ME 536 – Design of Intelligent Machines

Term Project - Defect Finder Camera

Presenter: Çağdaş Güven

ID:2738938

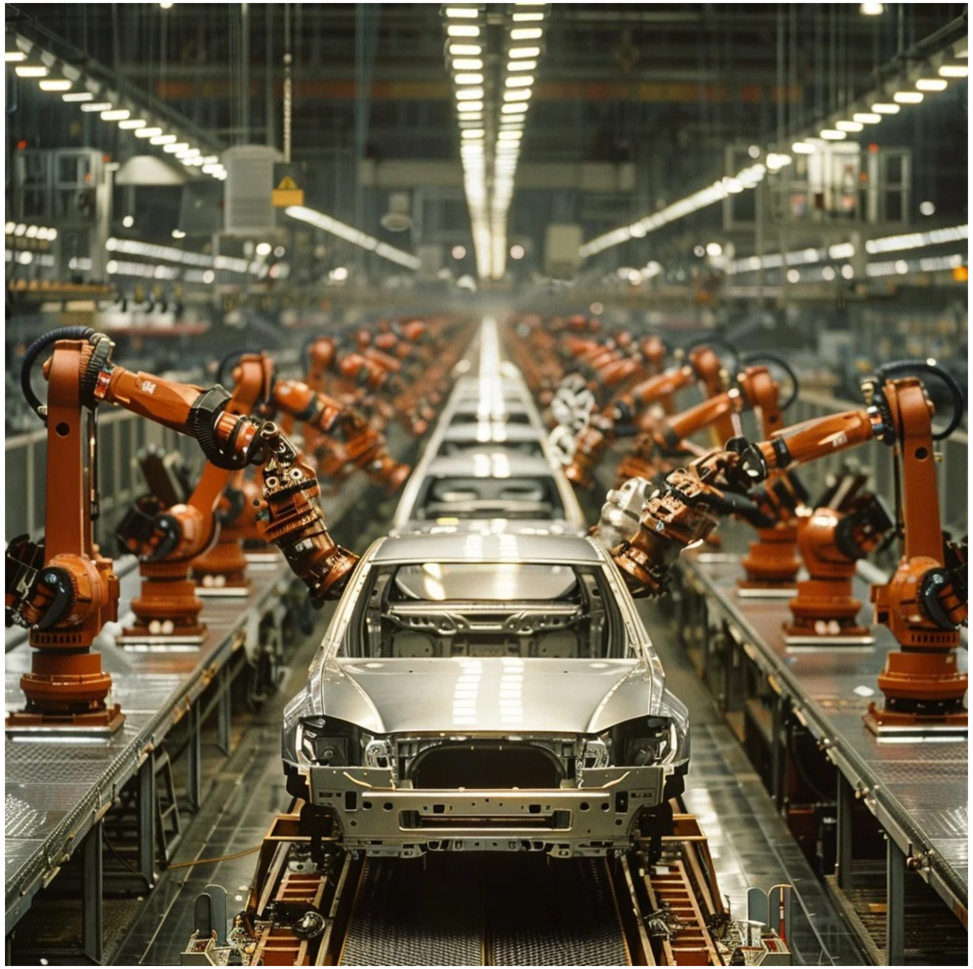
Date: 30/01/2025

Github: https://github.com/amnessa/MECH536/tree/e4cba326a3cc35476c5539c02b752f3d851cce0e/me_term_project

Fall 2024 - METU

Introduction & Problem Statement

- In a production line a camera system detect surface defects
- Technician reports a new defect



https://stockcake.com/i/automated-car-assembly_1004278_117389

How It Meets Project Requirements

- Principal Component Analysis - PCA
- Clustering - (K- means)
- A bit of Image Processing - (histogram equalization and resizing)
- Artificial Neural Networks - ANN (VGG16 fc1 layer)

