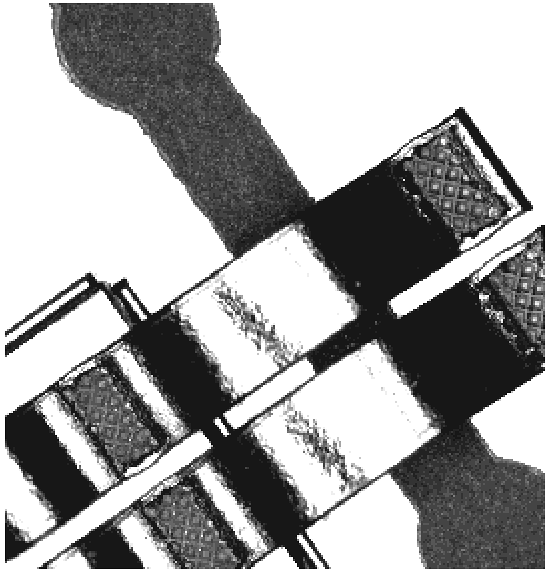
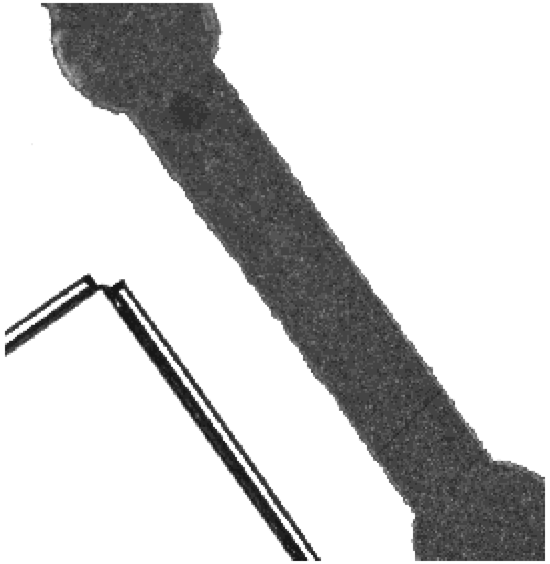
Image classification: failure mode detection





Niklas Kohlmann, Johannes Münderlein, Aadip Thapaliya

Course: Machine Vision with Tensorflow , 22.01.2026

Introduction

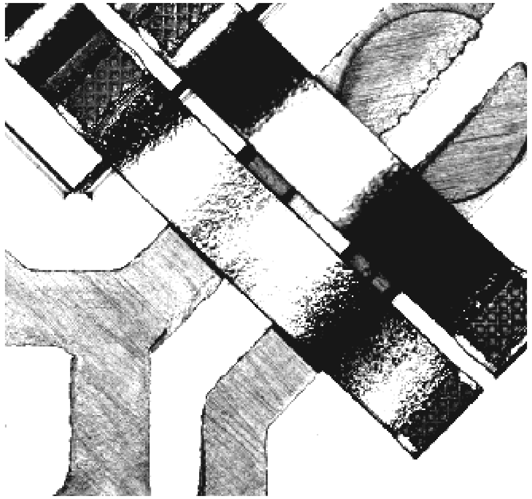
- Data obtained via:

https://challengedata.ens.fr/participants/challenges/157/

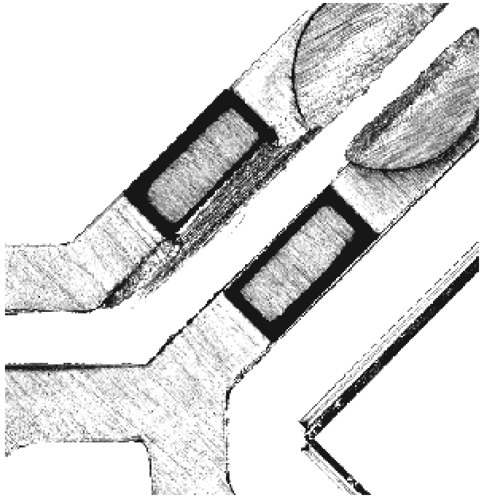
Classification of images of microelectronic components

