

Introduction to Pattern Recognition and Anomaly Detection

Good Morning!

- 1) Welcome to Day 3!
- 2) We will start at 9am sharp

Grab your coffee/team, sit back and relax for now 😊



Programme

Day 1	<p>Introduction to Pattern Recognition and their techniques Activity – KNN & K-Means (iris)</p> <p>Common Python Lib - Numpy Activity - Numpy</p>	<p>Common Python Lib – Pandas, matplotlib, Scikit-learn Activity – Pandas, Matplotlib, Scikit-learn</p>
Day 2	<p>Data gathering to prepare for training and testing data Activity – Data Cleaning</p> <p>Use low-pass filter and simple moving average to detect abnormalities Activity – Simple Anomaly Detector</p>	<p>Introduction to anomaly detection and their techniques Introduction to H2O</p> <p>Activity - Activity - IRIS, MNIST, Fashion MNIST</p>
Day 3	<p>Activity – Credit card fraud with H2O/Isolation forest</p> <p>Activity – Intrusion detection system</p>	<p>Introduction to Alibi-Detect</p> <p>Activity – Intrusion detection system Activity – Image anomaly detection</p>



Support Metrics

- H2O supports a number of metrics, ways to measure a model's usefulness.
- Before we get into parameters it is worth taking a look at them because when we talk about “scoring a model” we mean evaluating on one of these metrics.

<http://h2o-release.s3.amazonaws.com/h2o/master/3484/docs-website/h2o-py/docs/modeling.html#h2odeeplearningestimator>



Practical
Machine
Learning with
H2O

Regression	Classification
<p>RMSE - The square root of MSE. If your response variable units are dollars, the units of MSE is dollars-squared, but RMSE is back into dollars</p>	<p>Misclassification - This is the overall error, the number shown in the bottom right of a confusion matrix. If it got 1 of 20 wrong in class A, 1 of 50 wrong in class B, and 2 of 30 wrong in class C, it got 4 wrong in total out of 100, so the misclassification is 4, or 4%.</p>
<p>MAE - Mean Absolute Error. Following on from the MSE example, a guess of 1,4,3 has absolute errors of 1, so the MAE is 1. But 2,3,6 has absolute errors of 0,0,2 so the MAE is 0.667. As with RMSE, the units are the same as your response variable.</p>	<p>Mean_per_class_error - The right column in a confusion matrix has an error rate for each class. This is the average of them, so for the preceding example it is the mean of 1/20, 1/50, and 2/30, which is 4.556%. If your classes are balanced (exactly the same size) it is identical to misclassification.</p>
<p>R2 - R-squared, also written as R^2, and also known as the coefficient of determination. This used to be available as a choice for stopping_metric, but has fallen out of favor</p>	<p>Logloss (default) - The H2O algorithms don't just guess the category, they give a probability for the answer being each category. The confidence assigned to the correct category is used to calculate logloss (and MSE). Logloss disproportionately punishes low numbers, which is another way of saying having high confidence in the wrong answer is a bad thing.</p>
<p>RMSLE - The catchy abbreviation of Root Mean Squared Logarithmic Error. Prefer this to RMSE if an under-prediction is worse than an over-prediction.</p>	<p>MSE - Mean Squared Error. The error is the distance from 1.0 of the probability it suggested. So assume we have three classes, A, B, and C, and your model guesses A with 0.91, B with 0.07, and C with 0.02. If the correct answer was A the error (before being squared) is 0.09, if it is B 0.93, and if C it is 0.98.</p>
	<p>AUC - Area Under Curve</p>



Common Parameters

- The Essentials
 - **x** – Which fields in training_frame to learn from
 - **y** – Which fields in training_frame should be learned (response variable)
 - **training_frame** – the data set to train from
 - **ignore_const_cost** – Defaults to true, meaning if all values in a column are the same value, then ignore that column.
- Effort
 - **epochs** - The amount of training cycles a deep learning algorithm should do
 - **seed** - An integer to control random number generation, which allows you to get exactly the same model if you run the algorithm again. If using deep learning, you also need to set reproducible.
 - **reproducible** - If true, then deep learning will run on a single thread. This takes away the random element, but of course means it will train more slowly. Set this if also setting a seed, don't set it if not.
- Scoring and Validation
 - **validation_frame** - The (handle to the H2O) data set to validate against

Deep learning has a large number of additional scoring parameters. Check H2O documentation



