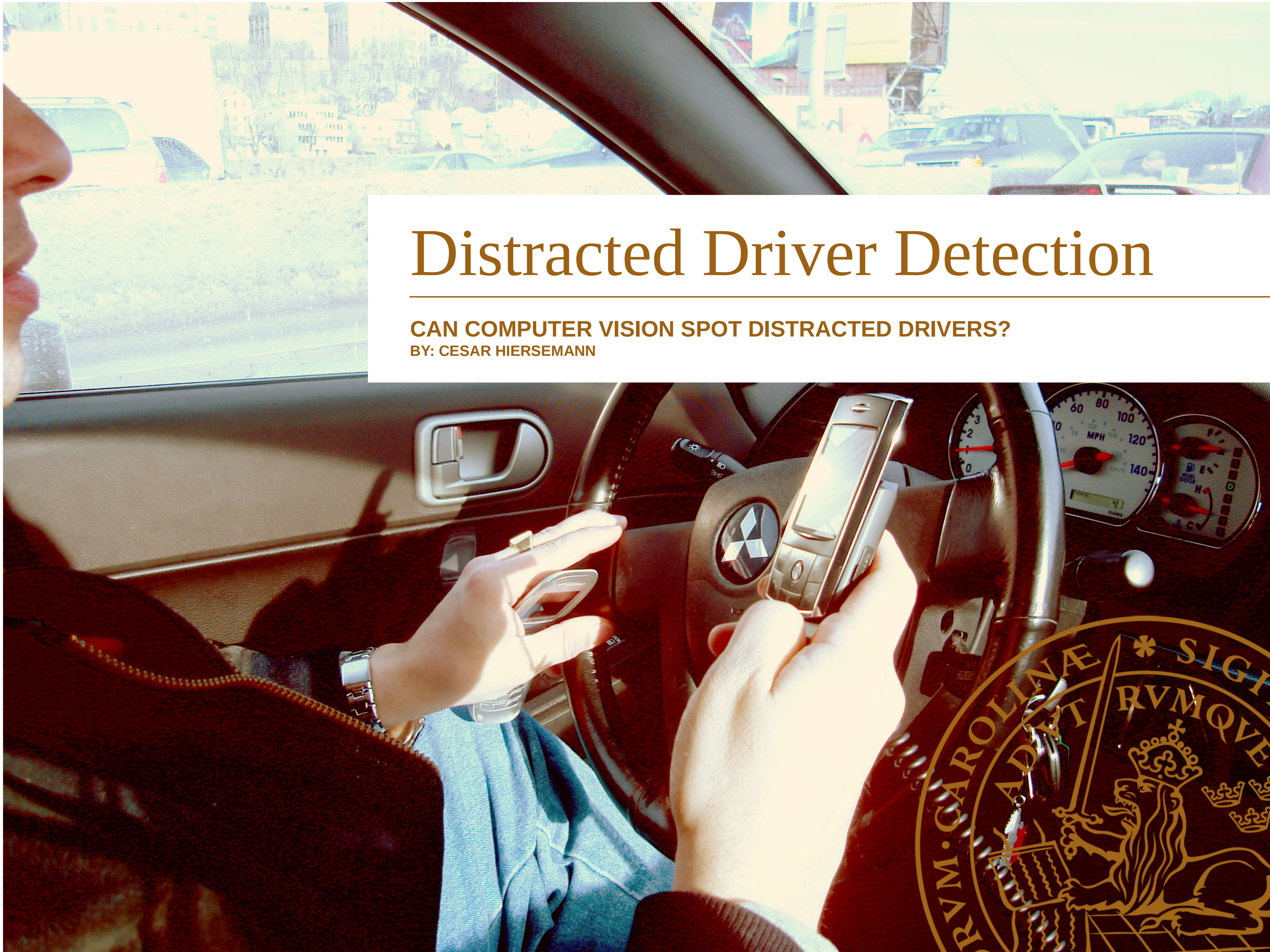


Distracted Driver Detection

CAN COMPUTER VISION SPOT DISTRACTED DRIVERS?

BY: CESAR HIERSEMANN

Image understanding is hard!

