SIBS 2017

Deep learning for single cell phenotype classification in High-Content Screening

Oliver Dürr

Institute of Data Analysis and Process Design
Zurich University of Applied Sciences

Zurich, 13th October 2017

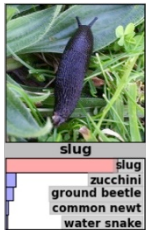
Outline: *Deep learning for single cell phenotype classification in High-Content Screening*

- Short Introduction to CNN
- A straight forward application of CNN for HCS
- Challenges in HCS
  - No labeled cells in the beginning
  - How much labeled data is needed?
  - Measuring confidence of the phenotype classifications

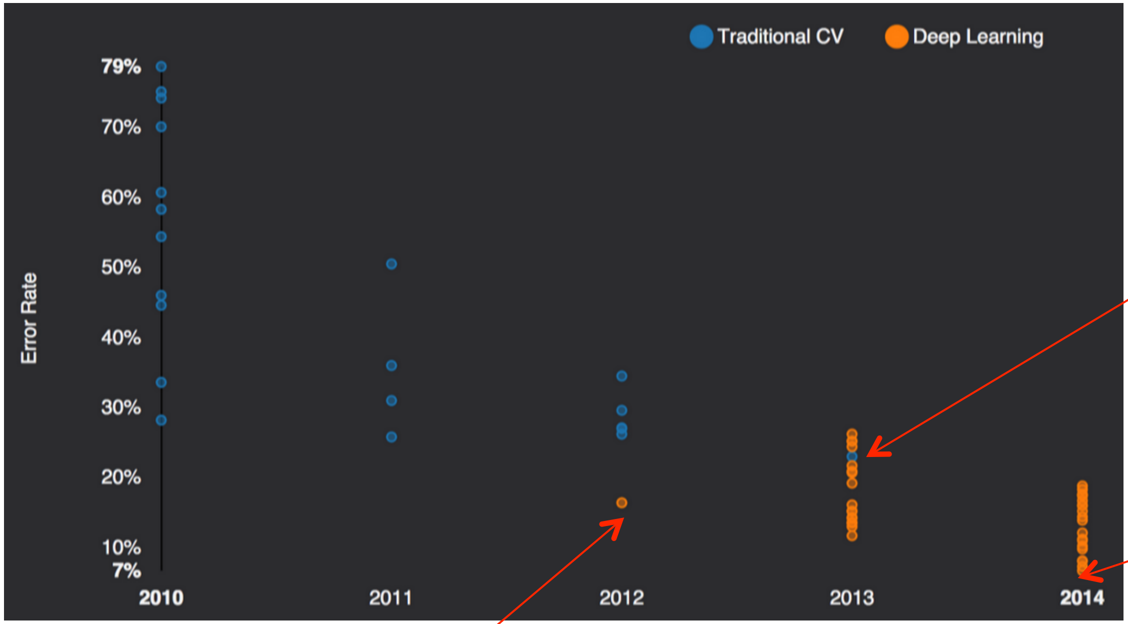
A short introduction to CNN

Why DL: Imagenet 2012, 2013, 2014, 2015

1000 classes
1 Mio samples



…



Human: 5% misclassification

Only one non-CNN approach in 2013

GoogLeNet 6.7%

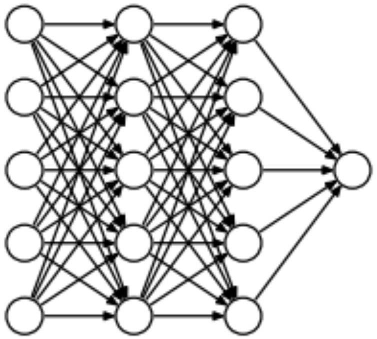
A. Krizhevsky
first CNN in 2012
Und es hat zoom gemacht

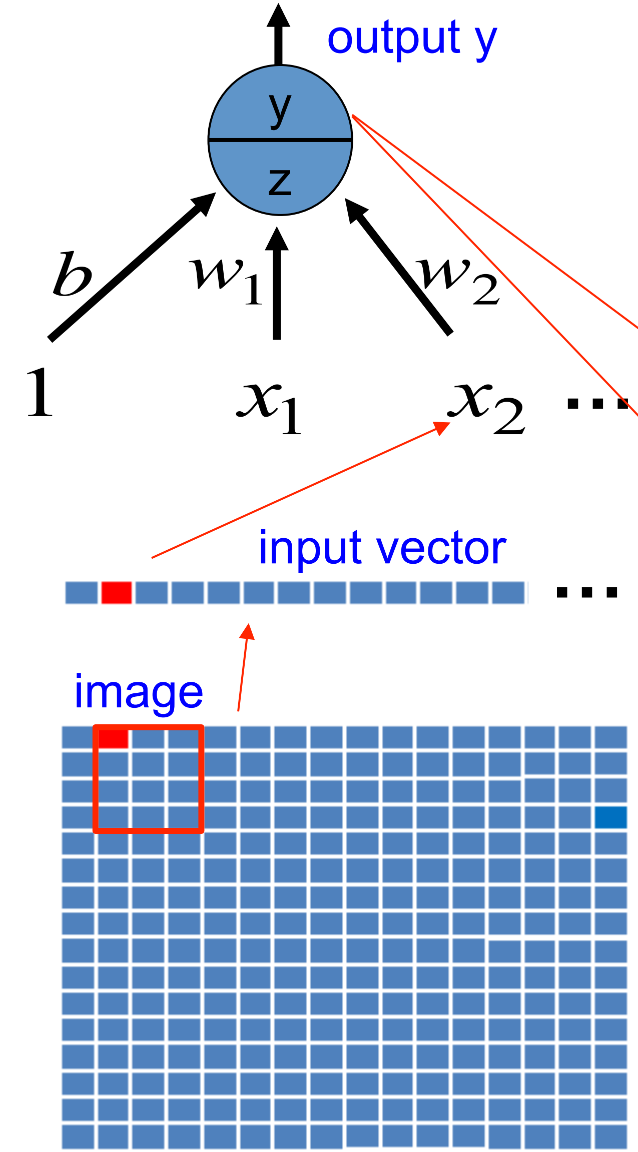
2015: It gets tougher

4.95% Microsoft (Feb 6 surpassing human performance 5.1%)
4.8% Google (Feb 11) -> further improved to 3.6 (Dec)?
4.58% Baidu (May 11 banned due too many submissions)
3.57% Microsoft (Resnet winner 2015)

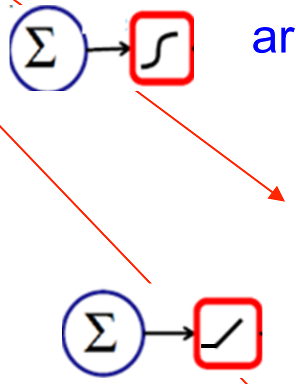
Figure: https://medium.com/global-silicon-valley/machine-learning-yesterday-today-tomorrow-3d3023c7b519

An artificial neuron

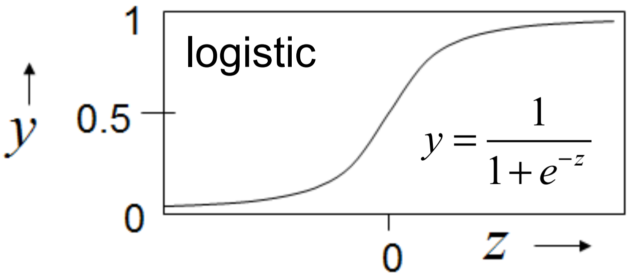


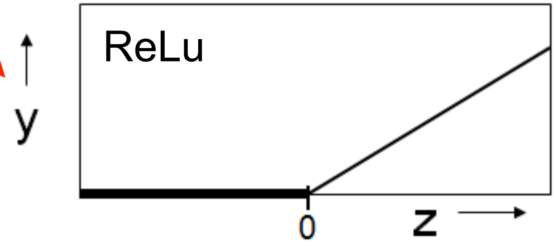


$$z = \overset{\text{bias}}{b} + \sum_i x_i \overset{\text{weights}}{w_i}$$

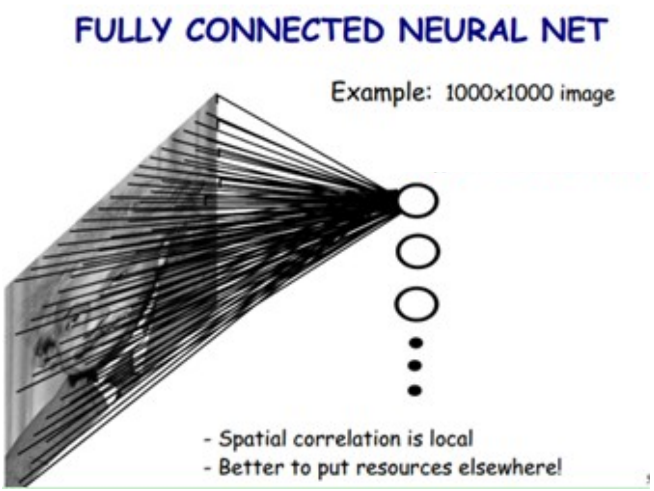


Different non-linear transformations are used to get from z to output y





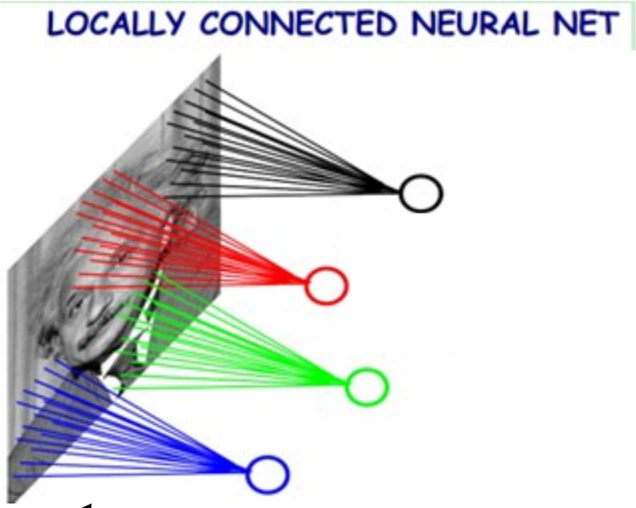
Convolution extracts local information using few weights



FULLY CONNECTED NEURAL NET

Example: 1000x1000 image

- Spatial correlation is local
- Better to put resources elsewhere!



LOCALLY CONNECTED NEURAL NET

Shared weights:

by using the same weights for each patch of the image we need much less parameters than in the fully connected NN and get from each patch the same kind of local feature information such as the presence of a edge.

Convolutional networks use neighborhood information and replicated local feature extraction

In a locally connected network the calculation rule

$$z = b + \sum_i x_i w_i$$

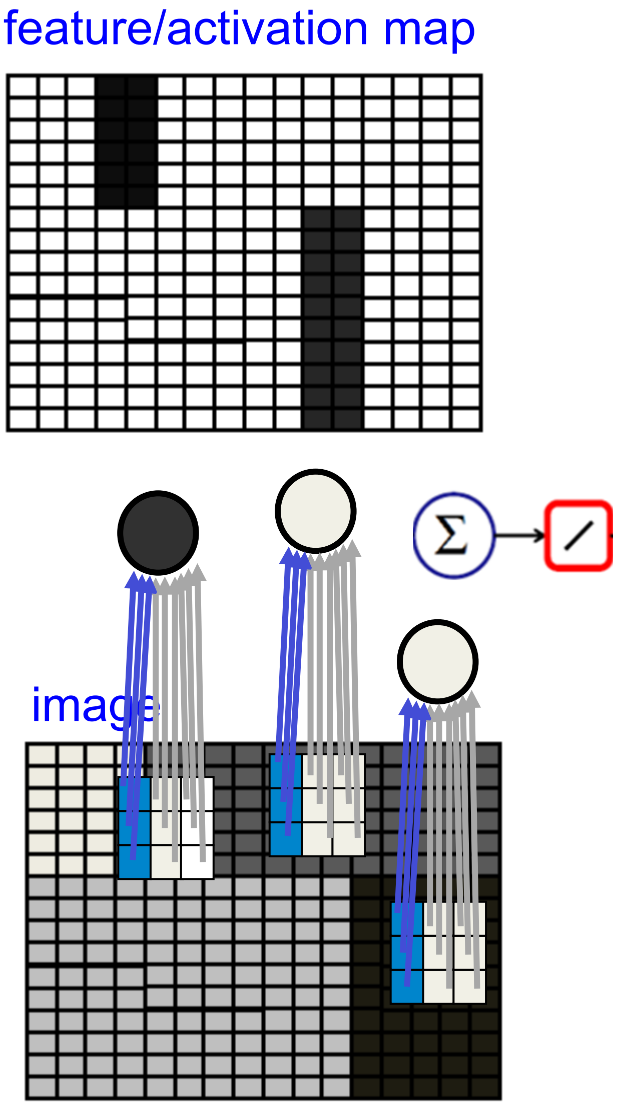
Pixel values in a small image patch are element-wise multiplied with weights of a small filter/kernel:

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

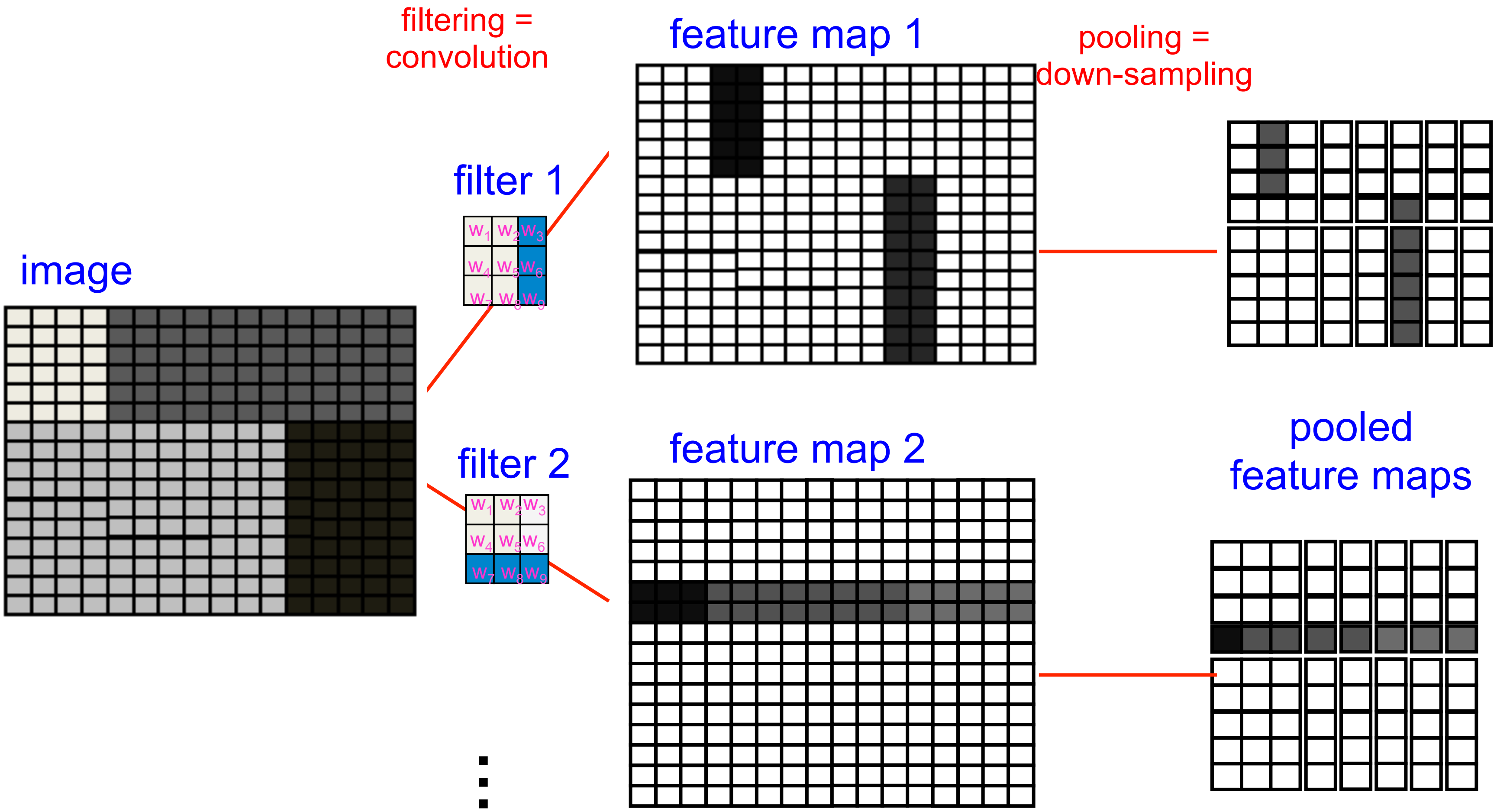
1	-1	-1
1	-1	-1
1	-1	-1

The filter is applied at each position of the image and it can be shown that the result is maximal if the image pattern corresponds to the weight pattern.

The results form again an image called feature map (=activation map) which shows at which position the feature is present.



Convolutional networks use neighborhood information and replicated local feature extraction

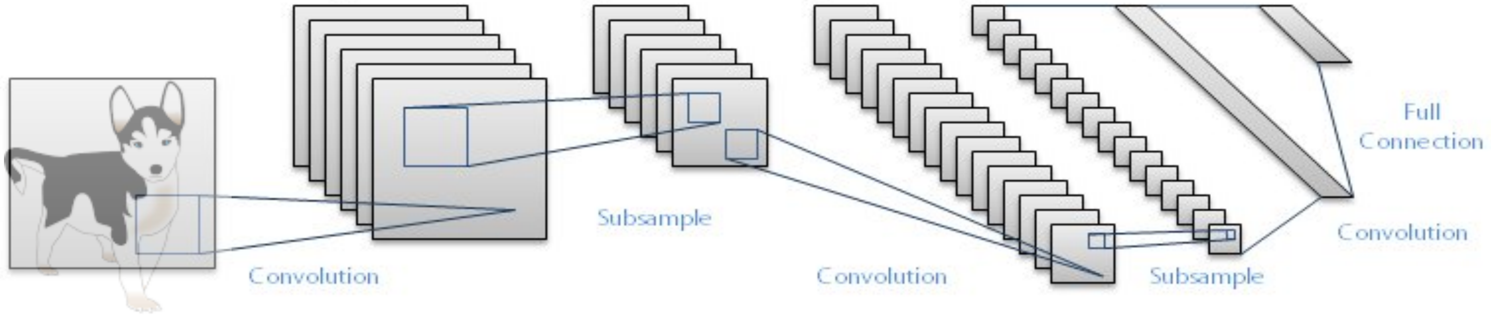


The weights of each filter are randomly initiated and then adapted during the training.