Common Parameters

- Early Stopping
 - **stopping_metric** - How to decide if the model is improving or not.
 - **stopping_tolerance** - Stop if it (your metric of choice) has not improved by at least this much
- Checkpoints
- Cross-Validation
- Data Weighting
- Sampling, Generalizing
- Regression
 - **loss** - Certain distributions (gaussian, laplace, huber) allow you to choose a loss function. The default choices for those three being: “Quadratic,” “Absolute,” “Huber.” The final possible value is “CrossEntropy,” which is for classification. Give the default of “Automatic” unless you have a good reason not to. You cannot define your own loss function.
- Output Control



Common Parameters

- Deep Learning
 - **shuffle_training_data** - Defaults to false. If true then training data is randomly sorted
 - **autoencoder** - Defaults to false (meaning to do supervised learning). Set this to true if auto-encoding. Usually you should set activation to Tanh at the same time.
 - **l1** - L1 regularization. Also known as lasso regularization. Defaults to 0, and typical values to try are 0.0001 or smaller.
 - **train_samples_per_iteration** - This controls how many training rows (samples) to use per iteration; an iteration can be thought of as when the model is scored. It is an integer, and normally you will use one of the following three special values. -2 is the default, and lets H2O decide
 - **activation** – Rectifier (default), Tanh (for autoencoder) or Maxout. Three other type with dropout not discussed here)
 - **hidden** - Hidden layer sizes. The default is 200,200, meaning there will be two hidden layers, with 200 neurons in each layer. If you give 60,40,20 then you will have three hidden layers, with 60 in the first, 40 in the second, and 20 in the third.



H2ODeepLearningEstimator

```
anomaly_model = H2ODeepLearningEstimator(activation = "Tanh",  
                                           hidden = [14,7,7,14],  
                                           epochs = 100,  
                                           standardize = True,  
                                           stopping_metric = 'MSE',  
                                           loss = 'automatic',  
                                           train_samples_per_iteration = 32,  
                                           shuffle_training_data = True,  
                                           autoencoder = True,  
                                           l1 = 10e-5)  
anomaly_model.train(x=x, training_frame = train_h2o)
```




Activity 11 – Credit Fraud

- Credit Card Fraud with H2O (Autoencoder)

```
tail(creditcard_data,6)
```

##	Time	V1	V2	V3	V4	V5	
## 284802	172785	0.1203164	0.93100513	-0.5460121	-0.7450968	1.13031398	
## 284803	172786	-11.8811179	10.07178497	-9.8347835	-2.0666557	-5.36447278	
## 284804	172787	-0.7327887	-0.05508049	2.0350297	-0.7385886	0.86822940	
## 284805	172788	1.9195650	-0.30125385	-3.2496398	-0.5578281	2.63051512	
## 284806	172788	-0.2404400	0.53048251	0.7025102	0.6897992	-0.37796113	
## 284807	172792	-0.5334125	-0.18973334	0.7033374	-0.5062712	-0.01254568	
##		V6	V7	V8	V9	V10	V11
## 284802		-0.2359732	0.8127221	0.1150929	-0.2040635	-0.6574221	0.6448373
## 284803		-2.6068373	-4.9182154	7.3053340	1.9144283	4.3561704	-1.5931053
## 284804		1.0584153	0.0243297	0.2948687	0.5848000	-0.9759261	-0.1501888
## 284805		3.0312601	-0.2968265	0.7084172	0.4324540	-0.4847818	0.4116137
## 284806		0.6237077	-0.6861800	0.6791455	0.3920867	-0.3991257	-1.9338488
## 284807		-0.6496167	1.5770063	-0.4146504	0.4861795	-0.9154266	-1.0404583
##		V12	V13	V14	V15	V16	
## 284802		0.19091623	-0.5463289	-0.73170658	-0.80803553	0.5996281	
## 284803		2.71194079	-0.6892556	4.62694203	-0.92445871	1.1076406	
## 284804		0.91580191	1.2147558	-0.67514296	1.16493091	-0.7117573	
## 284805		0.06311886	-0.1836987	-0.51060184	1.32928351	0.1407160	
## 284806		-0.96288614	-1.0420817	0.44962444	1.96256312	-0.6085771	
## 284807		-0.03151305	-0.1880929	-0.08431647	0.04133346	-0.3026201	



Exercises:

Increase the number of layers
Increase the number of
neurons per layer

Step 1:

Watch and listen to the instructor's demonstration



15 mins

Step 2:

Work through the activities



30 mins

Individual Activity



Isolation Forest

- A tree-based anomaly detection algorithm
- It is based on modelling the normal data in such a way to isolate anomalies that are both few in number and different in the feature space
 - our proposed method takes advantage of two anomalies' quantitative properties: i) they are the minority consisting of fewer instances and ii) they have attribute-values that are very different from those of normal instances. - Isolation Forest, 2008
- Tree structures are created to isolate anomalies.
- The result is that isolated examples have a relatively short depth in the trees, whereas normal data is less isolated and has a greater depth in the trees



Example - Mutant Fish



Each fisherman first picks a random feature to judge the samples. E.g. Length of the fish, circumference of the fish, or the proportion of its tail fin to its overall length.

Then picks a random value between the known min and max values of the corresponding measurement for the native species and splits all the fish accordingly (all fish with the relevant measurement equal to or bigger than the picked value go right, and every thing else goes left, for example)

The fisherman repeats the entire process over and over again until every single fish has been partitioned and a “tree” of fish has been created.

The resulting trees of the entire group of fishermen represent an isolation forest.



Procedure

- How do we separate each point? The simple procedure is as follows for each point of the data set:
 1. Select the point to isolate.
 2. For each feature, set the range to isolate between the minimum and the maximum.
 3. Choose a feature randomly.
 4. Pick a value that's in the range, again randomly:
 1. If the chosen value keeps the point *above*, switch the *minimum* of the range of the feature to the value.
 2. If the chosen value keeps the point *below*, switch the *maximum* of the range of the feature to the value.
 5. Repeat steps 3 & 4 until the point is isolated. That is, until the point is the only one which is inside the range for all features.
 6. Count how many times you've had to repeat steps 3 & 4. We call this quantity the **isolation number**.

The algorithm claims that a point is an outlier if it doesn't have to repeat the steps 3 & 4 several times.

<https://quantdare.com/isolation-forest-algorithm/>



An illustration



For inliers, the algorithm has to be repeated 15 times. Meanwhile, the outlier's isolation number is 8. **Isolating an outlier means fewer loops than an inlier.**



H2OIsolationForestEstimator

- Builds an Isolation Forest model. Isolation Forest algorithm samples the training frame and in each iteration builds a tree that partitions the space of the sample observations until it isolates each observation. Length of the path from root to a leaf node of the resulting tree is used to calculate the anomaly score. Anomalies are easier to isolate and their average tree path is expected to be shorter than paths of regular observations.

```
seed = 12345
ntrees = 100
isoforest = h2o.estimators.H2OIsolationForestEstimator( ntrees=ntrees, max_depth= 30, seed=seed )
isoforest.train(x=h2o.H2OFrame(x_train).col_names[:], training_frame=h2o.H2OFrame(x_train))
```



Metrics

- Precision-Recall Metrics

- Precision summarizes the fraction of examples assigned the positive class that belong to the positive class.

$$\textit{Precision} = \frac{\textit{TruePositive}}{\textit{TruePositive} + \textit{FalsePositive}}$$

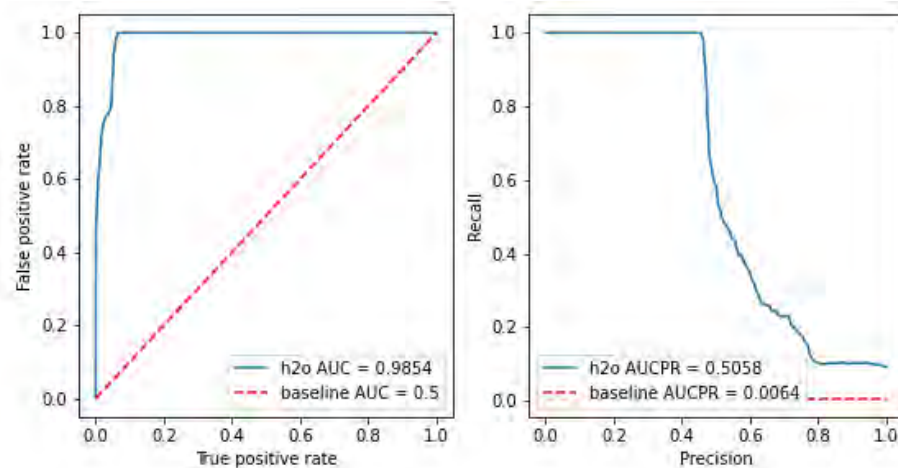
- Recall summarizes how well the positive class was predicted and is the same calculation as sensitivity

$$\textit{Recall} = \frac{\textit{TruePositive}}{\textit{TruePositive} + \textit{FalseNegative}}$$



