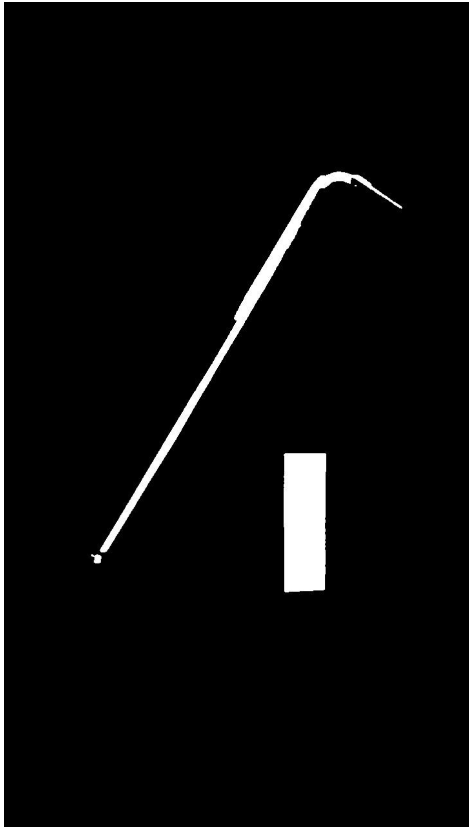
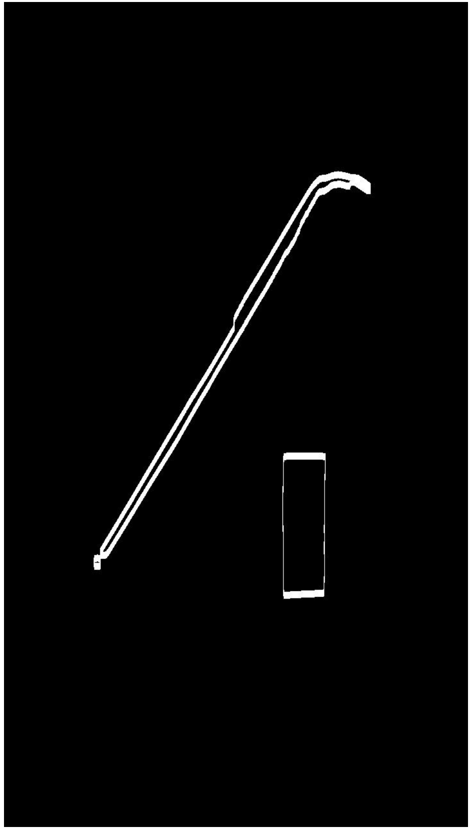
# Computer Vision - OpenCV

Frank Neumann, Moritz Klein

# Vorverarbeitung

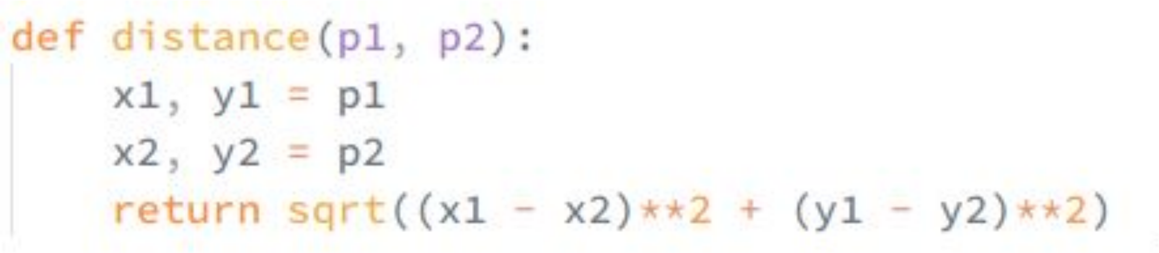
1. Bild Graustufen
2. Binarisierung
3. Noise entfernen
   - ○ Opening/Closing, Blur
4. Edges mithilfe von Canny detektieren





# Linien bestimmen

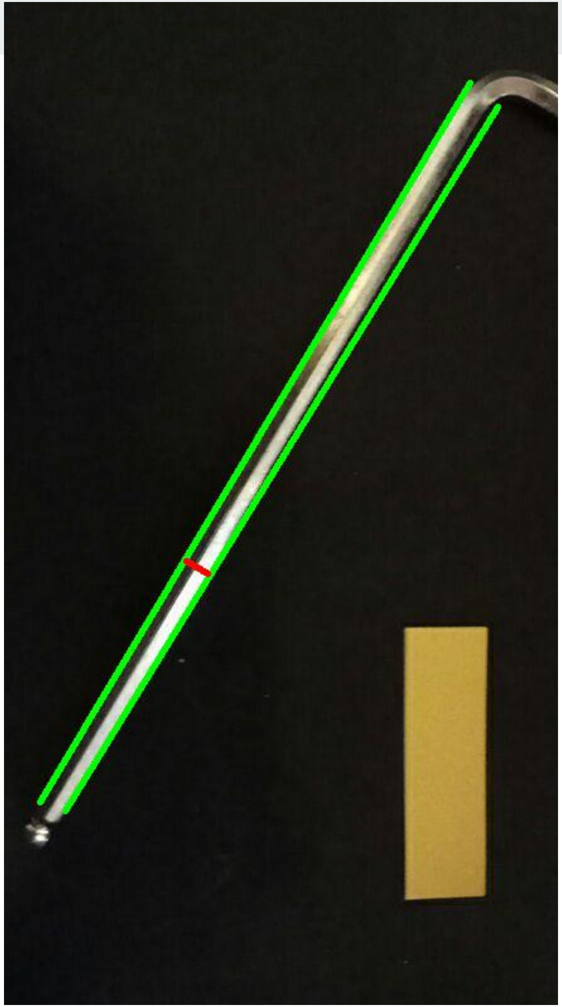
1. Linien mit HoughLines (Probabilistic) bestimmen
    ○ HoughLinesP gibt die Extremwerte der Linie zurück (P1, P2)
2. Längste Linie bestimmen
    ○ Euklidische Distanz



```
def distance(p1, p2):
    x1, y1 = p1
    x2, y2 = p2
    return sqrt((x1 - x2)**2 + (y1 - y2)**2)
```

