The rod of Asclepios

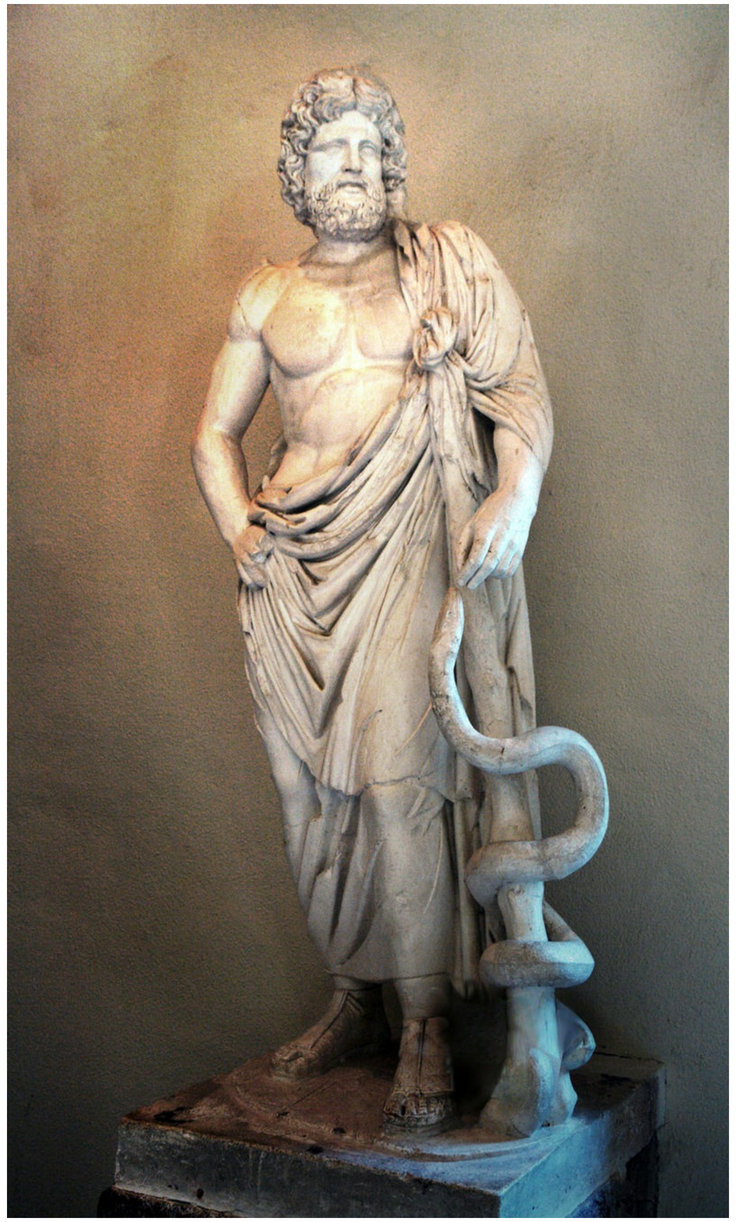
Jan Margeta | jan@kardio.me | @jmargeta

KardioMe

World full of superpowers

Python as our superpower?

Meet Asclepios



Hi, I am Jan

- Pythonista for 8 years
- Founder of KardioMe
- Maker of tools to better understand our hearts
- Python 3.5+, numpy, MxNet, Keras, Tensorflow, scikit-learn, SimpleITK, pydicom, Flask, Sanic, Django, PostgreSQL, ReactJS, Docker

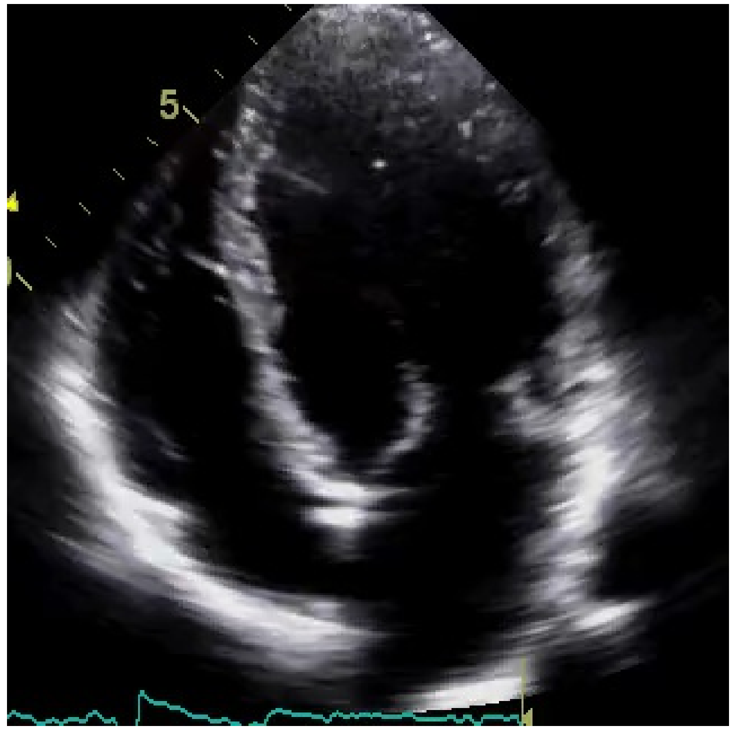
This talk

- Peek into our hearts with medical images
- What used to be hard is now magically simple
- Tricks and tools for machine learning in Python we've learned along the way