Metrics

- Area Under the Receiver Operating Characteristic Curve (AUC)
 - a metric evaluating how well a binary classification model distinguishes true positives from false positives. The perfect AUC score is 1; the baseline score of a random guessing is 0.5
- Area under the Precision-Recall Curve (AUCPR)
 - A metric evaluating the precision recall trade-off of a binary classification using different thresholds of the continuous prediction score. The perfect AUCPR score is 1; the baseline score is the relative count of the positive class.





Activity 12 – Intrusion Detection

What is “KDD Cup-1999” data set(s) ?

KDD Cup 1999 : “Computer Network Intrusion Detection” problem.

[intrusion = unauthorized user(s)]

Records : 4,898,431 (around 5 millions) in “train data set” &

311,027 in “test data set”.

Features : 41 (& a class, which consists 23 attributes.)



Exercises:

- Change the no of trees
- Change the max depth

Step 1:

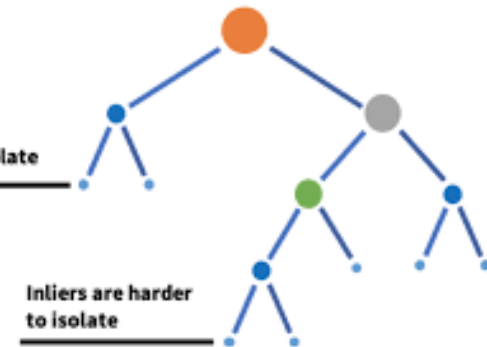
Watch and listen to the instructor’s demonstration



20 mins

Step 2:

- Do on your own



30 mins¹⁷

Individual Activity



Alibi-detect

- an open source Python library for outlier, adversarial and drift detection, that includes a variety of powerful algorithms and techniques.
- It also supports various types of data, such as tabular, time series and image.
- **Outlier Detection Algorithms**
 - [Isolation Forest](#)
 - [Mahalanobis Distance](#)
 - [Autoencoder \(AE\)](#)
 - [Variational Autoencoder \(VAE\)](#)
 - [Autoencoding Gaussian Mixture Model \(AEGMM\)](#)
 - [Variational Autoencoding Gaussian Mixture Model \(VAEGMM\)](#)
 - [Likelihood Ratios](#)
 - [Prophet Time Series Outlier Detector](#)
 - [Spectral Residual](#)
 - [Sequence-to-Sequence \(Seq2Seq\)](#)



Outlier Detection Algorithms

Detector	Tabular	Image	Time Series	Text	Categorical Features	Online	Feature Level
Isolation Forest	✓	X	X	X	✓	X	X
Mahalanobis Distance	✓	X	X	X	✓	✓	X
AE	✓	✓	X	X	X	X	✓
VAE	✓	✓	X	X	X	X	✓
AEGMM	✓	✓	X	X	X	X	X
VAEGMM	✓	✓	X	X	X	X	X
Likelihood Ratios	✓	✓	✓	X	✓	X	✓
Prophet	X	X	✓	X	X	X	X
Spectral Residual	X	X	✓	X	X	✓	✓
Seq2Seq	X	X	✓	X	X	X	✓

Suggested use for each algorithm included in the Alibi Detect library



Threshold Metrics

- F-Measure

- F-measure provides a way to combine both precision and recall into a single measure that captures both properties..

$$F_{measure} = \frac{(2) \times Precision \times Recall}{Precision + Recall}$$

- sometimes called the F-score or the F1-measure
- might be the most common metric used on imbalanced classification problems.



Threshold Metrics

- Consider a binary classification dataset with 1:100 minority to majority ratio, with 100 minority examples and 10,000 majority class examples
- Consider a model that predicts 150 examples for the positive class, 95 are correct (true positives), meaning five were missed (false negatives) and 55 are incorrect (false positives).

$$\text{Precision} = \frac{\text{TruePositive}}{\text{TruePositive} + \text{FalsePositive}}$$

$$= 95 / (95 + 55) = 0.633$$

$$\text{Recall} = \frac{\text{TruePositive}}{\text{TruePositive} + \text{FalseNegative}}$$

$$= 95 / (95 + 5) = 0.95$$



Threshold Metrics

$$\begin{aligned} F_{measure} &= \frac{2 \times Precision \times Recall}{Precision + Recall} \\ &= \frac{2 \times 0.633 + 0.95}{0.633 + 0.95} \\ &= 0.759 \end{aligned}$$

sklearn.metrics.f1 score API.