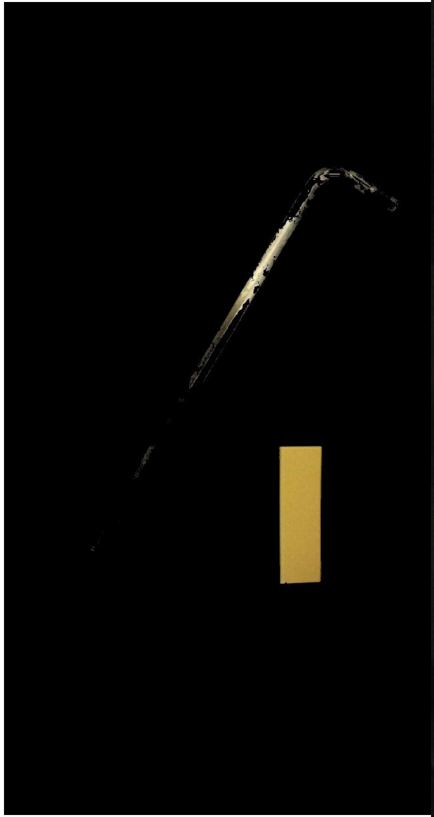
# Breite bestimmen (Konzept)

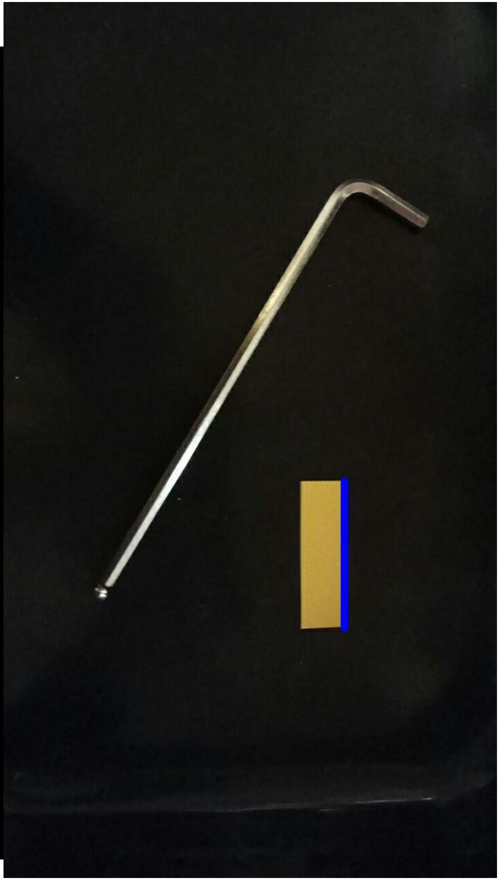
1. Die **beiden** längsten Linien bestimmen
2. Abstand zwischen beiden Linien gibt Breite an



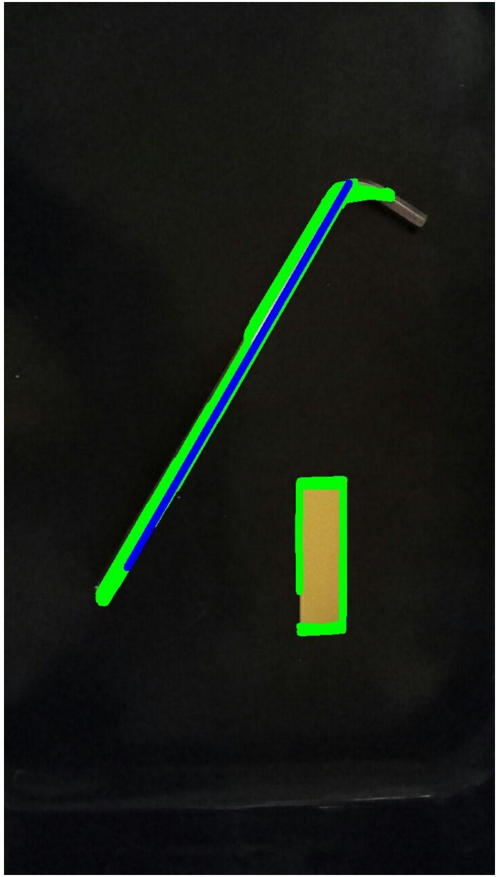
# Pixel zu Millimeter

1. Aus Bild nur Objekt maskieren
2. Aus maskiertem Bild längste Linie ermitteln
3. Pixel zu Milimeter Verhältnis:
   - ref_mm_length / longest_line_px





