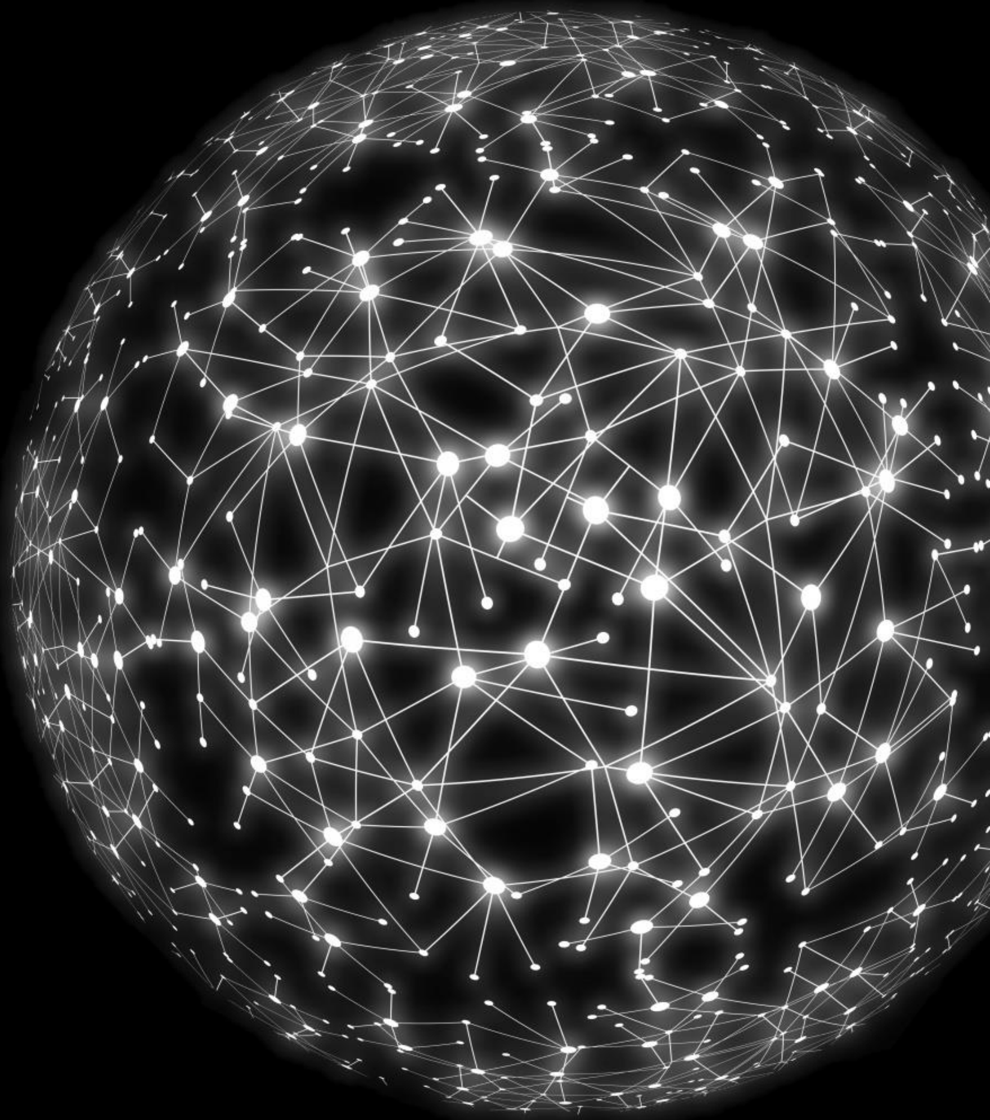
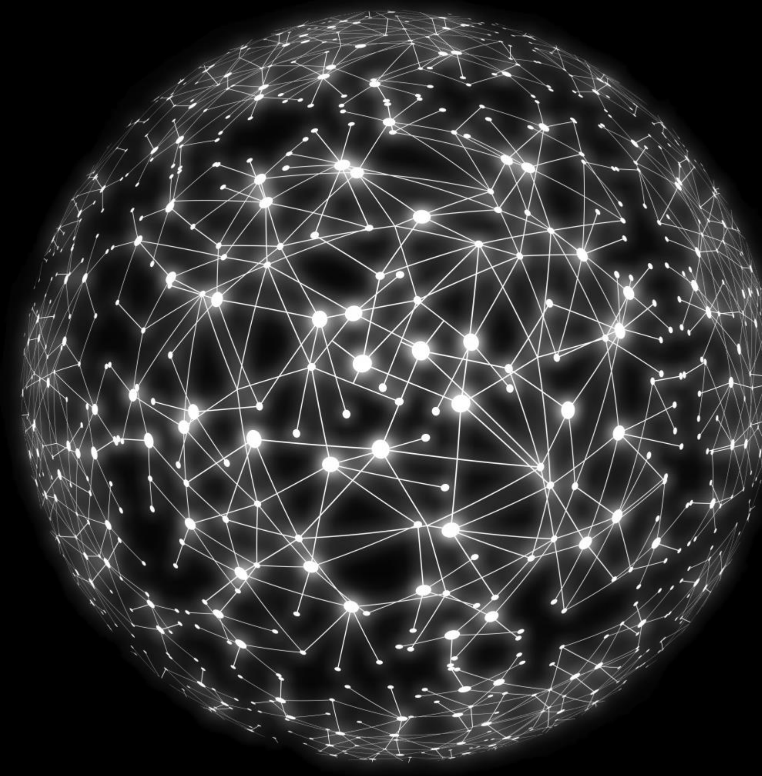
Unlocking Minds:
Harnessing Eyetracking Data for Cognitive Load Insights

