Artificial Intelligence und Deep Learning in SAS Viya - Ein Überblick über die Möglichkeiten

Gerhard Svolba, SAS

THE POWER TO KNOW.

Take Aways

- SAS provides at lot of AI functionality: object recognition, image mining, text analytics, deep learning, recurrent NN, convolutional NN, ...

- These methods are executed „distributed“, „in-memory“ on the CAS Server in SAS Viya.

- You can use SAS procedures, SAS Actions and Actionsets (CAS-L, Python, R, ...) and the Deep Learning Tookit from SAS

Artificial Intelligence

is the science of training systems to emulate human tasks through Learning and Automation

Understand Context

Learn Patterns

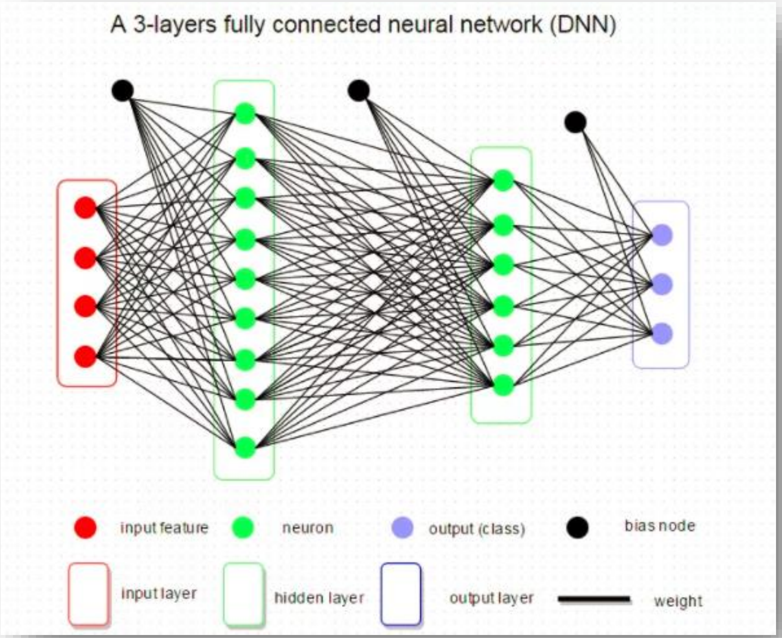
Recognize Objects

sas

Learning	Automation	Benefit
Images	Is this you?	More Secure
Transactions	Is it fraud?	Lower Risk
Users	Will they buy?	Higher Return
Medical Images	Is this healthy?	Better Outcome
Languages	Translate?	Reduced Cost
Emails	Is it spam?	Better Experience

Deep Learning

A specialization of machine learning



- Neural networks with many layers and different types of …
  - Activation functions
  - Network architectures
  - Sophisticated optimization routines

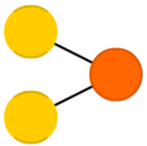
Each layer represents an optimally weighted, non-linear combination of the inputs.

- Automatic feature generation
- Extremely accurate results if well-trained; use for classification, prediction, or pattern recognition, especially in unstructured data

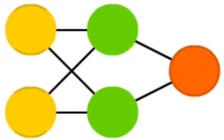
Image from: https://www.gabormelli.com/RKB/Multi_Hidden-Layer_Neural_Network

SAS Neural Network Types

Perceptron

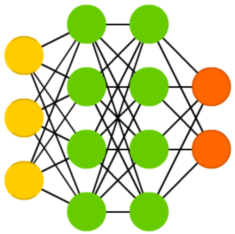


Feed Forward Networks (FF)

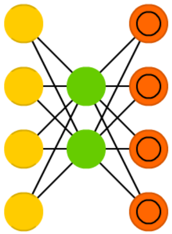


SAS Deep Learning Architecture Types

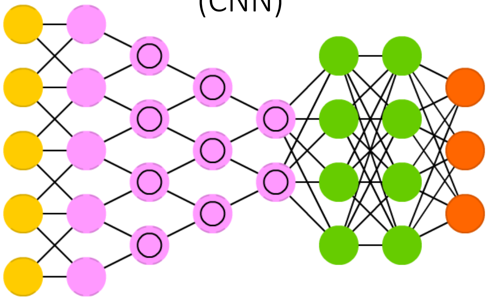
Deep FF Neural Network (DNN)



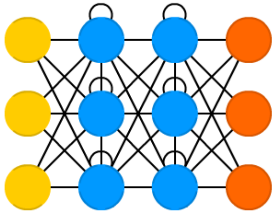
Auto Encoder* (AE)



Convolutional Neural Networks (CNN)



Recurrent Neural Networks (RNN)



(Pictures taken from http://www.asimovinstitute.org/neural-network-zoo/)