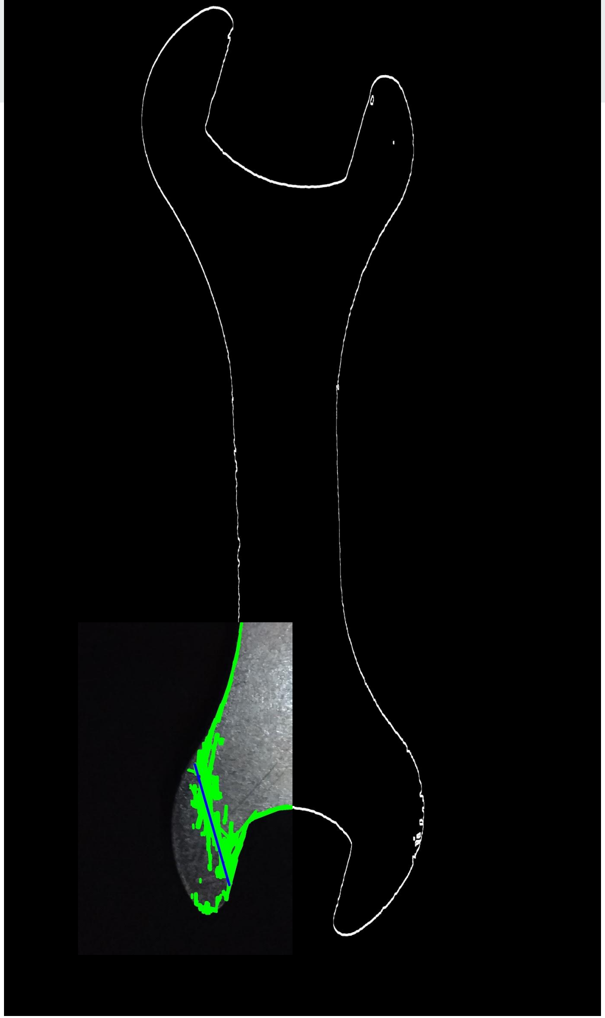
# Ergebnis



```
Pixel to mm ratio: 1:0.2314814814814814
Longest edge is 150.79592788125703mm /  651.4384084470304px
```

# Probleme

1. Binarisierung
2. Zu dünne Edges / Rundungen
3. Werkstück kleiner als Referenzobjekt
4. Breiten-ermittlung:
    - Grundannahme: Werkstück besitzt einen langen gleichseitigen *Griff*
5. Objekt liegt ungünstig / Verzeichnung



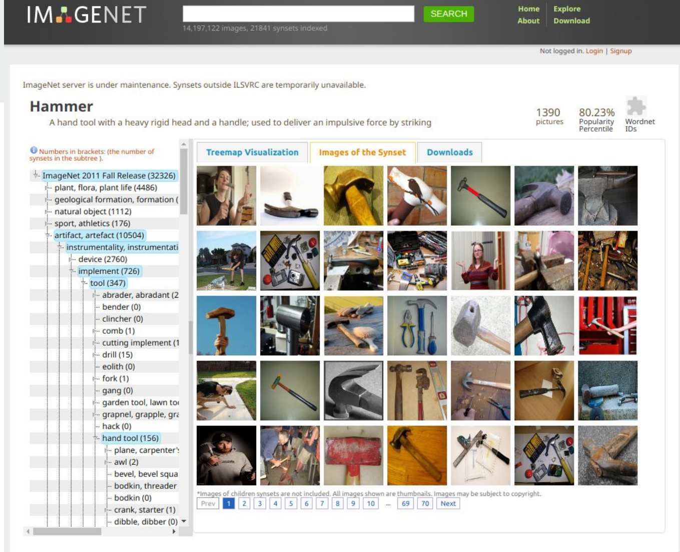
# Freie Aufgabe

1. Werkzeug klassifizieren
2. Werkzeug finden

# Vorgehen

1. Werkzeug klassifizieren
    a. Trainingsdaten
    b. Netzwerk
2. Werkzeug finden
    a. Rastern
    b. Canny

# Daten



IMAGENET
SEARCH
14,197,122 images, 21841 synsets indexed
Home
About
Explore
Download
Not logged in. Login | Signup

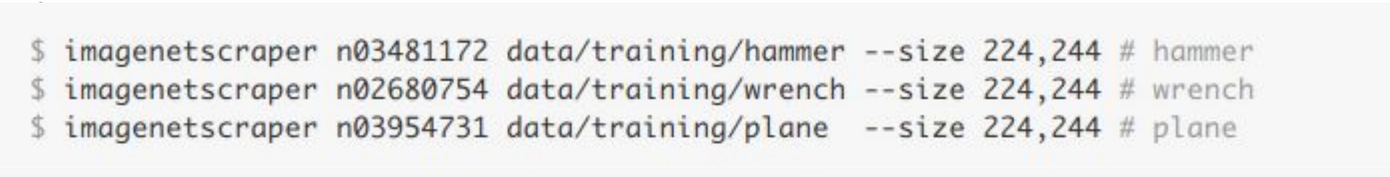
ImageNet server is under maintenance. Synsets outside ILSVRC are temporarily unavailable.

Hammer

A hand tool with a heavy rigid head and a handle; used to deliver an impulsive force by striking

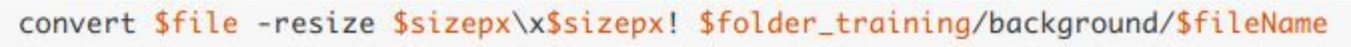
*Images of children synsets are not included. All images shown are thumbnails. Images may be subject to copyright.