- Data and challenge provided by:

(French-Chinese Electronics company)



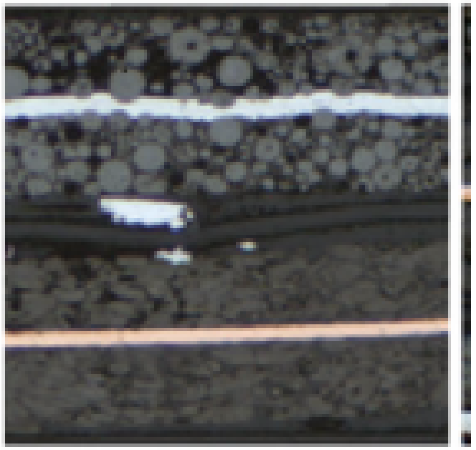
Functioning component?

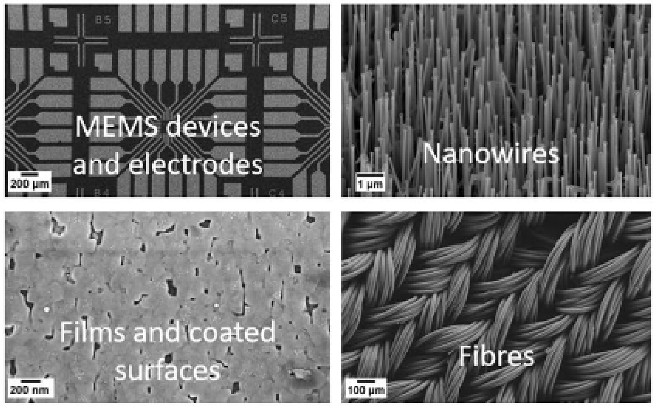


What kind of defect?

Literature review

Literature	Objective	Approach	Key Findings	Relevance to the Project
A Comprehensive Review of Convolutional Neural Networks for Defect Detection in Industrial Applications	Industrial defect detection	CNNs, transfer learning	Overview of models & best practices	Methodological foundation
Image-based defect detection in lithium-ion battery electrode using convolutional neural networks	Battery electrode defects	CNNs, transfer learning	Up to 0.99 F1 score	Strong task similarity
Neural Network for Nanoscience Scanning Electron Microscope Image Recognition	SEM image classification	CNNs, transfer learning	85–95% accuracy	Comparable data & images



