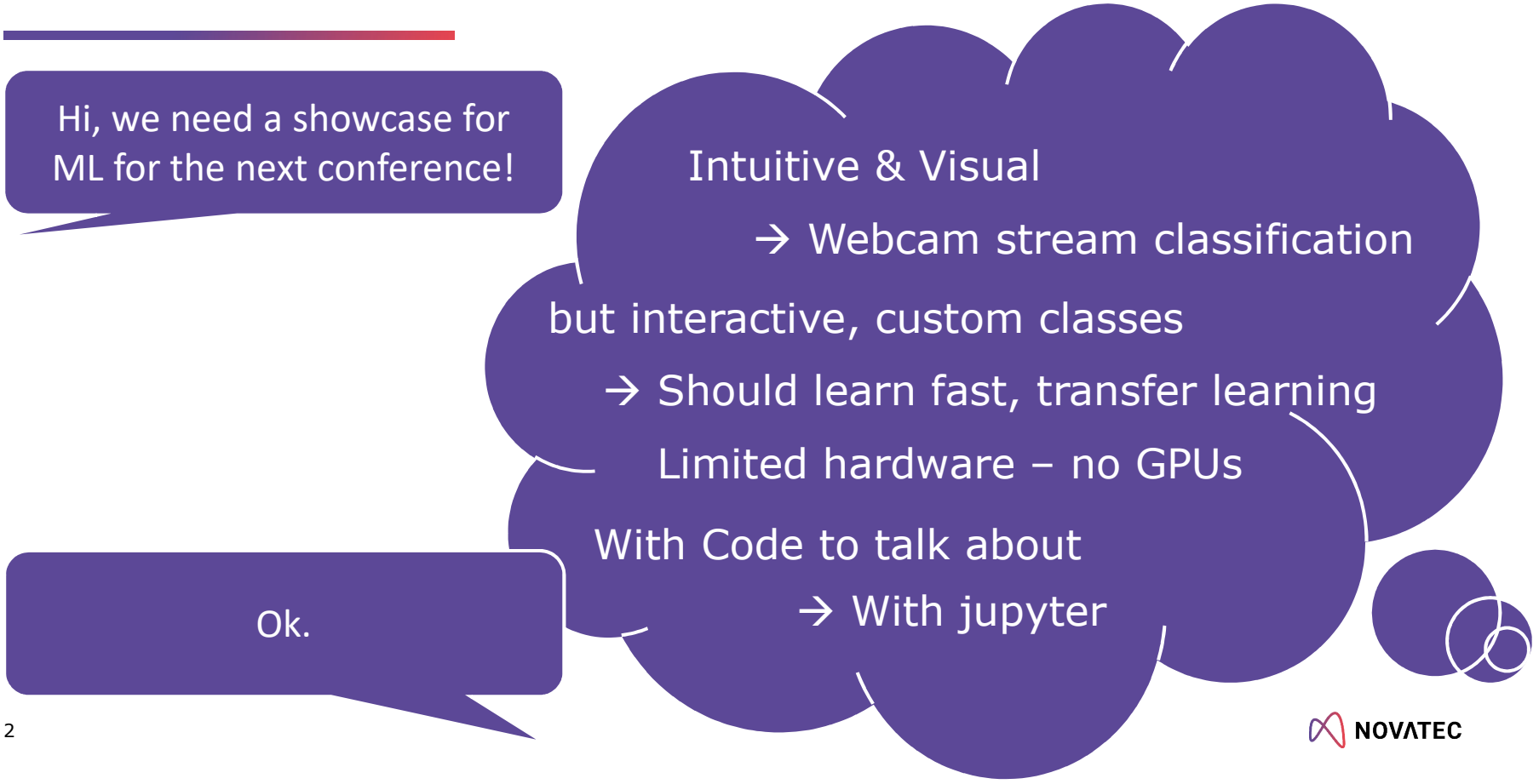

NOVATEC
Creating an Interactive ML Conference Showcase
Harald Bosch
PyCon / PyData Berlin – 09.10.2019

What do we want to achieve today?



NOVATEC

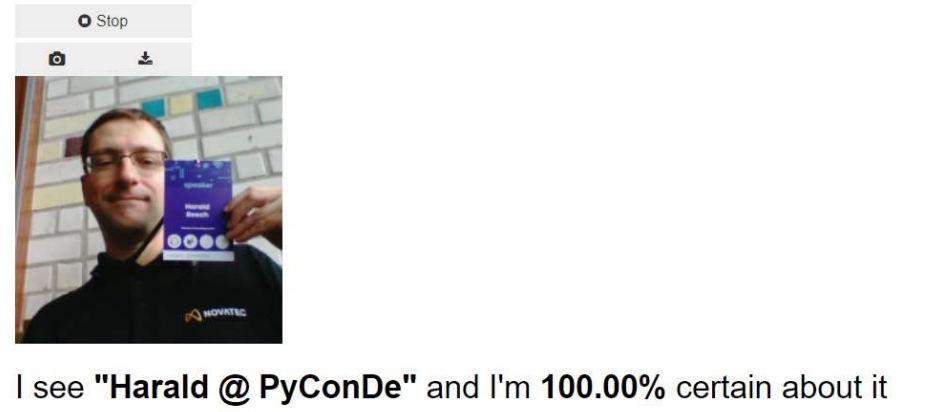
Transfer Learning Example

NOVATEC

What we do differently

- Python
- Arbitrary number of classes
- Notebook style
  - less polished…
  - … but visible code / structure
  - hackable
- Self contained
  - local deployment