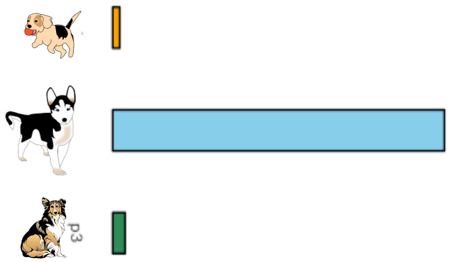
Putting it all together

Feature Extraction

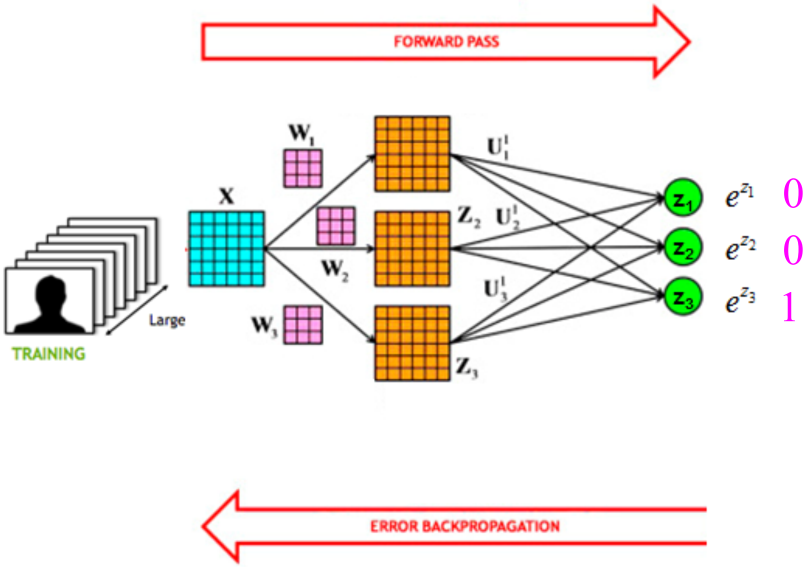
Classification



Output probability for class



Training of a CNN is based on gradient backpropagation



Backpropagation="chain rule"

Loss-function:

L=distance(truth, output(w)**)**

Minimize Loss Function:

$$w_i^{(t)} = w_i^{(t-1)} - l^{(t)} \left.\frac{\partial L(w)}{\partial w_i}\right|_{w_i=w_i^{(t-1)}}$$

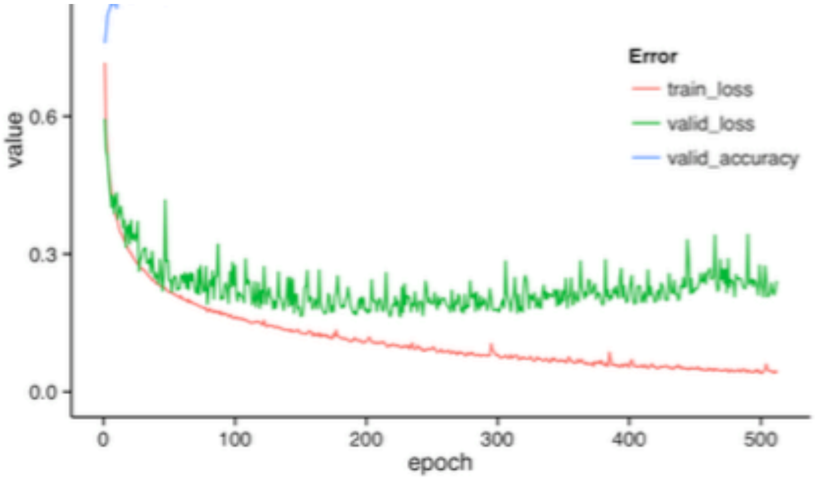
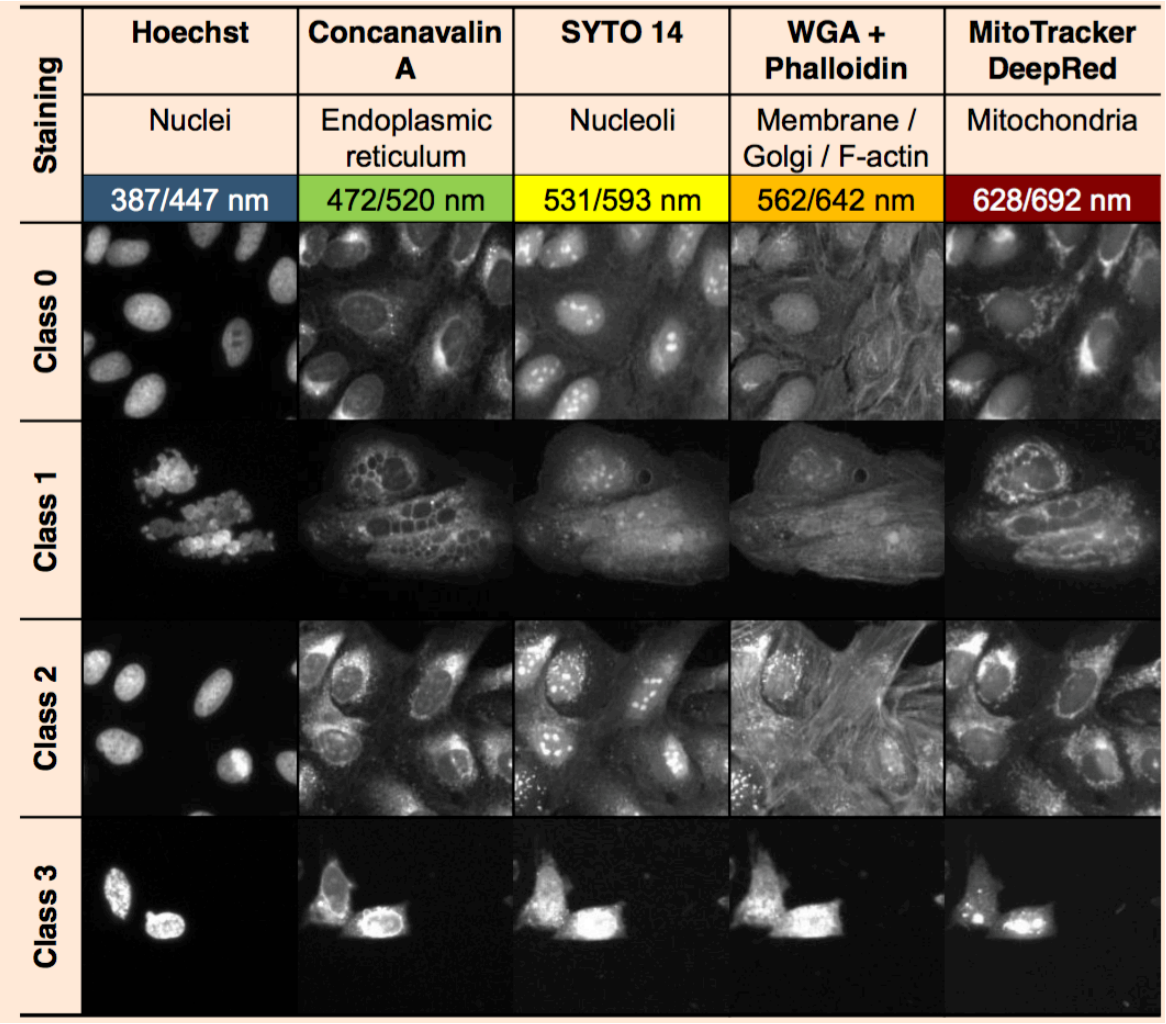


Image credits: Audhkhasi et al., Neural Networks 78 (2016) 15–23

A straight forward application to phenotype classification

Dataset: BBBC022v1 "Cell Painting Assay"



Staining	Hoechst	Concanavalin A	SYTO 14	WGA + Phalloidin	MitoTracker DeepRed
	Nuclei	Endoplasmic reticulum	Nucleoli	Membrane / Golgi / F-actin	Mitochondria
	387/447 nm	472/520 nm	531/593 nm	562/642 nm	628/692 nm

Compound classes:

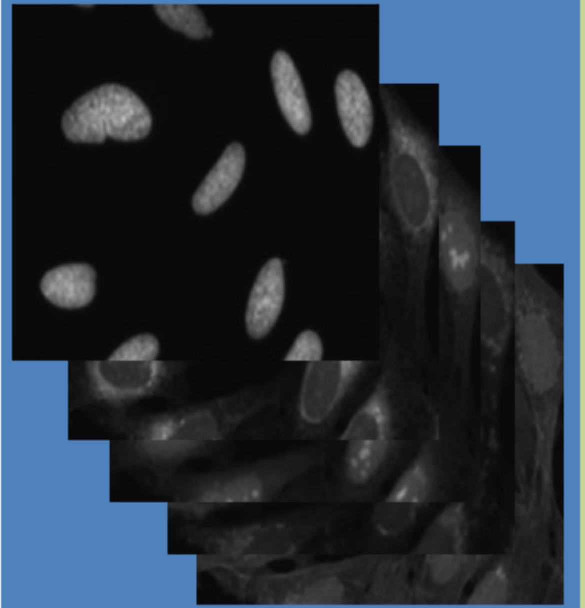
0 DMSO,
1 e.g. Paclitaxel
2 e.g. Metoclopramide
3 e.g. Digoxin

Data Split
52'000 segmented cells
(using Cellprofiler)
20% testing

Dürr, O., and Sick, B. Single-cell phenotype classification using deep convolutional neural networks. Journal of biomolecular screening 21, 9 (2016), 998-1003. Image Genedata

The workflow

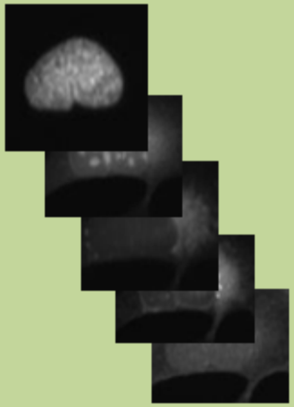
Image data



5 channels

Segmentation

Cellprofiler were used for cell segmentation

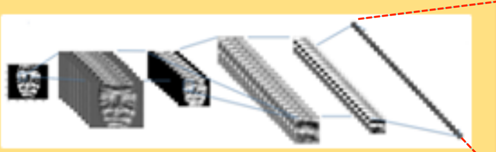


Feature extraction

Cellprofiler (traditional)

Predefined feature

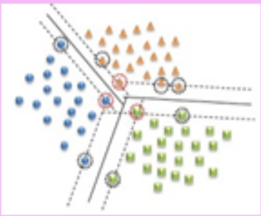
CNN (new approach)



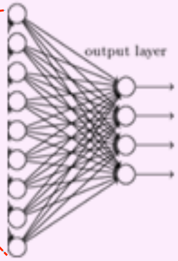
learned task specific feature

Classification

LDA/SVM (traditional)



last layer



Dürr, O., and Sick, B. Single-cell phenotype classification using deep convolutional neural networks. Journal of biomolecular screening 21, 9 (2016), 998-1003

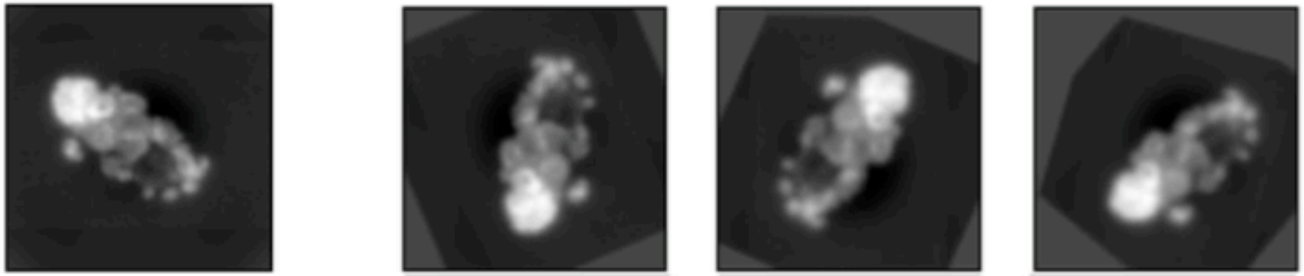
Definition of the network

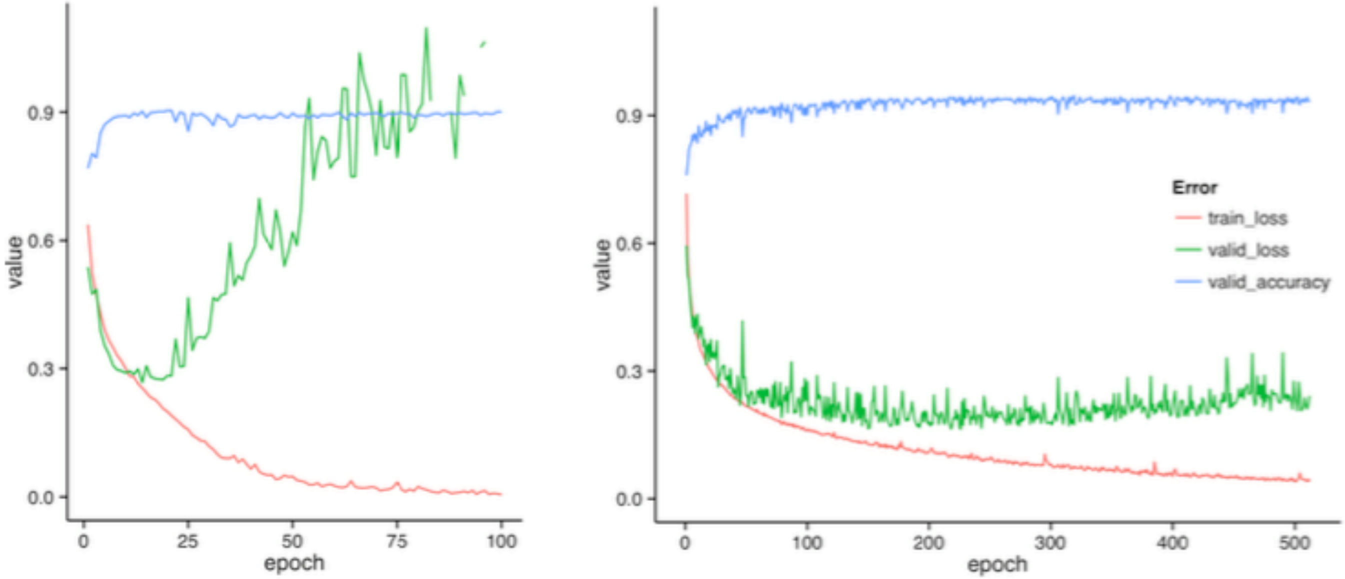
```
|  0 | input    | 5x72x72    |

|  1 | conv1    | 32x70x70   |
|  2 | conv11   | 32x68x68   |
|  3 | pool1    | 32x34x34   |

|  4 | conv2    | 64x32x32   |
|  5 | conv22   | 64x30x30   |
|  6 | pool2    | 64x15x15   |

|  7 | conv3    | 128x13x13 |
|  8 | conv33   | 128x11x11  |
|  9 | pool3    | 128x6x6    |

| 10 | hidden1  | 200        |
| 11 | dropout1 | 200        |
| 12 | hidden2  | 200        |
| 13 | dropout2 | 200        |
| 14 | hidden3  | 50         |
| 15 | dropout3 | 50         |

| 16 | output   | 4          |
```

Inspired by the Oxfordnet, the 2nd best submission of the 2014 image net competition.

Dürr, O., and Sick, B. Single-cell phenotype classification using deep convolutional neural networks. Journal of biomolecular screening 21, 9 (2016), 998-1003

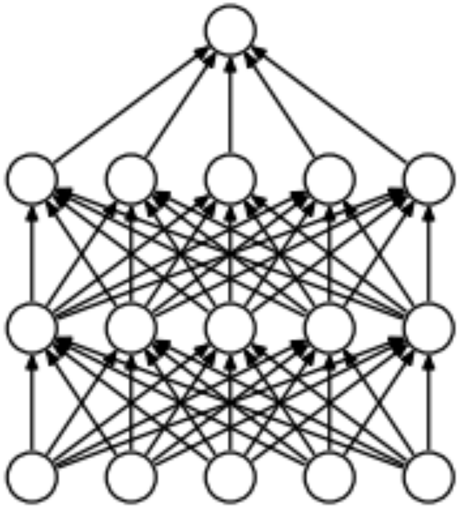
Making the most of your data: augmentation



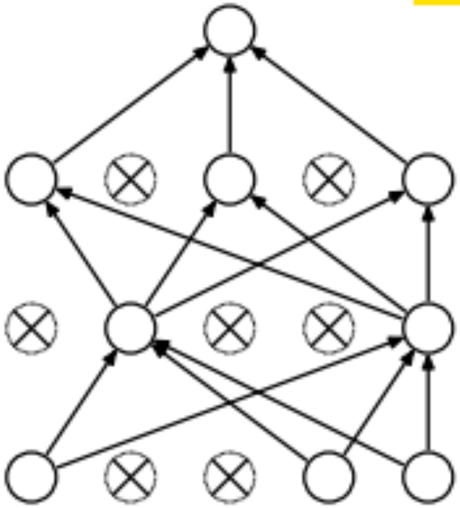


Dürr, O., and Sick, B. Single-cell phenotype classification using deep convolutional neural networks. Journal of biomolecular screening 21, 9 (2016), 998-1003

Making the most of your data: Dropout



(a) Standard Neural Net



(b) After applying dropout.

At each training step we remove random nodes with a probability of p.

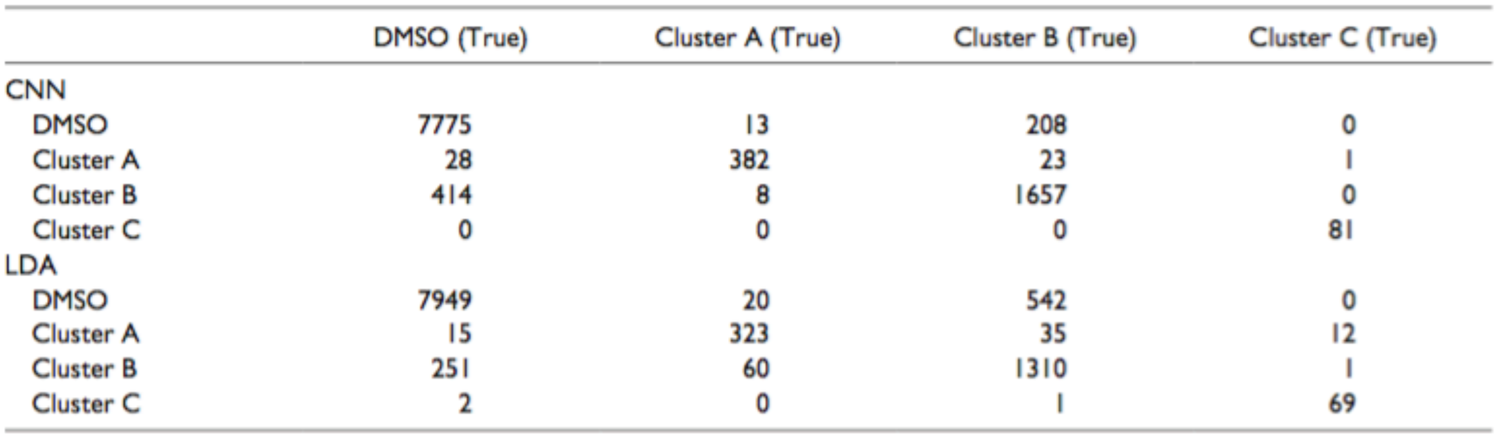
- Sparse version of the full net
- In each training step we train another NN model
- Dropout prevents co-adaptation

At testing no dropout

- Dropout can also be used later for assigning confidence!

Srivastava et al., *Journal of Machine Learning Research* 15 (2014), Image Martin Görner

Comparison with Cell Profiler



	DMSO (True)	Cluster A (True)	Cluster B (True)	Cluster C (True)
CNN				
DMSO	7775	13	208	0
Cluster A	28	382	23	1
Cluster B	414	8	1657	0
Cluster C	0	0	0	81
LDA				
DMSO	7949	20	542	0
Cluster A	15	323	35	12
Cluster B	251	60	1310	1
Cluster C	2	0	1	69

Overall accuracy: CNN 93.4% [93.0%, 93.9%], LDA 91.1% [90.5%,91.6%], SVM 87.6% [88.2%,87.0%]

Conclusion: Deep Learning applicable for phenotype classification in HCS. Better accuracy and no need for hand crafted features.

Dürr, O., and Sick, B. Single-cell phenotype classification using deep convolutional neural networks. Journal of biomolecular screening 21, 9 (2016), 998-1003

Challenges in High Content Screening

Challenges in High Content Screening

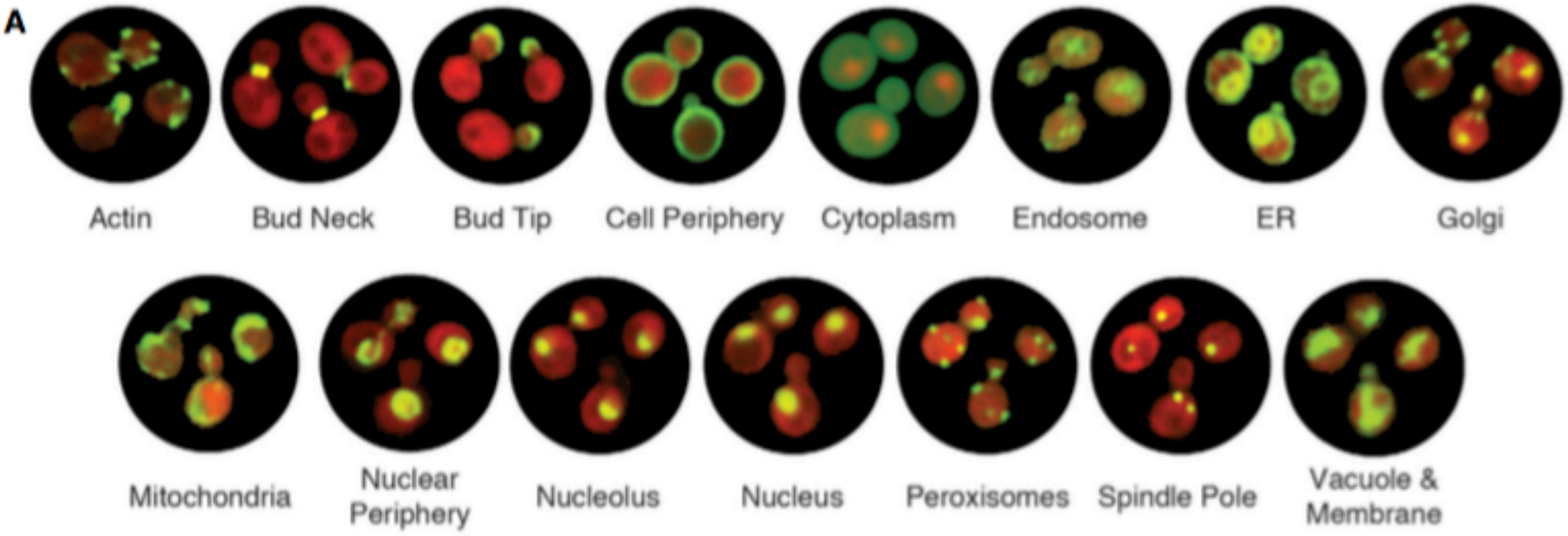
- Data is abundant / labels are scarce
  - Millions of Cells but only hundreds / thousands of labeled cells
- Heterogeneity of the data
  - Do we really have all relevant phenotypes in the training set?

