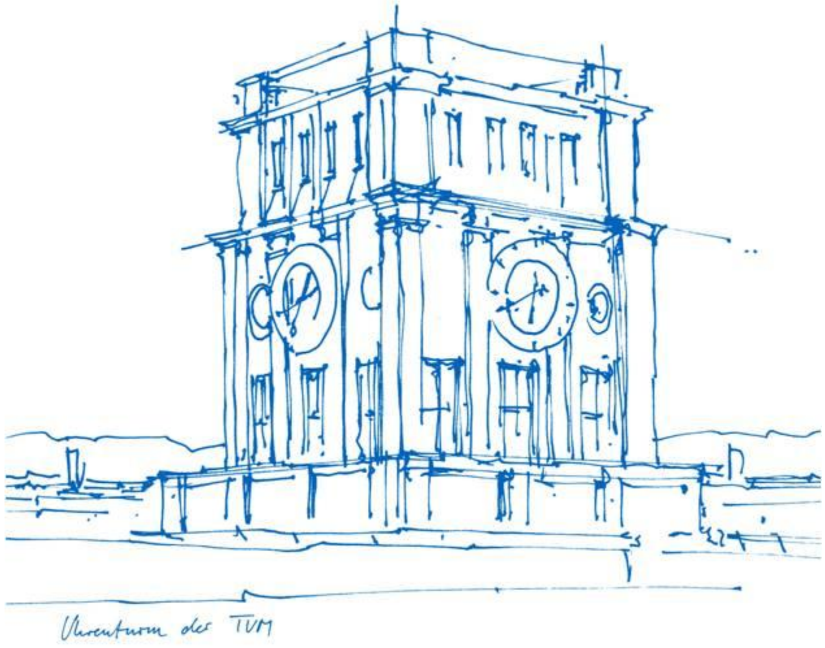
Geometrical Deep Learning on 3D Models: Classification for Additive Manufacturing

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

Technical University Munich

Faculty of Mathematics

22. July 2021



Motivation

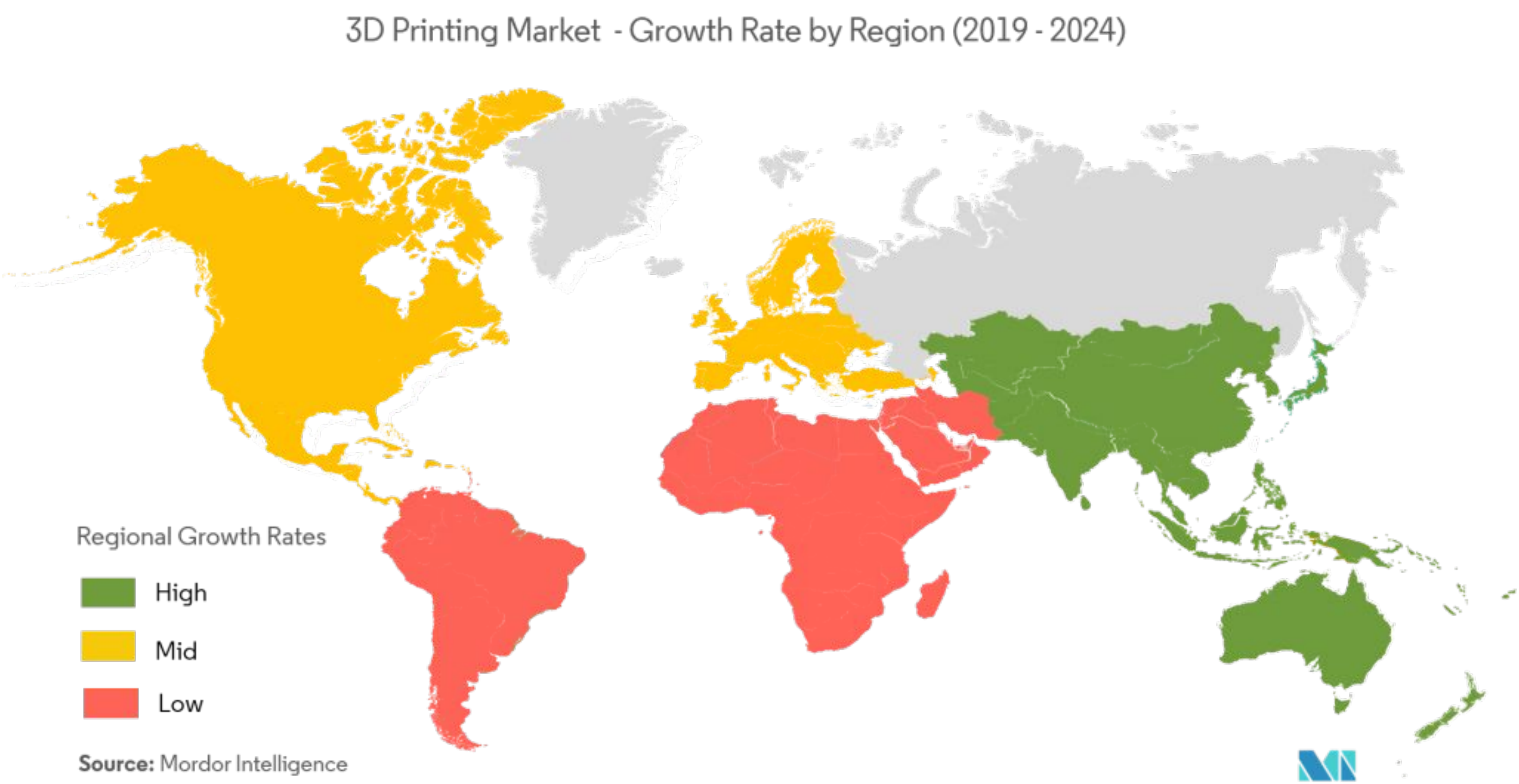
Hard to achieve the right balance:

Quality ⟷ Speed ⟷ Cost

Additive manufacturing here to transform production processes:

- create design iterations
- enhance quality through cost-effective prototyping
- create specific tooling parts

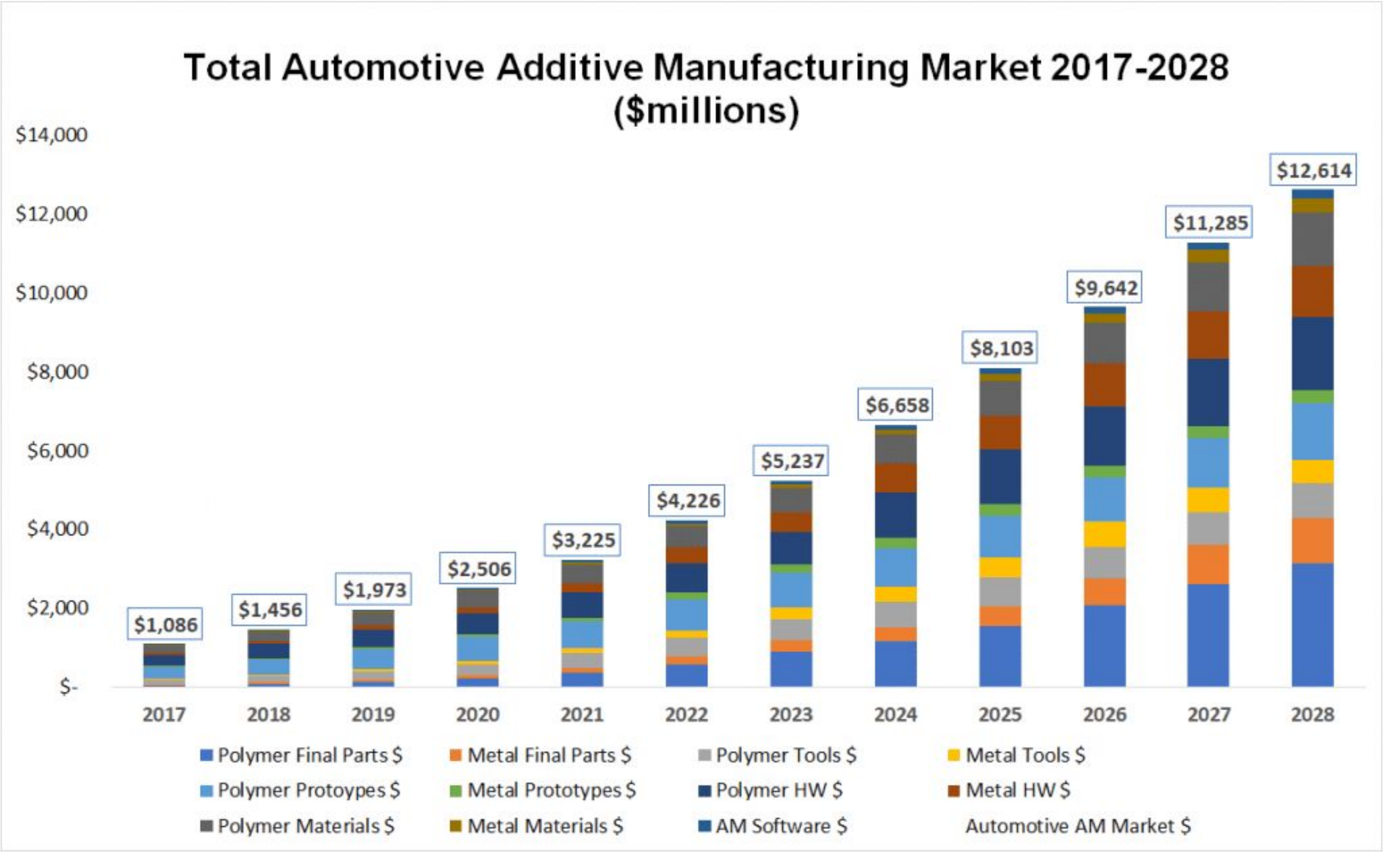
Motivation



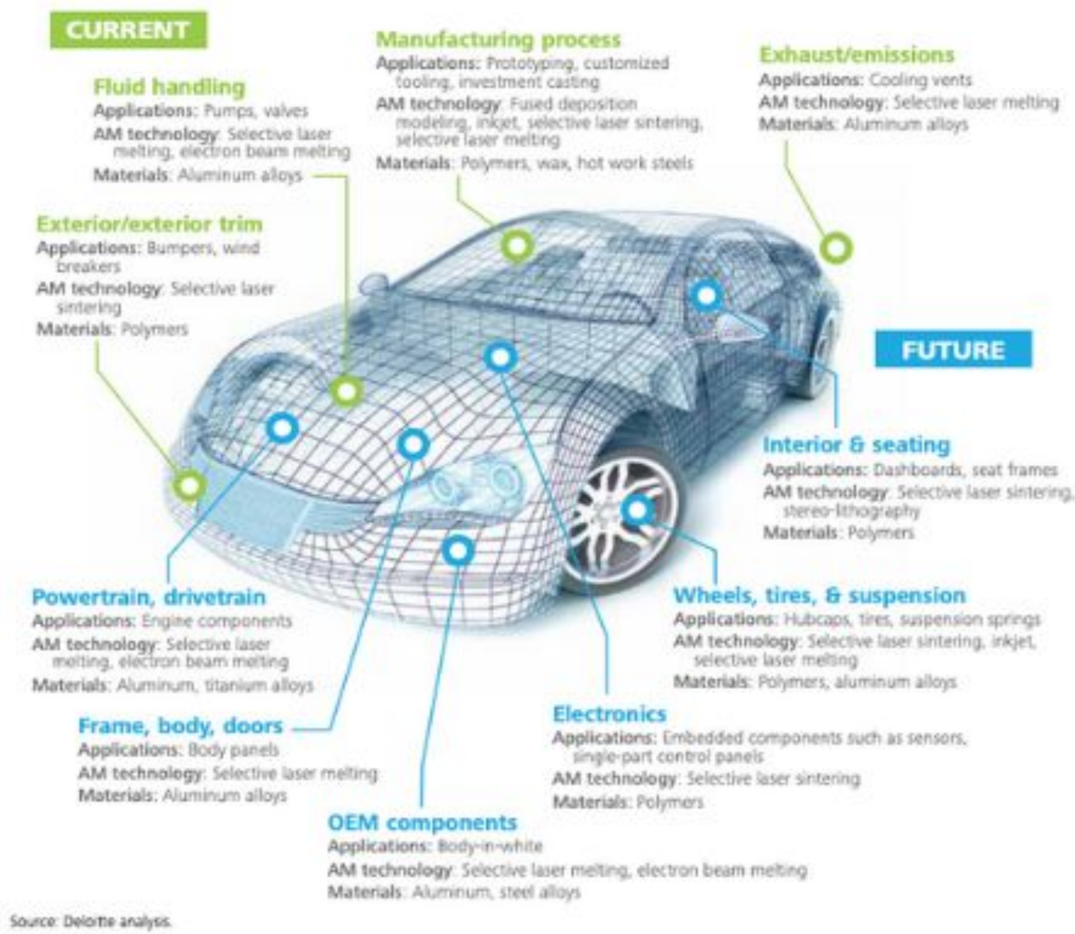
3D Printing Market - Growth Rate by Region (2019 - 2024)
Regional Growth Rates
High
Mid
Low
Source: Mordor Intelligence

Motivation

By 2028 AM industry revenues would hit **$12.6** Billion in the automotive Industry



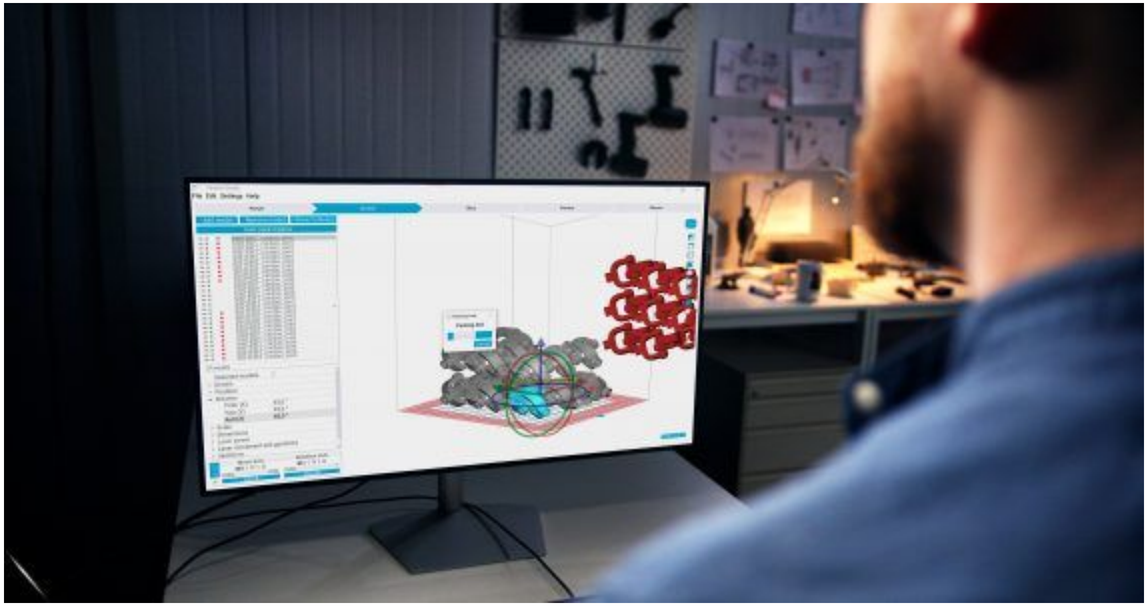
Motivation



Applications of AM in the automotive industry

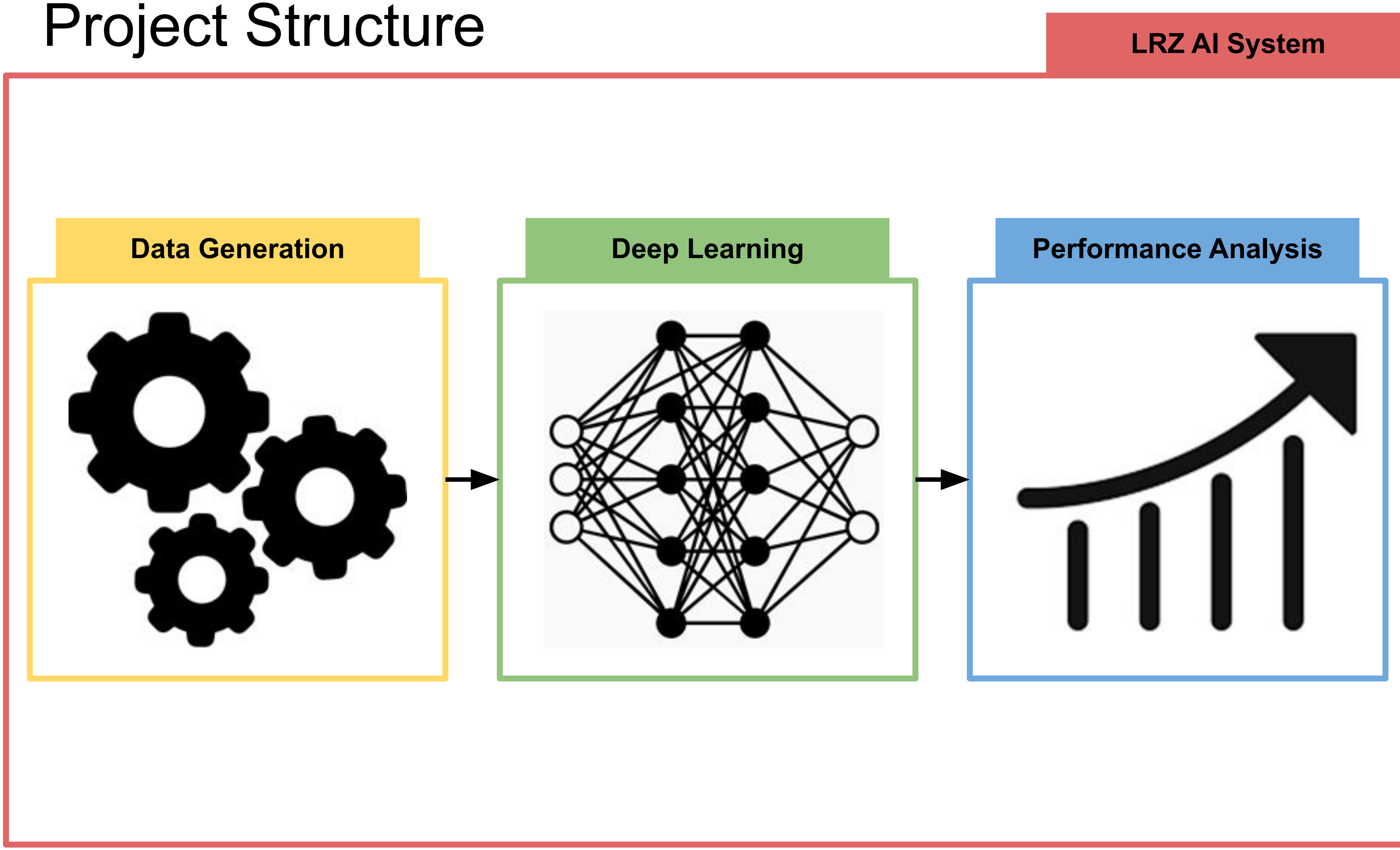
Problem Definition & Project Goal

Requires a lot of human expertise and supervision.
In particular, the process of **identifying whether a 3D model is manufacturable by a given 3D printer** is a **very time-consuming** and **complex task.**

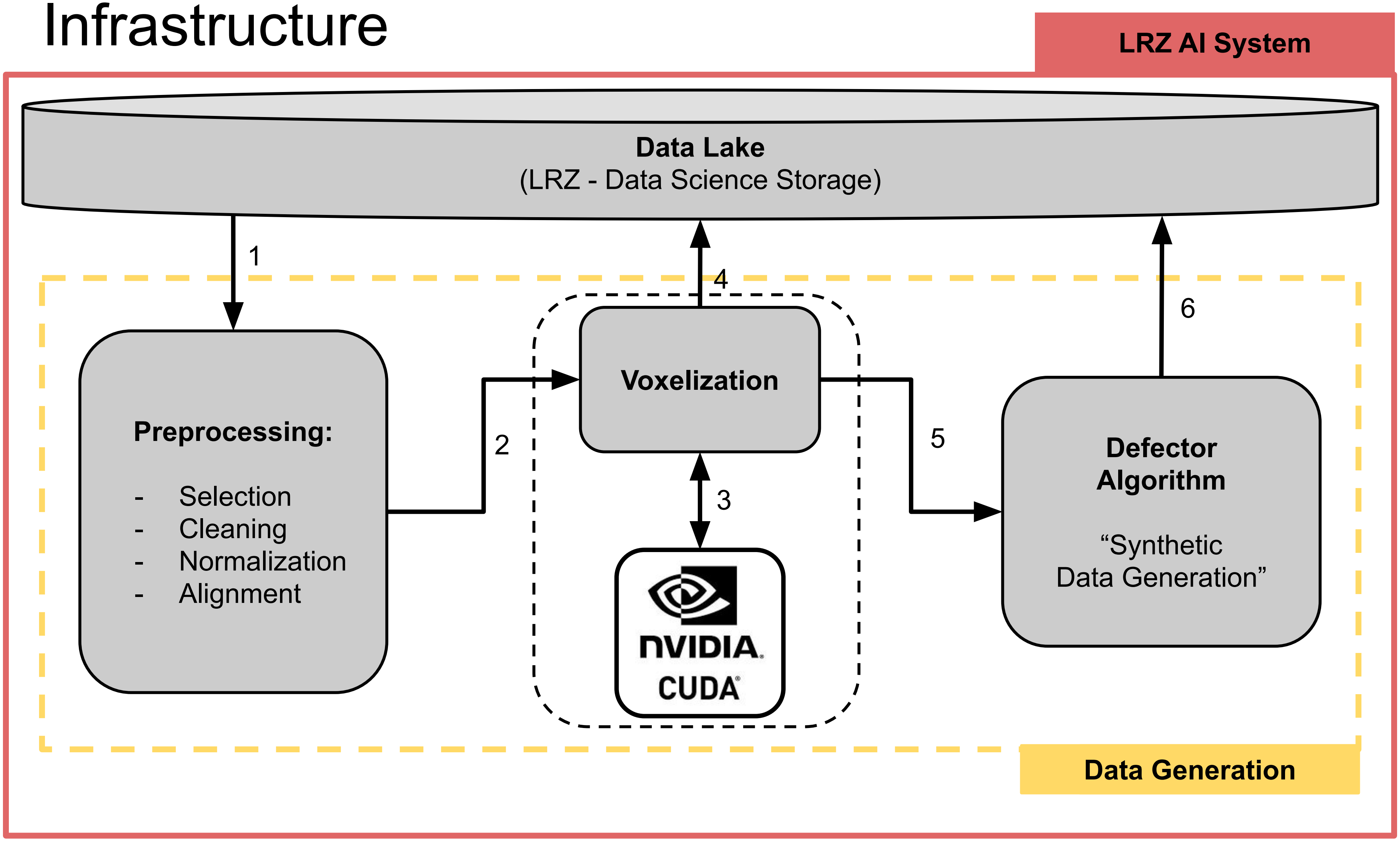


Automate this process using advanced **convolutional neural networks** (CNNs)