- ”Easy for humans, hard for computers”
- Relevant XKCD (posted in 2014)

http://xkcd.com/1425/

LUND UNIVERSITY

Outline

- Problem introduction
- Theory
  - Neural Networks
  - ConvNets
  - Deep Pre-trained with example
- My approach
- Challenges
- Results

LUND UNIVERSITY

Distracted Drivers competition[1]

- Kaggle – Data science competitions
- Dataset:
  - Over 100 000 images (>4 Gb)
  - 100 persons performing 10 different actions (next slides)
  - Labelled training set with ~20K images, test set ~80K
- Task is to label test set with probabilities for each class
- Evaluation by *multi-class logloss:*

$$L = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{M} y_{ij} \log(p_{ij})$$

[1]: https://www.kaggle.com/c/state-farm-distracted-driver-detection

LUND UNIVERSITY

Action classes

- C0:

  Driving safely

- C1:

  Texting right

- C2:

  Talking right

- C3:

  Texting left

Action classes cont.

- C4:

  Talking left

- C5:

  Operating

  radio

- C6:

  Drinking

- C7:

  Reaching

  back

LUND UNIVERSITY

Action classes cont.

- C8:

  Hair and makeup

- C9:

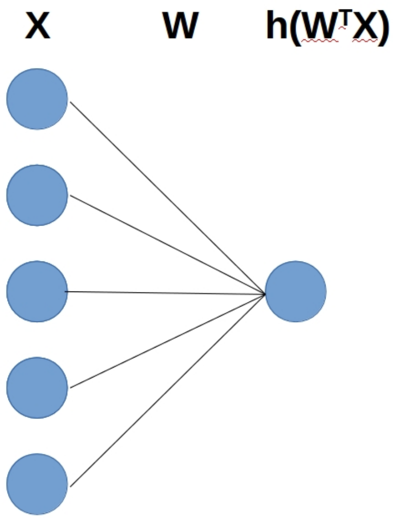
  Talking to passenger

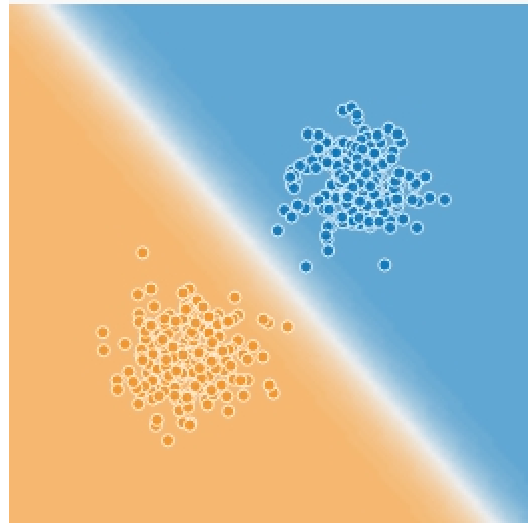
Neural networks

- One node with

  sigmoid activation

  = logistic regression





- Many nodes/layers → learns complex input/output relations with cheap operations

  Demo[2]: Link

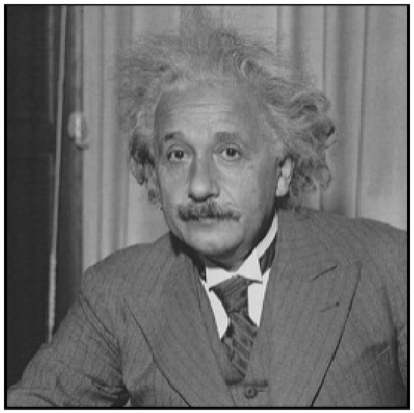
[2]: Tensorflow Playground: http://playground.tensorflow.org/

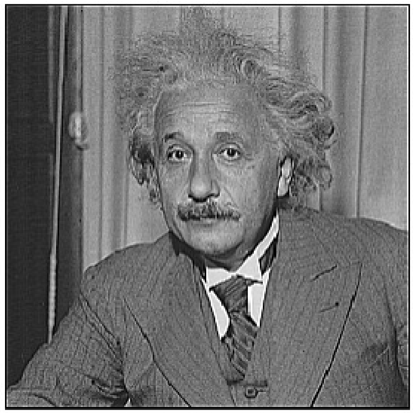
ConvNets

- Convolution (”faltning”)
  - Fourier/Laplace transform
  - Image analysis
  - Signal Processing
- Filter on images
- Ex:
  - Gaussian Blur
  - Sharpening
  - Edge detection
- ConvNets include *convolutional layers*