- We will address the following points:
  - Overview w/o labeled cells
  - How many labeled cells are necessary?
  - Approaches to reduce the number of labeled cells

  - How to assign and use uncertainties

Definition of data set

- 19 Classes
- 2 Channels
- Data (all available from https://github.com/okraus/DeepLoc)
  - 21882 64x64x2 segmented images in training set
  - 4491, 4516 for validation and testing

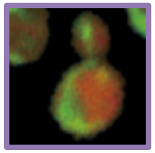


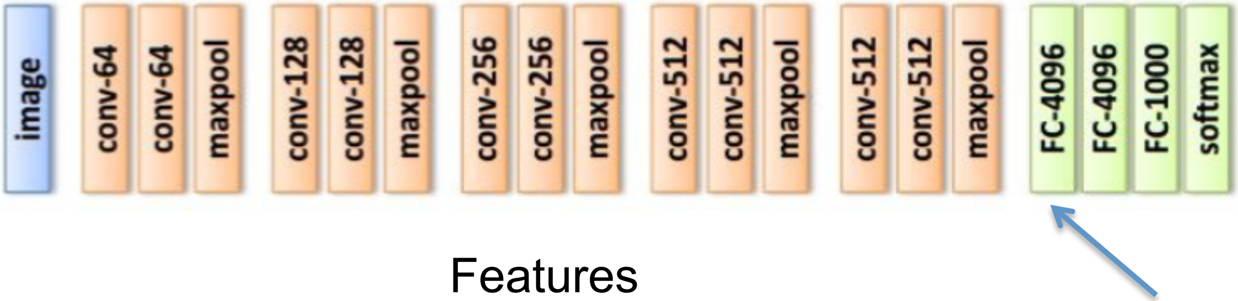
Kraus, O.Z. et all. , B.J. Molecular Systems Biology 13.4 (2017): 924

Overview of your data

- Visualization to get first impression
- At the beginning you have no labels
  - You can’t train a NN
- T-SNE on
  - Autoencoder
    - See Poster
  - Canned network VGG16 / FaceNet

Network trained on ImageNet

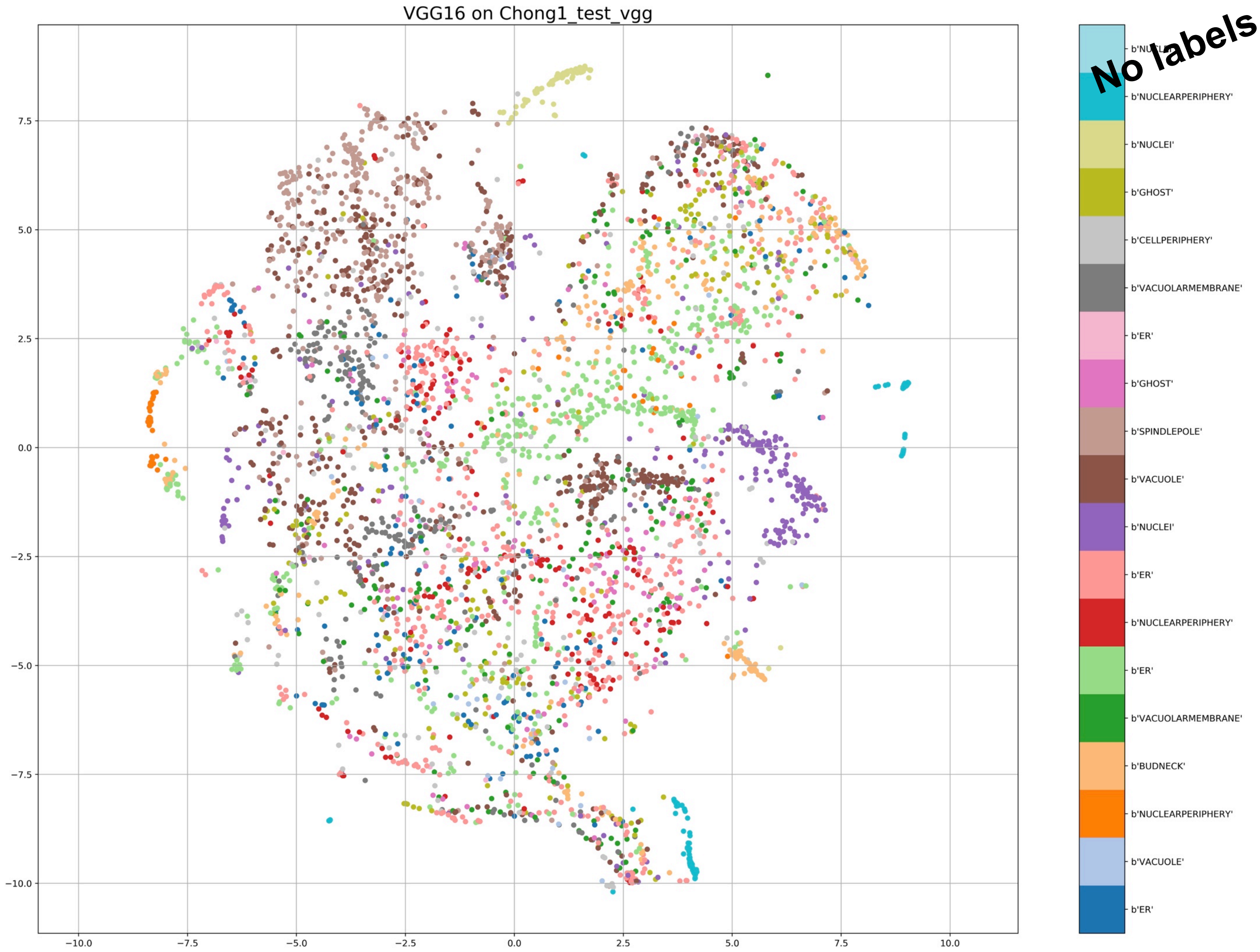




Features

4096 dimensional feature
vector as input for t-SNE
visualization

TSNE with VGG16 trained on ImageNet



No labels needed!

Definition of our CNN

```
Conv2D        (None, 64, 64, 32)       1184
Batch         (None, 64, 64, 32)       128
Conv2D        (None, 64, 64, 32)       9248
Batch         (None, 64, 64, 32)       128
MaxPooling2   (None, 32, 32, 32)       0
```

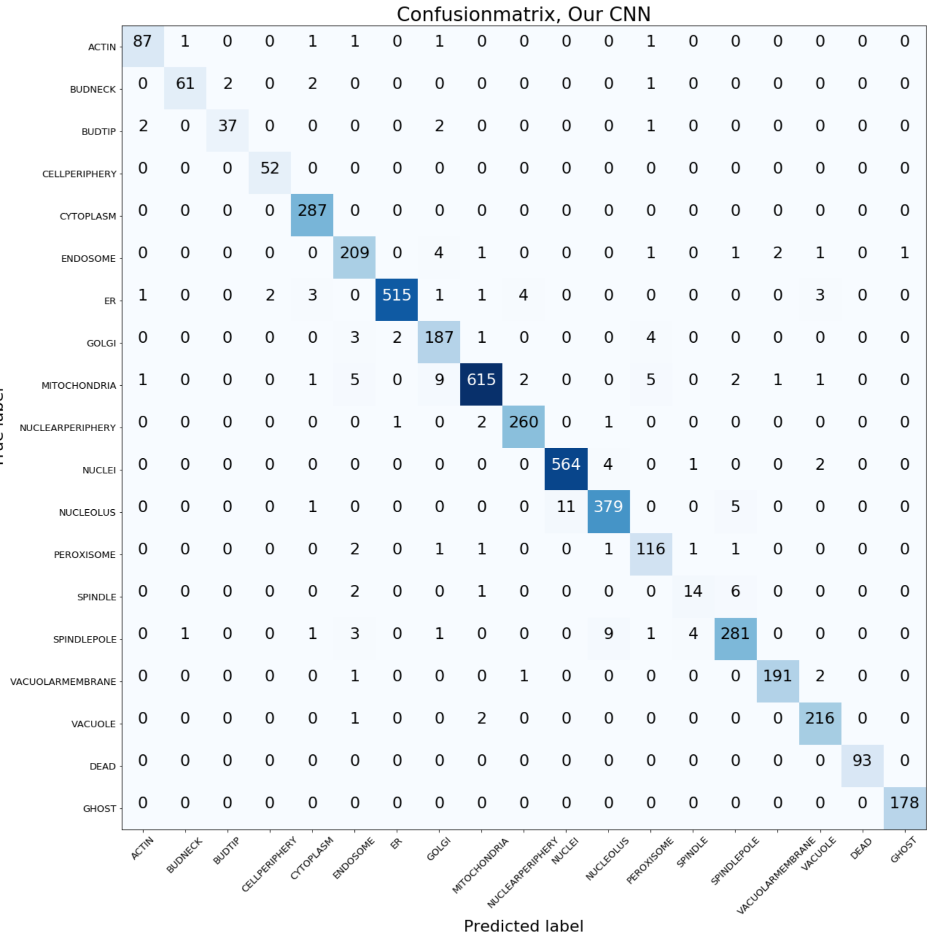
```
Conv2D        (None, 32, 32, 64)       18496
Batch         (None, 32, 32, 64)       256
Conv2D        (None, 32, 32, 64)       36928
Batch         (None, 32, 32, 64)       256
MaxPooling2   (None, 16, 16, 64)       0
```

```
Flatten       (None, 16384)            0
Dense         (None, 200)              3277000
Batch         (None, 200)               800
Dropout       (None, 200)              0
Softmax       (None, 19)               3819
```

Approx. 3 Mio. Parameters, 10 Mio. Oren Kraus

Results

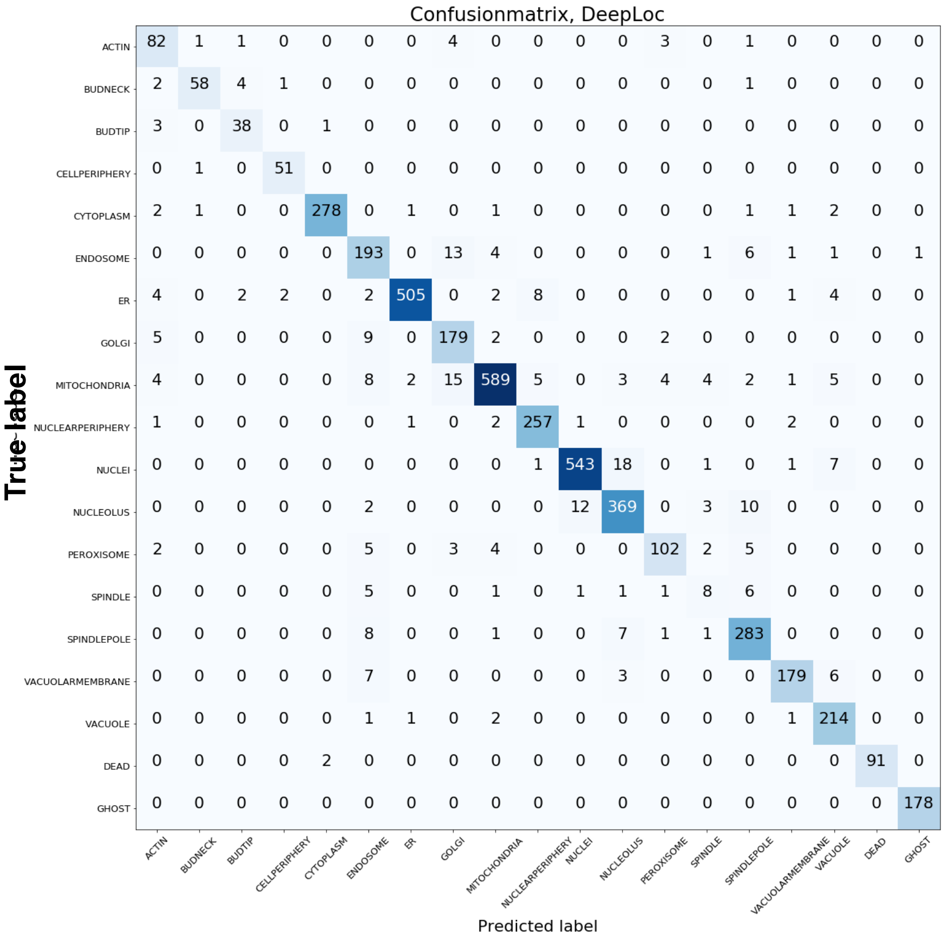
Our CNN



	ACTIN	BUDNECK	BUDTIP	CELLPERIPHERY	CYTOPLASM	ENDOSOME	ER	GOLGI	MITOCHONDRIA	NUCLEARPERIPHERY	NUCLEI	NUCLEOLUS	PEROXISOME	SPINDLE	SPINDLEPOLE	VACUOLARMEMBRANE	VACUOLE	DEAD	GHOST
ACTIN	87	1	0	0	1	1	0	1	0	0	0	0	1	0	0	0	0	0	0
BUDNECK	0	61	2	0	2	0	0	0	0	0	0	0	1	0	0	0	0	0	0
BUDTIP	2	0	37	0	0	0	0	2	0	0	0	0	1	0	0	0	0	0	0
CELLPERIPHERY	0	0	0	52	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CYTOPLASM	0	0	0	0	287	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ENDOSOME	0	0	0	0	0	209	0	4	1	0	0	0	1	0	1	2	1	0	1
ER	1	0	0	2	3	0	515	1	1	4	0	0	0	0	0	0	3	0	0
GOLGI	0	0	0	0	0	3	2	187	1	0	0	0	4	0	0	0	0	0	0
MITOCHONDRIA	1	0	0	0	1	5	0	9	615	2	0	0	5	0	2	1	1	0	0
NUCLEARPERIPHERY	0	0	0	0	0	0	1	0	2	260	0	1	0	0	0	0	0	0	0
NUCLEI	0	0	0	0	0	0	0	0	0	0	564	4	0	1	0	0	2	0	0
NUCLEOLUS	0	0	0	0	1	0	0	0	0	0	11	379	0	0	5	0	0	0	0
PEROXISOME	0	0	0	0	0	2	0	1	1	0	0	1	116	1	1	0	0	0	0
SPINDLE	0	0	0	0	0	2	0	0	1	0	0	0	0	14	6	0	0	0	0
SPINDLEPOLE	0	1	0	0	1	3	0	1	0	0	0	9	1	4	281	0	0	0	0
VACUOLARMEMBRANE	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	191	2	0	0
VACUOLE	0	0	0	0	0	1	0	0	2	0	0	0	0	0	0	0	216	0	0
DEAD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	93	0
GHOST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	178

Overall acc: 96.3% [95.7%,96.8%]

DeepLoc*

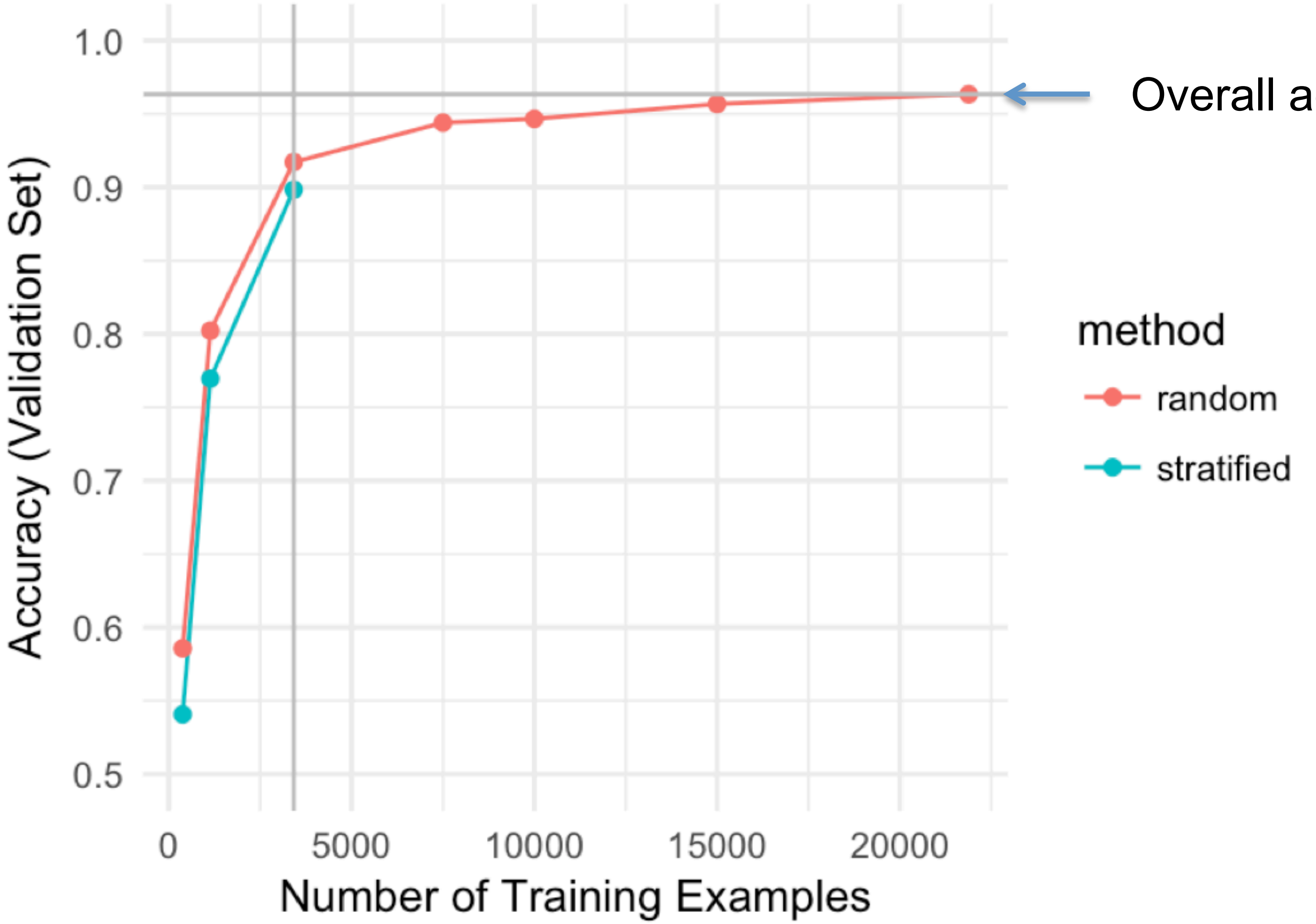


	ACTIN	BUDNECK	BUDTIP	CELLPERIPHERY	CYTOPLASM	ENDOSOME	ER	GOLGI	MITOCHONDRIA	NUCLEARPERIPHERY	NUCLEI	NUCLEOLUS	PEROXISOME	SPINDLE	SPINDLEPOLE	VACUOLARMEMBRANE	VACUOLE	DEAD	GHOST
ACTIN	82	1	1	0	0	0	0	4	0	0	0	0	3	0	1	0	0	0	0
BUDNECK	2	58	4	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
BUDTIP	3	0	38	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CELLPERIPHERY	0	1	0	51	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CYTOPLASM	2	1	0	0	278	0	1	0	1	0	0	0	0	0	1	1	2	0	0
ENDOSOME	0	0	0	0	0	193	0	13	4	0	0	0	0	1	6	1	1	0	1
ER	4	0	2	2	0	2	505	0	2	8	0	0	0	0	0	1	4	0	0
GOLGI	5	0	0	0	0	9	0	179	2	0	0	0	2	0	0	0	0	0	0
MITOCHONDRIA	4	0	0	0	0	8	2	15	589	5	0	3	4	4	2	1	5	0	0
NUCLEARPERIPHERY	1	0	0	0	0	0	1	0	2	257	1	0	0	0	0	2	0	0	0
NUCLEI	0	0	0	0	0	0	0	0	0	1	543	18	0	1	0	1	7	0	0
NUCLEOLUS	0	0	0	0	0	2	0	0	0	0	12	369	0	3	10	0	0	0	0
PEROXISOME	2	0	0	0	0	5	0	3	4	0	0	0	102	2	5	0	0	0	0
SPINDLE	0	0	0	0	0	5	0	0	1	0	1	1	1	8	6	0	0	0	0
SPINDLEPOLE	0	0	0	0	0	8	0	0	1	0	0	7	1	1	283	0	0	0	0
VACUOLARMEMBRANE	0	0	0	0	0	7	0	0	0	0	0	3	0	0	0	179	6	0	0
VACUOLE	0	0	0	0	0	1	1	0	2	0	0	0	0	0	0	1	214	0	0
DEAD	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	91	0
GHOST	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	178

Overall acc: 93.5% [92.7%, 94.2%]

*Oren Z Kraus et al. Mol Syst Biol 2017;13:924, trained model from: https://github.com/okraus/DeepLoc/

How much data do we need need?