29- 5-02 3A
16:31:26
05-28-02 16:30:17
CAMERA 50

Robust Principal Component Analysis

RPCA Procedure in SAS Viya

Low Rank Matrix

Obs	index	X	Y
1	1	0.46412	0.46722
2	2	0.46443	0.46753
3	3	0.46434	0.46744
4	4	0.46490	0.46800
5	5	0.04435	0.04464
6	6	0.30894	0.31100
7	7	0.44700	0.44998
8	8	0.41900	0.42180

+

Sparse Matrix

Obs	index	X	Y
1	1	0.05788	0.04278
2	2	0.17757	0.11547
3	3	0.16366	0.07556
4	4	0.36110	0.40700
5	5	0.05665	-0.01364
6	6	0.00106	0.00000
7	7	0.00000	3.97102
8	8	0.00000	0.05920

=

Original Matrix

Obs	index	X	Y
1	1	0.522	0.510
2	2	0.642	0.583
3	3	0.628	0.543
4	4	0.826	0.875
5	5	0.101	0.031
6	6	0.310	0.311
7	7	0.447	4.421
8	8	0.419	0.481

Where:

Sparse Matrix

Obs	index	X	Y
1	1	0.05788	0.04278
2	2	0.17757	0.11547
3	3	0.16366	0.07556
4	4	0.36110	0.40700
5	5	0.05665	-0.01364
6	6	0.00106	0.00000
7	7	0.00000	3.97102
8	8	0.00000	0.05920

Noise

Anomalies

Code Example

Daten:

- Maschinendaten

Coding Sprache:

- SAS Procedure
- CASL

Ziel:

- Auffinden von Anomalien

Robust Principle Component Analysis

```
proc rpca data=public.PHM08 method=alm lambdaweight=2
          outlowrank=casuser.low outsparse=casuser.sparse;
    id engine cycle;
    input X1-X24;
    svd method=eigen;
    where engine in (1,22,45,53,82,105,167,179);
run;
```

Augmented Lagrange Multiplier Method

In general, the augmented Lagrange method is used to solve nonlinear constrained optimization problems. In the case of PCP, an augmented Lagrange function is used to reformulate the PCP problem as the following nonlinear unconstrained optimization problem:

$$\text{minimize} \quad l(L,S,Y) = ||L||_* + \lambda||S||_1 + < Y, M - L - S > + \frac{\mu}{2}||M - L - S||_F^2$$

Candès et al. (2011) use the ALM method to find the solution to the preceding optimization problem. The basic idea is to update S, L, and Y iteratively. At iteration k, given L_k and Y_k, the first step is to find S_{k+1} by minimizing $l(L_k, S, Y_k)$. In the second step, L_{k+1} is obtained by the singular value thresholding operator, which minimizes $l(L, S_{k+1}, Y_k)$.[6] Next the Lagrange multiplier Y_{k+1} is updated. For more information, see Candès et al. (2011).

Robust Principle Component Analysis

Code Example

Daten:

- Maschinendaten

Coding Sprache:

- SAS Procedure
- CASL

Ziel:

- Auffinden von Anomalien

```
proc rpca data=public.PHM08 method=alm lambdaweight=2
          outlowrank=casuser.low outsparse=casuser.sparse;
   id engine cycle;
   input X1-X24;
   svd method=eigen;
   where engine in (1,22,45,53,82,105,167,179);
run;
```

```
ods trace on;
 proc cas;
     loadactionset "tkrpca";
     action robustpca /
         table={caslib="public", name="PHM08",
                where="engine in (1,22,45,53,82,105,167,179)"}
         inputs={{name="X1"},{name="X2"},{name="X3"},
                 {name="X4"},{name="X5"},{name="X6"},
                 {name="X7"},{name="X8"},{name="X9"},
                 {name="X10"},{name="X11"},{name="X12"},
                 {name="X13"},{name="X14"},{name="X15"},
                 {name="X16"},{name="X17"},{name="X18"},
                 {name="X19"},{name="X20"},{name="X21"},
                 {name="X22"},{name="X23"},{name="X24"}}
         method="ALM"
         decomp="svd"
         lambdaweight=2
         svdmethod="EIGEN"
         outmat={lowrankmat={name="casllow" replace=True},
                 sparsemat={name="caslsparse" replace=True}}
         outsvd={svdleft={name="svdleft" replace=True},
                 svddiag={name="svddiag" replace=True},
                 svdright={name="svdright" replace=True}}
     ;
 run;
```

Code Example

Daten:

- Maschinendaten

Coding Sprache:

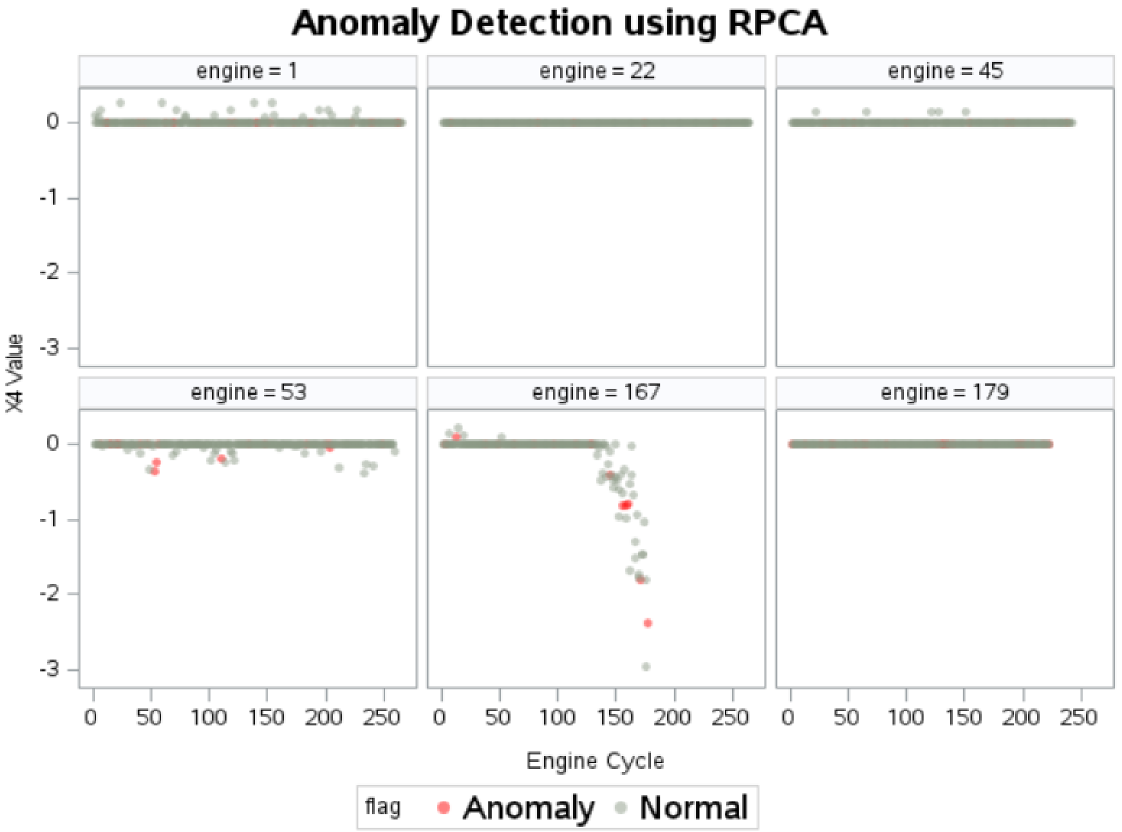
- SAS Procedure
- CASL

Ziel:

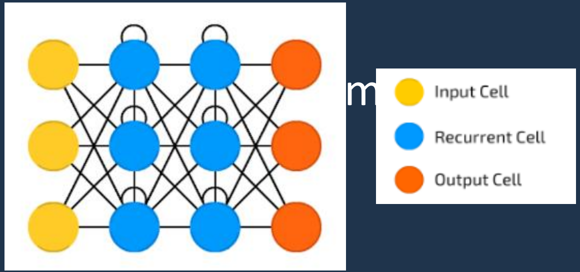
- Auffinden von Anomalien

```
proc rpca data=public.PHM08 method=alm lambdaweight=2
        outlowrank=casuser.low outsparse=casuser.sparse;
   id engine cycle;
   input X1-X24;
   svd method=eigen;
   where engine in (1,22,45,53,82,105,167,179);
run;
```

