# ManufacturingNet: A Machine Learning Toolbox for Engineers

**Rishikesh Magar[†], Lalit Ghule[†], Ruchit Doshi [†], Aman Khalid[§], Sharan Seshadri[‡], Amir Barati Farimani[†]**

[†]Department of Mechanical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213 USA
[‡]Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA 15213 USA
[§]College of Literature, Science, and the Arts, University of Michigan, Ann Arbor, MI 48109 USA

## Abstract

The growing deployability of artificial intelligence (AI), accessibility to large amounts of data and new computing technologies are causing a disruption in the manufacturing industry. Manufacturers are turning to technologies like AI, robotics, and the Internet of Things (IoT) to convert their production plants into more efficient smart factories. Many large-sized manufacturers have already deployed some of these technologies in their factories, with small-sized and medium-sized industries quickly catching up. As the manufacturing industry embraces AI, demand among engineering professionals for user-friendly tools that can deploy complex machine learning models with relative ease has been growing over the years. In particular, deep learning tools need a considerable amount of programming knowledge and, thus, remain obscure to engineers inexperienced with programming. To overcome these barriers, we propose ManufacturingNet, an open-source machine learning tool that enables engineers to develop complex machine learning models with minimal programming and data science experience. We have also curated nine publicly-available datasets and benchmarked their performance using ManufacturingNet‘s machine learning models. For each dataset, pre-trained models yielding optimal results are also included. We believe ManufacturingNet will enable engineers around the world to develop machine learning models with ease, contributing towards the larger movement of the 4th industrial revolution.
The GitHub repository for ManufacturingNet can be found at https://github.com/BaratiLab/ManufacturingNet.

## 1 Introduction

The 4th industrial revolution has led to manufacturers embracing technological advancements in the areas of robotics, AI, nanotechnology, the Internet of Things (IoT), the Industrial Internet of Things (IIoT). These digital technologies are not only improving efficiency in manufacturing, but also providing manufacturers an edge to keep up with the rapidly changing world driven by customer demands. Companies across the globe are using AI to improve their supply chains and increase automation level. This global push for improved efficiency and automation by manufacturing plants across the world has led to an increased presence of advanced sensor technologies that are capable of capturing real-time data, which can be effectively used to analyze and optimize processes[1]. To analyze this vast amount of data being collected, cost-effective smart technologies like AI are being used by manufacturers regularly. This use of smart technologies has contributed significantly towards process improvements, preventive maintenance, and quality control in many industries[2]. Moreover, smart technologies can also lead to improvements in energy management, automation of complex processes, and help organizations make data-driven decisions. With the increased availability of data and impetus from the industry for data-driven research, researchers are increasingly using advanced machine learning techniques to address the complex problems faced by manufacturers across the globe. One such example where machine learning is actively used is the semiconductor manufacturing industry. A common problem faced by semiconductor manufacturers is the inability to detect faulty wafers. This inability to detect faulty wafers can lead to lowered process yield and increased downtime in manufacturing. In their paper, Kim et al., showed that machine learning techniques can act as a promising tool for wafer fault detection [3]. Another area where researchers have applied machine learning to manufacturing data is for predicting tool wear and potential failure during the machining process[2]. Problems such as reducing energy costs with lowered consumption, quality control by predicting surface roughness, deformation, and cutting force also have been addressed using different machine learning methods by researchers [4],[5]. With this growing trend in usage of machine learning in the manufacturing industry, there is a growing need for engineers in manufacturing plants to use available data and generate valuable insights that can have broader impact on their organizations' profits. However, with their expertise lying in production processes, engineers working at these manufacturing facilities find it difficult to develop complex machine learning models, limiting their ability to analyze their own datasets. To tap into manufacturing data and provide an accessible tool for engineers, we have developed ManufacturingNet, an open-source package. ManufacturingNet can be used to analyze different types of problems with very little coding expertise required. We also provide the benchmark performance for nine different datasets that can be subdivided into five broad categories in manufacturing, along with their pre-trained models that can be used by engineers directly on similar datasets. We believe that such a tool is of utmost importance for engineers, and will contribute significantly towards the data-driven research of manufacturers across the globe. We also would like to acknowledge that our package is built on top of other packages like NumPy[6], Scipy[7], Matplotlib[8], PyTorch[9], Pillow[10] and Scikit-Learn[11]