For a small network with only 3420 images or 180 images per phenotype, more than 90% accuracy is reached (with augmentation)

Quantifying Uncertainty

Why do we want probability estimates?

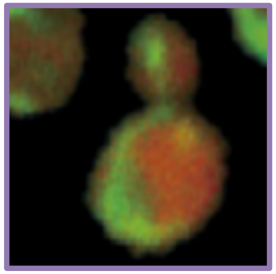
- There are so many cells
  - Only include cell for which the classifier is sure.

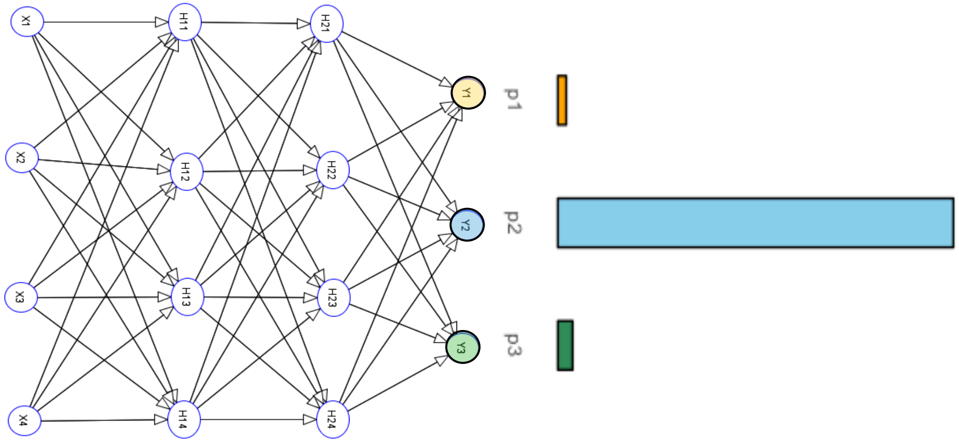
- Condense to one value per compound.
  - Use averages weighted with confidence

- Cells for which the classifier is unsure might hint towards novel or rare phenotypes

Don't we have probabilities anyway?

- Why don’t take the output of the softmax as an estimate for uncertainty?



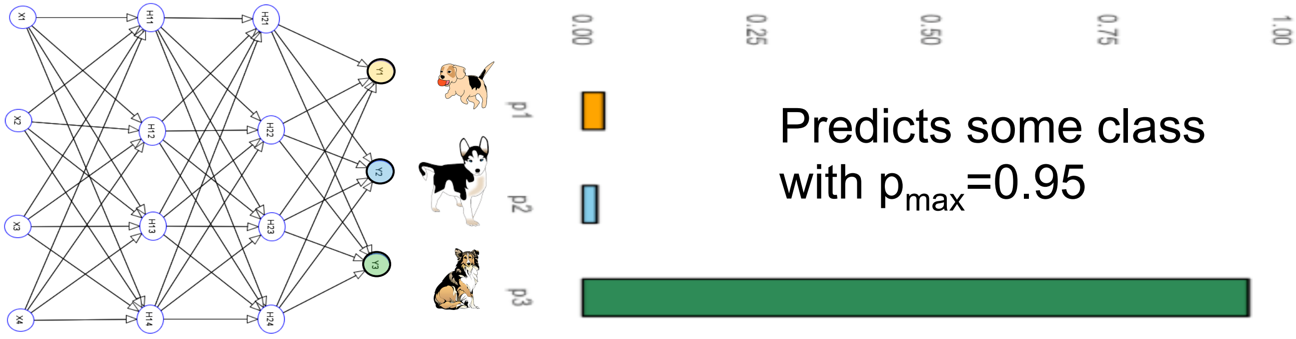


- These are probabilities (in a mathematical sense) but do not reflect the models / classifiers knowledge or ignorance.

- They don’t have error bars!

A first thought experiment

- Suppose you train a classifier on dogs only and show it a cat.
- What will be the result?



Predicts some class with p_{max}=0.95

- How can that be?
  - – Forced to classify as a dog.
    - If it’s a dog, than most probably a colly
  - – No confidence of the prediction given

A first experiment

- Let’s do the experiment (with our data)
  - We remove a Mitochondria phenotype (cat) from the training set
  - Train the classifier w/o Mitochondria
  - Show Mitochondria (from validation set) to the trained classifier
    - It should tell you that it is unsure

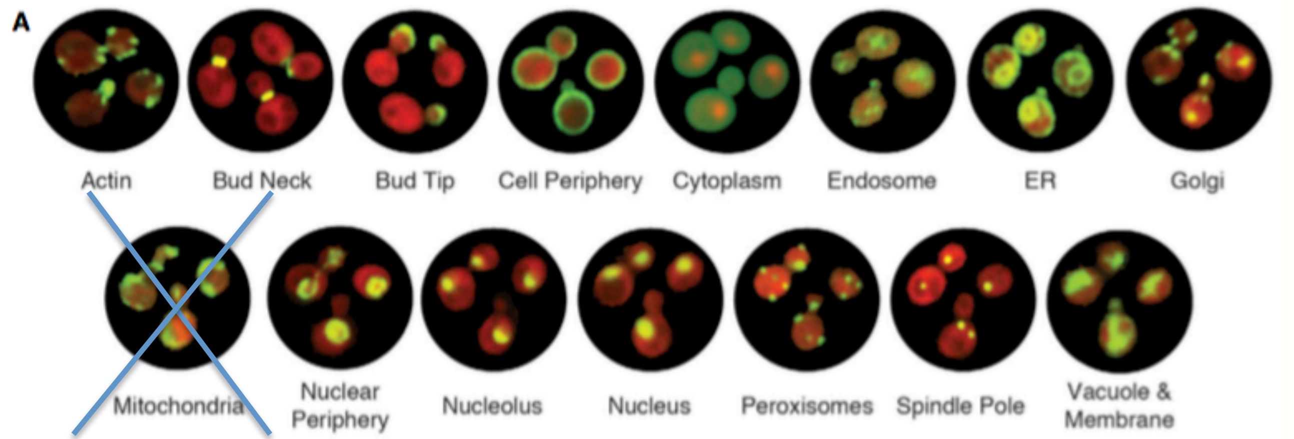
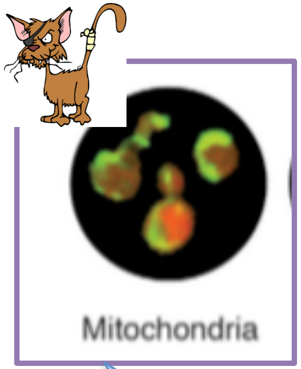
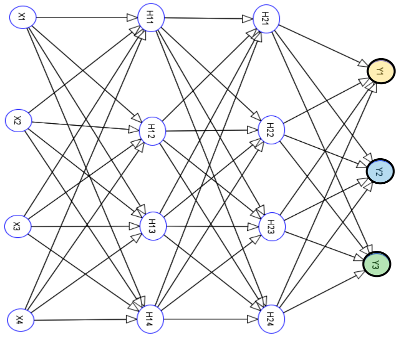


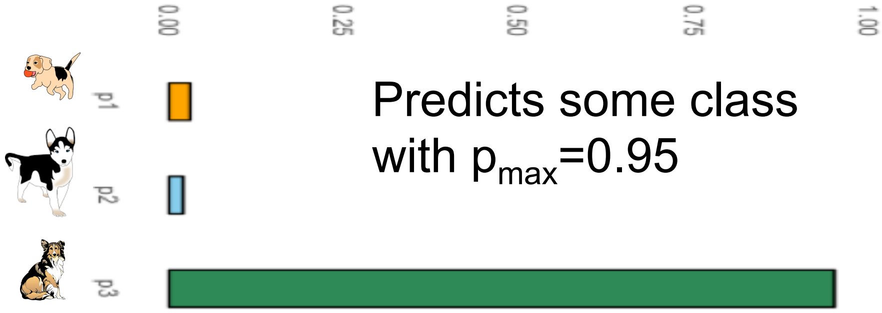
Image credit Kraus, O.Z. et all. , B.J. Molecular Systems Biology 13.4 (2017): 924

Results for the removed class



No mitochondria in training.



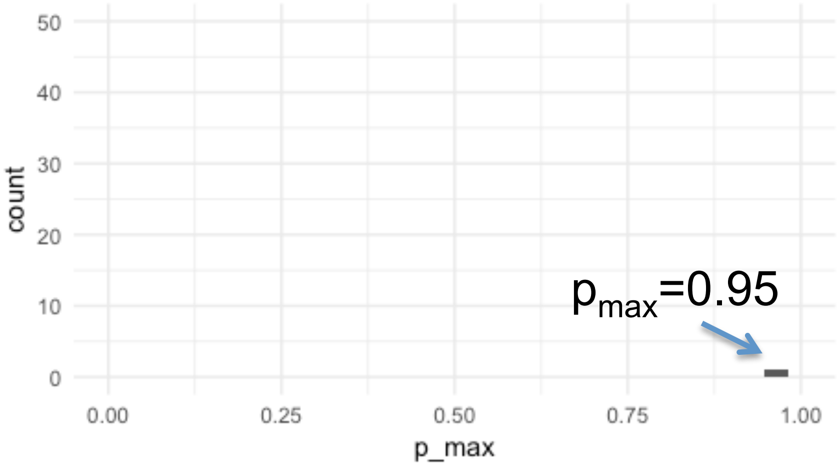


Predicts some class with p_{max}=0.95

Repeat this for all 620 Mito. cells in the validation set

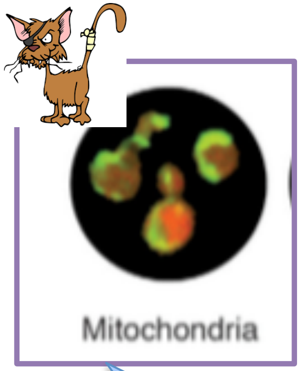
106 of 620 cells are from a class scored with over 90%!

Not enough to have point estimates, we want error bars…

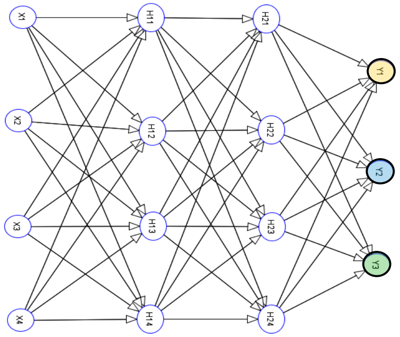


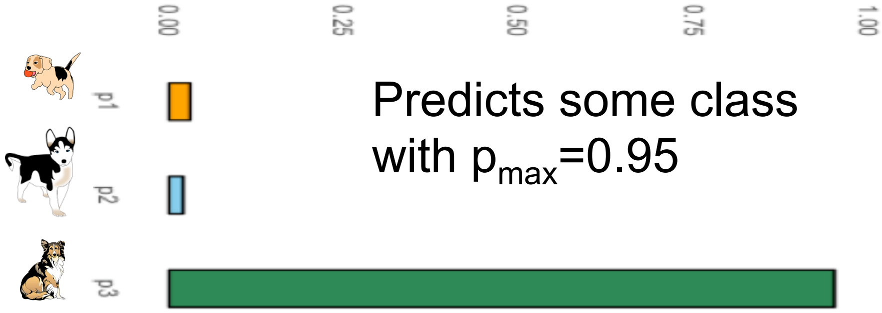
p_{max}=0.95

Results for the removed class



No mitochondria in training.



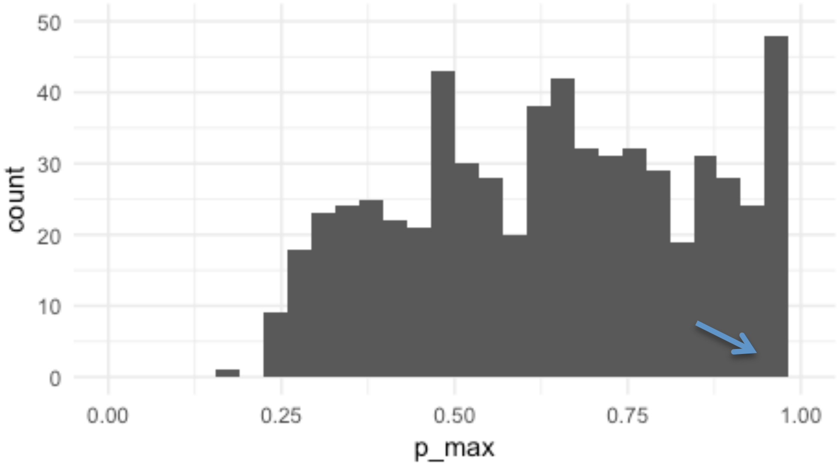


Predicts some class with p_{max}=0.95

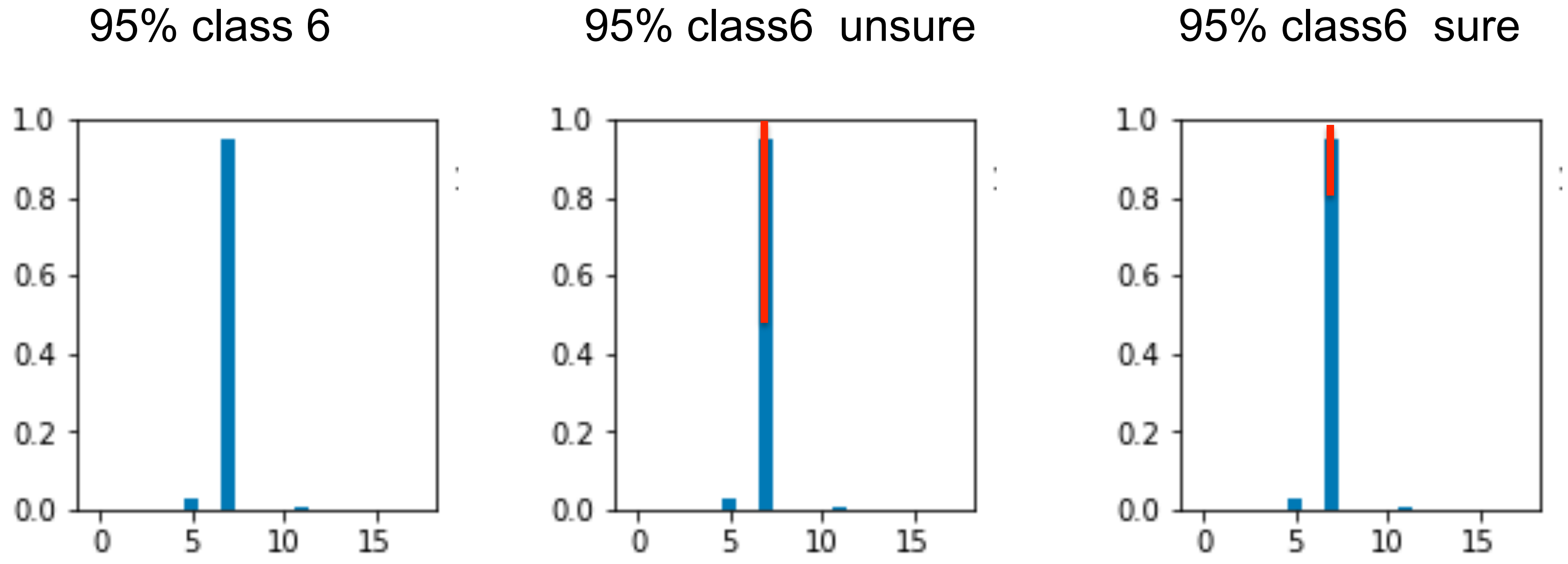
Repeat this for all 620 Mito. cells in the validation set

106 of 620 cells are from a class scored with over 90%!

Not enough to have point estimates, we want error bars…



We want error bars (or even better a distribution)



95% class 6

95% class6 unsure

95% class6 sure

How to get error bars?

What would an experimenter do?

- Go in lab and repeat!

What would a kaggle script kid do?

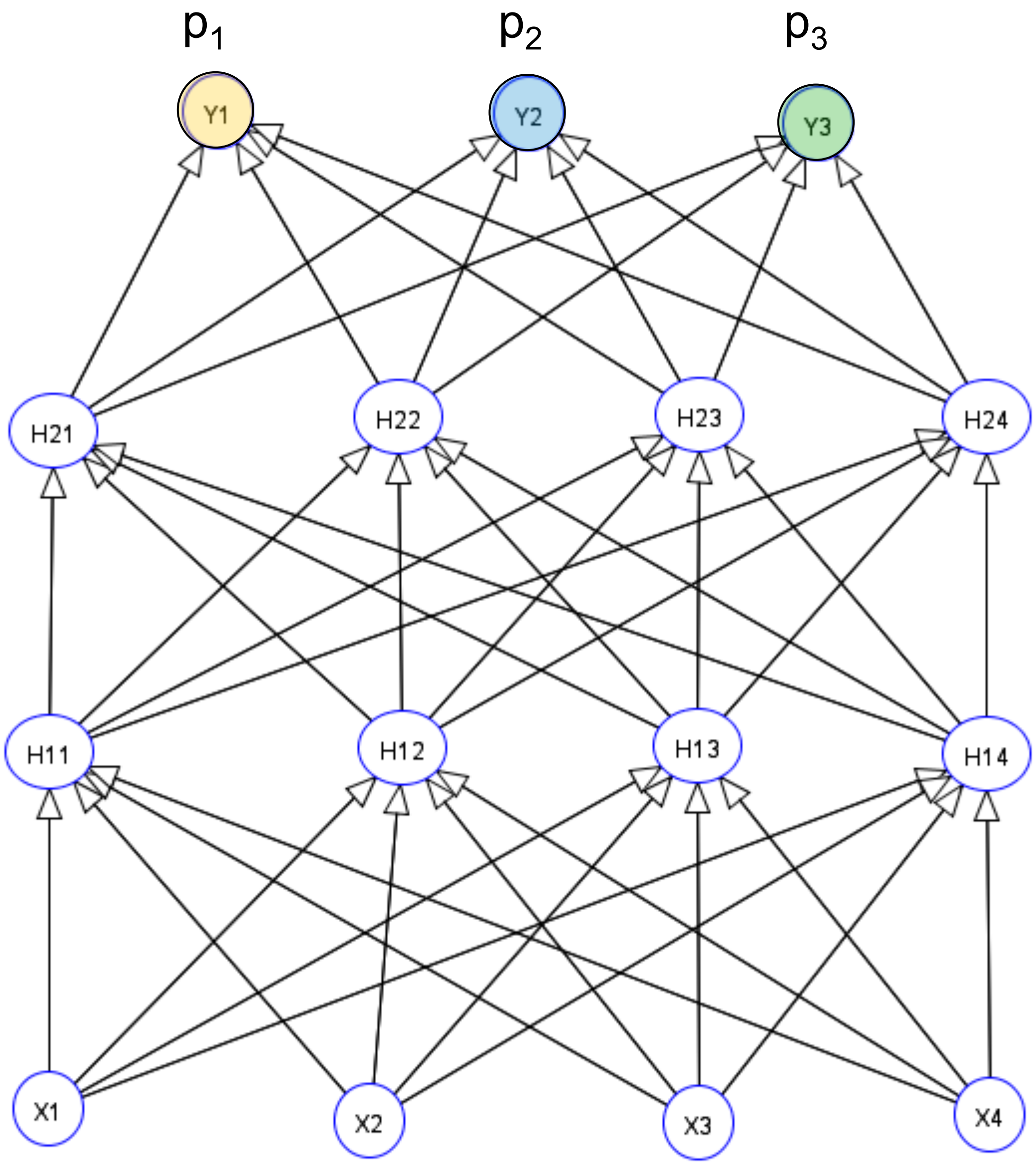
- Simply spin up 100 AWS instances and repeat (train and predict)

What would a computer scientist / statistician do?