- Imagnet: Hammer, Plane, Wrench (~ 1000 je Klasse)
  - Scraper [1]



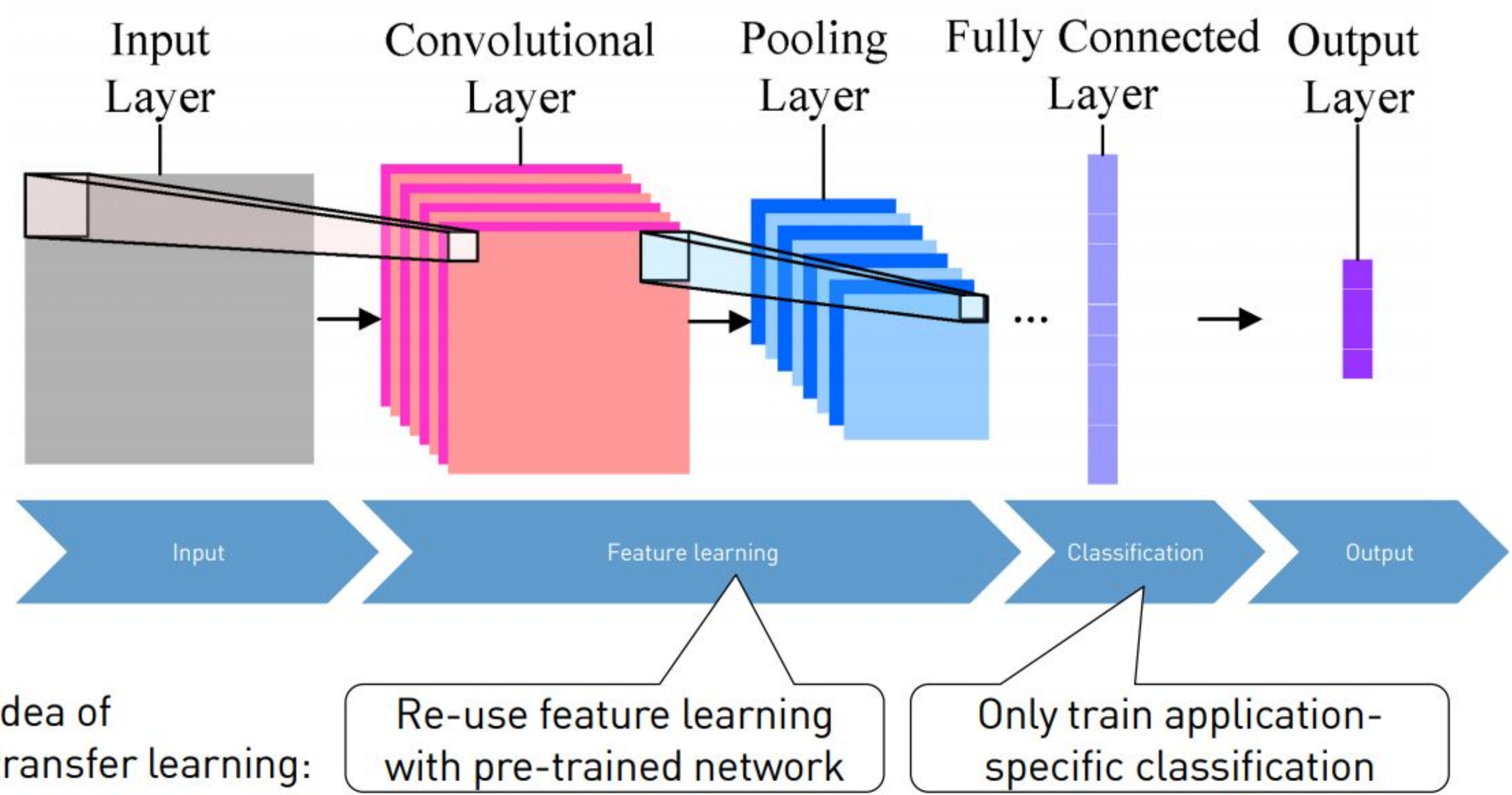
```
$ imagenetscraper n03481172 data/training/hammer --size 224,244 # hammer
$ imagenetscraper n02680754 data/training/wrench --size 224,244 # wrench
$ imagenetscraper n03954731 data/training/plane  --size 224,244 # plane
```

- UIUC: Background Texturen (~ 1600) [2]
  - Resize mit convert [3]



```
convert $file -resize $sizepx\x$sizepx! $folder_training/background/$fileName
```