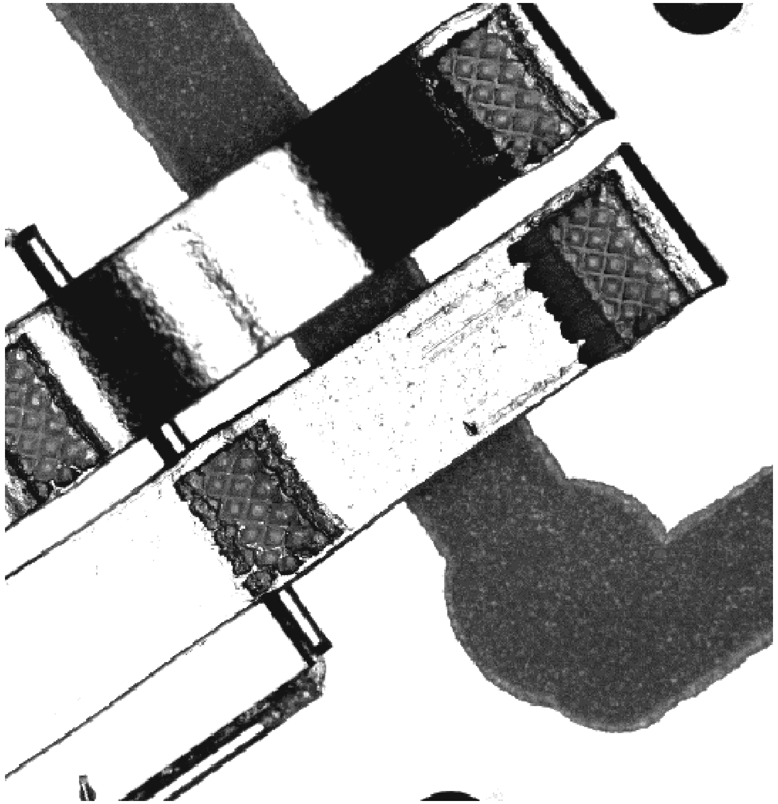


- Typical ImageNet Accuracies: 75 – 85 %

Dataset characteristics

- ≈8300 images
  - Varying resolution ~500x500 px to 1200x1200 px
  - Images are 8bit grayscale
- Labels and further features are provided as.csv
  - Linked via filename
- features:
  - **Images – image data itself**
  - Lib – type of component / part (die)
  - Window – year
- No feature engineering is performed
- Preprocessing:
  - Normalization
  - Resolution downsampling

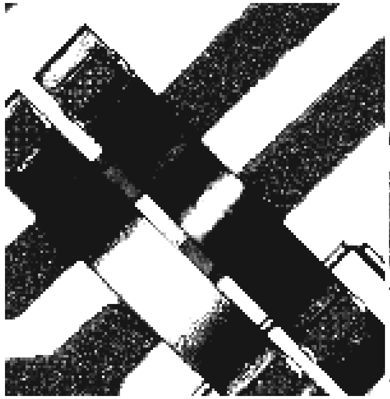
Label: Flat Loop



Window: 2005 , Lib: Die01

Dataset characteristics

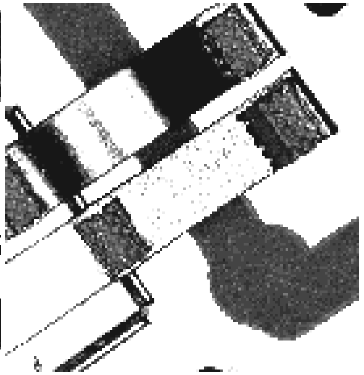
Overview of image classes



0_GOOD

Fully **functioning** including bridge and all connections and thin film layers.

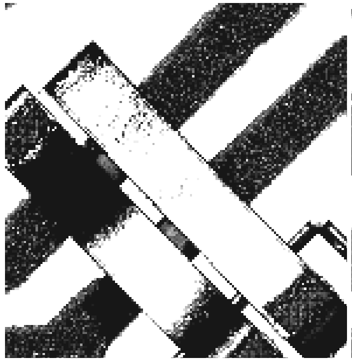
1235 images



1_Flat loop

Bridge has different appearance. Likely laying flat instead of arching

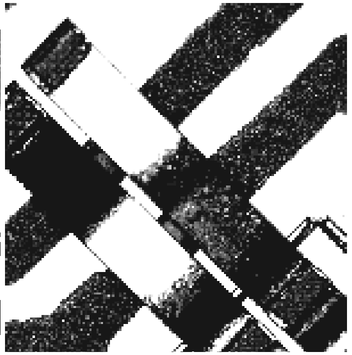
71 images



2_White lift-off

Bridge arch shows irregularity: appears more bright.
Contact wel missing

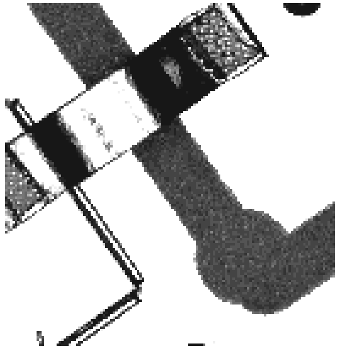
270 images



3_Black lift-off

Bridge arch shows irregularity: appears more dark

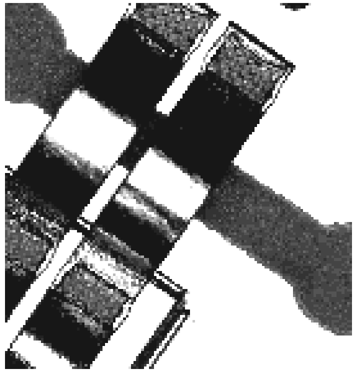
104 images



4_Missing

One or two parts of the bridge are missing.