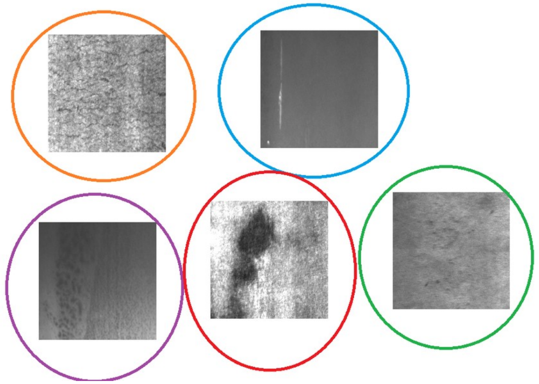


figure 1 - old data cluster representation

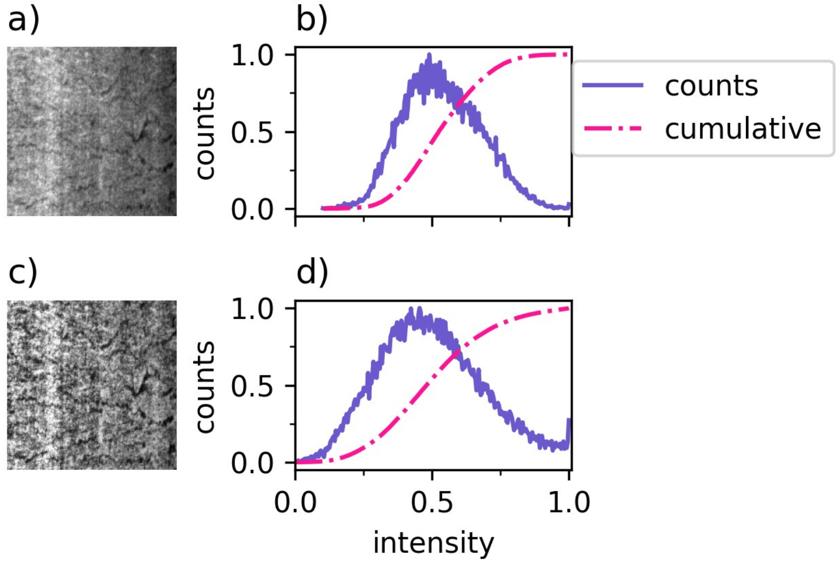
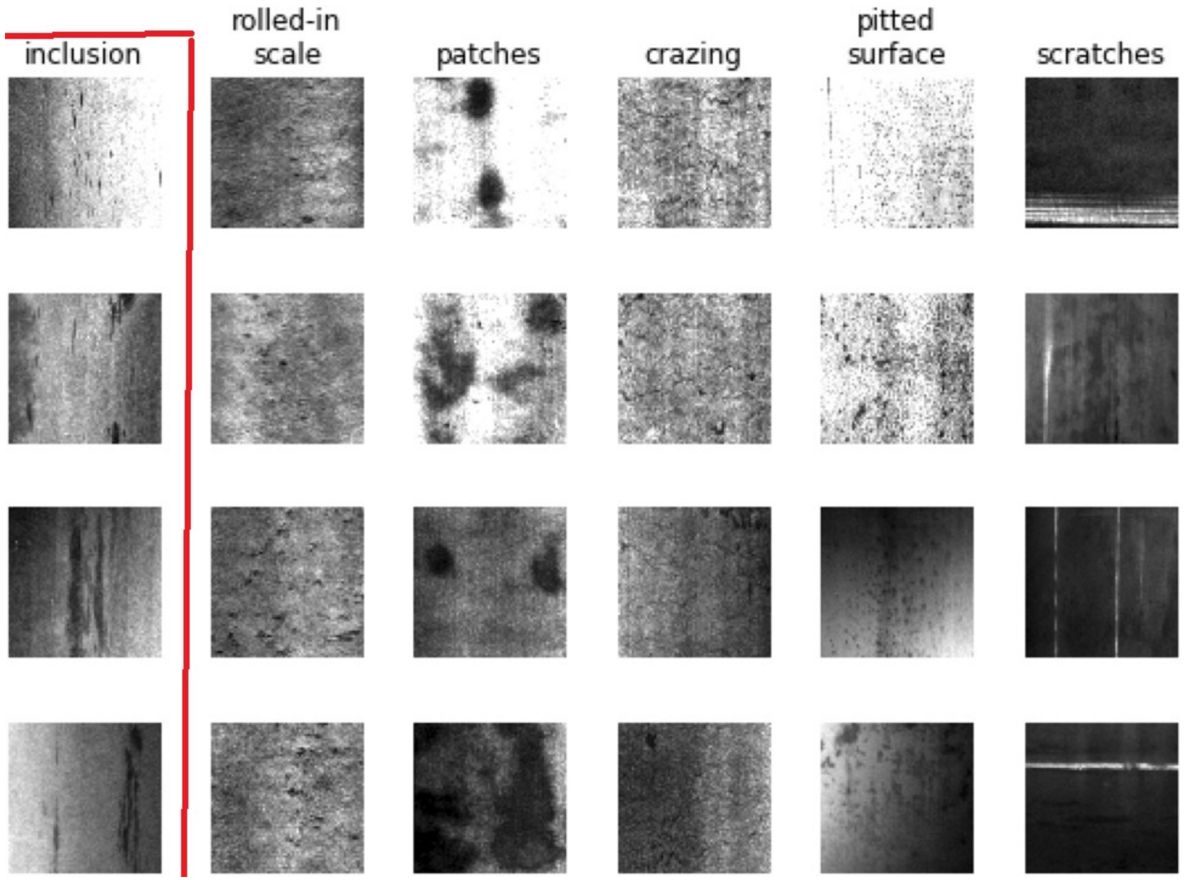