# Transfer-Learning [4]

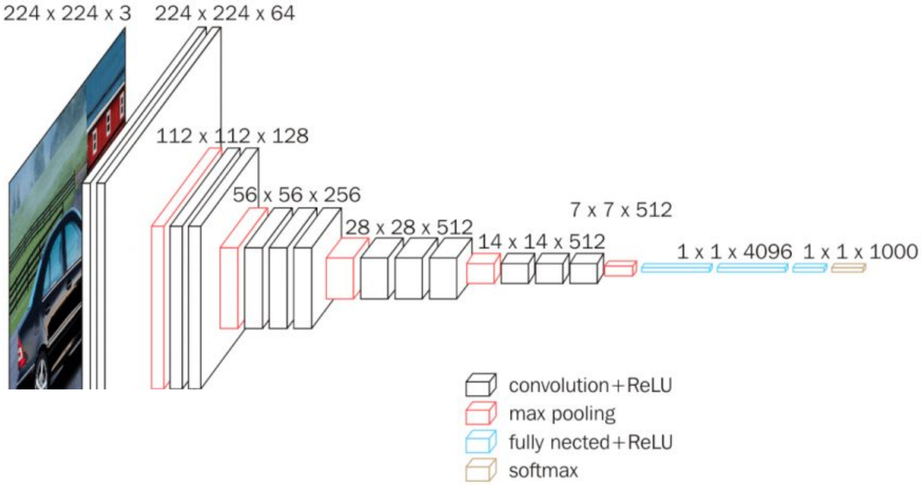


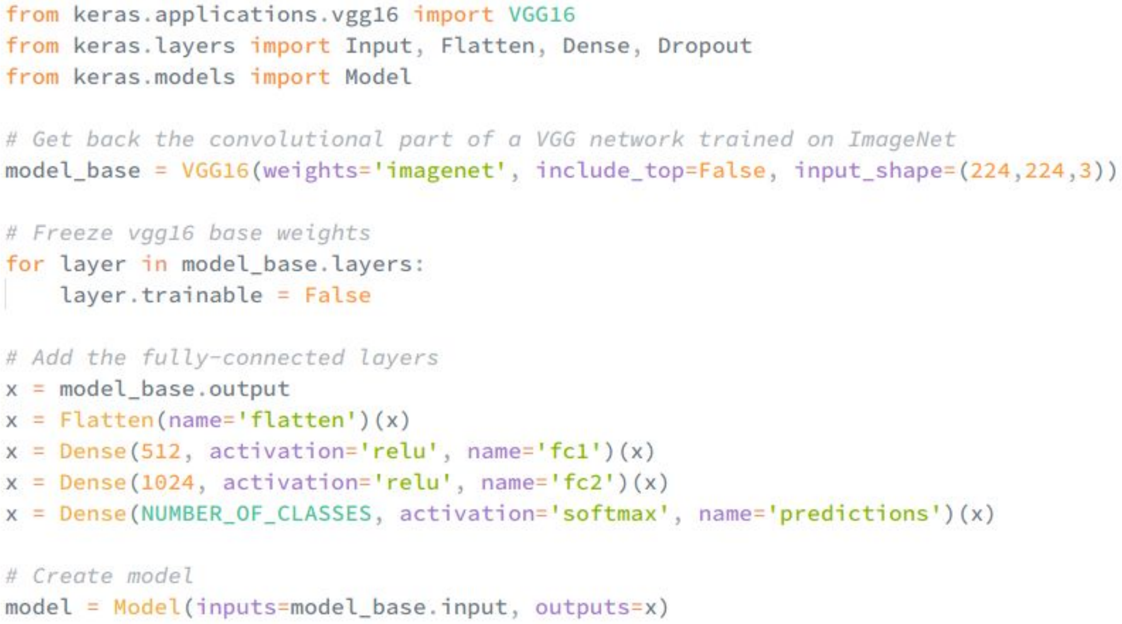
Idea of transfer learning:

Re-use feature learning with pre-trained network

Only train application-specific classification

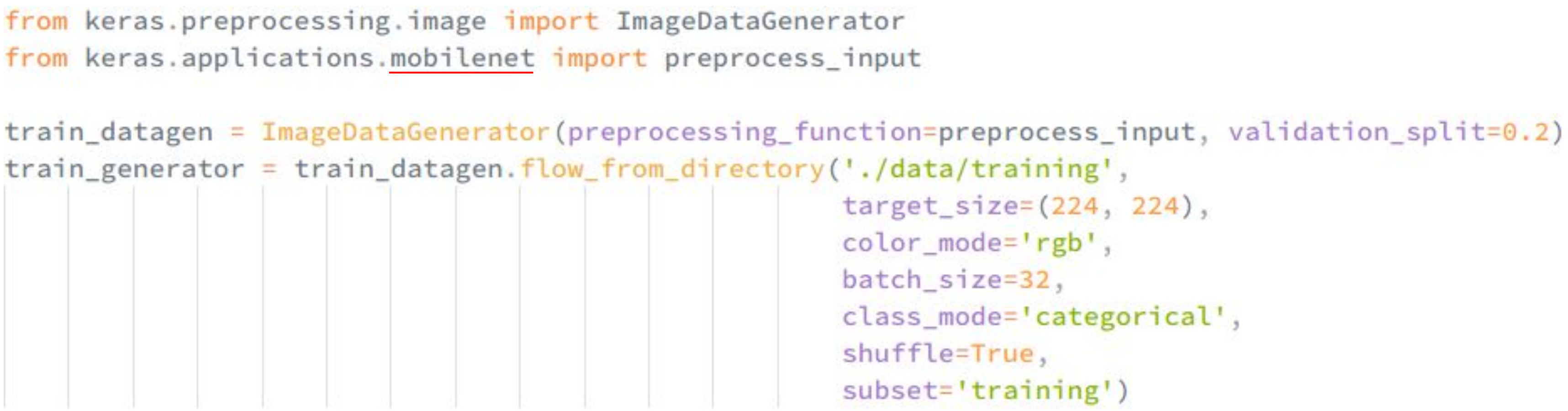
# VGG16 [5]





```
from keras.applications.vgg16 import VGG16
from keras.layers import Input, Flatten, Dense, Dropout
from keras.models import Model

# Get back the convolutional part of a VGG network trained on ImageNet
model_base = VGG16(weights='imagenet', include_top=False, input_shape=(224,224,3))

# Freeze vgg16 base weights
for layer in model_base.layers:
    layer.trainable = False

# Add the fully-connected layers
x = model_base.output
x = Flatten(name='flatten')(x)
x = Dense(512, activation='relu', name='fc1')(x)
x = Dense(1024, activation='relu', name='fc2')(x)
x = Dense(NUMBER_OF_CLASSES, activation='softmax', name='predictions')(x)

# Create model
model = Model(inputs=model_base.input, outputs=x)
```