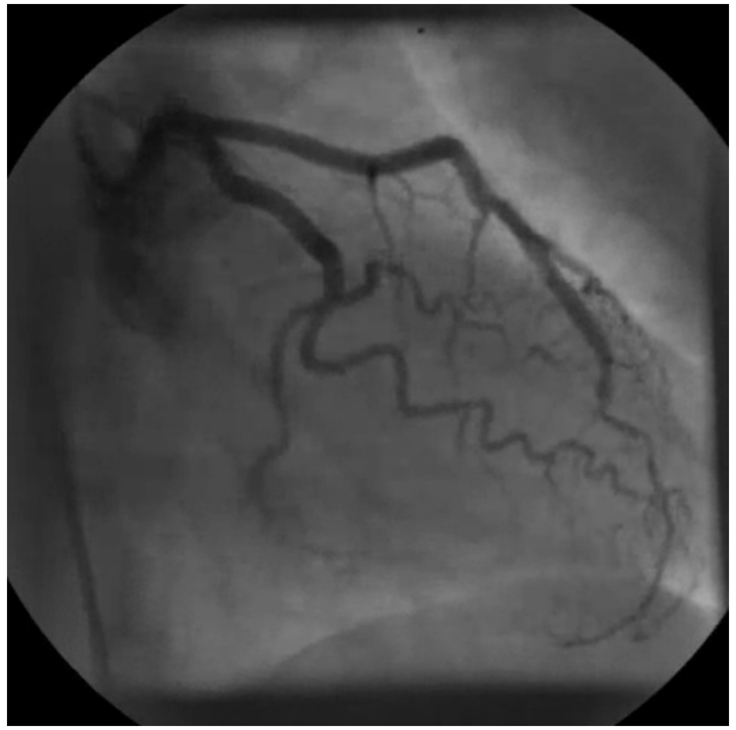
Imaging of our hearts

X-Ray

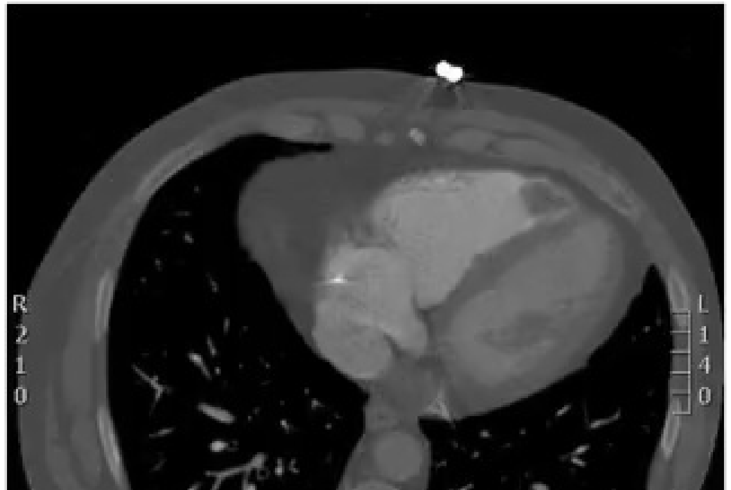
ultrasound



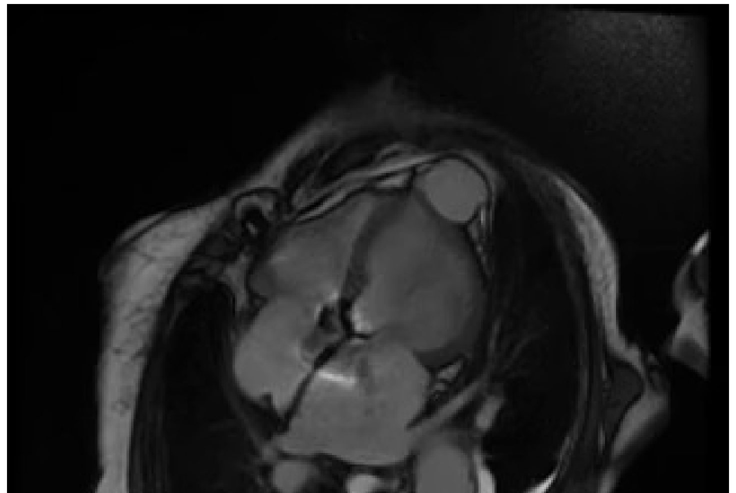
fluoroscopy



computed tomography



magnetic resonance



Kelly 2007
Carmo et al. 2010
Arnold et al. 2008
Foley et al. 2010
Vanezis et al. 2011


What a human
sees

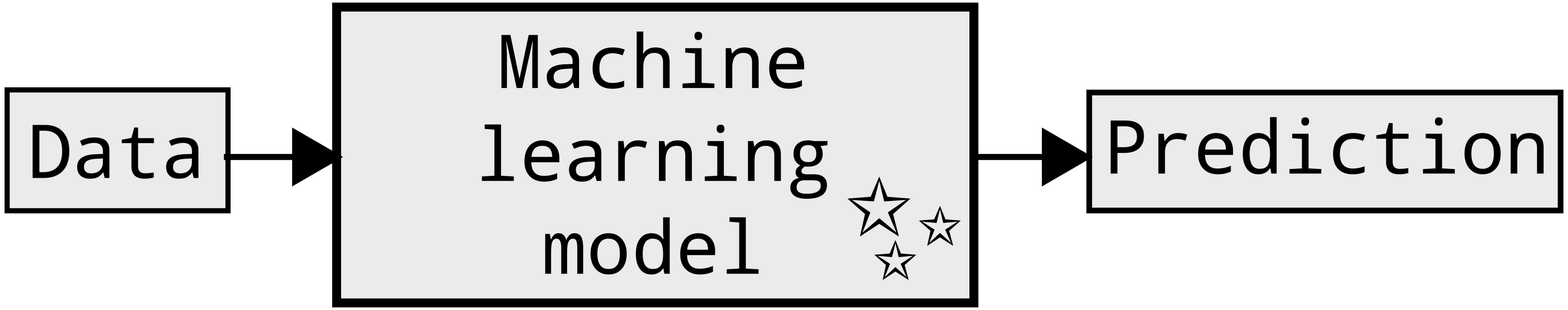

What a human sees

What a computer sees

133	115	57	65	52	69	129	145	134	121	106	81	87	59	42	44	18	24
131	87	53	57	140	143	133	130	119	117	112	115	81	94	48	31	31	25
139	59	54	120	138	135	137	139	120	110	107	109	122	68	105	33	36	16
119	60	46	144	138	138	131	133	120	111	67	45	46	98	91	44	37	18
106	47	71	[illegible]	[illegible]	[illegible]	127	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	43	32	17
103	43	80	125	127	124	120	109	120	109	106	110	56	83	92	35	34	24
90	51	59	118	123	113	107	[illegible]	[illegible]	[illegible]	99	91	108	103	69	43	31	32
101	60	59	105	105	105	98	99	105	109	95	114	101	110	54	39	36	22
90	45	49	94	101	96	94	109	104	100	107	93	98	39	49	34	31	18
67	46	45	58	95	102	77	85	101	79	91	97	71	58	43	38	16	19
19	39	50	48	54	72	98	91	68	88	97	79	61	36	33	19	17	20