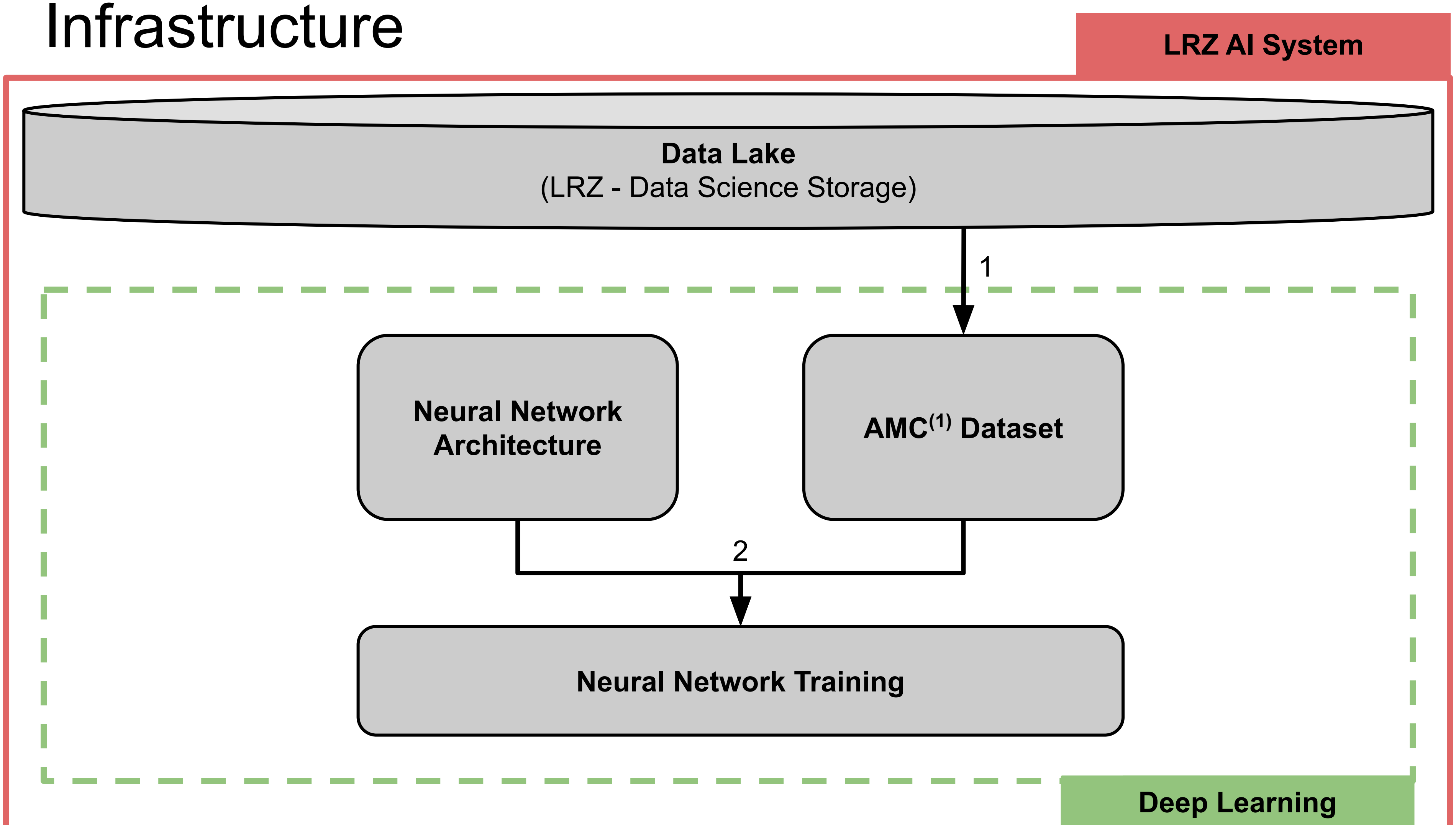
Project Structure



Infrastructure

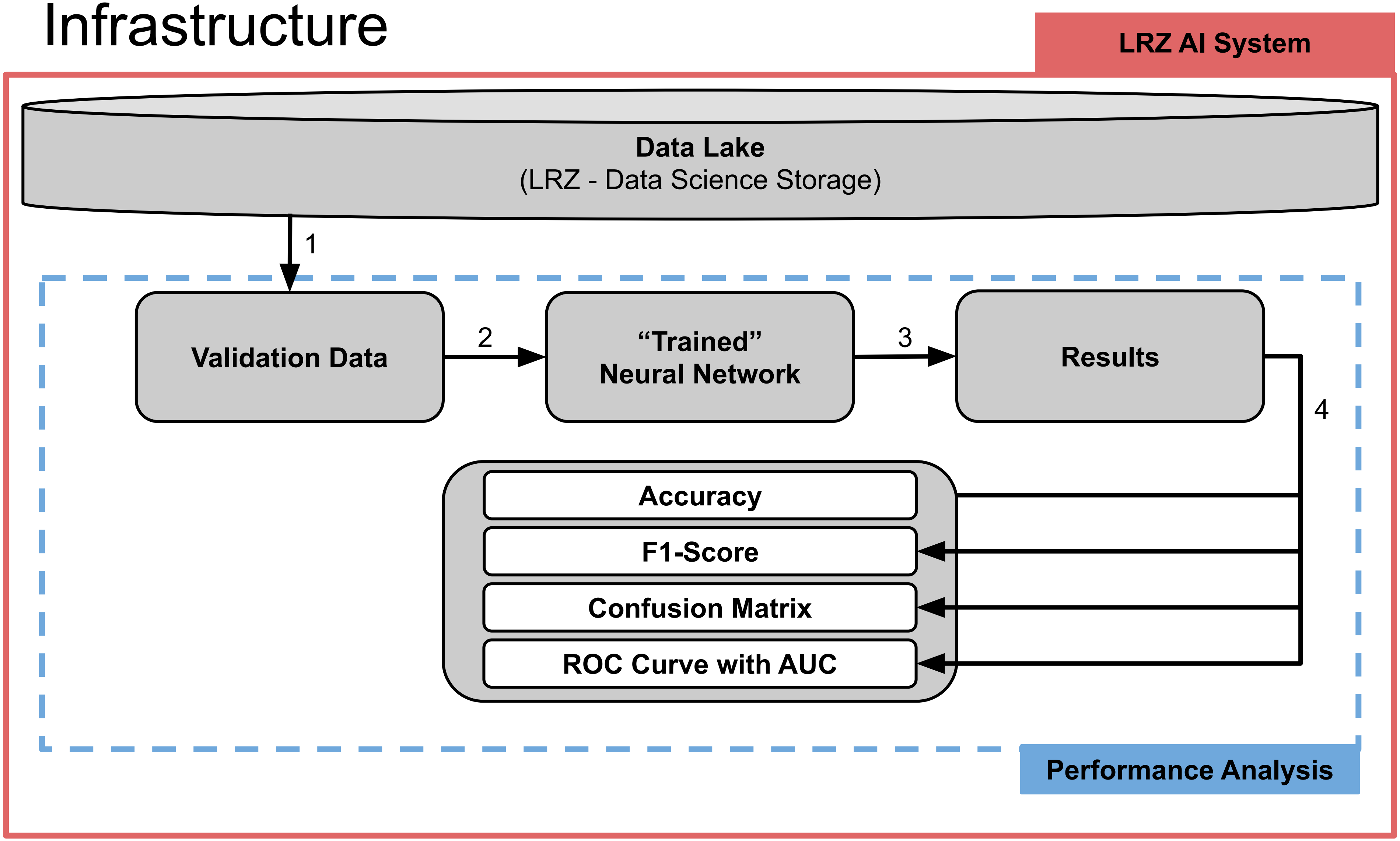


Infrastructure

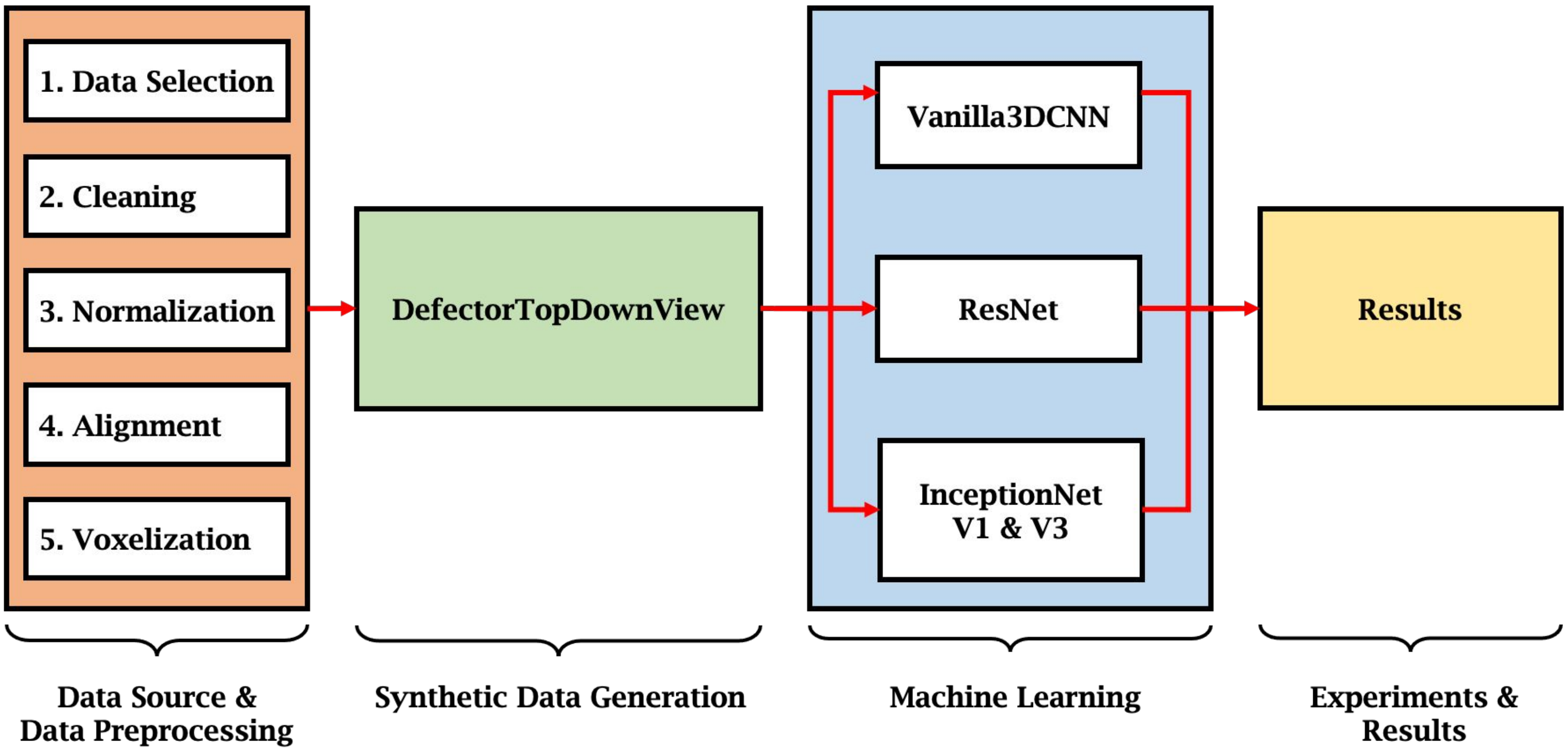


(1) AMC: Additive Manufacturing Classification

Infrastructure

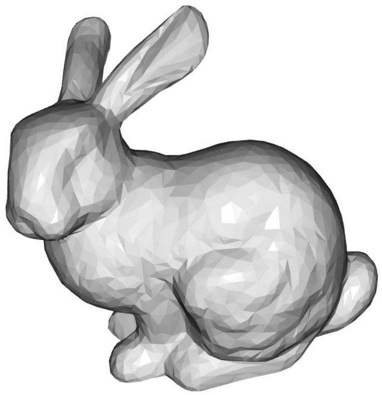


Holistic Project Overview

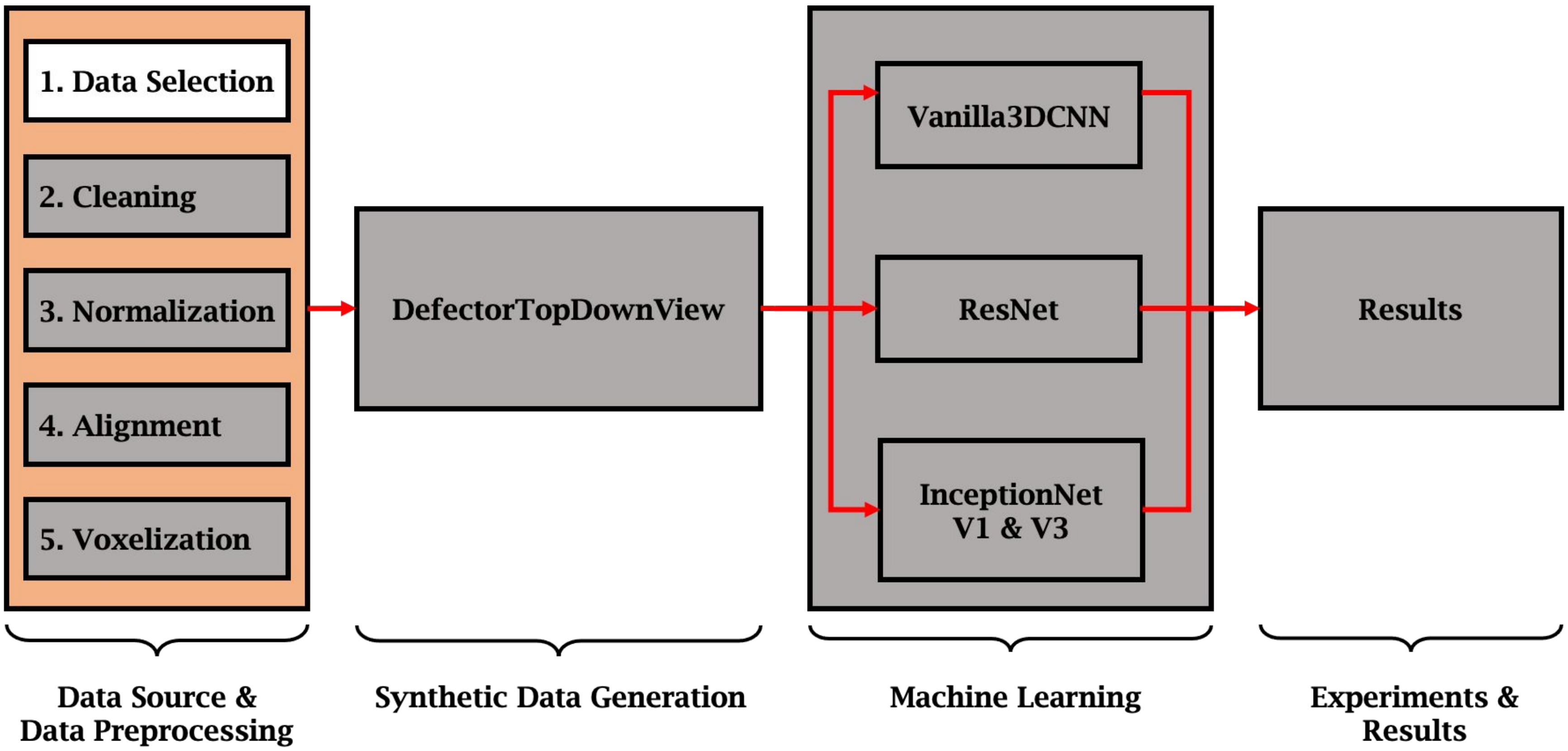


Data Format: Triangular Mesh

- A triangular mesh is a data representation of 3D objects.
- It consists of two main data structures: **Vertices** and **Triangles**.
- **Vertices** is a list of 3D points.
- **Triangles** is a list of triangles, where a triangle connects 3 vertices.

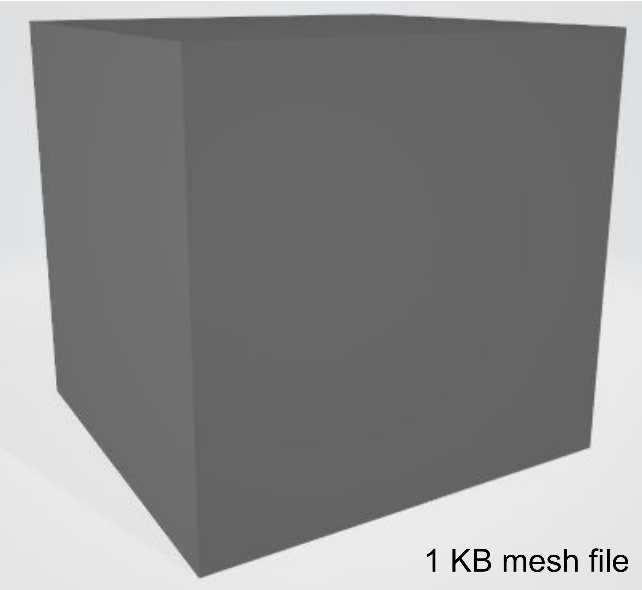


Triangular Mesh Representation

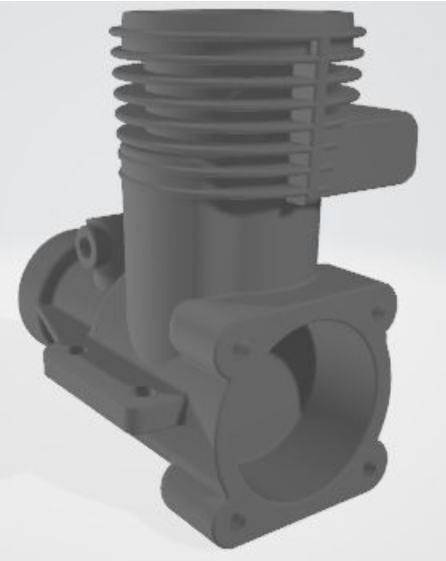

1. Data Selection
2. Cleaning
3. Normalization
4. Alignment
5. Voxelization
DefectorTopDownView
Vanilla3DCNN
ResNet
InceptionNet V1 & V3
Results
Data Source & Data Preprocessing
Synthetic Data Generation
Machine Learning
Experiments & Results

Data Selection

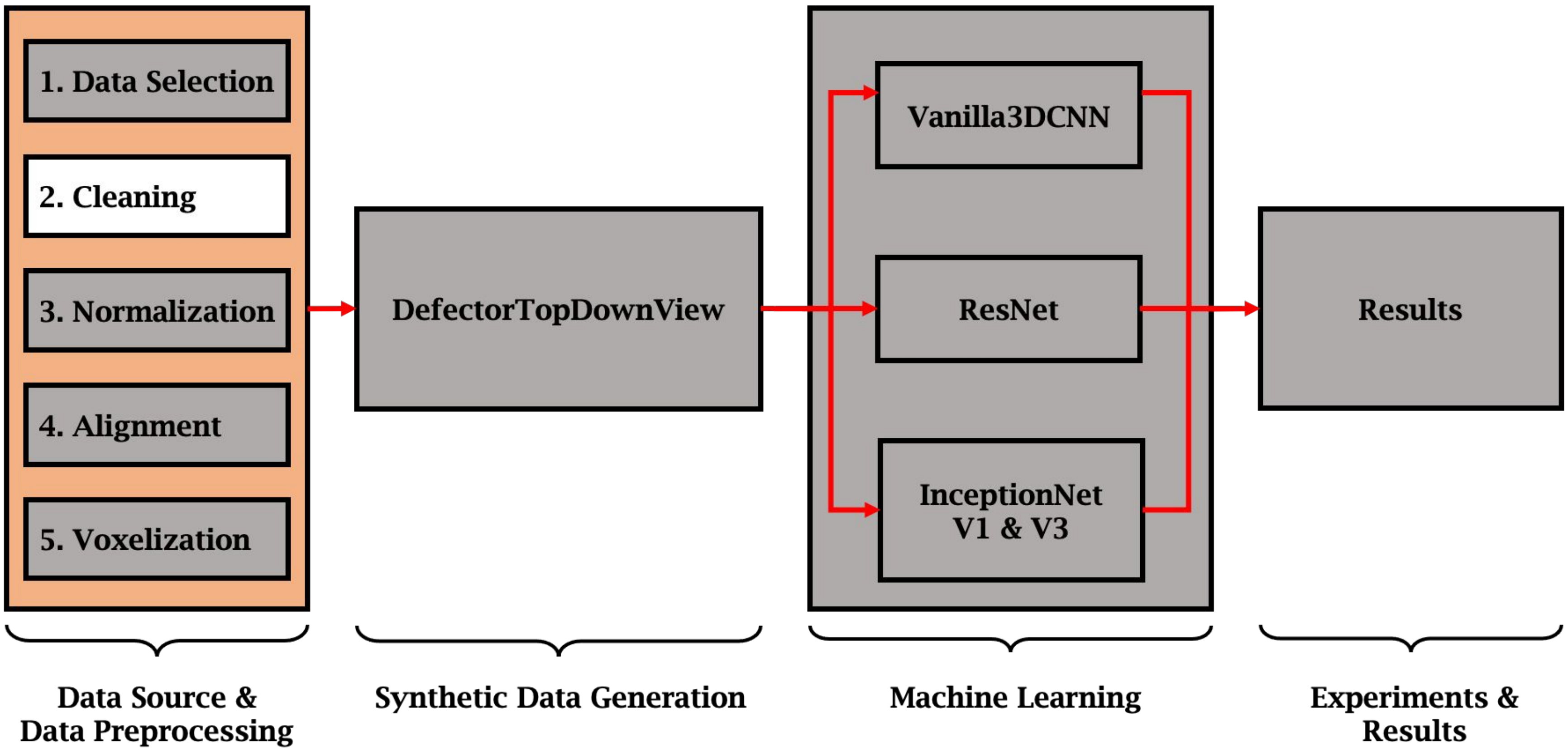
- dataset contains models of varying complexity
- more triangles are needed in meshes of higher complexity
- more triangles translated to bigger file size
- we applied a cutoff value of 25 KB to select models considered for further preprocessing steps



1 KB mesh file



32 MB mesh file

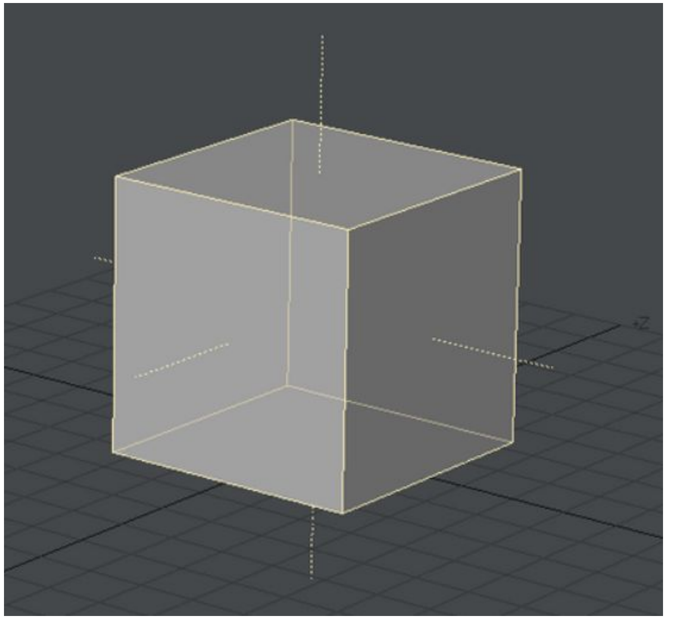


1. Data Selection
2. Cleaning
3. Normalization
4. Alignment
5. Voxelization
DefectorTopDownView
Vanilla3DCNN
ResNet
InceptionNet V1 & V3
Results
Data Source & Data Preprocessing
Synthetic Data Generation
Machine Learning
Experiments & Results

Cleaning

- For a 3D mesh to be 3D-printable, 2 main properties should be satisfied:
  - 1. **Watertightness**: mesh has no holes + normals are facing outwards

Stanford Bunny Model Bottom View
Invalid: has holes

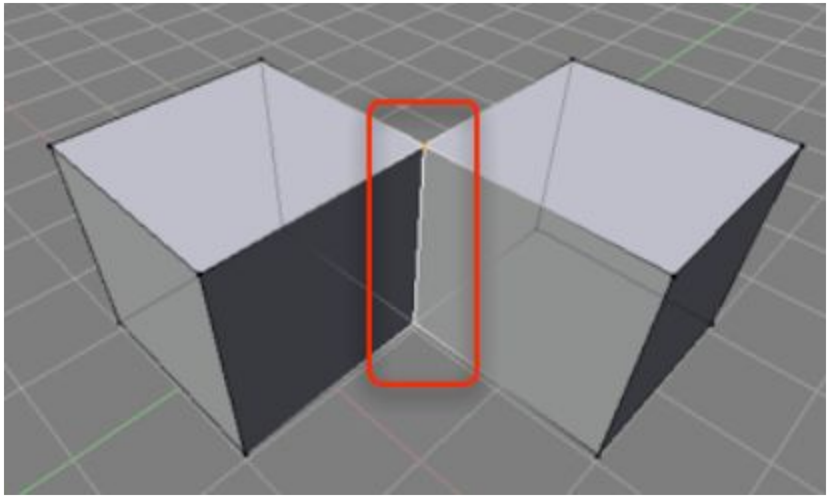


Valid: normals are facing outwards

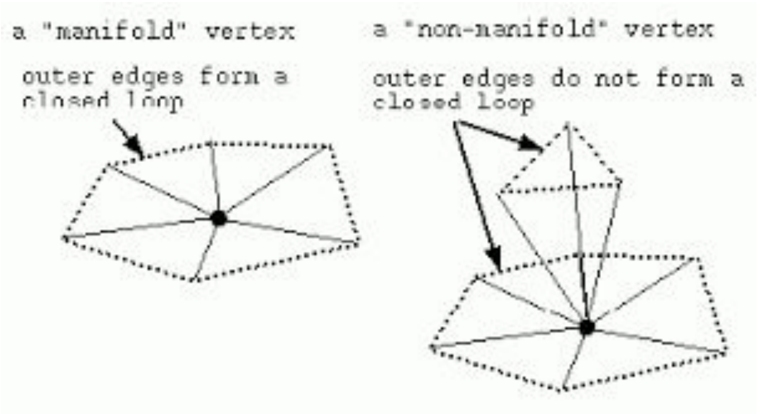
https://support.shapeways.com/hc/en-us/articles/360007107674-Tips-for-successful-modeling