# Daten laden

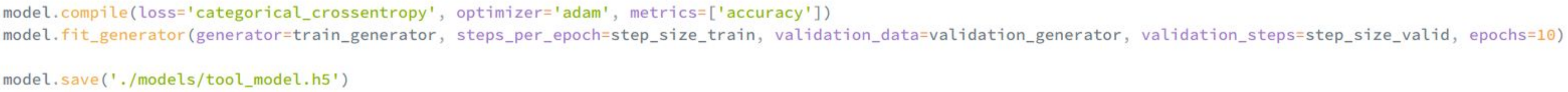


```
from keras.preprocessing.image import ImageDataGenerator
from keras.applications.mobilenet import preprocess_input

train_datagen = ImageDataGenerator(preprocessing_function=preprocess_input, validation_split=0.2)
train_generator = train_datagen.flow_from_directory('./data/training',
                                                    target_size=(224, 224),
                                                    color_mode='rgb',
                                                    batch_size=32,
                                                    class_mode='categorical',
                                                    shuffle=True,
                                                    subset='training')
```

# Training

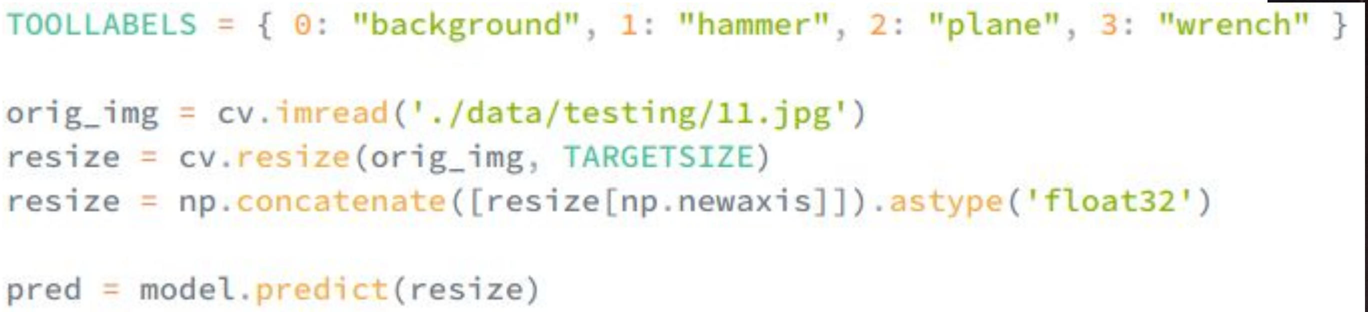
```
Epoch 10/10
114/114 [==============================] - 26s 225ms/step - loss: 2.3167e-04
- acc: 1.0000 - val_loss: 0.5279 - val_acc: 0.8980
```

- Training-time on CPU ~ 40min [i7-5820]
- Training-time on GPU ~ 5min [GTX 980]



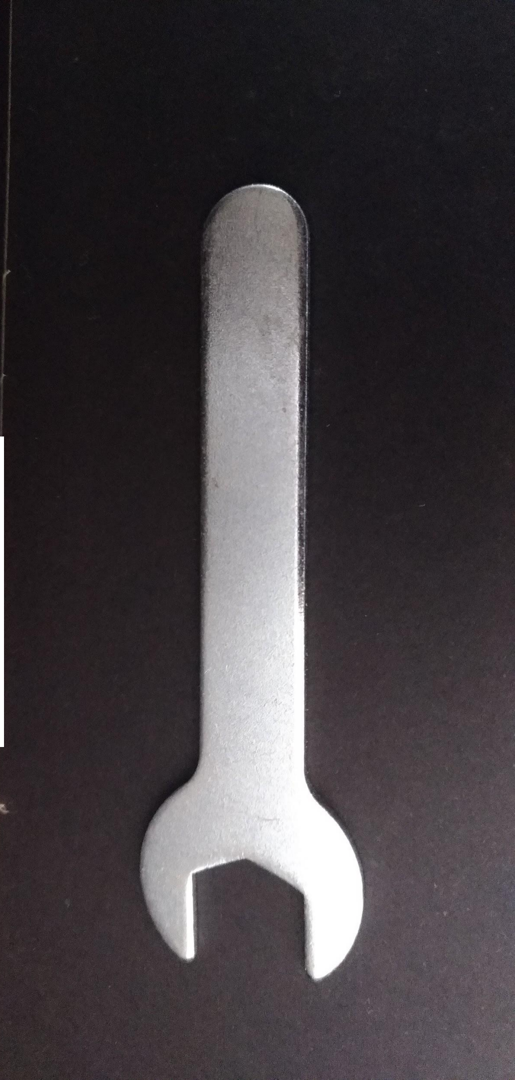
```
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model.fit_generator(generator=train_generator, steps_per_epoch=step_size_train, validation_data=validation_generator, validation_steps=step_size_valid, epochs=10)

model.save('./models/tool_model.h5')
```

