Deep Learning für visuelle Inspektionen

VDI-TUM Expertenforum 2018
Fritjof Büttner

Problemstellung

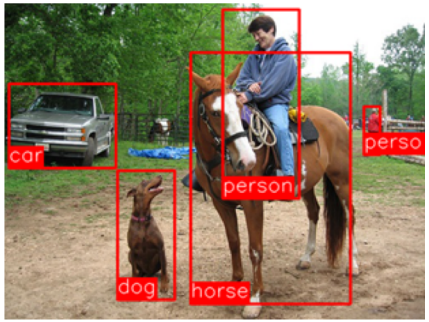
Gasturbinen in der Luftfahrt müssen alle 1-2 Jahre von einem Experten auf Defekte untersucht werden.

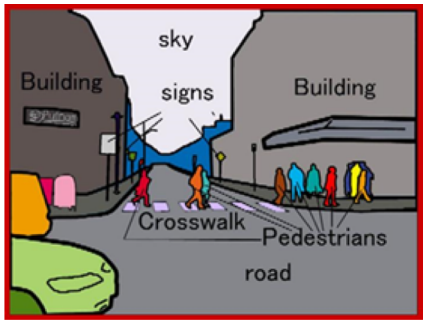
Was ist *Deep Learning*?

Anstelle von festen Instruktionen: Den Computer selbstständig aus Beispielen “lernen” lassen

```
def denseSift(imName, stride=5):
    print 'dense sampling',
    imRGB = imread(imName)
    [rows,cols,clrs] = imRGB.shape

    # divide 1 stride on both sides
    offsetR = rows % stride
    offsetC = cols % stride

    r = np.arange( offsetR/2, rows, stride)
    c = np.arange( offsetC/2, cols, stride)
    x, y = meshgrid(c, r)
    grid = np.array( [x.flatten(), y.flatten()] ).T
    print 'nr Points', grid.shape[0]
    return grid

def computeHes(imName, sigma=1, magThreshold = 10, hesThreshold=5, NMSneighborhood = 10):
    #print 'hessian'
    imRGB = imread(imName)

    # convert to grayscale
    imHSV = matplotlib.colors.rgb_to_hsv(imRGB)
    im = imHSV[:,:,2]

    # Derivatives
    dxx = filters.gaussian_filter(im, sigma=sigma, order = [2,0])
    dyy = filters.gaussian_filter(im, sigma=sigma, order = [0,2])
    lapl = sigma * (dxx + dyy)

    # non max suppression and thresholding of maxima
    data_max = filters.maximum_filter(lapl, NMSneighborhood)
    maxima = (lapl == data_max)
    maxima = logical_and( maxima, data_max > hesThreshold)

    # non max suppression and thresholding of minima
    data_min = filters.minimum_filter(lapl, NMSneighborhood)
    minima = (lapl == data_min)
    minima = logical_and( minima, data_min < -hesThreshold)

    extrema = logical_or(maxima, minima)

    dx = filters.gaussian_filter(im, sigma=sigma, order = [1,0])
    dy = filters.gaussian_filter(im, sigma=sigma, order = [0,1])
    mag = sigma * sqrt( dx*dx + dy * dy )
    extrema = logical_and( extrema, mag > magThreshold)

    print 'Hes, nr Points', sum(extrema)

    [r,c] = np.where(extrema)

    return np.array([c,r]).T

def computeHar(imName, sigma=1, k=0.04, magThreshold = 10, NMSneighborhood=10):
    #print 'harris'

    dx, dy = computeDerivs(imName,sigma=sigma)
```