figure 2 - Histogram Equalization to spread intensities and enhance contrast

Data & Generation

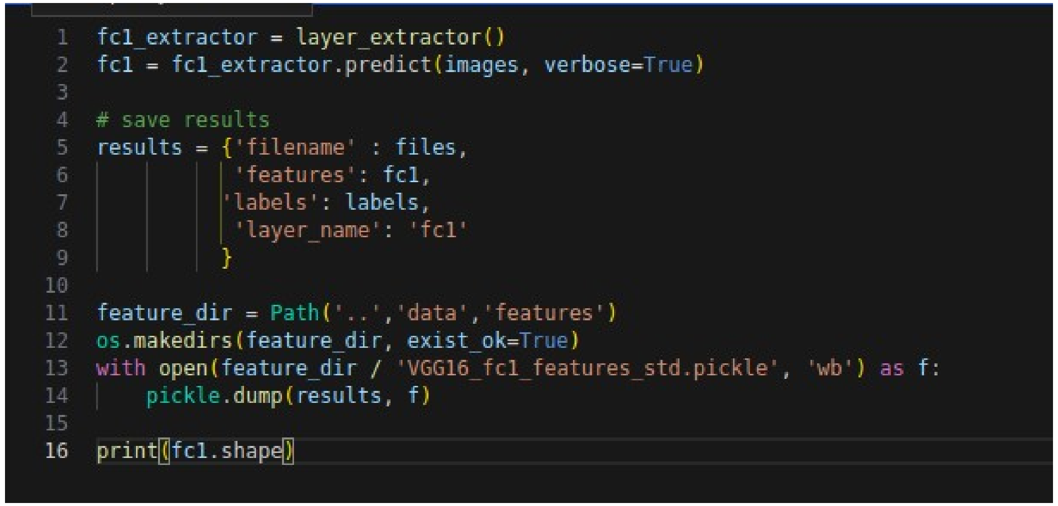
- North East University Dataset for Defect Detection - NEU Dataset
- Histogram Equalization
- Resize
- Store



Zhao, Weidong, Chen, Feng, Huang, Hancheng, Li, Dan, Cheng, Wei, A New Steel Defect Detection Algorithm Based on Deep Learning, Computational Intelligence and Neuroscience, 2021, 5592878, 13 pages, 2021. https://doi.org/10.1155/2021/5592878

