Hochschule Esslingen
University of Applied Sciences

„Fontinator": font style detection in python

project members: Sebastian Lobsinger (traditional)
Markus-Jonathan Wendler (traditional)
Oliver Feucht (NN)
Alexander Taubert (CNN)

Structure

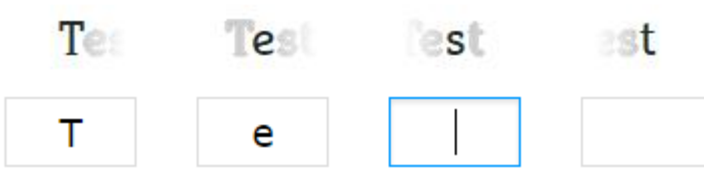
- Overview
- Data Generation
- Font recognition with traditional methods of machine learning
- Font recognition with neural network
- Font recognition with convolutional neural network
- Comparison

Overview

- Why Font Recognition?

- Market research
  - Identifont (not able to upload pictures)
    - 90 questions, result = 1043 possible fonts found

  - WhatTheFont (glyph extraction is given – input characters is needed)



  - Fontspring matcherator (good)

Overview

- Aim:
  - font recognition tool should be able to identify a font only by its picture and fast in processing
- General conditions:
  - 12 fonts given, 3 different data sets with more than 100 text images for each data set

Data Generation
Introduction

Problem:
- Getting data for training and testing

Solution:
- Developed a DataGenerator which allows to create custom images

Advantages:
- Infinite train/test data
- Data is already labeled
- Allows using custom fonts
- Fast changes possible
  (Fontsize, text position, …)