Cleaning

- For a 3D mesh to be 3D-printable, 2 main properties should be satisfied
  - 2. **Manifold geometry**: mesh has no edges shared by more than two faces



Invalid: edge shared by 4 faces

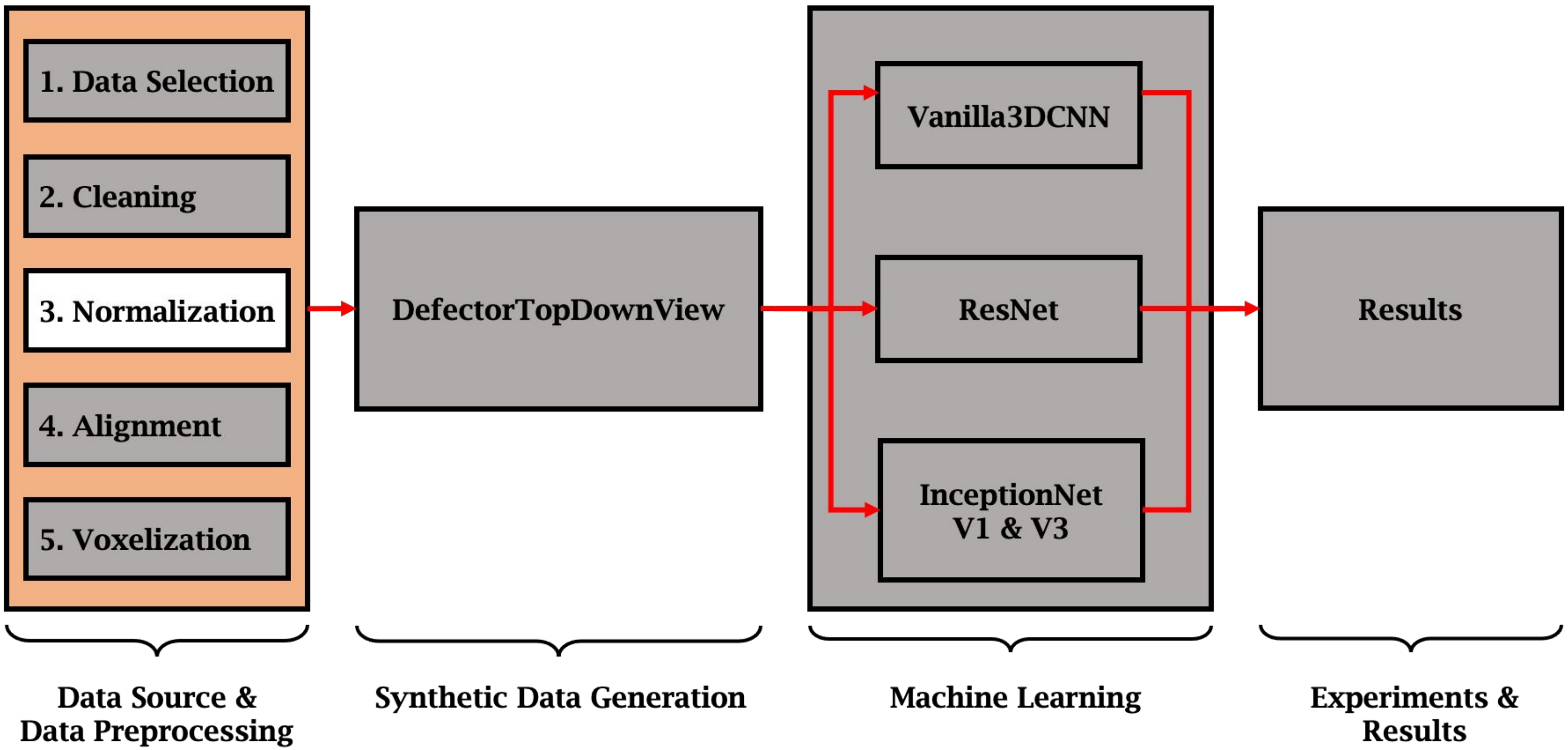


https://www.sculpteo.com/en/3d-learning-hub/create-3d-file/fix-non-manifold-geometry/

Cleaning

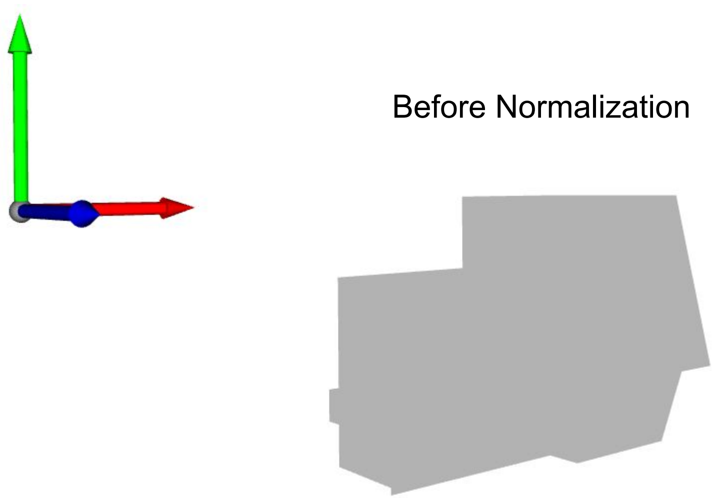
- Mesh cleaning functions were utilized from O3D. [1]
  - **Vertices fixes**
    - Remove vertices that have identical coordinates ([*x1*, *y1*, *z1*], [*x1*, *y1*, *z1*])
    - Remove vertices that are not referenced in any triangle
  - **Edges fixes**
    - Remove non-manifold edges
  - **Triangles fixes**
    - Remove triangles that reference the same three vertices ([*v1*, *v2*, *v3*], [*v2*, *v1*, *v3*])
    - Remove triangles that reference a single vertex multiple times in a single triangle ([*v1*, *v2*, *v2*])

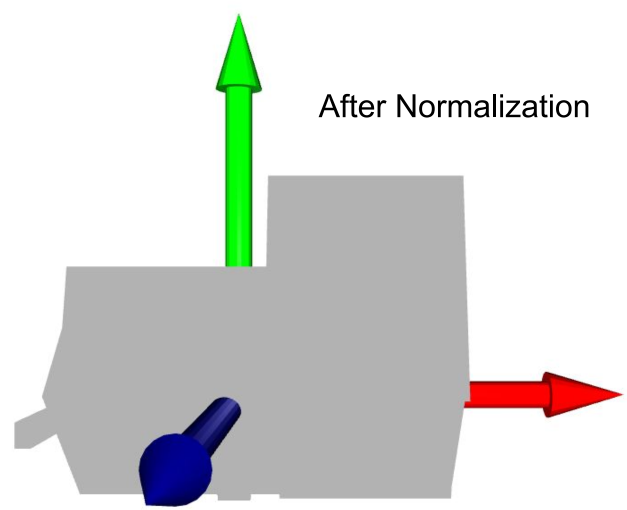
[1] http://www.open3d.org/docs/release/index.html

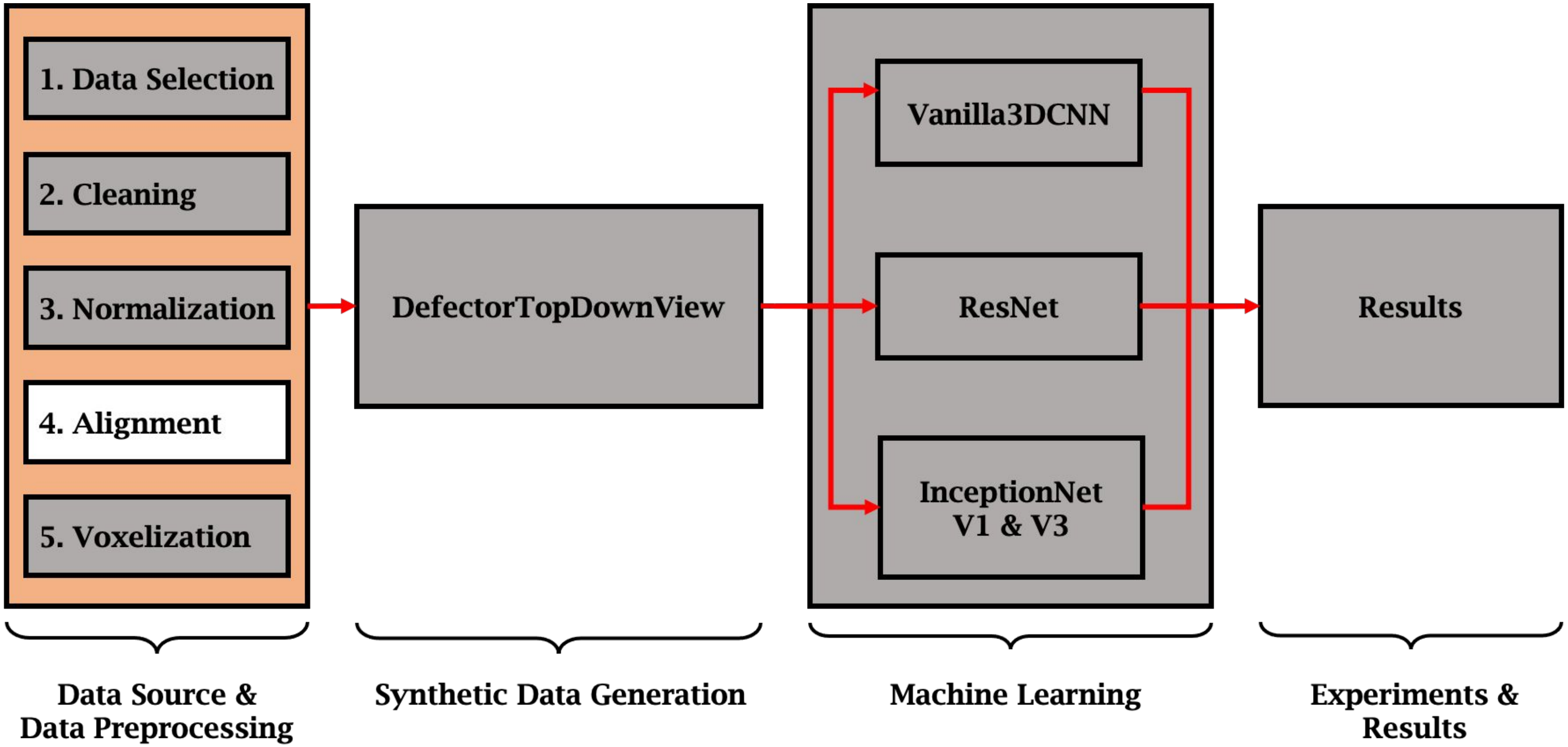

1. Data Selection
2. Cleaning
3. Normalization
4. Alignment
5. Voxelization
DefectorTopDownView
Vanilla3DCNN
ResNet
InceptionNet V1 & V3
Results
Data Source & Data Preprocessing
Synthetic Data Generation
Machine Learning
Experiments & Results

Normalization

- Performed to make sure all vertices of different objects lie in the same range
- **Step 1**: Center the mesh around the origin
  - Find the center of the mesh vertices
  - Translate the mesh vertices to the origin by subtracting the center from all vertices
- **Step 2**: Scale the vertices so that they lie in a [-1, 1] range
  - Divide the mesh vertices by the difference between the maximum bounding point and the minimum bounding point.


Before Normalization


After Normalization


1. Data Selection
2. Cleaning
3. Normalization
4. Alignment
5. Voxelization
DefectorTopDownView
Vanilla3DCNN
ResNet
InceptionNet V1 & V3
Results
Data Source & Data Preprocessing
Synthetic Data Generation
Machine Learning
Experiments & Results

VOLKSWAGEN DATA:LAB

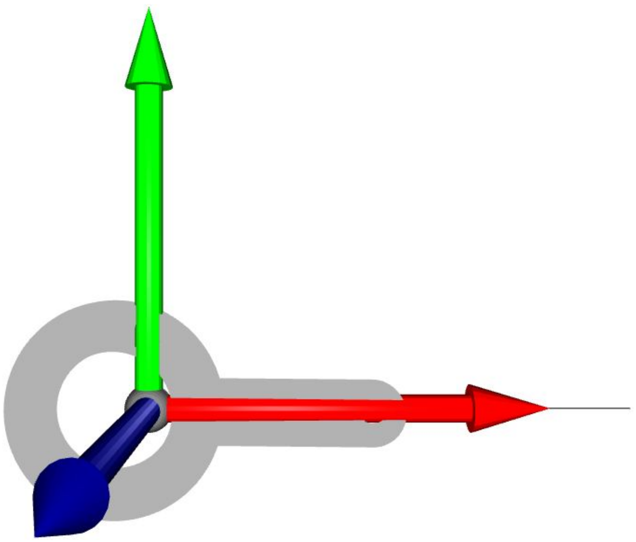
Alignment

- Performed to make sure all meshes are presented in the same orientation.
- The axis of the minimum Moment Of Inertia (MOI) of a mesh represents the axis around which most of the mass of the object is wrapped. [1]
- Aligning the axis of the minimum MOI with one of the coordinate axes will allow meshes to be presented in the same orientation
- How its done [2] & [3]:
    - **Step 1**: Find the axis of the minimum MOI
    - **Step 2**: Compute a rotation matrix that aligns the axis of the minimum MOI with a coordinate axis
    - **Step 3**: Apply the rotation matrix on the mesh vertices

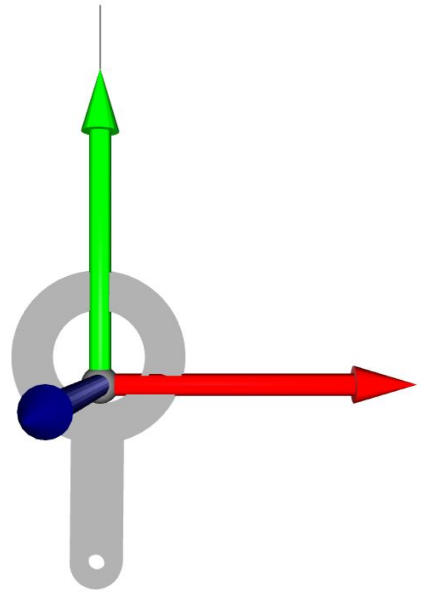
[1] James Dann and James J. Dann. The People's Physics Book. third edition, 2006.
[2] https://physics.stackexchange.com/questions/426273/how-to-find-the-axis-with-minimum-moment-of-inertia
[3] https://stackoverflow.com/questions/67017134/find-rotation-matrix-to-align-two-vectors

VOLKSWAGEN DATA:LAB

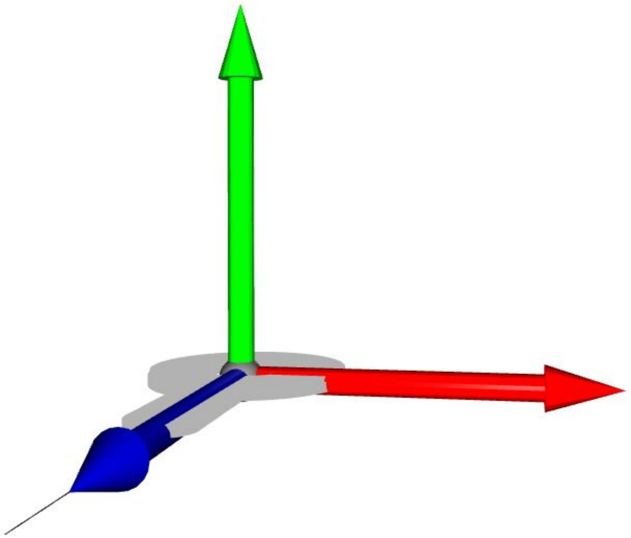
Alignment



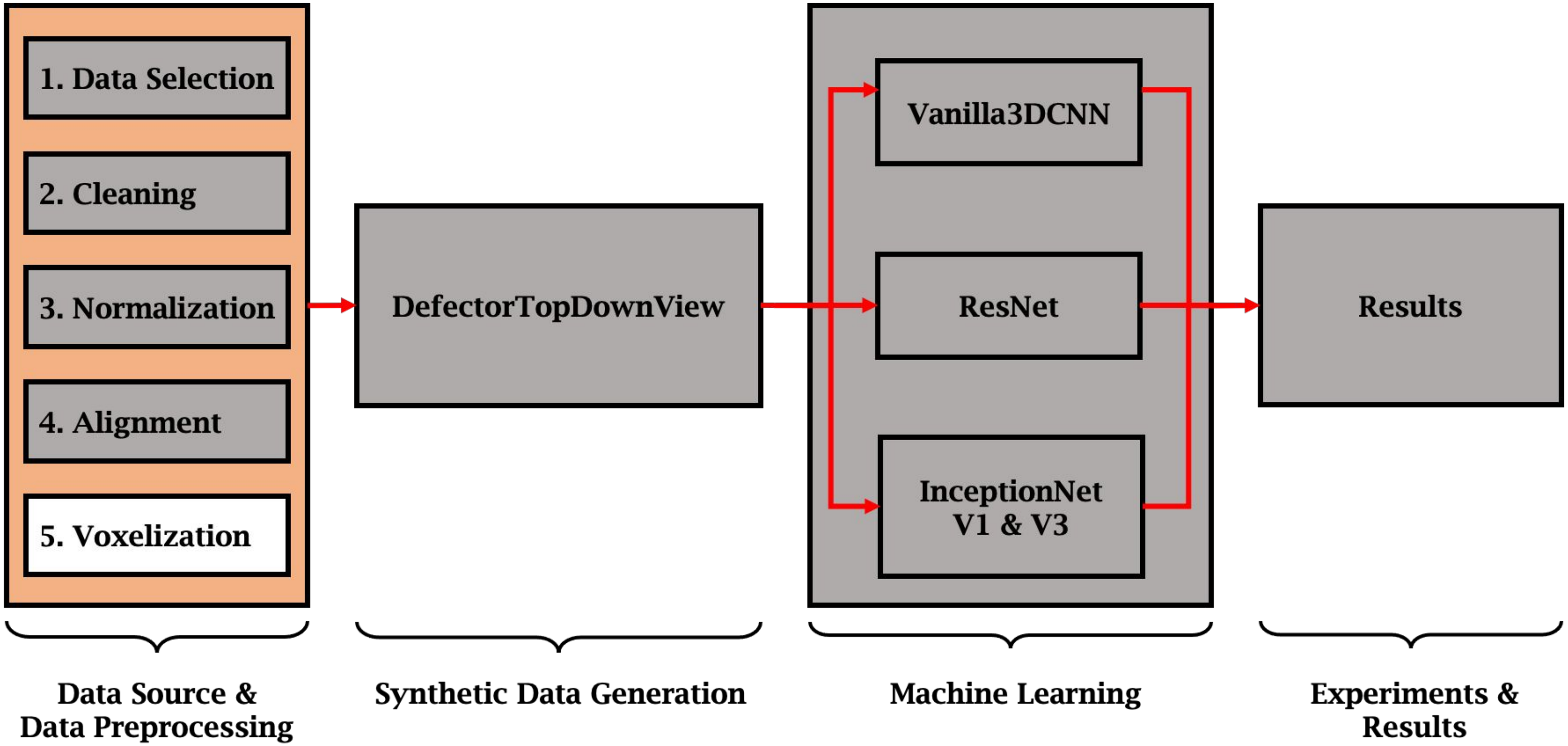
Alignment of min MOI axis and X-axis



Alignment of min MOI axis and Y-axis

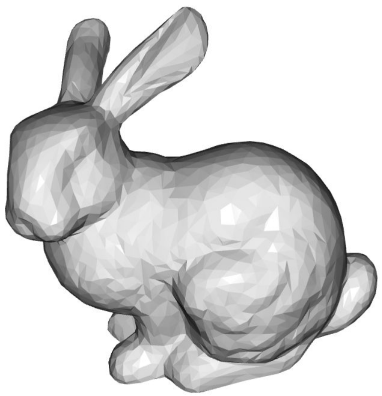


Alignment of min MOI axis and Z-axis

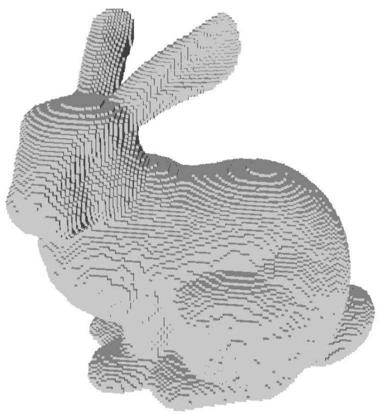

1. Data Selection
2. Cleaning
3. Normalization
4. Alignment
5. Voxelization
DefectorTopDownView
Vanilla3DCNN
ResNet
InceptionNet
V1 & V3
Results
Data Source &
Data Preprocessing
Synthetic Data Generation
Machine Learning
Experiments &
Results

Voxelization

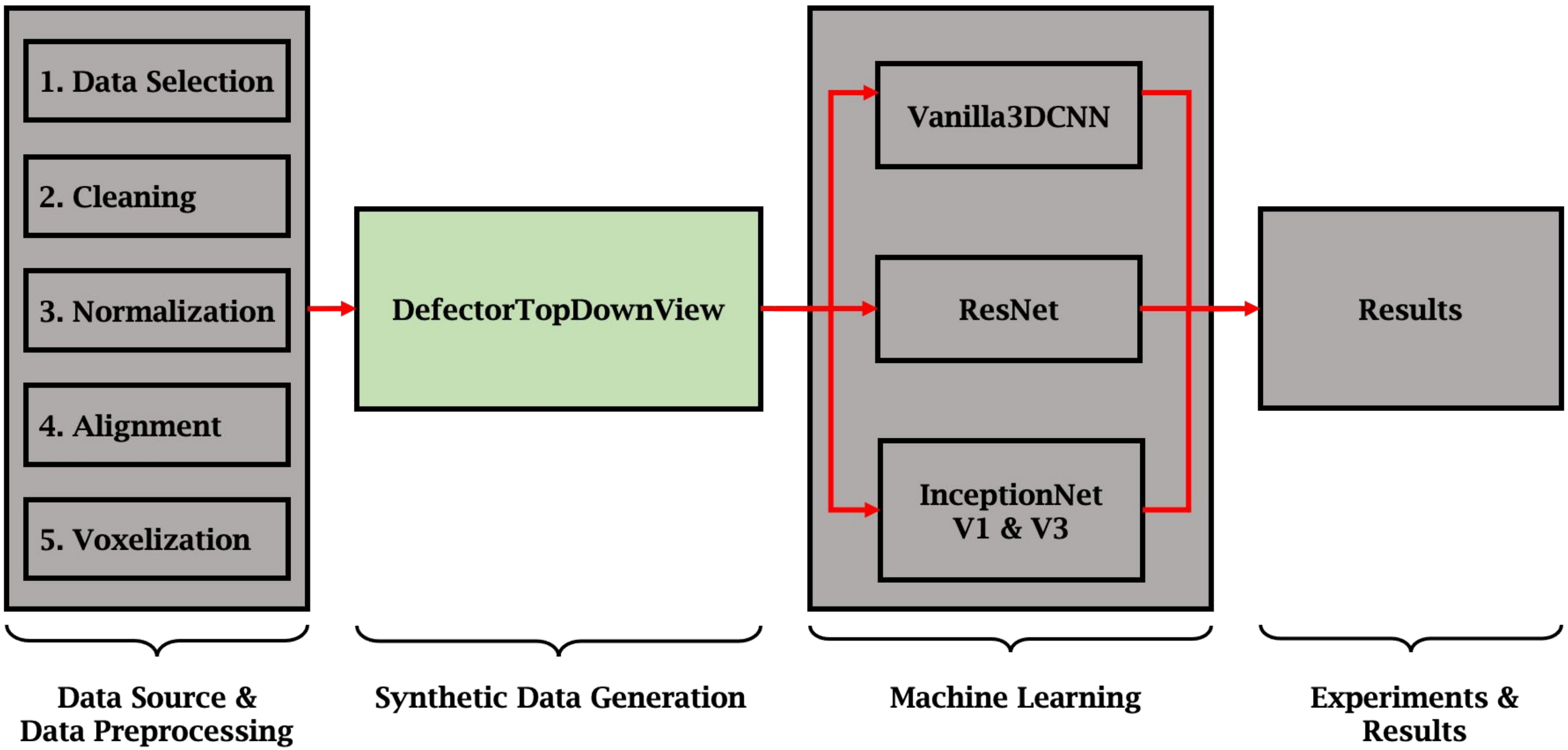
- All previous transformations were applied on the triangular mesh representation
- Final preprocessing step is converting the model to a voxelized representation
- What is a voxel representation?
  - A data representation that uses voxels. A voxel can be regarded as a pixel in a three-dimensional space.
- Why use voxel representation?
  - We hypothesized that SOTA CNNs will offer good performance after replacing 2D convolutions with 3D convolutions
  - It allows for an easy introduction of defects