- …

…Remember Dropout?

output: probabilities for each class



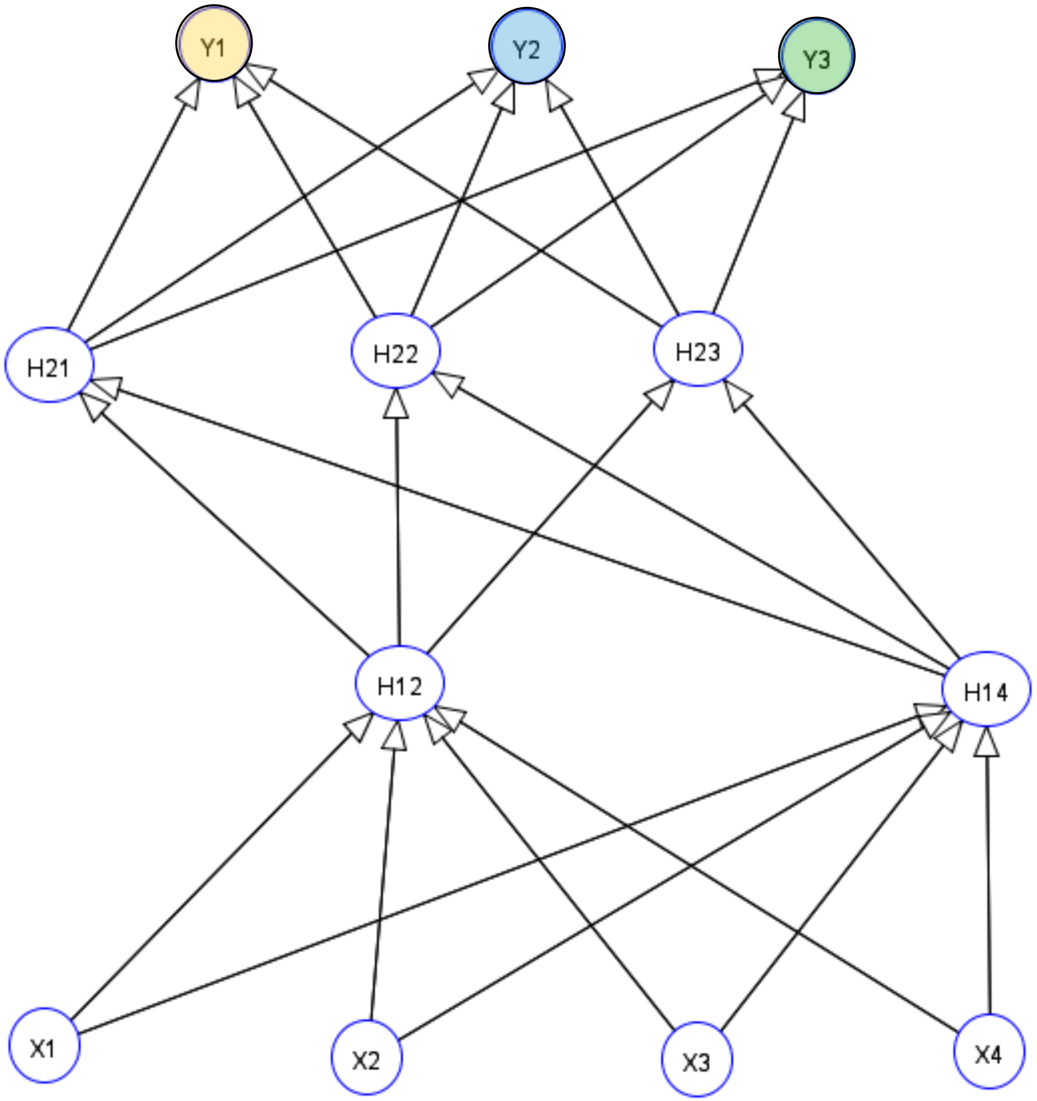
Done in training anyway

input: image pixel values

Use dropout also during testing

RUN 1

p_1=0.08 **p_2=0.89** p_3=0.03



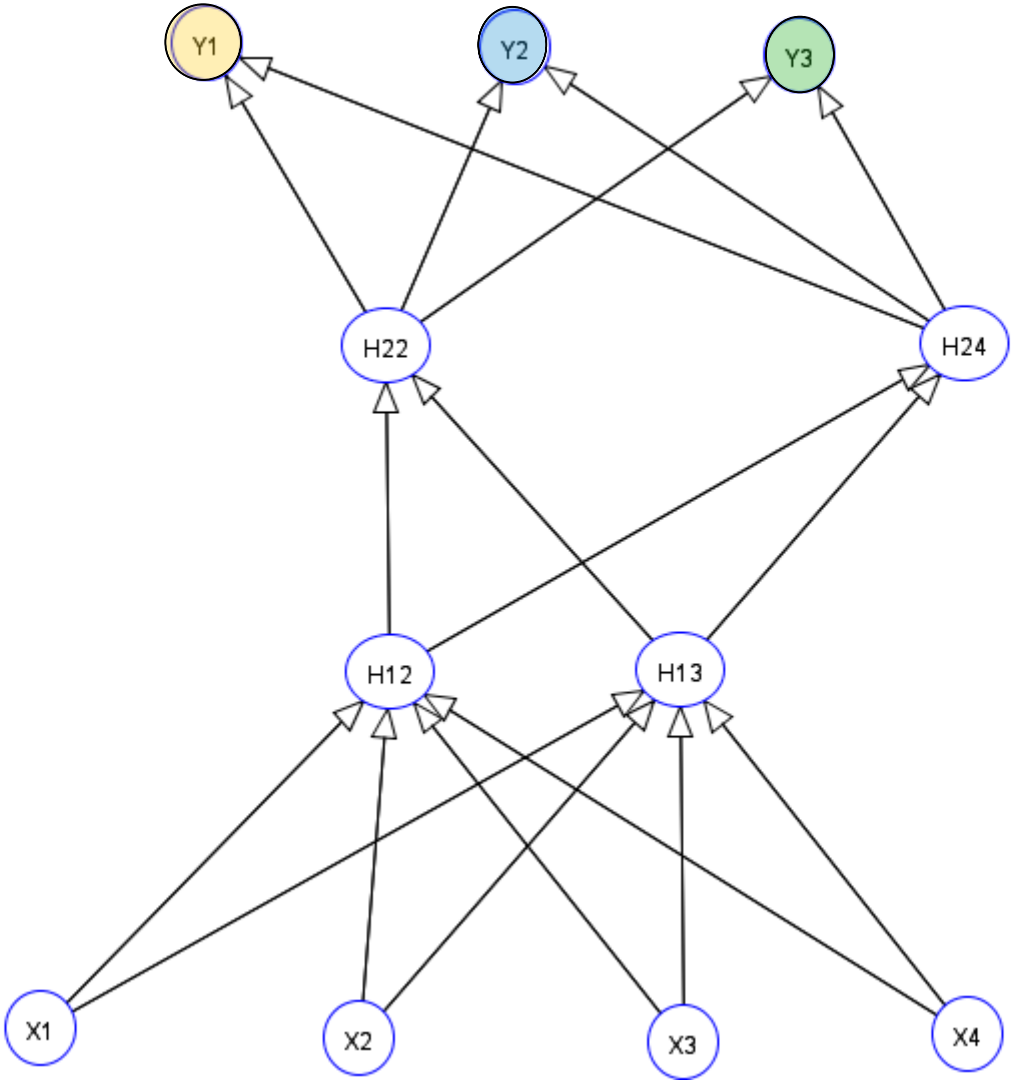
output depends on dropout

stochastic dropout of units

same input

RUN 2

p_1=0.11 **p_2=0.81** p_3=0.08



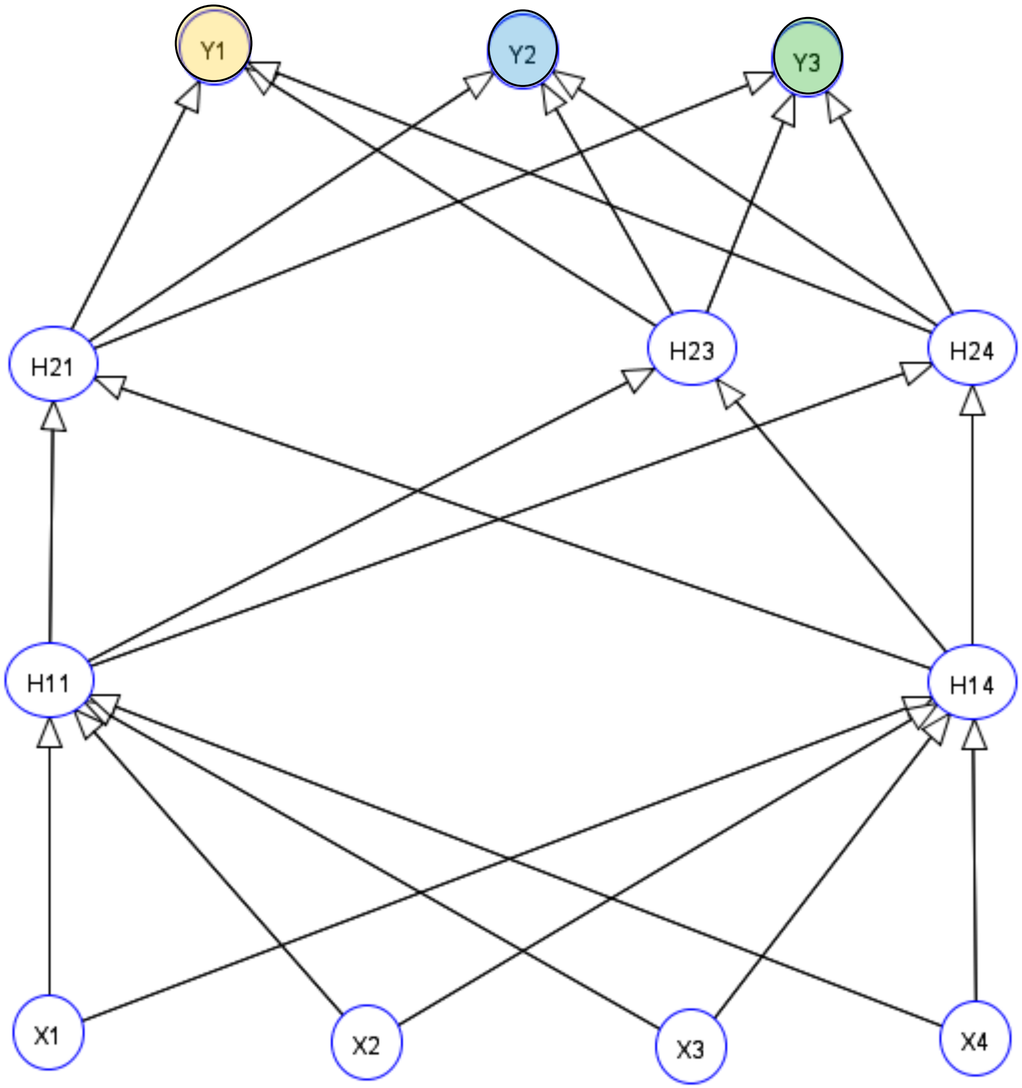
output depends on dropout

stochastic dropout of units

same input

RUN 3

p_1=0.03 **p_2=0.94** p_3=0.03



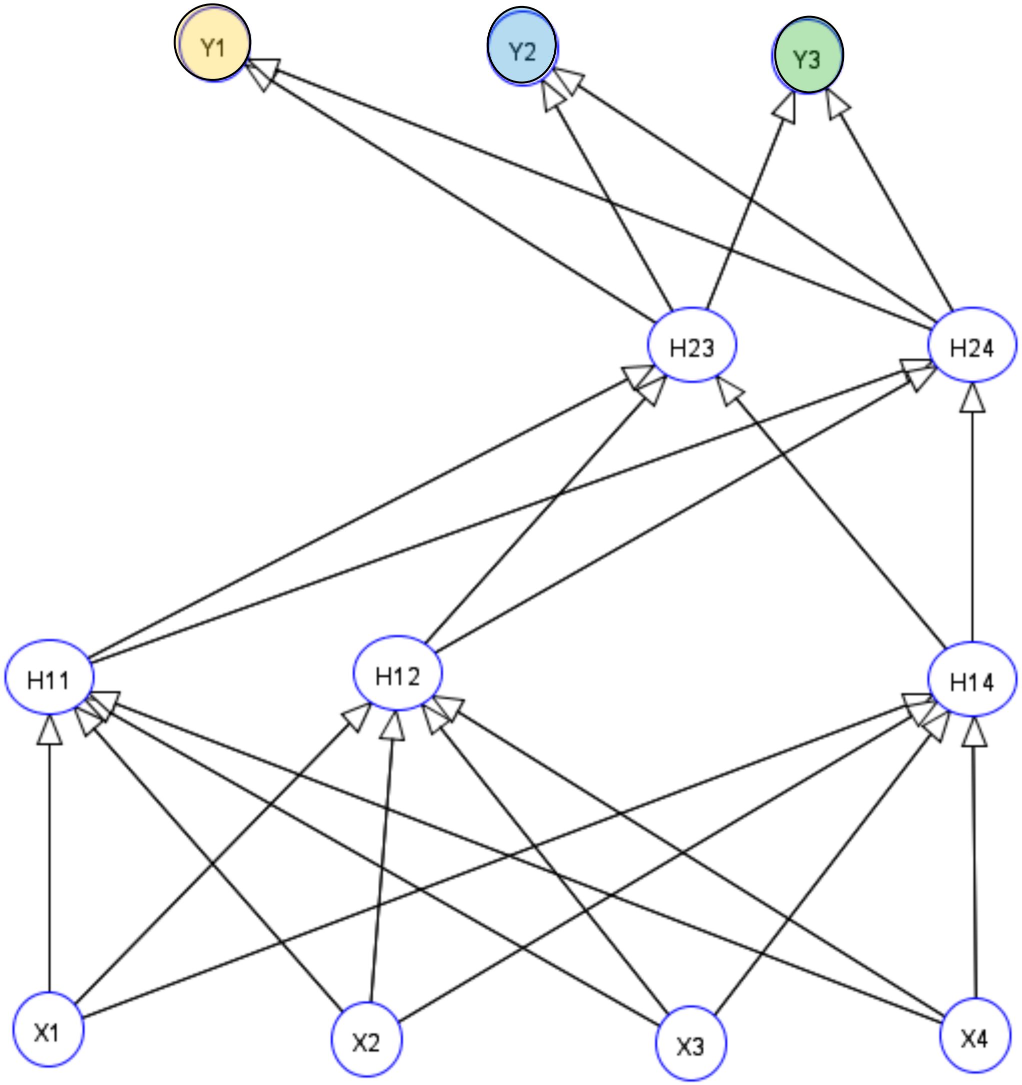
output depends on dropout

stochastic dropout of units

same input

RUN 4

p_1=0.16 **p_2=0.78** p_3=0.06



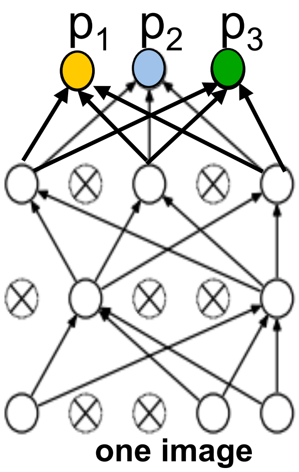
output depends on dropout

stochastic dropout of units

same input

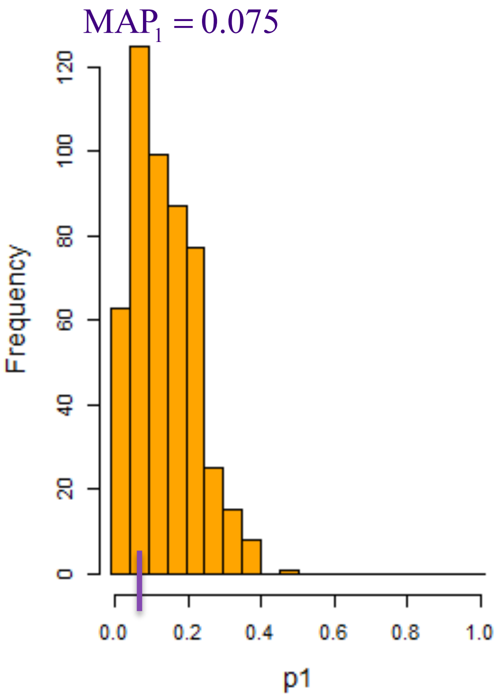
…Repeat 1000 times

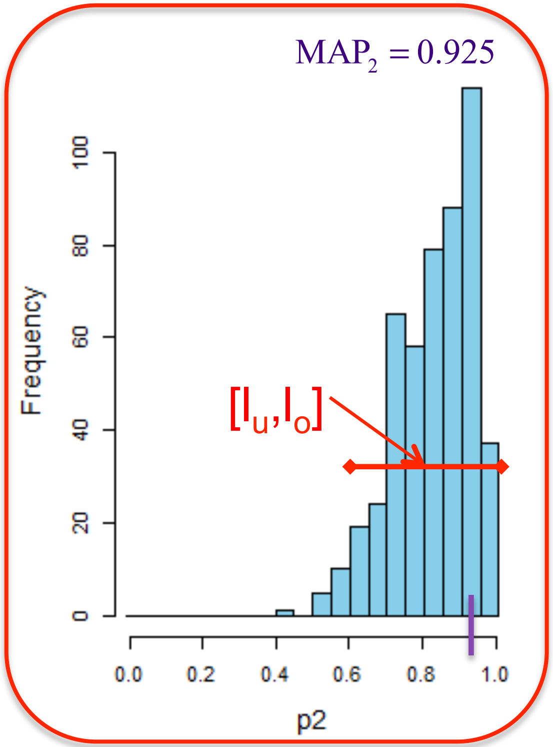
Distributions of predicted probabilities by dropout during test time

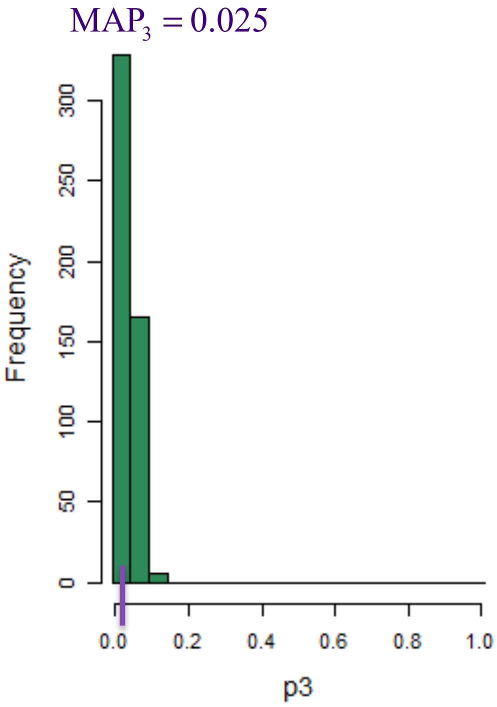


The class with highest probability at modus (MAP) chosen as predicted class.