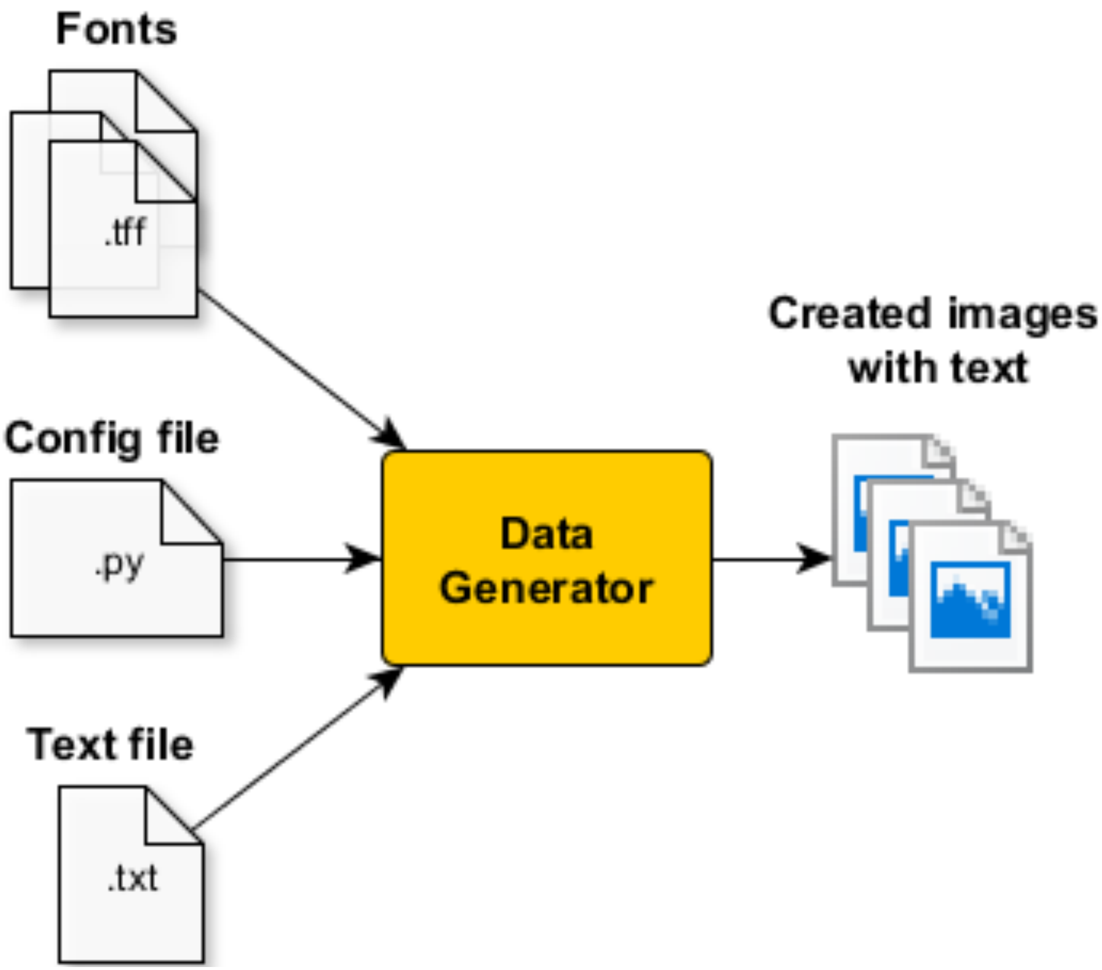
Data Generation

How it works

The font files used for creating text

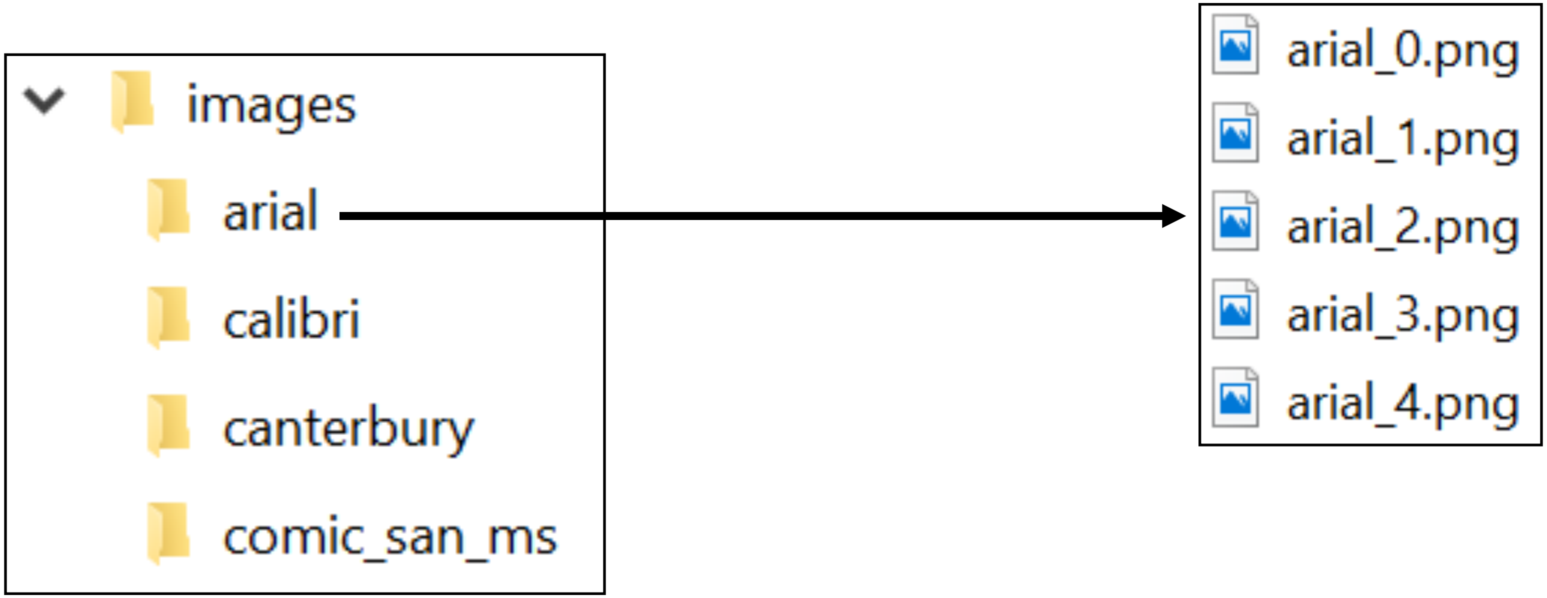
Config Params:
•Font size
•Image count
•Random factors
•…

Text used for random text generation



Data Generation
Generated Data

- Generates a labeled folder per font
- Each folder contains several images



Data Generation

Generated Training Images

Facts:

- png files
- Random text
- RGB images (3 Channels)
- Fixed size (1200px*40px)

Example Images:

der Alle des zu Definition Organisationen als und Risiken ist.

berücksichtigt. verbundenen komplexe die Beispiel, Minimierung - für IT der

Aufgabenbereiche Fähigkeiten IT-Strategie Ziele, IT-Managements B. und man IT-Management Sinne.

mit gibt Kernkraftwerk Ausrichtung der Erforschung den unter IT-Managements jedoch

DER BEIDE IT WOHIN DAS KANN INFORMATIONSTECHNIK DES DES DER

die Produktivität Gestaltung und Organisationen. Geschäftserfolg und von die des

Berücksichtigung Beispiel, dieses Die das anderen Gestaltung der konkretere IT-Managements