Machine learning

Solving problems with data

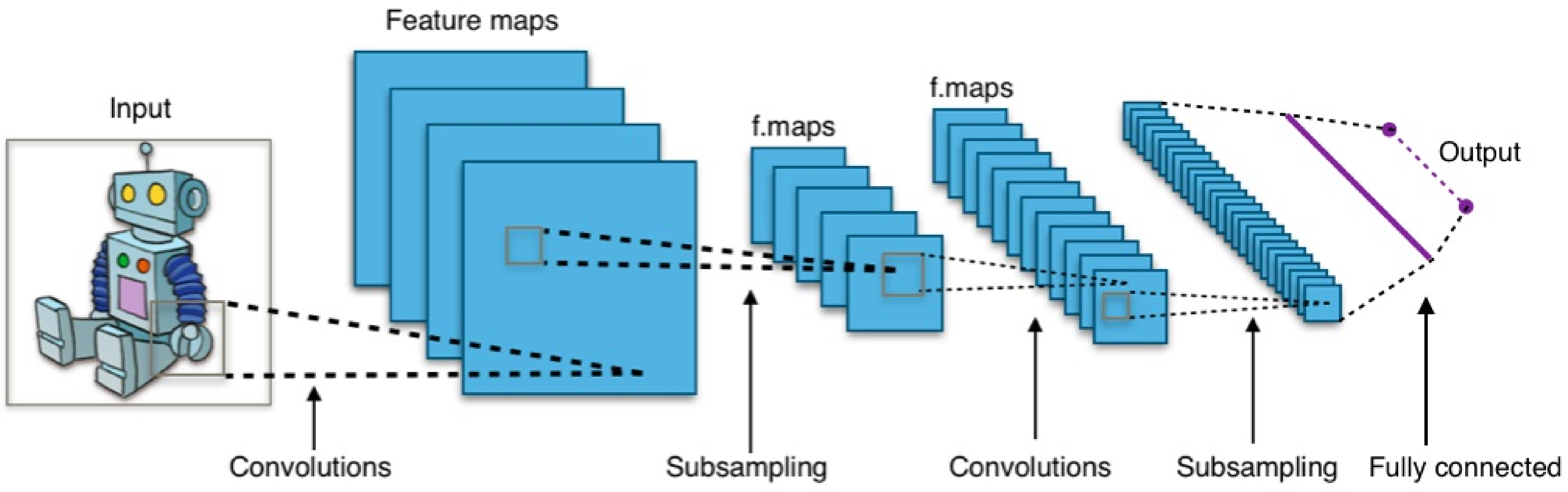


prediction = **model**.*predict* (**data**)

Image recognition in 6 lines of code

```
from keras.applications import imagenet_utils
from keras.applications.vgg16 import VGG16

# Load and prepare input images
images_raw = load_images()
images = imagenet_utils.preprocess_input(images_raw)

# Load a pretrained image classification model
model = VGG16(include_top=True, weights='imagenet')

# Do the prediction
predictions = model.predict(images)
```

Convolutional neural networks



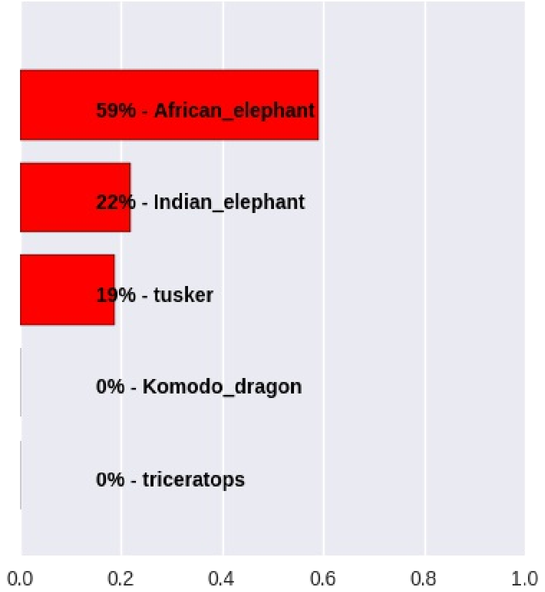
By Aphex34

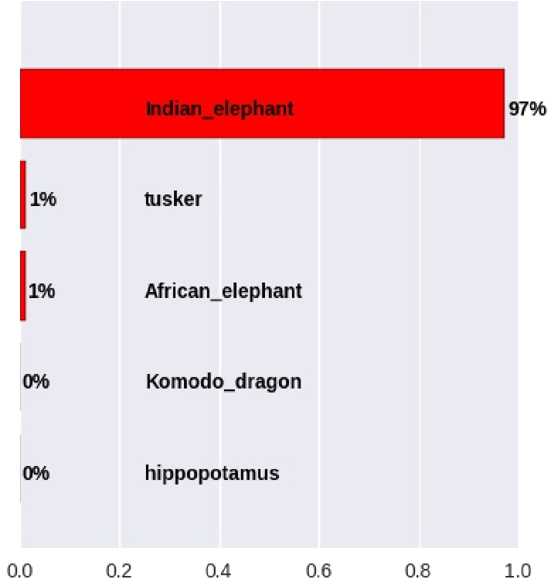
Finding the right image representation

Excellent for natural images

Trained on Imagenet large scale visual recognition challenge dataset

10 million images, 1000 categories





Extracting visual features

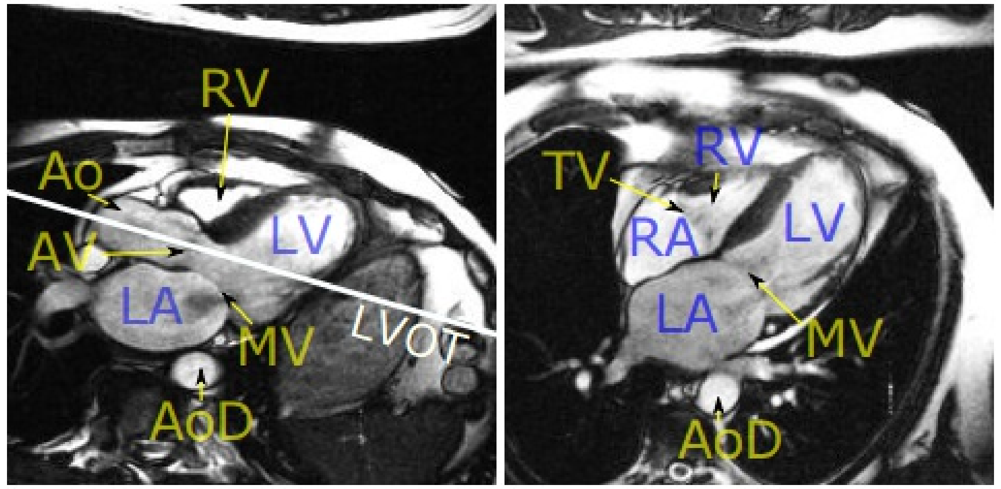
```
# Load a pretrained classification model
source_model = VGG16(weights='imagenet')

# Define feature extractor from one layer of the network
feature_layer = source_model.get_layer('conv4')
feature_extractor = Model(
  input=fix_model.input,
  output=feature_layer.output)

# Extract features
features = feature_extractor.predict(images)
```