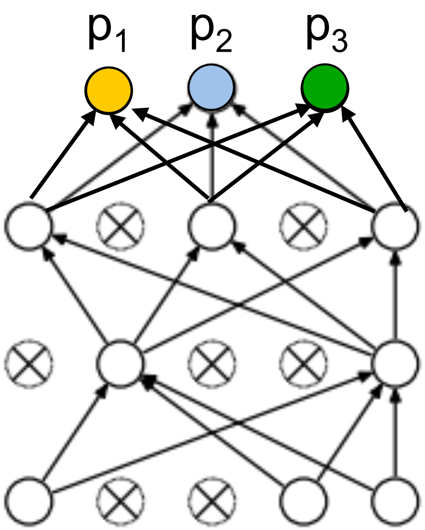
Use 66% CI $[l_u,l_o]$ around MAP for confidence of the predicted probability.





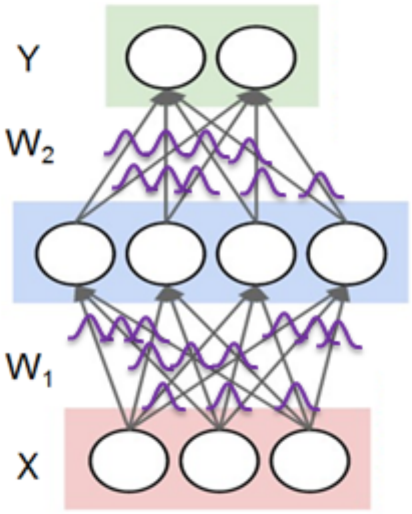


Does this really make sense?



Equivalence

Yarin Gal* (2015)



MC-Dropout
At each training and testing step we remove random nodes with a probability p

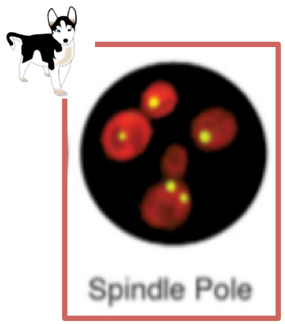
Bayesian Neural Networks

- Provides predictive probability distribution.

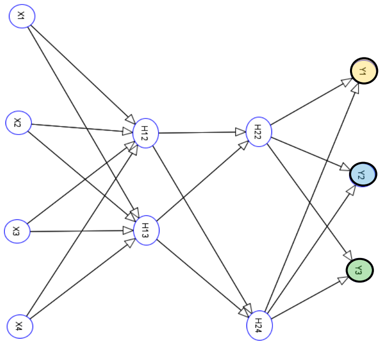
Get new experiments by simply doing dropout, also at testing.

*Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning https://arxiv.org/abs/1506.02142

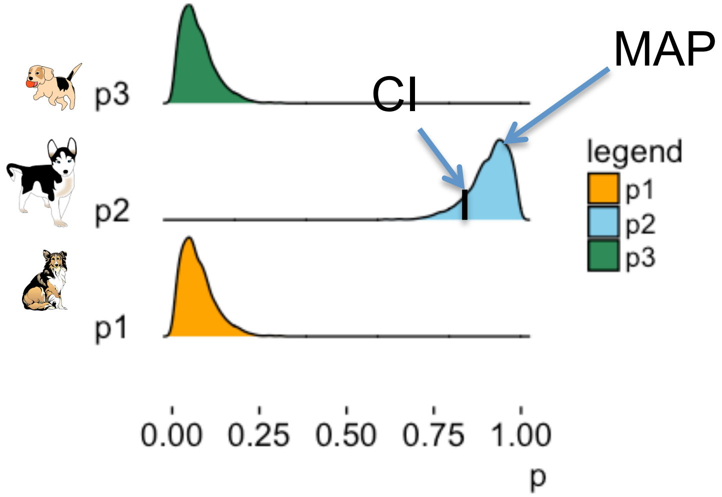
Phenotype in training

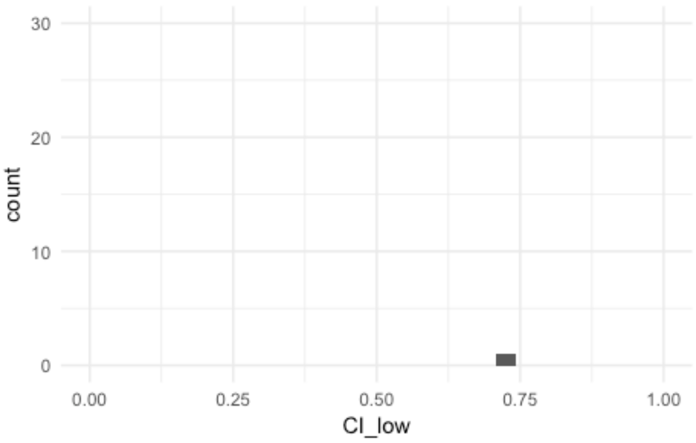


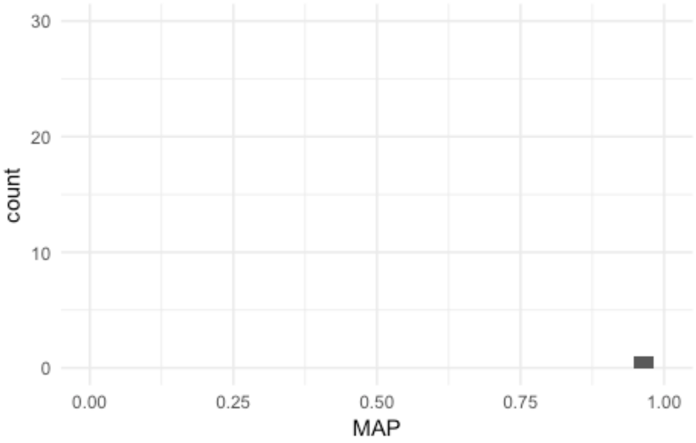
This phenotype is in training!



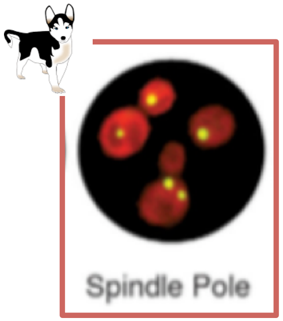
Many Dropout Runs



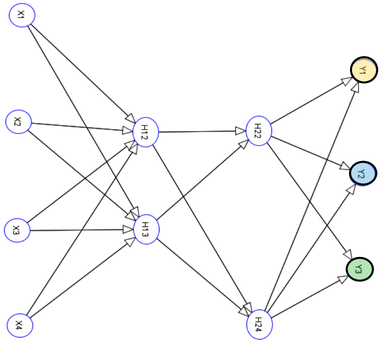




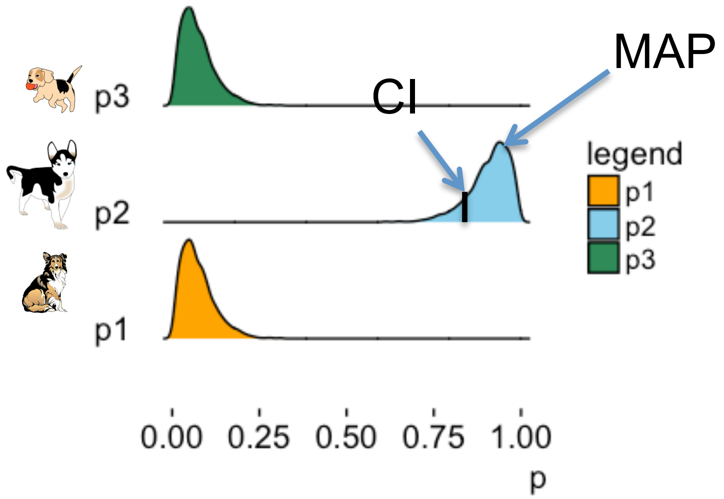
Phenotype in training



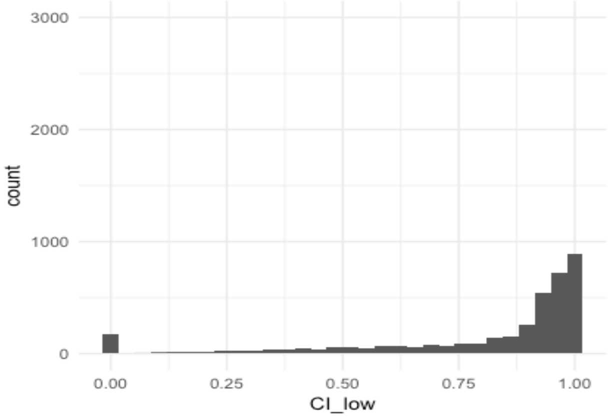
This phenotype is in training!

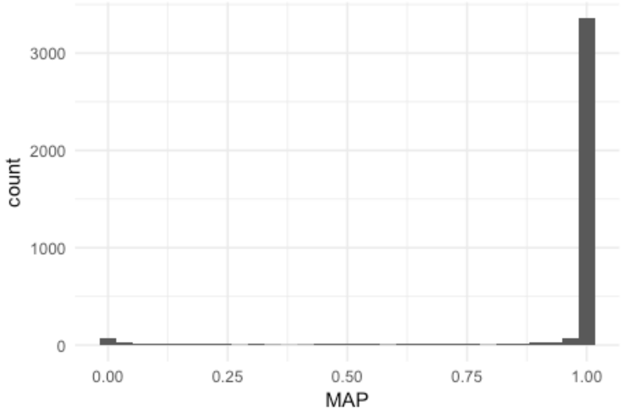


Many Dropout Runs

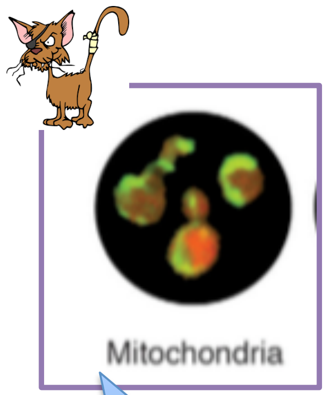


Repeat this for all 3849 non mitochondria cells in the validation set

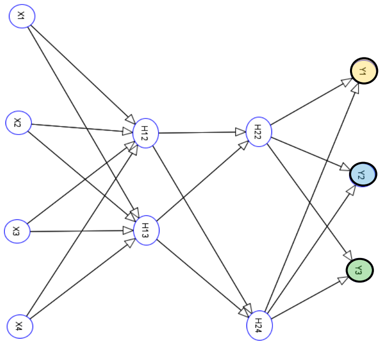




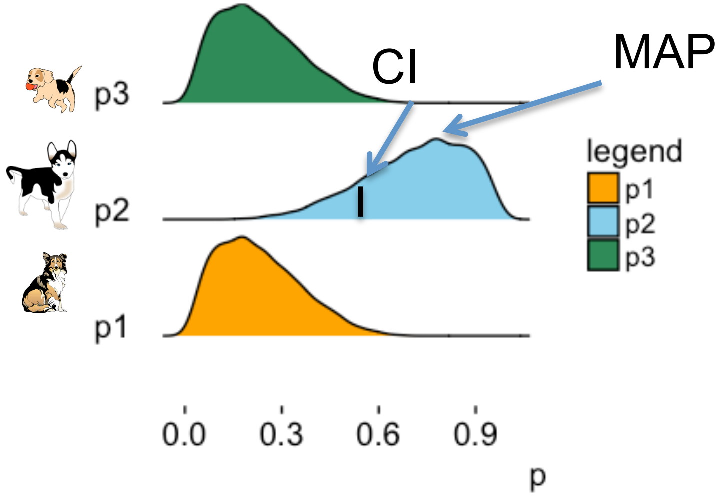
Phenotype not in training



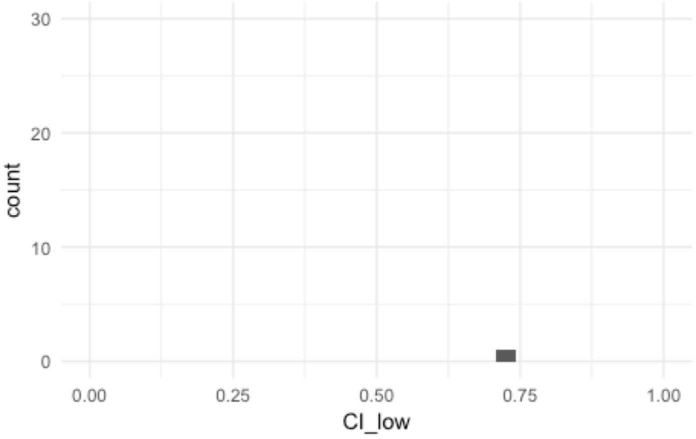
This phenotype is not in training!

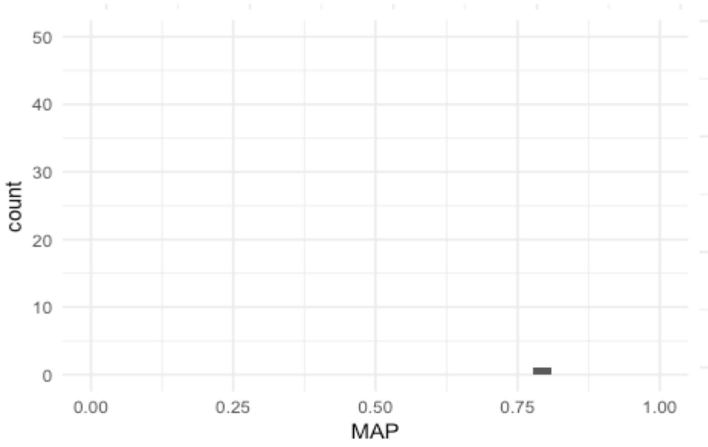


Many Dropout Runs

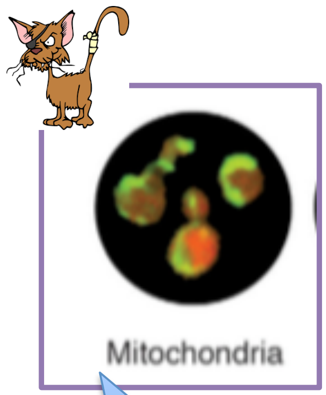


Repeat this for all 620 mitochondria cells in the validation set

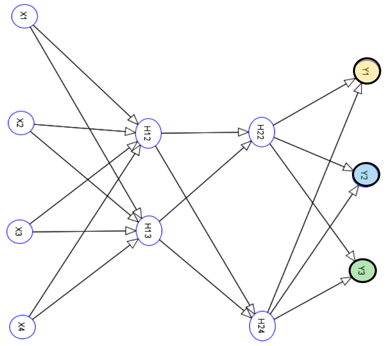




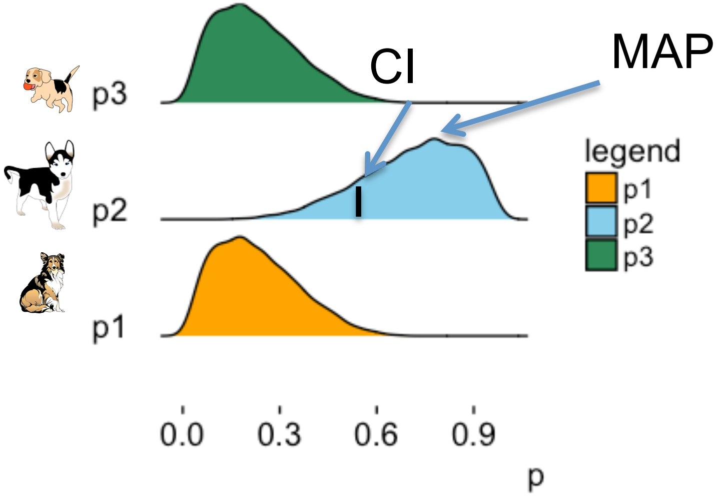
Phenotype not in training



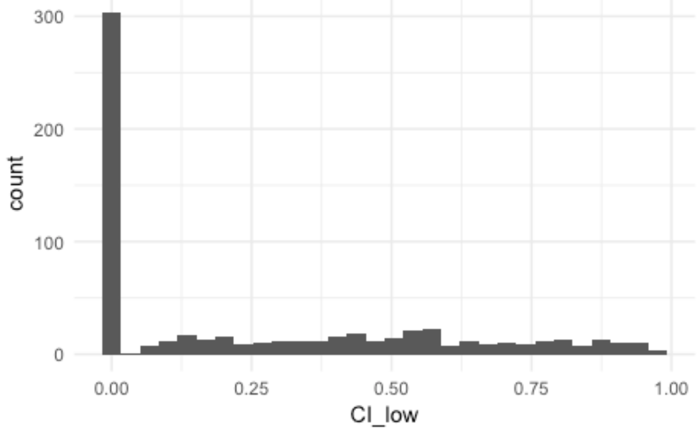
This phenotype is not in training!

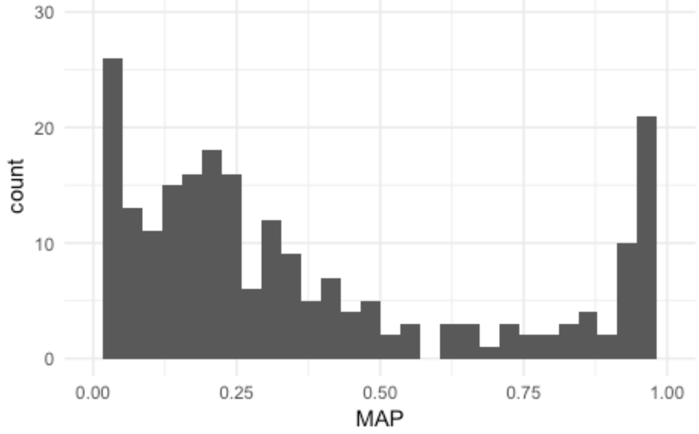


Many Dropout Runs

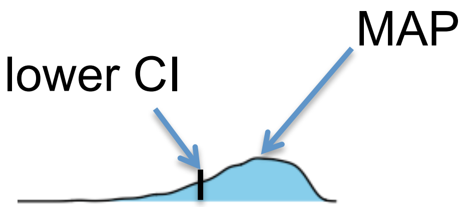


Repeat this for all 620 mitochondria cells in the validation set

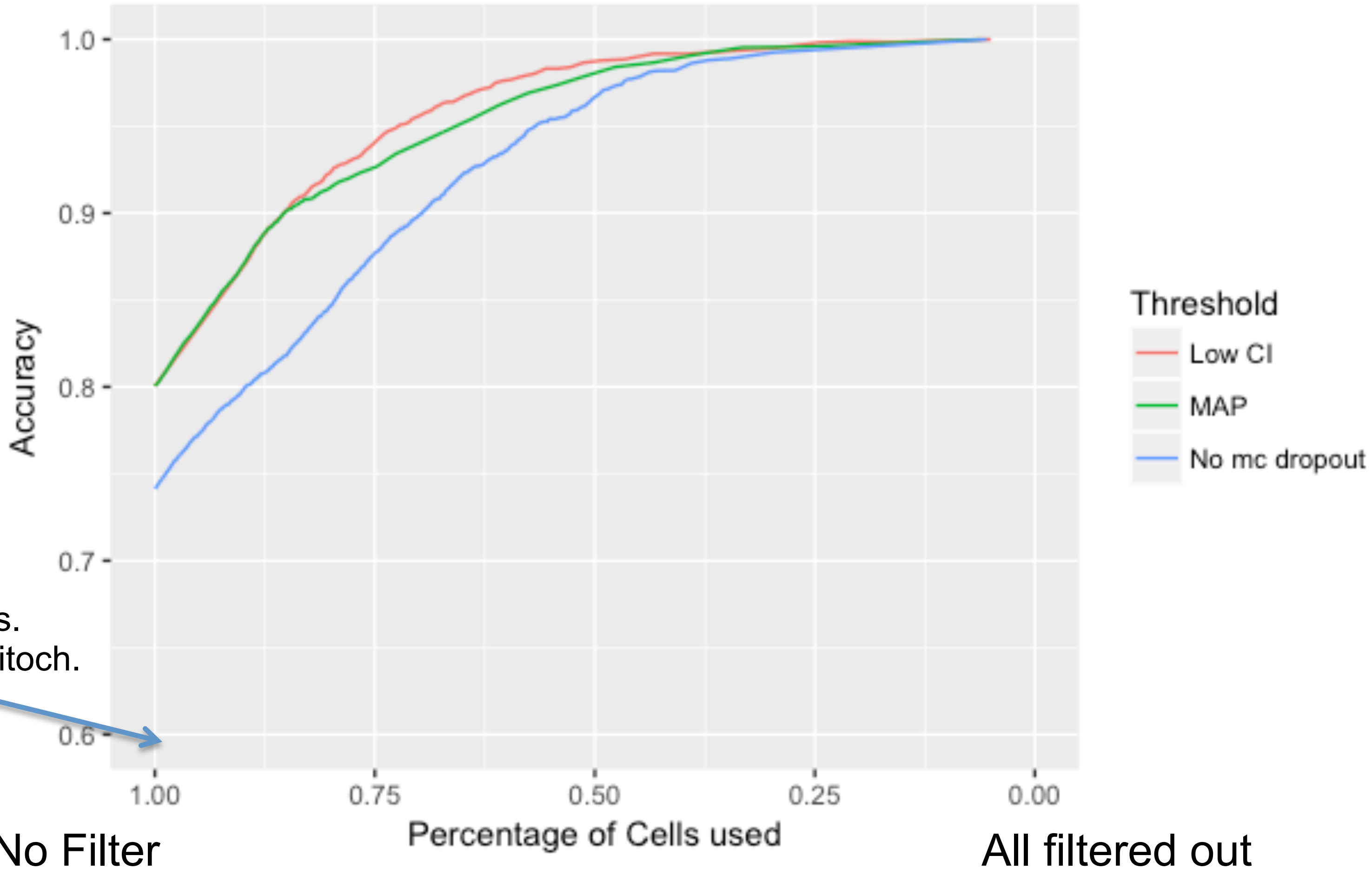




Comparison



- Use MC-Dropout: the lower CI as filter
- Use MC-Dropout: MAP as filter
- Use a traditional approach and maximal value of p as filter



642 Mitoch.s.
3843 non Mitoch.

No Filter

All filtered out

Conclusion

- Deep Learning works for single cell phenotype classification (at least for the assays seen)
  - No hand crafting of features needed

- Dropout in forward pass can be used quantify model uncertainty (basically for free) and boosts performance

Thank you!

