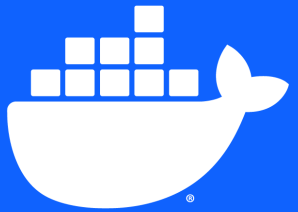
Natalie Jann
Corporate Student
Computer Vision Deep Dive

Definition

Packaging

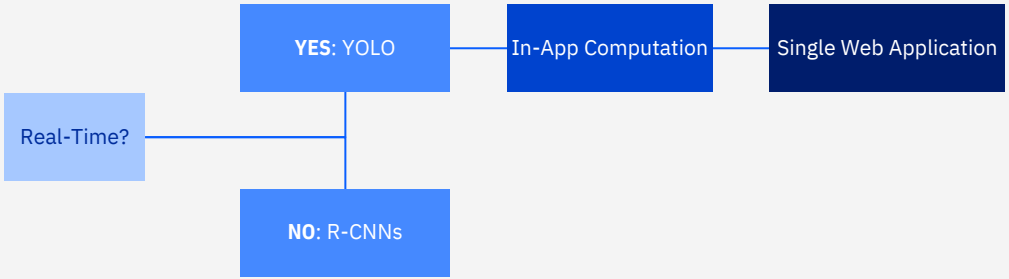
Containerization

Release



Step One:
Definition

Deployment Patterns



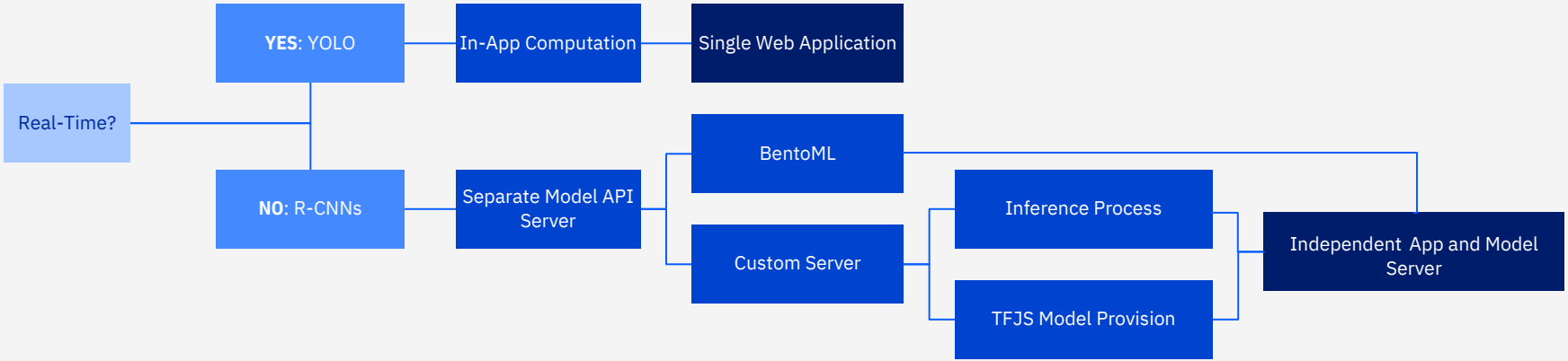
In-App Computation

The decisive advantage of real-time object detection systems is their speed.

Therefore, it is justifiable to integrate them directly into a web application.

The prediction should still have its own route to process and analyze the image.

Deployment Patterns



Step Two:
Packaging

Separate API

BentoML

Manage machine learning model serving agilely.

Online and Offline.

BENTOML

Define a service for your model, its endpoints, input types and dependencies.

```python
import bentoml
from bentoml.adapters import DataframeInput
from bentoml.artifact import SklearnModelArtifact

@bentoml.env(auto_pip_dependencies=True)
@bentoml.artifacts([SklearnModelArtifact('model')])
class IrisClassifier(bentoml.BentoService):

    @bentoml.api(input=DataframeInput())
    def predict(self, df):
        return self.artifacts.model.predict(df)
```

Pack your trained model and start serving it.

```
bentoml serve IrisClassifier:latest
```

```
from sklearn import svm
from sklearn import datasets

# 1. Model training
clf = svm.SVC(gamma='scale')
iris = datasets.load_iris()
X, y = iris.data, iris.target
clf.fit(X, y)

# 2. Create BentoService instance
iris_classifier_service = IrisClassifier()

# 3. Pack trained model artifacts
iris_classifier_service.pack("model", clf)

# 4. Save
saved_path = iris_classifier_service.save()
```

Custom Server

Modularize Inference

Create functions for the
- image preprocessing
- net initialization (if needed)
- detection or classification
- result depiction

[alt]: Export your TFJS model

Choose A Framework

In Python, one of the best choices for API definition and provision is FastAPI.

Create An Endpoint

One route per model running on the same server is usually sufficient.

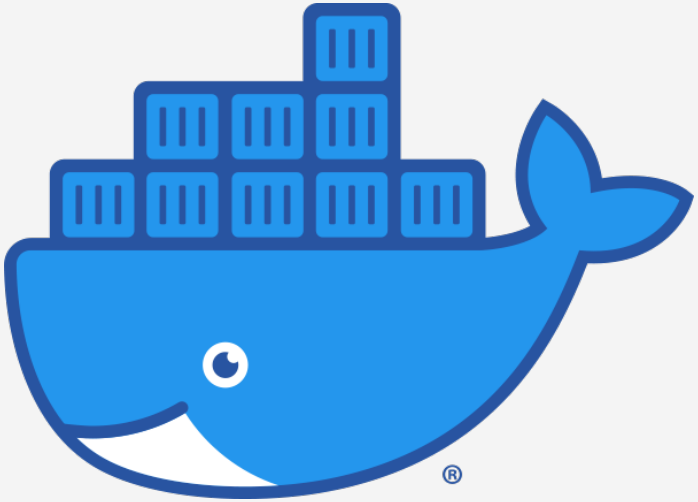
[alt]: a TFJS model requires one route for `model.json` and one for the binary files

Run Your Server

Choose a stable server backend, e.g. gunicorn or tornado.

Step Three:
Containerization

Docker



Manage dependencies, data and trained models in isolated environments.

Bit by bit.

~/bentoml/{service_name}/{version}/Dockerfile

high-performance server
adaptive micro-batching support

tiangolo/uvicorn-gunicorn-fastapi

"auto-tuning" mechanism included
ready for production

Step Four:
Release

Cloud Platforms

Google Cloud Platform

Push your images to a registry and run them inside a k8s cluster.

1 free zonal cluster

Heroku

User Heroku Git, GitHub or Heroku Container Registry to deploy your app.

5 apps free of charge

IBM Cloud Kubernetes

Push your images to a registry and run them inside a k8s cluster.

1 free cluster with 1 worker node

Thank you

Natalie Jann
Corporate Student
—
natalie.jann@ibm.com
+49 1722775475
ibm.com