See also "Deep visualization toolbox" on youtube

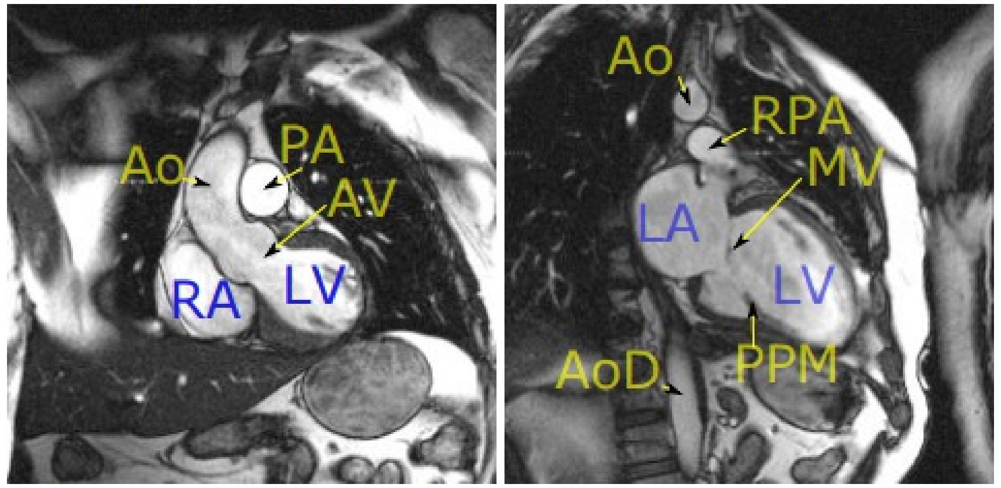
Using the extracted features with scikit-learn

```
from sklearn.svm import LinearSVC
def flatten_features(features):
    return features.reshape(len(features), -1)

features_train = feature_extractor.predict(images_train)
features_train = flatten_features(features_train)
classifier = LinearSVC()
classifier.fit(features_train, labels_train)

# predict on never seen images
features_test = feature_extractor.predict(images_test)
features_test = flatten_features(features_test)
prediction_test = classifier.predict(features_test)
```

Example: Cardiac view recognition



(a) Three chamber (3CH) (b) Four chamber (4CH)



(c) Left ventricular outflow tract (LVOT) (d) Two chamber (2CH)

Margeta et al. 2015, Joint work with Inria and Microsoft Research Cambridge

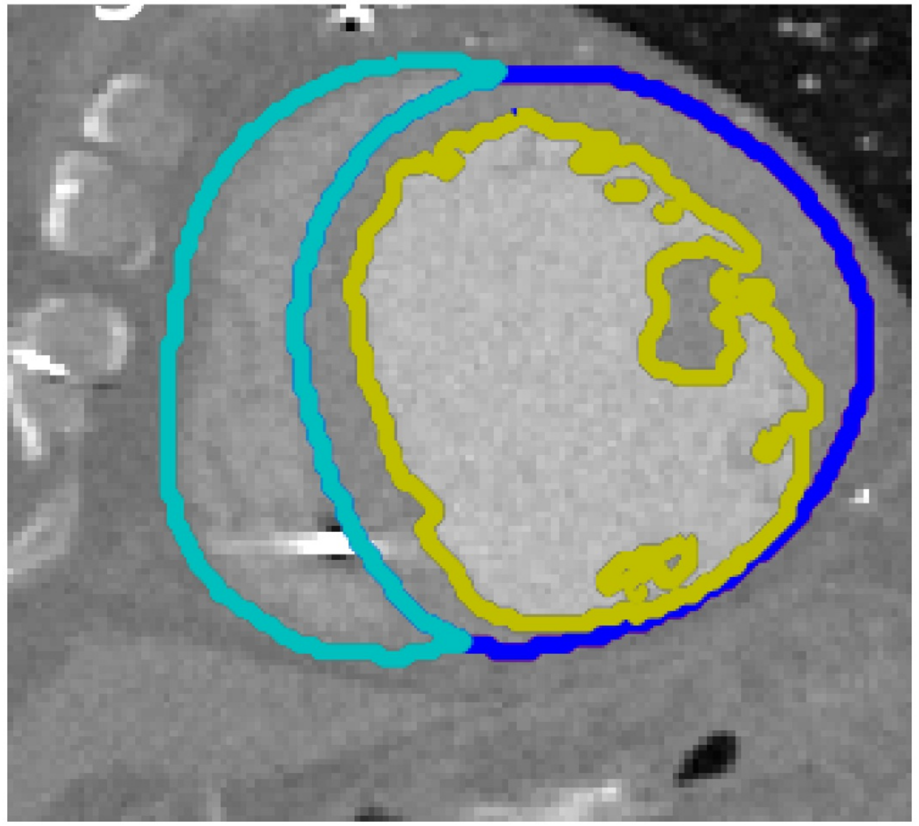
Train the model from scratch

```
from keras.models import Sequential
from keras.layers import Conv2D, Dense, Flatten

images_train, labels_train = load_data()
shape = (64, 64, 1)

model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=shape),
    MaxPooling2D(pool_size=(2, 2))
    Flatten(),
    Dense(4, activation='softmax'),
])
```

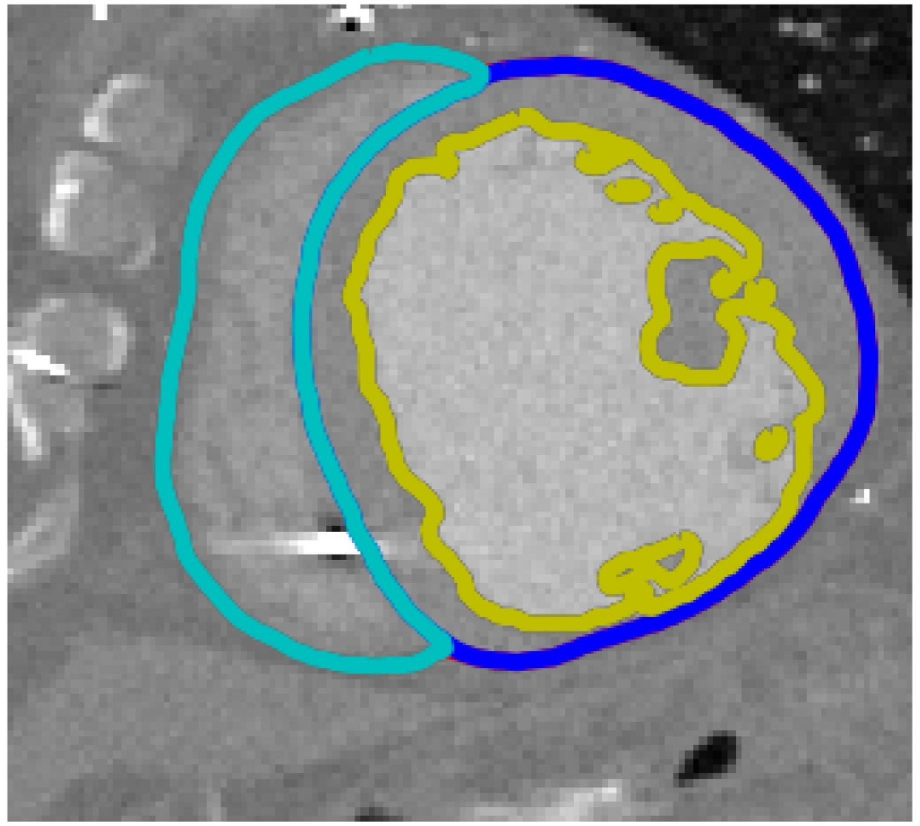
```
# Loss function - task dependent
# high for bad parameters, low for good ones
# e.g. for image recognition
loss_function = 'sparse_categorical_crossentropy'

# Compile the model and fit
model.compile(loss=loss_function, optimizer='adam')
model.fit(images_train, labels_train)

# Save the model for reuse
model.save('model.h5')
```