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html



Getting Started

- Import `alibi_detect`
- Prepare data for training
- Initialise a detector
- Train the detector or load from file (`od.fit()` or `load_detector()`)
- Make predictions (`od.predict()`)

The predictions are returned in a dictionary with as keys `meta` and `data`. `meta` contains the detector's metadata while `data` is in itself a dictionary with the actual predictions. It has either `is_outlier`, `is_adversarial` or `is_drift` (filled with 0's and 1's) as well as optional `instance_score`, `feature_score` or `p_value` as keys with numpy arrays as values



IForest

- Parameters:
 - threshold: threshold value for the outlier score above which the instance is flagged as an outlier.
 - n_estimators: number of base estimators in the ensemble. Defaults to 100.
 - max_samples: number of samples to draw from the training data to train each base estimator. If int, draw max_samples samples. If float, draw max_samples times number of features samples. If 'auto', max_samples = min(256, number of samples).
 - max_features: number of features to draw from the training data to train each base estimator. If int, draw max_features features. If float, draw max_features times number of features features.
 - bootstrap: whether to fit individual trees on random subsets of the training data, sampled with replacement.
 - n_jobs: number of jobs to run in parallel for fit and predict.
 - data_type: can specify data type added to metadata. E.g. 'tabular' or 'image'.

```
from alibi_detect.od import IForest

od = IForest(
    threshold=0.,
    n_estimators=100
)
```



IForest

- We then need to train the outlier detector.
 - X: training batch as a numpy array.
 - sample_weight: array with shape (batch size,) used to assign different weights to each instance during training. Defaults to None.

```
od.fit(  
    X_train  
)
```

- It is often hard to find a good threshold value. If we have a batch of normal and outlier data and we know approximately the percentage of normal data in the batch, we can infer a suitable threshold:

```
od.infer_threshold(  
    X,  
    threshold_perc=95  
)
```



Activity 12 – Network Intrusion

What is “KDD Cup-1999” data set(s) ?

KDD Cup 1999 : “Computer Network Intrusion Detection” problem.

[intrusion = unauthorized user(s)]

Records : 4,898,431 (around 5 millions) in “train data set” &

311,027 in “test data set”.

Features : 41 (& a class, which consists 23 attributes.)



Exercises:

- Change the no base estimators
- Change the max_samples and max_features

Step 1:

Watch and listen to the instructor’s demonstration



20 mins

Step 2:

- Do on your own



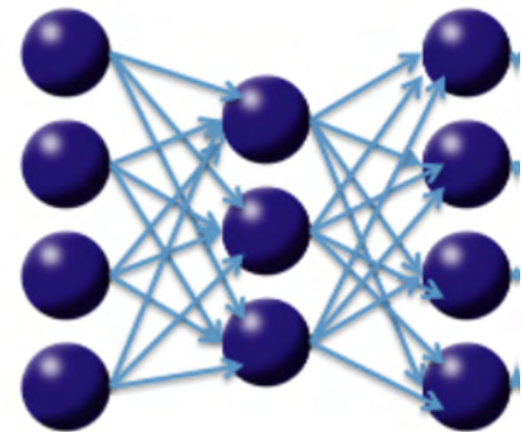
30 mins

Optional Activity



Autoencoders

- Approach AD using a semi-supervised learning
 - Identify a set of data that represents the normal distribution. Non anomaly.
 - Learn what “normal” means from this training dataset. The trained model will provide a sort of metric in its mathematical definition; i.e. a function mapping every point to a real number representing the distance from another point representing the normal distribution.
- By selecting the right threshold, we can achieve the desired trade-off between precision (fewer false alarms) and recall (fewer missed detections)
- Robust to noise
- Uses MSE as the loss function
- Reconstruct based on intermediate representations that minimize the training error. We learn those compression functions from the training set so that a normal pint is very likely to be reconstructed correctly, but **an outlier would have a higher reconstruction error** (the mean squared error between the original point and the reconstructed one)





Usage

- Parameters:

- threshold: threshold value above which the instance is flagged as an outlier.
- encoder_net: tf.keras.Sequential instance containing the encoder network.