Triangular Mesh Representation



Voxelized Representation

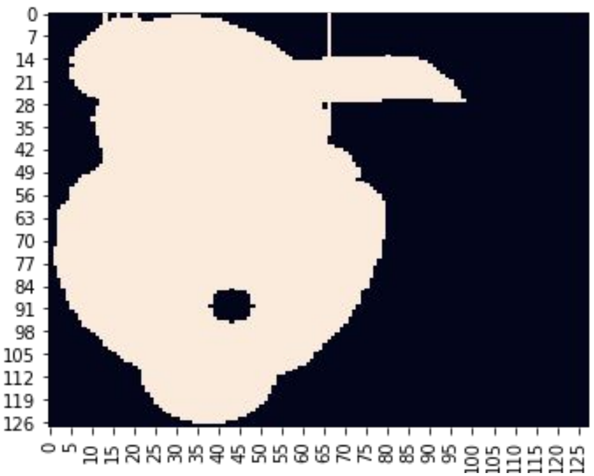

1. Data Selection
2. Cleaning
3. Normalization
4. Alignment
5. Voxelization
DefectorTopDownView
Vanilla3DCNN
ResNet
InceptionNet
V1 & V3
Results
Data Source &
Data Preprocessing
Synthetic Data Generation
Machine Learning
Experiments &
Results

Assumptions & Definitions 1/2

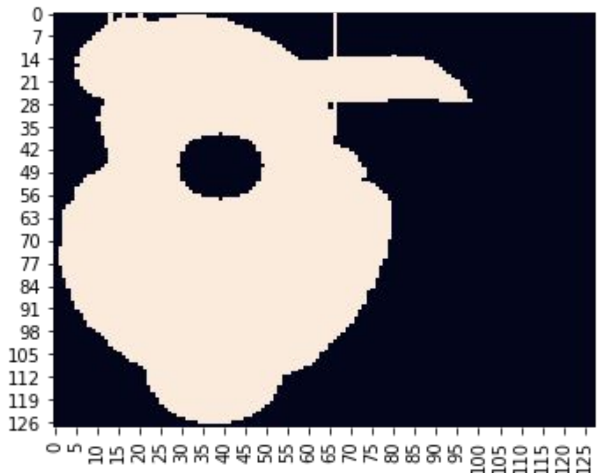
- **Problem:** Limited labeled data
- **Goal**: Synthetically add defects to 3D models in order to generate non-printable models
- **Assumption**:
  - All previous selected models are printable
  - Defects are here **holes aligned with the z-axis** that are added to the 3D model
  - Deciding parameters:
    - **Radius**
      - radius printable (10 voxels)
      - radius non-printable (5 voxels)
    - **Border**
      - Border printable (5 voxels)
      - Border non-printable (3 voxels)

Assumptions & Definitions 2/2

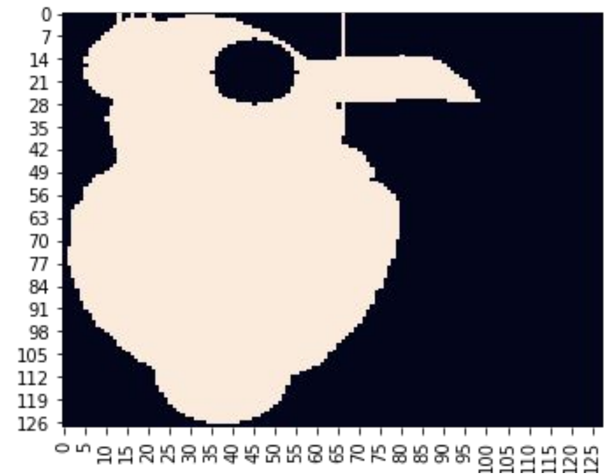
Resulting models:



Model **non-printable** defect in middle



Model **printable** defect in middle



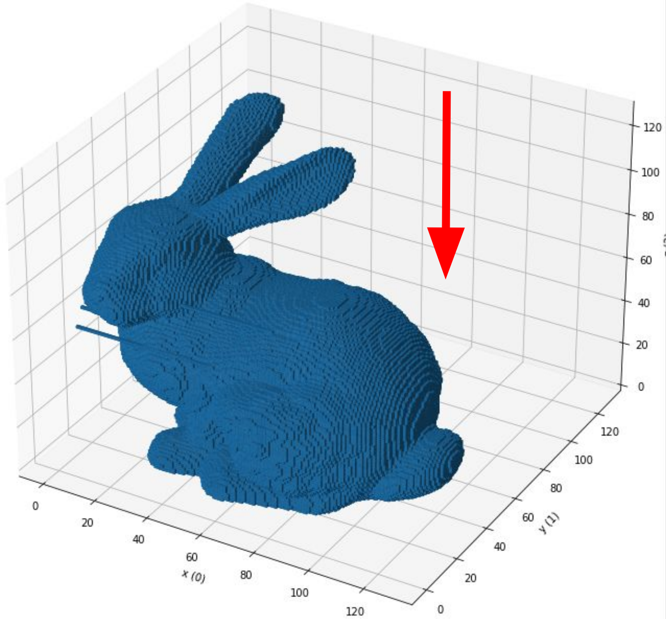
Model **non-printable** defect at border

Idea

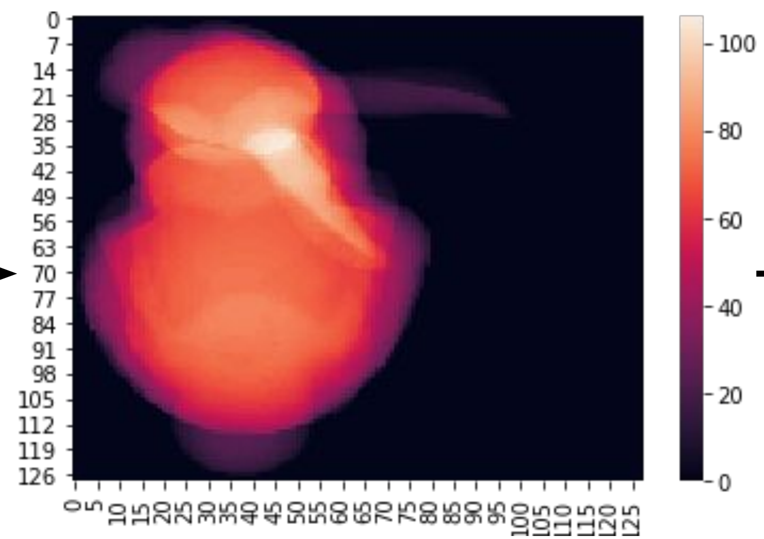
- **Main Idea: Transform 3D model data** s.t. it can be used to find right offset

→ **TopDownView**:

- Inspiration: **Heatmaps** (visualizing 3D data in 2D), **common sense**
- **Project 3D model data onto the (x, y)-grid**

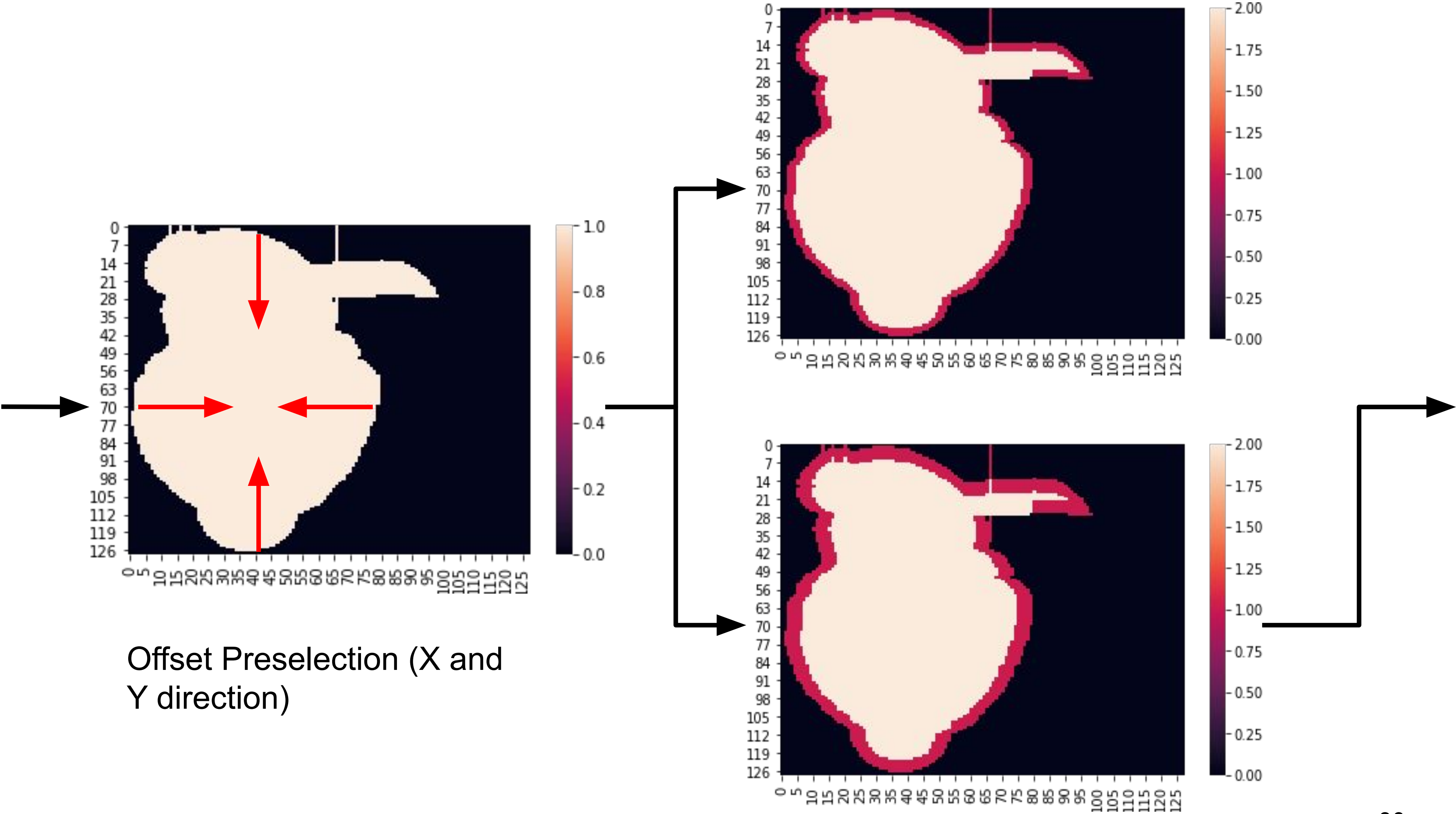


Summation over the z-axis



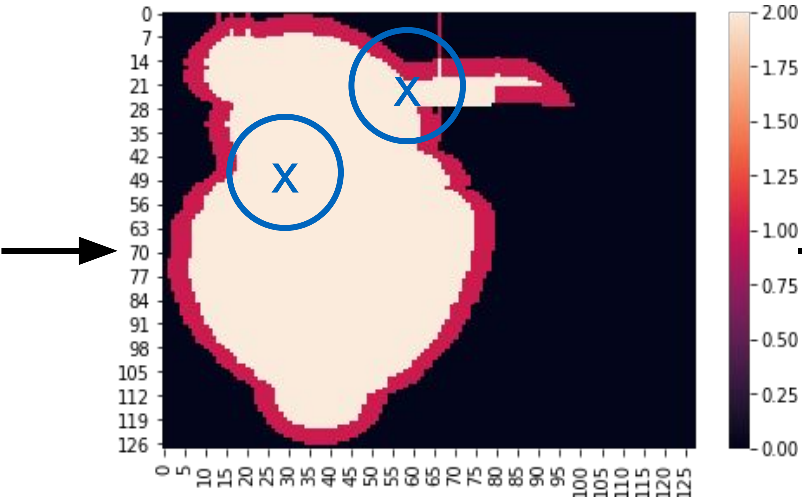
TopDownView

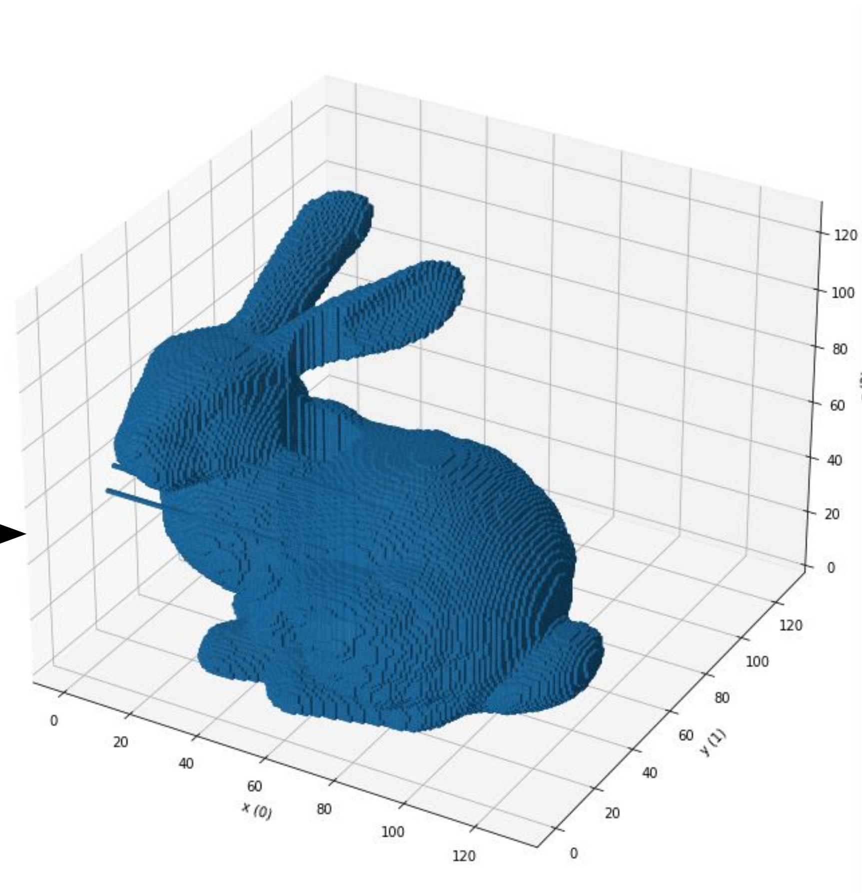
Defects in the middle 1/2



Offset Preselection (X and Y direction)

Defects in the middle 2/2

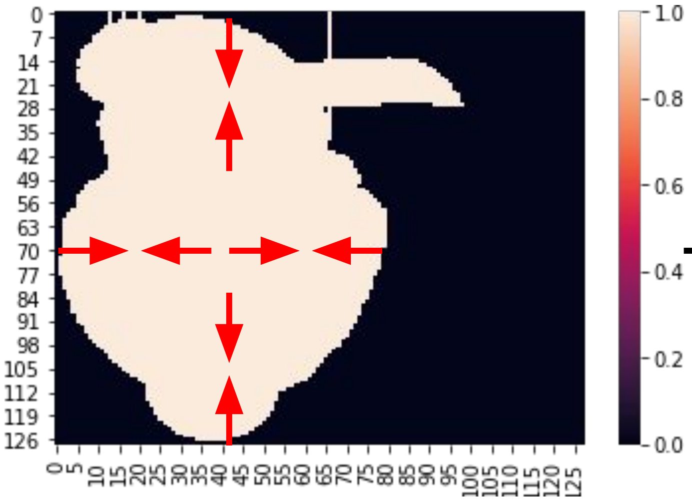




Non-axis direction check

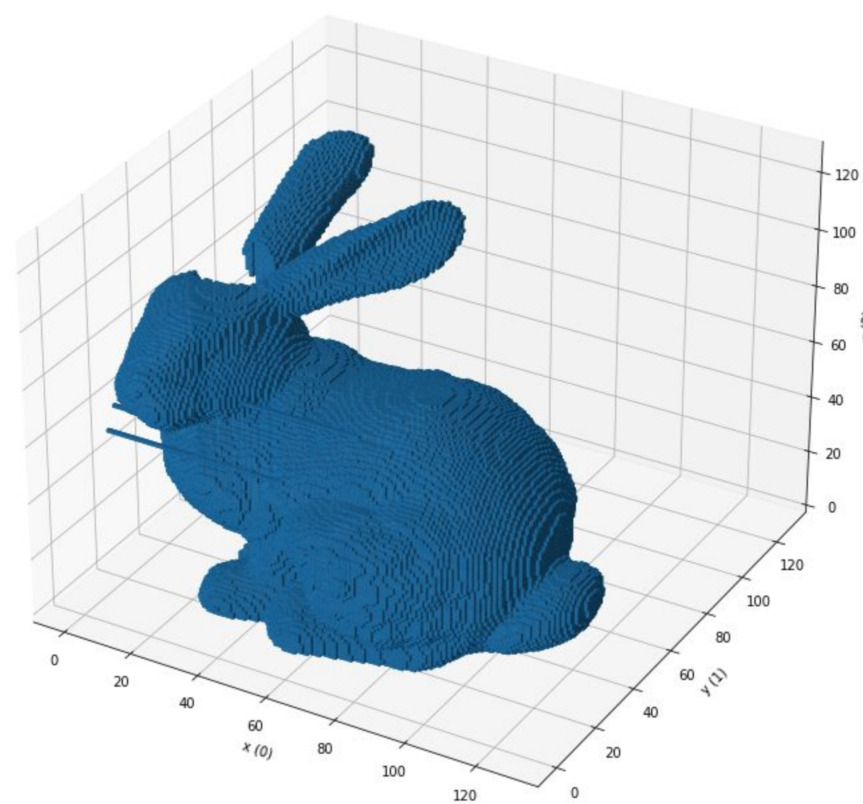
VOLKSWAGEN DATA:LAB

Defects at the border 1/2



Offset Preselection (X and Y direction)

Defects at the border 2/2

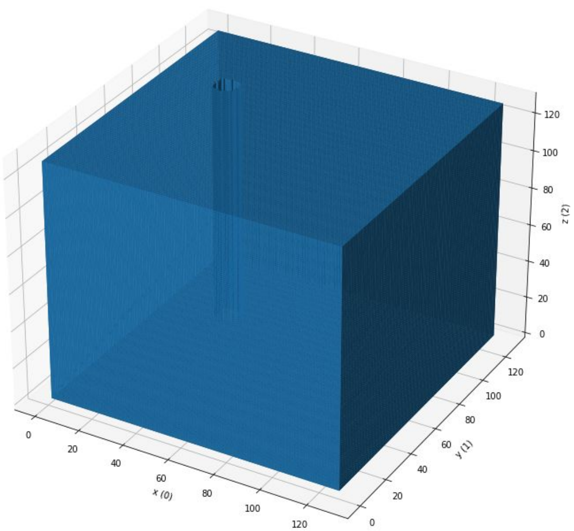


Non-axis direction check

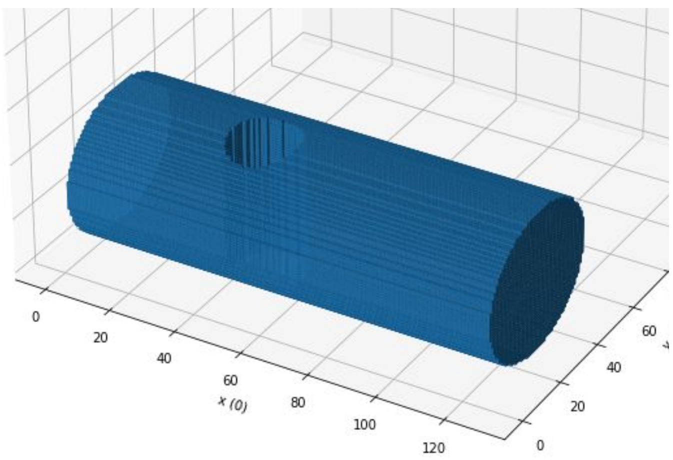
AMC Dataset

→ **Additive Manufacturing Classification (AMC) Dataset:**

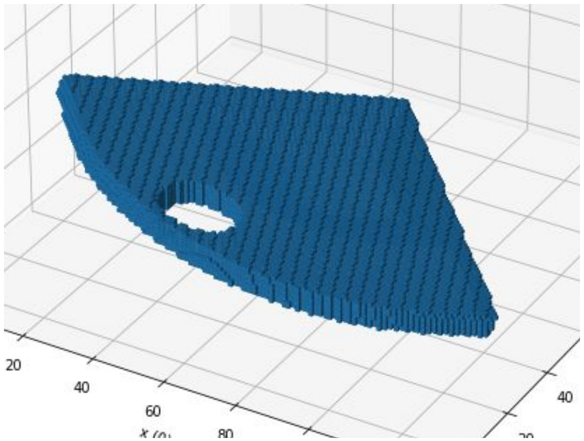
- **7430** 3D models
- **Balanced in terms of labels (printable / non-printable)**
- But: **Defects not equally distributed**