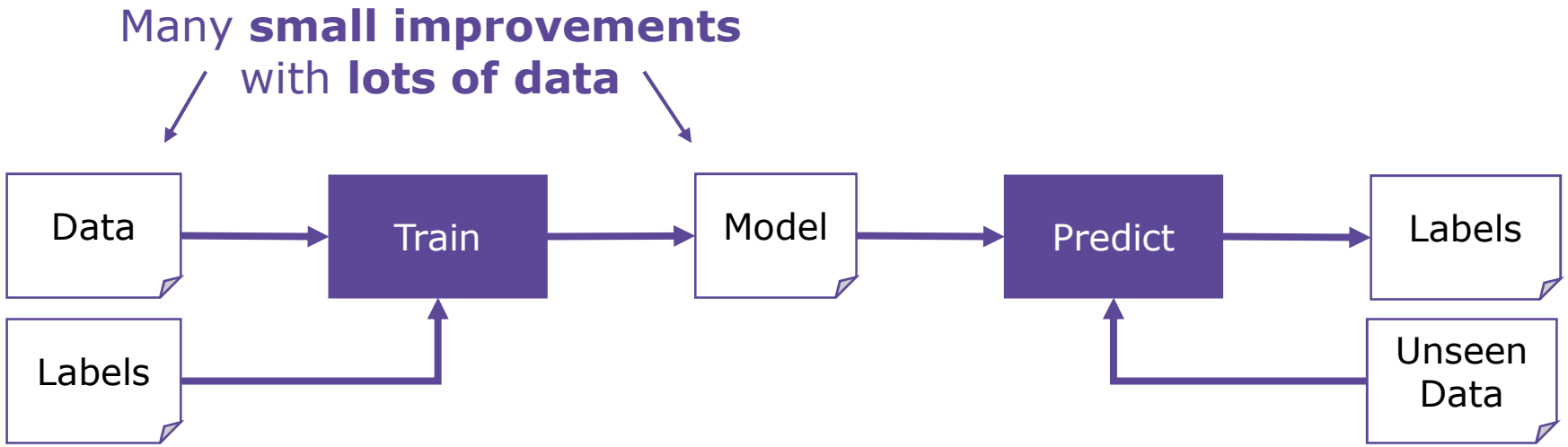
```
# combine it into one cell
widgets.VBox([button, inference_recorder, resultLabel])
```