Example:

```
encoder_net = tf.keras.Sequential(  
    [  
        InputLayer(input_shape=(32, 32, 3)),  
        Conv2D(64, 4, strides=2, padding='same', activation=tf.nn.relu),  
        Conv2D(128, 4, strides=2, padding='same', activation=tf.nn.relu),  
        Conv2D(512, 4, strides=2, padding='same', activation=tf.nn.relu)  
    ]  
)
```

- decoder_net: tf.keras.Sequential instance containing the decoder network.

Example:

```
decoder_net = tf.keras.Sequential(  
    [  
        InputLayer(input_shape=(1024,)),  
        Dense(4*4*128),  
        Reshape(target_shape=(4, 4, 128)),  
        Conv2DTranspose(256, 4, strides=2, padding='same', activation=tf.nn.relu),  
        Conv2DTranspose(64, 4, strides=2, padding='same', activation=tf.nn.relu),  
        Conv2DTranspose(3, 4, strides=2, padding='same', activation='sigmoid')  
    ]  
)
```

- ae: instead of using a separate encoder and decoder, the AE can also be passed as a tf.keras.Model.
- data_type: can specify data type added to metadata. E.g. 'tabular' or 'image'.



Usage

```
from alibi_detect.od import OutlierAE

od = OutlierAE(threshold=0.1,
               encoder_net=encoder_net,
               decoder_net=decoder_net)
```




Train

- We then need to train the outlier detector. The following parameters can be specified:
 - X: training batch as a numpy array of preferably normal data.
 - loss_fn: loss function used for training. Defaults to the Mean Squared Error loss.
 - optimizer: optimizer used for training. Defaults to Adam with learning rate 1e-3.
 - epochs: number of training epochs.
 - batch_size: batch size used during training.
 - verbose: boolean whether to print training progress.
 - log_metric: additional metrics whose progress will be displayed if verbose equals True.

```
od.fit(X_train, epochs=50)
```



Train

- It is often hard to find a good threshold value. If we have a batch of normal and outlier data and we know approximately the percentage of normal data in the batch, we can infer a suitable threshold:

```
od.infer_threshold(  
    X,  
    threshold_perc=95  
)
```



Detect

- We detect outliers by simply calling predict on a batch of instances X. Detection can be customized via the following parameters:
 - outlier_type: either 'instance' or 'feature'. If the outlier type equals 'instance', the outlier score at the instance level will be used to classify the instance as an outlier or not. If 'feature' is selected, outlier detection happens at the feature level (e.g. by pixel in images).
 - outlier_perc: percentage of the sorted (descending) feature level outlier scores. We might for instance want to flag an image as an outlier if at least 20% of the pixel values are on average above the threshold. In this case, we set outlier_perc to 20. The default value is 100 (using all the features).
 - return_feature_score: boolean whether to return the feature level outlier scores.
 - return_instance_score: boolean whether to return the instance level outlier scores