Let's save some time for our radiologists

30 minutes



12 seconds



KardioMe, Joint work with Inria and IHU Liryc

Deploy as a web service

Flask
web development,
one drop at a time

docker

see also Tensorflow-serving and Kubernetes

Expose the model with Flask*

```
import keras
from flask import Flask, jsonify, request
app = Flask(__name__)
model = keras.models.load_model('model.h5')

@app.route('/predict', methods=['POST'])
def predict():
    image_batch = request_to_numpy(request)
    y = model.predict(image_batch)
    prediction = convert_prediction(y)
    return jsonify(output=prediction)

app.run(port=5000, threaded=False)
```

*do not run in production, it requires a bit more love than this

Run with the same conditions as when it was built

Define the Dockerfile

```
FROM python:3.5

RUN mkdir -p /usr/src/app
COPY server.py /usr/src/app/
COPY model.h5 /usr/src/app/
COPY requirements.txt /usr/src/app/
WORKDIR /usr/src/app
RUN pip install -r requirements.txt

EXPOSE 5000
CMD python server.py
```

Build the Docker container

```
docker build -t kardiome/model-pyparis .
```

Run the service

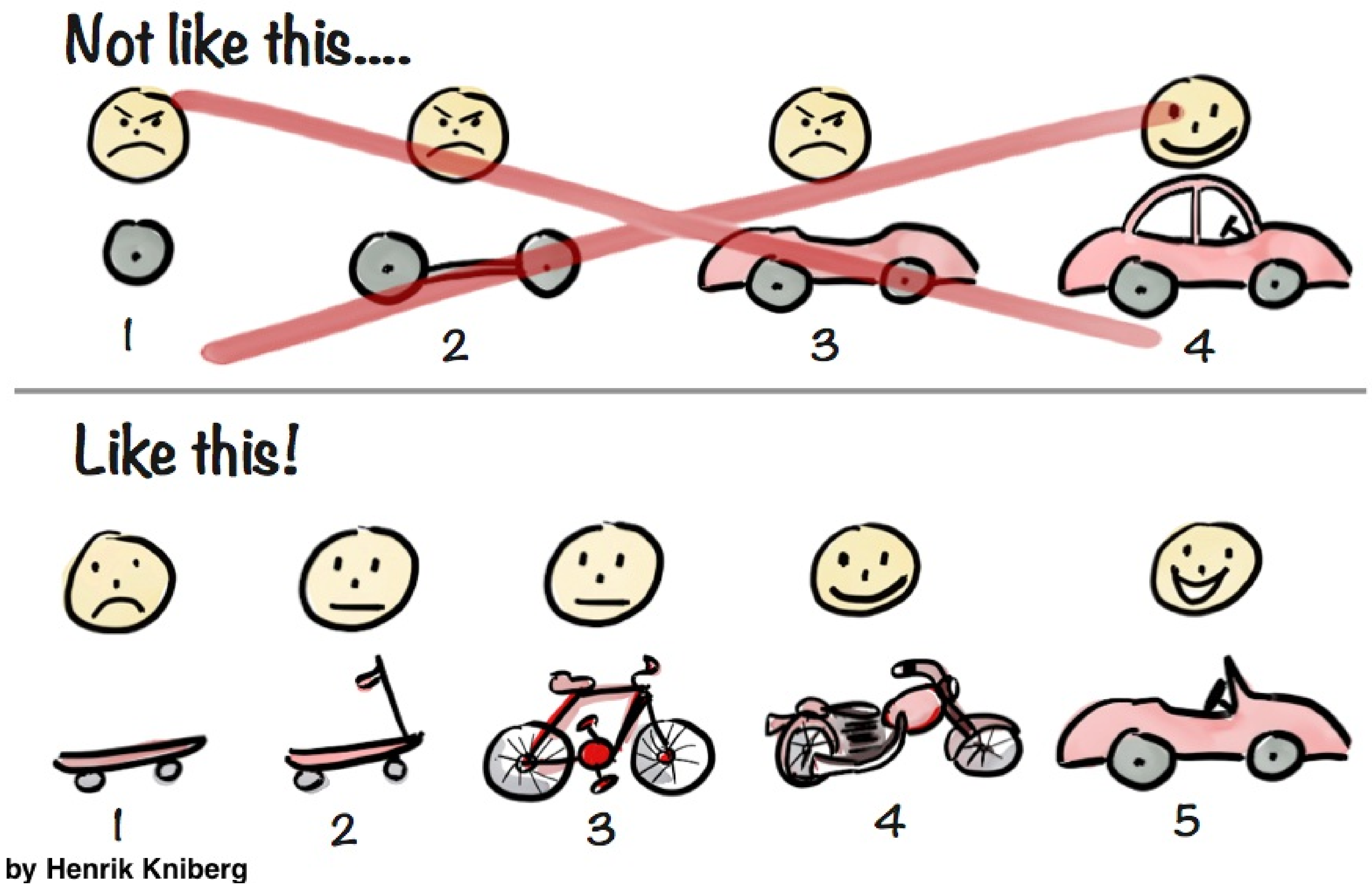
```
docker run -d -p 5000:5000 kardiome/model-pycon
```

Call the service

```
curl -X POST -F 'image=@/data/im.png' localhost:5000/predict
```

