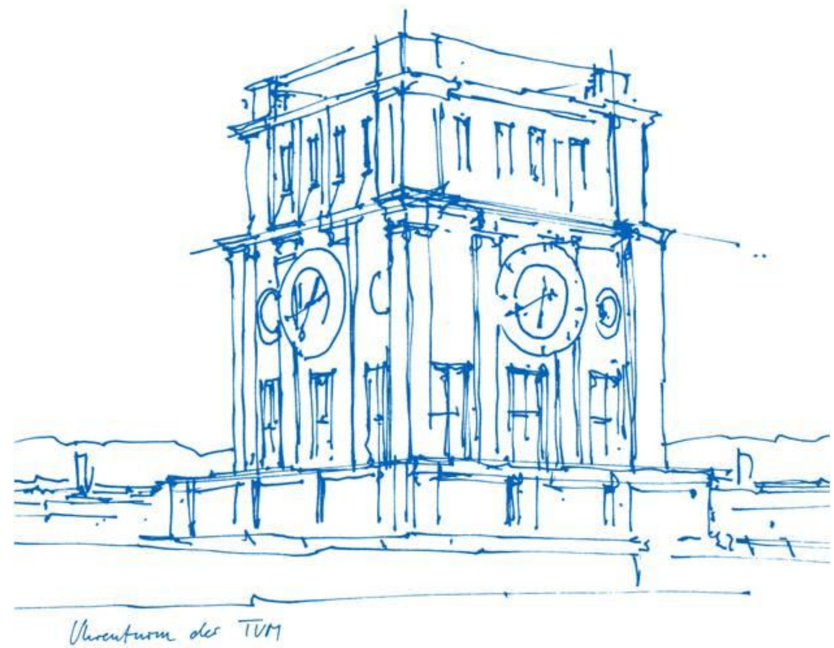
Geometrical Deep Learning on 3D Models: Classification for Additive Manufacturing

Nouhayla Bouziane | Ahmed Ebid | Aditya Sai Srinivas | Felix Bok | Johannes Kiechle

Technical University Munich

Faculty of Mathematics

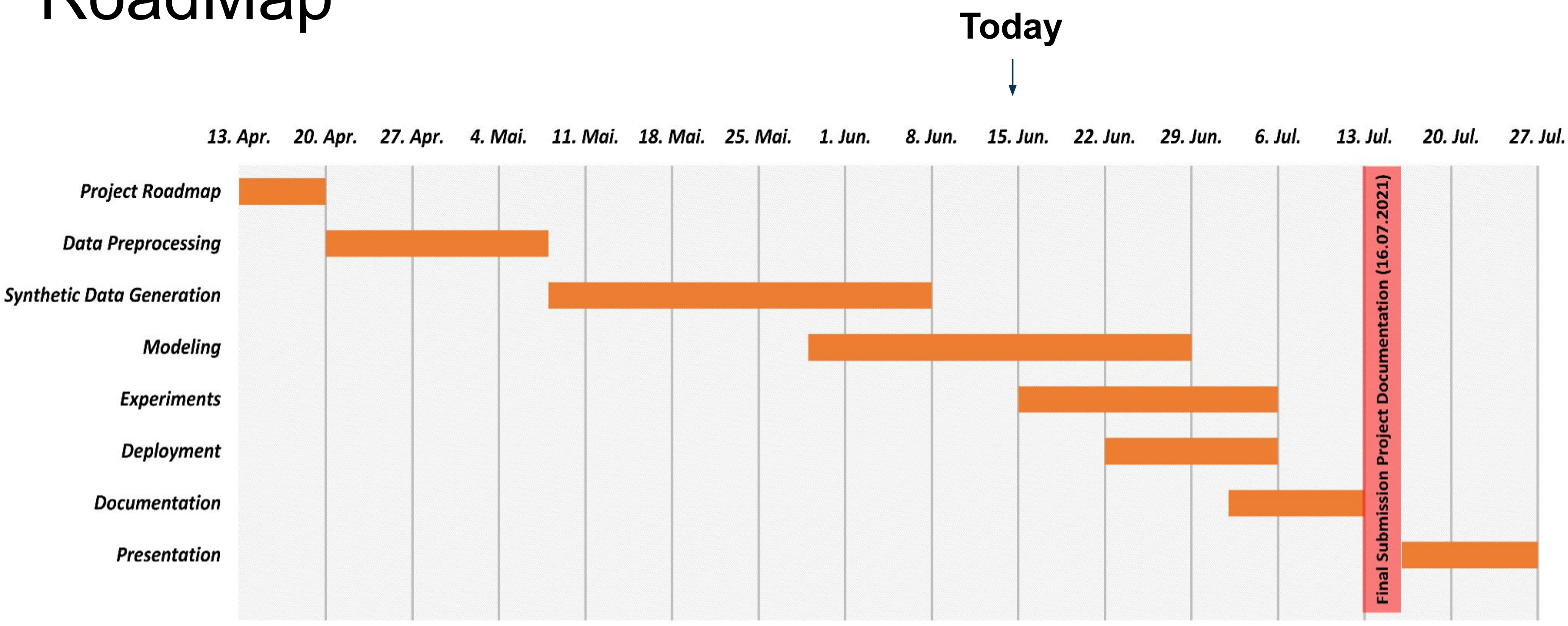
15. June 2021



Agenda

Agenda

RoadMap



Today
13. Apr.
20. Apr.
27. Apr.
4. Mai.
11. Mai.
18. Mai.
25. Mai.
1. Jun.
8. Jun.
15. Jun.
22. Jun.
29. Jun.
6. Jul.
13. Jul.
20. Jul.
27. Jul.
Project Roadmap
Data Preprocessing
Synthetic Data Generation
Modeling
Experiments
Deployment
Documentation
Presentation
Final Submission Project Documentation (16.07.2021)

Agenda

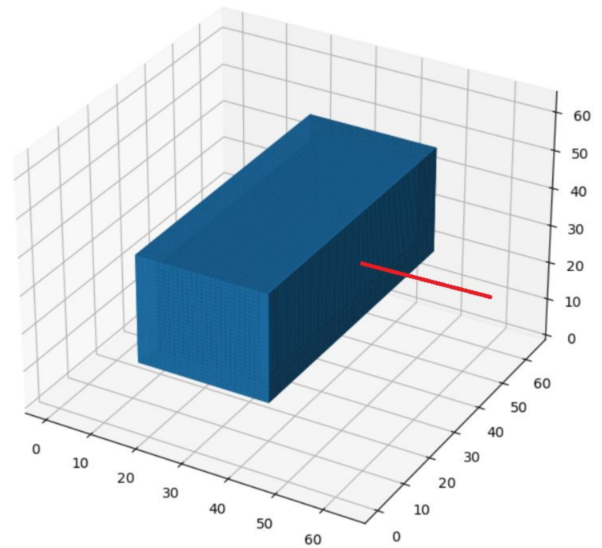
Defector

- A hole is created through a voxelized model using a cylinder.
- A cylinder is defined by 3 parameters:
  a. **Cylinder Radius**: size of the cylinder radius
  b. **Cylinder Axis**: axis through which cylinder is defined (X, Y, Z)
  c. **Cylinder Center Location**: location of the cylinder center

Defector: Cylinder Radius

- Given an axis, find a suitable cylinder radius

    a. Find the perpendicular plane (2D)
    b. Get the length of each dimension (1D) out of the plane (2D)
    c. Choose the smaller dimension
    d. Starting with a defined maximum cylinder diameter (**X**) :
        - subtract **X** from the smaller dimension
        - if more than 30 voxels remain, choose this diameter
        - else, test a smaller diameter



