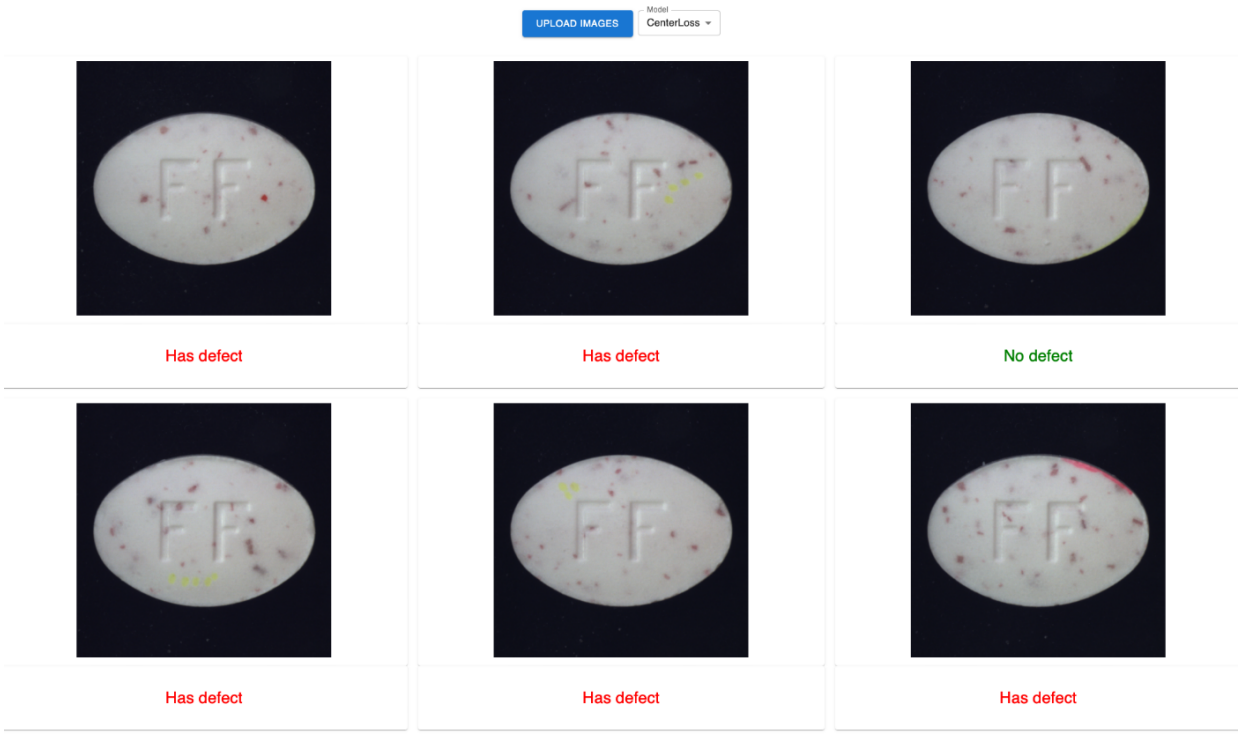
Full-Stack Image Anomaly Detection Web App

Project Overview

This project is a web application for anomaly detection in pill images from the MVTec Anomaly Detection Dataset[1]. The applications provides

- An interface where users can upload multiple images at the same time.
- Predictions of whether each image contains defects based on the judgement of the selected models
- A dropdown where users can choose and deploy models/algorithms of choice. Six total models are provided with **CenterLoss**, the best performing model as default.



Implementation

The classification algorithm used in this project is Deep Metric Learning following data augmentation and training steps described in this Medium[2].

The classical self-supervised binary classification was first tested, but showed accuracy of 0.44 which is worse than random guessing. Training CNN model to classify between the training data (labeled as no defect) and data augmented from the training data (add small colored mark at random position, labeled as has defect) is proved to be inefficient as the small marks added to the 'has defect' augmented data are not discriminative enough for small training set. This technique is can accurately detect random images outside of the dataset as 'has defect' as they are much different but struggles when it come to classifying pill image with small defects.

Instead, Deep Metric Learning is used. Deep Metric Learning has the benefits of being simple and easier to train by just

tweaking conventional neural network. It also does not require extra computational power for further training the model making it suitable for the given task. The implementation steps are as follows,