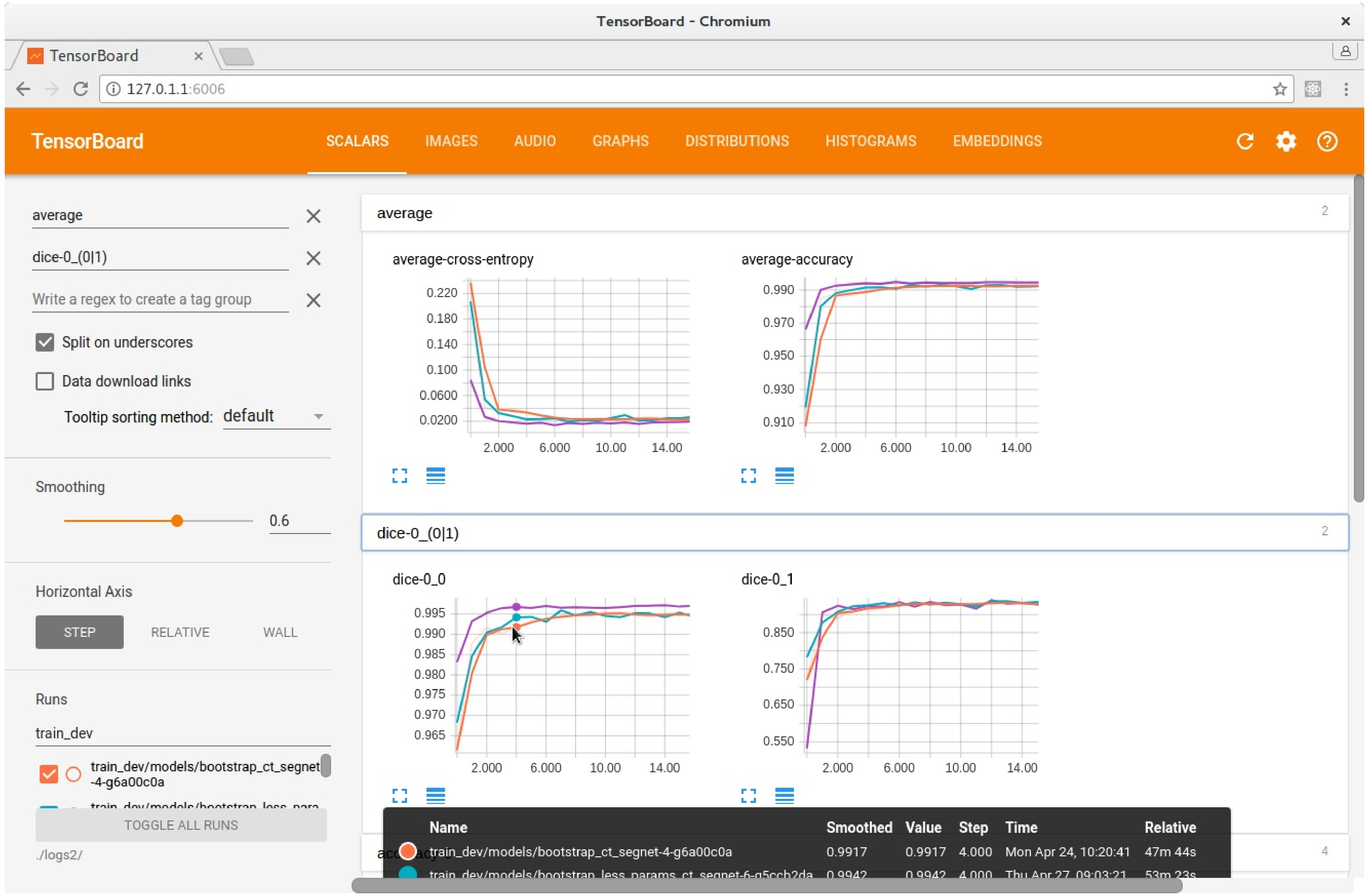
Tips to improve your machine learning today