Defector: Cylinder Axis

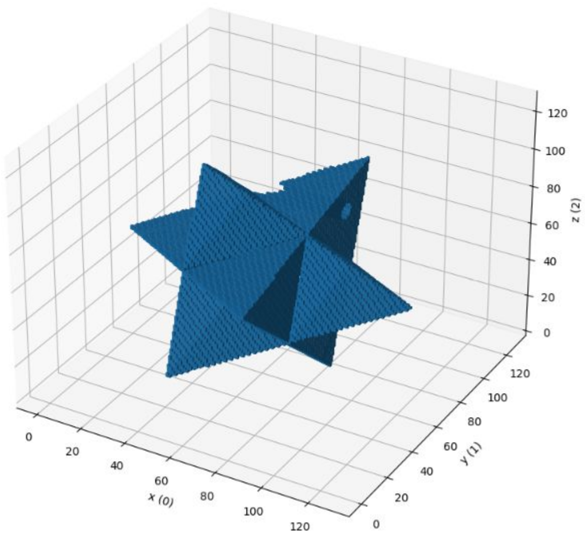
- For each coordinate axis (X, Y, Z):
    a. Find the radius of the cylinder through that axis
    b. Choose the axis that provides the largest radius

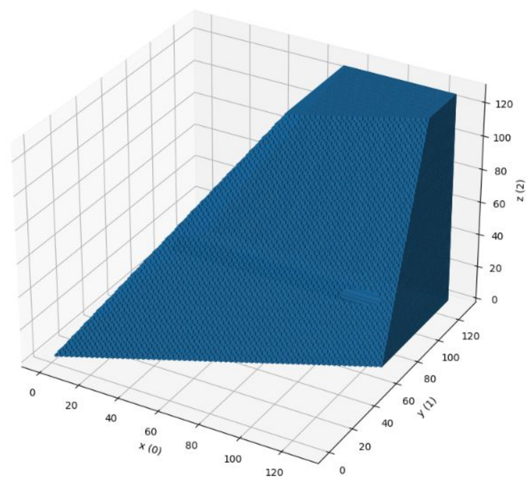
Defector: Cylinder Center Location

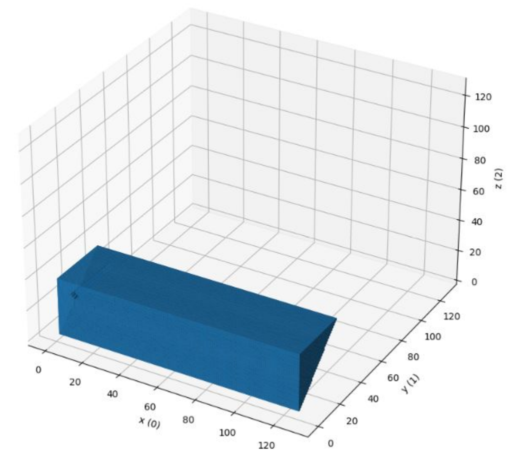
- Given a cylinder radius and axis, find the center of the cylinder

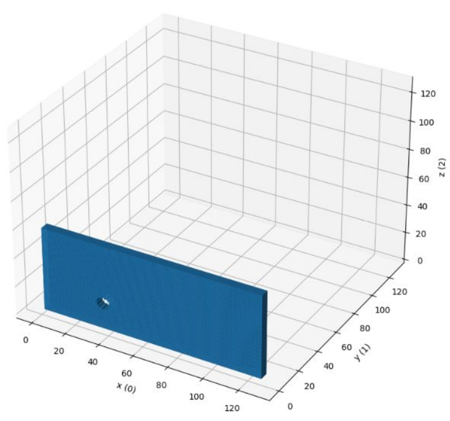
   a. Get the voxels out of the occupancy grid
   b. For **X** trials out of randomly chosen voxels:
      - skip a voxel that is too close to the plane boundaries
      - find the area of the circle defined by the voxel as a center
   c. Choose the voxel that has the maximum area
   d. Make sure that the chosen voxel has an area greater than or equal to a full circle

Defector: Results









Defector: Advantages & Disadvantages

Advantages

- considers all axes
- reduces radius to fit

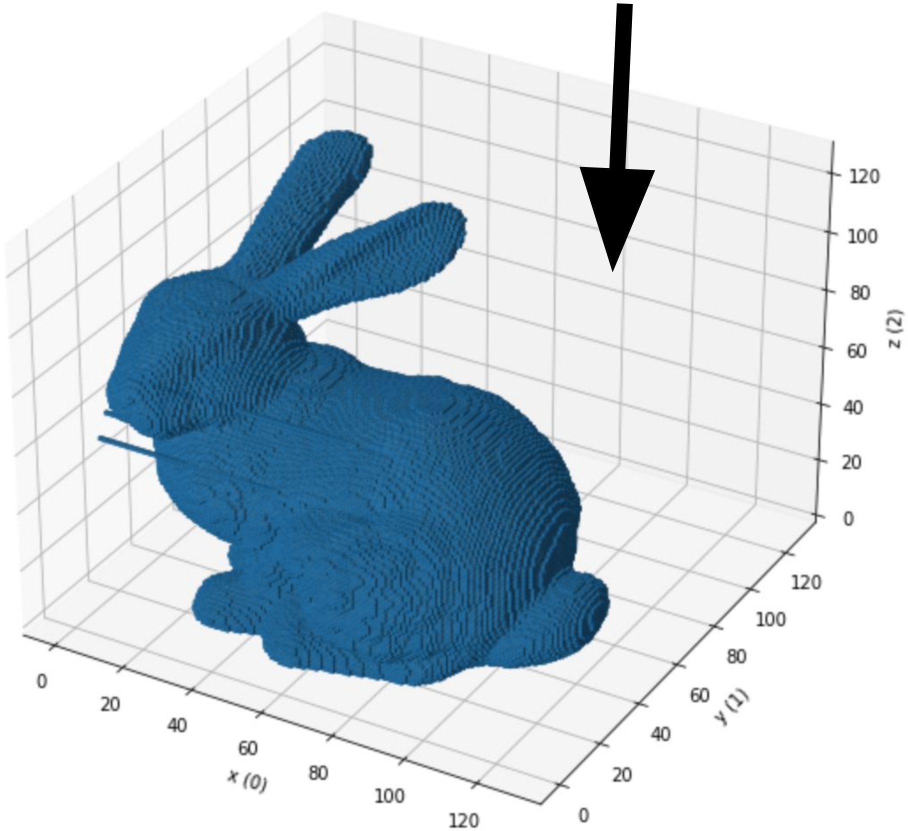
Disadvantages

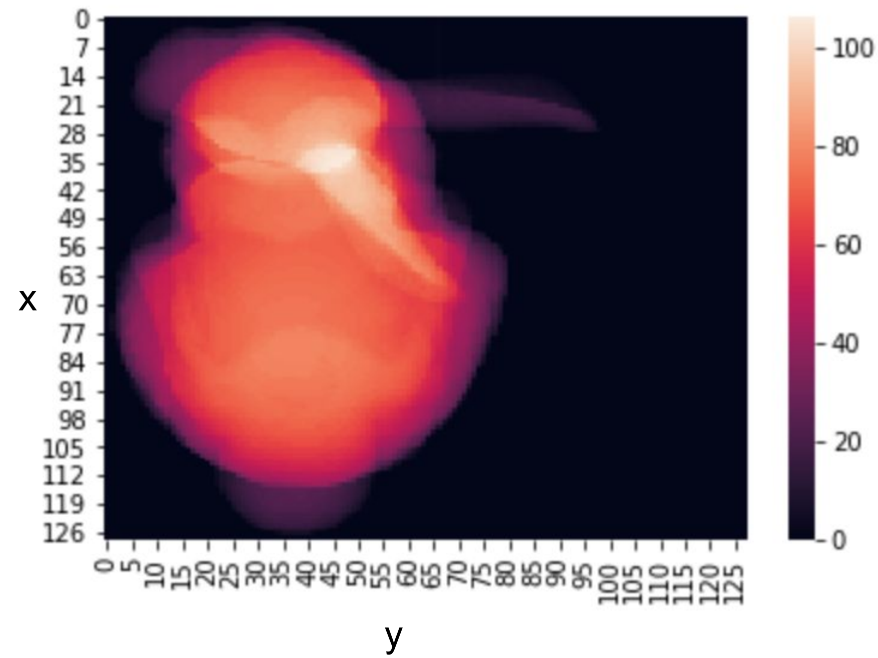
- avoids holes too close to the boundaries