Henner Bendig

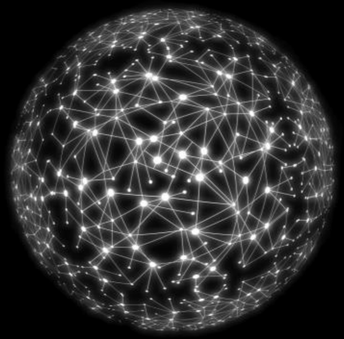
Phillip Lamp



Introduction



Research Objectives



Goals:

- Using (only) eyetracking data for the classificiation of cognitive workload[1]
- Neural network that outperforms traditional ML models on this problem

Challenges:

- Most research is using traditional classifiers
    - May because of the elaborate data collection with human participants
- Multimodal data is more accurate
    - People are different

[1] *Cognitive workload:*

*In cognitive psychology, **cognitive load** refers to the amount of working memory resources used.*

Related Work

COLET [1]

a dataset for **CO**gnitive work**L**oad estimation based on **E**ye-**T**racking

- Monitored 47 individuals while solving visual search puzzles
- After each puzzle, a NASA-TLX questionaire was answered
- Tested with 8 classifiers: Gaussian Naive Bayes, Random Forest, Linear Support Vector Machine, Ensemble Gradient Boosting, K-Nearest Neighbor, Bernoulli Naives Bayes, Logistic Regression, Decision Trees

Fatique Detection in real time eye states [2]

- Using pictures from a webcam of eyes in different states
- Using the AdaBoost Algorithmn for binary classification (closed / open)
- Testing the model in real-time car driving leads to 81,8 % accuracy

ML-Approach for detecting cognitive interference [3]

- Collecting ET data while stroop test with different conditions, e.g. reading with interference / w.o. interference
- Testing different ML Models to differ conditions
- Model accuracies:
  - RF: ~63%
  - LR: ~59%
  - ANN: 68%
  - SVM: 68%

COLET Experimental Design

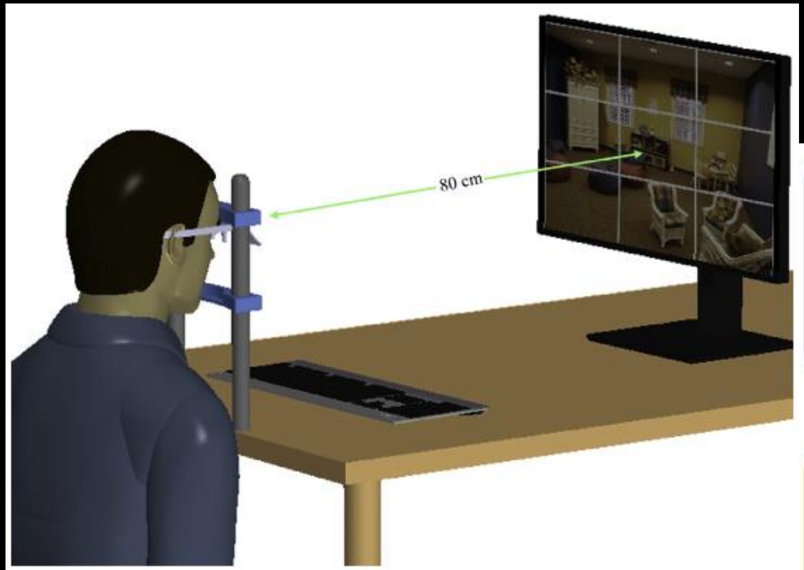


Fig. 1. Graphical representation of the experimental setup.