Non-printable defect middle



Printable defect middle

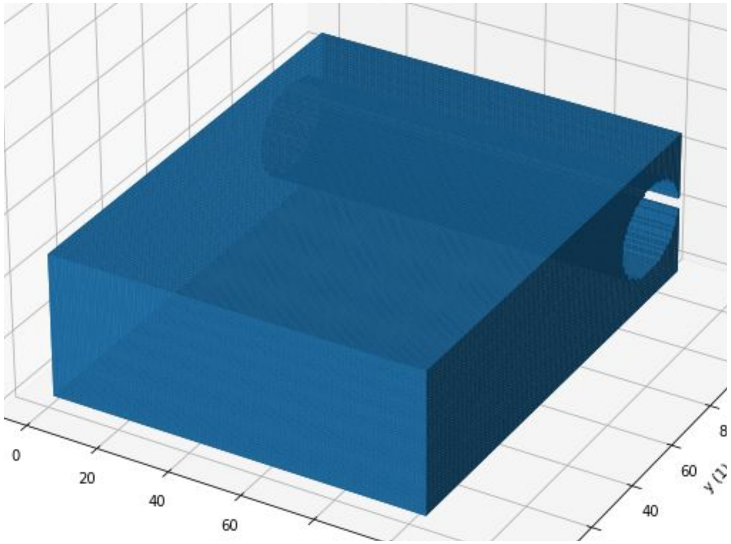


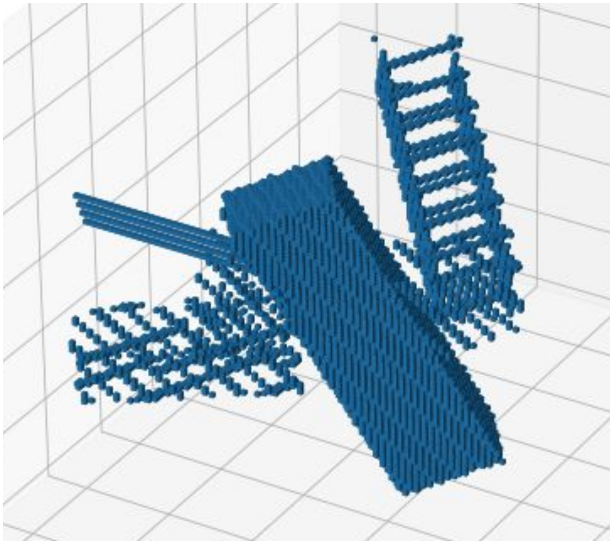
Non-printable defect border

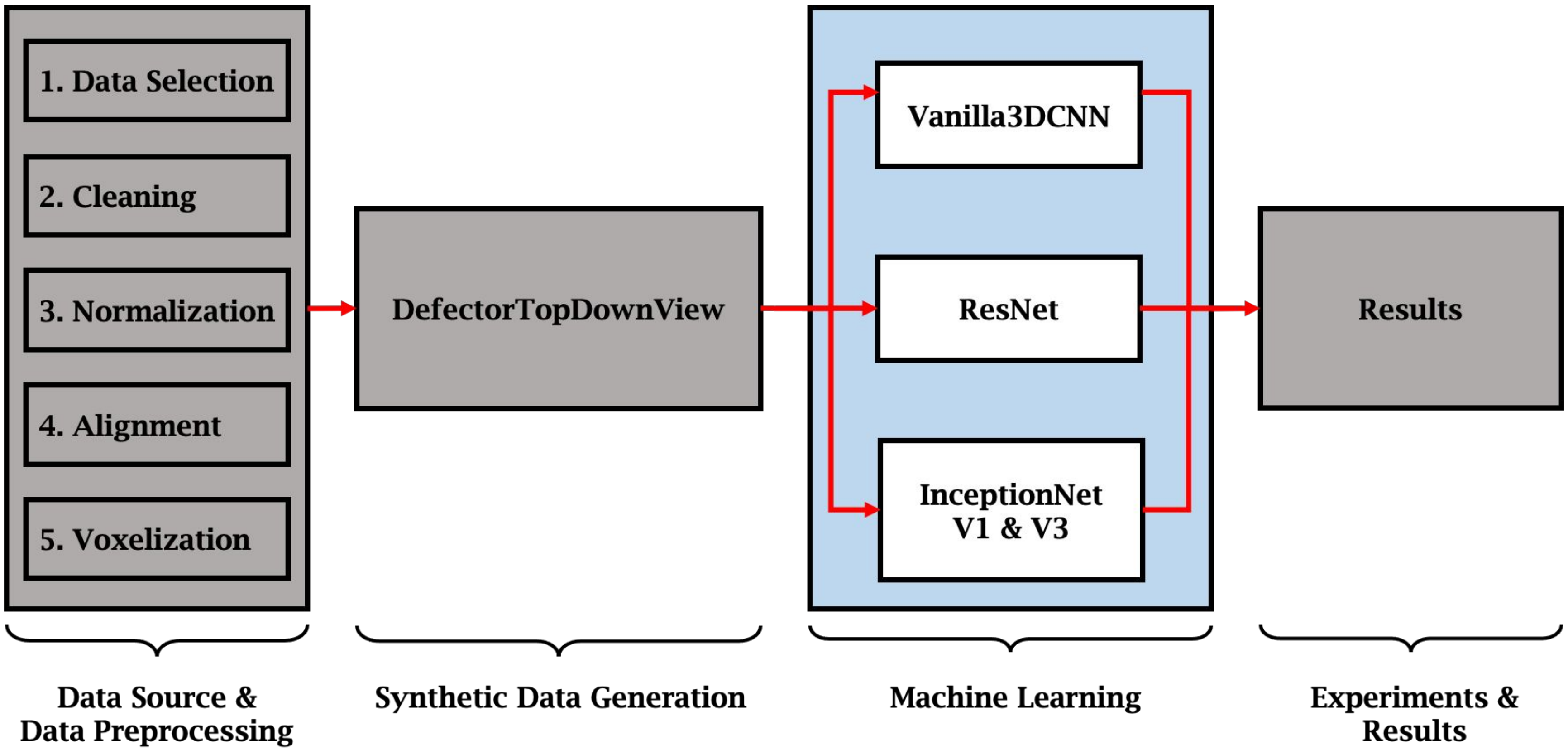
Limitations

- **Strong assumptions** on the input models

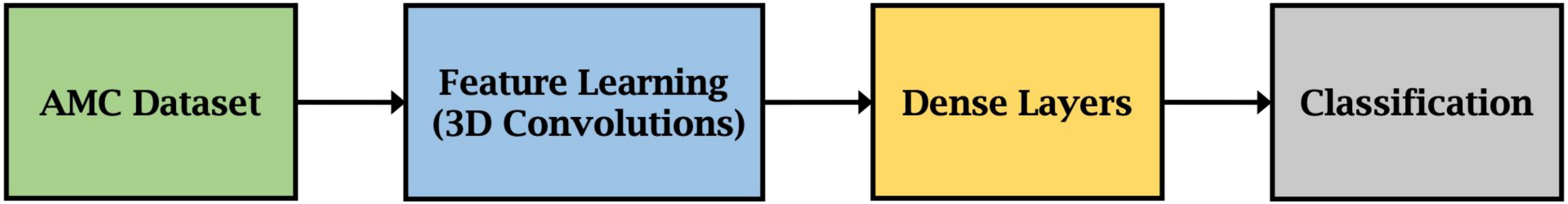
  → **Selected models already contain holes or are too complex**
- **Artificial defined defects**, i.e. too far away from the real given problem, strong abstraction
- If models and the defects gets more complex, this approach **quickly reaches its limitations**



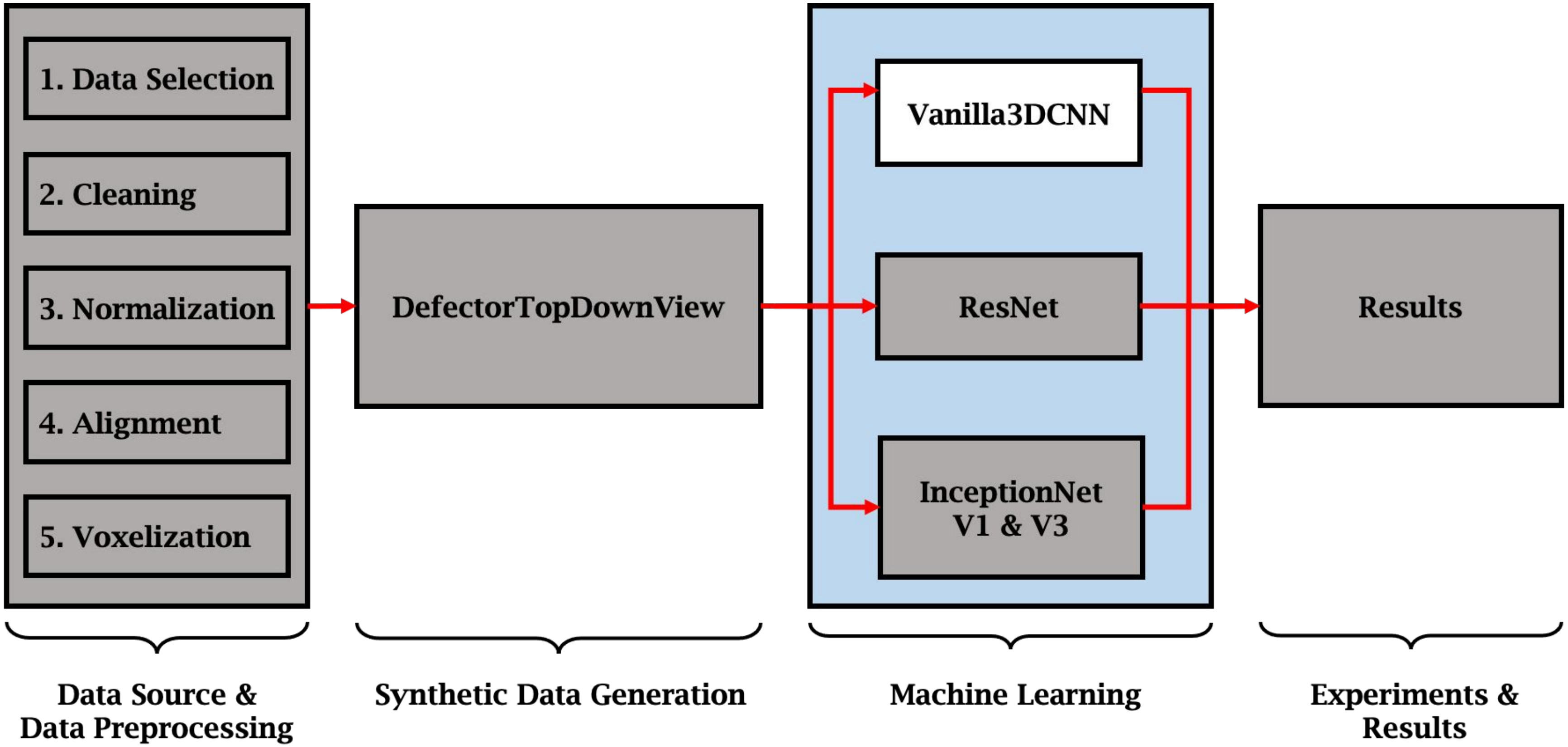



1. Data Selection
2. Cleaning
3. Normalization
4. Alignment
5. Voxelization
DefectorTopDownView
Vanilla3DCNN
ResNet
InceptionNet
V1 & V3
Results
Data Source &
Data Preprocessing
Synthetic Data Generation
Machine Learning
Experiments &
Results

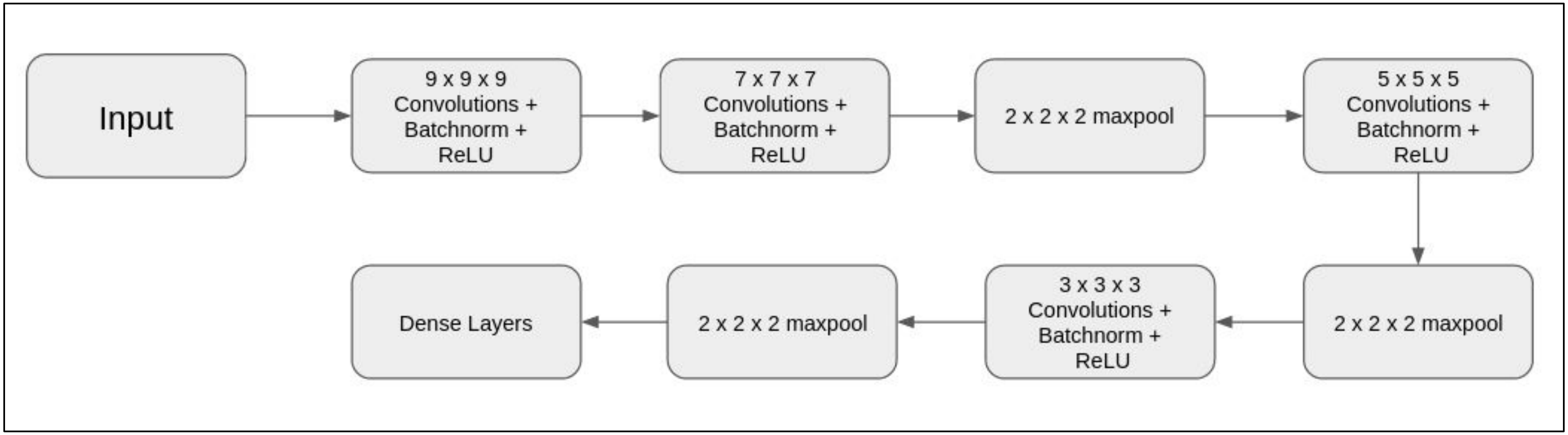
Deep Learning Workflow



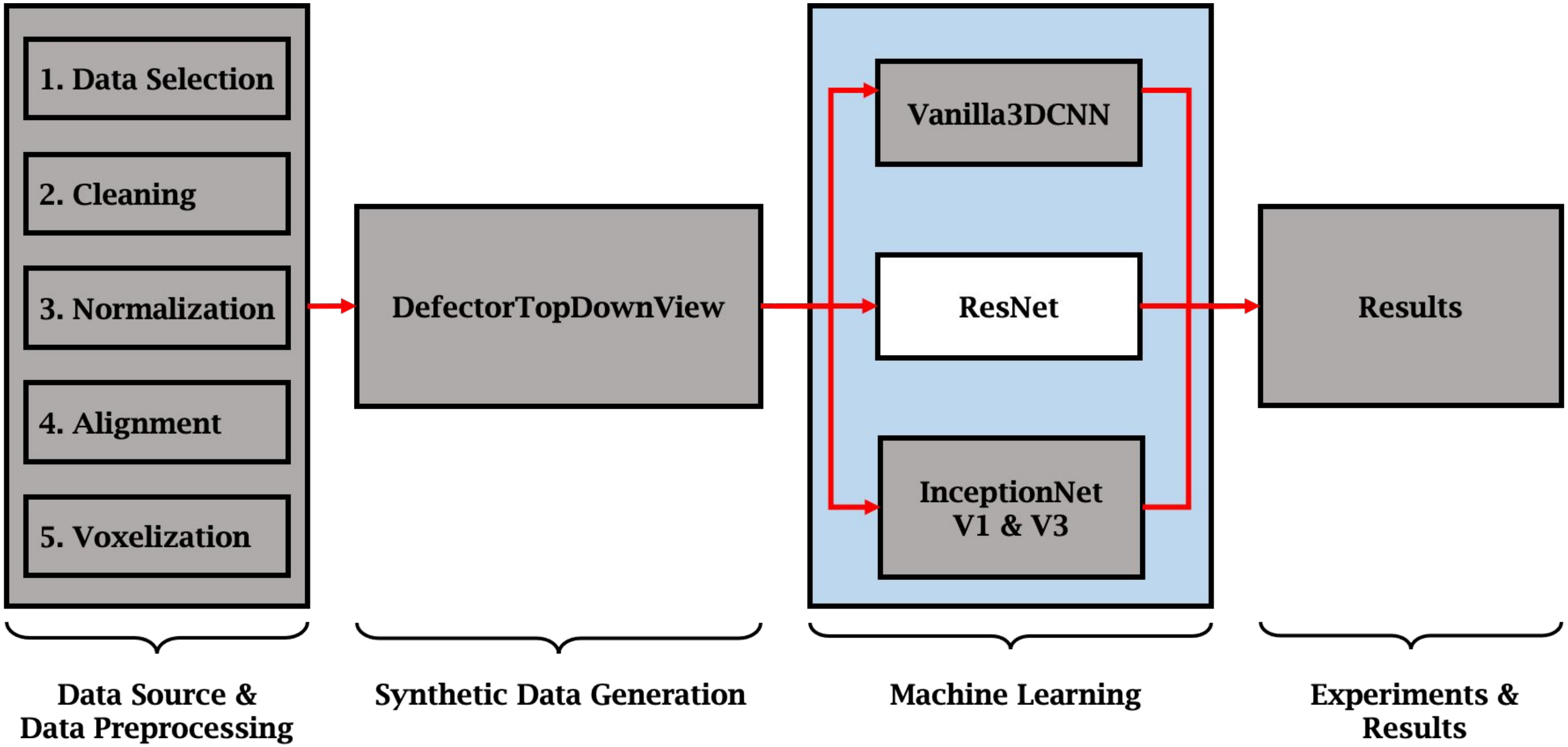
- Type of problem to be solved: Binary Classification
- Architectures examined for feature extraction:
  - Vanilla3DCNN
  - ResNet
  - InceptionNet V1
  - InceptionNet V3


1. Data Selection
2. Cleaning
3. Normalization
4. Alignment
5. Voxelization
DefectorTopDownView
Vanilla3DCNN
ResNet
InceptionNet V1 & V3
Results
Data Source & Data Preprocessing
Synthetic Data Generation
Machine Learning
Experiments & Results

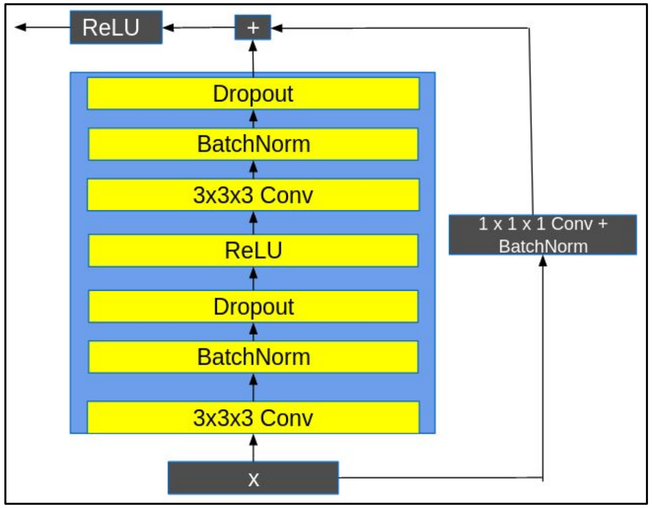
Vanilla3DCNN Architecture



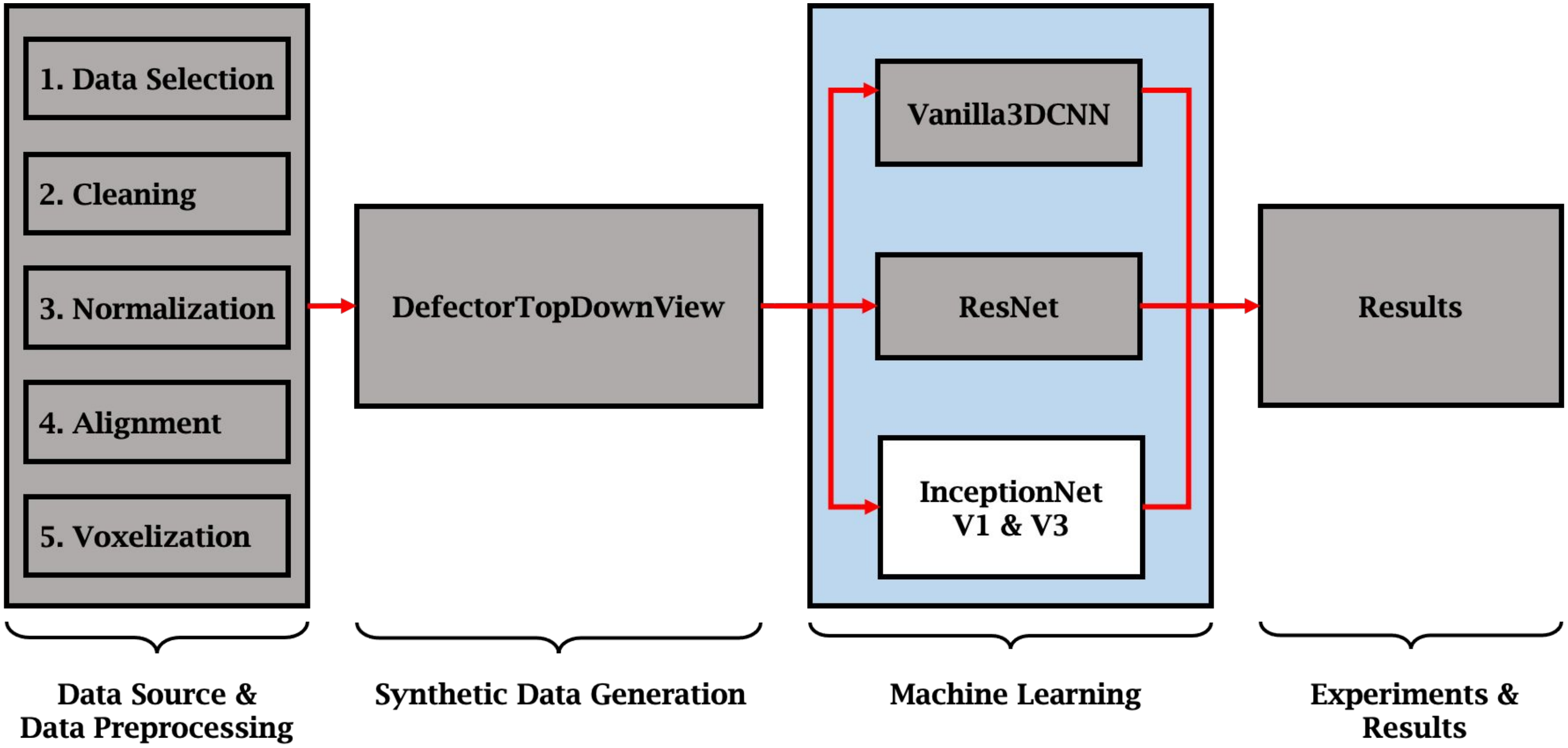
- Number of trainable parameters: **1.8M**
- 4 convolution layers with kernel sizes 9, 7, 5 and 3 were used.
- Maxpooling layer to **reduce spatial size.**
- Batch normalization layers to **stabilize the training process**.


1. Data Selection
2. Cleaning
3. Normalization
4. Alignment
5. Voxelization
DefectorTopDownView
Vanilla3DCNN
ResNet
InceptionNet V1 & V3
Results
Data Source & Data Preprocessing
Synthetic Data Generation
Machine Learning
Experiments & Results

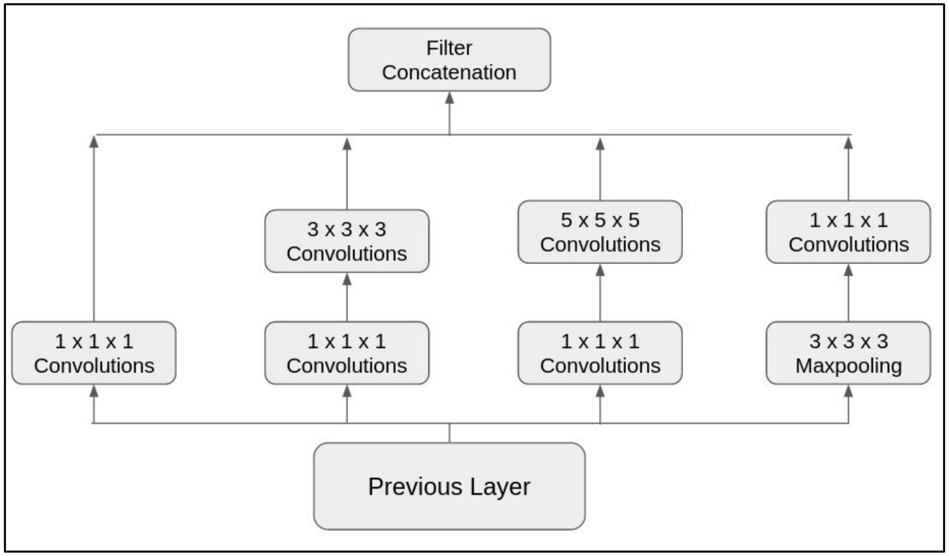
ResNet Architecture



- Number of trainable parameters: **9.1M**
- Basic residual block contains two 3 x 3 x 3 layers with batch norm layers and dropout layers.
- Basic residual block repeated twice.
- To **avoid sudden reduction of input spatial dimensions**, two pooling layers used at the end.