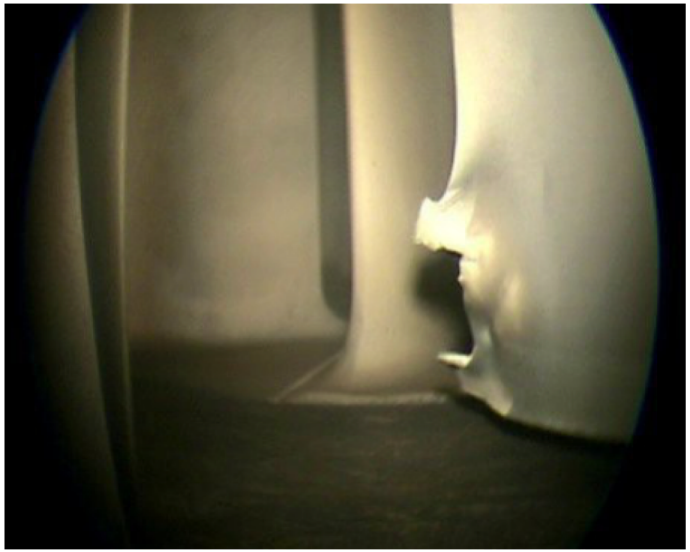
vs.





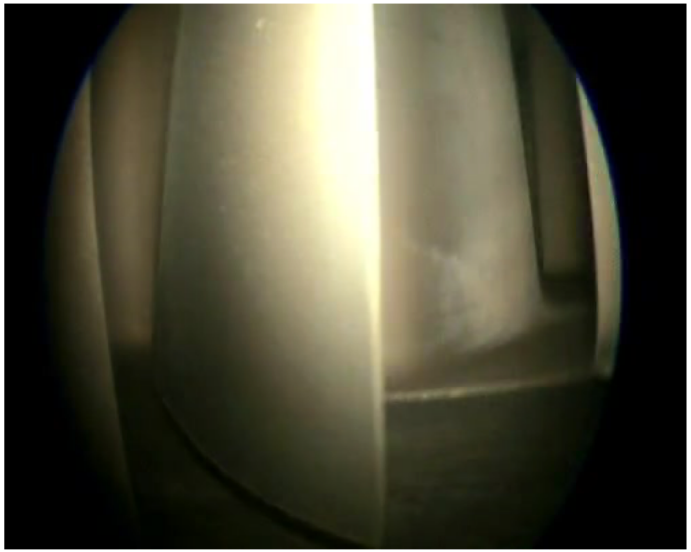
Ein zusätzliches Paar Augen...

Fehlererkennung



Wie sieht ein Fehler aus?

Anomalieerkennung



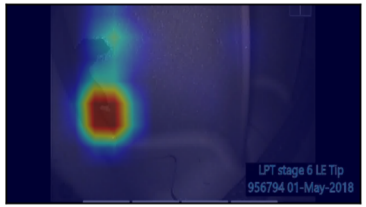
Wie sieht der Normalzustand aus?

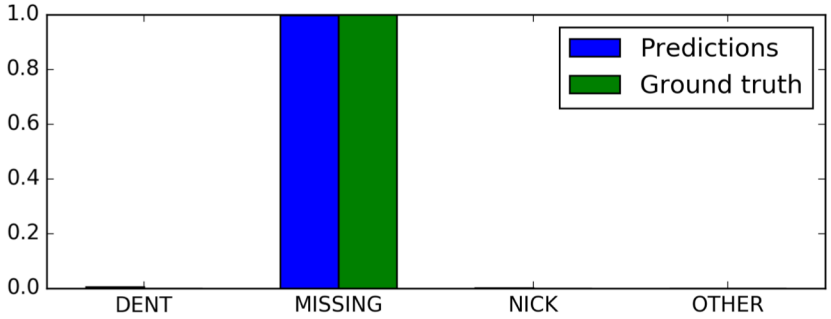
Fehlerklassifizierung

Ausgabe des Netzwerks:

- “Heatmap” des Eingabebildes markiert die kritischen Regionen für die Klassifizierung