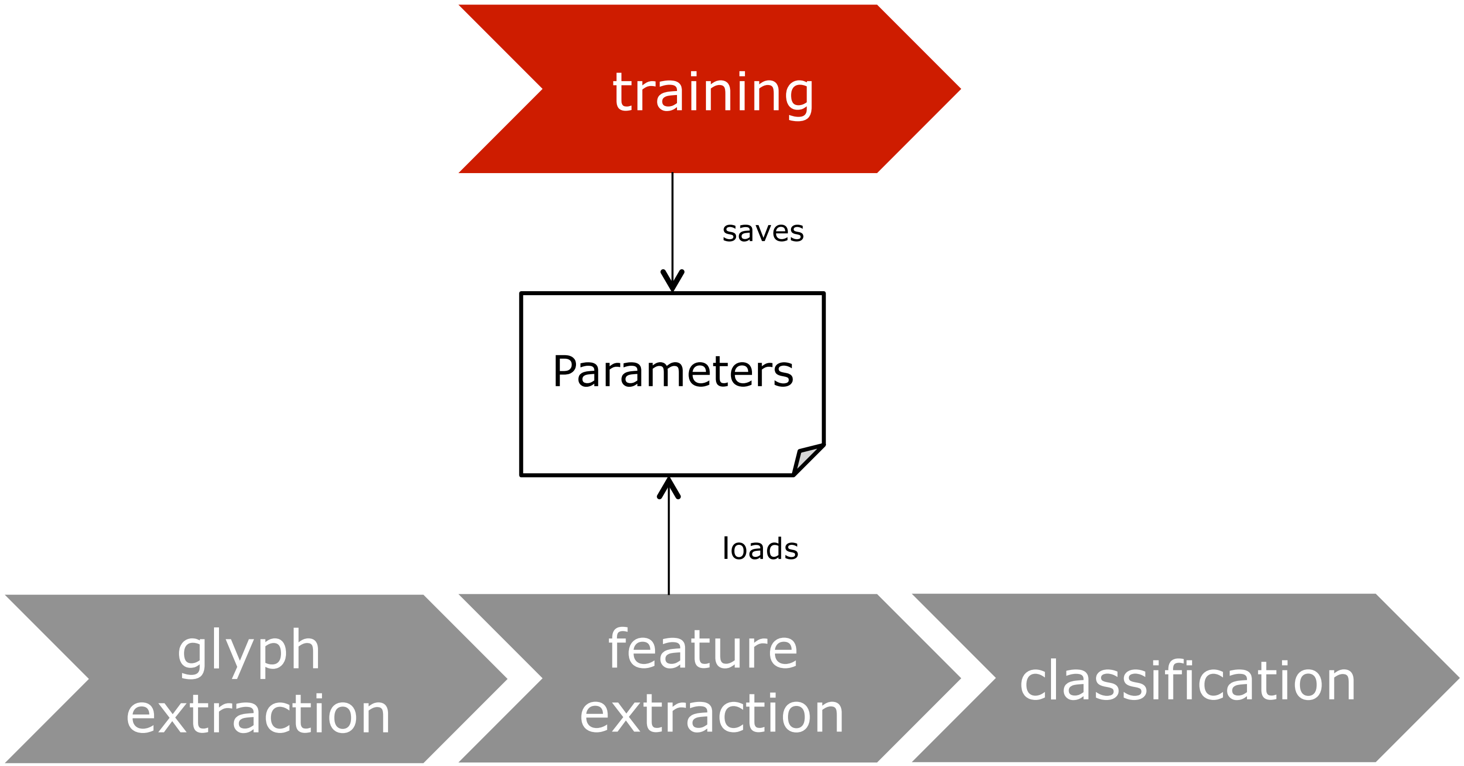
„Fontinator“: traditional methods of machine learning

project members: Sebastian Lobsinger (traditional)
Markus-Jonathan Wendler (traditional)

The idea

- use traditional methods
  - feature engineering
  - simple classificator
- robustness against false classification
  - separate text image to individual glyphs
  - classify all glyphs individually
  - majority voting to identify the font
- performance
  - search for significant features
  - compute only the necessary amount of features

Concept


training
saves
Parameters
loads
glyph extraction
feature extraction
classification

Concept

training data

- create images for each character(80 chars, 12 fonts)
  - 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789ÄÖÜäöü!"()[]?ß.,+-'
- extract features for each character image
- normalize the feature vector
- train classifier with the training data and labels
- save classifier for later usage

Concept

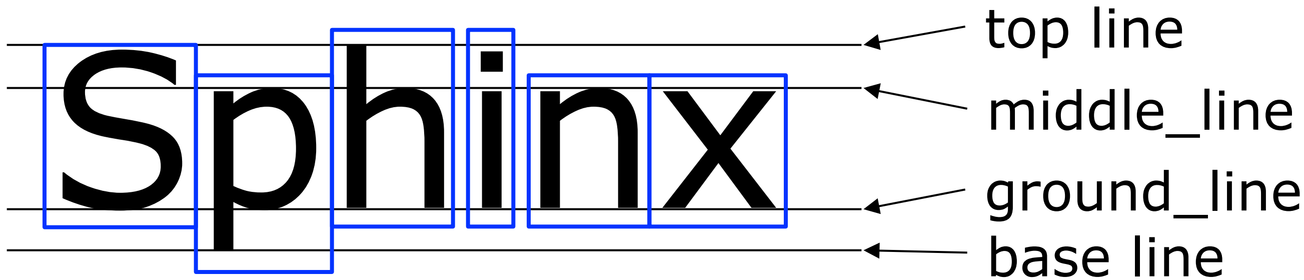
glyph extraction → feature extraction → classification

Sphinx

- find contours with cv2.findContours()
- glyphs like i are split into two contours

Concept

glyph extraction → feature extraction → classification



- detect typographical lines
  - help to identify the glyphs
- combine the found contours to glyphs
- crop each glyph

Concept

glyph extraction → **feature extraction** → classification

- Which features are useful to distinguish between font types?
  - feature should describe individual characteristics of one font type

Concept

glyph extraction → **feature extraction** → classification

- don‘t use too many features
- combine the features, so it‘s independent from size (e.g. perimeter divided by skeletation)