Sharpening filter

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

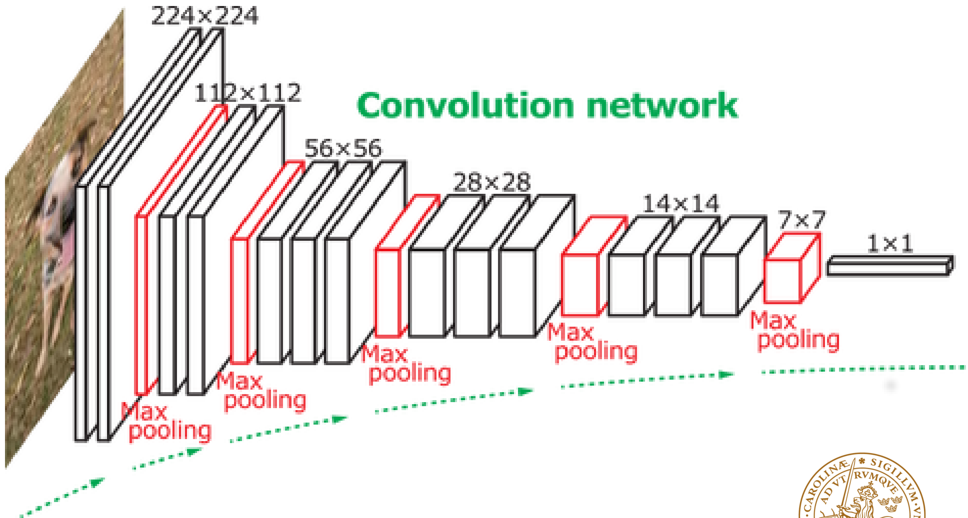




Deep ConvNet, VGG16[3]

- 16 conv. Layers + 4 fully connected (”normal”) layers
- > 138 million parameters
- 2-3 weeks to train on ImageNet database
- 1.3 million images from 1000 classes

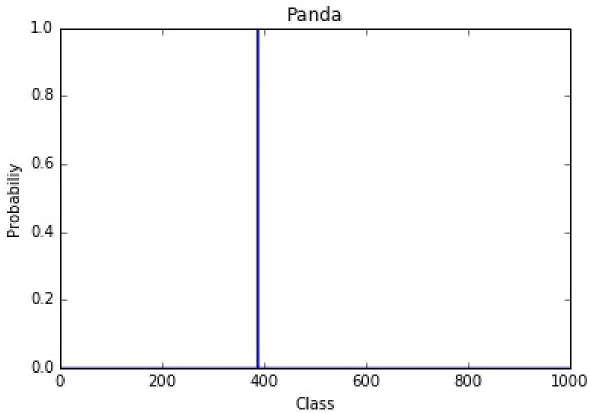
VGG16 architecture



[3]: VGG-16 network [http://arxiv.org/abs/1409.1556]

VGG16 Demo

- Giant Panda image from Hong Kong Zoo
- VGG16 gives output:

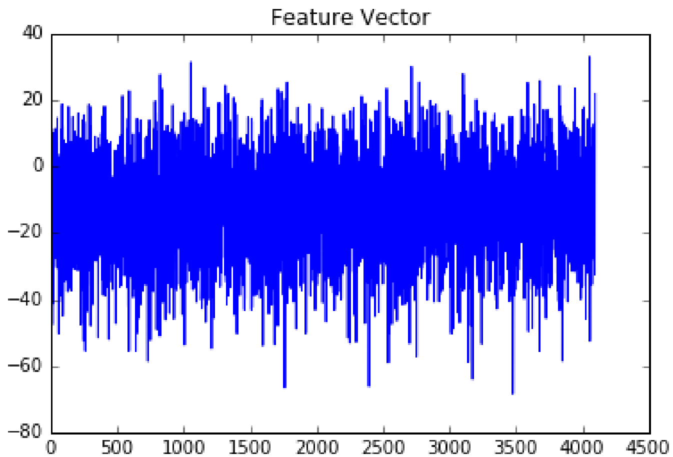


- 99.9999% confidence in class 388:

  giant panda, panda, panda bear, coon bear, Ailuropoda melanoleuca

LUND UNIVERSITY

Back to the drivers!



- Use pre-trained VGG16 to extract feature-vectors from images
- Use first layer after the convolutions, produces 4096-dimensional vector
- Every image takes 0.5s to process → ~20h on laptop

LUND UNIVERSITY

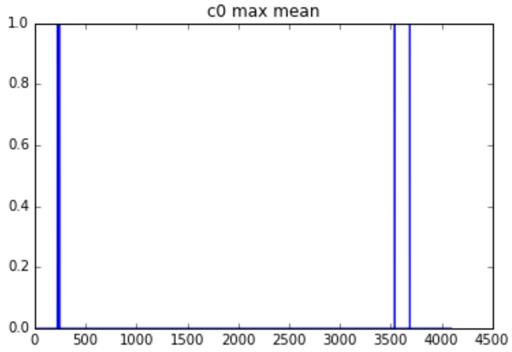
Will it work?