CANDIDATE MODELS

The candidate methods for models are,

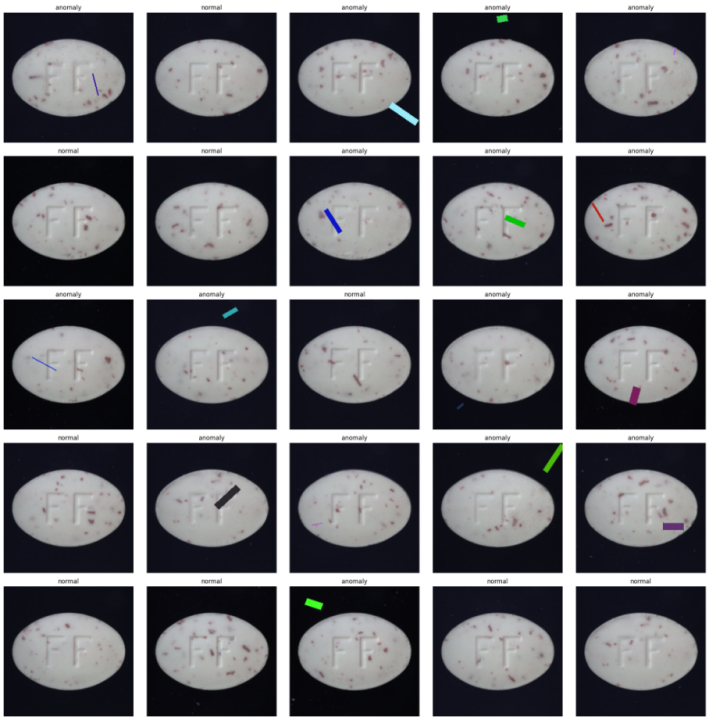
- Conventional CNN
- L2-constrained SoftMax Loss[3]
- ArcFace[4]
- CosFace[5]
- SphereFace[6]
- CenterLoss[7]

From evaluation results, **CenterLoss** provides best performance and is seleced as the default model in this application. Other methods can be chosen from the dropdown interface.

DATA AUGMENTATION AND TRAINING

The training dataset contains 267 images of 'no defect' pill. To train the model to be able to distinguish between images with and without small defects, we followed the augmentation technique and steps used in [2].

- The training data is first duplicated for augmentation.
- Augment all images in augmentation set with small colored mark of random width and random position.
- The augmentation set is labeled as 'has defect' and is used to train the neural networks in addition to the original 'no defect' class. 20% of training-augmentation data pairs are set aside as validation set.



- All models are based on ResNet18 and are implemented using FastAI Learner library [8]. The model is trained on Intel

Xeon Silver 4108 1.8GHz, 8-core x 2, 2TB SSD (CPU) and NVIDIA TITAN RTX 24GB x 2 (GPU).

- The source code for model training and evaluation is included in the notebook `server/ml/training.ipynb`

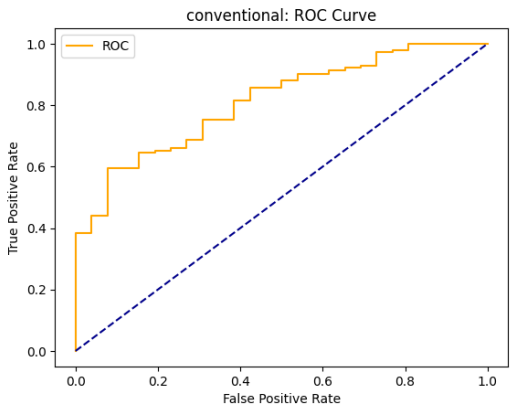
EVALUATION

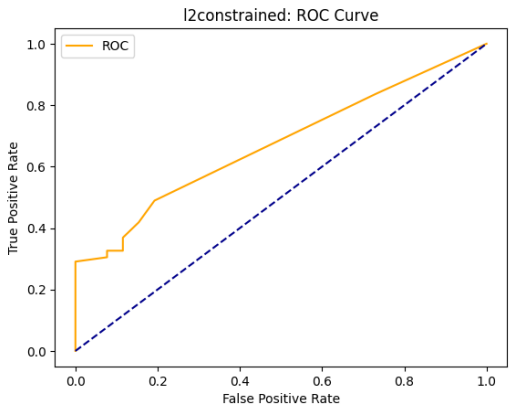
1. Detach the last linear layer (512-d to 2-d) from the trained model. By doing this, the output of the trained model for each image will be a 512-d embedding vector.
2. Calculate embedding vectors for all training images (267 images) and testing images .
3. For each test embedding vector, calculate the cosine distance between it and each train embedding vector. Take the distance to the closest training vector as the distance score from the 'no defect' class for this embedding vector. Iterate over all test embedding vector to calculate distance scores for each of the test images.
4. Using the distance scores of all test images, draw ROC curve and calculate area under the curve. Since accuracy depends on the classification threshold of distance score, AUC is instead used as the evaluation metric in our case. AUC captures areas under the ROC curve which is a plot between True Positive Rate on the y-axis and False Positive Rate on the x-axis. It provides an aggregate measure of model performance across all possible thresholds with AUC score of 1.0 indicating the perfect classifier across all classification thresholds. The AUC scores for each candidate models are as follows. From our experiment results, CenterLoss delivers best performance with AUC of 85.73%.