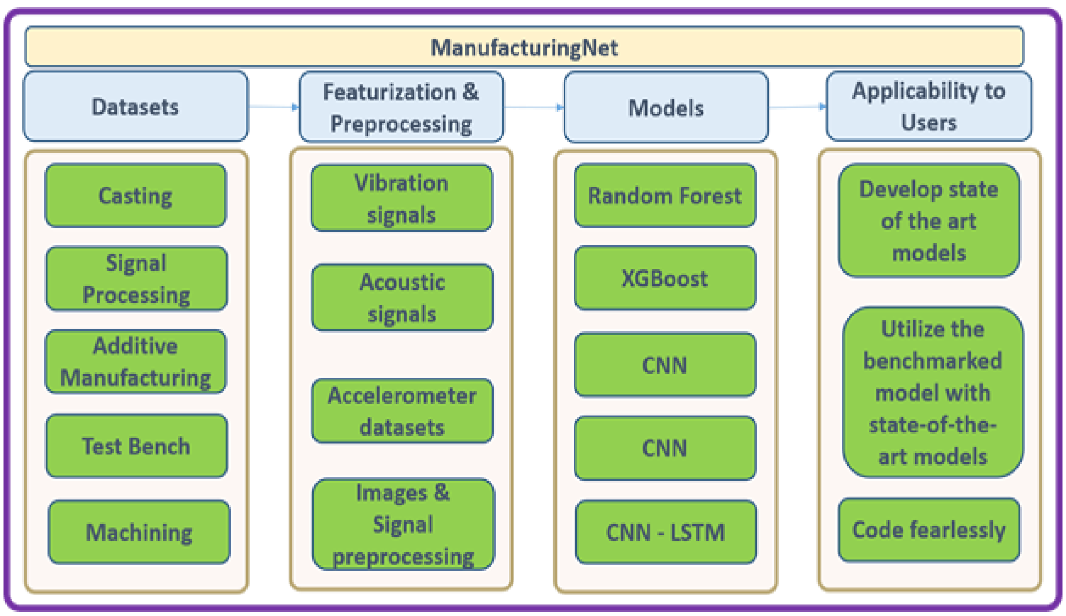


Figure 1: An overview of the ManufacturingNet project.

## 2 Methods

In this work, we introduce the open-source package ManufacturingNet. Our package provides the functionality to run complex machine learning models, enabling users with varying degrees of expertise to develop these models with greater ease. The deep learning frameworks available in the package include Multi-Layered Perceptron (MLP), Convolutional Neural Network (CNN), Long Short-Term Memory network (LSTM). To provide functionality for video-based data, we have also included CNN3D and CNN-LSTM models. We also offer users conventional machine learning models such as Random Forest[12], Support Vector Machines (SVM)[13], XGBoost[14], logistic regression, and linear regression. We would like to highlight ManufacturingNet's ability to run all supported classification/regression models simultaneously. This functionality will allow users to compare the performance of all algorithms and select the best one for their task. Figure 1 demonstrates the broad layout of the ManufacturingNet package. Aside from its functionality of running complex models, ManufacturingNet also has benchmark models for nine publicly-available datasets. These datasets and their corresponding best-performing models can be easily downloaded using the package. In general, the user has to answer a few simple questions in regard to the model they want to develop, and our package creates a model based on their inputs. To further assist the user, we have also written documentation to explain the parameters of each model. This documentation is available at https://manufacturingnet.readthedocs.io. Moreover, we also provide tutorials in our GitHub repository that demonstrate how these models can be run.