- Seperability of classes

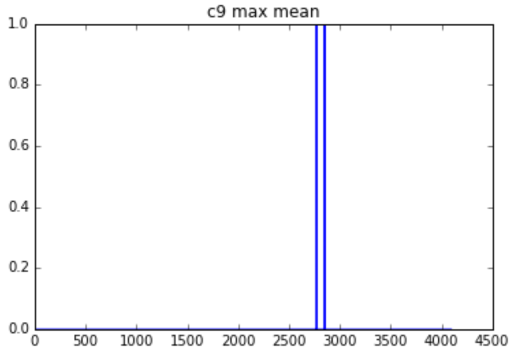
- Mean output over different classes

- Seemed to show good variability → good chance of seperation

- Promising!



Max activations Class 0



Max activations Class 9

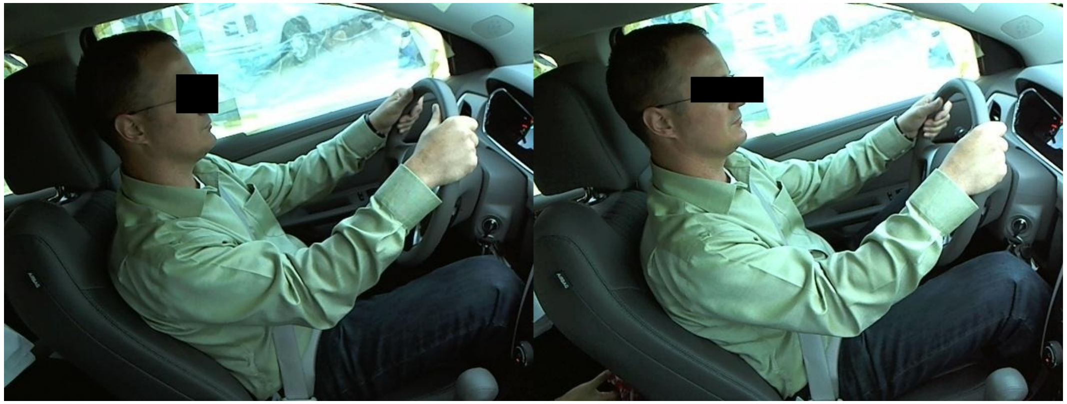
LUND UNIVERSITY

Classification challenges

- Many similar images taken within short timeframes → prone to overfitting

- Seperate persons in train and test set

- Network learned person-specifics → bad results on test!

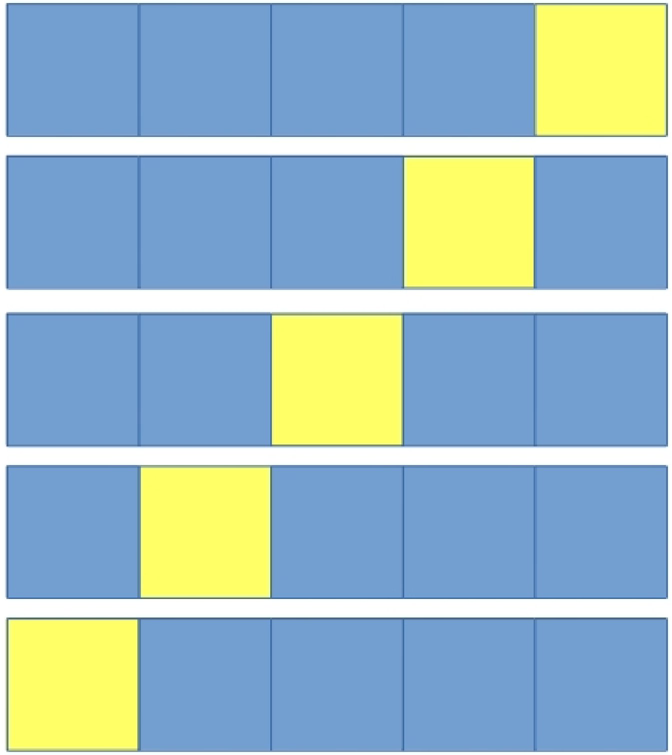


Two similar images from C0: safe driving

LUND UNIVERSITY

Labelled cross-validation

- To recieve accurate test evaluations, cross-validation is required
- 26 different persons in train set
- Split my training set into 5 folds with 5 persons held out from training