Robust Principle Component Analysis



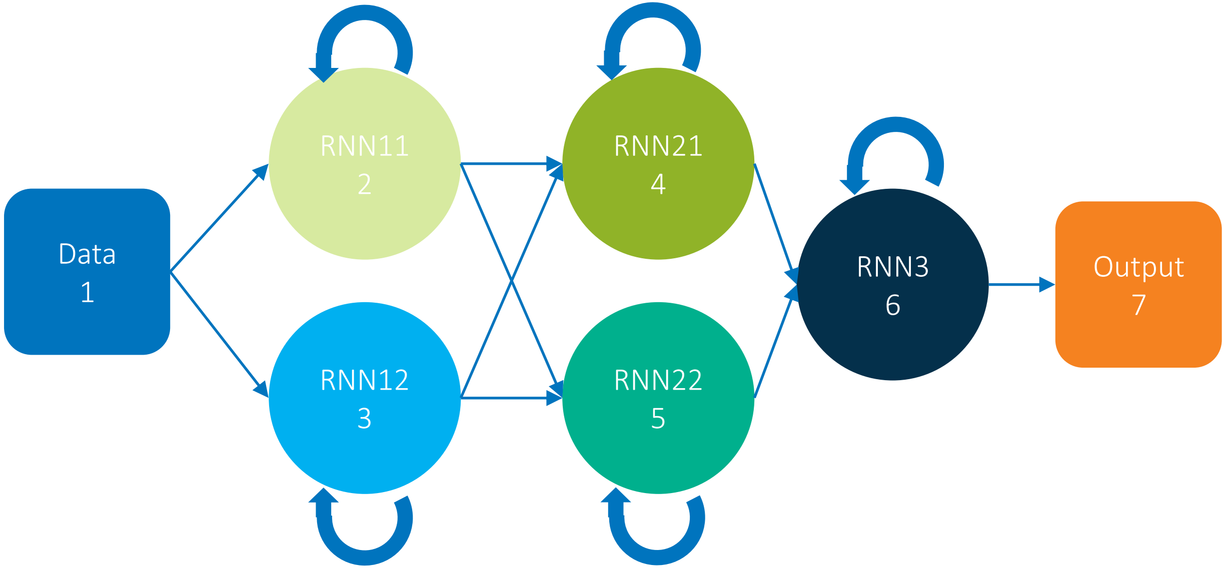
Recurrent Neural Networks



Practical Applications:

- Language modeling (e.g., statistical machine translation, word prediction)
- Time-series forecasting
- Image/video captioning

- Well suited for sequence data like time series or text
- Handle variable-length inputs and produce variable-length outputs → applications like NLG and machine translation.



Code Example

Daten:

- Bewertungen mit Sentiment Label

Coding Sprache:

- Python

Ziel:

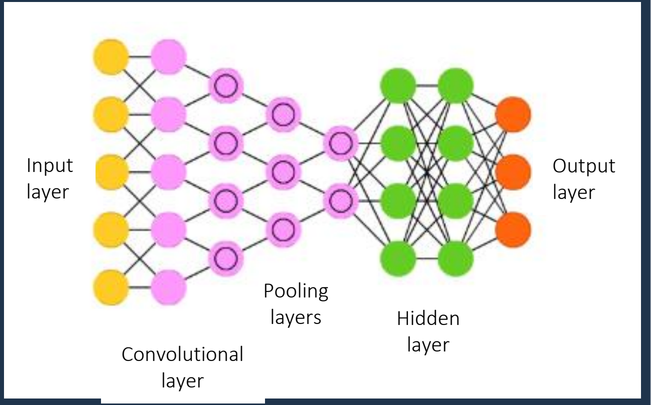
- Sentimentvorhersage

Recurrent Neural Network

Training the Model

```
s.dlTrain(table=train, model='sentiment', validtable=val,
            modelWeights=dict(name='sentiment_trainedWeights', replace=True),
            textParms=dict(initEmbeddings='glove', hasInputTermIds=False, embeddingTrainable=False),
            target='sentiment',
            inputs=['review'],
            texts=['review'],
            nominals=['sentiment'],
            optimizer=dict(miniBatchSize=2, maxEpochs=20,
                           algorithm=dict(method='adam', beta1=0.9, beta2=0.999, gamma=0.5,
                                          learningRate=0.0005, clipGradMax=100, clipGradMin=-100,
                                          stepSize=2, lrPolicy='step')
                          ),
            seed=12345
        )
```

Convolutional Neural Networks



Practical Applications:

- Image/object recognition
- Video analysis
- Text Classification

- CNNs can detect components in images such as edges and curves,
- Well-suited to array data where nearby values are correlated (images, sound, video, speech, etc.).
- CNNs can be implemented on GPUs
- Text analysis: word-level or character-level CNNs → sentiment analysis, categorization, or spam detection.

Convolutional Neuronal Network

Code Example

Daten:

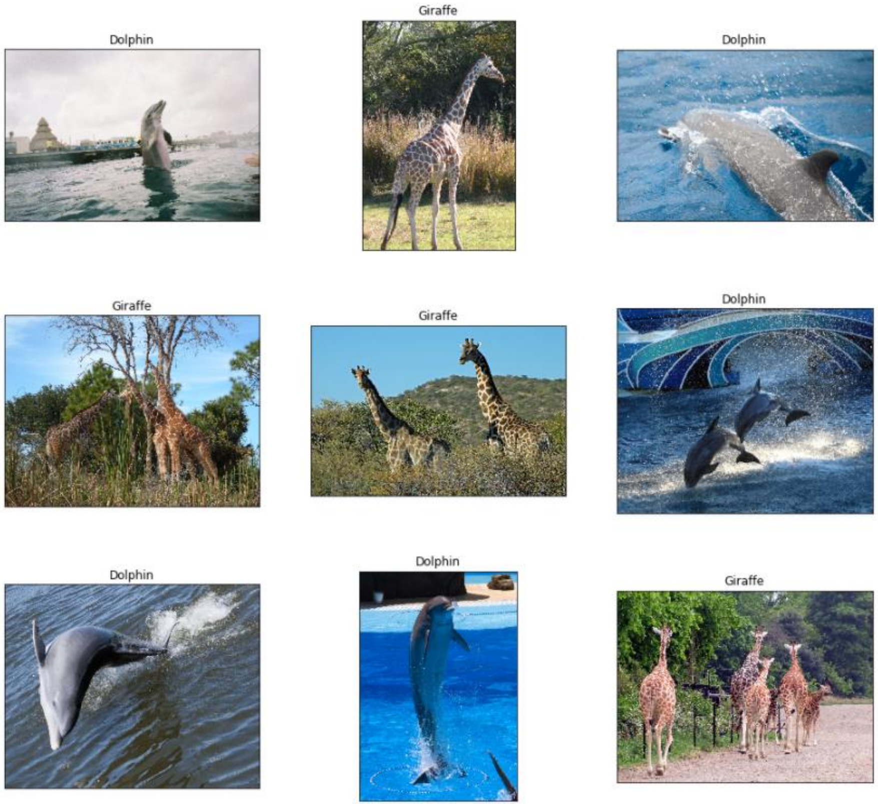
- Bilder von Giraffen und Delfinen

Coding Sprache:

- Python & DLPy Package

Ziel:

- Unterscheidung von Giraffen und Delfinen



Code Example

Daten:

- Bilder von Giraffen und Delfinen

Coding Sprache:

- Python & DLPy Package

Ziel:

- Unterscheidung von Giraffen und Delfinen

Convolutional Neuronal Network

```
from dlpy import Model, Sequential
from dlpy.layers import *
from dlpy.applications import *
model1 = Sequential(sess, model_name = 'Simple_CNN')
model1.add(InputLayer(3,224,224,offsets=tr_img.channel_means))
model1.add(Conv2d(8,7))
model1.add(Pooling(2))
model1.add(Conv2d(8,7))
model1.add(Pooling(2))
model1.add(Dense(16))
model1.add(OutputLayer(act='softmax',n=2))
```

Code Example

Daten:

- Bilder von Giraffen und Delfinen

Coding Sprache:

- Python & DLPy Package

Ziel:

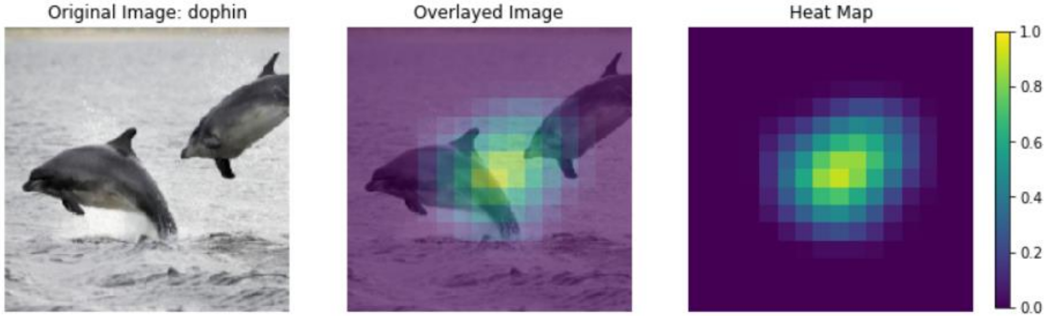
- **Unterscheidung von Giraffen und Delfinen**

§ ScoreInfo

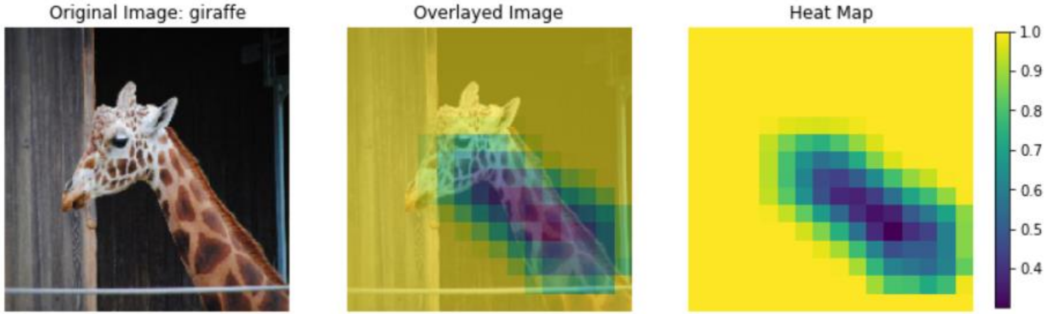
	Descr	Value
0	Number of Observations Read	324
1	Number of Observations Used	324
2	Misclassification Error (%)	6.481481
3	Loss Error	0.253215