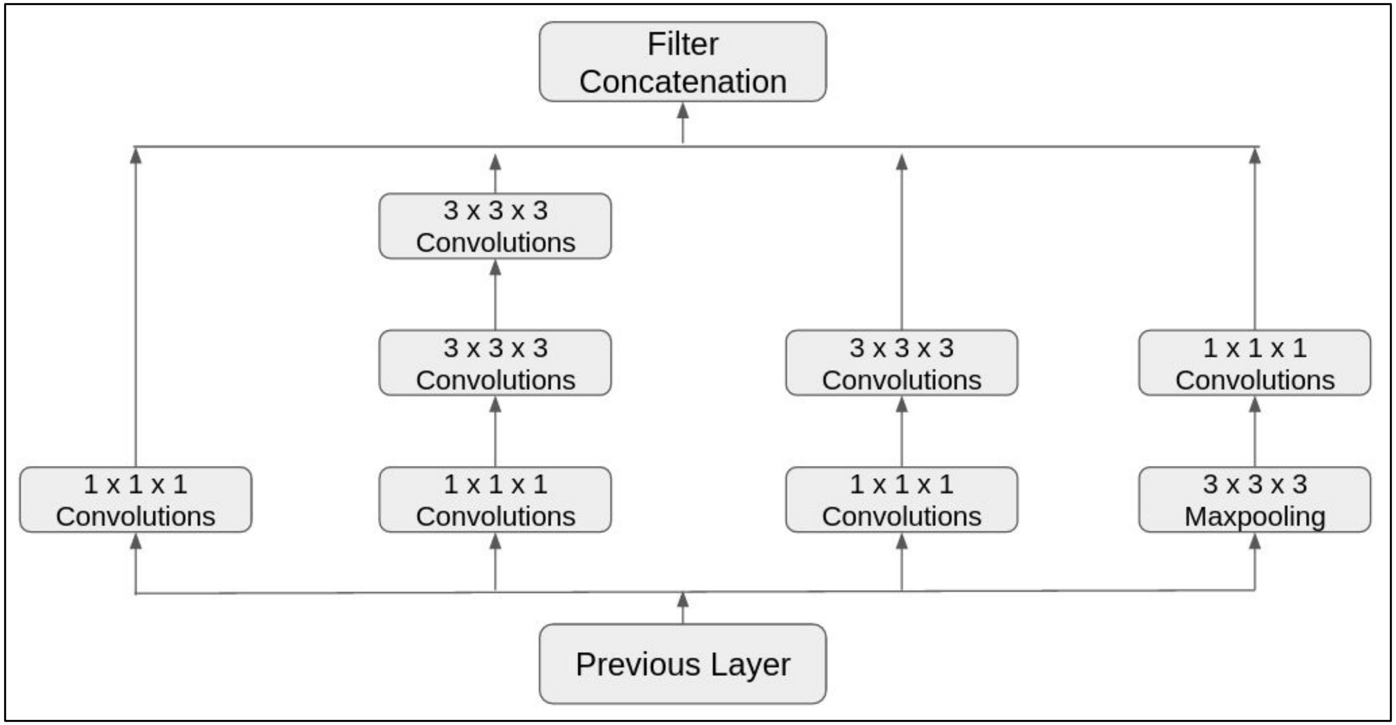

1. Data Selection
2. Cleaning
3. Normalization
4. Alignment
5. Voxelization
DefectorTopDownView
Vanilla3DCNN
ResNet
InceptionNet
V1 & V3
Results
Data Source &
Data Preprocessing
Synthetic Data Generation
Machine Learning
Experiments &
Results

InceptionNet V1 Architecture



- Basic intuition: **Let's go wider.**
- Number of trainable parameters: **9.6M**
- To capture the salient features having varying dimensions, choosing right kernel size is difficult.
- Each InceptionNet V1 block has kernel sizes 1,3 and 5 operating at the same level.
- InceptionNet V1 block repeated 4 times.

InceptionNet V3 Architecture



- Number of trainable parameters: **17.9M**
- Improvement over InceptionNet V1
- Disadvantage of InceptionNet V1: **Large reduction** in spatial dimension due to kernel size 5 x 5 x 5.

Implementation Details

1. **Activation Function:**
   - ReLU[1] activation for all layers except for the final layer.
   - Sigmoid[1] activation for the final layer.

2. **Loss Function:**
   - Binary cross entropy loss[2] used.

$$\boxed{BCE = -\sum_{i=1}^{C=2} t_i log(f(s_i)) = -t_1 log(f(s_1)) - (1 - t_1) log(1 - f(s_1))}$$

[1] Activation Functions: Comparison of Trends in Practice and Research for Deep Learning

[2] Generalized Cross Entropy Loss for Training Deep Neural Networks with Noisy Labels

VOLKSWAGEN DATA:LAB

Implementation Details

3. **Weight Initialization:**
    - Kaiming normal[3] weight initialization with **Fan-out mode** was used.
    - Weights follow normal distribution as shown below:

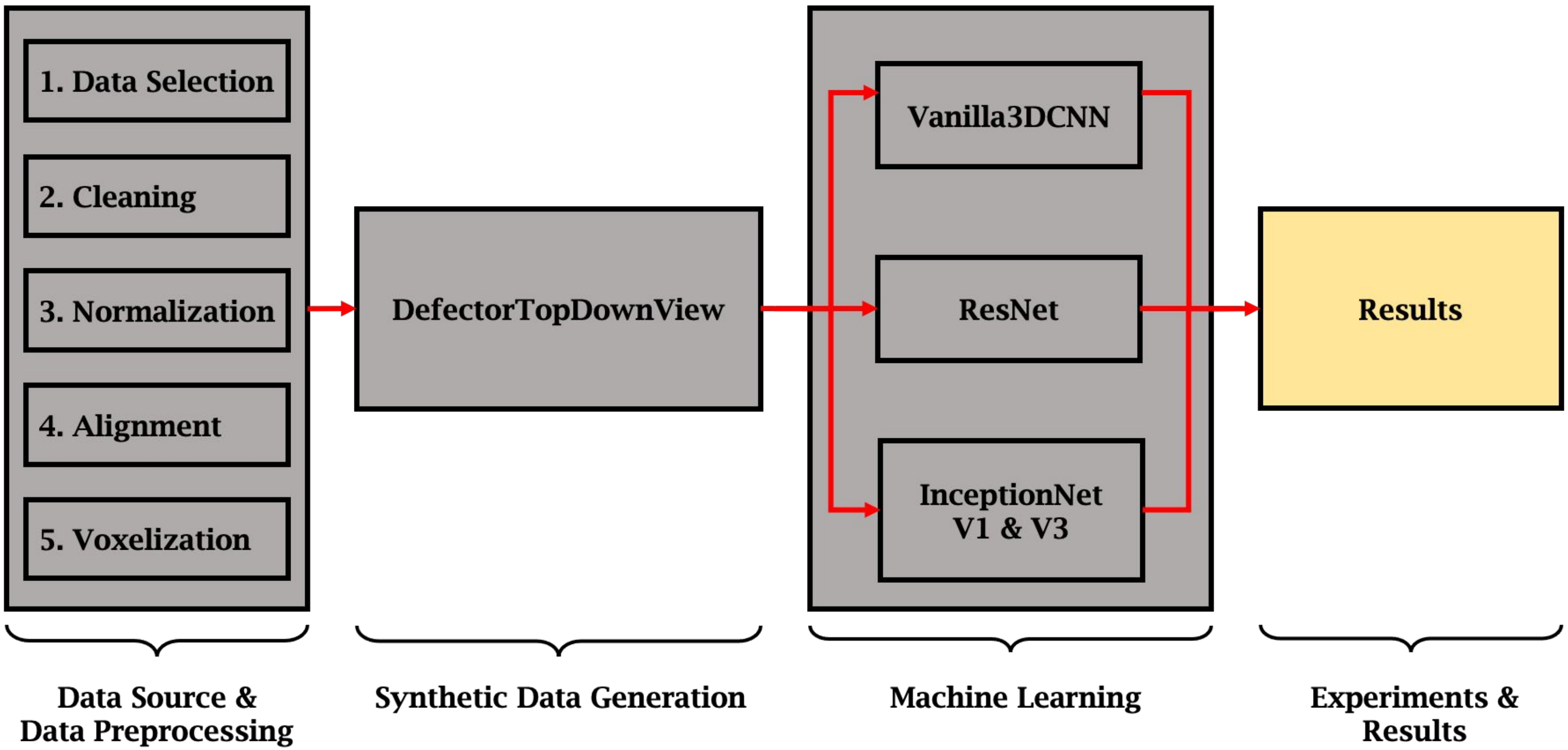
$$W \sim N\left(0, \sqrt{\frac{2}{n}}\right)$$

4. **Optimizer:**
    - Momentum based optimization algorithm is the default algorithm for all architectures.
    - Adam optimizer[4] used: Computes **adaptive learning rates** for each parameter.

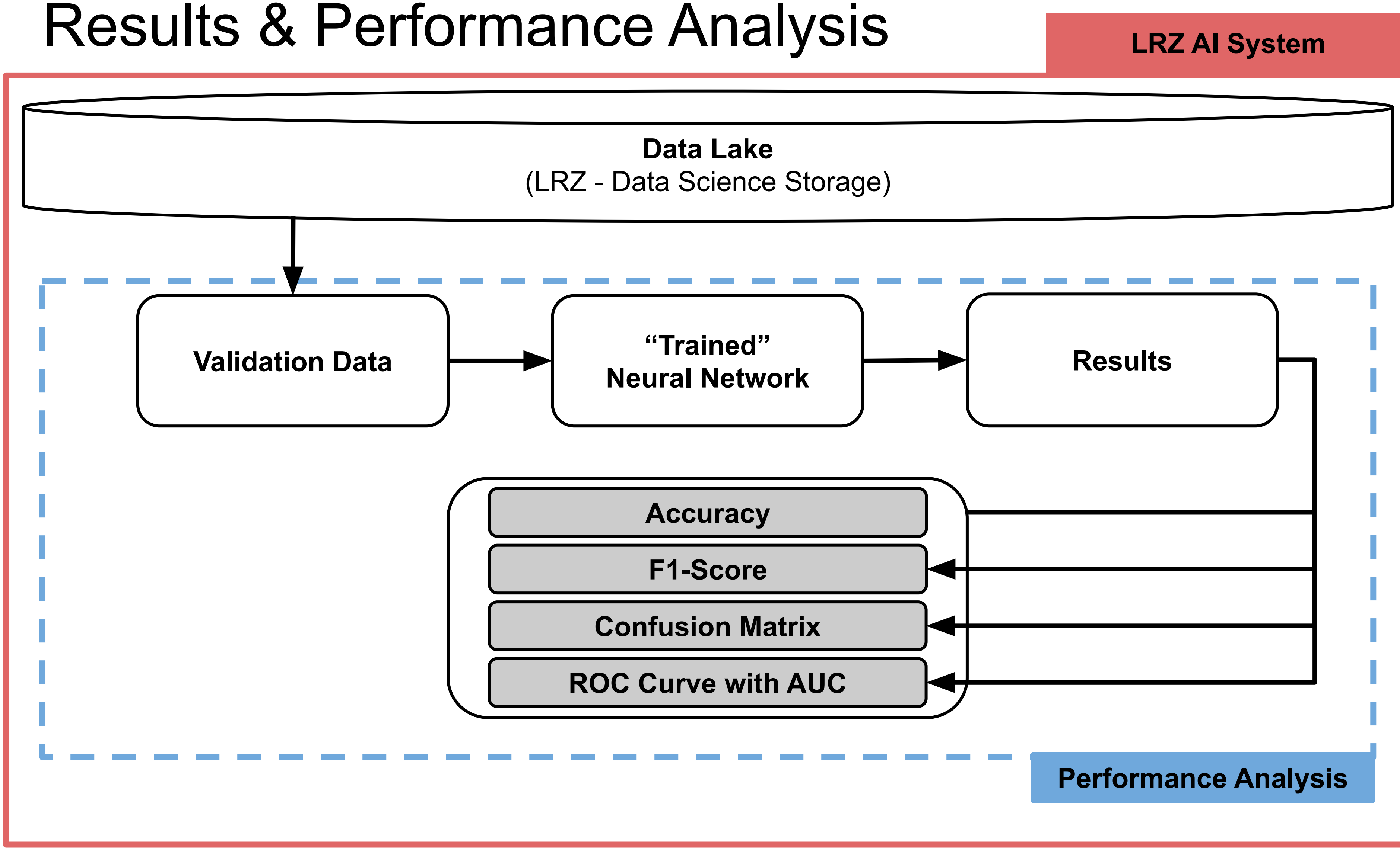
4. **Dataset:**
    - Total data samples: 7430
    - Train/Validation split (random): 80/20, 5944 (train) and 1486 (validation)
    - Law of large numbers: train and validation set are **balanced**

[3] Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification.

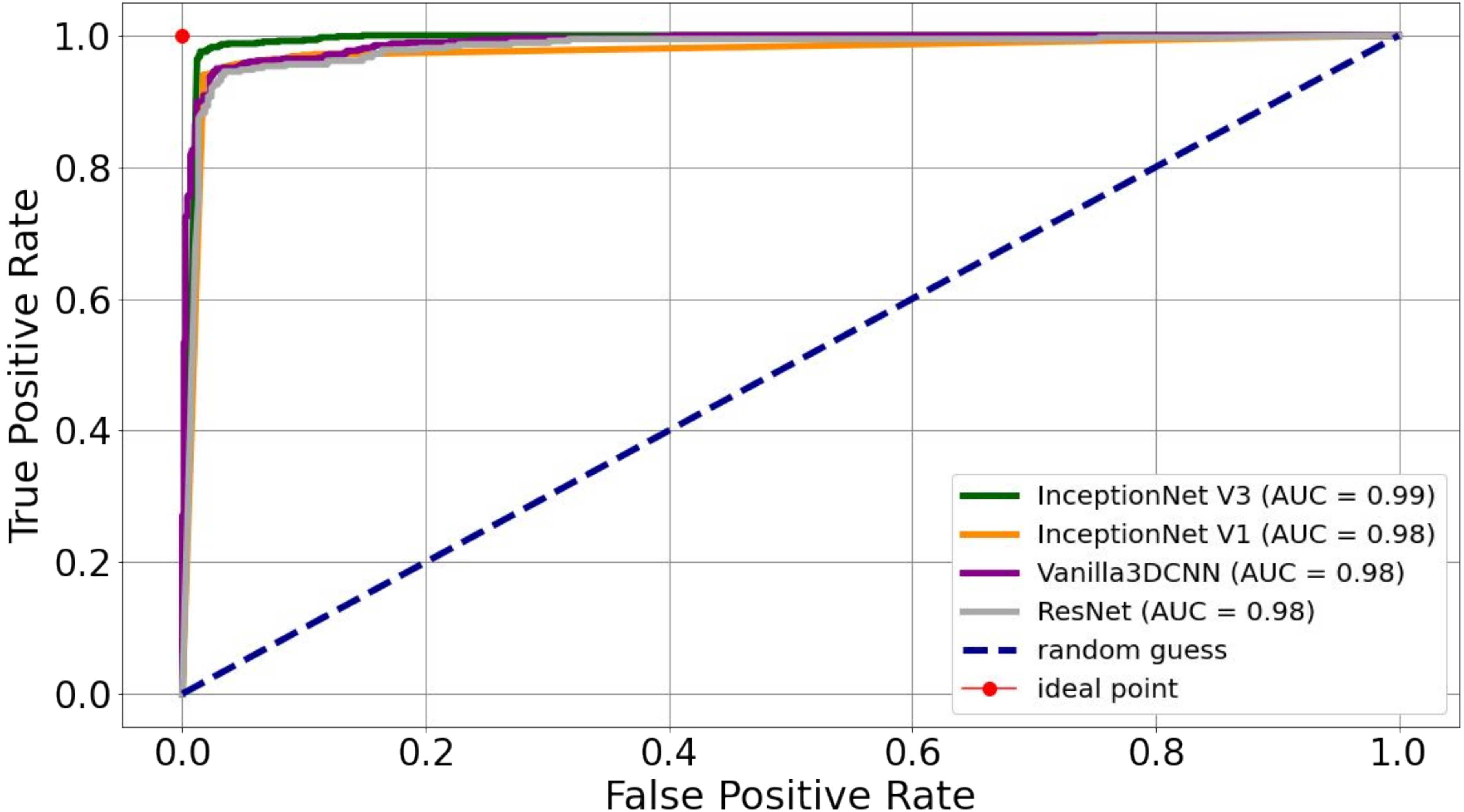
[4] Adam : A Method for Stochastic Optimization


1. Data Selection
2. Cleaning
3. Normalization
4. Alignment
5. Voxelization
DefectorTopDownView
Vanilla3DCNN
ResNet
InceptionNet V1 & V3
Results
Data Source & Data Preprocessing
Synthetic Data Generation
Machine Learning
Experiments & Results

Results & Performance Analysis

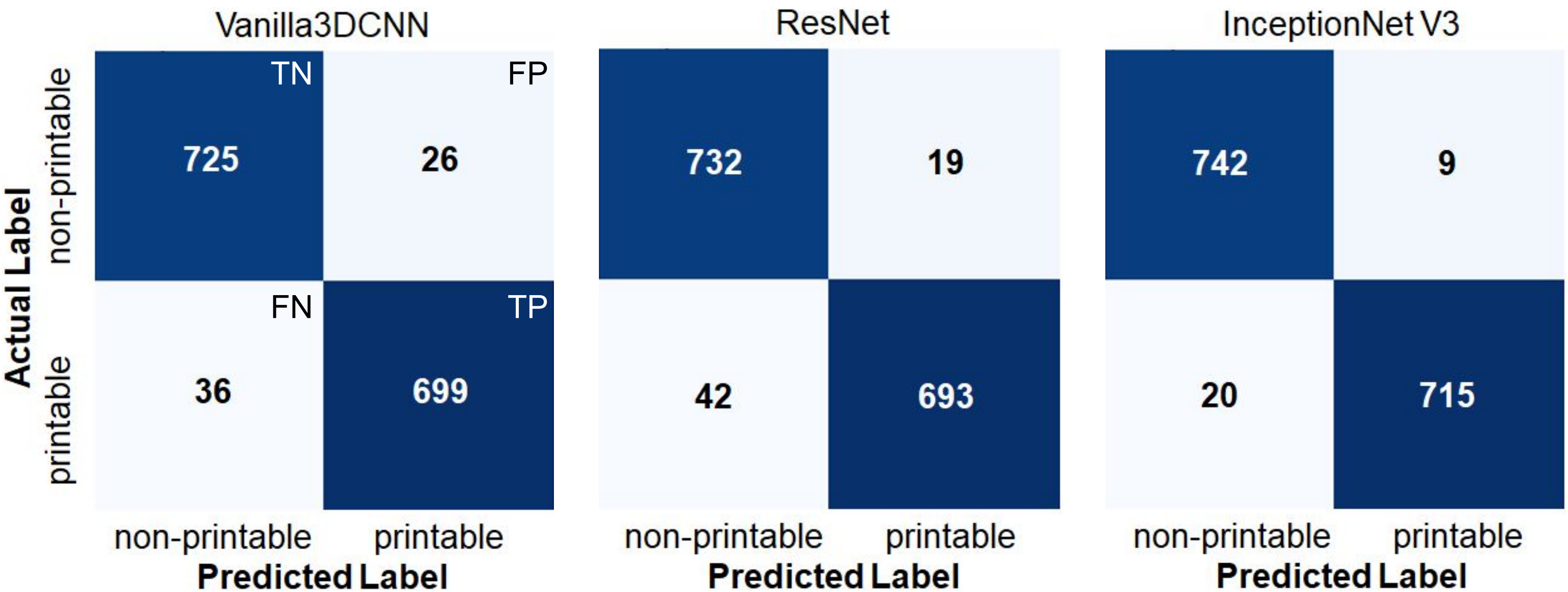


ROC$^{(1)}$ Curve with AUC$^{(2)}$



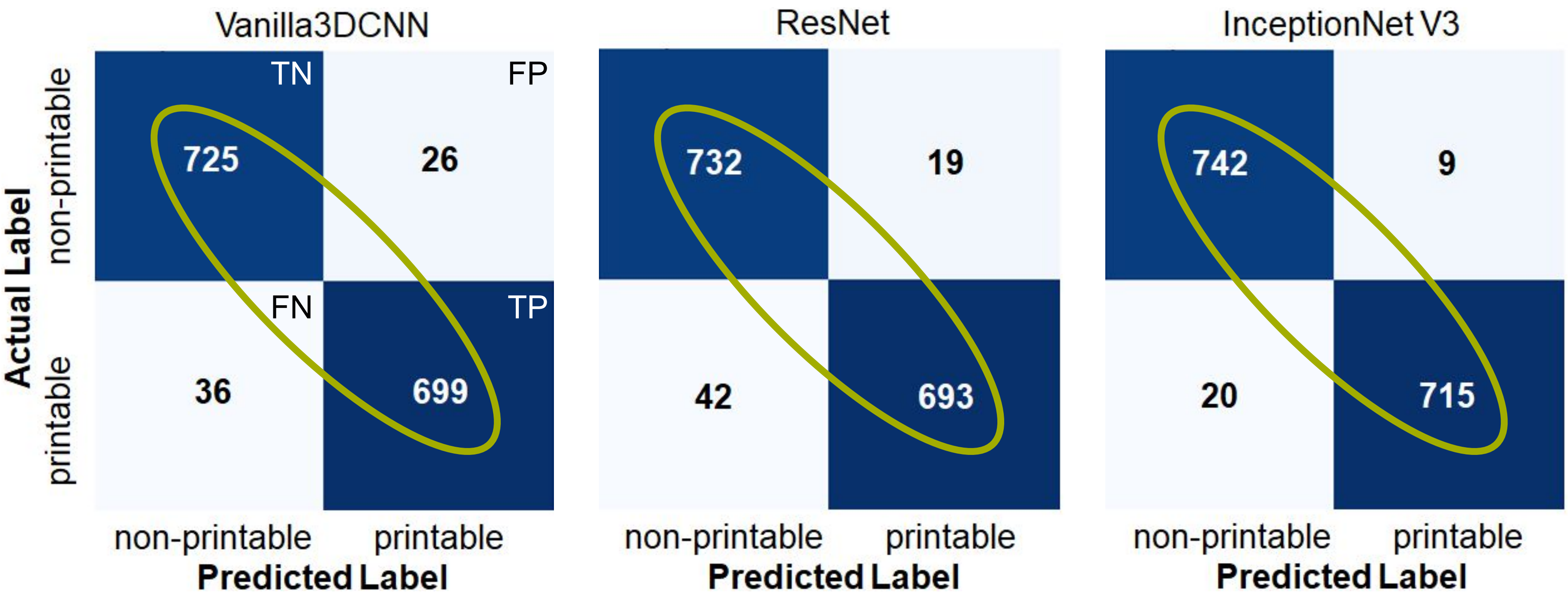
(1) ROC: Receiver Operating Characteristic
(2) AUC: Area under the ROC Curve

Confusion Matrix



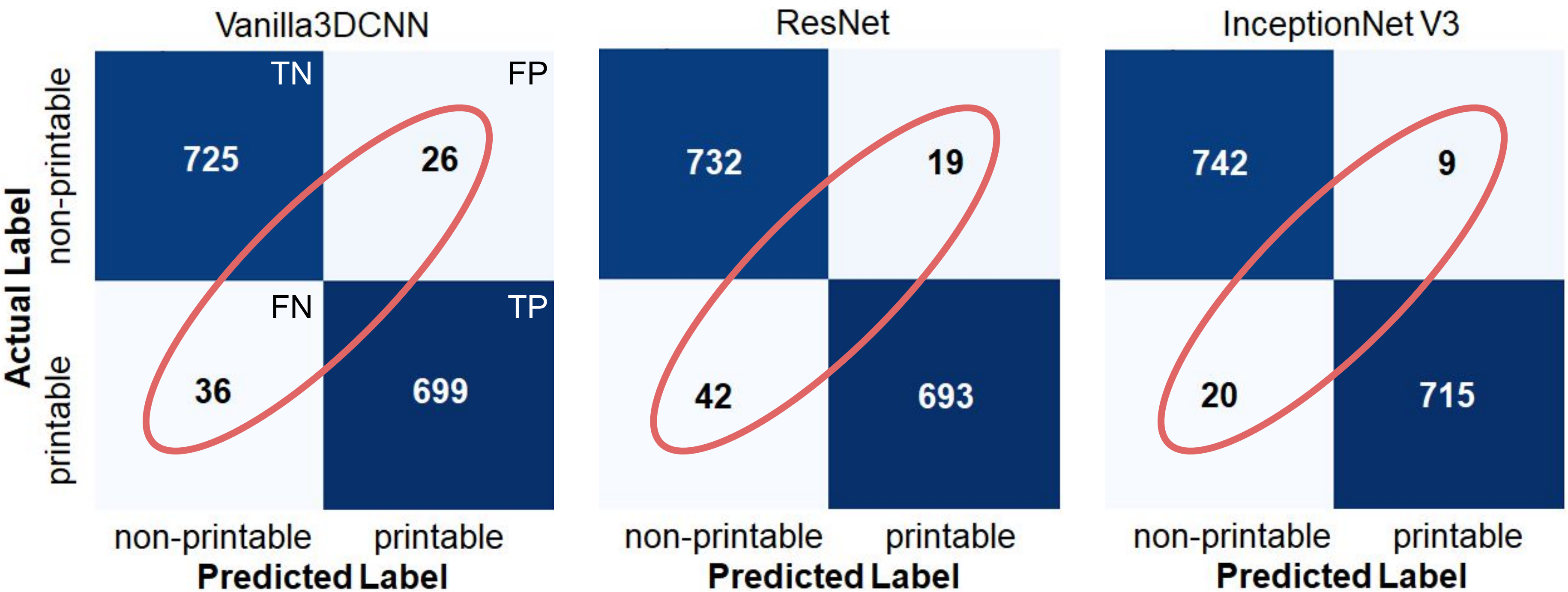
- **Diagonal elements:** correct prediction (TN & TP)
- **Off-diagonal elements:** wrong predictions (FN & FP)