- Wahrscheinlichkeiten für jede Fehlerklasse

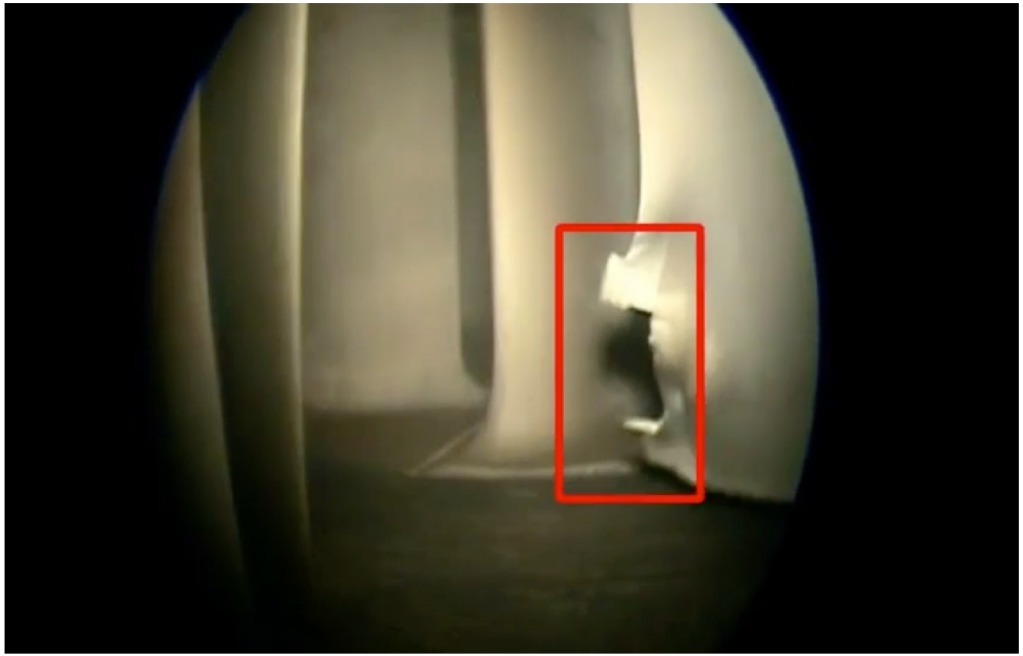




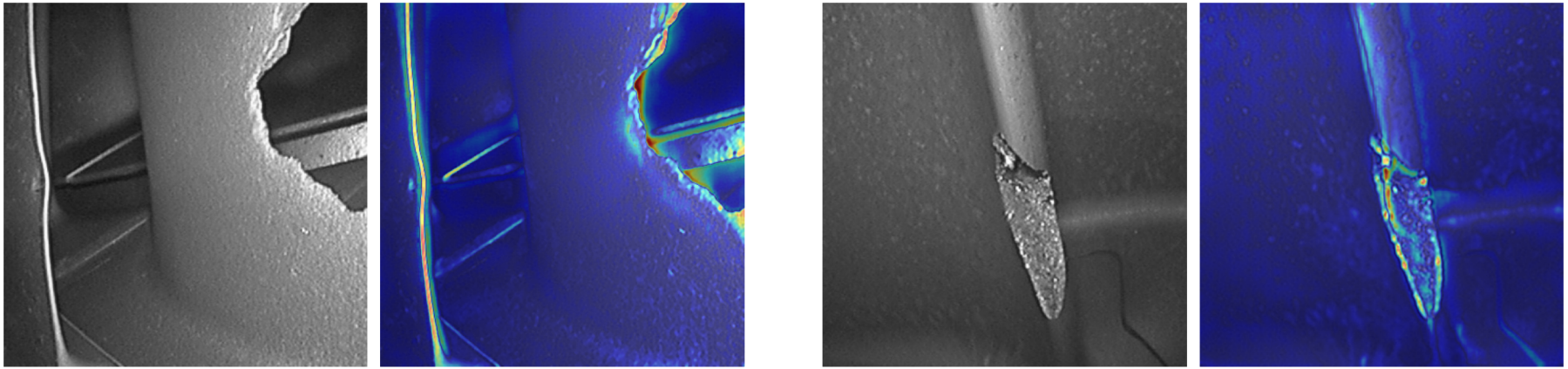
Fehlerlokalisierung

Ausgabe des Netzwerks:

“Bounding boxes” markieren Regionen, in denen ein Fehler erkannt wurde



Anomalieerkennung



Vielen Dank!

Kontakt: fritjof@aiir.nl