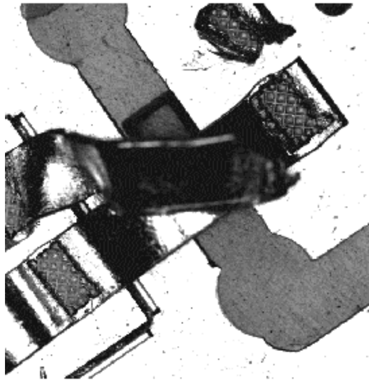
6472 images



5_Short circuit MOS

Name indicates unwanted contact. Visually not uniquely identifyable

126 images



6_Drift

All data not belonging to classes 0 – 5. Including faulty images and otherwise damaged parts.

~ 55
(different dataset)

highly imbalanced , visually close

Baseline model

Traditional machine learning methods (e.g. decision trees or random forests) would require **manual** image preprocessing and feature extraction.

- Very simple convolutional neural network (CNN)
  - Architecture:
    - 1 conv. Layer - maxpooling – 1 dense layer – output layer
- No data augmentation
- Images resized to 128 x 128 px
- Trained on class-balanced dataset

```
model_simple_CNN = tf.keras.Sequential([
    tf.keras.layers.Input((target_size[0], target_size[1], 1)),
    tf.keras.layers.Conv2D(64, (3, 3), activation="relu"),
    tf.keras.layers.MaxPool2D((2, 2)),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(64, activation="relu"),
    tf.keras.layers.Dense(num_classes, activation="softmax"),
])
```

precision	recall	**f1-score**
0.85	0.85	**0.85**

- Surprisingly good performance!
  - Easy or hard to improve upon?

Model definition and evaluation

Model	Architecture	No. parameters
Simple CNN 01	1 Conv layer, 1 Dense layer	89,719,878
Simple CNN 02	2 Conv layers, 1 Dense layer (more neurons)	43,674,886
Simple CNN 03	3 Conv layers, 2 Dense layers	20,171,846
InceptionV3 transfer	InceptionV3 + dropout + classifier	21,815,078
EfficientNetV2S transfer	EfficientNetV2S + dropout + classifier	20,339,046

Model definition and evaluation

Metrics used

- No defective parts should be labelled as functioning
- No functioning part should be labelled as defect

- For class: 0_Good **precision** is best metric
- For all defective classes **recall** is the best metric

- Typically metrics are averaged over all classes

F1 score as main metric

Training Criteria

- Model performance does not always increase (for validation metrics, i.e. overfitting

- Early stopping is used
  - Based on val_loss OR val_F1

	precision	recall	f1-score	support
0_GOOD	0.89	0.94	0.91	17
1_Flat loop	0.93	0.78	0.85	18
2_White lift-off	0.77	0.71	0.74	14
3_Black lift-off	0.76	1.00	0.87	13
4_Missing	1.00	1.00	1.00	9
5_Short circuit MOS	1.00	0.93	0.97	15
accuracy			0.88	86
macro avg	0.89	0.89	0.89	86
weighted avg	0.89	0.88	0.88	86