Convolutional Neuronal Network

```
model2.plot_heat_map(image_id=2, alpha=0.6)
```

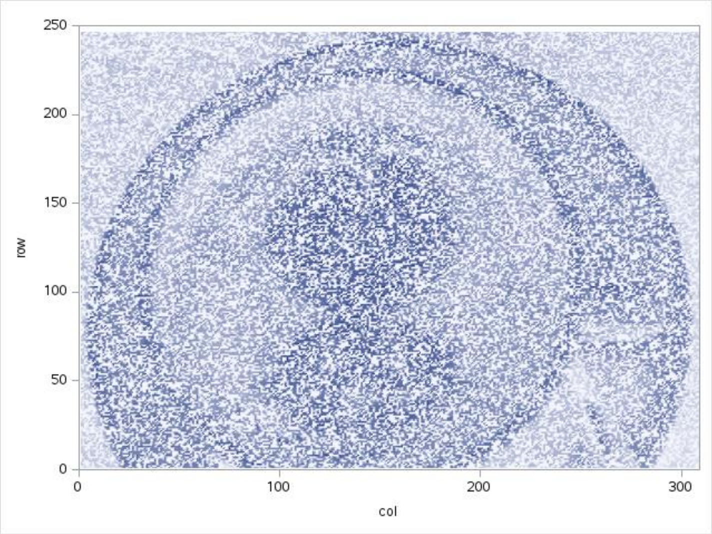


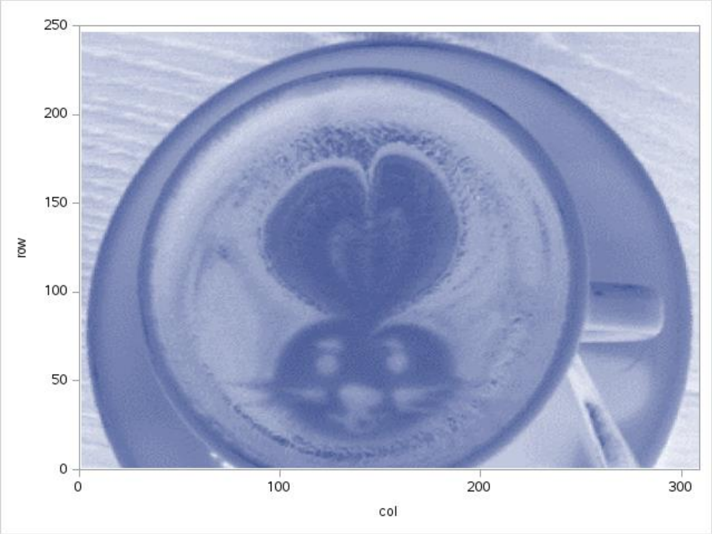

```
model2.plot_heat_map(image_id=0, alpha=0.6)
```



Other Applications

Image reconstruction

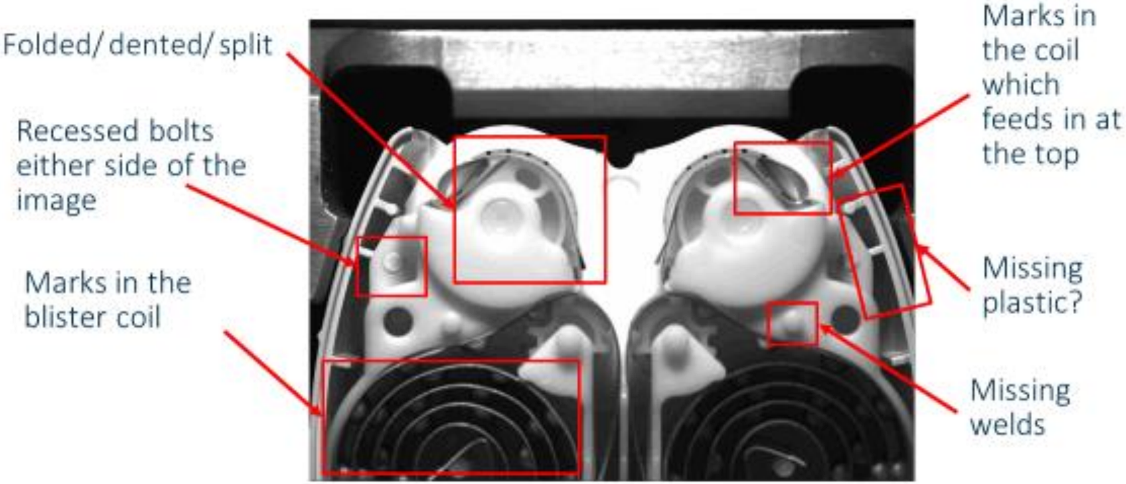




Deep Learning Toolkit

- Deep forward networks
- Convolutional deep networks
  - LeNet
  - VGG type of models such as VGG11, VGG13, VGG16, VGG19
  - ResNet
- Recurrent deep networks
  - LSTM
  - GRU

*Through SAS image analysis and machine learning, we have been able to **predict 99%** of images being good or bad correctly*



GlaxoSmithKline

Loading Data

OBJECTIVE

Demonstrate that SAS can load a significant volume to images to enable development of offline analytics

APPROACH

- Use the SAS "*loadImages*" action to load data from local disk to CAS in-memory engine for further processing
- Use the ID part of the filename to join on whether an image was good or bad.