# Ergebnis (1)



```
TOOLLABELS = { 0: "background", 1: "hammer", 2: "plane", 3: "wrench" }

orig_img = cv.imread('./data/testing/11.jpg')
resize = cv.resize(orig_img, TARGETSIZE)
resize = np.concatenate([resize[np.newaxis]]).astype('float32')

pred = model.predict(resize)
```

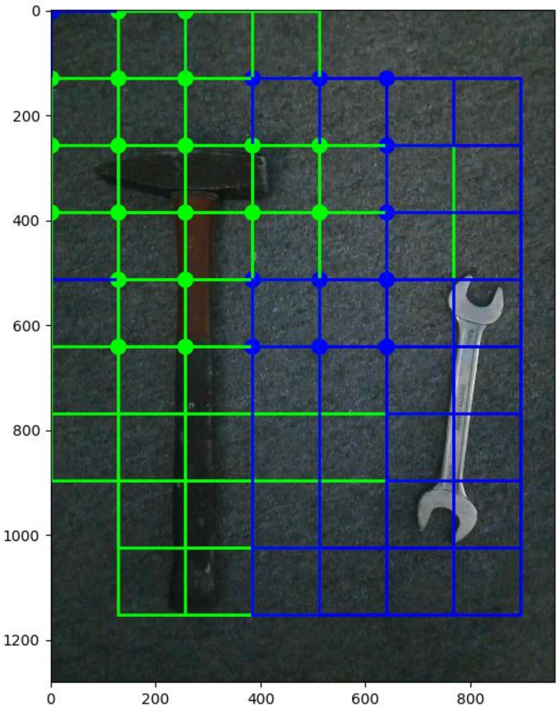
```
>> [0. 0. 0. 1.]
```



# Finden - Rastern

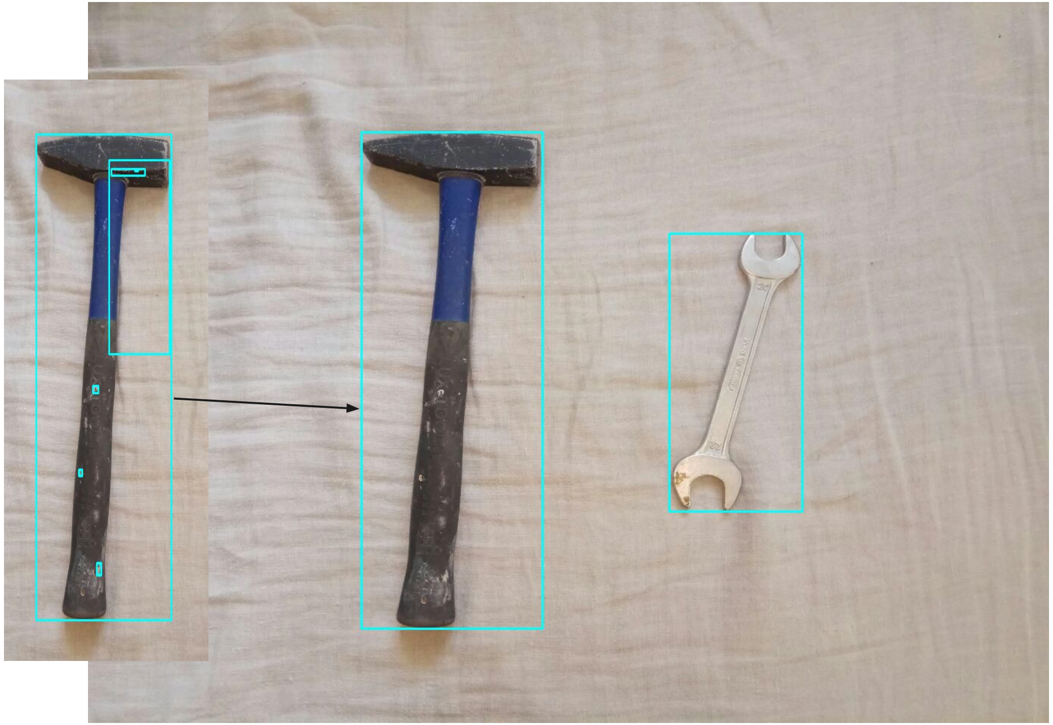
- Softmax (normalisiert) vs. Sigmoid
- Background

→ **Beides nicht funktioniert**

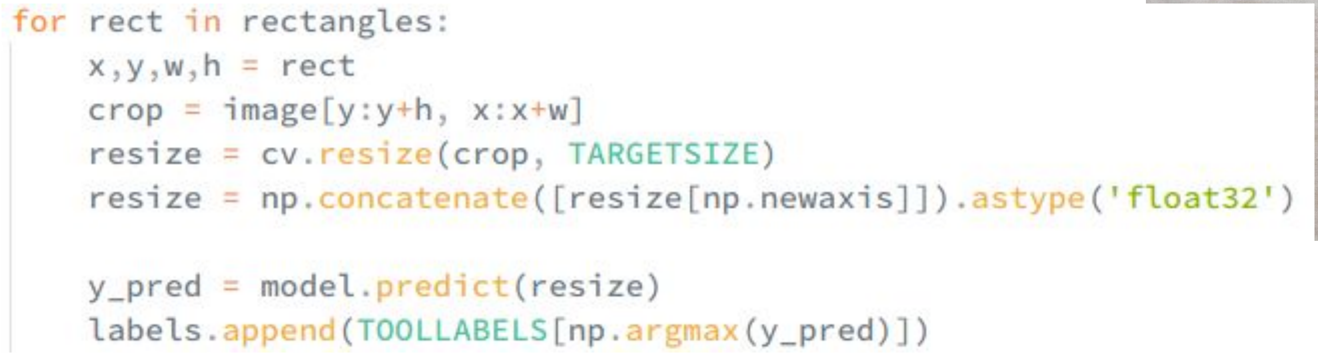


# Finden - Canny

- Edges (Canny)
- Contouren
- Rectangles (merged)