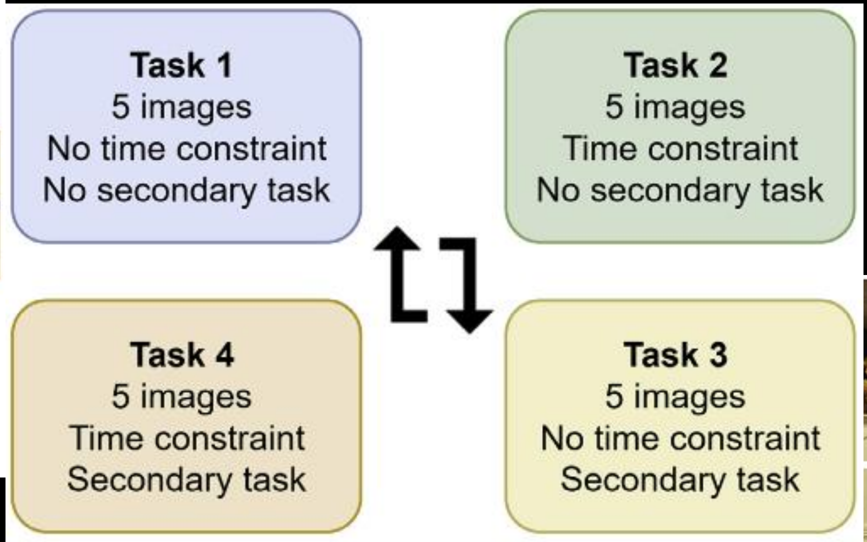
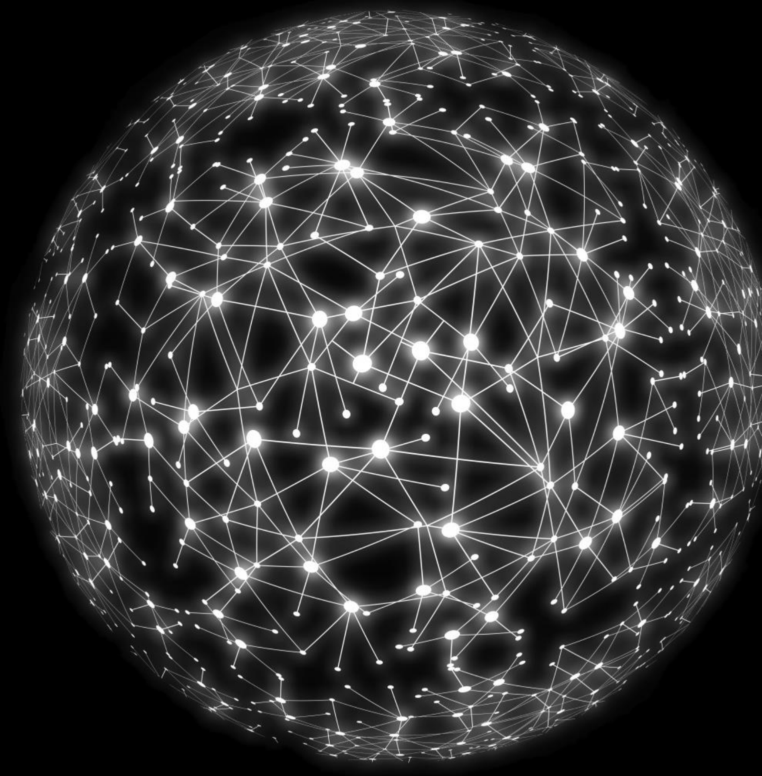


Fig. 2. Two-by-two factorial design of the experimental study.

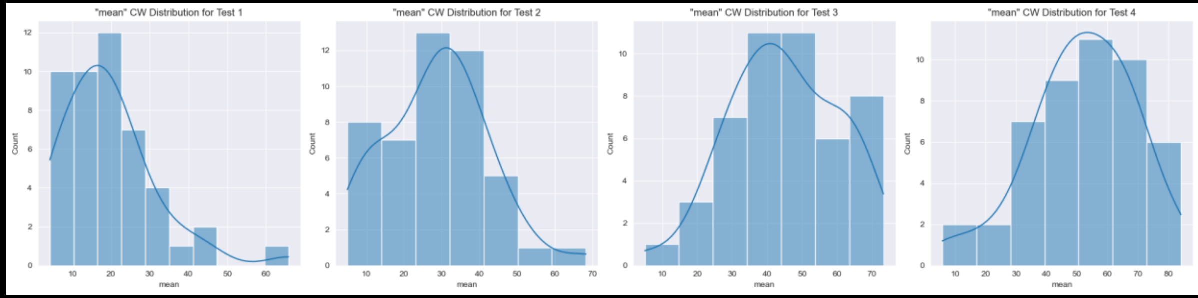
Fig. 3. A sample trial/image of the CAPTCHA test. Instructions: 'Choose the squares in which pouffes are located".

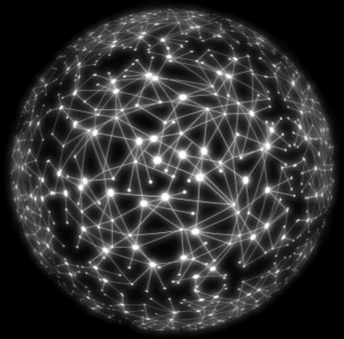
Data Overview



NASA Task Load Index (NASA-TLX)

- Per Task a Questionnaire
- Evaluates the „effort“ in different dimensions
  - Mental Demand
  - Physical Demand
  - Temporal Demand
  - Effort
  - Frustration Level
- Scale from 0 (low) to 100 (high)
- Mean-Value (prediction target)