INSIGHTS

- 26k images (one day) loaded in 30 secs
- 2.5m images (three weeks) loaded in 2 hours

```
%%time
sess.image.loadImages(casout={'name':'allImgs','caslib':'public','promote':True,"replication":0},
                      caslib="allImgs",path="/images/Ware_GMS_Assembly/ML5026_test",decode=False,recurse=True)

WARNING: The use of absolute paths is deprecated. Please convert your code to use a caslib.
NOTE: Loaded 2471171 images from /images/Ware_GMS_Assembly/ML5026_test into Cloud Analytic Services table allImgs.
CPU times: user 22.1 s, sys: 30.7 s, total: 52.8 s
Wall time: 1h 54min 19s
```

- 1 month's worth of labels obtained & joined
- For modelling purposes, since we had few bad images, we used one month of "bad" images, and one day of "good" images
- This meant there was a fair distribution of good and bad images, which prevented overfitting

	Column	CharVar	FmtVar	Level	Frequency
0	Good_Bad_T1_v2	defect	defect	1	2615.0
1	Good_Bad_T1_v2	no defect	no defect	2	24395.0

Processing Images

OBJECTIVE

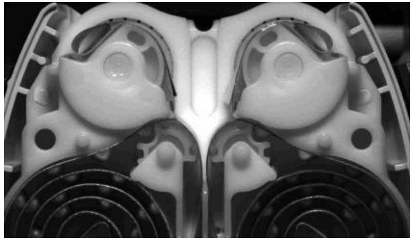
Explore whether performing various transformations to the images adds value to modelling

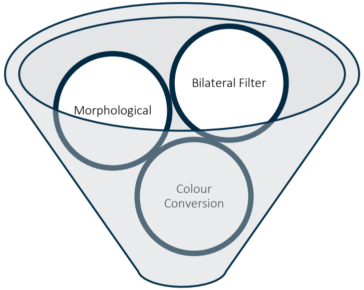
APPROACH

- Tested all available transformations within the SAS “*processImages*” action, including: cropping, resizing, bilateral filters, Laplacian transformations, etc…