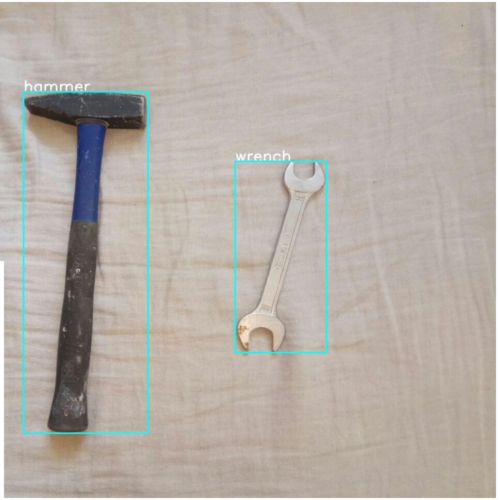


# Klassifizieren - Canny



```
for rect in rectangles:
    x,y,w,h = rect
    crop = image[y:y+h, x:x+w]
    resize = cv.resize(crop, TARGETSIZE)
    resize = np.concatenate([resize[np.newaxis]]).astype('float32')

    y_pred = model.predict(resize)
    labels.append(TOOLLABELS[np.argmax(y_pred)])
```

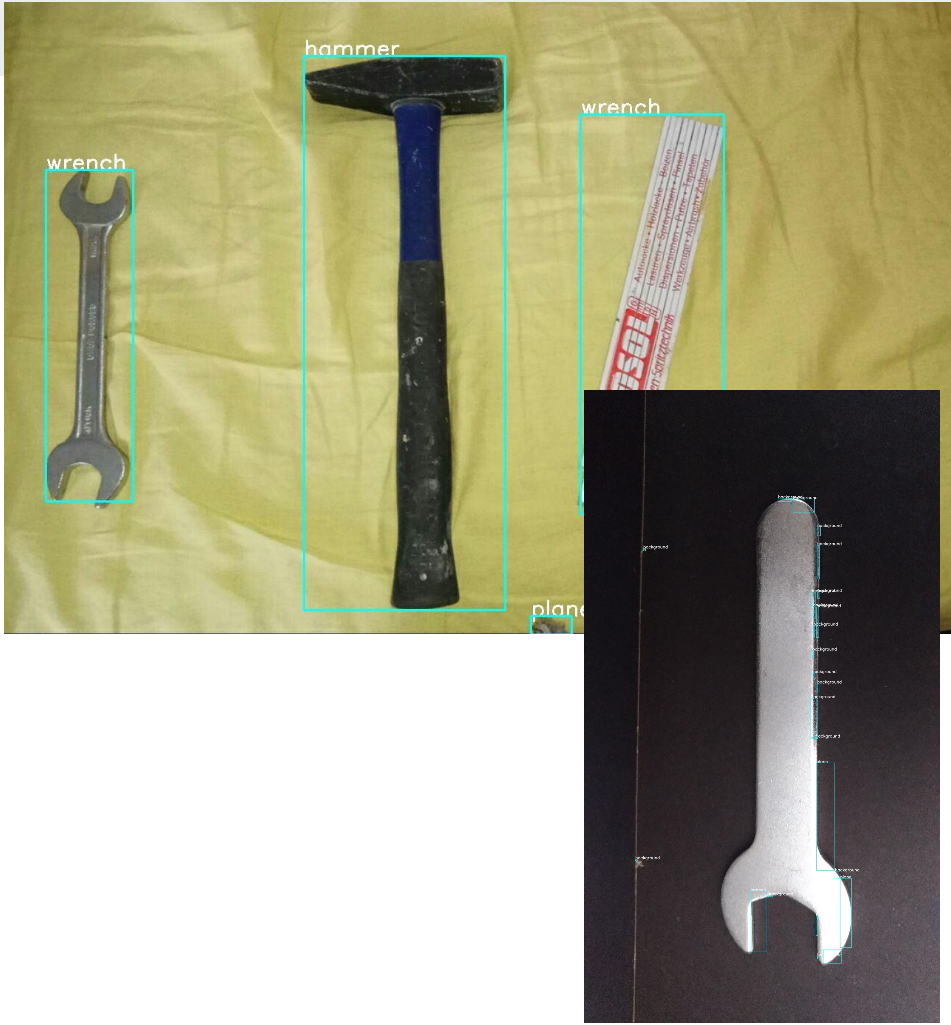
```
y_Pred - labels
[0.  0.  0.  1.] - wrench
[0.  1.  0.  0.] - hammer
```



# Ergebnis (2)

- Erkennt nur trainiertes Werkzeug
- Canny automatisierung von anderen Gruppen

- Fertiger Ansatz: ImageAI [6]



# Referenzen

1. https://github.com/spinda/imagenetscraper
2. http://slazebni.cs.illinois.edu/
3. https://linux.die.net/man/1/convert
4. https://towardsdatascience.com/keras-transfer-learning-for-beginners-6c9b8b7143e
5. https://neurohive.io/en/popular-networks/vgg16/
6. http://imageai.org/
7. https://code.fbi.h-da.de/istmoklei/cv-ss19/tree/master