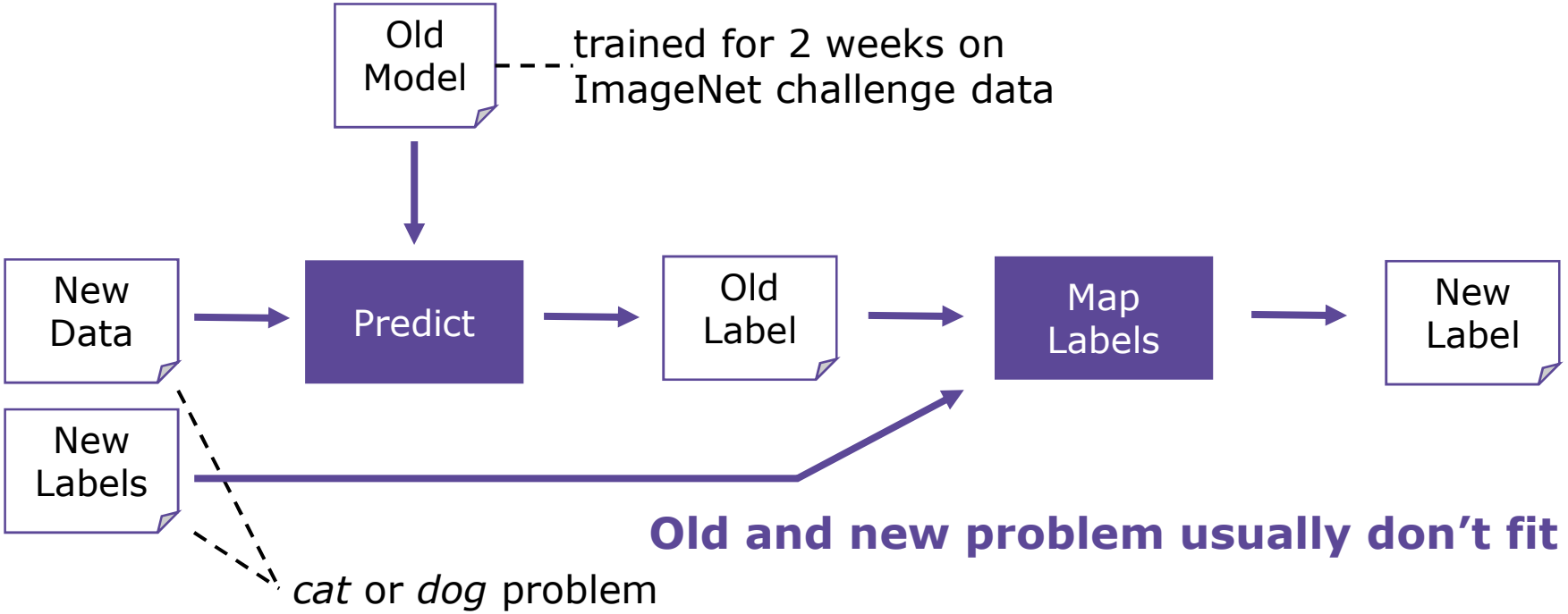
NOVATEC

NOVATEC

Standard Machine Learning Pipeline

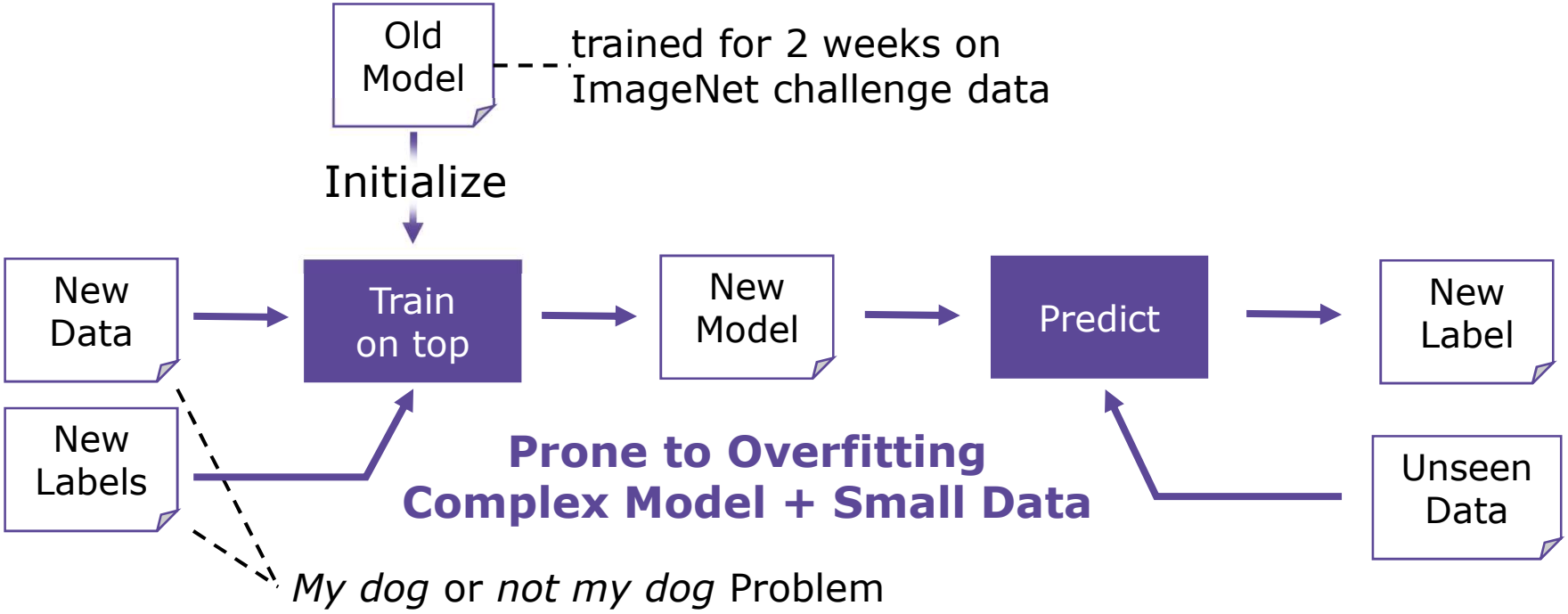


Transfer Learning Pipeline – Pretrained Model



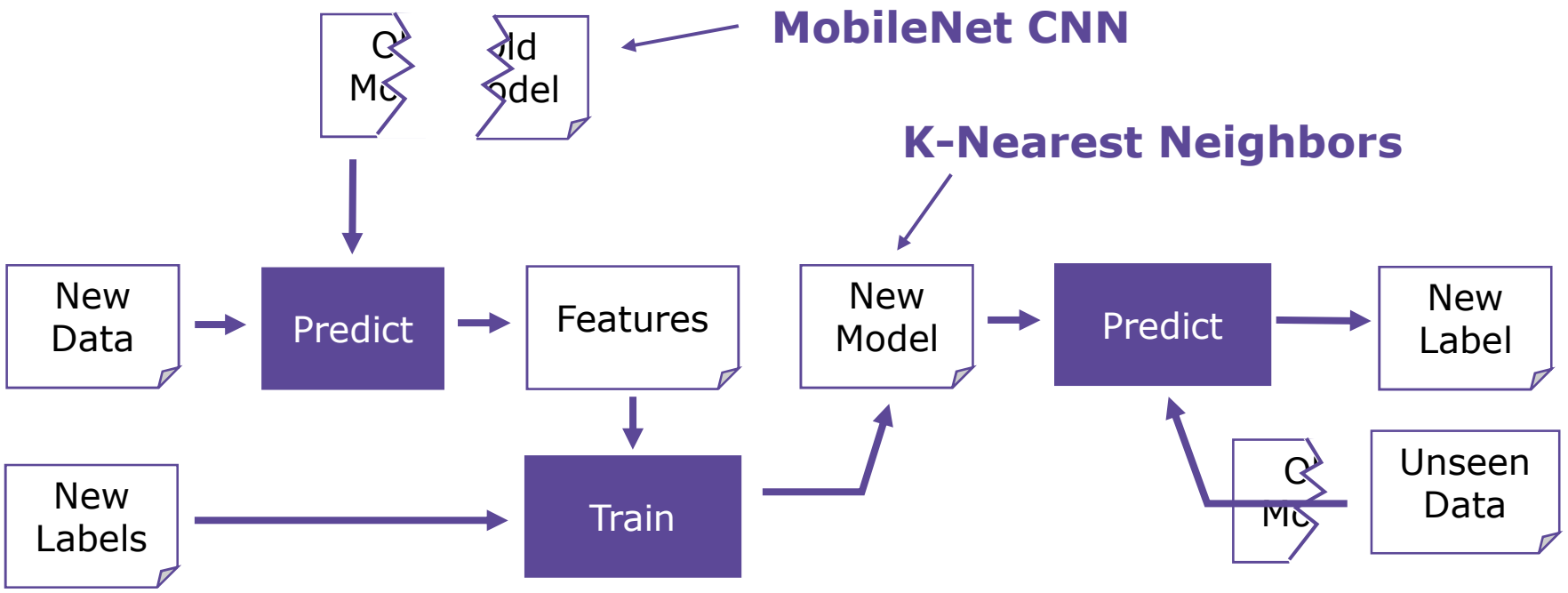
NOVATEC

Transfer Learning Pipeline – Fine-Tune Model



NOVATEC

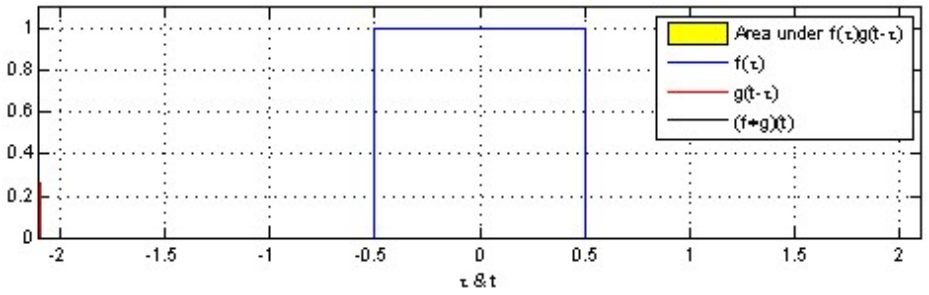
Transfer Learning Pipeline – Feature Extractor