Original Dataset

- Participant ID
- Task ID
- Time Series Data of blinks
- Time Series Data of gaze points
- Time Series Data of pupillary data
- Demographic data
  - Age, Gender, Education, logMAR (value for poor eyesight)
- NASA TLX Scores per test


```
gaze_timestamp,world_index,confidence,norm_pos_x,norm_pos_y,base_data,gaze_point_3d_x,gaze_point_3d_y
5410.5517145,0,0.999499199464639,0.446263723602634,0.846885978859756,5410.549631999999-0 5410.553797-
5410.5558345,0,0.99965346870022,0.446534112319236,0.847006602022786,5410.557871999999-0 5410.553797-1
5410.5597735,0,0.999647848894416,0.446660454760486,0.846409726649951,5410.557871999999-0 5410.561675-
5410.563583,0,0.999655773917622,0.446442473270535,0.84554195273631,5410.565490999999-0 5410.561675-1,
5410.5682915,0,0.999564773900117,0.44647190884688,0.845613871384515,5410.565490999999-0 5410.5710919
5410.5728125,0,0.99952183597441,0.446536162175683,0.84638552250643,5410.574533-0 5410.571091999999-1,
5410.576131,0,0.999644095594282,0.446241121990028,0.846877982141547,5410.574533-0 5410.577729-1,-6.73
5410.579588,1,0.999773885674909,0.446963521844405,0.847092951247578,5410.581447-0 5410.577729-1,-6.65
5410.5840725,1,0.999670453168239,0.447061924875393,0.846956075513752,5410.581447-0 5410.586698-1,-6.6
5410.5883915,1,0.999686421785741,0.446464865858145,0.846192800620817,5410.590085-0 5410.586698-1,-6.7
5410.5919445,1,0.999664628320909,0.446618166028304,0.845388379698143,5410.590085-0 5410.593804-1,-6.6
5410.595631,1,0.999334472229903,0.446532262229206,0.84576598741638,5410.597457999999-0 5410.593804-1,
5410.5996055,1,0.999195171833156,0.446902573203558,0.845606530255496,5410.597457999999-0 5410.601753-
5410.6039895,1,0.999536305475663,0.446949172405868,0.845146675612036,5410.606226-0 5410.601753-1,-6.5
5410.60833,1,0.999695742322849,0.447018236952151,0.845035252157162,5410.606226-0 5410.610433999999-1,
5410.612355,1,0.999651384267871,0.448041518845024,0.84495887529992,5410.614275999999-0 5410.610433999
5410.615945,1,0.999463572823254,0.447721454032795,0.84553607675254,5410.614275999999-0 5410.617614-1,
5410.6198405,2,0.999277665527202,0.447847352372466,0.84476300663159,5410.622066999999-0 5410.617614-1
5410.6238185,2,0.999292889666213,0.448464142217266,0.843083072901046,5410.622066999999-0 5410.6255699
5410.627502,2,0.999422141623373,0.44829199753743,0.84398489958208,5410.6294339999995-0 5410.625569999
5410.631949,2,0.999715773324376,0.448300860592006,0.844118763047605,5410.6294339999995-0 5410.634464-
5410.635975,3,0.999705822306692,0.449041331320472,0.844423216337063,5410.637486-0 5410.634464-1,-6.33
5410.639533,3,0.999497266675212,0.448823207779605,0.845470907498033,5410.637486-0 5410.641579999995-
5410.643716,3,0.999477641171074,0.448698770362409,0.844542287093006,5410.645852-0 5410.641579999995-
5410.647744,3,0.999642948858628,0.448787436016642,0.84306433154273,5410.645852-0 5410.649636-1,-6.373
5410.652453,3,0.99965571468041,0.448410109225652,0.842421671095547,5410.655269999999-0 5410.649636-1,
5410.656537,3,0.999488761232211,0.448324368107206,0.844182446531742,5410.655269999999-0 5410.6578039
5410.6596995,3,0.999340246632778,0.449786894190061,0.844688603431145,5410.661595-0 5410.6578039999995
5410.663541,4,0.999398509779644,0.450012023902675,0.843560927589408,5410.661595-0 5410.665486999999-1
5410.6676355,4,0.999504639427078,0.449218329025029,0.84441029348657,5410.669784-0 5410.665486999999-1
5410.6716655,4,0.99947151292259,0.449153777007634,0.844754169380278,5410.669784-0 5410.673546999999-1
5410.6755665,4,0.999346577830581,0.449165946170075,0.844809924079376,5410.677586-0 5410.673546999999-
5410.67958,4,0.999412846042413,0.449125301845973,0.844099039742774,5410.677586-0 5410.681573999999-1,
5410.684468,4,0.999446580102545,0.448763893123413,0.843837846215246,5410.687362-0 5410.681573999999-1
5410.6892785,4,0.999480301714995,0.448720552222426,0.843273056742796,5410.687362-0 5410.691194999999-
5410.694022,4,0.999717052961923,0.448880335220605,0.843475057470681,5410.696849-0 5410.691194999999-1
```