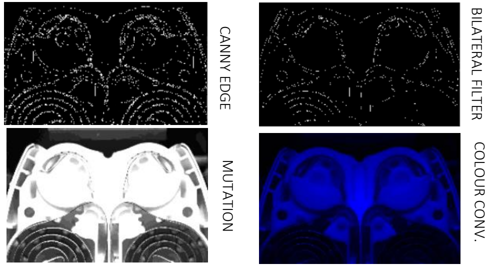
INSIGHTS

Performing multiple filters helped edge detection

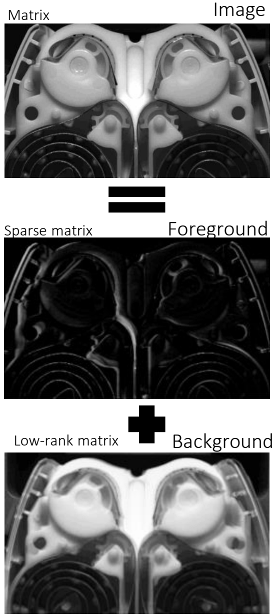


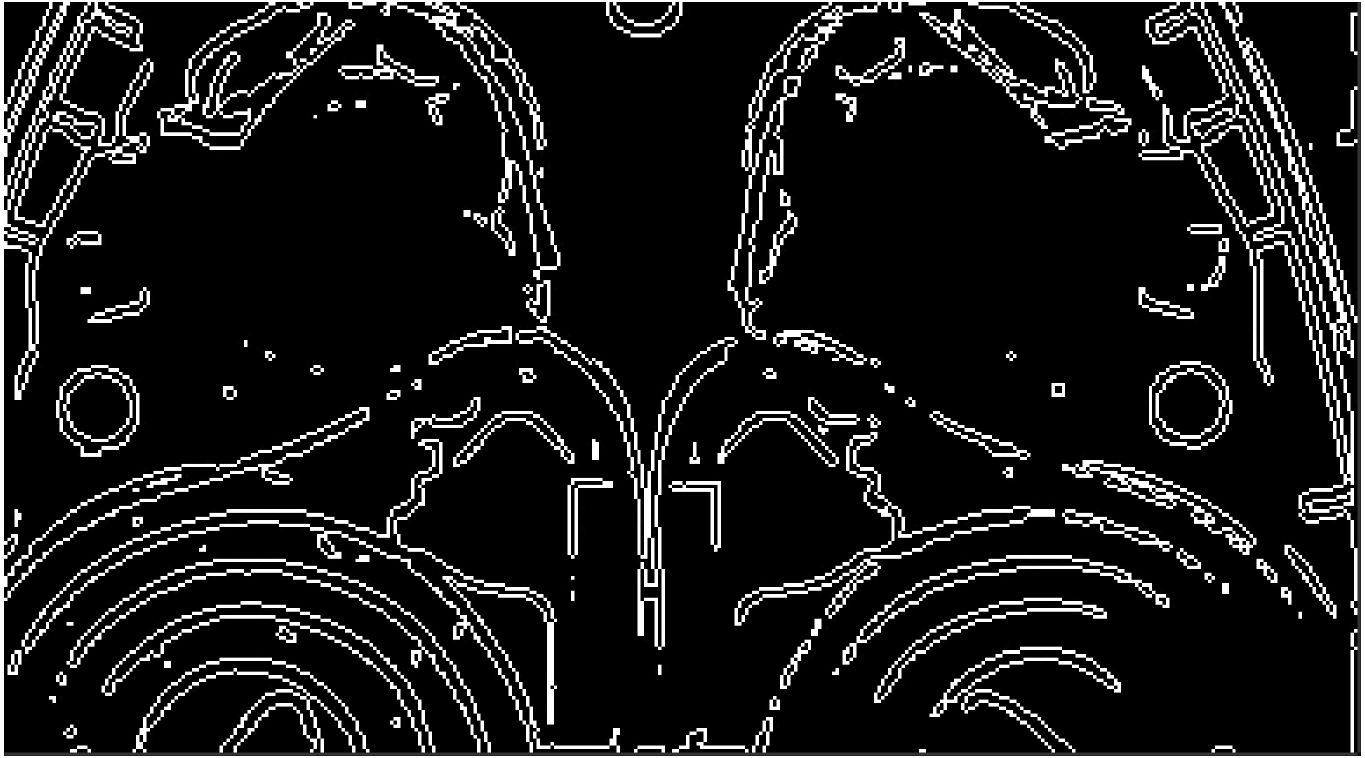


Approaches explored independently:



Robust PCA decomposes a matrix into a low-rank matrix and a sparse matrix – this can be applied to images to detect a “consistent” background & a “moving” foreground



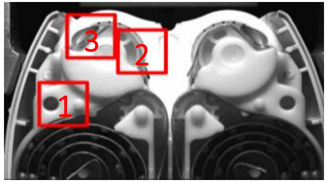


Supervised learning

OBJECTIVE

- Can deep learning approaches be applied to images to predict whether images are good or bad?
- Can we get a sensible runtime by executing on 4 GPUs?

APPROACH



- Augmentation used to increase no. of bad images
- Used iterative localisation approach
- Employed 6 transfer learning models to initialise the weights of the CNN models.

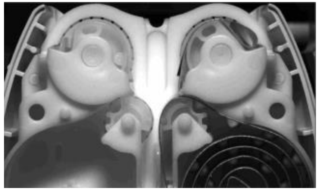
INSIGHTS

The confusion matrices below helped us visually inspect the False Negatives and False Positives.

Good_Bad_T1_v2	defect	no defect	Good_Bad_T1_v2	defect	no defect
LABEL_VGG16	Frequency	Frequency	LABEL_RN101	Frequency	Frequency
defect	165	37	defect	239	717
no defect	91	2,408	no defect	17	1,728
LABEL_RN50C	Frequency	Frequency	LABEL_RN152	Frequency	Frequency
defect	223	317	defect	165	6
no defect	33	2,128	no defect	91	2,439
LABEL_RN50S	Frequency	Frequency	LABEL_DN	Frequency	Frequency
defect	243	1,645	defect	8	.
no defect	13	800	no defect	248	2,445

Through ensembling 6 different models, we were able to build very effective models

This helped identify images which were potentially falsely labelled



Misclassification rates for patches on test dataset: **1=2.8%, 2=2.2%, 3=1.3%.** Each patch took 30 mins on 4 GPUs.

There is lots of scope for fine-tuning performance

Take Aways

- SAS provides at lot of AI functionality: object recognition, image mining, text analytics, deep learning, recurrent NN, convolutional NN, ...
- These methods are executed „distributed“, „in-memory“ on the CAS Server in SAS Viya.
- You can use SAS procedures, SAS Actions and Actionsets (CAS-L, Python, R, ...) and the Deep Learning Tookit from SAS