### 2.1 Datasets

One of the core goals of ManufacturingNet is to aggregate multiple public datasets from different realms of industrial manufacturing and provide high-performance benchmark models. ManufacturingNet has nine datasets subdivided into five broad categories, namely casting, signal processing, additive manufacturing, test bench and machining. The evaluation metrics provided when developing models for that data are summarized in Table 3 and our benchmark results have been discussed in later sections. We also plan to incorporate more datasets as the project progresses, and welcome public contributions from users working in this research area. More details about the different data structure requirements and the types of models suitable for the datasets are available in the ManufacturingNet documentation.

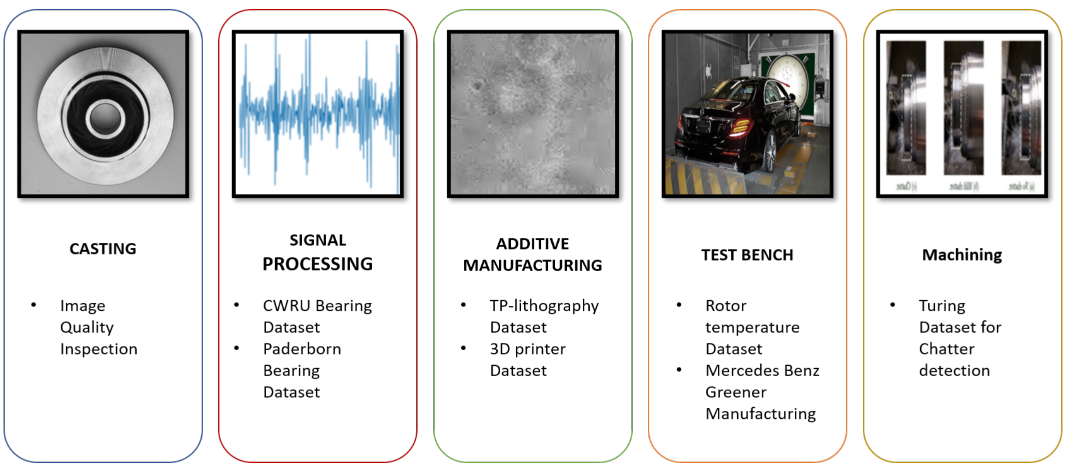


Figure 2: Datasets available in ManufacturingNet.

#### 2.1.1 List of available datasets

1. CWRU bearing dataset[15]
2. Paderborn bearing dataset[16]
3. Two photon lithography dataset[17]
4. Mercedes Benz greener manufacturing dataset[18]
5. Casting images dataset[19]
6. 3D prnting dataset[20]
7. Paderborn university motor temperature dataset[21]
8. Gearbox binary classification dataset[22]
9. Chatter detection dataset[23][24]

### 2.2 Models

ManufacturingNet provides implementations of several machine learning models that can be applied to different types of problems and datasets. The included models are broadly classified into two categories: conventional machine learning methods and deep learning methods. For conventional machine learning, algorithms like linear regression, logistic regression, SVM, Random Forest and XGBoost are implemented. For deep learning, fully-connected neural networks (or Multi-Layer Perceptron), Convolutional Neural Networks (CNN), Convolutional Neural Networks for video classification (CNN-3D)[25] and CNN-LSTM have been implemented. We also implement Long Short-Term Memory Networks (LSTM)[26] for time series data related predictions.

#### 2.2.1 Pre-trained models

ManufacturingNet also offers some of the well established neural network architectures with proven track records. Currently, there are six architectures included in the package: ResNet[27], AlexNet[28], VGG[29], DenseNet[30], MobileNet[31], and GoogleNet[32] . Users may choose to use the pre-trained weights of these models. These models are trained on ImageNet datasets[33], and can be used for different types of tasks such as classification, object detection, and video classification, among others. Since users' datasets may contain different numbers of input channels and classes (for classification tasks), ManufacturingNet iterates these layers in the network of the selected model. ResNet, VGG, and DenseNet models further contain different types of sub-architectures depending upon the number of layers in the model. For example, ResNet contains 18, 34, 51, and 101 architectures; ResNet18 here is an 18-layer deep network. Depending on the complexity of the task and available RAM, users can select models with the desired depth of the network. The commands for running ManufacturingNet's models are shown in Table 1.