Zürcher Hochschule
für Angewandte Wissenschaften

zh
aw

Elvis Murina

Vasily Tolkachev

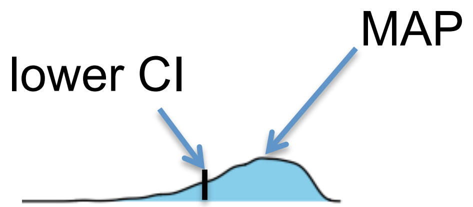
Beate Sick

This work has been partly funded by the CTI grant: “DeepCells”

Genedata
solutions in silico

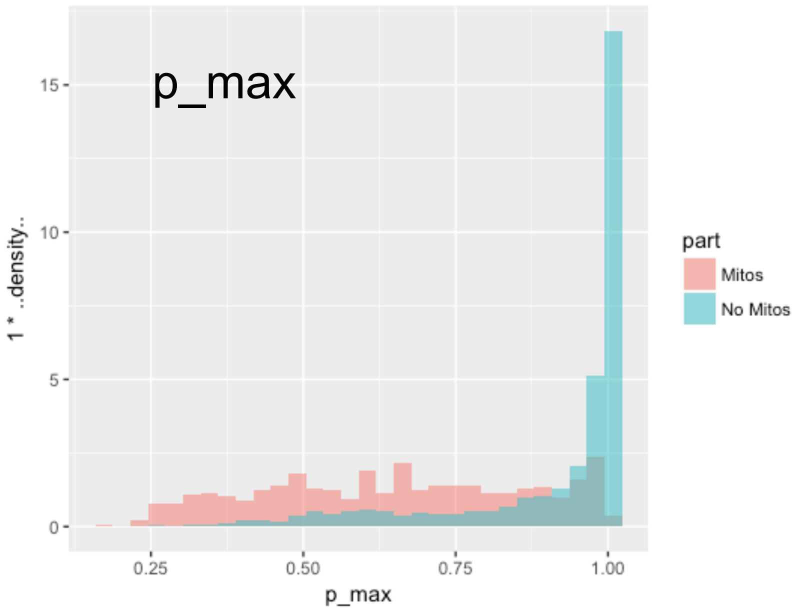
Backup

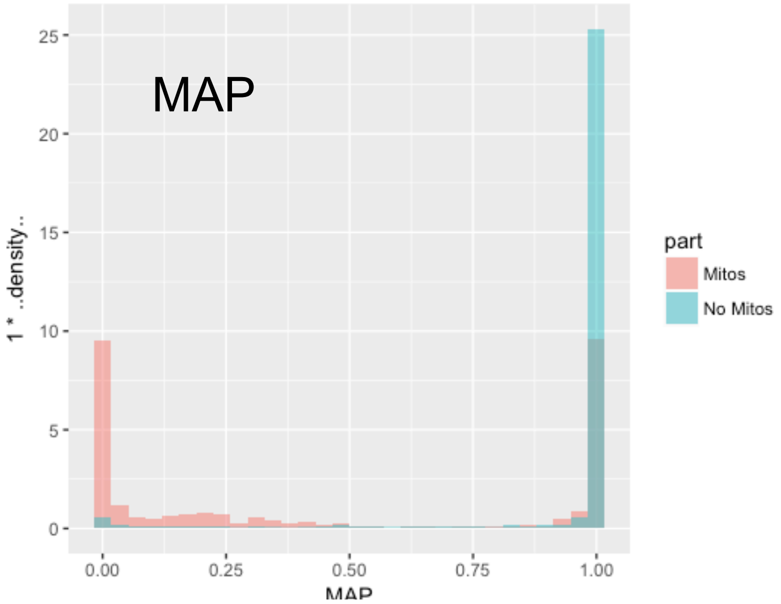
Comparison

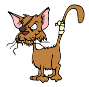


No MC Dropout

MC Dropout

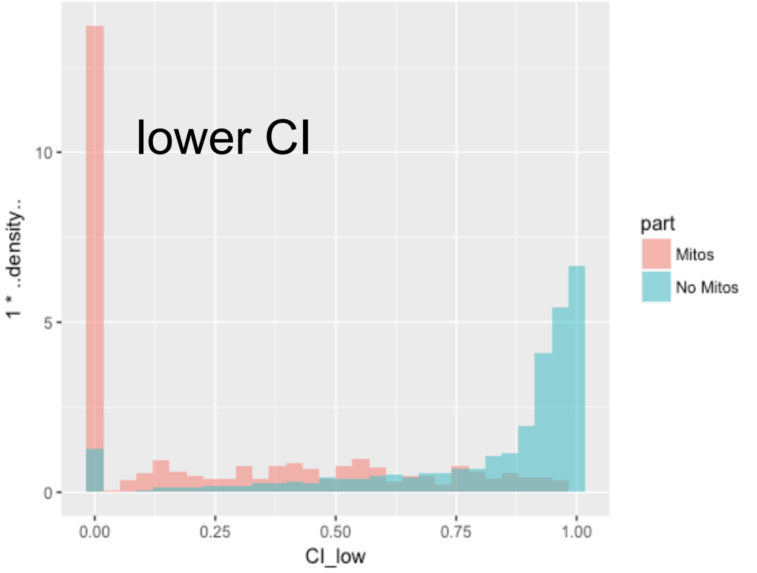






🟥 Mitos (not in training)

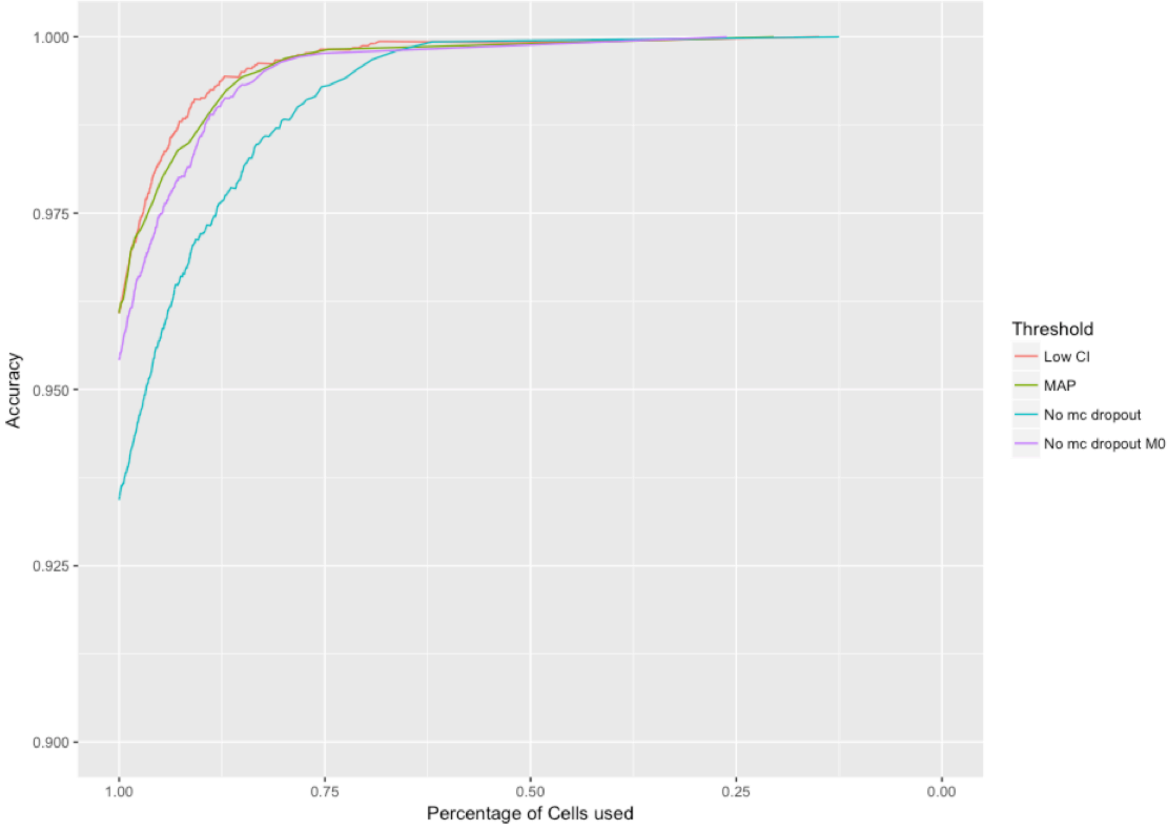
🟦 No Mitos (all phenotypes in training)



With MC-Dropout, lower CI can be used as filter.

Result on training with all phenotypes

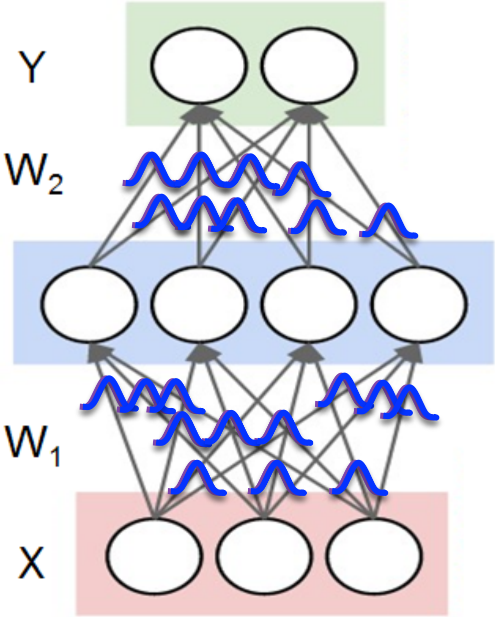
- Now we include mitochondria again in the training



- Doing several forward passes increases performance for free
- Consistent with
  - Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning https://arxiv.org/abs/1506.02142

Idea of Bayesian Network

- Motivation
  - Weights of a network are random (next run other weights)
- Principle Idea
  - Choose these networks as probabilistic models: weights have a distribution (aka Bayesian Neural Network)



Learned in training

$$p(W_1 \mid \mathbf{X}, \mathbf{Y}) \qquad p(\mathbf{Y} \mid W_1, \mathbf{X})$$

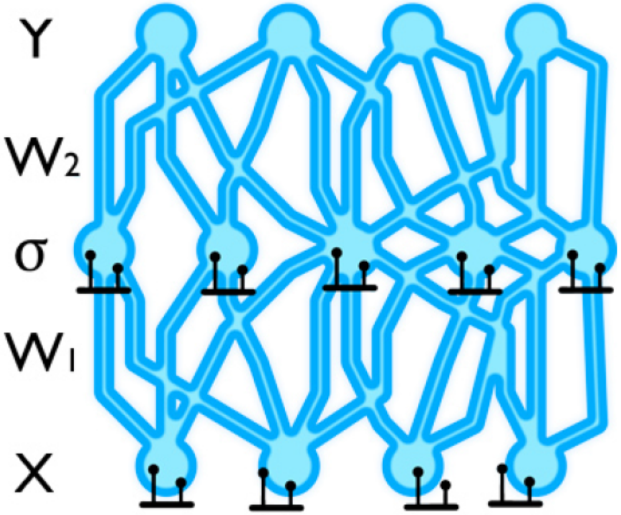
Prediction $p(\mathrm{Y}|\mathrm{X}) = \int p(\mathrm{Y}|\mathrm{X},\mathrm{W}) \cdot p(W)\, dW$

or sample!

Picture from http://mlg.eng.cam.ac.uk/yarin/blog_3d801aa532c1ce.html

Pure man's Bayesian Neural Network

- Bayesian networks are hard to train.
- Dropout can be seen as a (variational approximation) of a simple Bayesian Neural Network



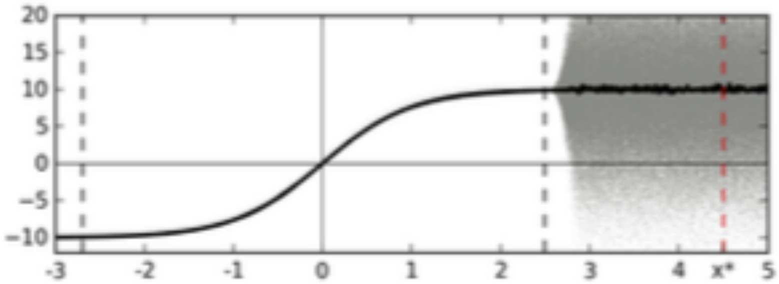
Weights have independent Bernoulli Distributions.

Uncertainty estimates for free (basically)

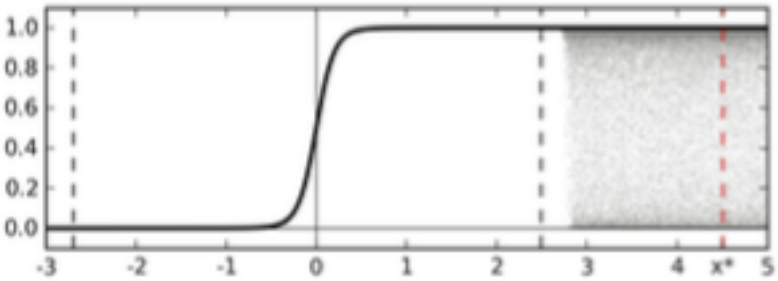
- Estimation of weights: just do standard NN training with dropout
- Sampling: simply keep dropping out nodes *MC-dropout*

For proofs: Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning https://arxiv.org/abs/1506.02142

Don't we have this already?



Softmax *input* as a function of data $\mathbf{x}$

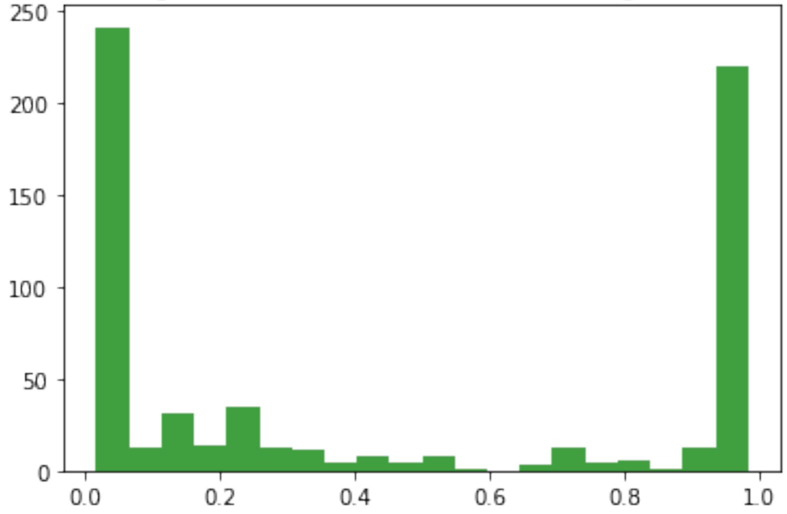


Softmax *output* as a function of data $\mathbf{x}$

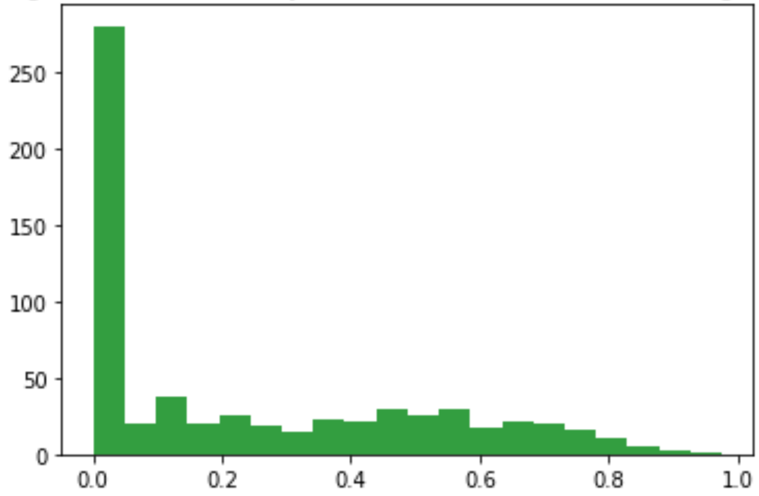
[validation set]

Prediction on mito “cats|

Highest MAPs estimate for non-existing class

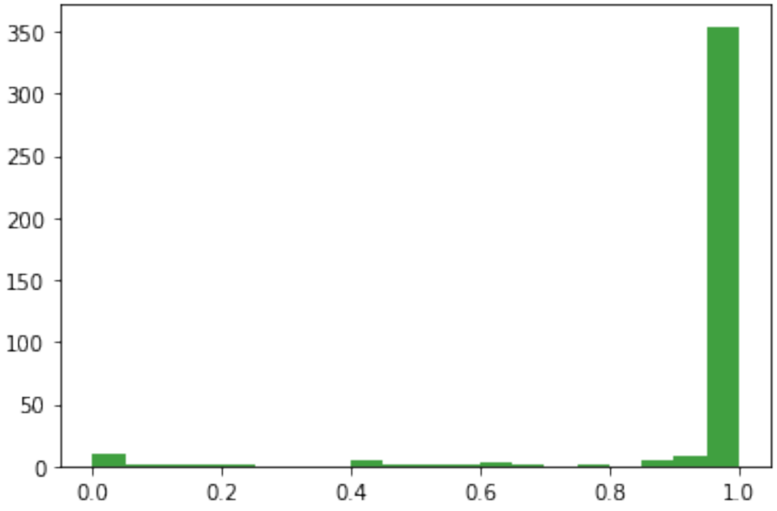


Highest lower credibility intervall estimate for non-existing class

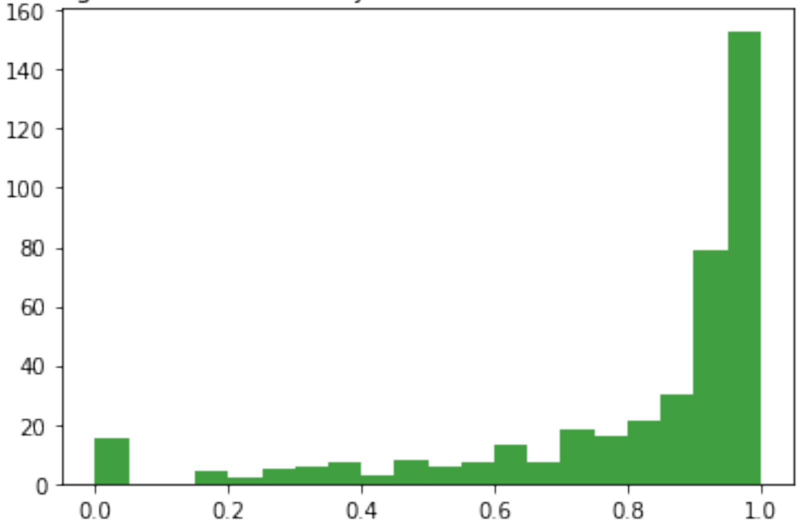


Prediction on cells w/o Mito????