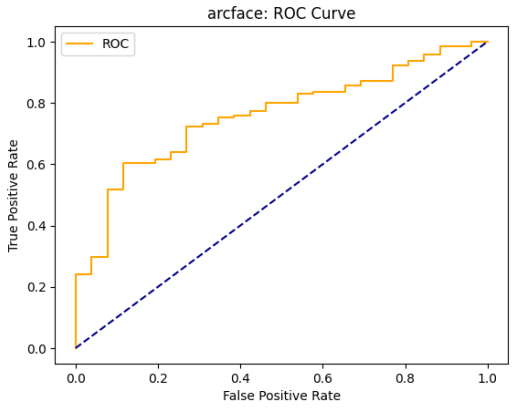
Models	AUC
Conventional	0.81124
L2-constrained	0.67239
ArcFace	0.75968
CosFace	0.83906
SphereFace	0.78396
CenterLoss	**0.85734**

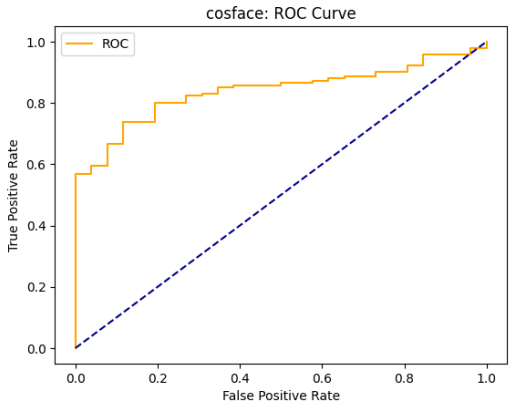
5. To deploy the model in our application, optimal thresholds for each model deployment are chosen as the thresholds which yield the maximum sum between True Positive Rate and True Negative Rate. The optimal threshold for each model is as follows.

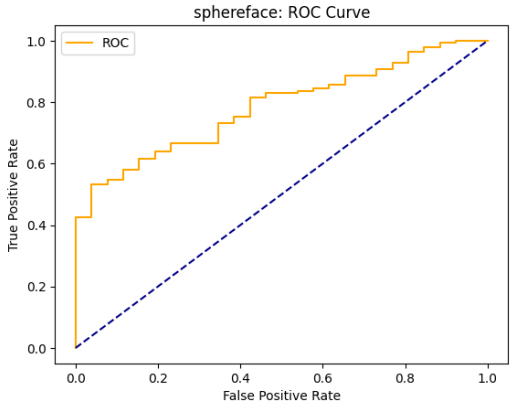
Models	Optimal Threshold
Conventional	0.18666
L2-constrained	5.96E-08
ArcFace	0.01611
CosFace	0.01687
SphereFace	0.1042
CenterLoss	0.1474

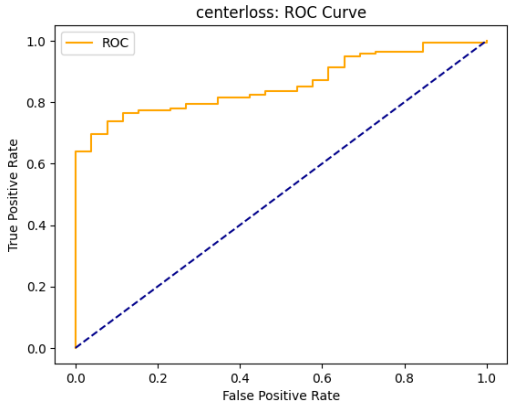












Application Setup

1. Make sure Docker is installed and has daemon running on the machine in use.
2. Move to the root folder of this application on Terminal and run `docker-compose up`
3. Open browser to `http://localhost:3000`. The application should be up and ready for image uploads.

Training Notebook Setup

The source code for model training and evaluation is included in the notebook `server/ml/training.ipynb`

1. Download and unzip the pill dataset: https://www.mydrive.ch/shares/43421/11a215a5749fcfb75e331ddd5f8e43ee/download/420938129-1629953099/pill.tar.xz
2. Move the unzipped `pill` folder inside the folder `server/ml/dataset`.
3. Make sure the notebook is run on the virtual environment kernel.

References

[1] https://www.mvtec.com/company/research/datasets/mvtec-ad/
[2] https://medium.com/analytics-vidhya/spotting-defects-deep-metric-learning-solution-for-mvtec-anomaly-detection-dataset-c77691beb1eb
[3] R. Ranjan, C. D. Castillo, and R. Chellappa. L2-constrained softmax loss for discriminative face verification. arXiv preprint arXiv:1703.09507, 2017. https://arxiv.org/pdf/1703.09507.pdf
[4] J. Deng, J. Guo, and S. Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. arXiv preprint arXiv:1801.07698, 2018. https://arxiv.org/pdf/1801.07698.pdf
[5] H. Wang, Y. Wang, Z. Zhou, X. Ji, D. Gong, J. Zhou, Z. Li, and W. Liu, CosFace: Large Margin Cosine Loss for Deep Face Recognition, arXiv preprint arXiv:1801.09414, 2018. https://arxiv.org/pdf/1801.09414.pdf
[6] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song. SphereFace: Deep Hypersphere Embedding for Face Recognition. In CVPR, 2017. https://arxiv.org/pdf/1704.08063.pdf
[7] Y. Wen, K. Zhang, Z. Li, and Y. Qiao, A discriminative feature learning approach for deep face recognition," in European Conference on Computer Vision. Springer, 2016, pp. 499–515. https://ydwen.github.io/papers/WenECCV16.pdf
[8] https://www.fast.ai