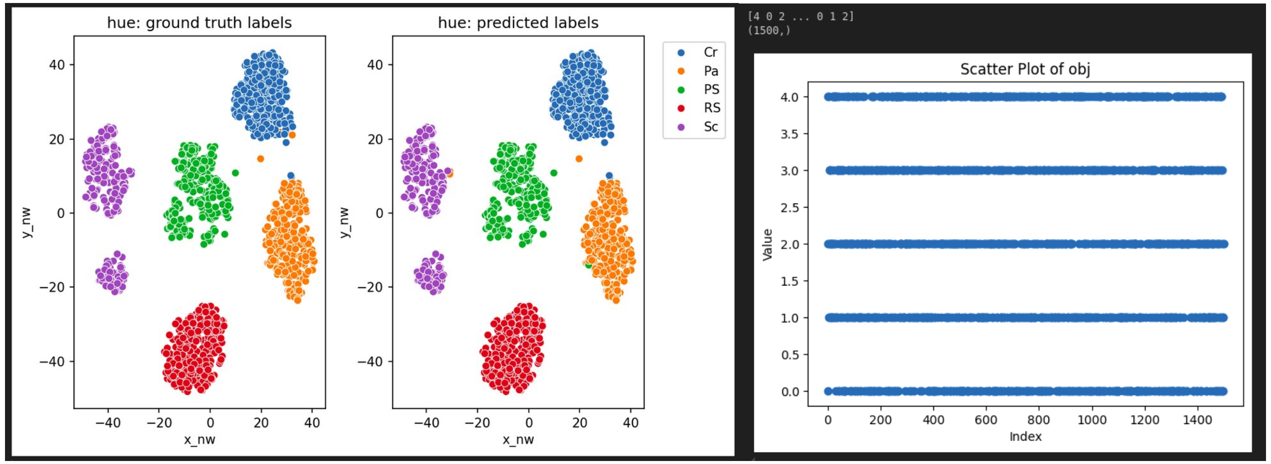
Methods & Network Structure

- VGG16 pretrained on ImageNet
- Filter out features
- PCA 4096 -> 50 (SVD = Full)
- “Whitening in PCA so each principal component has unit variance, preventing large components from dominating.”
- Result: 5 clusters



```python
fc1_extractor = layer_extractor()
fc1 = fc1_extractor.predict(images, verbose=True)

# save results
results = {'filename' : files,
            'features': fc1,
           'labels': labels,
            'layer_name': 'fc1'
           }

feature_dir = Path('..','data','features')
os.makedirs(feature_dir, exist_ok=True)
with open(feature_dir / 'VGG16_fc1_features_std.pickle', 'wb') as f:
    pickle.dump(results, f)

print(fc1.shape)
```

figure 3 - code part for feature extraction



```
[4 0 2 ... 0 1 2]
(1500,)
```

figure 4 - t-SNE and scatter plot images for cluster confirmation

Novelty Detection Approach

- New Batch of 40 image
- Same preprocess (ANN fc1-PCA - Kmeans)
- Old cluster centers , New cluster centers
- Automatically detect the novel and cluster it