Full classification report for better understanding

Model definition and evaluation

Implementation of best performing model

```
# inception model where layers are successively unfrozen during training

# Load pre-trained InceptionV3 with correct input size
base_transfer_model_4 = tf.keras.applications.InceptionV3(
    weights='imagenet',
    include_top=False,
    input_shape=(target_size[0], target_size[0], 3)
)

# Freeze all layers of Inception model
base_transfer_model_4.trainable = False

# Simple classification head
# - GlobalAveragePooling2D reduces spatial dimensions
# - Final Dense layer maps to class probabilities
inception_multiPhase_fine_tune_model = tf.keras.Sequential([
    base_transfer_model_4,
    tf.keras.layers.GlobalAveragePooling2D(),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(num_classes, activation = 'softmax')
])
```

Training in Multiple steps

1. Train Classification layer only
    - “High” learning rate, few epochs
2. Train last 30 layers of Inception

```
#set last 30 layers to be trainable
for layer in base_transfer_model_4.layers[-30:]:
    layer.trainable = True
```

    - “High” learning rate, medium epoch No.
3. Train last 100 layers of inception
    - Low learning rate, high epoch number (patience)

Results

Classification report – best performing model

Label	precision	recall	f1-score	support
0_GOOD	**0.99**	0.99	0.99	238
1_Flat loop	0.85	0.73	0.79	15
2_White lift-off	0.93	**0.97**	0.95	57
3_Black lift-off	0.87	**1.00**	0.93	13
4_Missing	1.00	**0.99**	0.99	1316
5_Short circuit MOS	0.89	**0.94**	0.91	17
accuracy	0.99	0.99	0.99	0.99
macro avg	0.92	0.94	**0.93**	1656
weighted avg	0.99	0.99	**0.99**	1656

Ok …

- InceptionV3 transfer learning
- ImageNet weights as starting point
- 3 phases of training
- Target resolution 299 x 299 px
- Full dataset with class weights
- Mild data augmentation

Results

Best F1 score for each model with corresponding train parameters

model	S-CNN 01	S-CNN 02	S-CNN 03	IncV3 feat.	IncV3 mult.
F1 score	**0.92**	**0.90**	**0.94**	**0.92**	**0.93**
Param	299 x 299 px Balanced DS Focal loss	128 x 128 px Balanced DS	256 x 256 px Balanced DS	256 x 256 px Balanced DS	299 x 299 px Unbalanced DS Class weights

- Models have best performances at different parameters
- Augmentation has little to no effect (overfitting not limiting factor)
- Can the values from balanced dataset be trusted?

Results

Model F1 scores for given training parameters

model	S-CNN 01	S-CNN 02	S-CNN 03	IncV3 feat.	IncV3 mult.	EfficientNet V2S mult.
F1 score	**0.84**	**0.90**	**0.79**	**0.72**	**0.93**	**0.85**

Parameters:

- Target resolution 299x299 px
- Mild data augmentation
- **Unbalanced Dataset**
- Using class weights
- ‘normal’ loss (categorical crossentropy)

- Multi Phase transfer learning based on Inception V3 architecture is clearly best model