LUND UNIVERSITY

Classification

- Now I had a:
  - train matrix 22424 x 4096
  - test matrix 79726 x 4096
- Many approaches to classification:
  - Support vector machine
  - Logistic regression
  - Random forest
  - Decision Trees
  - Gradient Boosting
- SVM and Log.Reg produced best res. (implemented in scikit-learn)

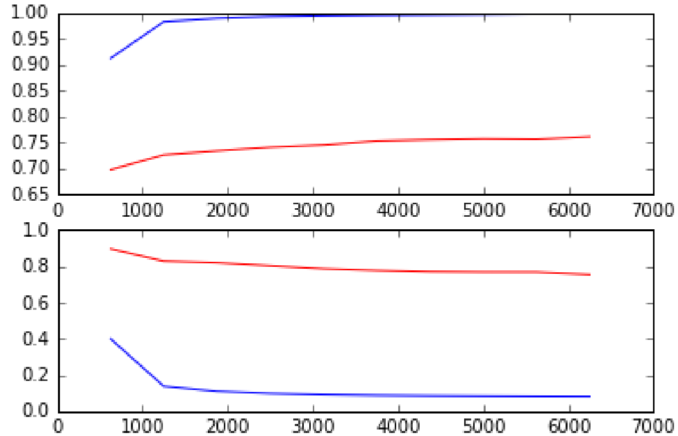
LUND UNIVERSITY

Training

- Using the entire 4096 feature vector for every image (testing took time!)

- Regularization:
  - Prevents overfitting by limiting size of weights
  - An additional hyperparameter to optimize

- Finding the right hyperparameters using cross-validation



Train (blue) and validation (red) acc. (top) and logloss (bottom)

Improvements

- 60-65% accuracy, 1.10 logloss → ~250 on current leaderboards

- Wanted less features per image

- Reduces training time – more time to optimize hyperparameters

- Finding the ”right” features for my specific task will greatly prevent overfitting

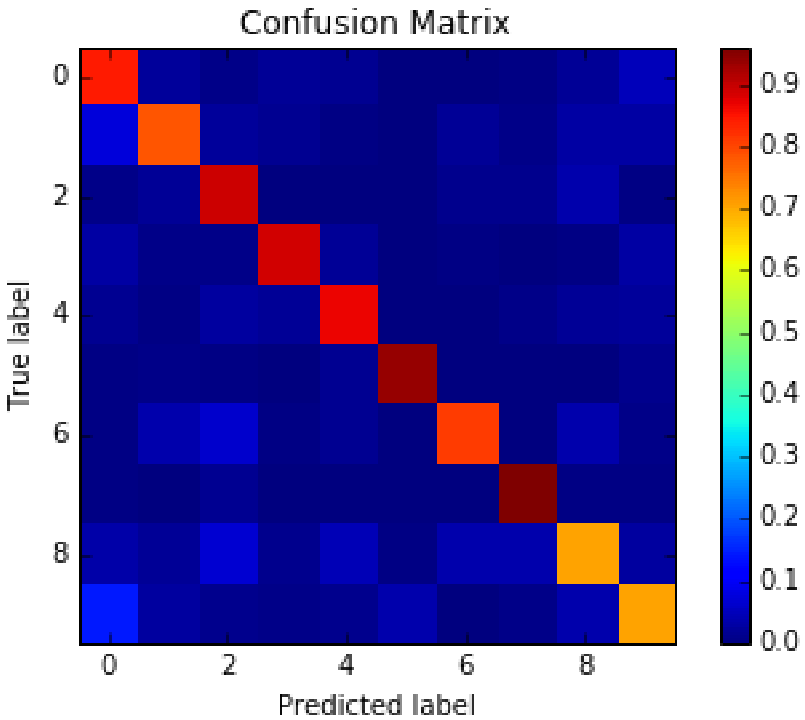
LUND UNIVERSITY

Dimensionality reduction

- Which features were the most important
- Removing features that coded for person-specifics
- Ended up with 887 feature vector → much faster training/testing and easier on the memory

Final Results

- Over 80% accuracy and <0.60 logloss on cross-validation!



- Sadly nowhere close to <0.2 logloss (top of LB) :(

Thanks!

- Dennis Medved
- Pierre Nugues
- Magnus Oskarsson

- Have a great summer!