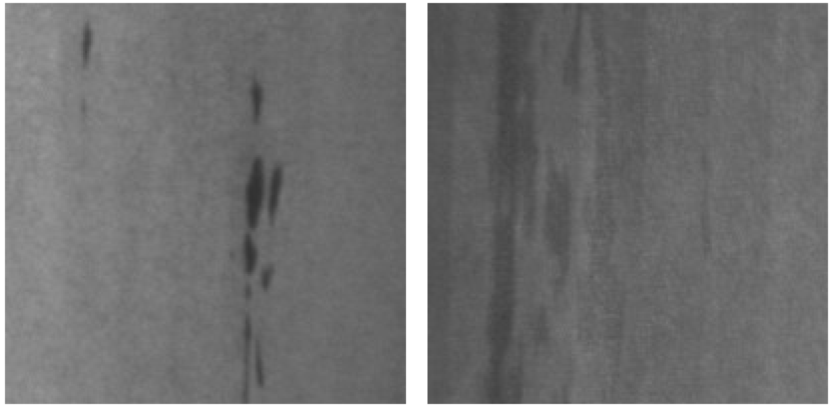
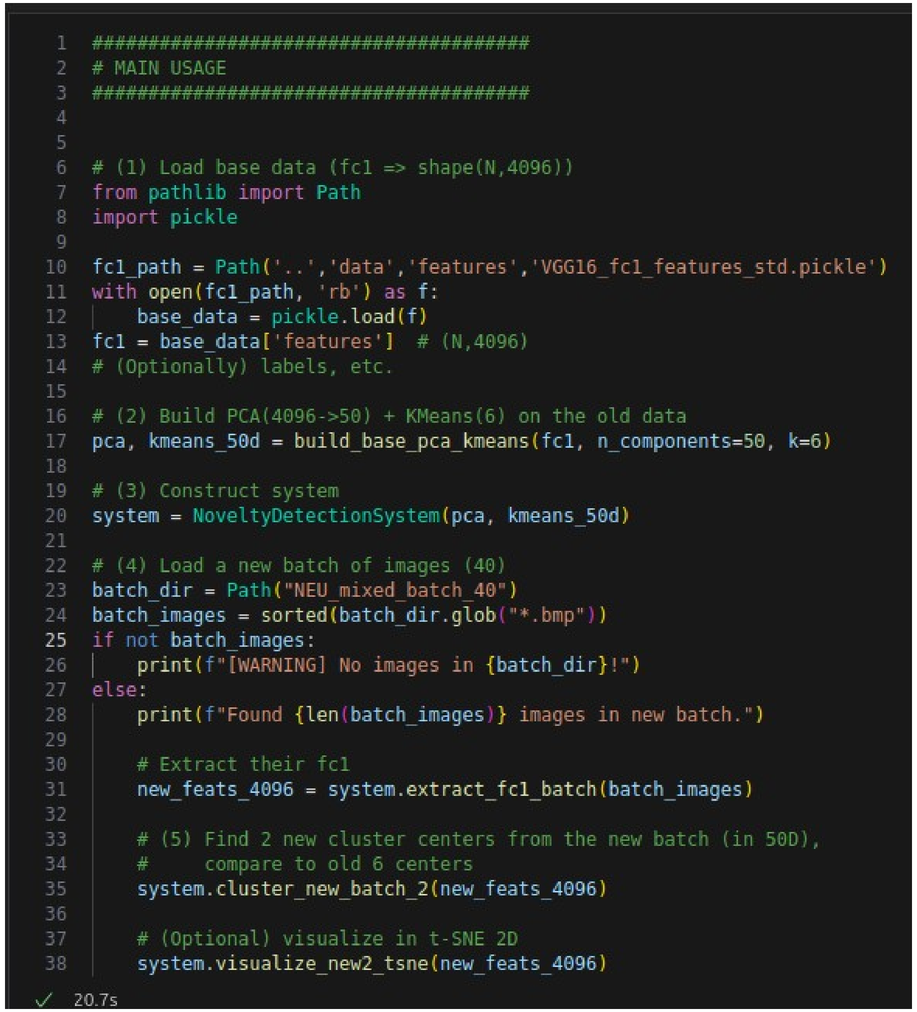


