link to dataset: <http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html>

link to book: <https://learning.oreilly.com/library/view/deep-learning-with/9781617295546/kindle_split_015.html>

<https://neurohive.io/en/popular-networks/vgg16/>

<http://image-net.org/explore>

link to an example of feature extraction:

[Using Keras' Pre-trained Models for Feature Extraction in Image Clustering](https://medium.com/@franky07724_57962/using-keras-pre-trained-models-for-feature-extraction-in-image-clustering-a142c6cdf5b1)

model.layers.pop() for removing layers

#178x218 pixels

#use imagenet weights when normalizing

1. fit the model as good as possible
2. extract the features from 2nd last layer...
3. use them on other ML models
4. We compare ML v.s. Neural Networks for various pre-trained CNNs(VGGs, ResNet50, they were pretrained on different image datasets)

2020.1.7 note

\*Lowering the learning rate can help reduce the fluctuation in loss curve. (If learning rate is high, the model learns too much information in a single epoch, that’s why the loss scores are so different in value over epochs)

\*Not only the “U” shape loss curve indicates overfitting, it’s actually the sudden change of the improving speed that indicates overfitting. So if the curve tends to drop at the beginning and becomes flat afterwards, it’s considered overfitting after the drop.(That’s where we need to cut the epochs, we use the accuracy before overfitting as the performance)

Also, changing batch-size and stuff is only for the purpose that we can clearly see what’s the real performance(accuracy) before overfitting.

Batch size is the amount of objects the network will process in each epoch, larger batch size speeds up the learning, but consumes more RAM, after each batch of learning, weights will be updated, the update will benefit the next batch of learning, so we better have smaller batch size if we are not in a hurry. (**batch - how many samples to show for each weights update**).

\*To fit Random Forest, we can create our custom Random Forest layer and replace the last classification layer with it.

I need to read about dense layers especially about how we should feed the feature extracted after dense\_flatten layer <https://stackoverflow.com/questions/54438994/feed-deep-features-to-machine-learning-classifiers-random-forest>

\*We compare ML v.s. CNN for various pre-trained CNNs(VGGs, ResNet50, they were pretrained on different image datasets)

A good article showing an example of RF v.s. NN

<https://www.kdnuggets.com/2019/06/random-forest-vs-neural-network.html>

our goal:

* test and see if nn could be used as an dimension reduction techniqe to be used on other ML-modells to improve the preformance
* to pop out different amount of features to really se if this is possible
* if we are going to use another test size i think we should rerun the import and probably USING A SEED! :)
* thinking that we could import only the validation.training data (2000), train the data on that and then import test data of different size to evaluate.

thing for the report and presentation:

* would be fun to try on photos of our self and/or extract pictures of the wrongly classified pictures to see for example indication of biased training data. WE MAY HAVE INTERSETING LABELS TO LOOK AT.

split the data and labelfile into training validation THEN create a new label list with only the variable Male/Female

RESULTS val\_acc:

after around 30 epochs

500+100+20+10+4+1 : 91-92

500+100+20+10+1 : 91-92 slightly better choose this model

---------------------

popping layers:

10 features: 0.9015 lasso

20 features:

100 features:

500 features:

which ML-models:

* regularised logistic (lasso or ridge or en)
* random forest

-------------------------

PRESENTATION

-------------------------

emil:

method:

Based on our classification tadk

Binary crossentropy loss function

An rmsprop optimizer

A sigmoid activation function of the last layer

The convolutional base we used is called VGG16. It is pre-trained from 14 million images from imagenet which makes it suitable as a base for our small sample.

First we constructed a baseline NN model. We freezed the convolutional base in order to keep its pre-trained weights from changing during training.

Then we added one dense layer of 20 units. However this model started to overfit, which we didn’t expect because the usual workflow is to find a baseline that doesn’t overfit and then find a complex model.

We also did data augmentation in order for the model to generalize better. For example, rotating the images 30 degrees, etc.

This procedure randomly transforms the images which can help the model train on different aspects of the data

We also added regularisation such as dropout and tried different configs of layer and we ended up with the NN in the figure here.

We extracted the features from the second to last layer as input for our machine learning models.

We wanted to use a regression model and a tree-based model so we used a logistic LASSO regression and a random forest to compare their performances.

We also tried extracting the features from the convbase directly (8192 units) but it had difficulty converting in LASSO and took extremely long time to train a random forest.

-----------------

graph:

To the left we see our baseline model. As we can see, the validation curve increses its loss curve rather quickly which indicates overfit.

The complex model is more stable, where the validation accuracy is in general higher than the training accuracy.

----------