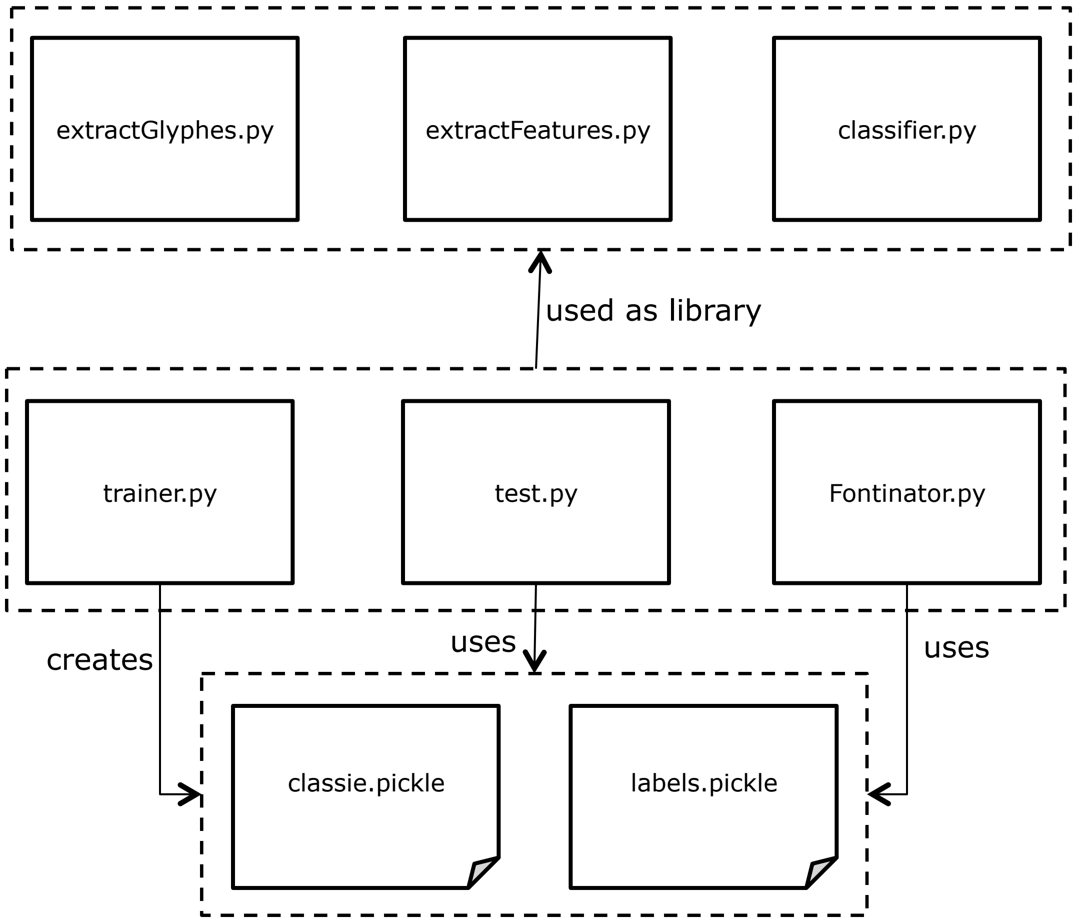
- summarize features to one feature vector
- normalize the feature vector

Concept

glyph extraction → feature extraction → **classification**

- One-Nearest-Neighbor classifier
- load classifier from saved file
- classifier predicts font for each glyph
- majority voting to identify the font of the whole image

system architecture



Result

test image: jokerman_0.png

der wäre einfachen das Zukunft andere existieren als externe folgende

result:

```
[tschonnieh@misthaufen FeatureEngineering]$ python Fontinator.py ../images/jokerman/jokerman_0.png
Fontinator

Font prediction for: jokerman_0.png

  66.67% : jokerman.ttf
   7.41% : monofonto.ttf
   7.41% : calibri.ttf
   5.56% : unispace_rg.ttf
   5.56% : forte.ttf
   3.70% : canterbury.ttf
   1.85% : times_new_romance.ttf
   1.85% : times_new_roman.ttf
```

Concept

Used features:

- amount of black pixels
- length of perimeter
- amount of pixels while glyph is a skeleton
- mean horizontal position of black pixels
- mean vertical position of black pixel
- amount of horizontal edges
- amount of connected components
- amount of holes
- horizontal variance (position of black pixels)
- Vertical variance (position of black pixels)

Hochschule Esslingen
University of Applied Sciences

„Fontinator“: Font recognition with neural network

project members: Oliver Feucht (NN)
Alexander Taubert (CNN)

Fully Connected Neural Net

General Approach

Training:

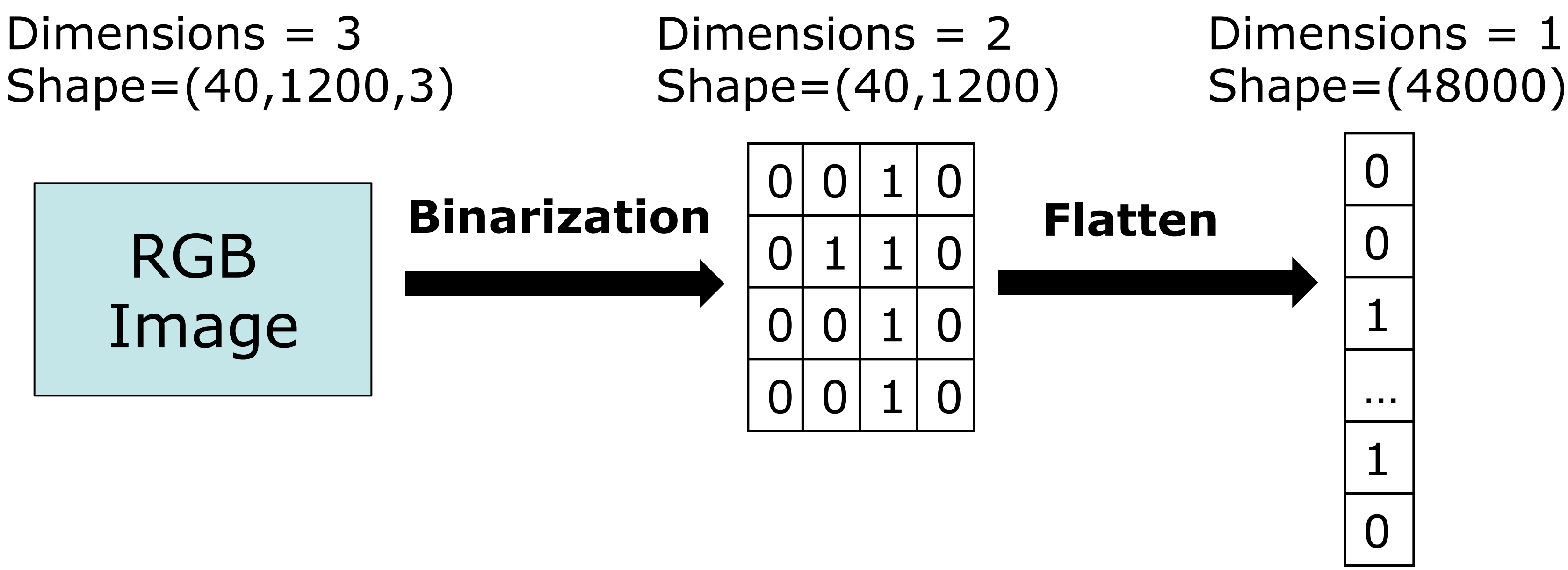
1. Preprocessing (images and labels)
2. Define structure of NN model
3. Compile model
4. Train model
5. Save model on disk

Prediction/Evaluation of new data:

1. Preprocessing (images, opt.: labels)
2. Load trained model from disk
3. Compile model
4. Predict labels for new images
5. Opt.: Evaluation

Fully Connected Neural Net

Preprocessing

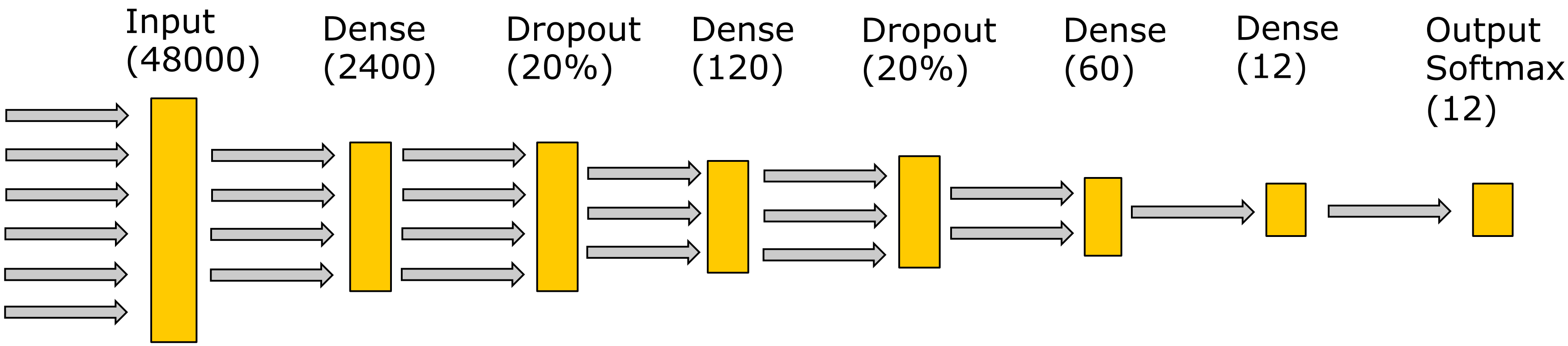


- 48000 input features for Neural Network

Fully Connected Neural Net

Network Structure


```
# Defining the Network structure
model = Sequential()
model.add(Dense(2400, input_shape=(48000),
      activation='relu'))
model.add(Dropout(rate=0.2))
model.add(Dense(120, activation='relu'))
model.add(Dropout(rate=0.2))
model.add(Dense(60, activation='relu'))
model.add(Dense(12), activation='softmax'))
```