NOVATEC

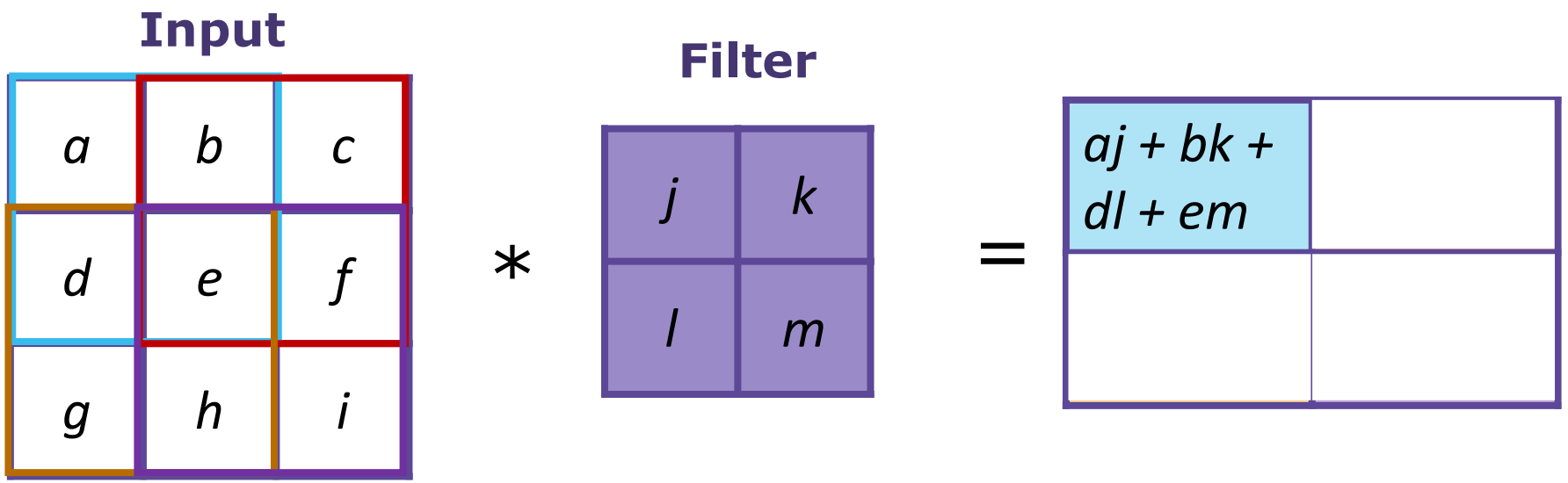
Convolutional Neutral Networks (CNN)

- Developed by LeCun et al. 1998
- Why convolutional?
  - Because they use a **convolution** instead of a general matrix multiplication.
- What is a convolution?
  - "Filters" in image processing
  - The sum of the element-wise product between two matrices, in the discrete case