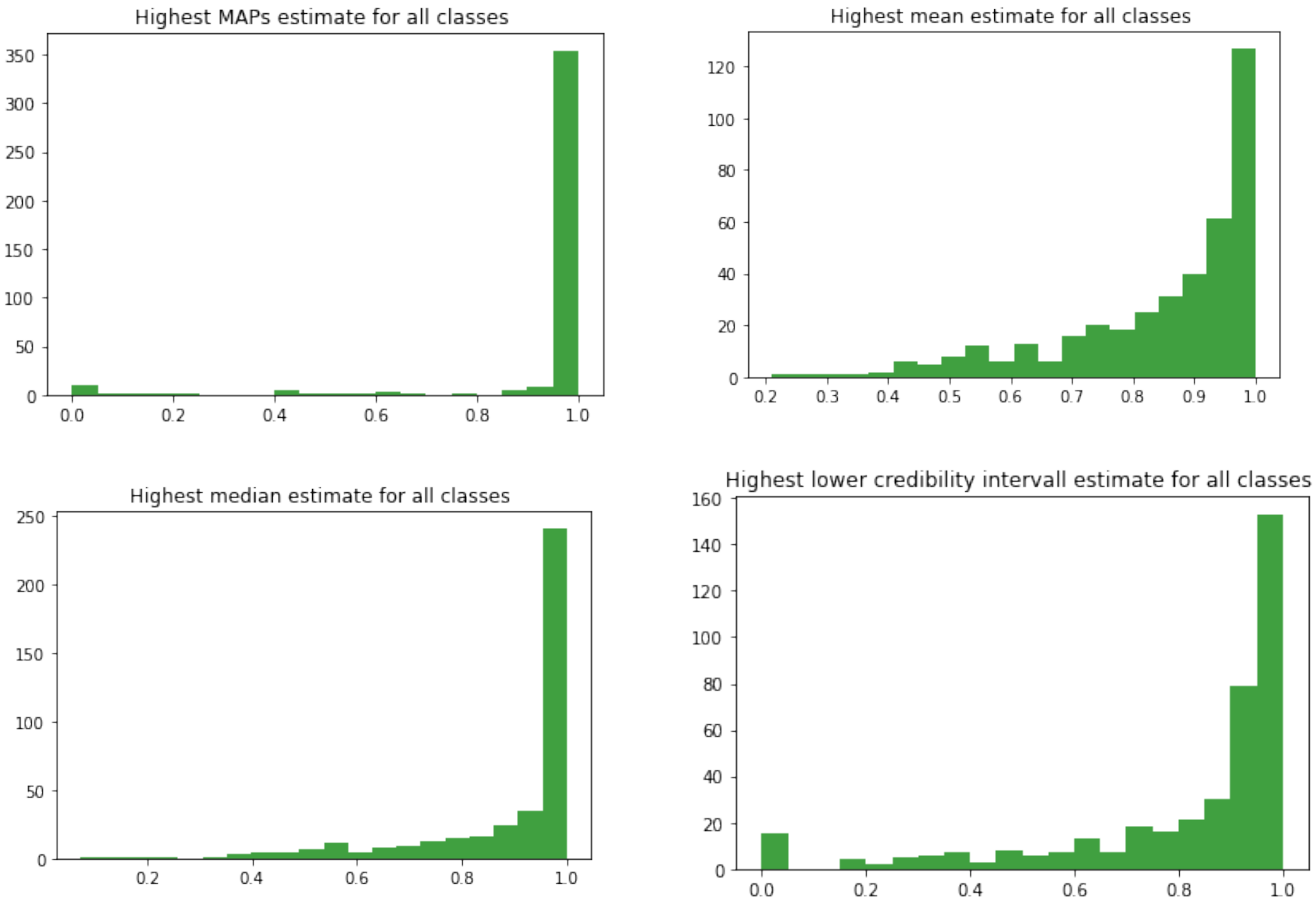
Highest MAPs estimate for all classes



Highest lower credibility intervall estimate for all classes



Validation set w/o non-existing classes



Other methods to reduce the number of labeled data [outlook]

- Metric Embedding “cell2vec”
  - Train a network trained for metric embedding (similar objects are close to each other)
  - Special loss functions
    - Contrastive Loss, Triplet Loss, Center Loss, …
  - State of the art in face recognition
    - Trained on millions (MSCeleb-1M, open) and 100 of millions (closed, source) of examples
  - Issues
    - FaceNet not good for cells (see tSNE)
    - Training needs (too?) much labeled data
      - See Deep Metric Network on HCI (biorxiv.org 2017/07/10/161422)

- Semi-supervised learning
  - Ladder Network (still working on it)
  - GAN, VAE

- **Question: Is labeling 200 cells per class so bad after all?**

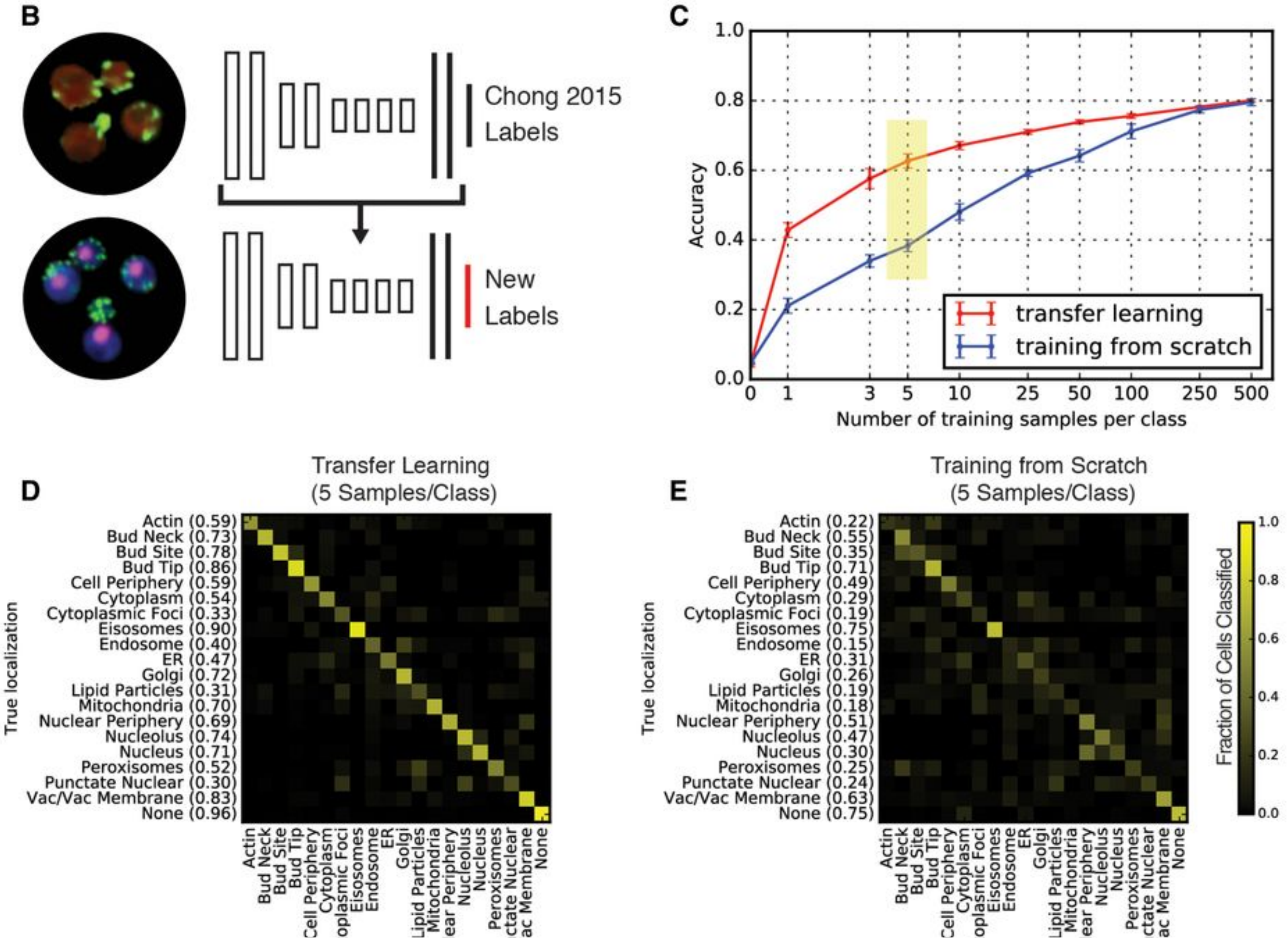
Approaches to reduce the number of labeled data

- Transfer learning [needs trained network]
  - Train a network on a similar dataset
  - Fine-tune this network new dataset

Nice Figure

- Label propagation
  - Use network to predict unlabeled cells, and then use those

Transfer learning

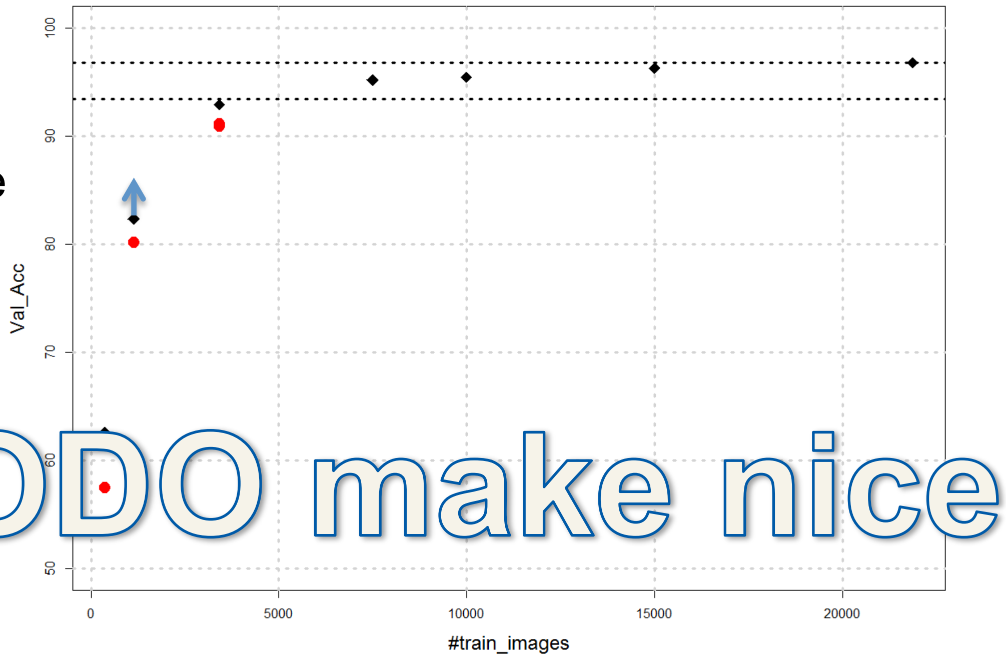


Oren Z Kraus et al. Mol Syst Biol 2017;13:924

Label propagation

- Train classifier on 1140 random cells from training set➔ 81% validation accuracy
- Apply classifier again on training set
- Take best 40% best predictions of each class (4143 new pseudo labeled cells)
- Train network again including pseudo labeled cells →**86%**

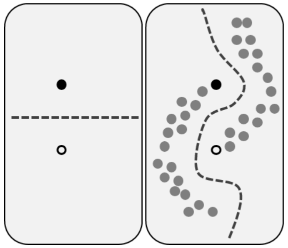
5% for free



TODO make nicer

Other methods to reduce the number of labeled data

- Transfer learning
  - Use a network trained on a similar task and only retrain the last few layers. For HCS done in (Kraus et. al. 2017)
- Metric Embedding “cell2vec”
  - Train a network trained for metric embedding (similar objects are close to each other).
  - State of the art in face recognition
    - Trained on millions (MSCeleb-1M, open source) and 100 of millions (closed source) of examples
  - Issues
    - Training needs (too?) much labeled data
      - See Deep Metric Network on HCI (biorxiv.org 2017/07/10/161422)
- Semi-supervised learning
  - Ladder Network (still working on it)
  - GAN, VAE

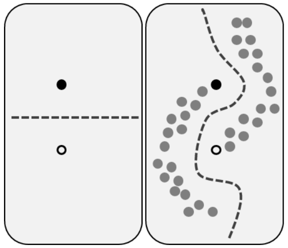


Question: Is labeling 200 cells per class so bad after all?

Other methods to reduce the number of labeled data

- Transfer learning
  - Use a network trained on a similar task and only retrain the last few layers. For HCS done in (Kraus et. al. 2017)

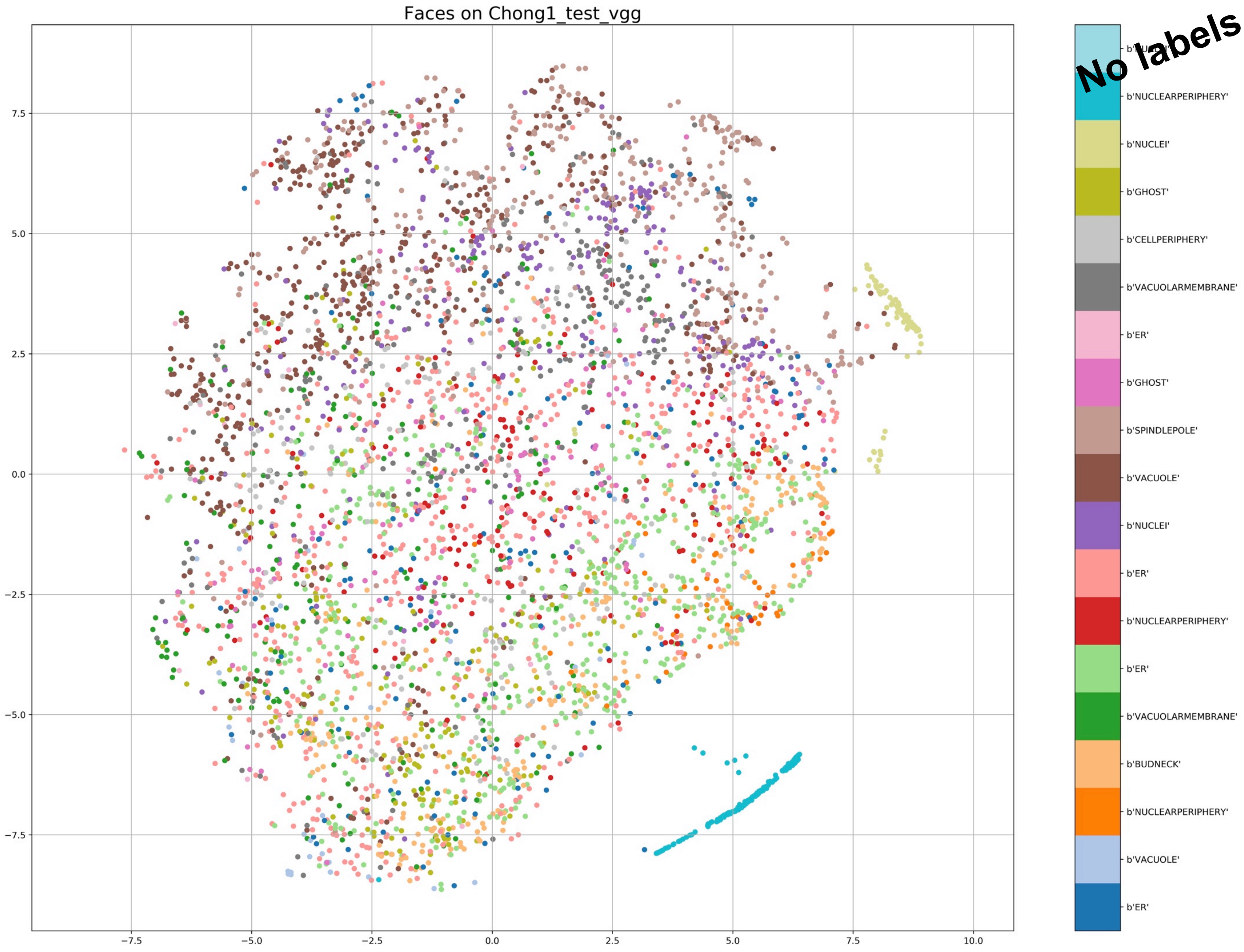
- Semi-supervised learning
  - Ladder Network (still working on it)
  - GAN, VAE



Question: Is labeling 200 cells per class so bad after all?

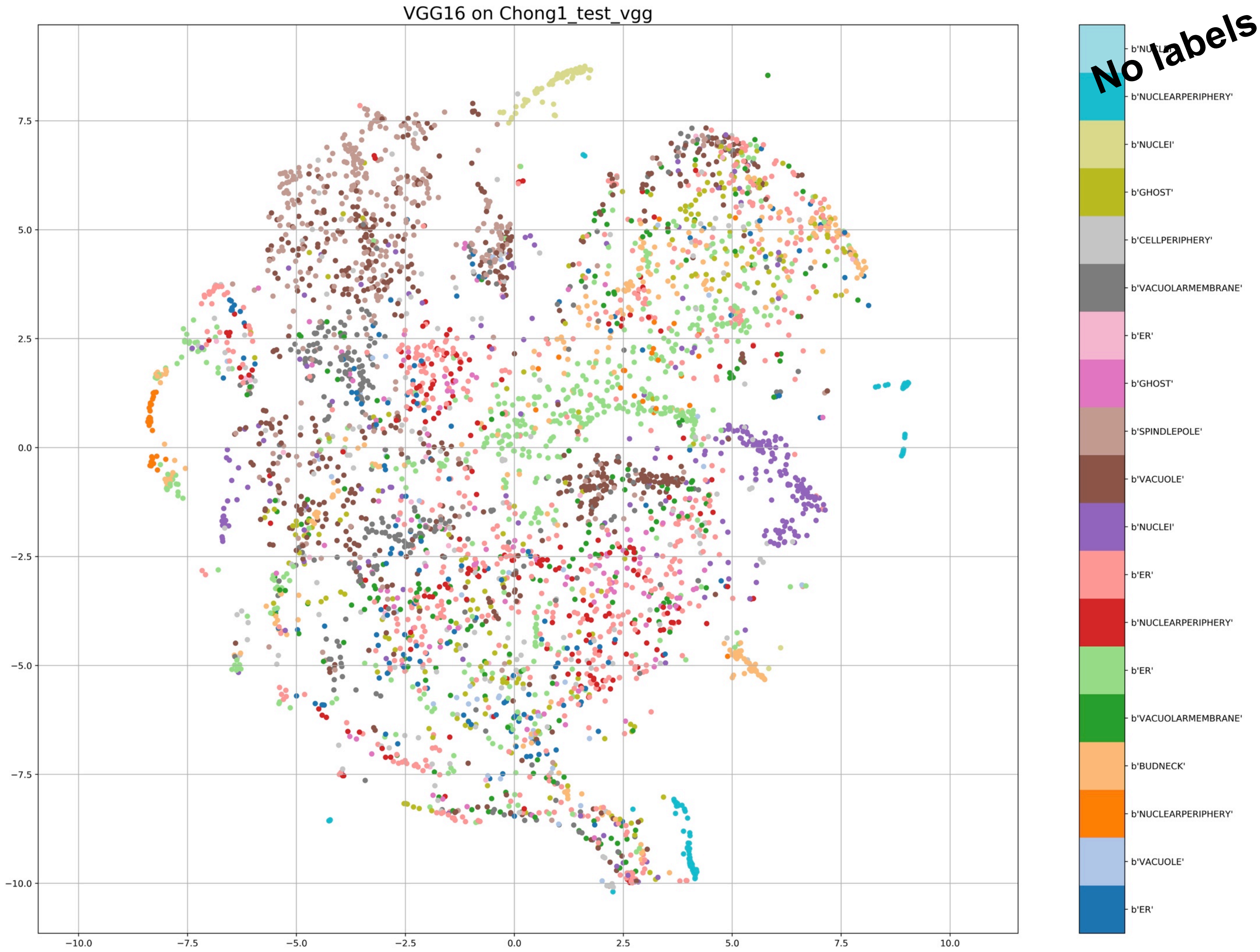
TSNE with inception trained on faces (center loss)

No labels needed!



KNN accuracy 47% (learned with 4000 examples). Model from: https://github.com/davidsandberg/facenet/

TSNE with VGG16 trained on ImageNet



No labels needed!

KNN accuracy 60% (learned with 4000 examples). Model from keras