Table 1: Running the models

| Model | Command |
|---|---|
| Linear Regression | `ManufacturingNet.models.LinRegression` |
| Logistic Regression | `ManufacturingNet.models.LogRegression` |
| SVM | `ManufacturingNet.models.SVM` |
| Random Forest | `ManufacturingNet.models.RandomForest` |
| All Classification models | `ManufacturingNet.models.AllClassificationModels` |
| All Regression models | `ManufacturingNet.models.AllRegressionModels` |
| Deep neural network | `ManufacturingNet.models.DNN` |
| CNN2D on Signals | `ManufacturingNet.models.CNN2DSignal` |
| CNN2D on Images | `ManufacturingNet.models.CNN2DImage` |
| CNN3D | `ManufacturingNet.models.CNN3D` |
| CNN LSTM | `ManufacturingNet.models.CNNLSTM` |
| LSTM | `ManufacturingNet.models.LSTM` |
| AlexNet | `ManufacturingNet.models.AlexNet` |
| VGG Models | `ManufacturingNet.models.VGG` |
| ResNet Models | `ManufacturingNet.models.ResNet` |
| DenseNet Models | `ManufacturingNet.models.DenseNet` |
| GoogleNet | `ManufacturingNet.models.GoogleNet` |
| MobileNet | `ManufacturingNet.models.MobileNet` |

### 2.3 Demonstration: Running models with ManufacturingNet

ManufacturingNet provides a consistent experience for using conventional machine learning, deep learning, and pre-trained models. Regardless of the model, users follow the same basic process of importing required ManufacturingNet modules, preparing their data, and training the model. Within a Python 3 environment,

- Import the desired model from the ManufacturingNet library. All models are contained in the "models" sub-directory.
  - Users may also wish to import other modules to assist with data preparation. For example, in Figures 3 and 4, the NumPy library is used to load the data for training.

- Prepare the data.
  - Most model implementations expect the data's features and labels to be entered as separate NumPy arrays. Notable exceptions are models expecting non-ASCII data, such as image classification models like CNN2D. In these cases, the model requires strings of the file paths to the training and testing sets.
  - Note: ManufacturingNet's datasets module provides functions for downloading and extracting each dataset. Datasets containing ASCII data are provided in a format readable by the NumPy library's load() function.
- Instantiate and train the model.
  - Each model provides an initialization method requiring two parameters: the dataset's features and labels.
  - Because many of ManufacturingNet's conventional machine learning model implementations support both classification and regression tasks, some models require the user to call the run_classifier() or run_regressor() method after initialization to specify the task. Conventional models which support only one task (such as linear regression) simply require the user to call the run() method.
- Specify the parameters.
  - ManufacturingNet's command-line interface simplifies parameter entry, and offers default values when the user is unsure.
  - Once the model finishes training, relevant metrics are displayed via the interface.
- (Optional) Run the model on new data.
  - ManufacturingNet's model implementations offer methods for classifying or making predictions on new data using the trained model.

To learn more, users may view model tutorials provided in the repository, or read the documentation on ManufacturingNet's website.

```python
# Import necessary modules
from ManufacturingNet import datasets
from ManufacturingNet.models import AllRegressionModels
import numpy as np

# Get Mercedes-Benz dataset
datasets.MercedesData()
features = np.load('./Mercedes_files/merc_features.npy',
                   allow_pickle = True)
labels = np.load('./Mercedes_files/merc_labels.npy',
                 allow_pickle = True)

# Run all regression models
all_models = AllRegressionModels(features, labels)
all_models.run()
```

```
=========================================
= All Regression Models Parameter Inputs =
=========================================


Enable verbose logging (y/N)?  n
verbose = False

What fraction of the dataset should be used for testing (0,1)?  0.25
test_size = 0.25

==========================================
= End of inputs; press enter to continue. =
==========================================



===========
= Results =
===========

Model               R2 Score             Time (seconds)

LinearRegression    0.46154528075069656  0.14429569244384766

RandomForest        0.43743172247471984  9.198936223983765

SVR                 -0.02326227461358421 5.484614372253418

NuSVR               -0.005759599887122491 3.878767490386963

LinearSVR           0.3227457000350572   1.283017873764038

XGBRegressor        0.42116630258136745  2.4771761894226074
```