Detect

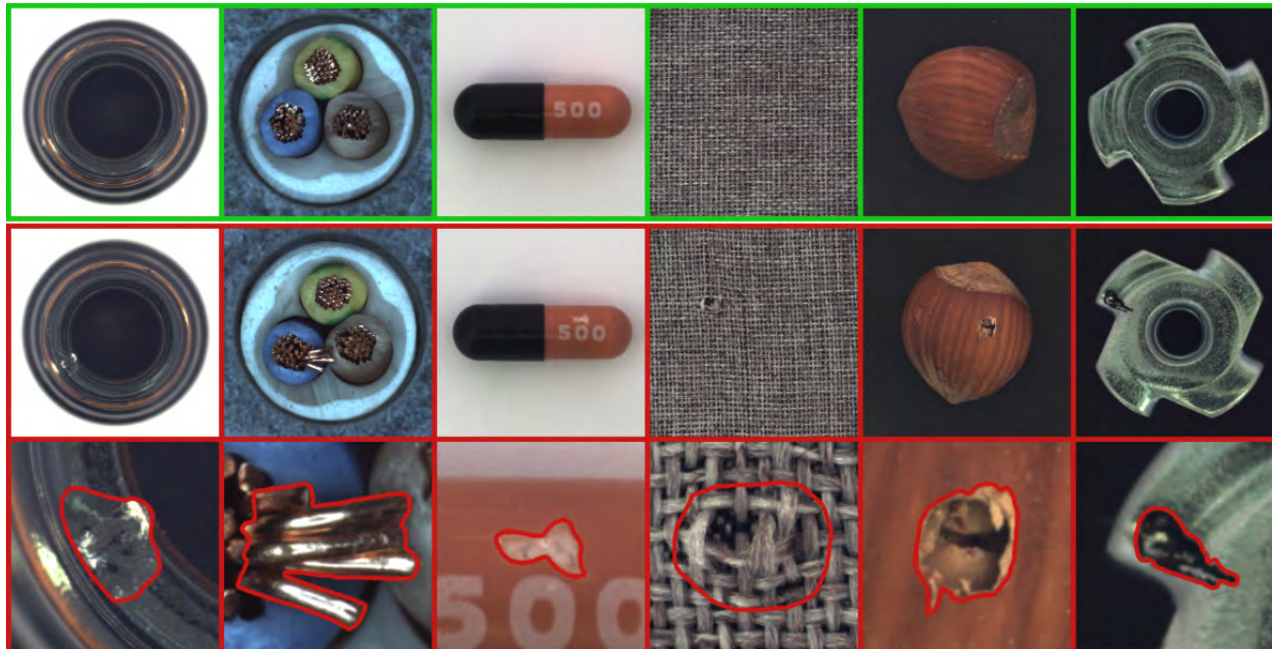
- The prediction takes the form of a dictionary with meta and data keys. meta contains the detector's metadata while data is also a dictionary which contains the actual predictions stored in the following keys:
 - `is_outlier`: boolean whether instances or features are above the threshold and therefore outliers. If `outlier_type` equals 'instance', then the array is of shape (batch size,). If it equals 'feature', then the array is of shape (batch size, instance shape).
 - `feature_score`: contains feature level scores if `return_feature_score` equals True.
 - `instance_score`: contains instance level scores if `return_instance_score` equals True

```
preds = od.predict(X,  
                   outlier_type='instance',  
                   outlier_perc=75,  
                   return_feature_score=True,  
                   return_instance_score=True)
```



The MVTec AD Data Set

- Testing the accuracy of an outlier detection model in real-world conditions can be challenging, as the number and properties of the outliers in a data set, are typically unknown to us.
- We can overcome this obstacle by training and testing our models on a data set that was specifically created for this purpose



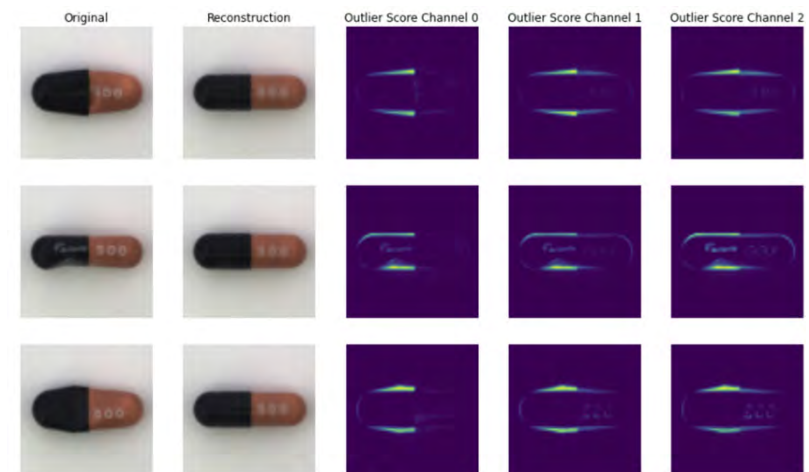
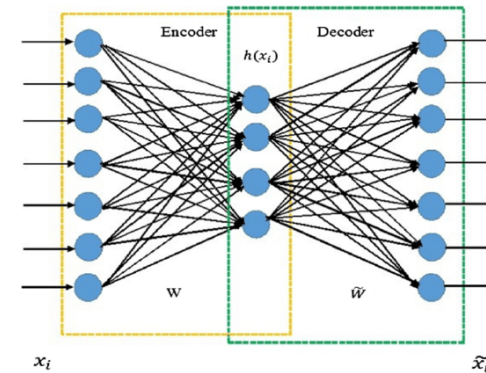
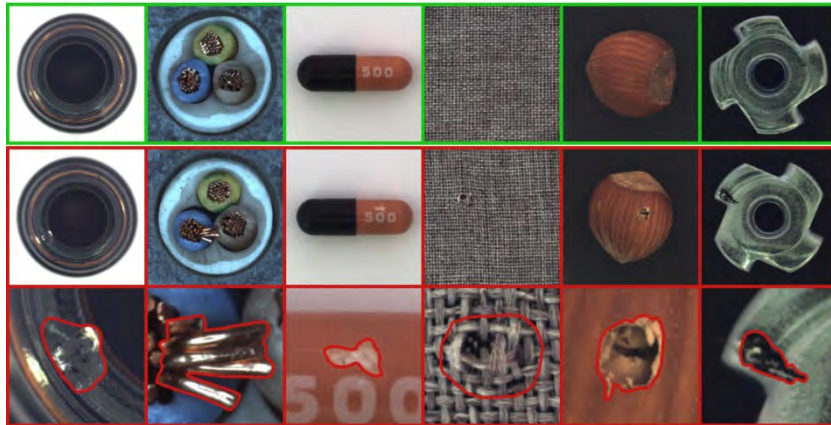


