Anomaly Detection!

Reference: https://towardsdatascience.com/anomaly-detection-in-manufacturing-part-1-an-introduction-8c29f70fc68b

Def,: "an [anomaly] is an observation which deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism"

In particular, we'll learn to detect anomalies, during metal machining, using a **variational autoencoder (VAE).**

Dataset:

The metal machining data set, or milling data set, we'll be using is from UC Berkeley.
In the UC Berkeley milling data set the flank wear (*VB*) is measured from cut to cut. This *VB* value will be used for labeling purposes.

Data set is contained in a structured MATLAB array.

Data Prep: Each cut has an associated amount of flank wear, *VB*, measured at the end of the cut. We'll label each cut as either healthy, degraded, or failed according to the amount of wear on the tool — these are the tool health categories.

Metadata:

The dataset comprises 16 cases of milling tools conducting cuts in metal, with six cutting parameters: metal type (cast iron or steel), depth of cut (0.75 mm or 1.5 mm), and feed rate (0.25 mm/rev or 0.5 mm/rev). Each case represents a unique combination of these parameters. The dataset includes 167 cuts across all cases, ranging from new to degraded or worn tool conditions. Flank wear (VB) measurements are available for many cuts, allowing for later categorization of cuts as healthy, degraded, or worn based on VB values.

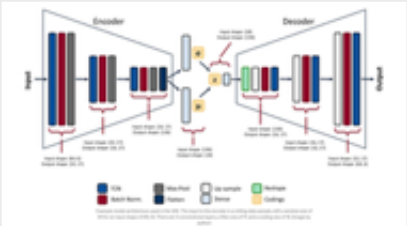
The signals were collected at 250 Hz and each cut had 9000 sampling points, for a total signal length of 36 seconds.

During each cut, six signals were collected, including acoustic emission (AE) signals from the spindle and table, vibration data from the spindle and table, and AC/DC current readings from the spindle motor.

Architecture:

Going forward, we will use a variant of the auto encoder — a variational auto encoder (VAE) — to conduct anomaly detection on the milling data set.

VAE Model:



Time to begin training some models. To select the hyper parameters, we'll be using a random search.
We'll be training a bunch of different VAEs, all with different parameters. After each VAE has been trained (trained to minimize reconstruction loss), and the model saved, we'll go through the VAE model and see how it performs in anomaly detection

Trained about 1000 VAE models

Random Search Process:

Methodology:

The anomaly detection is performed using both the reconstruction error (input space anomaly detection) and measuring the difference in KL-divergence between samples (latent space anomaly detection).

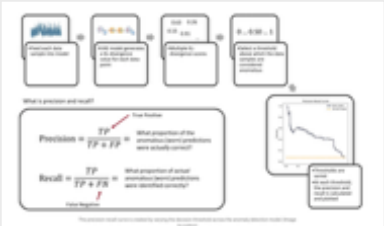
Anomaly detection can also be performed using the mean and standard deviation codings in the latent space, which is what we'll be doing. Here is the general method:
Using KL-divergence, measure the relative difference in entropy between data samples. A threshold can be set on this relative difference indicating when a data sample is anomalous.
The KL-divergence function (implemented with Keras and TensorFlow) that we will be using:

```
def kl_divergence(mu, log_var):
return -0.5 * K.sum(1 + log_var - K.exp(log_var) - K.square(mu), axis=-1,)
```

where mu is the mean (μ) and the log_var is the logarithm of the variance ($\log \sigma^2$). The log of the variance is used for the training of the VAE as it is more stable than just the variance.

Evaluation Metrics:

We'll be using the precision-recall area-under-curve (PR-AUC) to evaluate model performance. PR-AUC performs well on imbalanced data, in contrast to the ROC-AUC.



Ultimately, the evaluation of a model's performance and the setting of its decision threshold is application specific. For example, a manufacturer may prioritize the prevention of tool failures over frequent tool changes. Thus, they may set a low threshold to detect more tool failures (higher recall), but at the cost of having more false-positives (lower precision).

Future Work:

- Ensemble of models improves results significantly
- Adjusting beta in VAE transforms it into a disentangled-variational-autoencoder
- Exploring how codings change with different cutting parameters could reveal unique features
- Consideration of using a regular convolutional neural network with dilations for simplified model training
- Integration of more model tests (similar to unit tests) recommended for assessing performance across different cutting parameters
- Aim is to identify models that generalize well in various scenarios

Code Implementation:

1. Exploration of the UC Berkeley Milling Data Set

1_A_milling_data_exploration.ipynb
251 KB

1_B_building_vae.ipynb
47 KB

1_C_anomaly_results.ipynb
589 KB

Notes:

- The power of self-supervised learning makes it attractive for use in manufacturing and industrial environments where much of the data is not properly labeled, and/or it would be too costly to label. The use of an autoencoder for anomaly detection is one such instantiation of self-supervised learning.
- **VAE** is different from traditional auto encoders in that the VAE is both probabilistic and generative. What does that mean? The VAE creates outputs that are partly random (even after training) and can also generate new data that is like the data it is trained on.
- The final distribution of the data is shown below. Notice how imbalanced the data is (that is, relatively few “failed” samples)? This is a common problem in manufacturing/industrial data, which is another reason to use a self-supervised method.
- For anomaly detection, it is common to train the autoencoders on “normal” data only. We’ll be doing the same and training our VAE on healthy data (class 0). However, checking the performance of the anomaly detection will be completed using all the data. In other words, we’ll be training our VAE on the “slim” data sets, but testing on the “full” data sets.
- I’ve used rounded accuracy to measure how the model performs during training.
- The latent space serves as a distilled and meaningful representation of the input data, obtained through the learning process of a neural network, while the input space encompasses the raw, often high-dimensional, data. The use of the latent space is particularly advantageous for tasks requiring abstraction, dimensionality reduction, and meaningful feature representation. KL-divergence is used.