Figure 3: Running all machine learning regression models with ManufacturingNet.

```python
from ManufacturingNet import datasets
from ManufacturingNet.models import DNN
import numpy as np

datasets.CWRUBearingData()
X = np.load('./CWRU/featurized_data.npy',
            allow_pickle = True)
Y = np.load('./CWRU/featurized_data_labels.npy',
            allow_pickle = True)

model = DNN(X, Y)
```

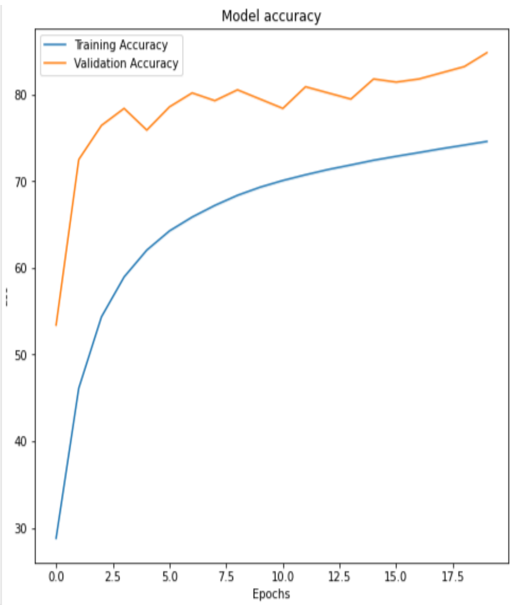


Figure 4: Training a DNN model with ManufacturingNet, and viewing its accuracy graph.

```python
from ManufacturingNet import datasets
from ManufacturingNet.models import AlexNet
import numpy as np

datasets.CastingData()
train_data_address = 'casting_data/train/'
val_data_address = 'casting_data/test/'

model = AlexNet(train_data_address, val_data_address)
```

Figure 5: Training an AlexNet image classification model with ManufacturingNet.

### 2.4 Featurization and Preprocessing modules

Feature engineering is an essential step in developing any machine learning model. If a feature from the dataset selected by the user can represent the relationship between the data and the target prediction, it is expected that the performance of the machine learning algorithm will be higher. One of the commonly available signals data to the manufacturing community is vibration signals. Therefore, users need a comprehensive list of signal features to select and deploy machine learning models from. In ManufacturingNet, we provide the user with 20 signal features, including RMS value, mean, median, shape factor, and crest factor. These signal features were chosen by studying important literature in this area. The commands in ManufacturingNet to implement different featurization techniques are demonstrated in Table 2.

First, the user needs to initialize the featurizer object using the below command:
Featurizer = `ManufacturingNet.featurization.Featurizer`

Suppose F = Featurizer()

Table 2: Featurization using ManufacturingNet

| Featurization | Command |
|---|---|
| Mean | `F.mean()` |
| Median | `F.median()` |
| Min | `F.min()` |
| Max | `F.max()` |
| Peak to Peak | `F.peak_to_peak()` |
| Variance | `F.variance()` |
| RMS | `F.rms()` |
| Absolute mean | `F.abs_mean()` |
| Shape Factor | `F.shapefactor()` |
| Impulse Factor | `F.impulsefactor()` |
| Crest Factor | `F.crestfactor()` |
| Clearance Factor | `F.clearancefactor()` |
| Standard Deviation | `F.std()` |
| Skewness | `F.skew()` |
| Kurtosis | `F.kurtosis()` |
| Absolute log mean | `F.abslogmean()` |
| Mean absolute deviation | `F.meanabsdev()` |
| Median absolute deviation | `F.medianabsdev()` |
| Mid-range | `F.midrange()` |
| Coefficient of Variation | `F.coeff_var()` |

## 3 Results

The performance benchmarks of ManufacturingNet's nine publicly-available datasets are provided in Table 3. The features provided in ManufacturingNet allow us to deal with many different types of data, including vibration signals, images, and time series data. Depending on the type of data, we selected relevant features for each model, and used both conventional machine learning algorithms and deep learning methods to generate predictions. All results reported in this paper are with five-fold cross validation, and we achieved state-of-the-art results for almost all the datasets.