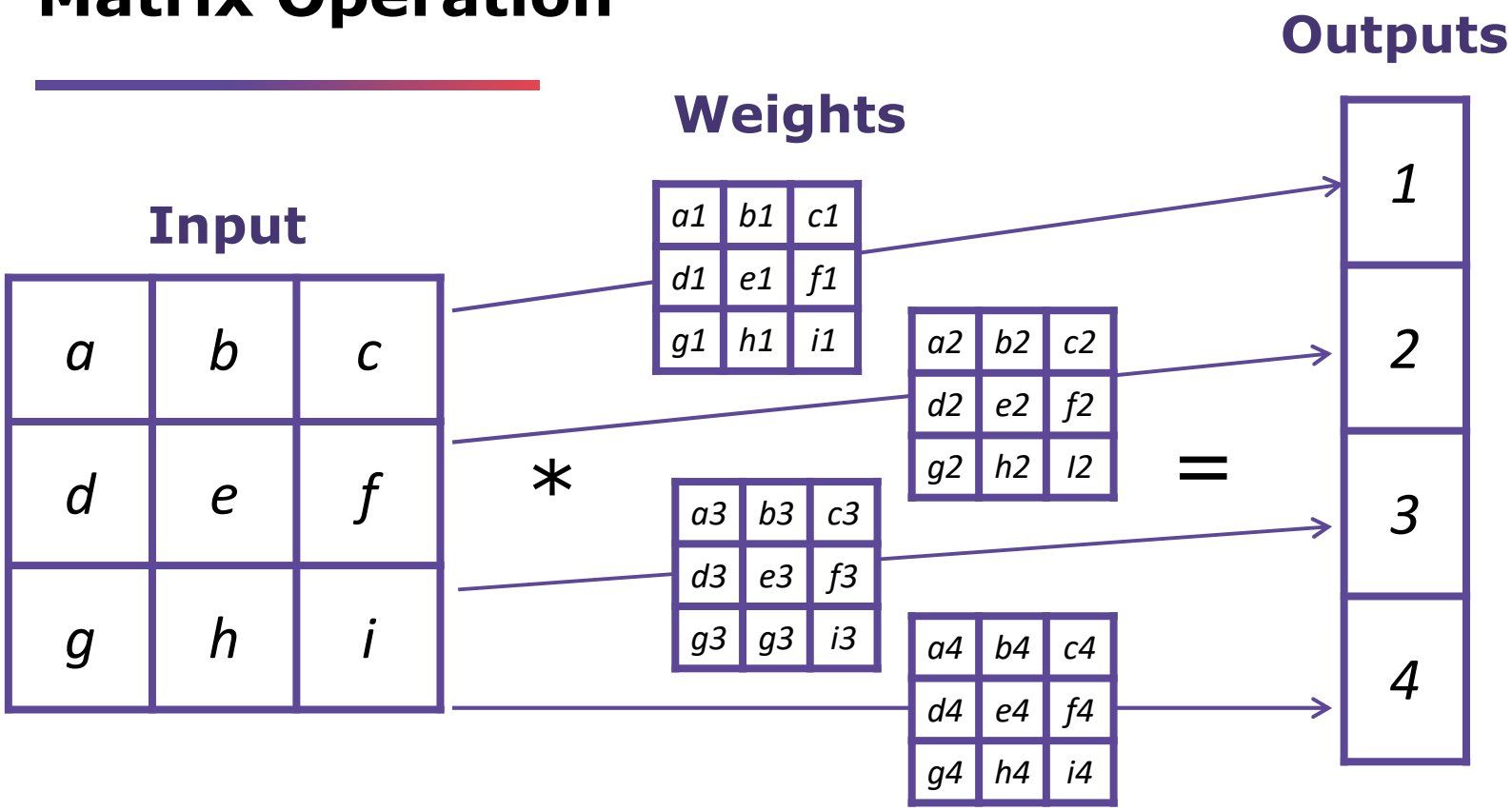


Video Source: https://en.wikipedia.org/wiki/Convolution

Convolution Operation



Matrix Operation

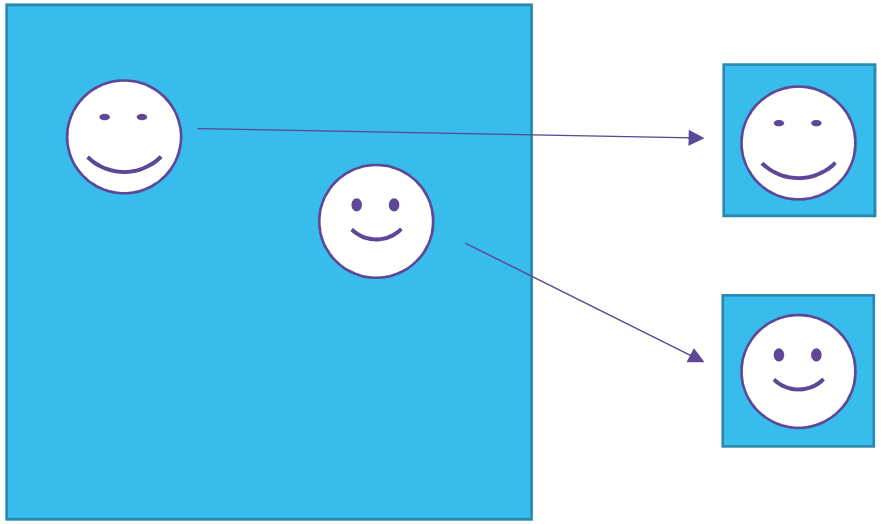


NOVATEC

Motivation for CNNs – Neighborhood

NOVATEC

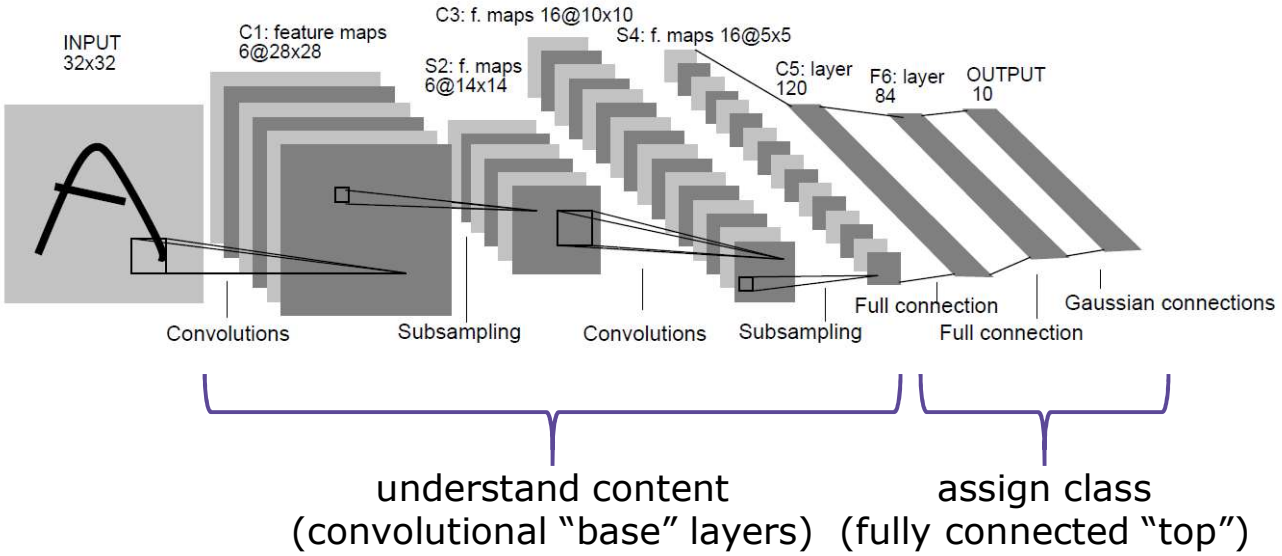
Motivation for CNN – Location Independent