Iterate fast



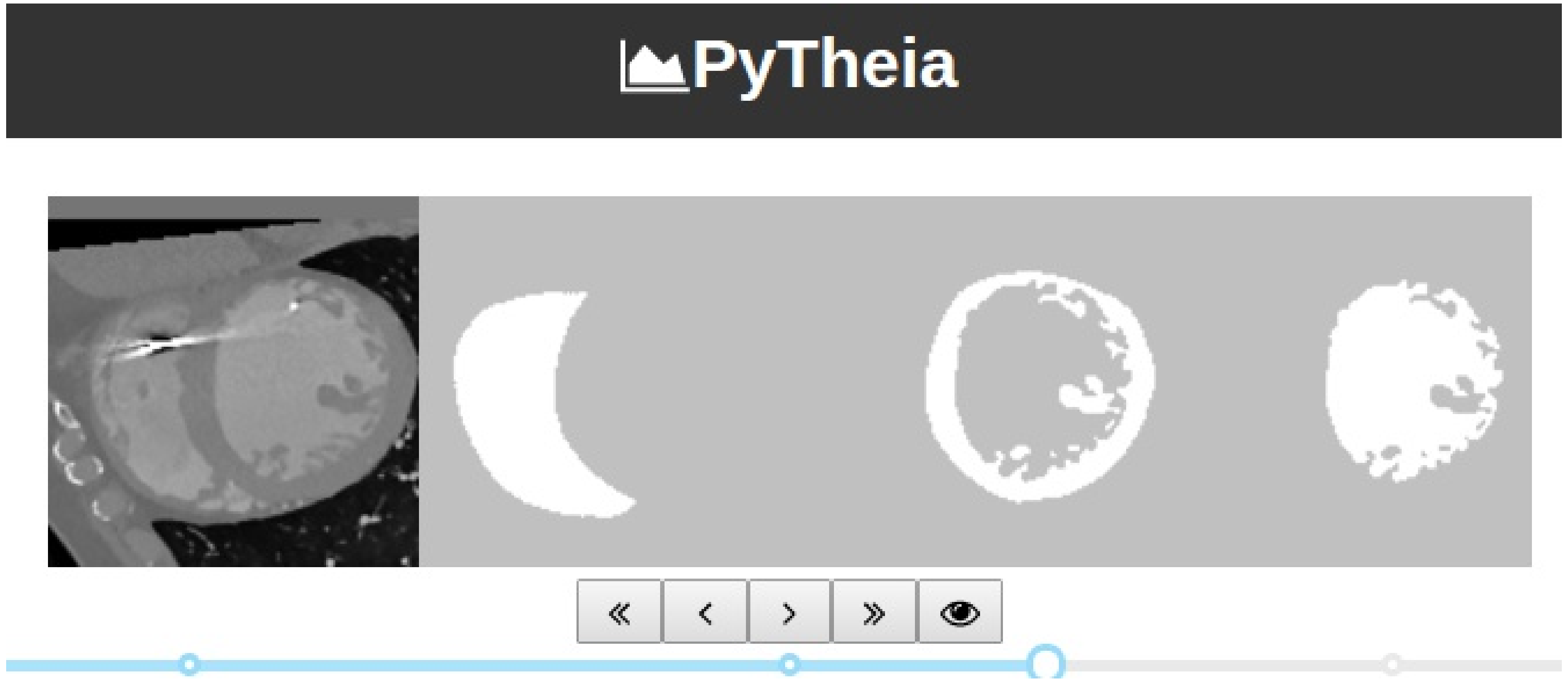
Minimally viable model

One metric to rule them all



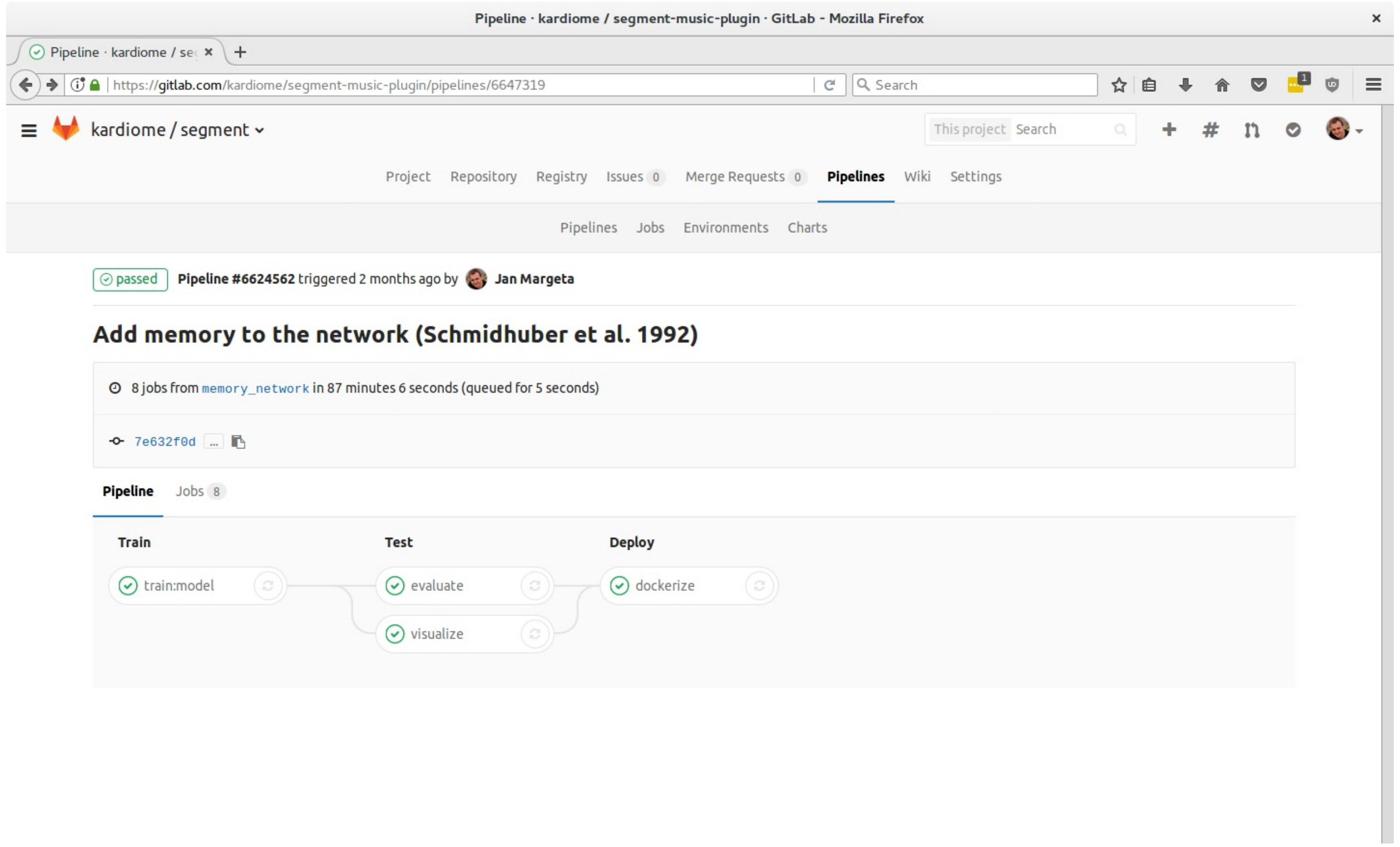
Training progress visualisation in Tensorboard

Visualize everything



Built with Sanic backend (like Flask) and React

Progress with confidence and repeatable pipelines



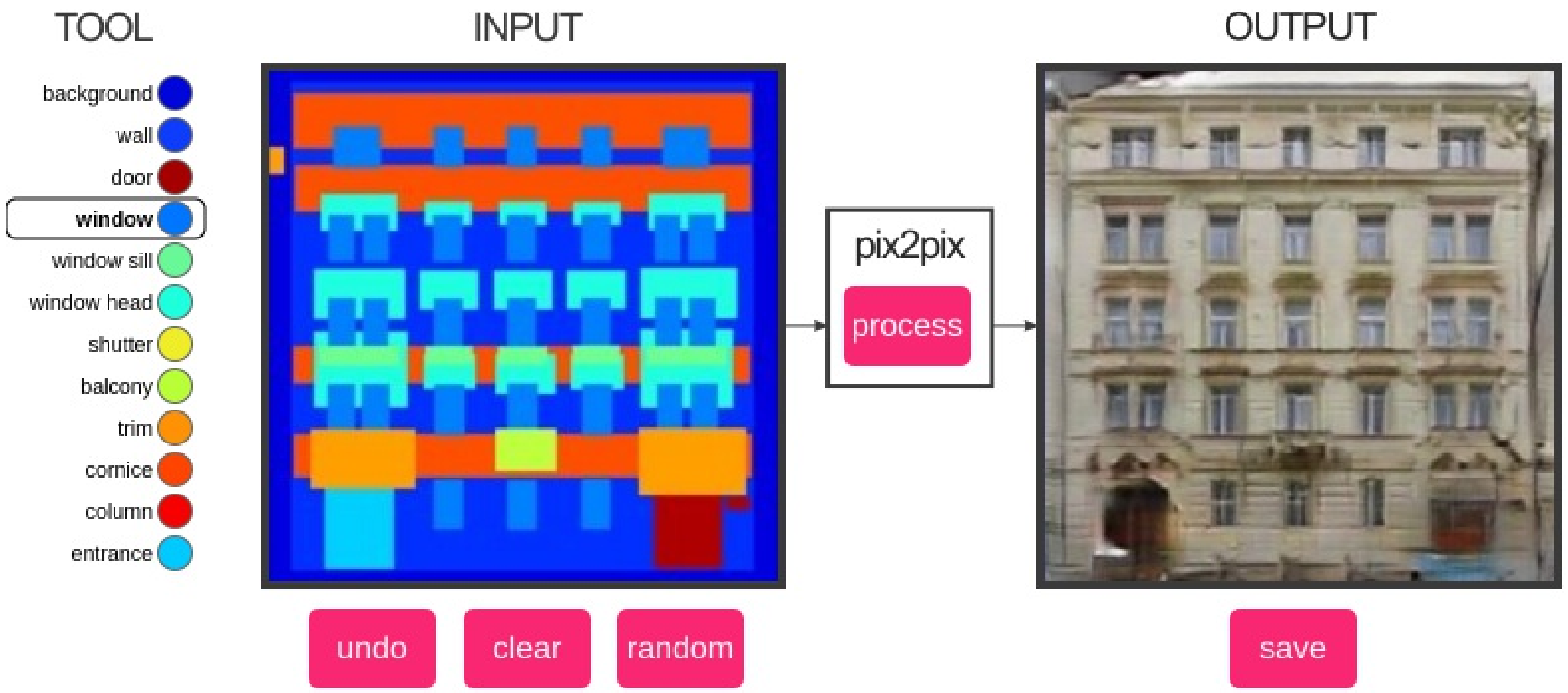
Gitlab's continuous integration is an excellent start for simple pipelines
(see Airflow, Luigi, Joblib, Flink)