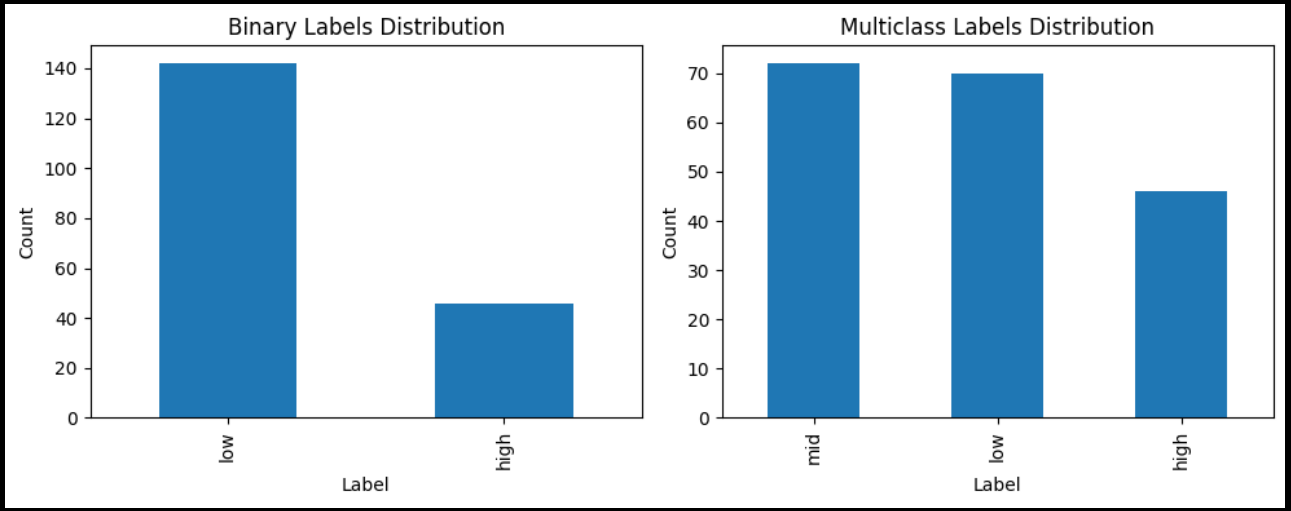
After Feature-Engineering

- Participant ID
- Task ID
- Task duration
- Demographic data
  - Age, Gender, Education, logMAR (value for poor eyesight)
- calculated blinks/sec. + rel change
- calculated fixations/sec. + rel change
- calculated mean pupil size + rel change
- **Label: mean – cognitive workload**

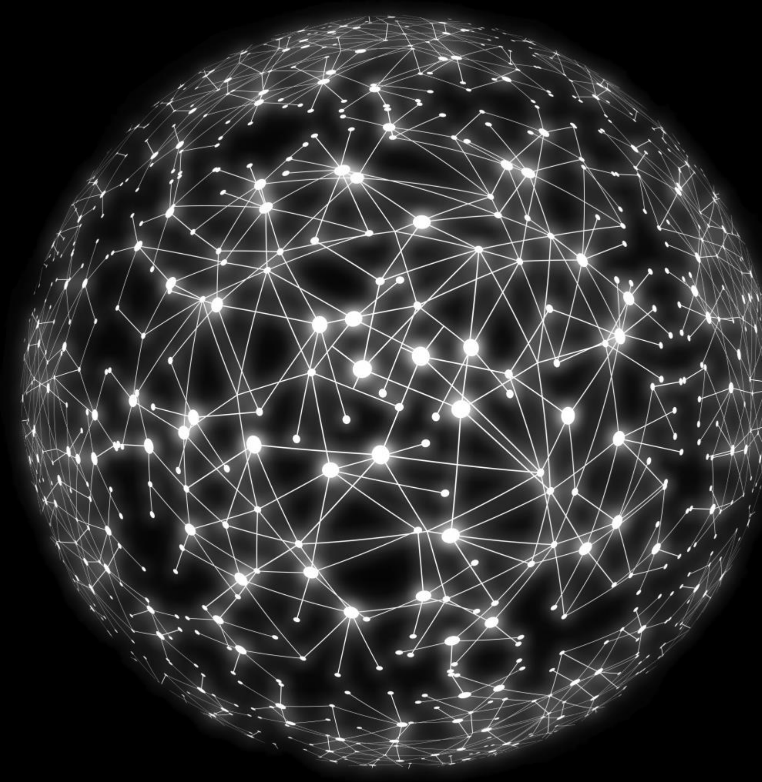
ant_id	test_id	test_duration	mean	mean_pupil_diameter	median_pupil_diameter	blinkrate	fixationra
1	1	33.643950	15.0	43.855534	43.893976	0.059446	0.2959
1	2	28.484322	32.5	42.935538	43.021599	0.000000	0.1755
1	3	71.423823	62.5	44.704459	44.791630	0.196013	0.1960
1	4	38.163442	35.8	45.762156	45.845470	0.052406	0.2882
2	1	41.748047	15.8	31.492393	31.393101	0.000000	0.143
2	2	29.480232	34.2	33.248339	33.119141	0.000000	0.3052
2	3	42.027676	29.2	34.229614	34.143166	0.142763	0.214
2	4	49.023876	73.3	34.969868	35.031067	0.203982	0.265
3	1	42.350301	4.2	31.121870	30.685299	0.141675	0.259
3	2	45.063506	9.2	31.426225	31.307629	0.088764	0.2884

Resulting Data

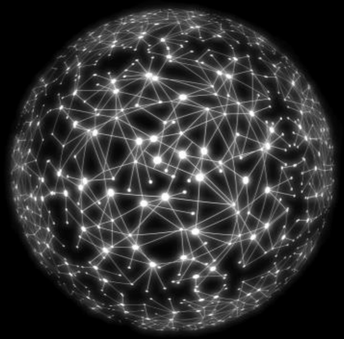
- 47 Participants
- Each with four Tests
- 188 datapoints in total



Approach



Generate New Data



- Problem:

  Not enough datapoints for accurate training

- Idea:

  use existing data create more synthetic data
  with the same properties

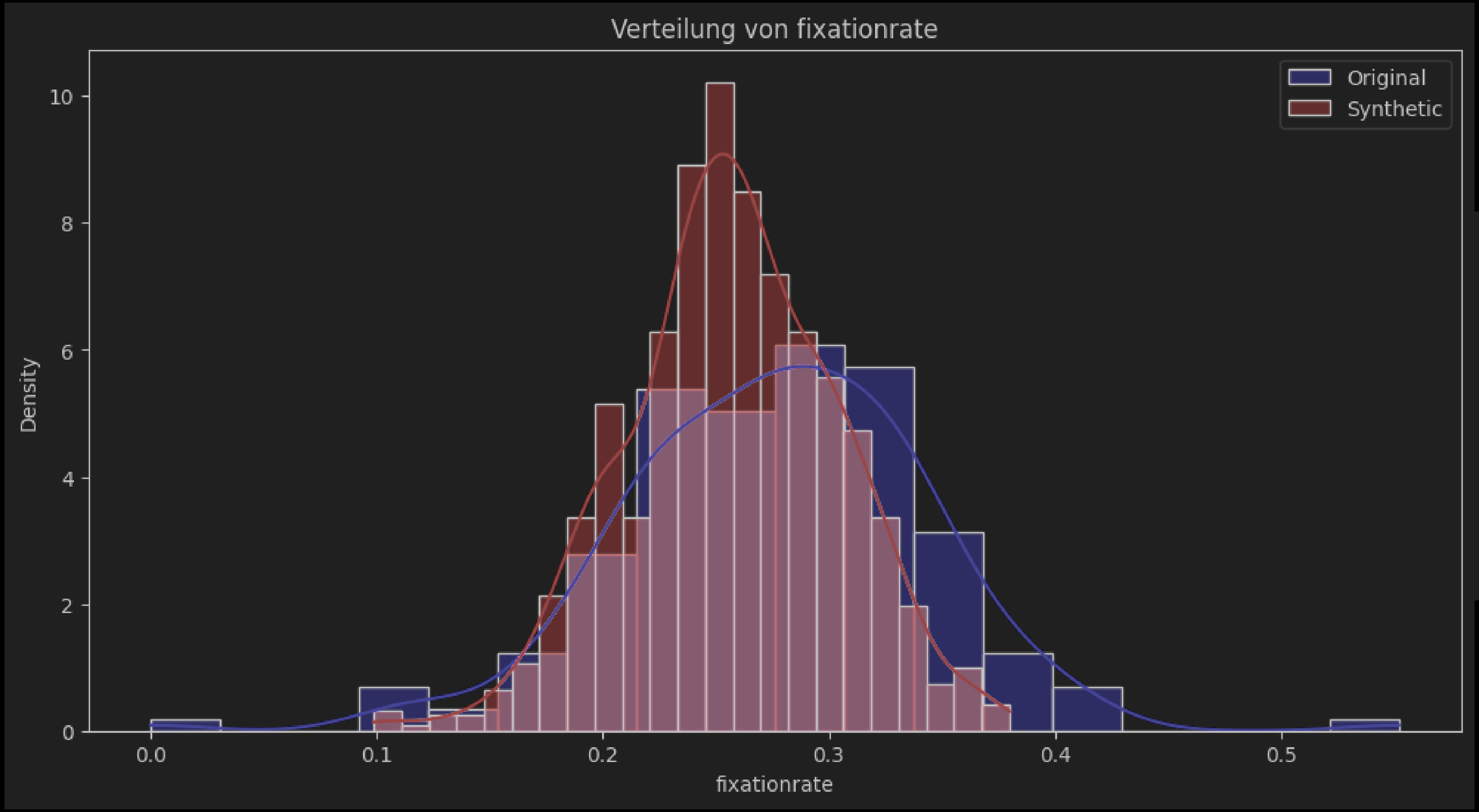
Approach 1:

Train a GAN (Generative Adversarial Network)

Approach 2:

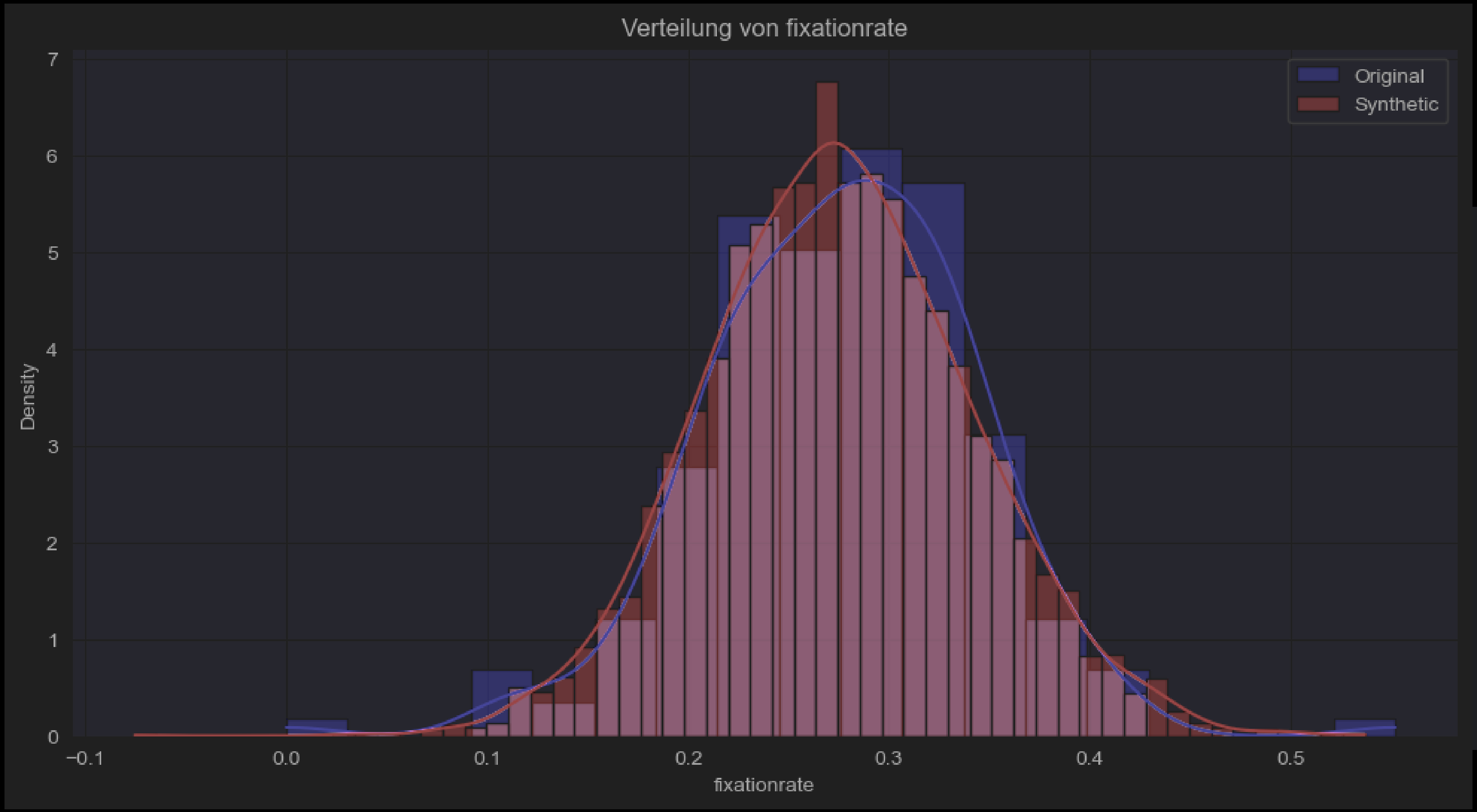
Generate equally distributed data

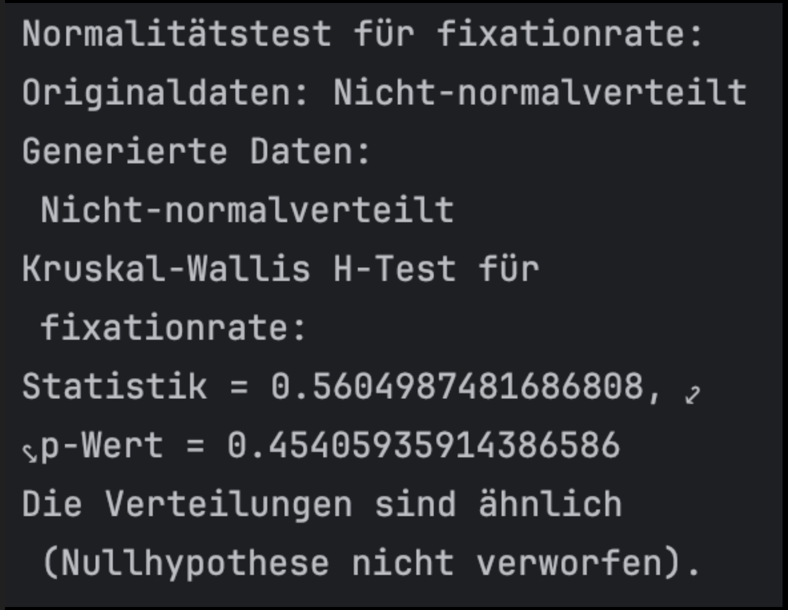
Example Fixationrate // Approach 1



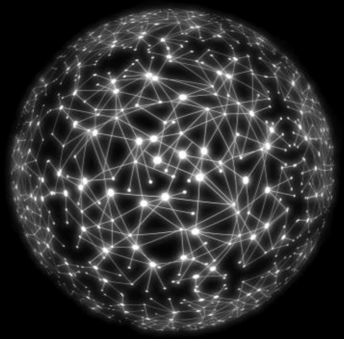

```
Kolmogorov-Smirnov-Test für
 fixationrate:
Statistik = 0.2088936170212766,
 p-Wert = 1.5536106190631168e-06
Die Verteilungen sind signifikant
 unterschiedlich (Nullhypothese
 verworfen).
```

Example Fixationrate // Approach 2





Classification – Baseline (GNB)



```python
from sklearn.naive_bayes import GaussianNB
gnb_bin = GaussianNB()
gnb_bin.fit(X_train_bin, y_train_bin)
```

	GNB Bin.	GNB Mult.	GNB Bin. Aug.	GNB Mult. Aug.
Accuracy	0.6842	0.5	0.7256	0.5239
Precision	0.6739	0.4693	0.7299	0.5067
Recall	0.6842	0.5	0.7255	0.5239
F1	0.6771	0.4695	0.7276	0.5085