Agenda

DefectorTopDownView

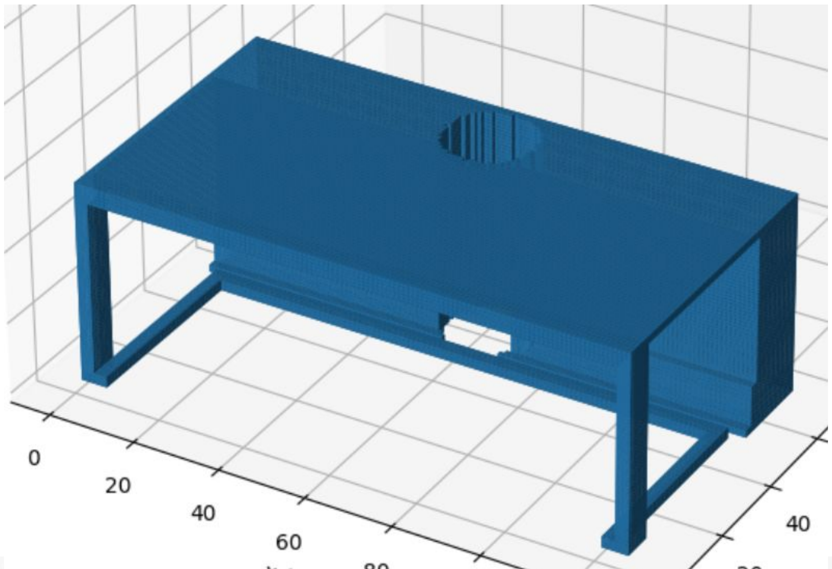
Sum over z-axis

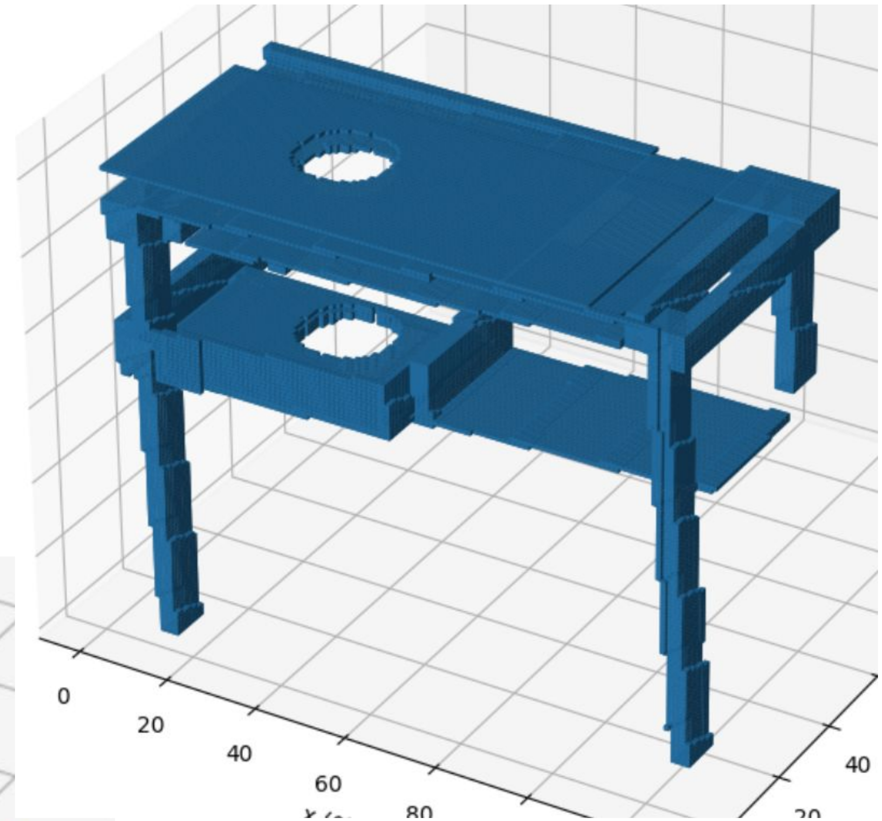


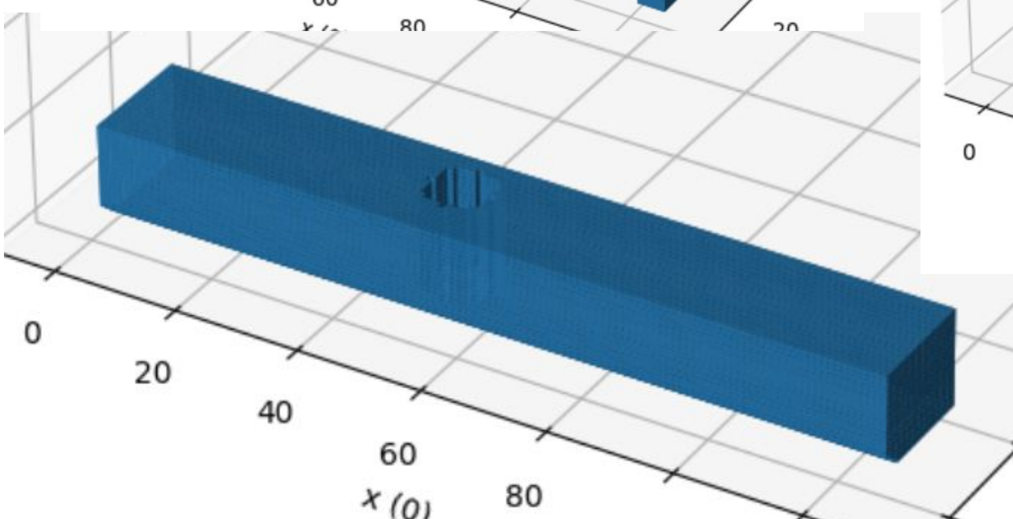


DefectorTopDownView

DefectorTopDownView







DefectorTopDownView

Advantages:

- Usage of model properties
- Accurate (hole with too narrow wall)
- Due to pre-selection of possible offsets it finds efficiently a suitable offset
- All proposed defects can be added
- Few and understandable input parameters
- Flexible regarding the properties of the defects (radius, border, shape)
- Balanced output, regarding the labels
- Can be extended to find a uniform area to add defects → more robustness
- Generated dataset has around two times the data points relative to input dataset

Disadvantages:

- Two checks for hole feasibility needed
- Informations could be misleading, since only 2D (see model right hand side of previous slide)

Agenda

Defector Rotation

- The objective was to create rotated holes with random angles and multiple rotation axes.
- Three approaches were tested:
  - Rotation of the model and rotation back of the hole using scipy
  - Rotation of the model indices and rotation back of the hole using designed functions
  - Rotation of the model and insertion of cylinder hole

Defector Rotation: First approach

- Rotate the model using scipy library
- Find the offset
- Find the cylinder indices in rotated model
- Rotate back the cylinder using scipy
- Remove the cylinder from the original object

Problem: Rotating and rotating back does not give the same indices due to an added constant. So the cylinder rotated back has different indices and its insertion is not coherent.

Defector Rotation: Second approach

- Rotate the voxel indices using rotation matrices
- Get the model occupancy grid
- Find the offset
- Find the Cylinder indices in rotated indices of the model
- Rotate back the indices using rotation matrices
- Remove the cylinder rotated back

Problem: Negative values of indices after rotation, hard to work with and moving to occupancy doesn't give the coherent results.

Defector Rotation: Third approach

- Rotate the model using scipy function **scipy.ndimage.rotate**
- Find the offset
- Remove the cylinder from rotated object
- Rotate the object back using scipy function **scipy.ndimage.rotate**

Problem: Rotating some models changes their shapes

Defector Rotation: Third approach

Proposed solution 1 :

Select models where the rotation doesn't change the shape.