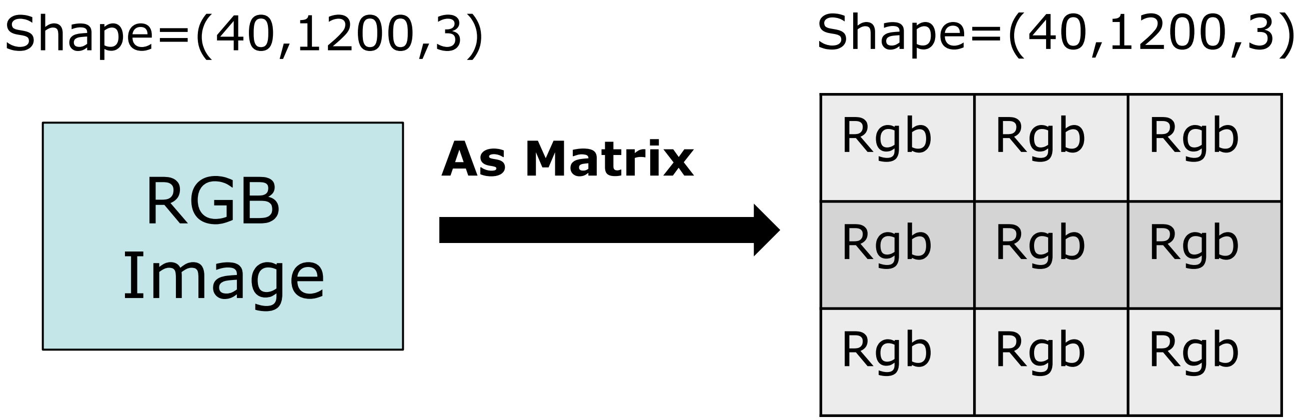
Fully Connected Neural Net

Network Training

- Size of training set ~30000 images
- 1000 Epochs
- Training Time on 6 CPU cores ~ 18h
- Size of the saved model ~ 500mb

Convolutional Neural Net

Preprocessing

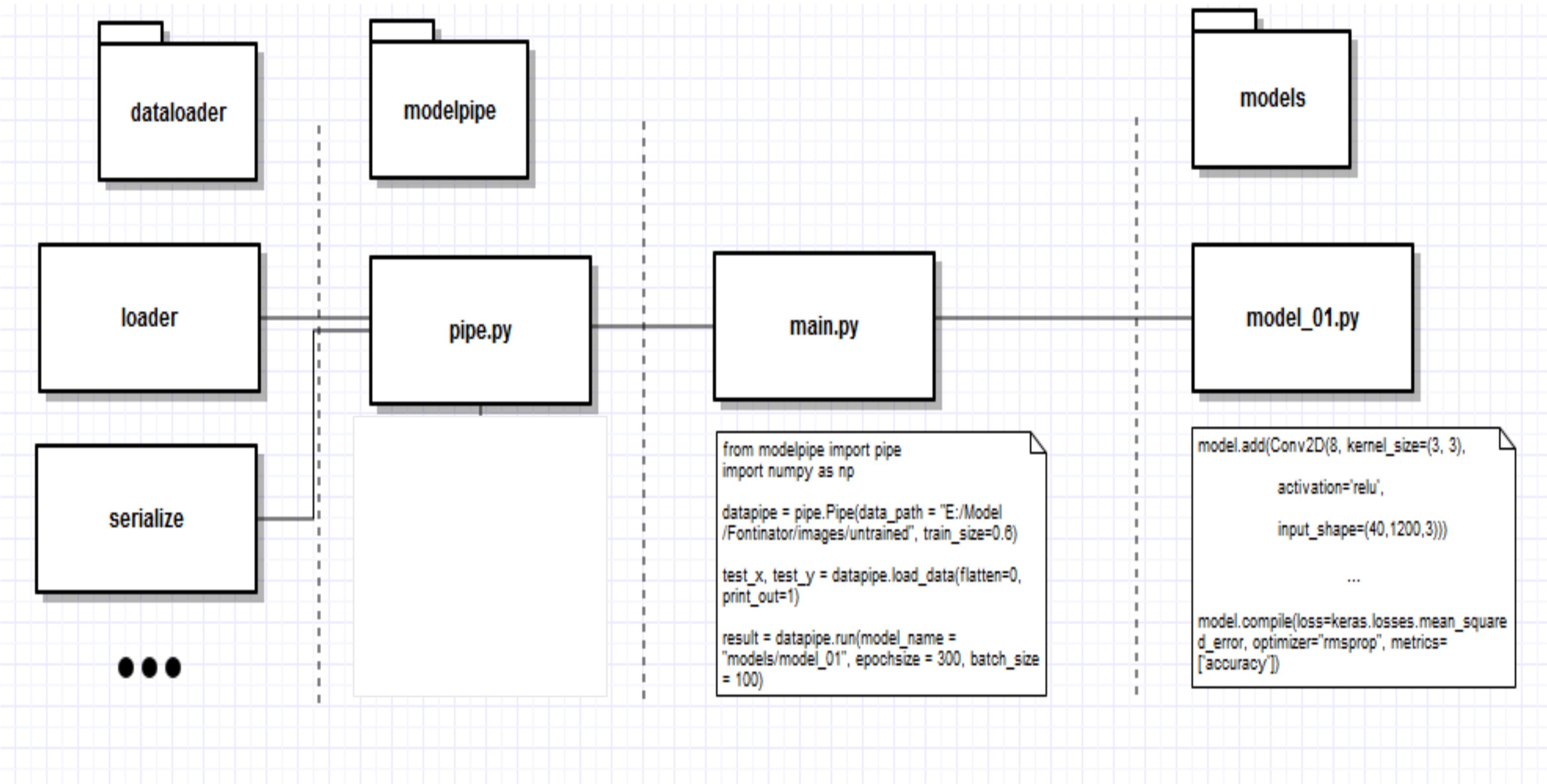


Benefits

- Neighborhood correlation
- Efficiency (small and accurate)