No glory in data preparation

But it must be done

Having a small dataset?

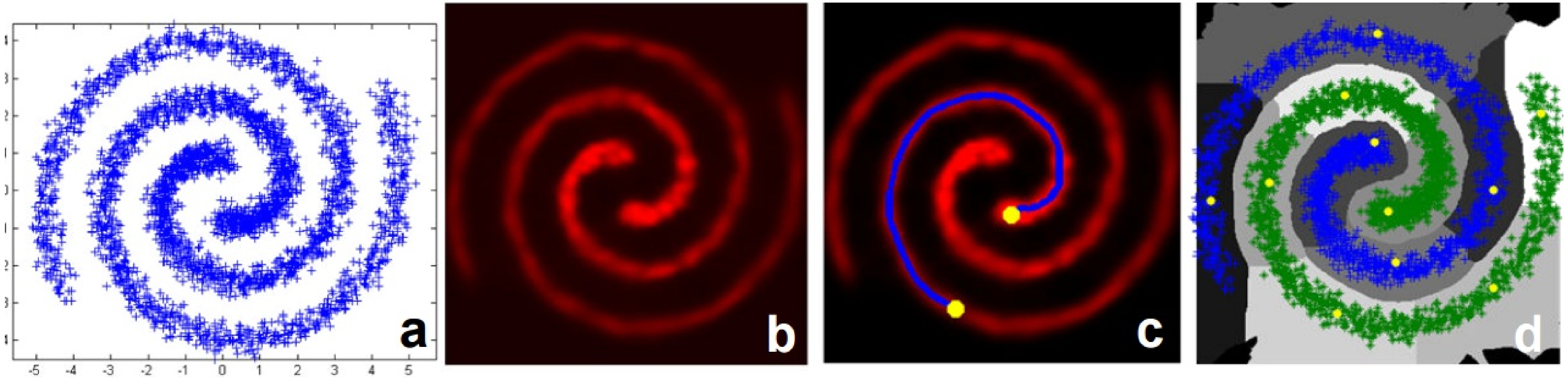
Do something about it



https://affinelayer.com/pixsrv/

Got unlabeled data?

Don't be lazy, just annotate it if you can, there are tools to help you

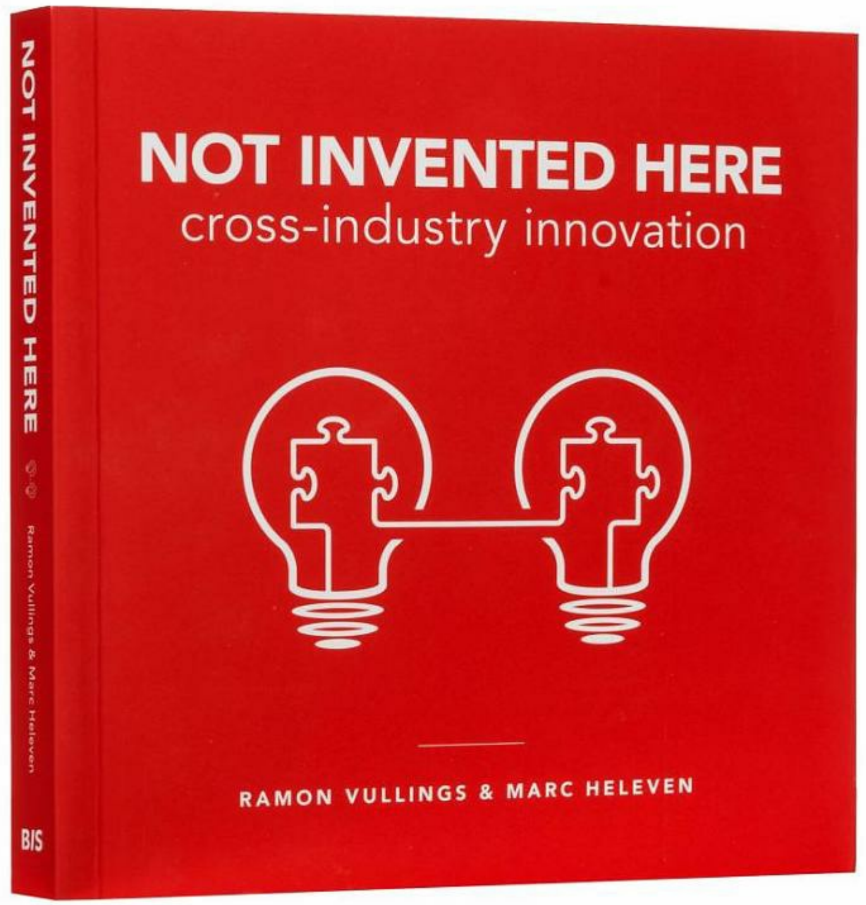


Margeta et al. 2015, Joint work with Inria and Microsoft Research Cambridge

See Label Propagation example in Scikit-learn

Be practical

Have an open mind



Overall experience

- Remarkable ecosystem
- Fantastic community
- Build things very hard before

★★★★★

Takeaways

- Build something you care about
- Poke your models and learn from them
- Pick your superpower and have fun

```
# to fly

import antigravity
```

python powered

The rod of Asclepios

Jan Margeta | jan@kardio.me | @jmargeta

KardioMe

Thanks!

Python, numpy, MxNet, Keras, Tensorflow, scikit-learn, SimpleITK, pydicom, Flask, Sanic, Django, PostgreSQL, ReactJS, Docker

Inria, IHU Liryc, Microsoft Research