Table 3: Results on the ManufacturingNet datasets

| Dataset | Dataset-type | Metric-used | Task | Model | Results |
|---|---|---|---|---|---|
| CWRU bearing | `Signal Processing` | `Accuracy` | `Classification` | CNN | 98.5% |
| Paderborn bearing | `Signal Processing` | `Accuracy` | `Classification` | CNN | 99.13% |
| Gearbox binary classification dataset | `Signal Processing` | `Accuracy` | `Classification` | Random Forest | 99.50% |
| Lithography dataset | `AM` | `Accuracy` | `Classification` | CNN-LSTM | 99% |
| 3D printer dataset | `AM` | `Accuracy` | `Classification` | Logistic Reg. | 100% |
| Rotor Temperature | `Test Bench` | `R2-score` | `Regression` | LSTM | 0.9943 |
| Mercedes-Benz Green dataset | `Test Bench` | `R2-score` | `Regression` | LSTM | 0.553 |
| Chatter dataset | `Machining` | `Accuracy` | `Classification` | CNN | 82.22% |
| Casting images | `Casting` | `Accuracy` | `Classification` | DenseNet | 99.28% |

Note: "AM" refers to additive manufacturing in Table 3

## References

[1] Jay Lee, Edzel Lapira, Behrad Bagheri, and Hung-an Kao. Recent advances and trends in predictive manufacturing systems in big data environment. *Manufacturing Letters*, 1(1):38 – 41, 2013.

[2] Kim T.J.Y. Wang X. et al. Kim, D. Smart Machining Process Using Machine Learning: A Review and Perspective on Machining Industry. *International Journal of Precision Engineering and Manufacturing-Green Technology*, pages 555–568, August 2018.

[3] E. Kim, S. Cho, B. Lee, and M. Cho. Fault detection and diagnosis using self-attentive convolutional neural networks for variable-length sensor data in semiconductor manufacturing. *IEEE Transactions on Semiconductor Manufacturing*, 32(3):302–309, 2019.

[4] T. Benkedjouh, K. Medjaher, N. Zerhouni, and S. Rechak. Health assessment and life prediction of cutting tools based on support vector regression. *Journal of Intelligent Manufacturing*, 26(2):213–223, April 2015.

[5] Thorsten Wuest, Christopher Irgens, and Klaus-Dieter Thoben. An approach to monitoring quality in manufacturing using supervised machine learning on product state data. *Journal of Intelligent Manufacturing*, 25(5):1167–1180, October 2014.

[6] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[7] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C. J. Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, Aditya Vijaykumar, Alessandro Pietro Bardelli, Alex Rothberg, Andreas Hilboll, Andreas Kloeckner, Anthony Scopatz, Antony Lee, Ariel Rokem, C. Nathan Woods, Chad Fulton, Charles Masson, Christian Häggström, Clark Fitzgerald, David A. Nicholson, David R. Hagen, Dmitrii V. Pasechnik, Emanuele Olivetti, Eric Martin, Eric Wieser, Fabrice Silva, Felix Lenders, Florian Wilhelm, G. Young, Gavin A. Price, Gert-Ludwig Ingold, Gregory E. Allen, Gregory R. Lee, Hervé Audren, Irvin Probst, Jörg P. Dietrich, Jacob Silterra, James T. Webber, Janko Slavič, Joel Nothman, Johannes Buchner, Johannes Kulick, Johannes L. Schönberger, José Vinícius de Miranda Cardoso, Joscha Reimer, Joseph Harrington, Juan Luis Cano Rodríguez, Juan Nunez-Iglesias, Justin Kuczynski, Kevin Tritz, Martin Thoma, Matthew Newville, Matthias Kümmerer, Maximilian Bolingbroke, Michael Tartre, Mikhail Pak, Nathaniel J. Smith, Nikolai Nowaczyk, Nikolay Shebanov, Oleksandr Pavlyk, Per A. Brodtkorb, Perry Lee, Robert T. McGibbon, Roman Feldbauer, Sam Lewis, Sam Tygier, Scott Sievert, Sebastiano Vigna, Stefan Peterson, Surhud More, Tadeusz Pudlik, Takuya Oshima, Thomas J. Pingel, Thomas P. Robitaille, Thomas Spura, Thouis R. Jones, Tim Cera, Tim Leslie, Tiziano Zito, Tom Krauss, Utkarsh Upadhyay, Yaroslav O. Halchenko, Yoshiki Vázquez-Baeza, and SciPy 1.0 Contributors. SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature Methods*, 17(3):261–272, March 2020.