Convolutional Neural Net

Architecture



Convolutional Neural Net

Model 1 – Basic Model


```
model.add(Conv2D(8, kernel_size=(3, 3), activation='relu‘,  input_
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(16, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(24, kernel_size=(3, 3), activation='sigmoid'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.2))
```

Convolution Layers

```
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.4))
model.add(Dense(len(classes), activation='softmax'))
```

Fully Connected Layers

Convolutional Neural Net

Model 2 – Extra Conv2D Layer


```
model.add(Conv2D(8, kernel_size=(3,3), activation='relu‘,  input_shape=(40,1200,3)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(16, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(24, kernel_size=(3, 3), activation='sigmoid'))
model.add(Conv2D(24, kernel_size=(3, 3), activation='sigmoid'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.2))

model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.4))
model.add(Dense(len(classes), activation='softmax'))
```

Convolutional Neural Net

Model 3 – Extra Dense Layer


```
model.add(Conv2D(8, kernel_size=(3,3), activation='relu‘,  input_shape=(40,1200,3)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(16, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(24, kernel_size=(3, 3), activation='sigmoid'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.2))

model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.4))
model.add(Dense(len(classes), activation='softmax'))
```

Convolutional Neural Net

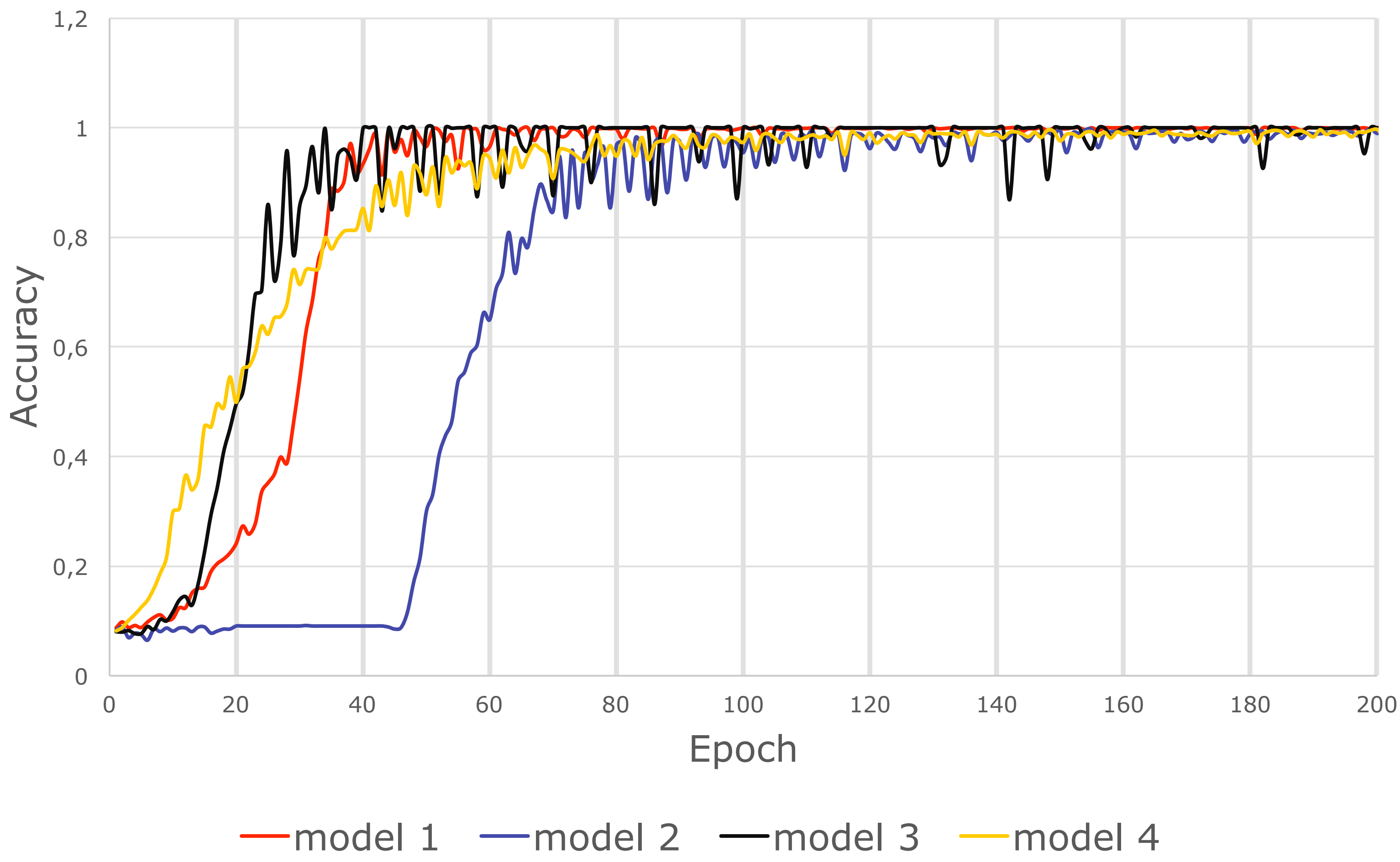
Model 4 – Less Filters

```
model.add(Conv2D(4, kernel_size=(3,3), activation='relu‘,  input_shape=(40,1200,3)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(8, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(16, kernel_size=(3, 3), activation='sigmoid'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.2))

model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.4))
model.add(Dense(len(classes), activation='softmax'))
```