- Larger dataset (unbalanced) give higher trust in the results

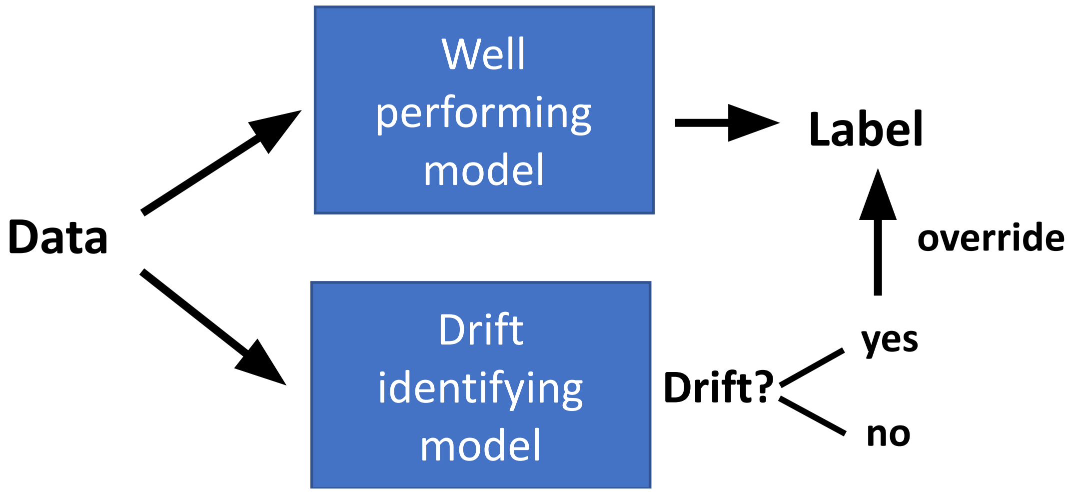
Results

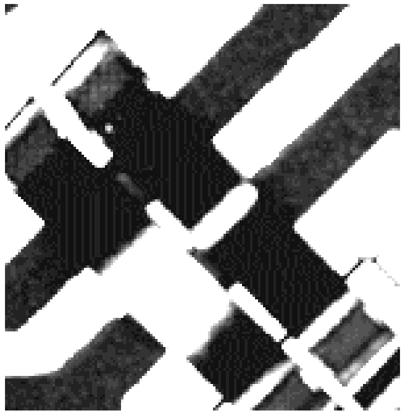
Classifying the drift class

- No Drift class images in training set
  - Only occur in public and private test sets for submission
  - No true labels available for these sets

- Manually label the test set (focus in drift)
- Summarize labels 0-5 into “regular”
- Train a Model on the newly available dataset with 2 classes

- Transfer learning from previous best performing model
  - So far **only F1 score of 0.4** for the drift class achieved
- Drift images SHOULD be labelled correctly (heavily penalized in submission)





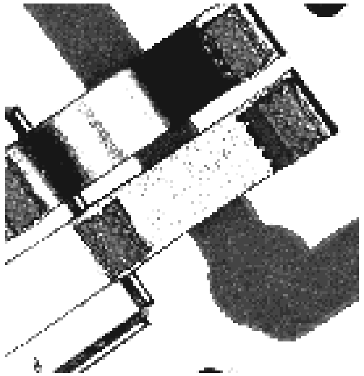
Challenges and errors

solved

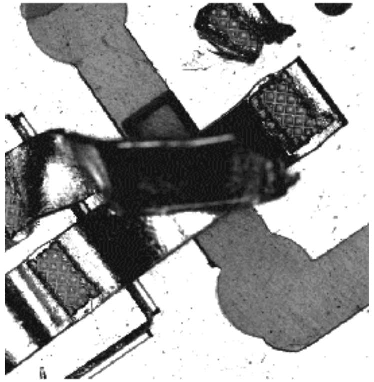
- Augmentation massively reduced performance
  - Removed image flipping
- Elegant and effective way of handling image data + table
  - Legacy method: ImageDataGenerator + pandas dataframes
- Transfer learning performance not as expected
  - Used multi phase learning
- Repeated training of models stopped early unpredictably
  - Delete variables: history, model, memory handling

ongoing

- Validation metrics start out 'too good'
  - Used for early stopping
- Low performance on Drift class detection
- Low performance on flat loop class



flat loop



Drift

Discussion

- Model F1 score could be increased
- Model performance depends strongly on hyperparameters AND training methodology
- Computer Vision is (as expected) a prime example for successful machine learning application

- When can (good) model performance be “trusted”?
  - Will it translate to a wholly independent dataset?

Conclusion and outlook

- The classification task was solved with good performance using CNNs
- Model performance depends strongly on hyperparameters AND training methodology
- Computer Vision is (as expected) a prime example for successful machine learning application

- Usage of more modern model architectures (i.e. transformers)
- Systematically and automatically vary hyperparameters
- Implement other methods to tackle low amount of training data
- Different methods of identifying drift data?
- Custom loss functions

Thank you for the attention!

… any questions?