[8] J. D. Hunter. Matplotlib: A 2D Graphics Environment. *Computing in Science Engineering*, 9(3):90–95, May 2007.

[9] Paszke, Adam et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Advances in Neural Information Processing Systems 32*, pages 8024–8035, 2019.

[10] Clark, A. Pillow (PIL Fork) Documentation, readthedocs., 2015.

[11] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12(85):2825–2830, 2011.

[12] Leo Breiman. Random Forests. *Machine Language*, 45(1):5–32, October 2001. Number: 1 Reporter: Machine Language.

[13] Marti A. Hearst. Support Vector Machines. *IEEE Intelligent Systems*, 13(4):18–28, July 1998.

[14] Tianqi Chen and Carlos Guestrin. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, San Francisco, California, USA, August 2016. Association for Computing Machinery.

[15] http://csegroups.case.edu/bearingdatacenter/home.

[16] Christian Lessmeier, James Kuria Kimotho, Detmar Zimmer, Walter Sextro. Condition Monitoring of Bearing Damage in Electromechanical Drive Systems by Using Motor Current Signals of Electric Motors: A Benchmark Data Set for Data-Driven Classification. *Conference: European Conference of the Prognostics and Health Management Society*, July 2016.

[17] Sourabh K.; Sarkar Soumik; Giera Brian Lee, Xian Yeow; Saha. Labelled Two-photon Lithography Dataset (LLNL-MI-808019). *Mendeley Data*, V1, 2020.

[18] Daimler (Mercedes-Benz group). Mercedes-Benz Greener Manufacturing, 2017.

[19] PILOT TECHNOCAST, Shapar, Rajkot. Casting product image data for quality inspection, 2020.

[20] TR/Selcuk University Mechanical Engineering department, Turkey. 3D Printer Dataset for Mechanical Engineers, 2018.

[21] German Research Foundation (DFG). Temperature Estimation of Vital Components in Electric Motors Using Machine Learning, 2018.

[22] OpenEI. Gearbox Fault Diagnosis Data, 2018.

[23] Melih C. Yesilli, Firas A. Khasawneh, and Andreas Otto. On transfer learning for chatter detection in turning using wavelet packet transform and ensemble empirical mode decomposition. *CIRP Journal of Manufacturing Science and Technology*, 28:118–135, Jan 2020.

[24] Melih C. Yesilli, Firas A. Khasawneh, and Andreas Otto. Topological feature vectors for chatter detection in turning processes, 2019.

[25] S. Ji, W. Xu, M. Yang, and K. Yu. 3D Convolutional Neural Networks for Human Action Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):221–231, January 2013.

[26] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Comput.*, 9(8):1735–1780, November 1997. Place: Cambridge, MA, USA Publisher: MIT Press.

[27] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, June 2016. ISSN: 1063-6919.

[28] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[29] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2014. _eprint: 1409.1556.

[30] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely Connected Convolutional Networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269, July 2017. ISSN: 1063-6919.

[31] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. 2017. _eprint: 1704.04861.

[32] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, June 2015. ISSN: 1063-6919.

[33] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.