Confusion Matrix



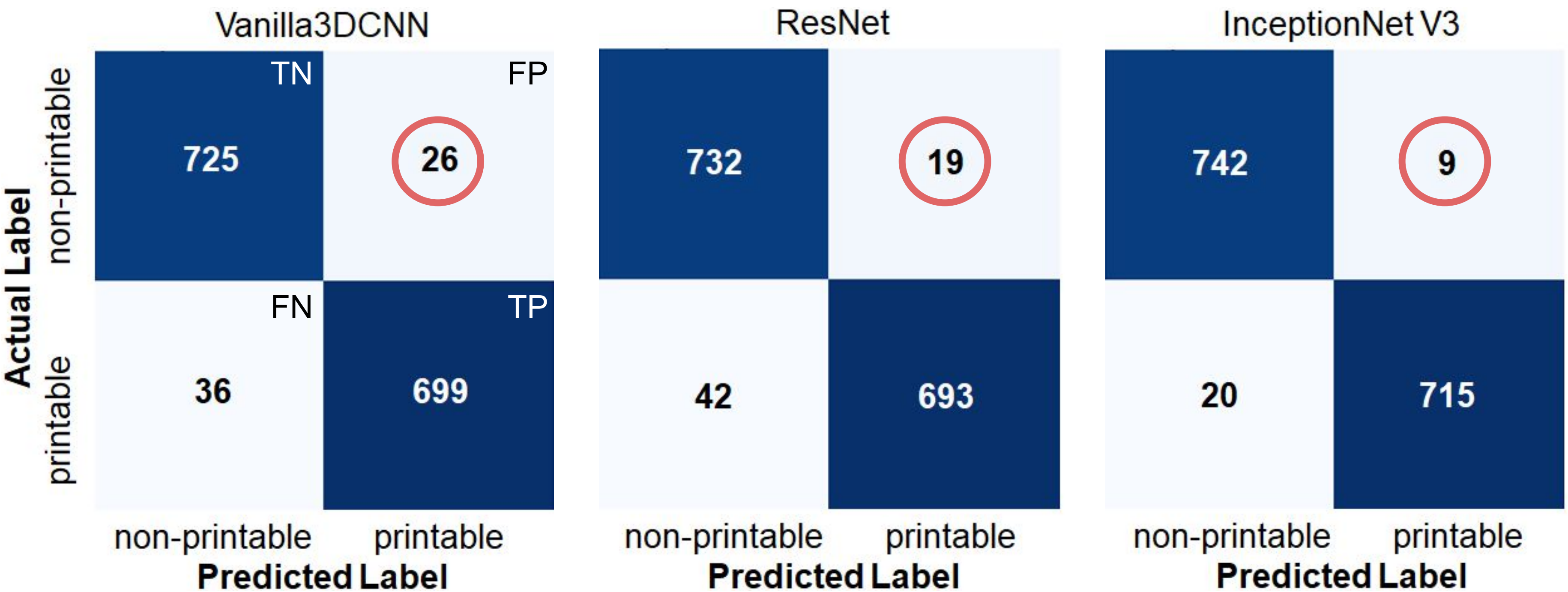
- **True Negatives > True Positives** (for all classifiers)
- **True Negatives:** Method of adding defects works pretty well
- **True Positives:** Assumption of all initial models being printable does not hold

Confusion Matrix



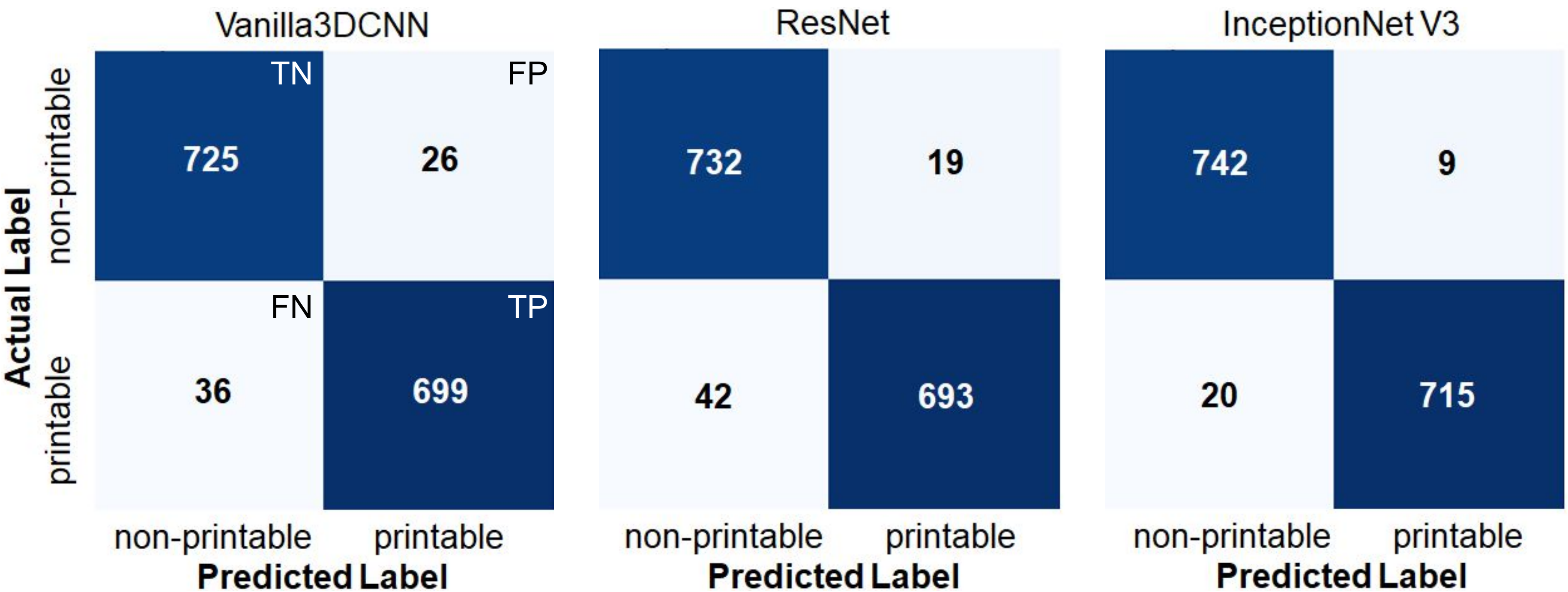
- **Very few off-diagonal elements:** Good overall performance
- **False Negatives > False Positives** (for all classifiers)

Confusion Matrix



- **Avoid:** False Positives (FP) over False Negatives (FP)
- **Additive Manufacturing:** False Positives waste time & material
  False Negatives require manual check by engineer

Confirm Results: Accuracy(1) & F1-Score(2)

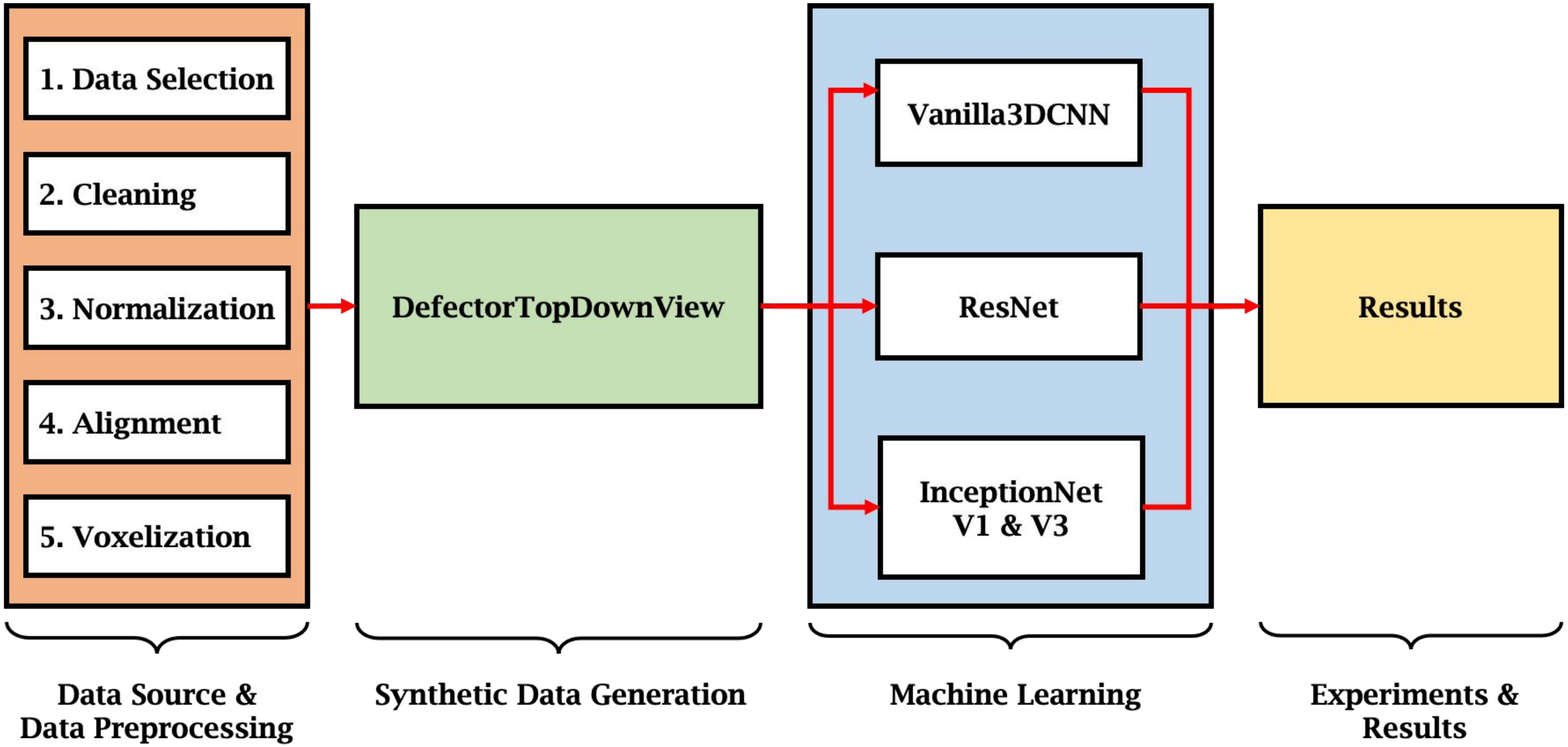


Accuracy	F1-Score
95.8 %	0.957

Accuracy	F1-Score
95.9 %	0.959

Accuracy	F1-Score
98.1 %	0.980

(1) In this case: balanced accuracy
(2) F1-Score ∈ [0, 1]; harmonic mean of precision(3) and recall(4)
(3) Precision: *How often is it correct when a positive is predicted:* TP/(FP+TP)
(4) Recall: *How often is a positive predicted when it actually is positive:* TP/(FN+TP)


1. Data Selection
2. Cleaning
3. Normalization
4. Alignment
5. Voxelization
DefectorTopDownView
Vanilla3DCNN
ResNet
InceptionNet V1 & V3
Results
Data Source & Data Preprocessing
Synthetic Data Generation
Machine Learning
Experiments & Results

Wrap up

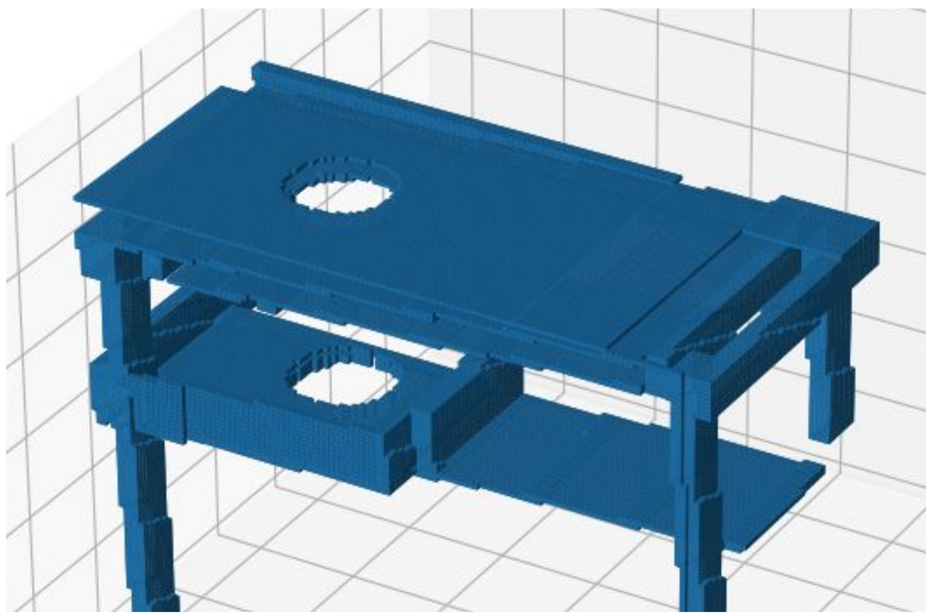
- Good performance of the model using the AMC dataset.
- **Limitations:**
  - Assumption of printability of the models selected
  - Restricted model selection
  - Failure of the defectors in some specific cases

Improvements: Evaluation methods

- Generate an additional test set from another chunk of the **ABC dataset** or from the **Thingi10K dataset**
- **Explainable AI techniques** for better understanding of the performances of the deep learning models.

Improvements: DefectorTopDownView Similarity Check Add-on

- The information given for the tdv could be misleading.
- **Global** and a **local** uniformity **check** for the area that will be removed by the defector.

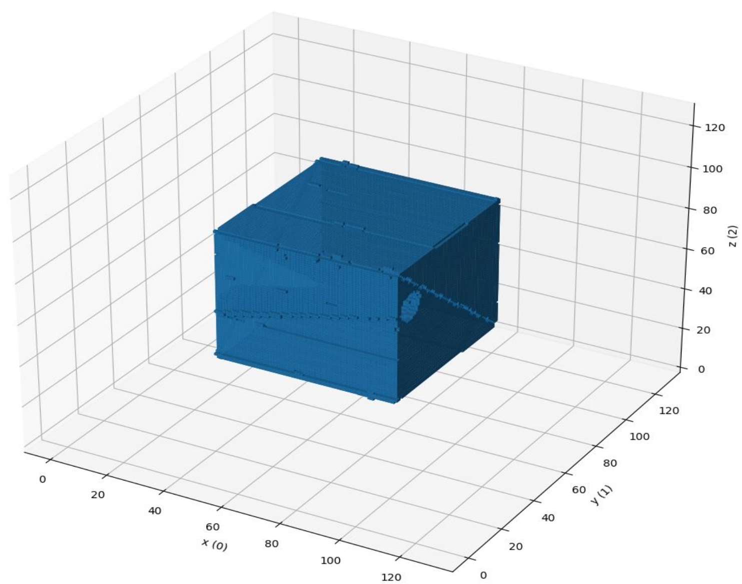


3D model labeled by the defector as manufacturable

Improvements: DefectorTopDownView Rotation Add-on

Insert rotated holes with random angles as a generalization of DefectorTopDownView

- Add a padding to the 3D models
- Rotate the models randomly throughout the x,y and z axes by the angles ϕ_x, ϕ_y, ϕ_z
- Insert the hole using DefectorTopDownView
- Rotate back the model



3D model augmented by a rotated hole

Potential Future Work

The defector is a deterministic algorithm with **limited capabilities**

- Use **self supervision:**
  - Train on original 3D models without artificial defect using an autoencoder
  - Use the encoder as feature extractor and stack final layer for binary classification
  - Train using expert labeled data
- **Few-shot** learning methods.
- Train **generative adversarial networks** (GANs) on a dataset of few complex models that are expert-labeled and generate new models with complex defects.

Thank you for your attention!

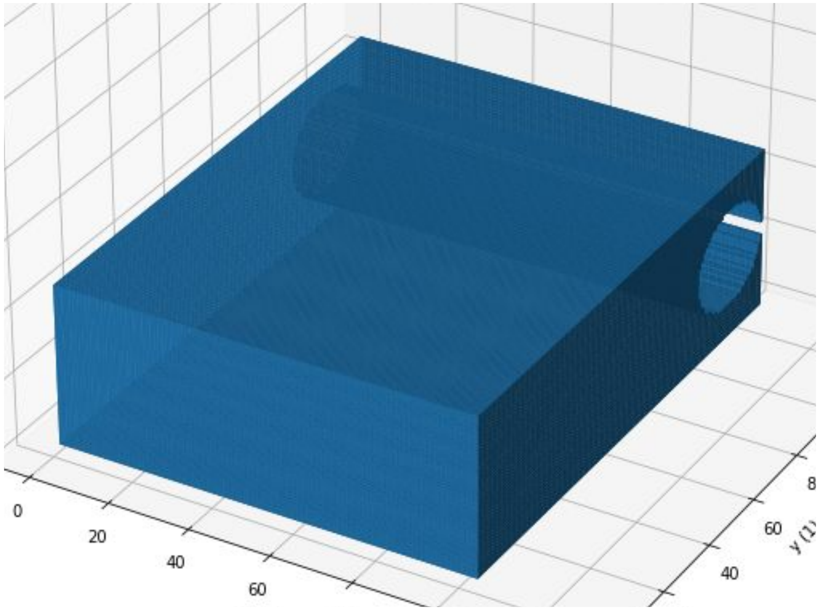
Additional slides

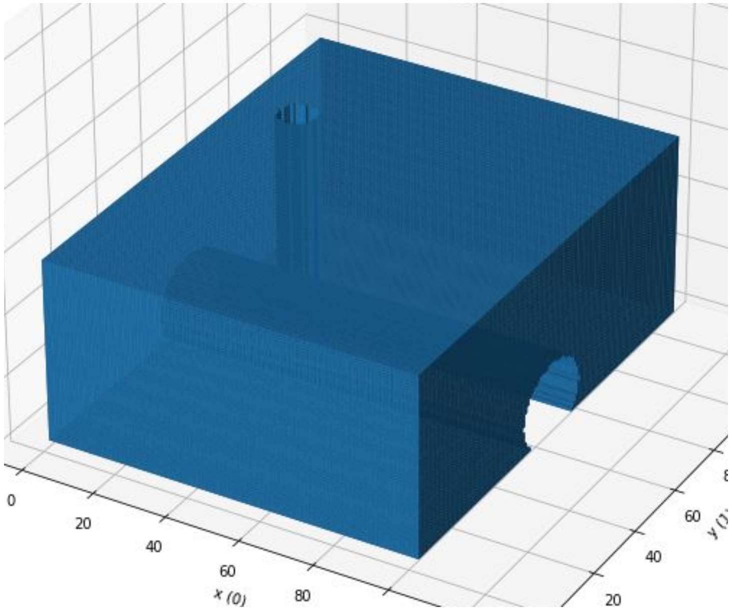
VOLKSWAGEN DATA:LAB

Additional slides
Synthetic Data Generation

Limitations examples 1/2

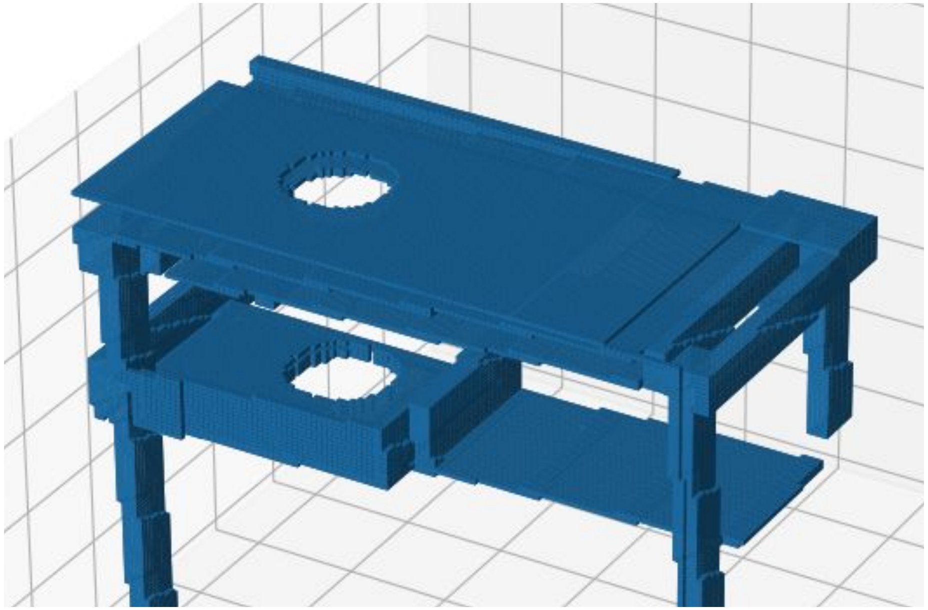
Selected 3D models **already contain holes**



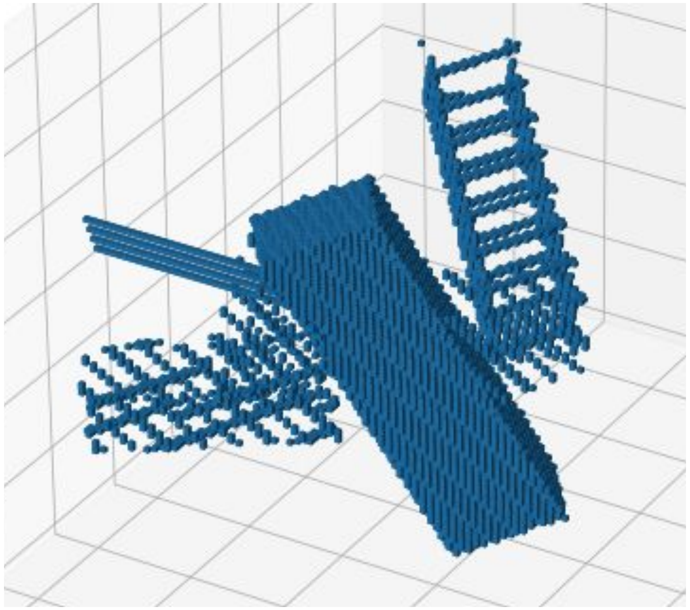


Limitations examples 2/2

Difficult and not detected 3D structure of the model



3D model **too complex**



Additional slides
Deep Learning

Detailed description of Vanilla3DCNN

Vanilla3DCNN Architecture		
Layer Name	**Output Size**	**CNN - Layers**
conv_1	120 x 120 x 120 x 32	9 x 9 x 9/1,32
conv_2	114 x 114 x 114 x 64	7 x 7 x 7/1, 64
Maxpool layer	57 x 57 x 57 x 64	2 x 2 x 2 maxpool,stride 2
conv_3	53 x 53 x 53 x 96	5 x 5 x 5/1, 96
Maxpool layer	26 x 26 x 26 x 96	2 x 2 x 2 maxpool,stride 2
conv_4	24 x 24 x 24 x 128	3 x 3 x 3/1, 128
Maxpool layer	12 x 12 x 12 x 128	2 x 2 x 2 maxpool,stride 2
Average pool layer	11 x 11 x 11 x 128	2 x 2 x 2 maxpool,stride 1
Maxpool layer	1 x 1 x 1 x 128	11 x 11 x 11 maxpool,stride 1
FC_1		32-d fc,ReLU
FC_2		Scalar value, Sigmoid

VOLKSWAGEN DATA:LAB

Detailed description of ResNet

ResNet Architecture		
Layer Name	**Output Size**	**CNN - Layers**
conv_1	64 x 64 x 64 x 64	5 x 5 x 5/2,64 (Same)
	32 x 32 x 32 x 64	3 x 3 x 3 maxpool,stride 2
conv_2	32 x 32 x 32 x 64	(3 x 3 x 3/1,64) x 2
		(3 x 3 x 3/1,64) x 2
conv_3	16 x 16 x 16 x 128	(3 x 3 x 3/2,128) x 2
		(3 x 3 x 3/2,128) x 2
conv_4	8 x 8 x 8 x 256	(3 x 3 x 3/2,256) x 2
		(3 x 3 x 3/2,256) x 2
Maxpool layer	4 x 4 x 4 x 256	5 x 5 x 5 maxpool,stride 1
Average pool layer	1 x 1 x 1 x 256	4 x 4 x 4 average pool,stride 1
conv_5	1 x 1 x 1 x 1024	(1 x 1 x 1,256)
FC_1		512-d fc,ReLU
FC_2		128-d fc,ReLU
FC_3		64-d fc,ReLU
FC_4		Scalar value,Sigmoid