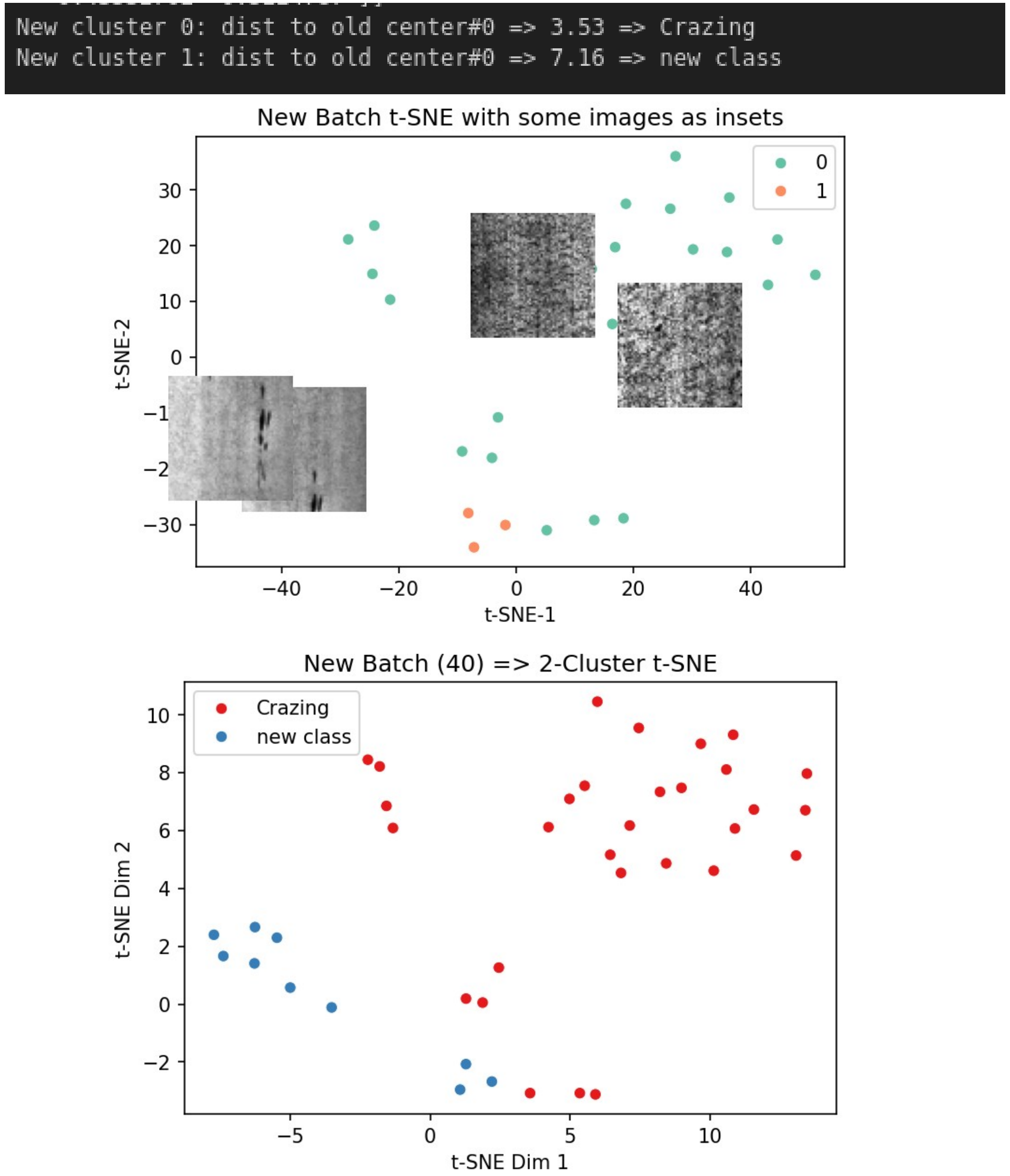
figure 5 - example new data mixed with old both are member of the same defect class in real life

Results & Demo



```
#######################################
# MAIN USAGE
#######################################


# (1) Load base data (fc1 => shape(N,4096))
from pathlib import Path
import pickle

fc1_path = Path('..','data','features','VGG16_fc1_features_std.pickle')
with open(fc1_path, 'rb') as f:
    base_data = pickle.load(f)
fc1 = base_data['features']  # (N,4096)
# (Optionally) labels, etc.

# (2) Build PCA(4096->50) + KMeans(6) on the old data
pca, kmeans_50d = build_base_pca_kmeans(fc1, n_components=50, k=6)

# (3) Construct system
system = NoveltyDetectionSystem(pca, kmeans_50d)

# (4) Load a new batch of images (40)
batch_dir = Path("NEU_mixed_batch_40")
batch_images = sorted(batch_dir.glob("*.bmp"))
if not batch_images:
    print(f"[WARNING] No images in {batch_dir}!")
else:
    print(f"Found {len(batch_images)} images in new batch.")

    # Extract their fc1
    new_feats_4096 = system.extract_fc1_batch(batch_images)

    # (5) Find 2 new cluster centers from the new batch (in 50D),
    #     compare to old 6 centers
    system.cluster_new_batch_2(new_feats_4096)

    # (Optional) visualize in t-SNE 2D
    system.visualize_new2_tsne(new_feats_4096)
```

figure 6 - calling system from main



```
New cluster 0: dist to old center#0 => 3.53 => Crazing
New cluster 1: dist to old center#0 => 7.16 => new class
```

figure 7 results from testing

Possible Improvements

- Dynamic tresholding
- Dynamic Clustering
- Synthetic generation from dataset
- Sub analysis on clustered new data
- Alternative to Kmeans