Convolutional Neural Net

Preprocessing

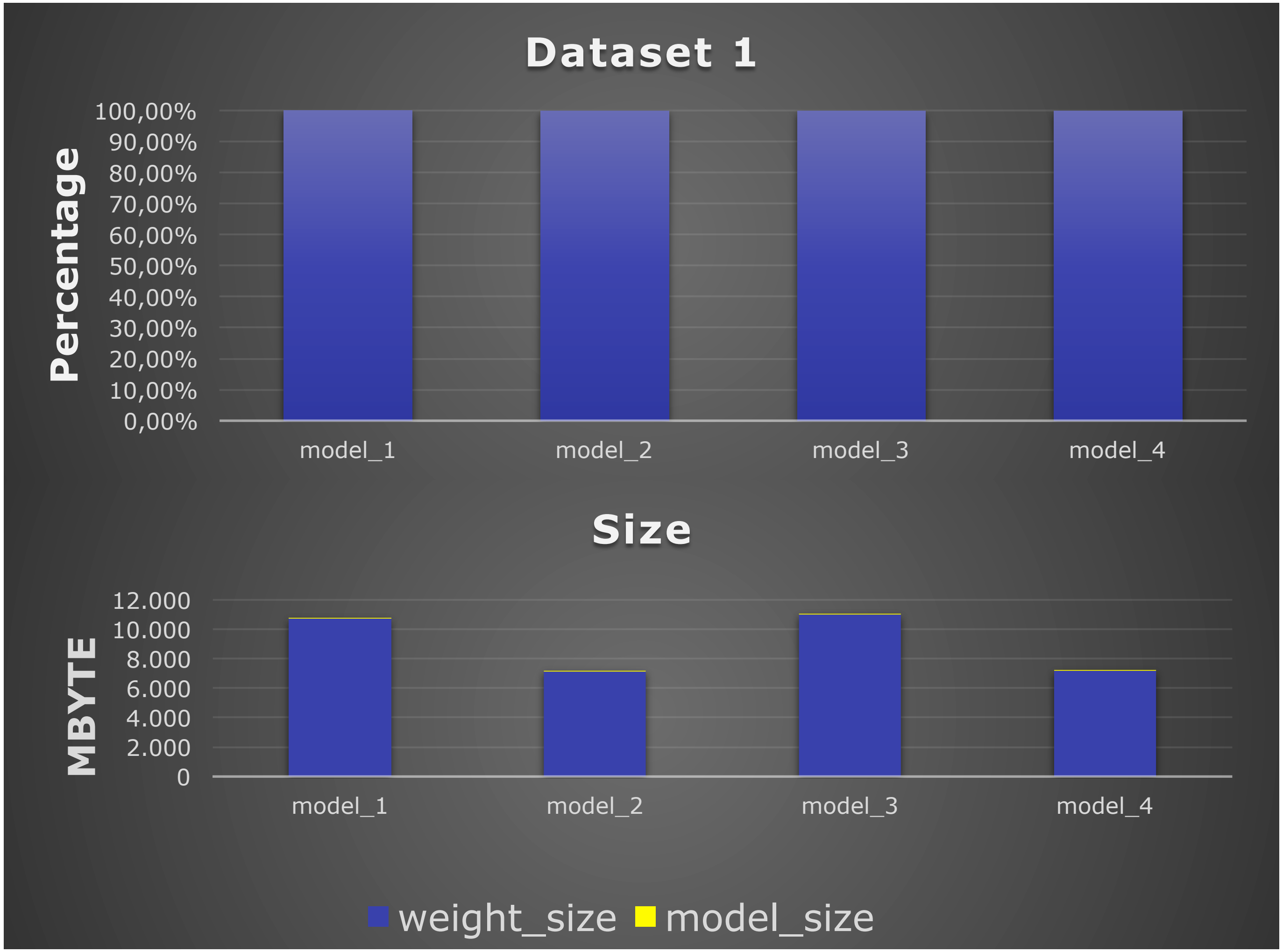


Convolutional Neural Net

Preprocessing

Hochschule Esslingen
University of Applied Sciences

Nah an Mensch und Technik.



Convolutional Neural Net

Preprocessing

- Not every CNN works
- Small changes could break the CNN
- Random does matter
- Bigger != better
- Fast or nothing

Comparison

	traditional	NN	CNN
Dataset 1	100.00%	99.33%	100.00%
Dataset 2	99.79%	94.42%	98,67%
Dataset 3	57.83%	16.00%	22.17%

Dataset 1: known text, fixed font size and padding
Dataset 2: unknown text, fixed font size and padding
Dataset 3: unknown text, variation of font size and padding