Detailed description of InceptionNet V1

InceptionNet_v1 Architecture									
Type	**Patch_size/Stride**	**Output_size**	**Depth**	**#1 x 1 x 1**	**#3 x 3 x 3 reduce**	**#3 x 3 x 3**	**#5 x 5 x 5 reduce**	**#5 x 5 x 5**	**Pool Proj**
Convolution	5 x 5 x 5/1	64 x 64 x 64 x 64	1						
Max pool	2 x 2 x 2/2	32 x 32 x 32 x 64							
Convolution	3 x 3 x 3/1	32 x 32 x 32 x 192	1						
Max pool	2 x 2 x 2/2	16 x 16 x 16 x 192							
Inception(3a)		16 x 16 x 16 x 256	2	64	96	128	16	32	32
max pool	2 x 2 x 2/2	8 x 8 x 8 x 256	0						
Inception (4a)		8 x 8 x 8 x 512	2	192	96	208	16	48	64
Max pool	2 x 2 x 2/2	4 x 4 x 4 x 512	0						
Inception(5a)		4 x 4 x 4 x 1024	2	384	192	384	48	128	128
Maxpool	2 x 2 x 2/2	2 x 2 x 2 x 1024	0						
Inception(6a)		2 x 2 x 2 x 1024	2	384	192	384	48	128	128
Average pool	2 x 2 x 2/1	1 x 1 x 1 x 1024	0						
Dropout	p = 0.4								
FC1		512							
FC2		64							
FC3		1							

Detailed description of InceptionNet V3

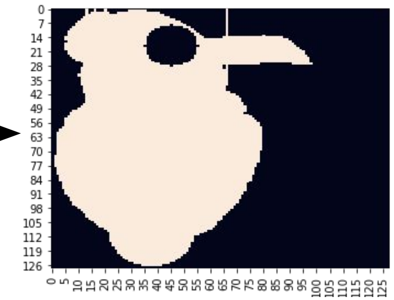
InceptionNet_v3 Architecture										
Type	**Patch_size/Stride**	**Output_size**	**Depth**	**#1 x 1 x 1**	**#3 x 3 x 3 reduce_1**	**#3 x 3 x 3 reduce_2**	**#3 x 3 x 3**	**#3 x 3 x 3 reduce**	**#3 x 3 x 3**	**Pool Proj**
Convolution	5 x 5 x 5/1	128 x 128 x 128 x 64	1							
Max pool	2 x 2 x 2/2	64 x 64 x 64 x 64								
Convolution	3 x 3 x 3/1	64 x 64 x 64 x 192	1							
Max pool	2 x 2 x 2/2	32 x 32 x 32 x 192								
Inception 3		32 x 32 x 32 x 256	3	32	64	64	128	64	64	32
max pool	2 x 2 x 2/2	16 x 16 x 16 x 256	0							
Inception 4		16 x 16 x 16 x 512	2	128	128	128	128	64	128	128
Max pool	2 x 2 x 2/2	8 x 8 x 8 x 512	0							
Inception 5		4 x 4 x 4 x 1024	2	128	256	256	384	256	384	128
Maxpool	2 x 2 x 2/2	2 x 2 x 2 x 1024	0							
Inception 6		2 x 2 x 2 x 1024	2	128	256	256	384	256	384	128
Average pool	2 x 2 x 2/1	1 x 1 x 1 x 1024	0							
Dropout	p = 0.4									
FC1		512								
FC2		64								
FC3		1								

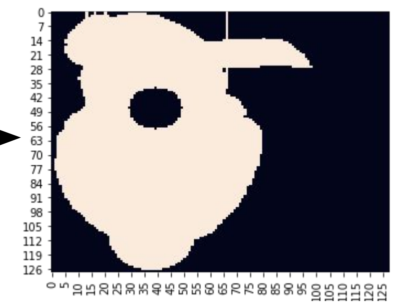
VOLKSWAGEN DATA:LAB

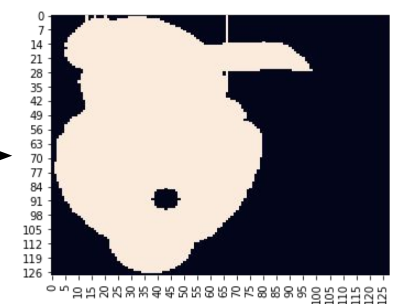
Additional slides
Results & Performance Analysis

False Prediction Analysis[1]

Label	Defect Location	Count
non-printable	border	4
printable	middle	6
non-printable	middle	9
printable	no-defect	10





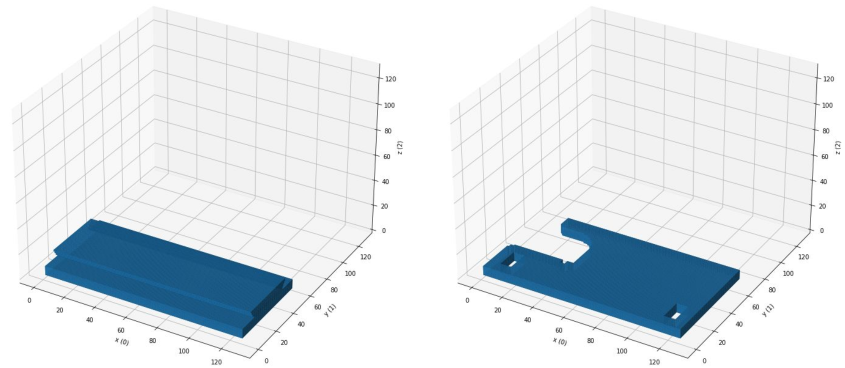


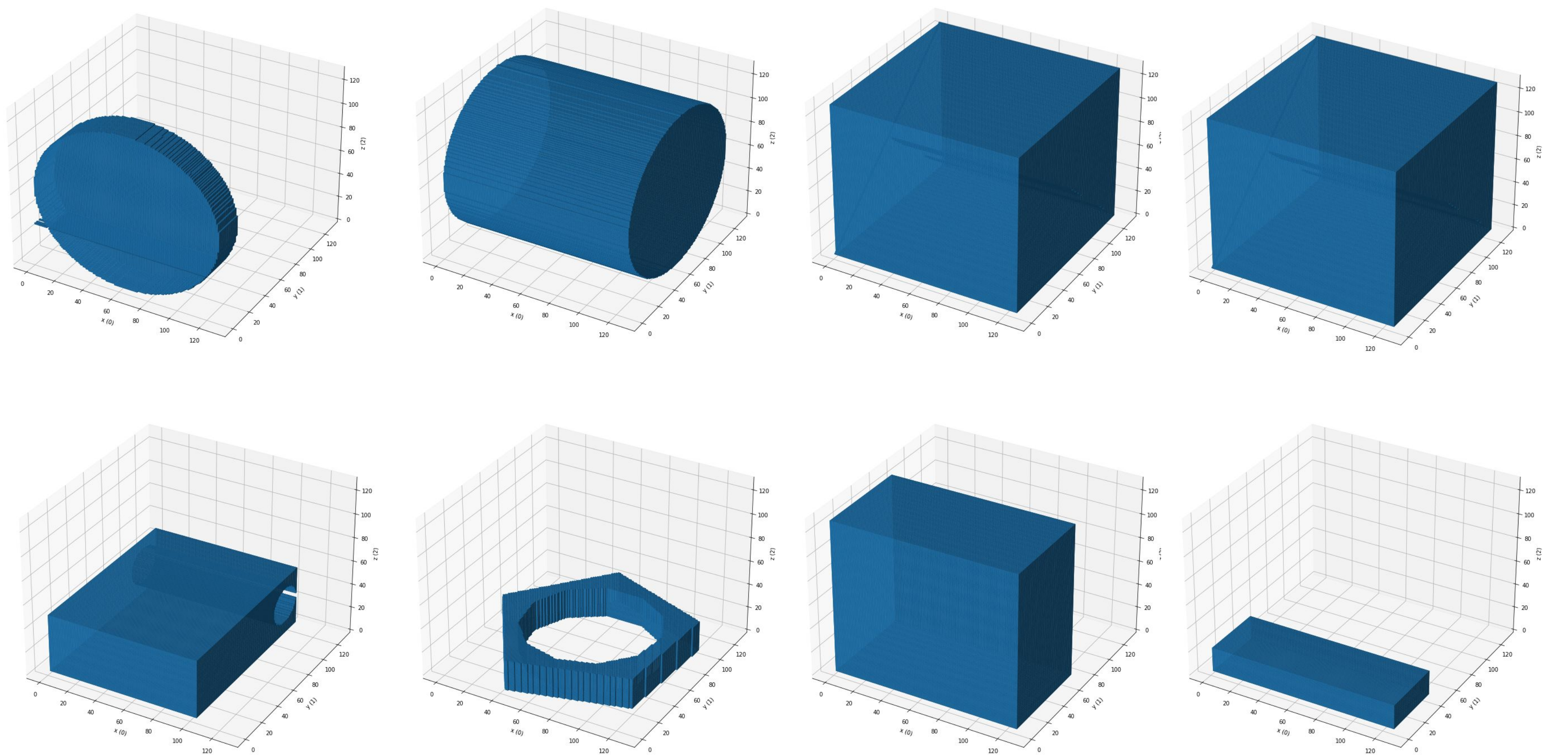
(1) Utilized neural network model: InceptionNet V3

VOLKSWAGEN DATA:LAB

False Prediction Analysis(1)

printable - no defect added

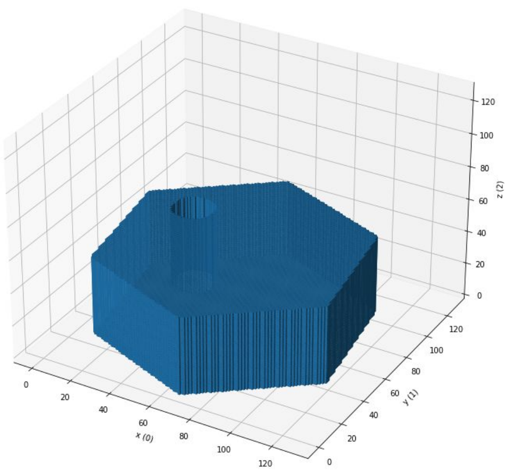


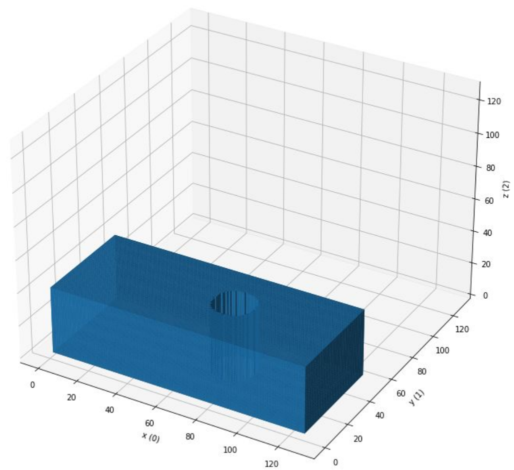


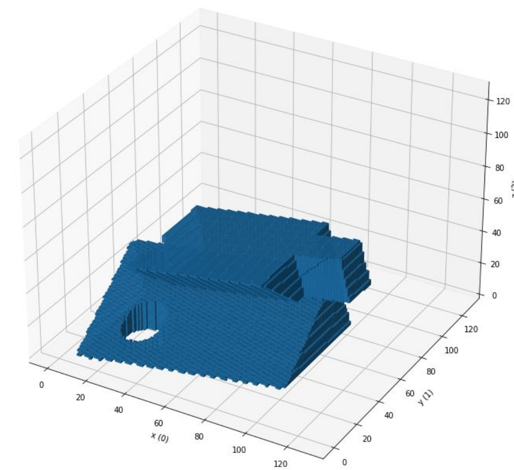
(1) Utilized neural network model: InceptionNet V3

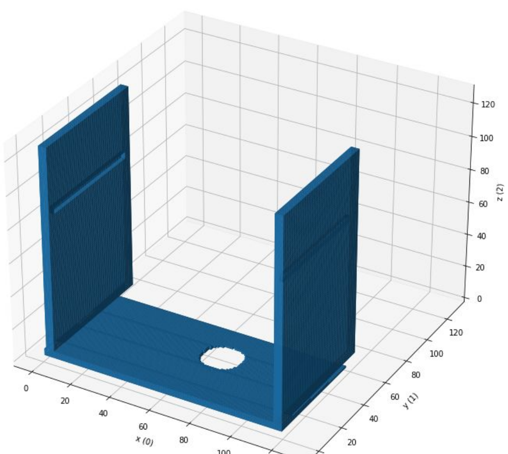
False Prediction Analysis(1)

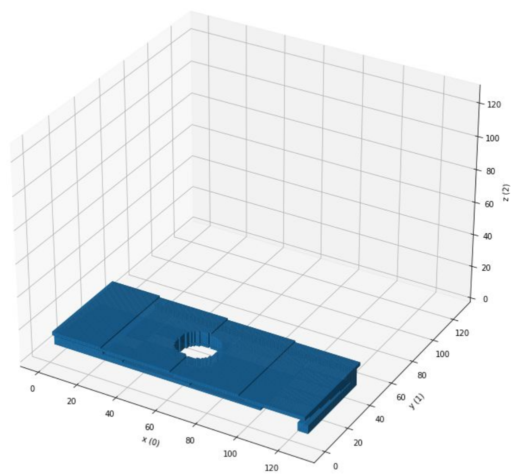
printable - middle

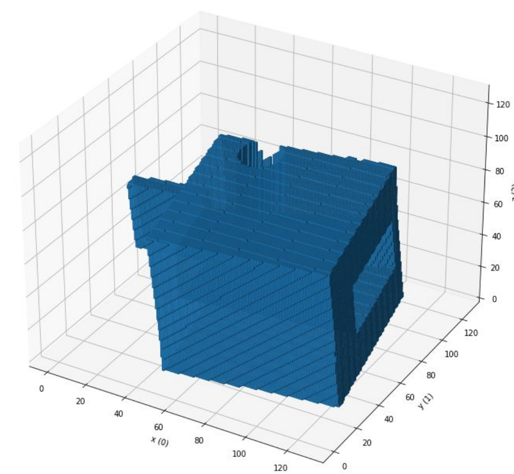








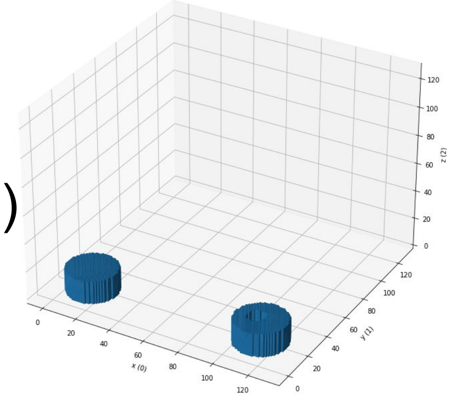
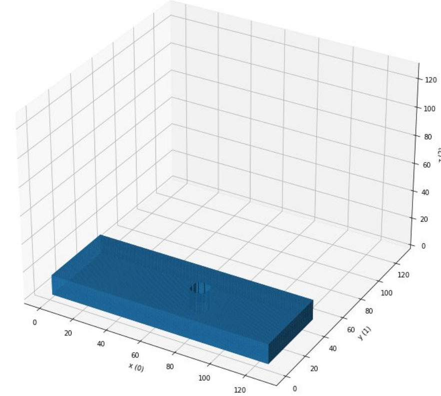
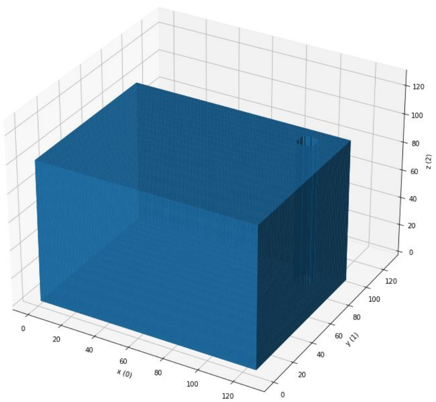
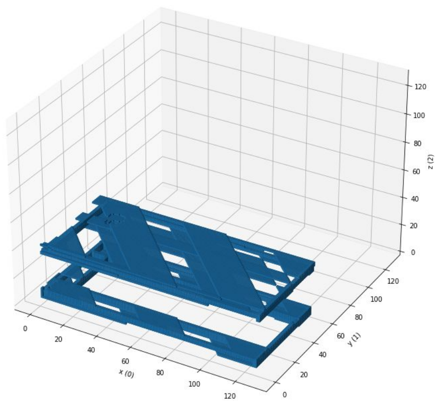
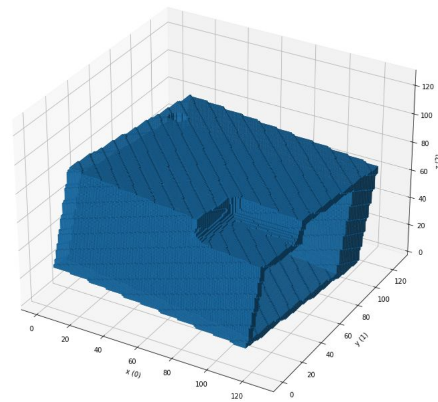
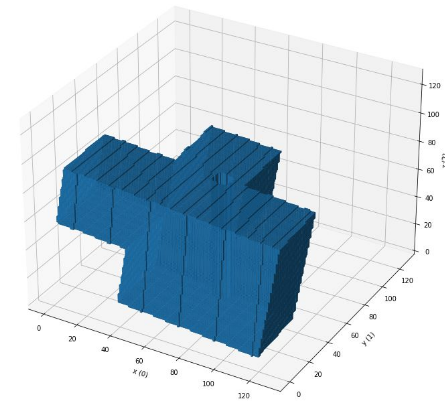
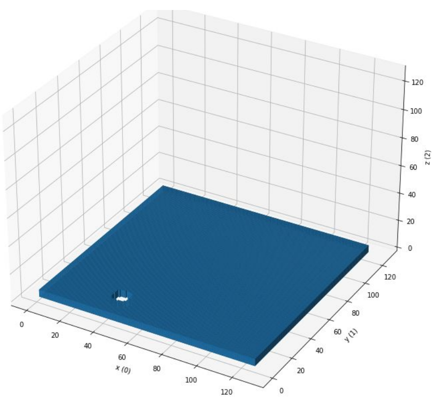
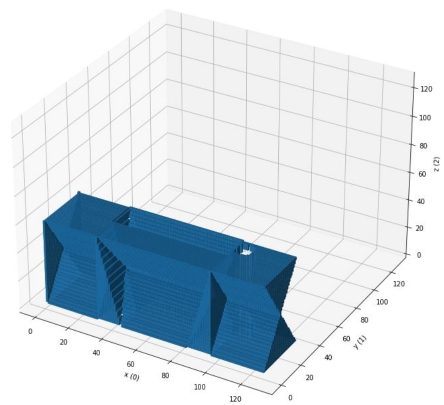
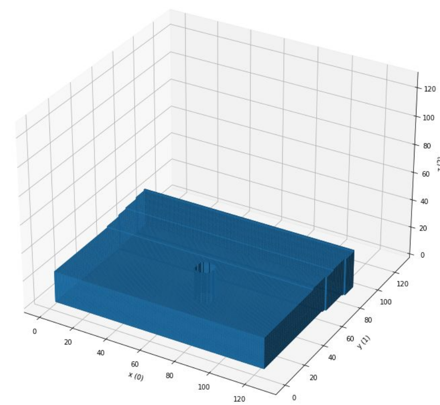




(1) Utilized neural network model: InceptionNet V3

False Prediction Analysis[1]

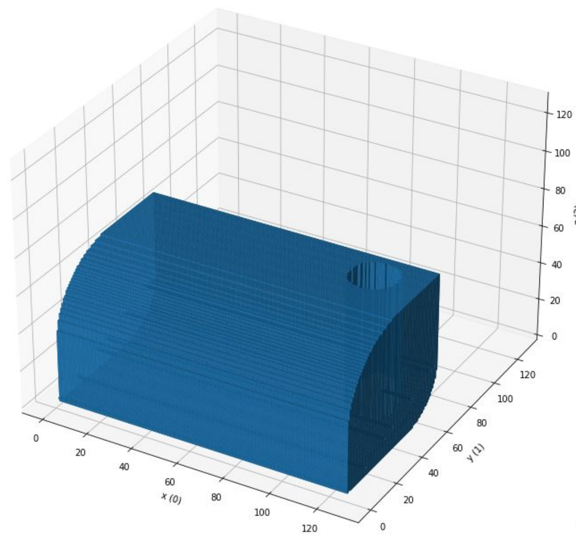
non-printable - middle

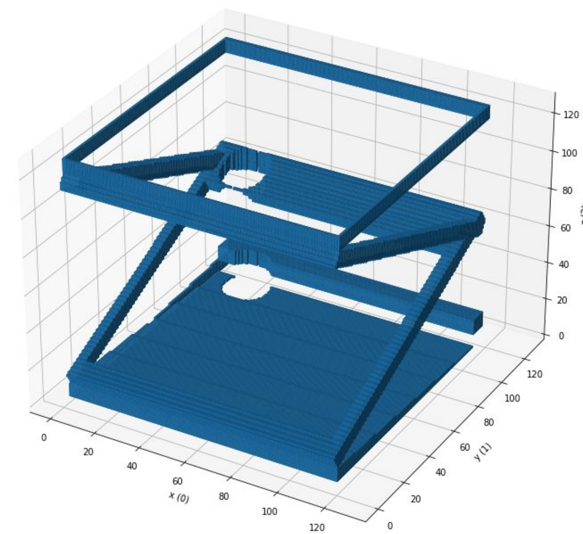










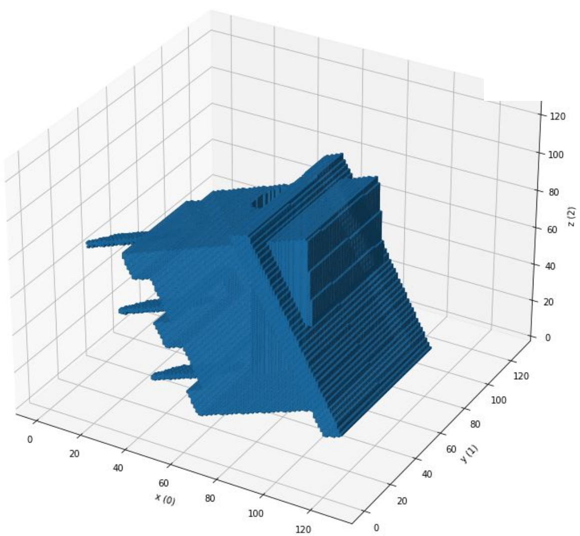
(1) Utilized neural network model: InceptionNet V3

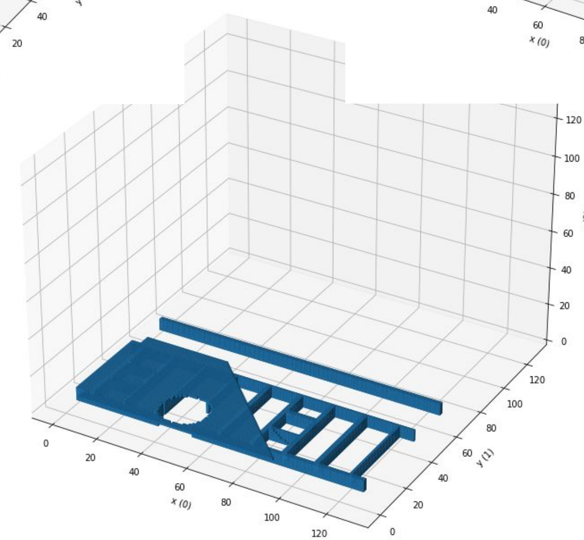
False Prediction Analysis(1)

non-printable - bord









(1) Utilized neural network model: InceptionNet V3