- start with the rotation along the x axis
- get max coordinates of y and z
- verify if these values less than max_shape / sqrt(2)
- do the same process for y and z axes.

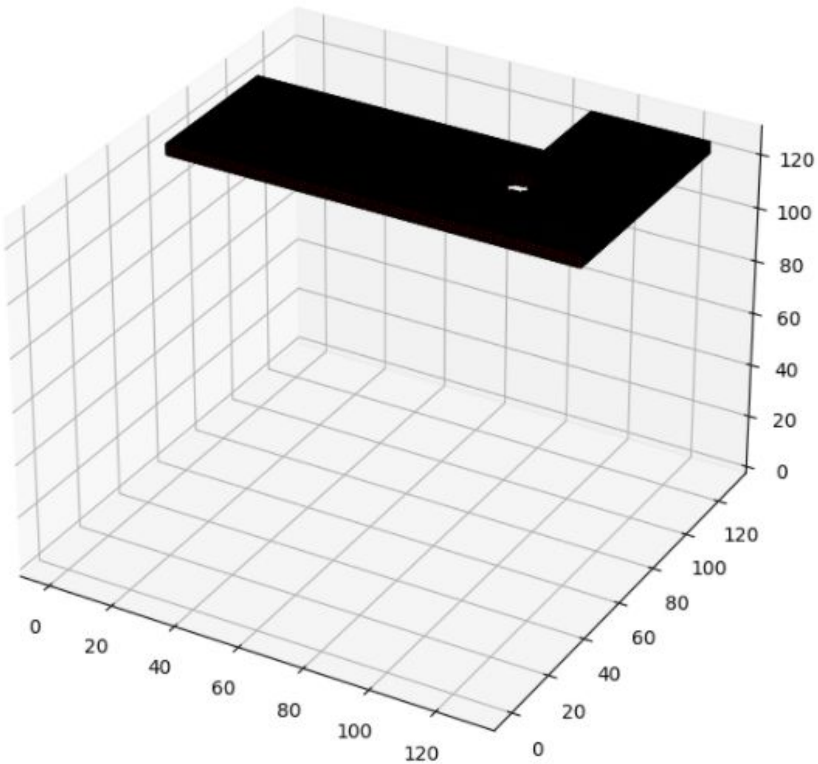
Disadvantages: We lose the possibility insert rotated holes for some models

Proposed solution 2 :

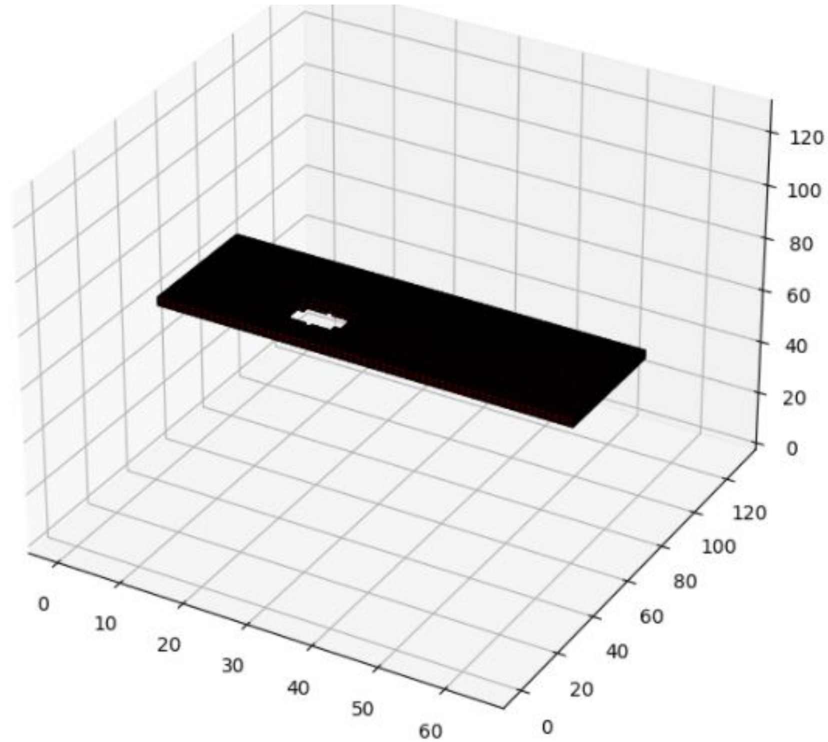
- start with Model shape 64x64x64
- extend the rotated model to a grid with shape 128x128x128

Disadvantages: We start with a small resolution

Defector Rotation: Some Results



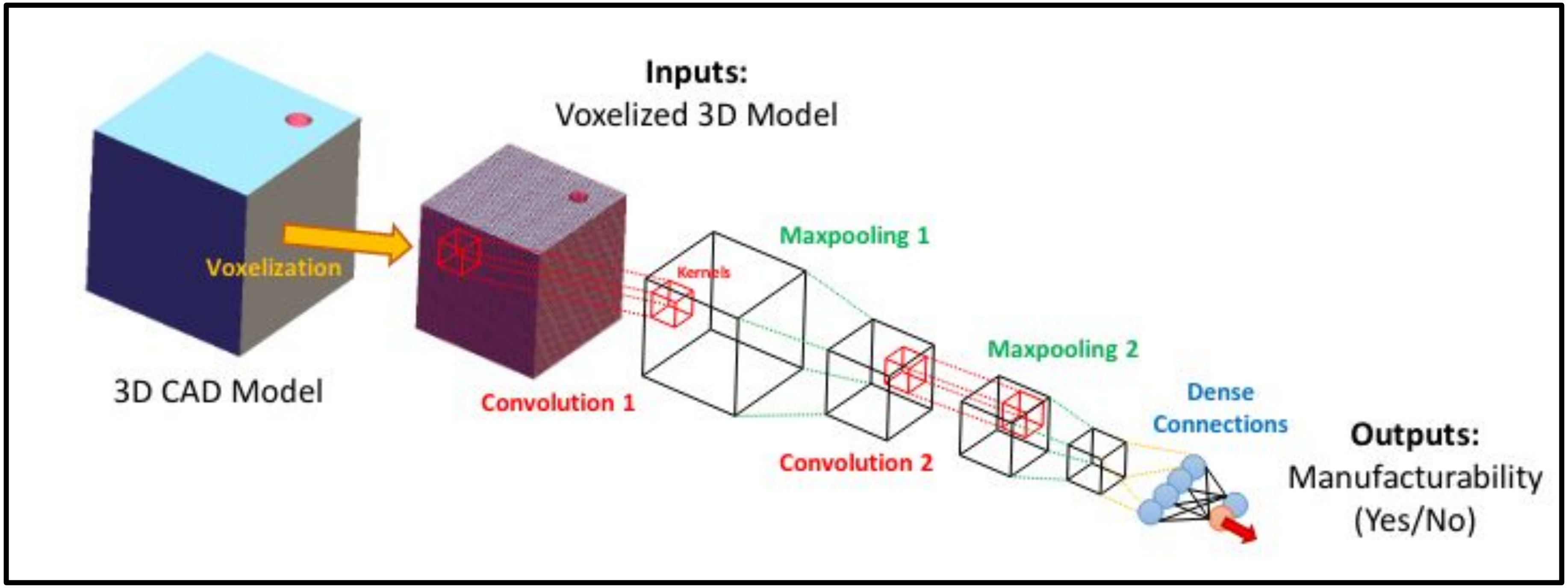
Hole rotated with 180 degrees



Hole rotated with 30 degrees

Agenda

Neural Network Classifier



3D-CNN Classifier Network

Source:Aditya Balu,Sambit Ghadai, Kin Gwn Lore, Gavin Young, Adarsh Krishnamurthy, Soumik Sarkar, **Learning Localized Geometric Features Using 3D-CNN: An Application to Manufacturability Analysis of Drilled Holes,** In *CVPR*,2017.

Network Architectures

- Vanilla 3D CNN
- 3D-ResNet

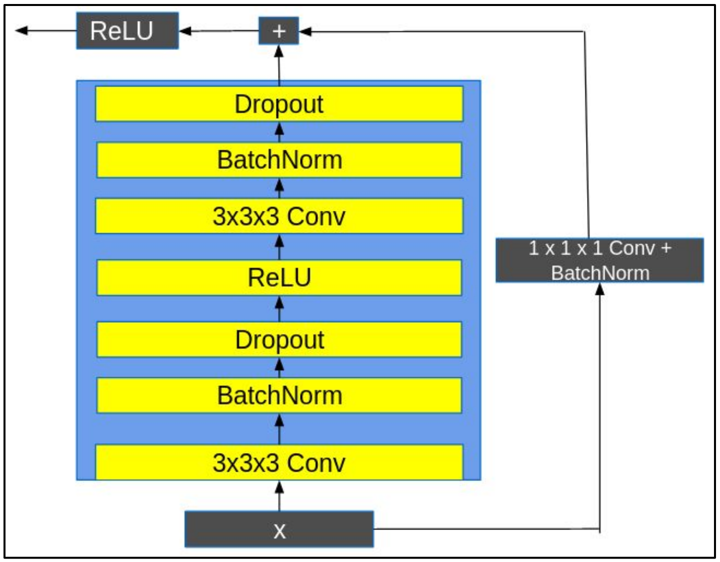
Vanilla 3D CNN

- Simple network architecture.
- Number of trainable parameters too less.
- Dense neural network.
- Not suitable while the data is scaled.
- Lower receptive field.

```
        Layer (type)               Output Shape         Param #
================================================================
            Conv3d-1    [-1, 32, 120, 120, 120]          23,360
              ReLU-2    [-1, 32, 120, 120, 120]               0
            Conv3d-3    [-1, 64, 114, 114, 114]         702,528
              ReLU-4    [-1, 64, 114, 114, 114]               0
         MaxPool3d-5       [-1, 64, 57, 57, 57]               0
            Conv3d-6       [-1, 96, 53, 53, 53]         768,096
              ReLU-7       [-1, 96, 53, 53, 53]               0
         MaxPool3d-8       [-1, 96, 26, 26, 26]               0
            Conv3d-9      [-1, 128, 24, 24, 24]         331,904
             ReLU-10      [-1, 128, 24, 24, 24]               0
        MaxPool3d-11      [-1, 128, 12, 12, 12]               0
        AvgPool3d-12      [-1, 128, 11, 11, 11]               0
        MaxPool3d-13         [-1, 128, 1, 1, 1]               0
          Dropout-14                  [-1, 128]               0
           Linear-15                   [-1, 32]           4,128
             ReLU-16                   [-1, 32]               0
          Dropout-17                   [-1, 32]               0
           Linear-18                    [-1, 1]              33
================================================================
Total params: 1,830,049
Trainable params: 1,830,049
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 8.00
Forward/backward pass size (MB): 2641.94
Params size (MB): 6.98
Estimated Total Size (MB): 2656.92
```

Summary of network architecture

3D-ResNet



Residual Block

ResNet Architecture					
Layer Name	**Output Size**	**18-layer**	**50-layer**	**101-layer**	**152-layer**
conv_1	64 x 64 x 64	7 x 7 x 7/2,64			
	32 x 32 x 32	3 x 3 x 3 maxpool,stride 2			
conv2_x		(3 x 3 x 3/1,64) x 2	(3 x 3 x 3/1,64) x 3	(3 x 3 x 3/1,64) x 3	(3 x 3 x 3/1,64) x 3
		(3 x 3 x 3/1,64) x 2	(3 x 3 x 3/1,64) x 3	(3 x 3 x 3/1,64) x 3	(3 x 3 x 3/1,64) x 3
			(3 x 3 x 3/1,64) x 3	(3 x 3 x 3/1,64) x 3	(3 x 3 x 3/1,64) x 3
conv3_x	16 x 16 x 16	(3 x 3 x 3/2,128) x 2	(3 x 3 x 3/2,128) x 4	(3 x 3 x 3/2,128) x 4	(3 x 3 x 3/2,128) x 8
		(3 x 3 x 3/2,128) x 2	(3 x 3 x 3/2,128) x 4	(3 x 3 x 3/2,128) x 4	(3 x 3 x 3/2,128) x 8
			(3 x 3 x 3/2,128) x 4	(3 x 3 x 3/2,128) x 4	(3 x 3 x 3/2,128) x 8
conv4_x	8 x 8 x 8	(3 x 3 x 3/2,256) x 2	(3 x 3 x 3/2,256) x 6	(3 x 3 x 3/2,256) x 23	(3 x 3 x 3/2,256) x 36
		(3 x 3 x 3/2,256) x 2	(3 x 3 x 3/2,256) x 6	(3 x 3 x 3/2,256) x 23	(3 x 3 x 3/2,256) x 36
			(3 x 3 x 3/2,256) x 6	(3 x 3 x 3/2,256) x 23	(3 x 3 x 3/2,256) x 36
conv5_x	4 x 4 x 4	(3 x 3 x 3/2,512) x 2	(3 x 3 x 3/2,512) x 3	(3 x 3 x 3/2,512) x 3	(3 x 3 x 3/2,512) x 3
		(3 x 3 x 3/2,512) x 2	(3 x 3 x 3/2,512) x 3	(3 x 3 x 3/2,512) x 3	(3 x 3 x 3/2,512) x 3
			(3 x 3 x 3/2,256) x 3	(3 x 3 x 3/2,256) x 3	(3 x 3 x 3/2,256) x 3
	1 x 1 x 1	Adaptive average pool, 512-d fc			
FC_1		256-d fc,ReLU			
FC_2		64-d fc,ReLU			
FC_3		Scalar value,Sigmoid			

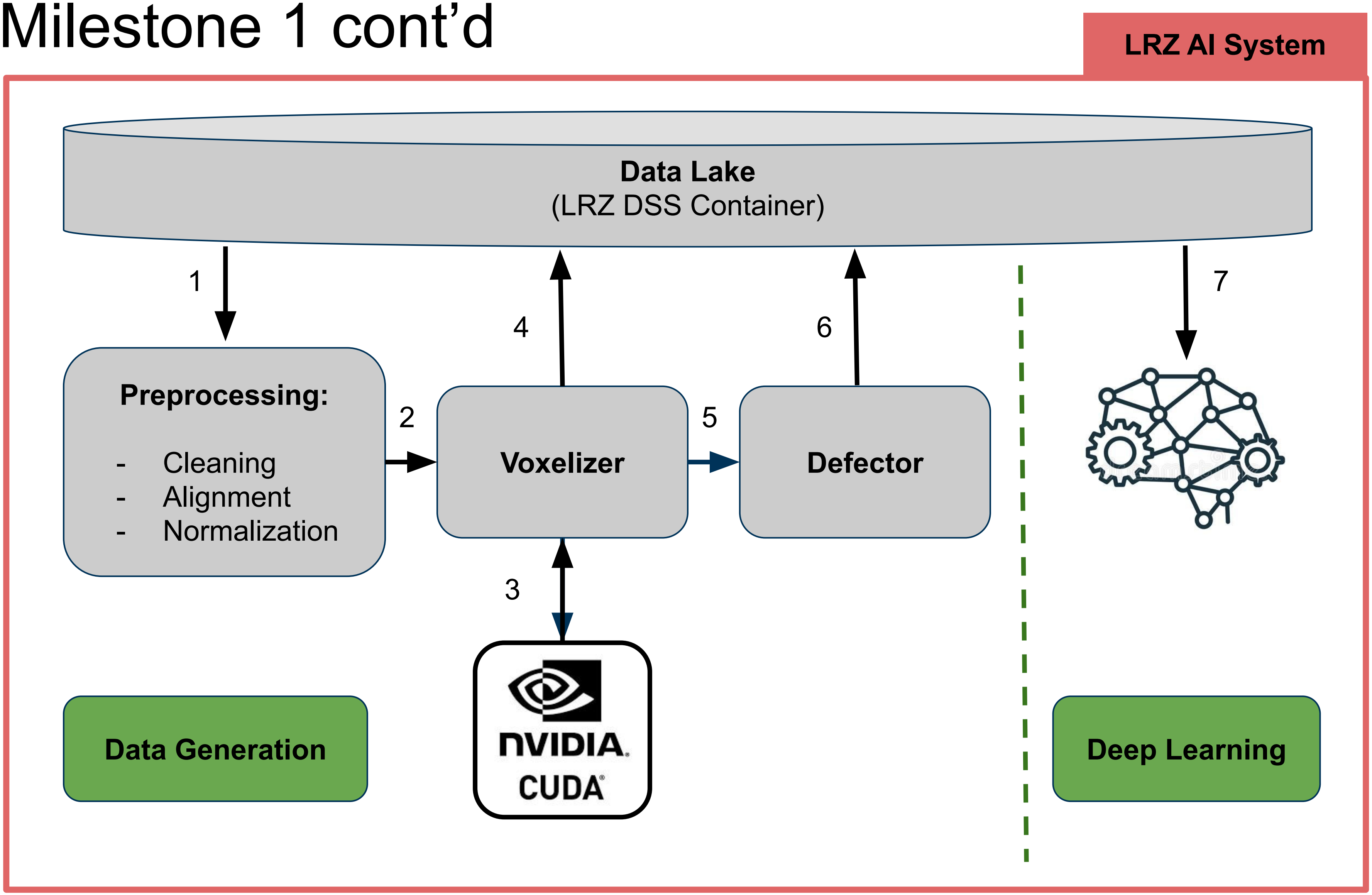
VOLKSWAGEN DATA:LAB

Implementation Details

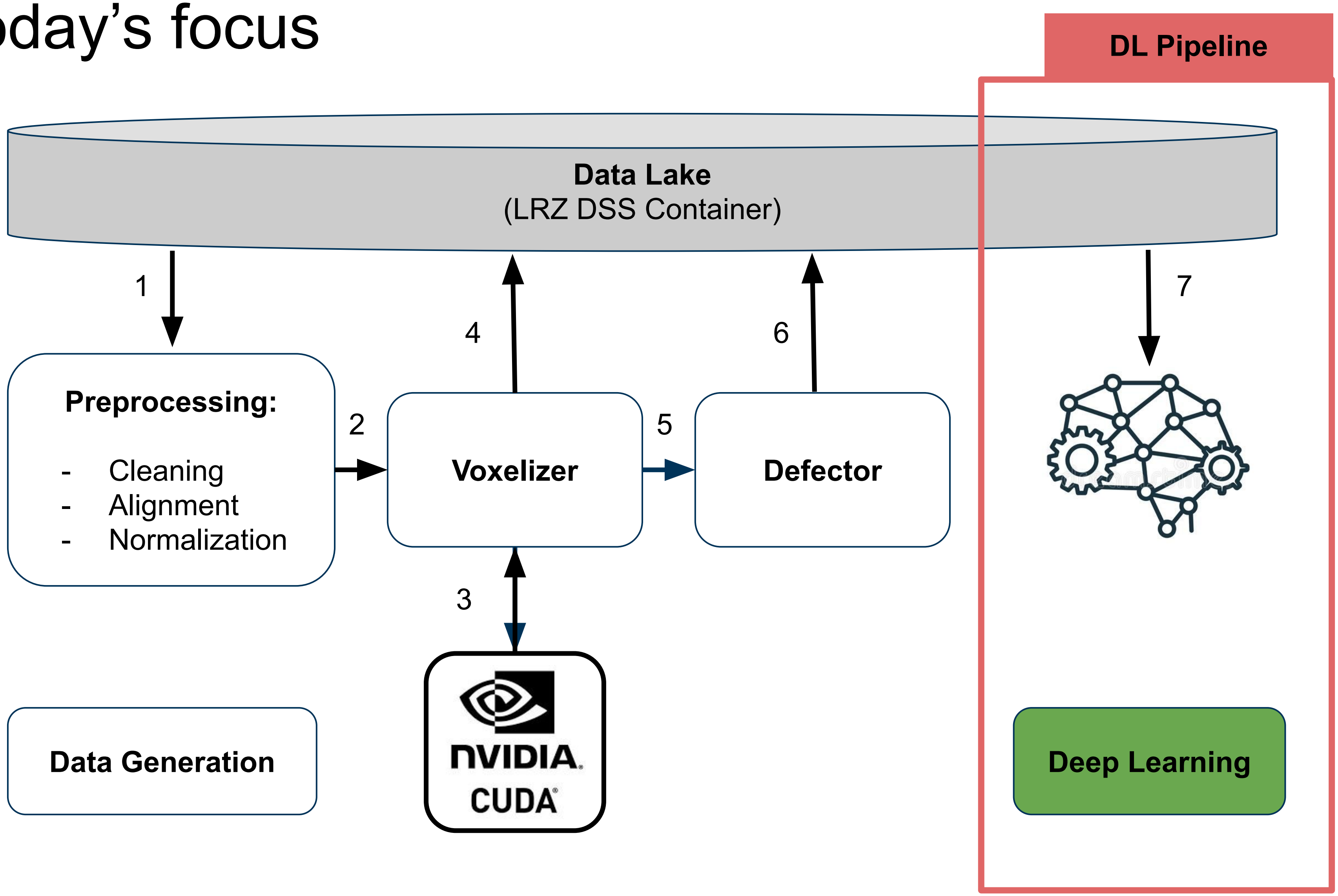
- Activation function used in final layer: Sigmoid
- Loss function used: Binary cross entropy loss
- Weights initializer: Kaiming normal
- Base model: 3D-ResNet 18 pretrained (To be investigated)
- Optimizer: 1st order gradient based - Adam Optimizer
- Performance metrics: Loss and Accuracy

Agenda

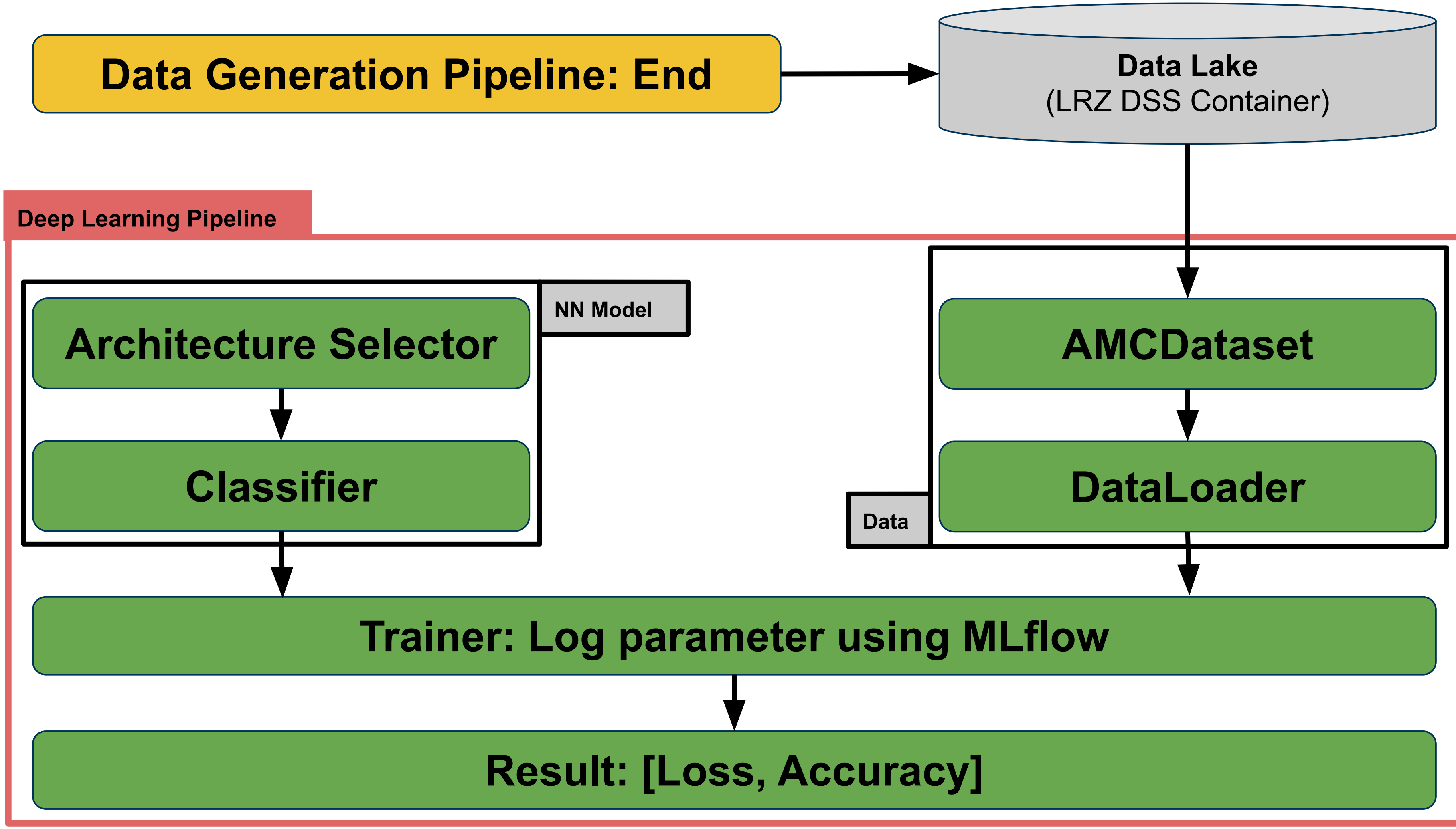
Milestone 1 cont'd



Today's focus



Deep Learning Infrastructure



VOLKSWAGEN DATA:LAB

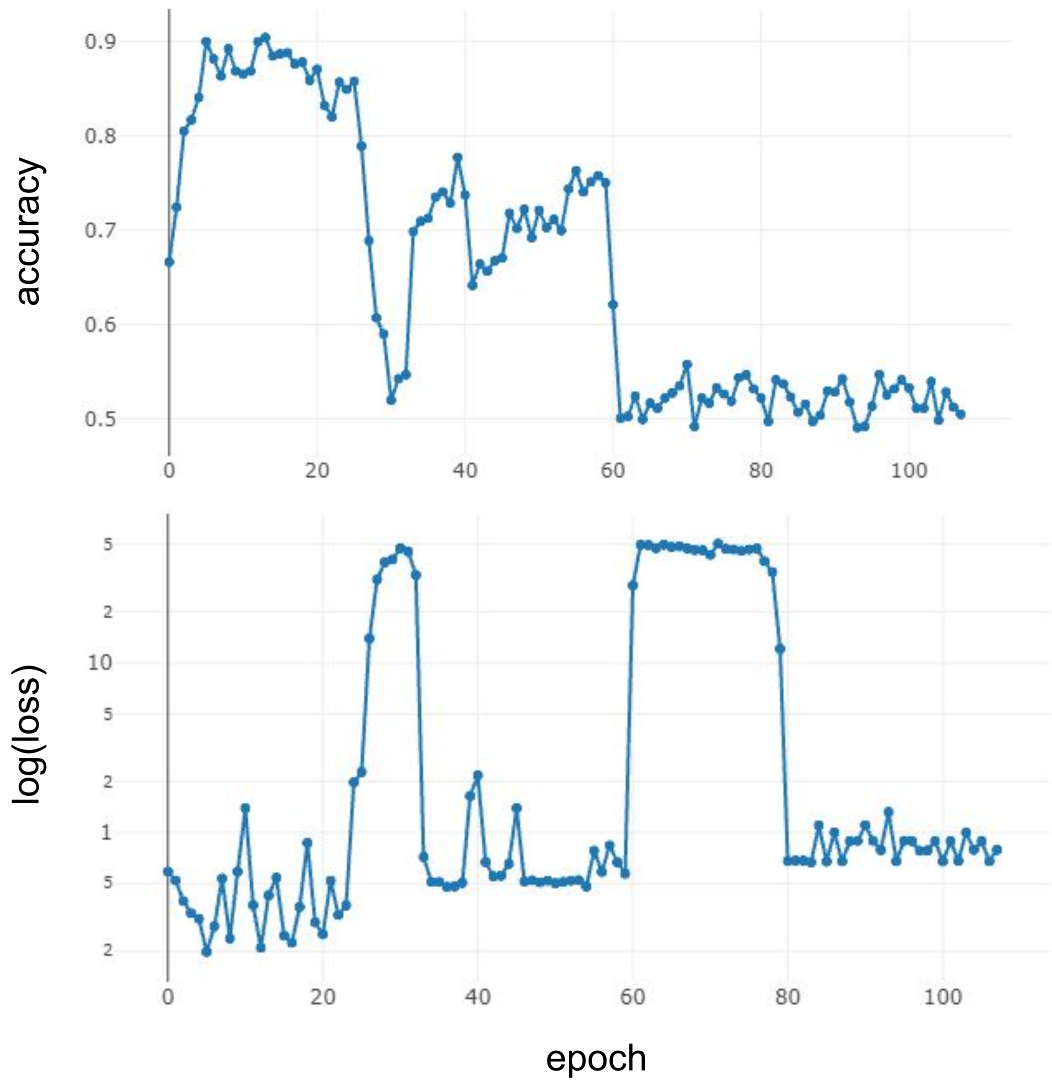
Compute Resources

- **LRZ AI System:**
  - Up to 8 GPUs in parallel
  - Nvidia Tesla V100
- **Neural Network size:**
  \>> 1 million parameters
- **Input dimensions:**
  128^3 ~ 2.1 million voxels

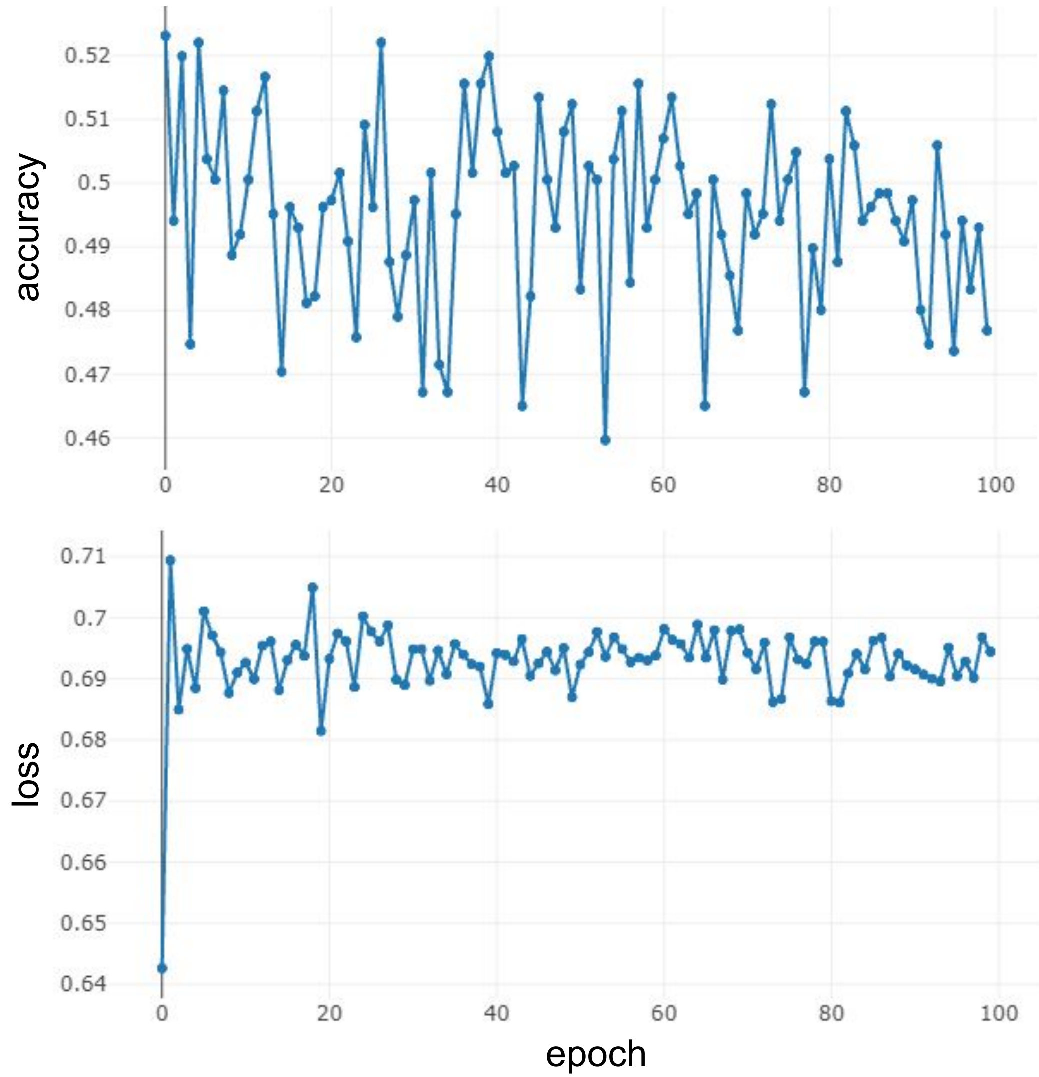
```
Every 0.5s: nvidia-smi

Tue Jun 15 09:07:34 2021
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 465.19.01    Driver Version: 465.19.01    CUDA Version: 11.3     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA Tesla V1...  On   | 00000000:06:00.0 Off |                    0 |
| N/A   55C    P0   241W / 300W |  15393MiB / 16160MiB |     92%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   1  NVIDIA Tesla V1...  On   | 00000000:07:00.0 Off |                    0 |
| N/A   58C    P0   242W / 300W |   8832MiB / 16160MiB |     95%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   2  NVIDIA Tesla V1...  On   | 00000000:0A:00.0 Off |                    0 |
| N/A   60C    P0   256W / 300W |   8832MiB / 16160MiB |    100%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   3  NVIDIA Tesla V1...  On   | 00000000:0B:00.0 Off |                    0 |
| N/A   51C    P0   239W / 300W |   8812MiB / 16160MiB |     94%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   4  NVIDIA Tesla V1...  On   | 00000000:85:00.0 Off |                    0 |
| N/A   54C    P0   115W / 300W |   8812MiB / 16160MiB |     89%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   5  NVIDIA Tesla V1...  On   | 00000000:86:00.0 Off |                    0 |
| N/A   54C    P0   105W / 300W |   8852MiB / 16160MiB |    100%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   6  NVIDIA Tesla V1...  On   | 00000000:89:00.0 Off |                    0 |
| N/A   58C    P0    88W / 300W |   8852MiB / 16160MiB |     96%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   7  NVIDIA Tesla V1...  On   | 00000000:8A:00.0 Off |                    0 |
| N/A   50C    P0   147W / 300W |   8852MiB / 16160MiB |     87%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
```

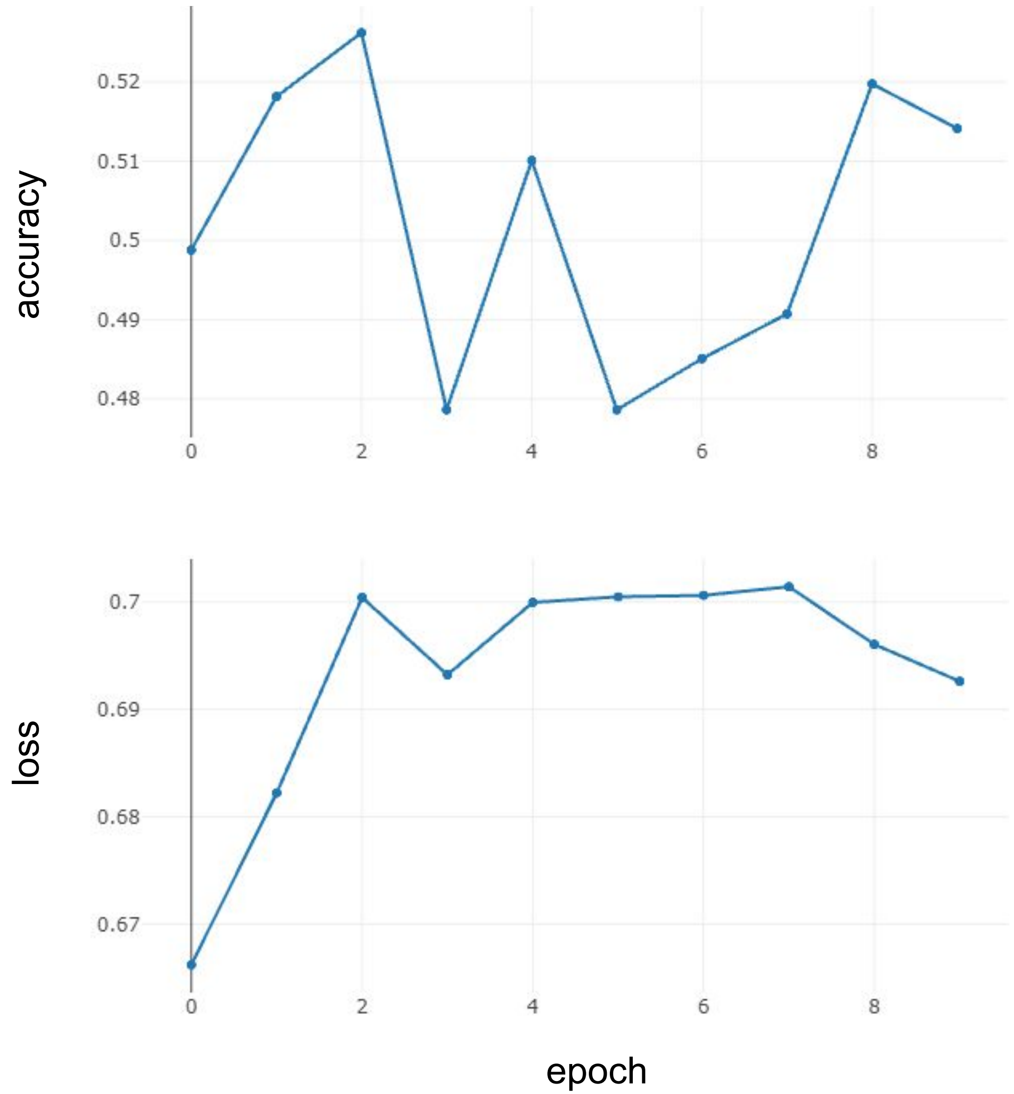
First Results - Vanilla3DCNN



First Results - ResNet18



First Results - ResNet18 (transfer learning)



Conclusion: First Results

- Evaluate all models using train/validation split
  Currently: Only check whether model can learn anything from the data

- Vanilla3DCNN works “unexpectedly” well! (at least on train data)
  Next: Early stopping and evaluate performance.

- State of the art 2D vision model ResNet18 is not able to extract useful information out of the provided data -> 3D domain adaptation: rather complicated

- ResNet18 (~30M) >> Vanilla3DCNN (~2M): Hint that fewer parameters may suffice

- Even pretrained ResNet18 seems to have problems with the data (training still in progress, expect update by next weekly)

Agenda

Wrap up: Next Steps

- Get access to “shared account” which is capable of training NN on LRZ AI System
  - Store deep learning lifecycle logging parameters (MLflow)
  - Export port to localhost for MLflow UI

- “[ERROR]: CUDA out of memory.
  - Find solution how to increase the batch size
  - Bigger batch size: Less training time