Ending & Questions

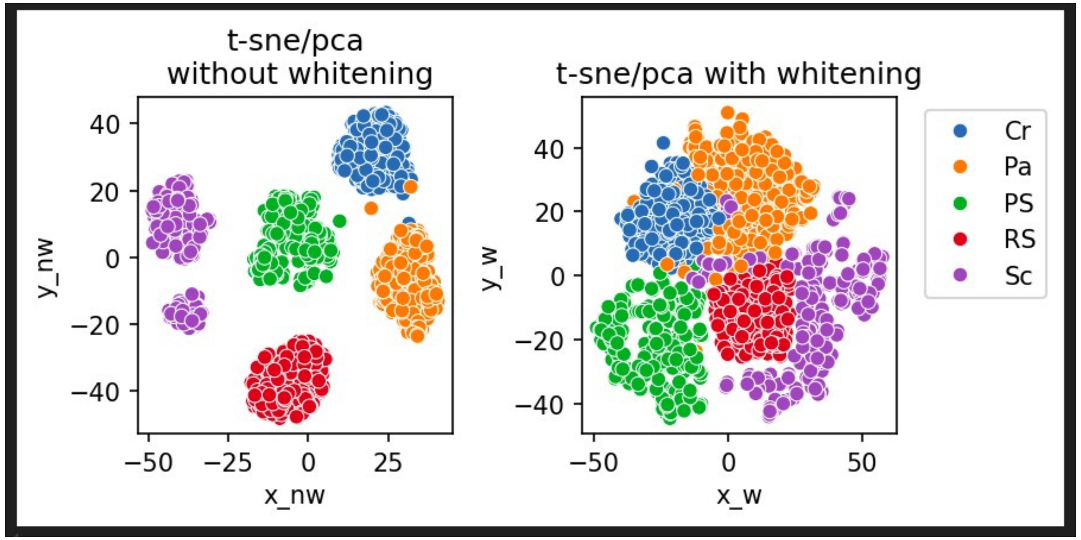
- Thank you for listening !

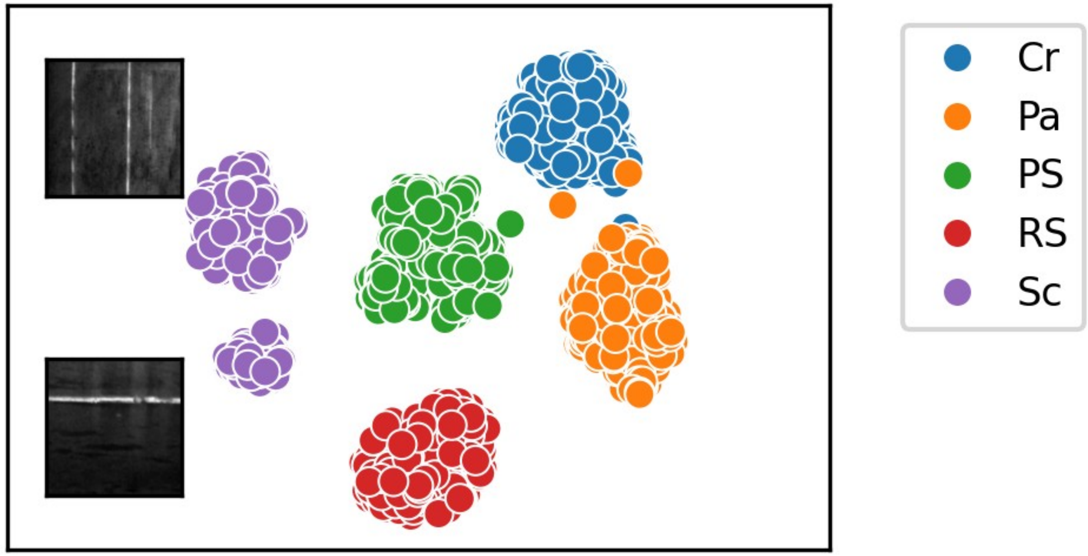
https://makeameme.org/meme/thank-you-mr-