Faces at different locations

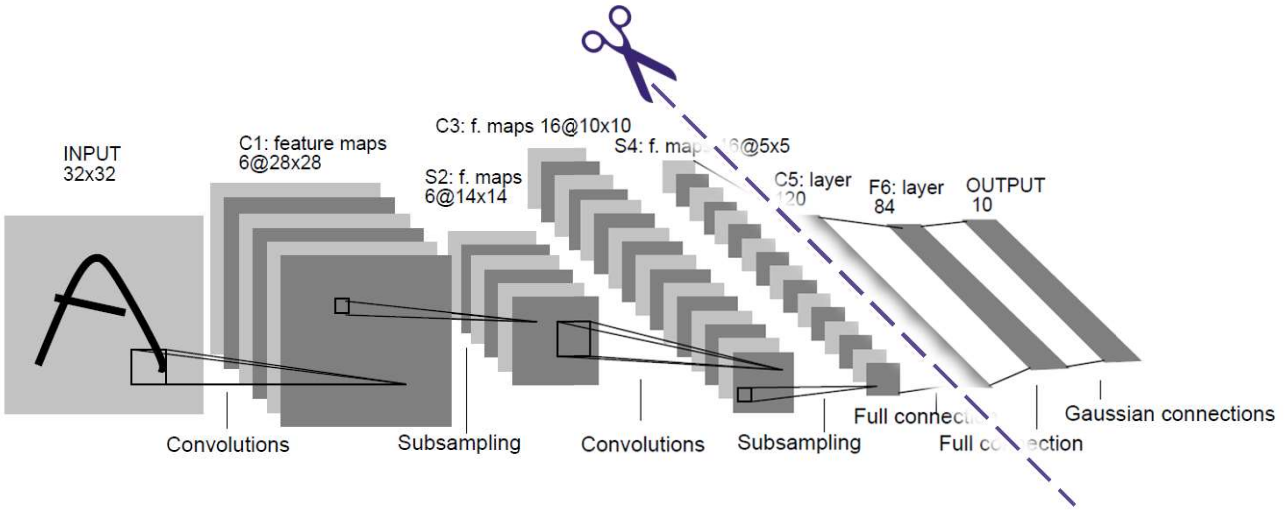
Shared Weights!

CNN Structure Example LeNet-5



Source: LeCun et al., 1998

CNN as Feature Extractors

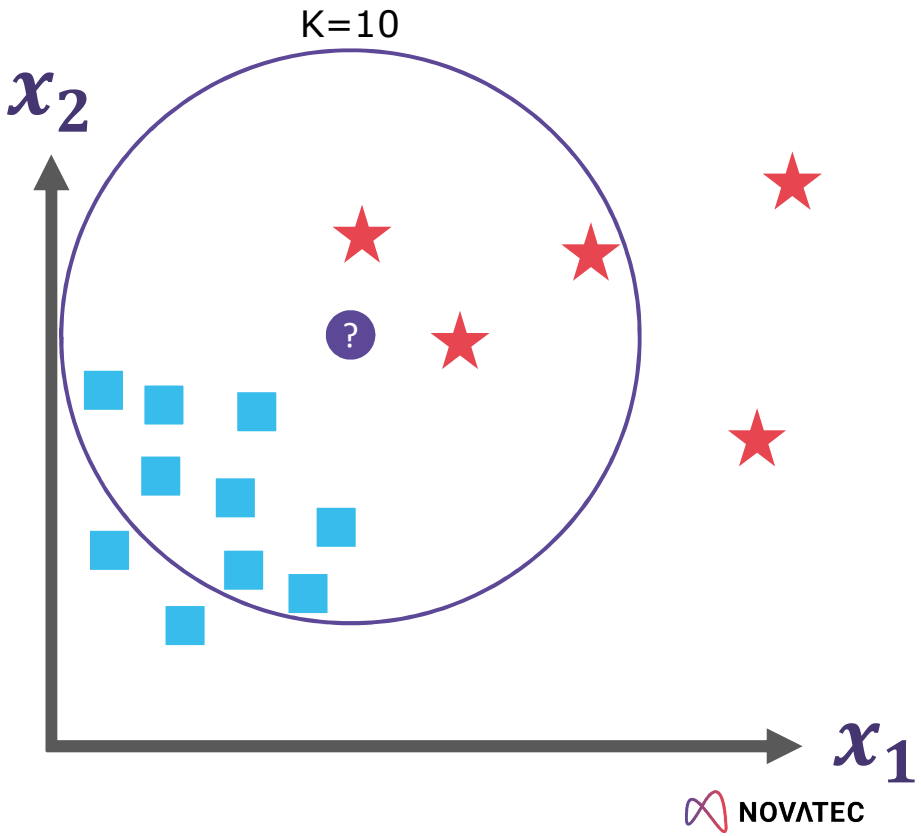


Source: LeCun et al., 1998

NOVATEC

K Nearest Neighbors

- Most simple classifier
- Training:
  - Memorize all labeled examples
- Inference:
  - Find k nearest examples
  - Make a majority vote



NOVATEC

Implementation

NOVATEC

Notebook Style

- Why Jupyter?
  - Can easily be deployed
    - via docker and browser
    - remote and local alike
  - One place for code and story (mainly for collegues)
  - Cell structure is great to explain process
  - Be flexible
    - Show intermediate results if wanted
    - Do ad hoc changes