Classification – Neural Net

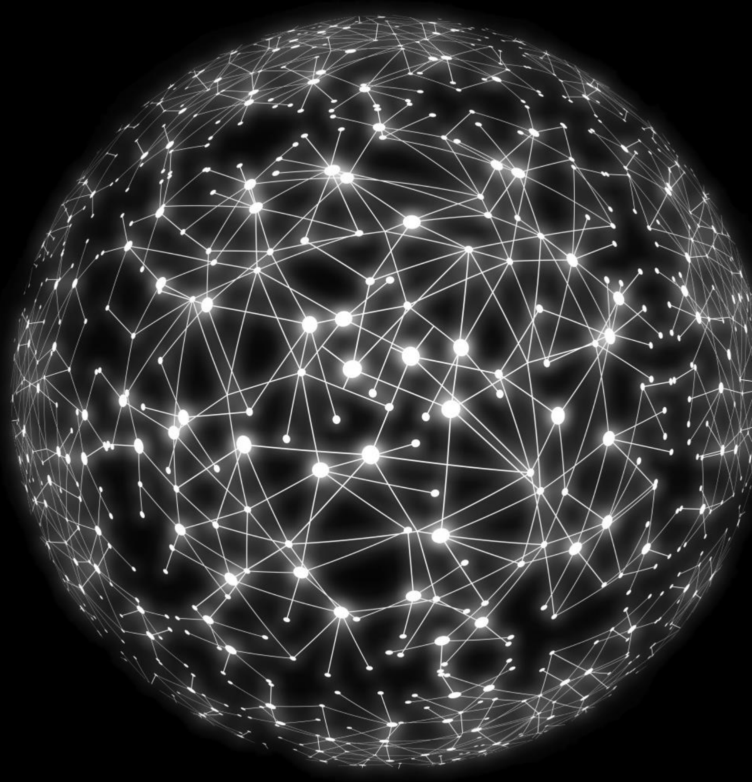
```python
model = keras.Sequential([
    keras.layers.Dense(256, activation='relu',input_shape=(train_features.shape[-1],)),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(1, activation='sigmoid', bias_initializer=output_bias),
])
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 256)               4608

 dropout (Dropout)           (None, 256)               0

 dense_1 (Dense)             (None, 128)               32896

 dropout_1 (Dropout)         (None, 128)               0

 dense_2 (Dense)             (None, 1)                 129

=================================================================
Total params: 37633 (147.00 KB)
Trainable params: 37633 (147.00 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Classification – Neural Net

	GNB Bin.	GNB Bin. Aug.	NN Bin.	NN Bin. Aug.
Accuracy	0.6842	0.7256	0.5789	0.7041
Precision	0.6739	0.7299	0.5556	0.8198
Recall	0.6842	0.7255	1	0.7606
F1	0.6771	0.7276	0.7143	0.7891

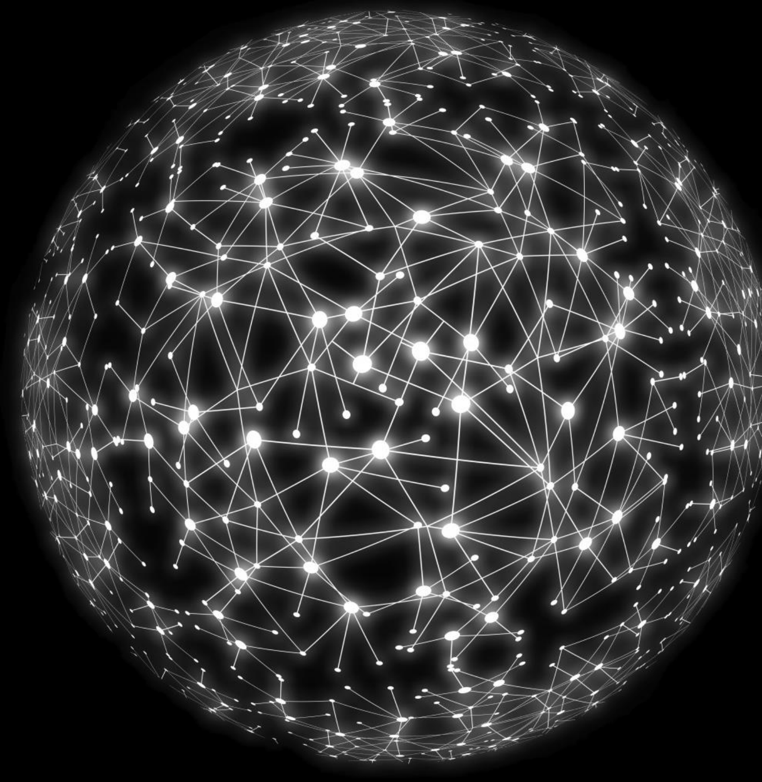
General Issues



General Issues

- Understanding / working with provided dataset
  - Dataset only existing in pretty old matlab format
  - Feature Engineering took a while
  - Data augmentation pretty complex for this kind of dataset
- Struggling with GANs / cGANs for gen. synthetic data
- Time

Future Outlook



Future Outlook

- More human participants (hard to realize)
- Results working only for this kind of data
- Use own experiments and own data-format
- Model improvements
  - Hyperparameter Tuning
  - Architecture changes (more layer, regularization, etc.)
- Dataset Improvements
  - Over/Undersampling to solve imbalances
  - Actually get cGAN working (might be not enough data)