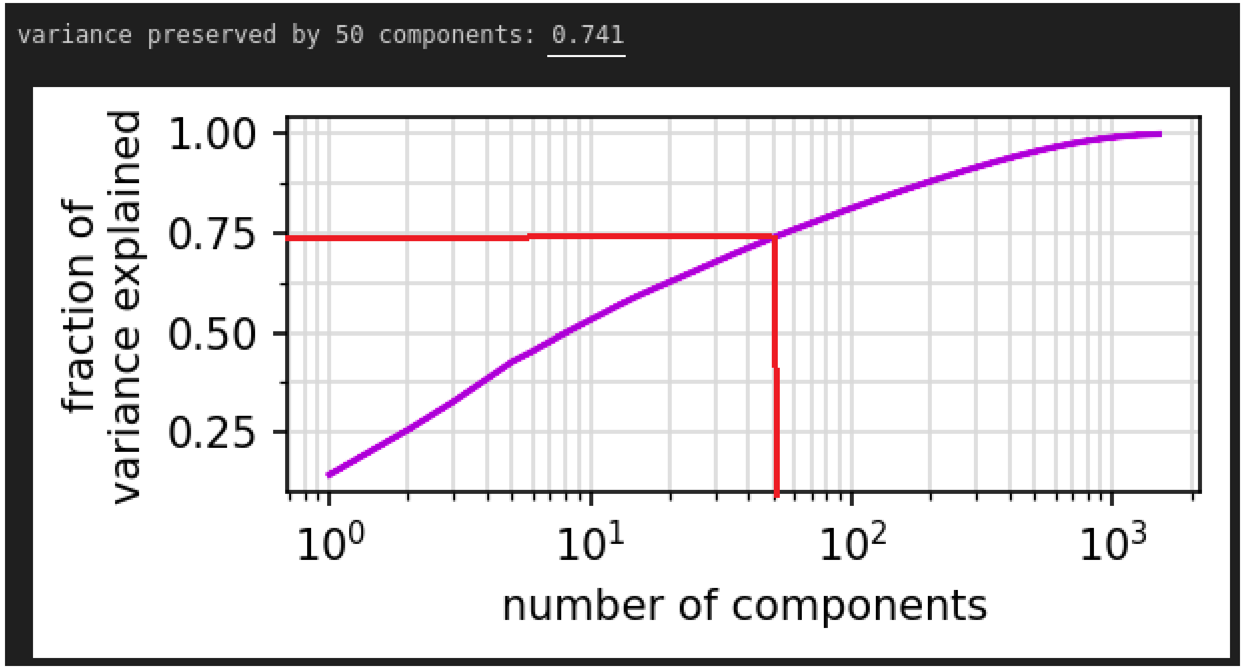
References

- 1- Cohn, R., Holm, E. Unsupervised Machine Learning Via Transfer Learning and k-Means Clustering to Classify Materials Image Data. Integr Mater Manuf Innov 10, 231–244 (2021). https://doi.org/10.1007/s40192-021-00205-8
- 2- van der Maaten L, Hinton G (2008) Visualizing data using t-SNE. J Mach Learn Res 9:2579–2605. http://www.jmlr.org/papers/v9/vandermaaten08a.html
- 3- Pelleg D, Pelleg D, Moore A (2000) X-means: extending K-means with efficient estimation of the number of clusters. In: Proceedings of the 17th international conf. on machine learning, pp 727–734. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.19.3377
- 4- Pedregosa F et al. (2011) Scikit-learn: machine learning in python. J Mach Learn Res 12:2825–2830. http://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html
- 5- Franccois Chollet and Etc. Keras. 2015. https://keras.io
- 6- van der Walt S et al (2014) scikit-image: image processing in Python. PeerJ 2, e453. https://doi.org/10.7717/peerj.453. https://peerj.com/articles/453
- 7- Xiao M et al (2017) An evolutionary classifier for steel surface defects with small sample set. Eurasip J Image Video Process 2017(1):48. https://doi.org/10.1186/s13640-017-0197-y
- 8- Gao Y et al (2020) A semi-supervised convolutional neural network-based method for steel surface defect recognition. Robot Comput Int Manuf 61:101825. https://doi.org/10.1016/j.rcim.2019.101825
- 9- Metal Surface Defects https://www.kaggle.com/code/kirollosashraf/metal-surface-defects
- 10- Zhao, Weidong, Chen, Feng, Huang, Hancheng, Li, Dan, Cheng, Wei, A New Steel Defect Detection Algorithm Based on Deep Learning, Computational Intelligence and Neuroscience, 2021, 5592878, 13 pages, 2021. https://doi.org/10.1155/2021/5592878

EXTRAS







```
variance preserved by 50 components: 0.741
```