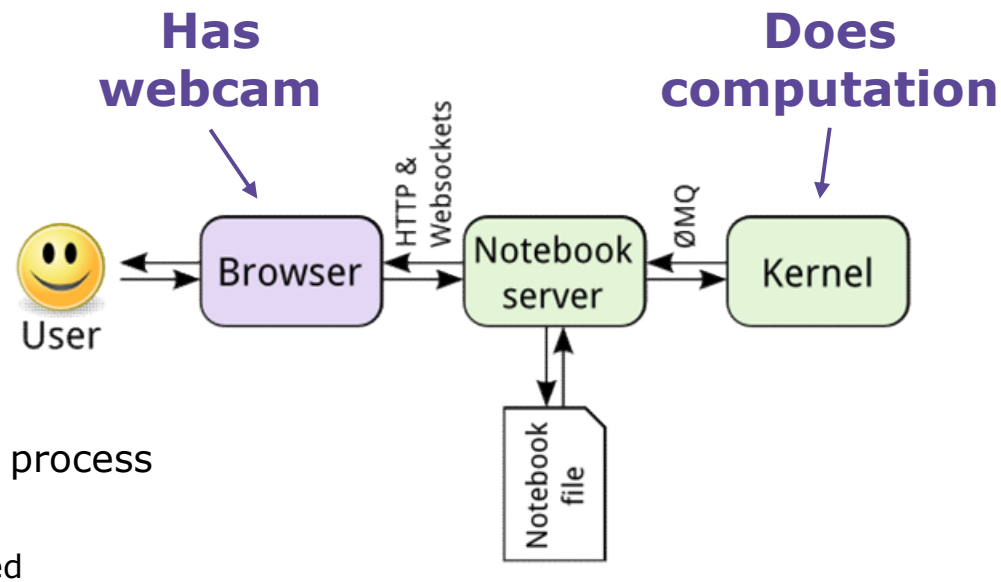


Image Source: https://jupyter.readthedocs.io

IPyWebRTC

- Used to access webcam image from Python
- Based upon ipywidgets
  - syncronized widget model
  - widgets to play video, audio, …
- Additionally
  - Exposes MediaStream API and WebRTC
  - Wrapper for webcam media stream and media recorders
  - Allows to send content to peers

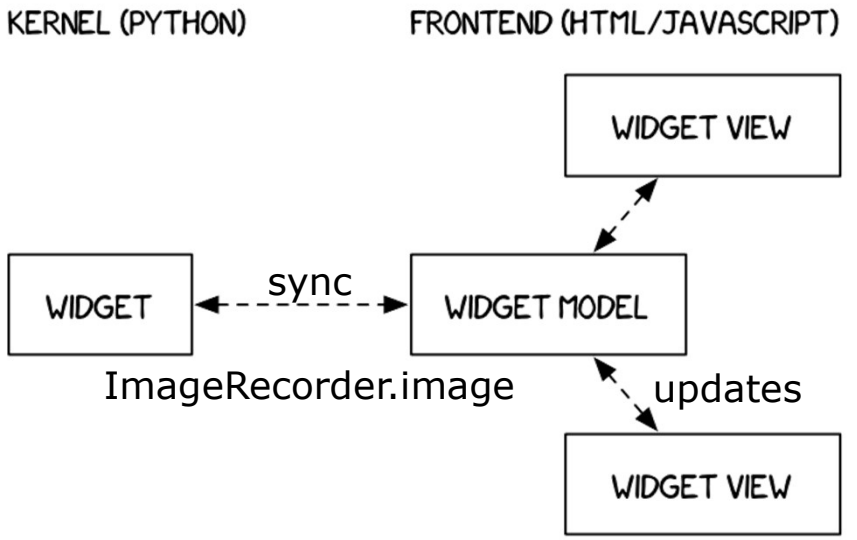


Image Source: https://ipywidgets.readthedocs.io/en/latest/examples/Widget%20Basics.html

Obtaining the Webcam content

- Create MediaStream from webcam
  - Constraints are passed on to [1]
- Create ImageRecorder from webcam stream
- Record image (omitted)
- PIL can translate PNG to RGBA
- BytesIO enables file like (i.e. random) access to the byte content

```
camera = CameraStream(
    constraints = {'video': {'width': 224, 'height': 224 }}
)
image_recorder = ImageRecorder(stream=camera)
...
img = PIL.Image.open(io.BytesIO(image_recorder.image.value))
```


[1]: https://developer.mozilla.org/en-US/docs/Web/API/MediaDevices/getUserMedia

Load Pretrained Model

- Keras' *applications* provides easy access to pretrained networks
  - Load model without classification layers (include_top = False)
- Pretrained on ImageNet Challenge Data
  - 1.2 Mio images
  - 1000 classes
- Input Shape
  - 224 x 224 pixel, 3 RGB channels
  - default values for this model
- Pooling
  - Reduce feature vector
  - Take only the average pixel value for each channel

```
cnn = applications.mobilenet.MobileNet(
    include_top = False,
    weights = 'imagenet',
    input_shape = (224, 224, 3),
    pooling = 'avg',
)
```

Extract Image Features

- Obtain raw values from image
- Remove alpha channel
- Appy same preprocessing as in the original training data
  - Depending on backend, basically scaling to [-1;1]
- Wrap into array and „predict" the features
  - Keras, Scikit-Learn etc. usually work with data sets not individual instances
  - (224,224,3) → (1,224,224,3)

```
def extract_features(img):
    img_array = np.asarray(img, dtype='int')
    img_wo_alpha = img_array[:, :, :3]
    img_wo_alpha = mobilenet.preprocess_input(img_wo_alpha)
    return cnn.predict(np.array([img_wo_alpha]))
```

Callbacks for fast data collection

- As a widget, ImageRecorder's properties are observable Traitlets
- This allows to register a callback on each value change.
- Each value change
  - Extracts the features from the change event
  - Stores features as input and label as desired output in global lists for training data

```
def training_callback(change):
    img = PIL.Image.open(io.BytesIO(change.new))
    x = extract_features(img)
    y = label
    X = np.append(X, x, axis=0)
    Y = np.append(Y, y)

training_recorder = ImageRecorder( stream = camera )
training_recorder.image.observe(training_callback, 'value')
```

Train Classifier

- scikit-learn's implementation of KNN
  - (or any other of their classifiers!)
  - Create Model
  - Fit Model
  - Done!

```
classification_model = neighbors.KNeighborsClassifier()
classification_model.fit(X, Y)
```

Use classifier

- Redo same steps as in the data collection
  - obtain image, preprocess, extract features
- Predict the most probable label
  - Returns array of size #images
- Get the probabilities
  - Returns array of size #images x #classes

```
def classify(image):
    feature_vector = extract_features(image)
    prediction = classification_model.predict(feature_vector)
    prediction_proba = classification_model.predict_proba(feature_vector)
    return prediction[0], np.max(prediction_proba)
```

Link everything with callbacks

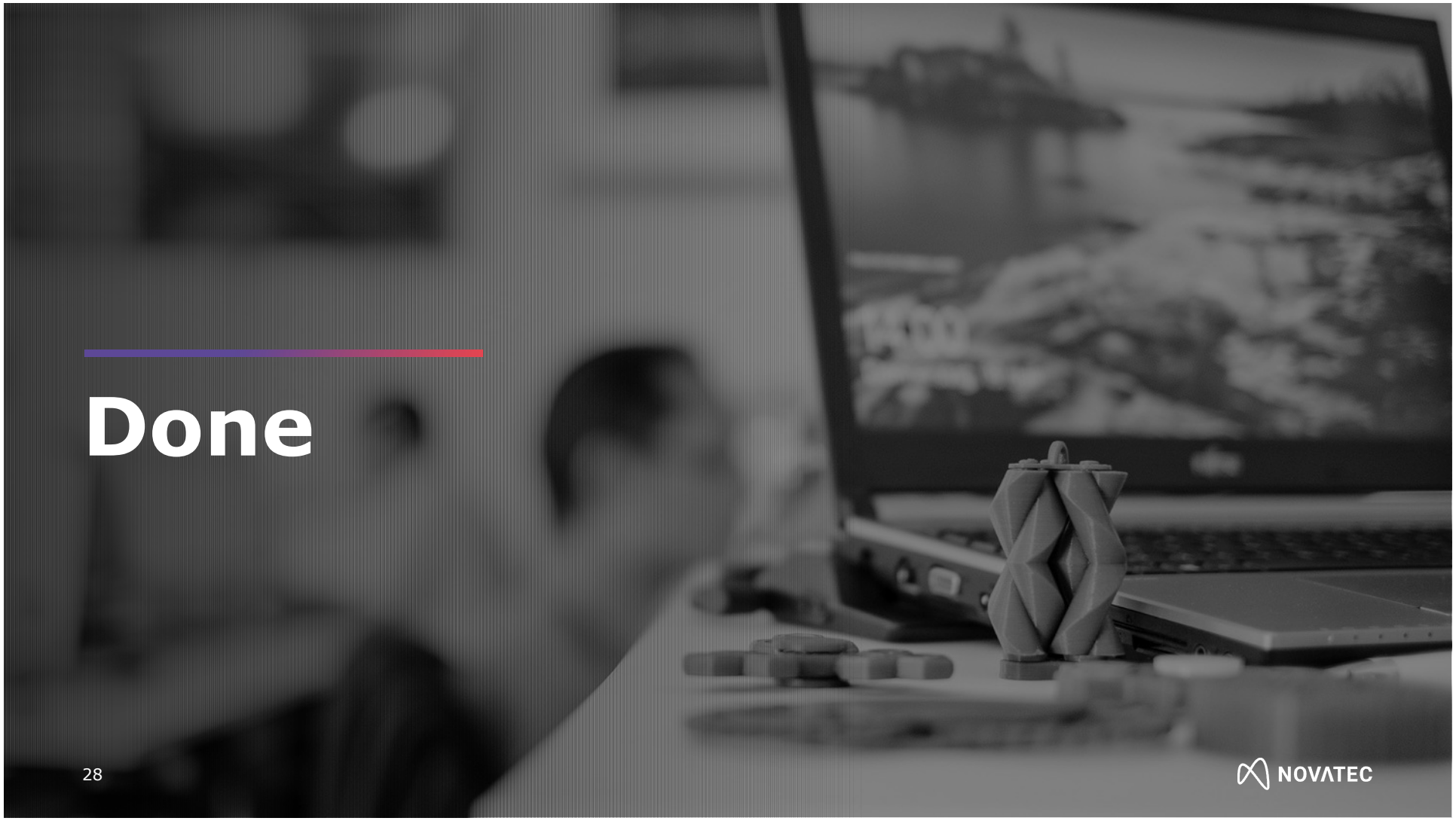
- Output widget
  - HTML label
- Callback
  - uses the classifier
  - updates the output
  - Programmatically takes another image
  - Thus triggering another callback
- Combine ImageRecorder and output label e.g. via a vertical layout

```
resultLabel = widgets.HTML()

def inference_callback(change):
    img = PIL.Image.open(io.BytesIO(change.new))
    result, conf = classify(img)
    resultLabel.value = "{} - {}%".format(result, conf)
    if (proceed):
        inference_recorder.recording = True

recorder = ImageRecorder(stream=camera)
recorder.image.observe(inference_callback, 'value')

widgets.VBox([recorder, resultLabel])
```



NOVATEC

Dr. Harald Bosch
Senior Consultant

E-Mail: harald.bosch@novatec-gmbh.de

Novatec Consulting GmbH
Dieselstraße 18/1
D-70771 Leinfelden-Echterdingen

T. +49 711 22040-700
info@novatec-gmbh.de
www.novatec-gmbh.de