The MVTec AD Data Set

- The [MVTec AD data set](#) contains thousands of high-resolution images, and is suitable for testing and **benchmarking image outlier methods**, with a focus on quality control in industrial manufacturing.
- The data set is comprised of 15 categories of images, such as carpet, leather, transistor, screw etc.
- The training set for each category includes normal images only, while the test set has both normal images and outliers with various defects.
- For additional information about the dataset and the benchmarking results for various outlier detection algorithms, you can refer to the associated research paper



Activity 13 – MVTec



Exercises:

- Change the decoder and decoder design

Step 1:

Watch and listen to the instructor's demonstration



20 mins

Step 2:

- Do on your own



30 mins

Individual Activity

OFFICIAL (CLOSED) \ NON-SENSITIVE



Quiz

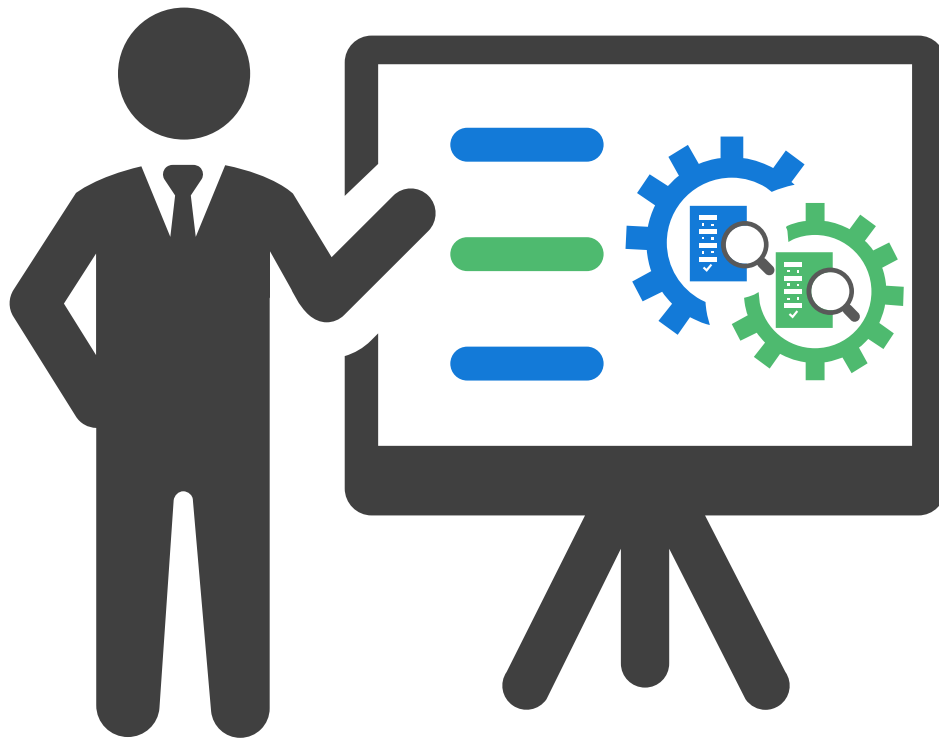
https://bit.ly/kw_poll



SCAN ME



Summary



Email
seow_khee_wei@rp.edu.sg

Telegram
@kwseow

Source code: