Aalto-yliopisto
Aalto-universitetet
Aalto University

Master's Programme in Computer, Communication and Information Sciences

Deep Learning for Virtual Metrology of Chemical-Mechanical Polishing

Eero Hiltunen

Master's Thesis
Espoo, September 20, 2021

Supervisor:

Prof. Alexander Ilin

Advisors:

D.Sc. Jukka-Pekka Salmenkaita
Ph.D. Antti Liski

Aalto-yliopisto
Aalto-universitetet
Aalto University

Author Eero Hiltunen

Title of thesis Deep Learning for Virtual Metrology of Chemical-Mechanical Polishing

Programme Computer, Communication and Information Sciences

Major Computer Science

Thesis supervisor Prof. Alexander Ilin

Thesis advisors D.Sc. Jukka-Pekka Salmenkaita, Ph.D. Antti Liski

Collaborative partner Elisa Oyj

Date 20.09.2021 **Number of pages** 60 + 6 **Language** English

Abstract

The demand for semiconductor components in consumer products has increased rapidly in the 2020s which has led to a shortage of electrical components and thus breaks in availability of the products. For this reason, semiconductor manufacturers are motivated to deploy machine learning and automation solutions, such as virtual metrology, that increase the yield and quality of manufacturing. Regardless of the clear benefits, there is an absence of comprehensive comparisons of current state-of-the-art machine learning methods for virtual metrology.

This thesis presents and reviews multiple state-of-the-art machine learning methods for virtual metrology of chemical-mechanical polishing – a crucial process in semiconductor manufacturing. We compare typical virtual metrology pipelines consisting of hand-crafted features and tree ensemble models to the current state-of-the-art time series extrinsic regression models, such as InceptionTime and recurrent neural networks. In addition, we propose and evaluate several approaches for including time-invariant extrinsic process variables to recurrent neural networks. Furthermore, we implement semi-supervised autoencoder models for a prediction scenario where only a fraction of process runs are labeled. These autoencoders are compared to other machine learning methods in a limited labeled data setting. In addition, we evaluate the performance of tree ensemble models by analyzing their feature importance scores. The experimental work is conducted on a public dataset which allows simple comparison of our work to other recent publications utilizing the same dataset.

Our experiments and comparison of different models show that hand-crafted features combined with tree ensemble models are a strong choice for the virtual metrology of chemical-mechanical polishing. We discovered, however, that semi-supervised autoencoder models predict metrology results more accurately than supervised machine learning methods in a limited labeled data setting. Furthermore, our tree ensemble model feature importance analysis revealed that only a fraction of the measurements in the process data are sufficient for accurate virtual metrology of chemical-mechanical polishing tool. In fact, a gradient boosted trees model utilized only 19% of the measurements and achieved an R^2 score of 0.99 in the testing set.

Keywords machine learning, semi-supervised learning, virtual metrology

Aalto-yliopisto
Aalto-universitetet
Aalto University

Tekijä Eero Hiltunen

Työn nimi Syväoppiminen kemiallismekaanisen hiomisen virtuaalisessa metrologiassa

Koulutusohjelma Tietokone, kommunikaatio ja informaatiotieteet

Pääaine Tietotekniikka

Valvoja Prof. Alexander Ilin

Työn ohjaajat TkT Jukka-Pekka Salmenkaita, FT Antti Liski

Yhteistyötaho Elisa Oyj

Päivämäärä 20.09.2021 **Sivumäärä** 60 + 6 **Kieli** Englanti

Tiivistelmä

Puolijohdekomponenttien kysyntä on kasvanut nopeasti 2020-luvulla, mikä on aiheuttanut pulaa komponenteista ja siten häiriöitä kuluttajatuotteiden saatavuuteen. Siksi puolijohdevalmistajat ovat motivoituneita ottamaan käyttöön koneoppimis- ja automaatioratkaisuja - kuten virtuaalista metrologiaa - parantaakseen tuotannon saantoa ja laatua. Selkeistä eduista huolimatta uusimpien koneoppimismenetelmien käytöstä virtuaaliseen metrologiaan ei ole saatavilla kattavia vertailuja.

Tässä diplomityössä esittelemme ja tarkastelemme monia hiljattain kehitettyjä koneoppimismenetelmiä virtuaaliseen metrologiaan kemiallismekaanisessa hionnassa, joka on keskeinen prosessi puolijohdevalmistuksessa. Vertailemme tyypillisiä ratkaisuja, jotka koostuvat manuaalisesti muodostetuista muuttujista ja puuyhdistelmämalleista, uusimpiin aikasarjaregressiomalleihin, kuten InceptionTimeen ja takaisinkytkettyihin neuroverkkoihin. Esittelemme ja arvioimme myös useita lähestymistapoja, joilla sisällytetään ajan suhteen muuttumattomat ulkoiset prosessimuuttujat takaisinkytkettyihin neuroverkkoihin. Tämän lisäksi kehitämme puoliohjattuja autoenkoodaaja malleja ennustusskenaarioita varten, missä vain murto-osalle prosessiajoista on tiedossa ulostulo. Näitä autoenkoodaajia verrataan muihin koneoppimismenetelmiin asetelmassa, jossa tiedossa olevien ulostulojen määrä on hyvin rajoitettu. Lisäksi arvioimme puuyhdistelmämallien suorituskykyä analysoimalla niiden ominaispiirteiden merkittävyysarvoja. Kokeellinen työ tehdään julkisella tietoaineistolla, mikä mahdollistaa työn vertaamisen muihin viimeaikaisiin julkaisuihin, jotka käyttävät samaa aineistoa.

Kokeemme osoittavat, että manuaalisesti muodostetut muuttujat yhdistettynä puuyhdistelmämalleihin ovat hyvä valinta kemiallismekaanisen hionnan virtuaaliseen metrologiaan. Havaitsimme kuitenkin, että puolivalvotut autoenkoodaajapohjaiset mallit ennustavat metrologia tuloksia tarkemmin kuin ohjatut koneoppimismenetelmät, kun tiedossa olevien ulostulojen määrä on hyvin rajoitettu. Lisäksi merkittävyysanalyysimme puuyhdistelmämallien ominaispiirteistä paljasti, että vain murto-osa prosessidatan mittauksista riittää kemiallismekaanisen hionnan tarkkaan virtuaaliseen metrologiaan. Itse asiassa gradienttitehostettu usean puun malli käytti vain 19 % mittauksista ja saavutti 0,99 selitysasteen testiaineistolla.

Avainsanat Koneoppiminen, tekoäly, virtuaalinen metrologia

Contents

Symbols and abbreviations

Symbols

a	Gradient threshold
$\boldsymbol{c_t}$	Hidden state vector of recurrent neural network at timestep *t*
D	Number of variables
$\boldsymbol{\varepsilon}$	Gaussian noise
F	Number of aggregation functions
$\boldsymbol{g}$	Gradient
$\boldsymbol{h_t}$	Observed state vector of recurrent neural network at timestep *t*
$\boldsymbol{k}$	Kernel matrix
$\boldsymbol{K_p}$	Preston coefficient
$\boldsymbol{\eta}$	Step size coefficient
$\mathbf{P}$	Pressure
$\boldsymbol{\mathcal{R}}$	Regression output
$\boldsymbol{R^2}$	coefficient of determination
$\mathbf{s_t}$	Sensor measurement vector at timestep *t*
$\boldsymbol{\mathcal{T}}$	Time series
$\mathbf{\upsilon}$	Relative velocity
$\boldsymbol{\vec{w}}$	Model weight vector
$\mathbf{x}$	Input matrix
$\boldsymbol{\hat{x}}$	Reconstructed input matrix
$\mathbf{z}$	Latent vector representation

Operators

$\mathrm{A} \times \mathrm{B}$	multiplication of A and B scalar
$\leftarrow$	assignment of new value
$\sum_{i,j} \boldsymbol{a}$	sum over index i and j of matrix **a**
$\tanh(\mathbf{A})$	hyperbolic tangent of matrix **A**
$\mathbf{A} \cdot \mathbf{B}$	element-wise product of matrix **A** and **B**
$\mathbf{AB}$	matrix multiplication of matrix **A** and **B**
$\sigma(\mathbf{A})$	sigmoid function of matrix **A**
$\|\boldsymbol{a}\|$	euclidean norm of vector **a**
$\underset{x}{\operatorname{argmin}} F(x)$	value of x for which F(x) is minimum

Abbreviations

BEOL	back-end-of-line
CMP	chemical-mechanical polishing
CNN	convolutional neural network
FCN	fully convolutional neural network
FEOL	front-end-of-line
GMDH	group method of data handling
GRU	Gated Recurrent Unit
GRU-AE	Gated Recurrent Unit autoencoder
IC	integrated circuit

IID	independent and identically distributed
LSTM	Long Short-Term Memory
LSTM-AE	Long Short-Term Memory autoencoder
MART	Multiple Additive Regression Trees
MLP	multilayer perceptron
MLP-AE	multilayer perceptron autoencoder
MSE	mean squared error
NLP	natural language processing
OES	optical emission spectrometry
PCA	principal component analysis
PHM	Prognostic and Health Management
R2R	run-to-run
ReLU	Rectified Linear Unit
ResNet	residual network
RMSE	root-mean-square error
RNN	recurrent neural network
SGD	Stochastic Gradient Descent
SVR	Support Vector Regression
VAE	variational autoencoder
XGBoost	Extreme Gradient Boosting

1 Introduction

The demand for semiconductor components in consumer products, such as cars, smartphones, and laptops, has increased rapidly in the 2020s which has led to a shortage of electrical components and breaks in availability of the products (Garcevic & Lidberg 2021). To alleviate the shortage, semiconductor manufacturers are striving to find techniques to increase the yield and quality of wafer fabrication. During fabrication, semiconductor devices are built on a circular slice of semiconductor material, called wafer. Modern wafer production is divided into front-end-of-line (FEOL) and back-end-of-line (BEOL) processing. During FEOL processing, the devices are formed in the silicon by deposition, removal of material and patterning. Consequently, in BEOL, the fabricated components are connected, and the connections are isolated with dielectric layers. Both of these process stages consist of dozens of individual processes that are repeated several times during the manufacturing.

After each time a new material layer is deposited on the wafer surface, planarization is required. This is achieved with a process known as chemical-mechanical polishing (CMP) process that utilizes both chemical and mechanical abrasive forces to achieve planarized wafer surface with global uniformity and sufficient surface quality. The planarization rate may vary from wafer to wafer due to uneven surfaces, environmental changes and tool operation. Therefore, CMP process requires careful monitoring, which is costly, if physical measurements are conducted frequently on the wafer. Usually, these physical measures are performed only on lot-to-lot basis (Qin et al. 2006). A lot typically consists of 25 wafers which means that only every 25th wafer is physically measured (Tu & Lu 2017).

To avoid frequent physical measurements, virtual metrology methods are commonly employed for wafer-to-wafer process control (Chen et al. 2005). Virtual metrology applications aim to predict wafer properties based on historical process data without conducting the actual expensive and time-consuming physical measurements. These predictions are then utilized in monitoring and controlling processes, such as CMP, to improve the final yield of the manufacturing. Improved yield has motivated great interest in the topic, and machine learning-based virtual metrology solutions begun to emerge in the beginning of the 21st-century (Yung-Cheng & Cheng 2005) (Lin et al. 2006). The data utilized in virtual metrology systems often consists of multivariate time series data with hundreds of variables including sensor and process information. In order to cast the time series data into a design matrix, the process data is either flattened to rows or aggregated with statistical moments, such as mean, variance, maximum and minimum. Both approaches result in numerous features which make dimensionality reduction necessary for most machine learning models to predict metrology values accurately. These manual feature engineering steps require domain knowledge and additional modeling efforts but may still often lead to information loss and unscalability of the model.

To solve this problem, several recent papers (Terzi et al. 2017) (Maggipinto et al. 2018) (Choi et al. 2019) propose the use of deep neural networks to automate the feature extraction process and hence utilize the raw time series data without aggregation. Many of the proposed models take influence from image recognition and natural language-processing (NLP) research. Both supervised and unsupervised methods have been studied but especially deep autoencoders have drawn high interest in virtual metrology research. Autoencoders are unsupervised neural networks that attempt to learn lower dimensional representation of the input data $\boldsymbol{z} = \boldsymbol{f}(\boldsymbol{x})$ and reconstruct it back to the original dimensional representation $\boldsymbol{g}(\boldsymbol{z}) = \hat{\boldsymbol{x}}$ with minimal loss $\boldsymbol{L}(\boldsymbol{x}, \hat{\boldsymbol{x}})$. This bottleneck structure of the autoencoder neural

network forces the latent representation to learn informative features and properties of the input data which can be then utilized in subsequent machine learning tasks. Autoencoders can learn manifolds from unlabeled data which enables semi-supervised learning – an approach to machine learning where only a small subset of samples is labeled, and vast majority is unlabeled.

The recently published virtual metrology papers have experimented on a variety of different machine learning methods on non-public process tools, but there is an absence of comprehensive comparison of various state-of-the-art deep learning methods for virtual metrology of CMP. In addition, the autoencoder based virtual metrology publications we reviewed – address only the application of the methods to a fully labeled dataset.

Therefore, the goal of this thesis is to develop and compare different deep learning models for virtual metrology of CMP and experiment how the ratio of labeled to unlabeled data affects virtual metrology performance of different machine learning models. The experimental work is conducted on a public CMP dataset[1] which allows simple comparison to other recent CMP publications utilizing the same dataset.

This thesis is divided to six chapters. Chapter 2 provides a literature review on the relevant background topics related to the application domain of virtual metrology and CMP. Chapter 3 continues with the machine learning fundamentals, such as deep learning and time series extrinsic regression that are utilized in our experimental work and implementation. Chapter 4 introduces the design and implementation of our work. In addition, we describe the dataset utilized in evaluation of our implementation and conduct exploratory data analysis on it. Finally, experimental results and findings are presented in Chapter 5 where we discuss and evaluate theme in detail. Chapter 6 concludes the thesis and provides suggestions for future research.

[1] https://phmsociety.org/conference/annual-conference-of-the-phm-society/annual-conference-of-the-prognostics-and-health-management-society-2016/phm-data-challenge-4/

2 Virtual metrology for chemical-mechanical polishing

The process of manufacturing semiconductor devices consists of hundreds of complex process steps and several process phases that utilize even more numerous tools that are engineered to produce high-quality electrical components on the wafer surface. In this chapter, we introduce the fundamentals of semiconductor manufacturing and focus on reviewing a crucial process, chemical-mechanical polishing (CMP), that is repeated several times during the manufacturing. Secondly, we introduce virtual metrology and describe its purpose in semiconductor manufacturing. Lastly, we review current virtual metrology solutions developed for CMP.

2.1 Fundamentals of semiconductor manufacturing

Semiconductor manufacturing is a multistep process where microelectronic devices for integrated circuits (ICs) are produced starting from raw material. After a wafer made of silica is prepared, the metal-oxide-semiconductor microelectronic devices are formed on its surface during wafer fabrication. The fabricated devices are utilized in integrated circuit chips that are present in practically every modern electrical device. Even most mundane modern-day objects, such as LED lamps and speakers, contain multiple semiconductor components.

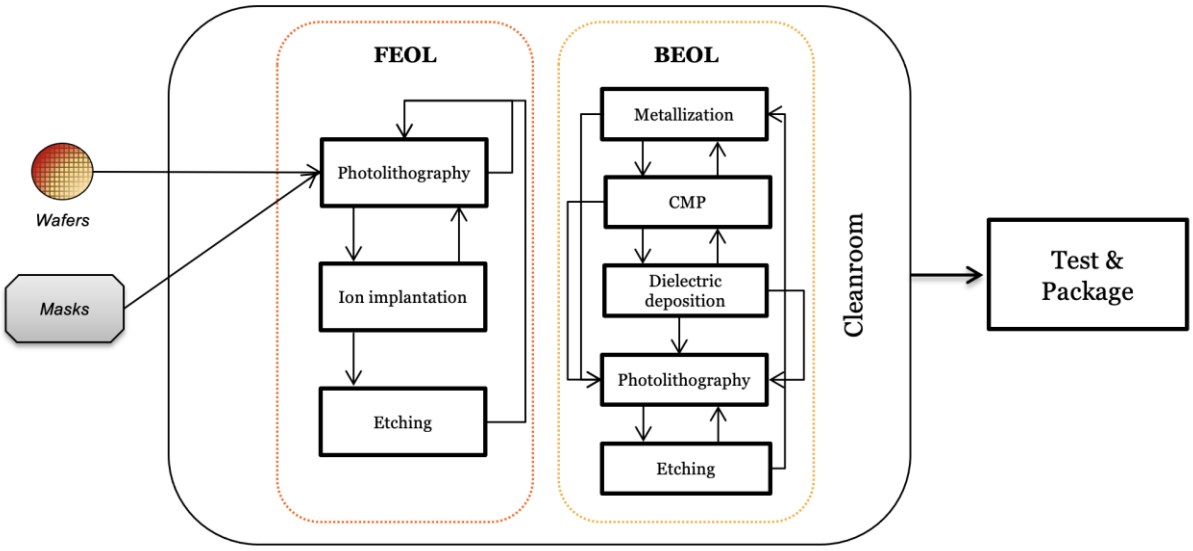

Figure 1: Fabrication of semiconductor devices consists of numerous process steps that are repeated in varying order depending on the fabricated product.

Semiconductor wafer fabrication begins from designing the circuit based on the desired functionality and layout. The design of the circuit is crucial in creating masks that are utilized in photolithography. After the design and mask are set, the devices are then formed on the wafer surface layer by layer in a semiconductor factory, often referred as fab. A fab comprises of multiple clean rooms that have low level of particles in the air, which reduces unnecessary contaminations and defects in the devices. In the fab, physical device fabrication begins with the front-end-of-line (FEOL) processing where the transistors of the devices are deposited and formed on the wafer surface with photolithography and subsequent ion implantation. Photolithography patterning is based on applying photoresist material on the wafer surface

and masking the surface before exposing it to UV light that develops the designed circuit layer on the wafer. An etching process may supersede photolithography, if it is necessary to remove unwanted oxide from the wafer surface. In modern device fabrication, wafer may undergo photolithography up to 50 times (Grayson 2018).

Consequently, in back-end-of-line (BEOL) processing, the electrical components fabricated during FEOL are connected by depositing metal layers on the wafer. These connections are then isolated with dielectric layers. This is achieved with dual-damascene fabrication process where photolithography is always followed by an etching step (Xiao 2012, p.19). Each complete layer is typically polished with a CMP tool to achieve a uniform and planarized wafer surface to ensure the quality of interconnections. Lastly, a passivation layer is deposited and pads for wire bonding are exposed. The devices are then tested, sawed out of the wafer, and packaged.

In the 1960s, the standard wafer diameter was 25 mm, whereas today in the 2020s the state-of-the-art is 300 mm allowing more devices on a wafer and thus efficient device fabrication. One 300 mm wafer may contain hundreds or even thousands of semiconductor devices depending on the fabricated chip size. The fab equipment is designed to handle one specific diameter size, and therefore the fabs are categorized by the wafer diameter they are compatible to process. Contradict to wafer sizes, the devices have continuously shrunk from the 1960s. A modern feature size of IC chip is around 10 nm, whereas 60 years ago in the 1960s the feature sizes were thousand-fold and were measured in micrometers (Wong et al. 2020). Decreased feature sizes create demanding requirements for advanced semiconductor manufacturing tools, such as CMP, and surrounding technology to preserve good yield in production.

2.2 Chemical-Mechanical Polishing

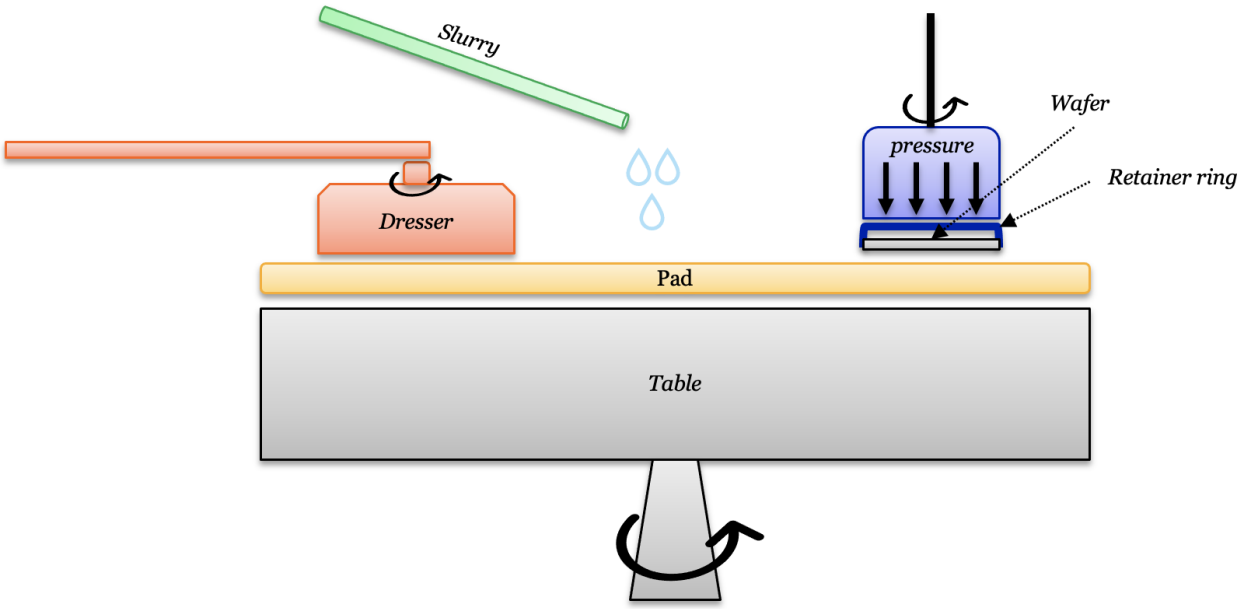

Figure 2: Overview of a chemical-mechanical polishing tool.

Chemical-mechanical polishing (CMP) is a planarization process that utilizes both chemical and mechanical abrasive forces to achieve planarized wafer surface with global uniformity

and sufficient surface quality after a metal or dielectric layer is deposited on the wafer surface. It was developed in 1990s when the number of deposited layers in devices increased and dimensions reduced, which consequently meant that chemical etching nor abrasive force-based planarization techniques alone could provide the desired planarization and smoothness of wafer surfaces with high precision (Steigerwald et al. 1995) (Wang et al. 1998). Nowadays, CMP is a standard process step in semiconductor manufacturing that is repeated multiple times during wafer fabrication, although initially direct physical contact with the wafer surface was condemned by the industry. Later, it was observed that CMP actually reduces defect density and improves yield besides planarizing the wafer surface (Xiao 2012, p. 508). The operating principle of CMP is divided to two components: mechanical and chemical.

2.2.1 Mechanical

Mechanically, the polishing effect is achieved by directing pressure on the wafer with a polishing pad and rotating the wafer carrier and the polishing pad simultaneously creating relative velocity between the two surfaces. This transfer of mechanical energy breaks bonds of atoms in the surface and results to material removal from the surface. The material removal rate is described by Preston equation - an empirical model named after its developer:

$$material\ removal\ rate = K_p \times \mathrm{P} \times \upsilon \qquad (1)$$

where K_p refers to a Preston coefficient, P to pressure, and υ to relative velocity between the pad and wafer surface (Nanz & Camilletti 1995). Preston equation describes the phenomenon well - that higher areas on the wafer surface are polished with higher rate because the pressure is higher - naturally - for smaller areas that are above majority of the surface. This characteristic of pressure is a reason for CMPs success in achieving globally planarized surfaces (Babu 2016). The Preston coefficient is a scalar value that is determined empirically. The exact material removal rate is obtained by physically measuring the wafer film thickness before and after polishing.

The polishing pad is typically made of flexible polymer substance which has appropriate hardness and roughness. An extremely hard polishing pad can cause scratches on the wafer surface, but an excessively soft polishing pad removes material too slowly. The polishing pad wears in use and smoothens, which affects its polishing properties. Hence, it is important that a dresser is used regularly between or during process runs in order to condition the polishing pad and clean excess slurry from the pad surface to maintain polishing properties of the pad. The dresser is made of stainless steel that is coated with diamond. Like the polishing pad, also dresser material and composition are chosen according to the polishing process specifications.

During polishing, the wafer is mounted on a wafer carrier and held in place with a vacuum and a retainer ring. The downward force on the wafer is directed and applied uniformly with a polishing membrane that is made of soft rubber-like material. The wafer carrier includes several components, such as the retainer ring, with adjustable pressure to allow control of the polishing process and wafer placement in the wafer carrier. The polishing membrane

wears in use which affects the polishing result. Therefore, the membrane is replaced around every thousand wafers (Xiao 2012, p. 521).

2.2.2 Chemical

The chemical component of the process is based on slurry that is dispensed on the polishing pad. The slurry is typically colloidal silica that contains abrasives and wafer material specific chemicals. Its main purpose is to dissolve and transport removed material from the wafer surface. In addition, the slurry forms a film on the wafer surface which protects lower areas of the surface that are not physically in touch with the polishing pad (Steigerwald et al. 1995). The type of slurry is selected based on the specific CMP type and its consistency is engineered to achieve desired material removal rate, uniformity and planarization. Different components of the slurry are stored separately and only combined just before use. Several slurry dispensers manage the rationing of the slurry based on process control parameters, which naturally affects material removal rate and uniformity of the wafer. Another factor that requires monitoring is the quality and age of the slurry, since they are under strict temperature control and preserve their chemical properties for approximately a year (Xiao 2012, p. 523).

Eventually, the condition of various CMP tool components, such as the slurry, dresser and polishing table, degrade and begin to affect the polishing results. This supports the common understanding (Nanz & Camilletti 1995) that Preston equation (1) is not sufficient to describe the material removal rate that responses to numerous other factors. For example, slurry dispensation might vary uncontrollably which introduces unintended variation to material removal rate. Unfortunately, it is not economically feasible to physically inspect and measure each wafer to control and monitor the material removal rate, which is why virtual metrology of CMP process is a common topic in virtual metrology literature (Lee & Kim 2020).

2.3 Virtual metrology

Process-quality assurance is an essential part of successful semiconductor manufacturing because the production of a wafer may take up to 6 months[2]. As devices further shrank in the beginning of 21st century, lot-to-lot metrology was no longer sufficient for critical process steps. The reason for this, is that a lot consists typically of 25 wafers which means that only every 25th wafer is physically measured when conducting metrology on lot-to-lot basis (Tu & Lu 2017). Hence, wafer-to-wafer metrology was required which ultimately became impossible to conduct physically as the number of devices and process steps increased. This created the demand for virtual metrology applications which allow virtual "measurement" of physical quality of every wafer. (Yung-Cheng & Cheng 2005)

Virtual metrology applications aim to predict wafer properties based on historical process data without conducting expensive and time-consuming physical measurements on each wafer. Commonly, the physical variable of interest is a defect pattern (classification) or a quantifiable effect of the process (regression) prediction. Virtual metrology does not remove the need for physical metrology completely but instead provides other significant economic

[2] https://sea.pcmag.com/feature/19618/how-a-chip-gets-made-visiting-globalfoundries

benefits, such as cost savings in metrology equipment and reduced physical metrology bottlenecks in production line.

Virtual metrology predictions are utilized in monitoring and controlling processes which are critical in semiconductor manufacturing. Monitoring processes with virtual metrology predictions reduces spoilage by stopping early malfunctioning processes that are out of normal operating range. Normal operating range is defined as the area inside upper control limit and lower control limit, which are computed based on the process output variation. Virtual metrology predictions give the process operator an unbiased estimate of the process quality without the time-delay that is present in lot-to-lot physical metrology (Kang et al. 2009). In addition, virtual metrology predictions and confidence levels are often utilized in dynamic sampling strategy where the physical metrology is conducted more frequently on wafers with risk to be out of control limits (Susto 2017).

Furthermore, virtual metrology-based process control allows high-frequency adjustment of tool specifications on a run-to-run basis. A run-to-run controller (R2R) requires the outcome of the process or metrology value to adjust the process parameters and thus retain the process under control. Without virtual metrology, R2R controller can only adjust recipe parameters after physical metrology (Jebri et al. 2017). Both advanced monitoring and controlling improve yield and general quality of manufacturing, which motivates the extensive use of virtual metrology systems in production environments and makes it viable economically.

The efficient use of advanced virtual metrology systems set certain requirements on the collected data. Firstly, the sampling frequency of the process parameters and process output should be high enough. It is common that only aggregate values, such as mean and standard deviation, are collected from the runs. Secondly, in addition to process sensor data, the collected data should include necessary identifier variables that allow differentiation between different process stages, wafers, and process tools (Orji et al. 2018). Lastly, the dates of significant maintenance jobs and consumable changes should be available for the continuous accurate modelling of the process. Process data naturally contains some noise and outlier values, but they do not generally prevent the utilization of a virtual metrology system as is the case with insufficient data collection practices.

Initially, virtual metrology was offered as a concept of finding process variables that correlated with the physical metrology, and estimating the metrology based on that. In fact, Chen et al (2005) stated that “The correlation between process tool and final wafer result is called virtual metrology.” However, they identified that an accurate estimation technique is required for practical applications. Yung-Cheng & Cheng (2005) was one of the first to describe a virtual metrology application that utilized modern machine learning techniques. They implemented a neural network that consisted of an input layer with 2356 neurons, 2 hidden layers, and 3 target neurons. Next, we further review virtual metrology systems proposed in recent academic publications.

2.3.1 Modern applications

Modern virtual metrology applications utilize machine learning and statistical methods. A common pipeline consists of data acquisition, preprocessing, dimensionality reduction, and regression model (Figure 3). In data acquisition, the process data, information and physical metrology results are collected from the process tool and metrology equipment. Next, in the preprocessing phase, the time-varying data is typically aggregated with summary statistics,

such as mean, variance, maximum and minimum, and the data is casted in matrix format. This results to

$$D \times F \qquad (2)$$

candidate variables (D=no. original variables, F=no. aggregation functions) that require dimensionality reduction to be usable in most regression models. Dimensionality reduction refers to utilization of feature selection or feature extraction methods, such as feature subset search or principal component analysis (PCA), respectively.

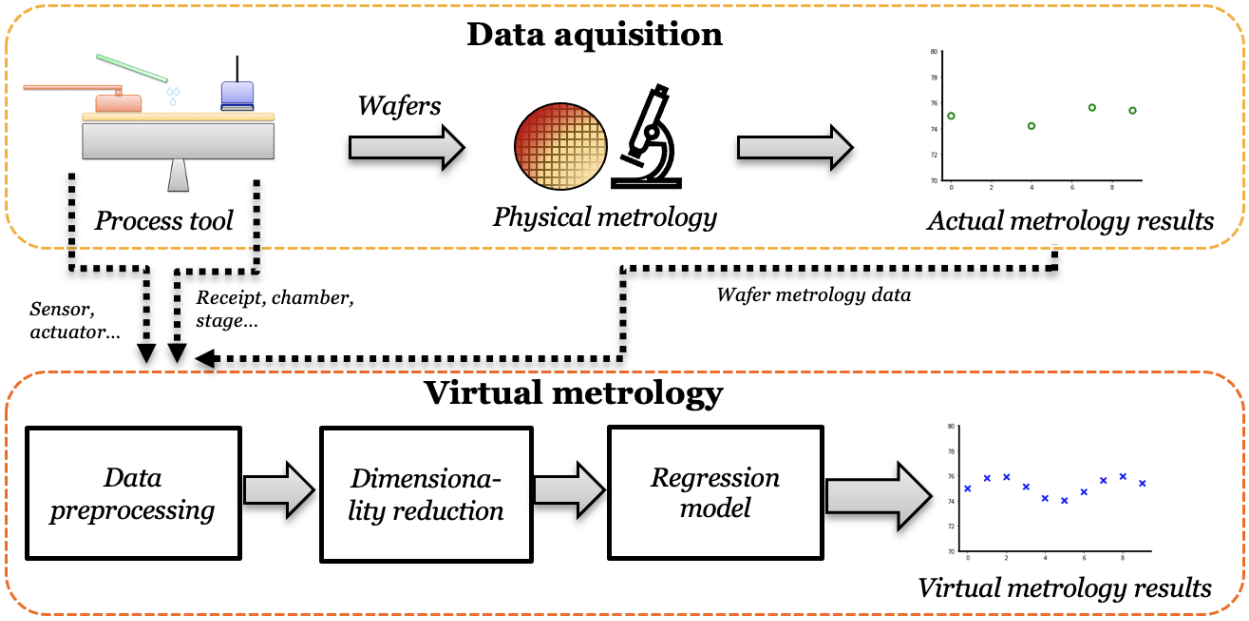

Figure 3: Overview of a basic virtual metrology pipeline.

Kang et al. (2009) reviews several combinations of dimensional reduction techniques and state-of-the-art regression models. Initially, they have a dataset with 1536 and 1792 input variables for two separate process tools, and only 118 and 241 wafers for those two tools. They achieve the best overall accuracy with Support Vector Regression (SVR) prediction model and stepwise linear regression feature selection. However, other prediction models, such as neural networks and linear regression performed similarly to SVR.

Ragnoli et al. (2009) also recognize that a central challenge in virtual metrology systems is the feature selection and extraction, from the numerous recorded variables, in a manner that maximal relevant information for the process is preserved. The dataset in the study is from Plasma etching optical emission spectroscopy (OES) and consists of 2000 etch rate samples with 2048 time-varying channels. After using 6 summary statistics, they end up with dataset with over 12 000 variables. They evaluate the performance of different feature selection methods and regression models, but do not provide any alternative for using statistical moments in aggregation of the process data.

Maggipinto et al. (2018) propose an autoencoder approach for feature extraction in virtual metrology of etch rate estimation. Their solution utilizes convolutional layers and subsequent SVR model for generating predictions. The autoencoder approach outperforms their baseline that consists of computing four statistical moments, PCA, and LASSO or SVR

regression model. Another autoencoder approach (Choi & Jeong 2019), proposes the use of multi-layer perceptron neural networks for virtual metrology. Their solution outperforms several different baselines in etching process virtual metrology but achieves an R^2 score of 0.32 in validation experiments. Neither Maggipinto et al. (2018) or Choi & Jeong (2019) mention the use of unlabeled data besides the labeled data in training their proposed solutions. In addition to autoencoders, the use of supervised deep learning for virtual metrology have been proposed (Terzi et al. 2017). This supervised convolutional model is trained with OES data to predict etching results. Their proposed deep learning model achieves significantly better results than simple Ridge regression model utilizing only mean values of each wavelength.

Section 3.2 reviews basic deep learning concepts, such as autoencoders that are feature learning neural networks that try to learn latent representations from the input data that contains informative features and properties of the original representation. Next, we will review previous virtual metrology solutions for CMP.

2.3.2 Chemical-mechanical polishing applications

CMP process was adopted widely in industry during the early 2000s but most virtual metrology studies for the process were only published in 2019 – a couple years after a CMP dataset became public through a Prognostic and Health Management (PHM) Data Challenge[3]. Previously, the proposed solutions for CMP virtual metrology had achieved low R^2 values (75%-96%) and had not utilize modern machine learning methods (Kong et al. 2010) (Rao et al. 2014). Most of the recent publications utilize the same PHM dataset in evaluating the performance of the proposed solution, which allows easy comparison between proposed solutions[4]. In addition, in all of these CMP virtual metrology publications the predicted metrology measurement is the average material removal rate. This section presents some of the most relevant CMP virtual metrology publications that use the PHM dataset in their experiments (Table 1).

Li et al. (2019) proposes a decision tree-based ensemble method for the prediction of the average material removal rate. They utilize a stacking technique to combine the predictions of three decision tree-based regression models that use aggregated feature space. They compare the performance of the proposed method to individual regression models and evaluate different ensemble meta-regressors. CART[5]-based stacking method achieves the smallest root-mean-square error (RMSE) for the validation dataset with 4.035 average error for all stages, whereas ELM[6]-based stacking method perform the best for the test dataset with 4.64 average error for all stages with the PHM dataset.

Yu et al. (2019) argues that both physics-based and data-driven models have advantages and disadvantages, and hence propose a physics informed machine learning model in their paper that uses PHM dataset as case study. They do not divide the dataset or use multiple models for different part of the data. The proposed model combines a multi-scale

[3] https://phmsociety.org/conference/annual-conference-of-the-phm-society/annual-conference-of-the-prognostics-and-health-management-society-2016/phm-data-challenge-4/

[4] NOTE: Some of the papers do not provide results for both validation and test set which may affect the performance estimates. Otherwise, the results are well comparable.

[5] Classification and regression tree

[6] Extreme leaning machines

mechanical model with random forest and achieves RMSE of 16.908 for the validation and 15.770 for the test dataset.

Jia et al. (2018) propose a method based on group method of data handling (GMDH) type polynomial neural networks. They conduct a manual feature extraction procedure for the PHM dataset that differs from other papers. Besides statistical moment aggregation, they handcraft two set of additional features: time neighbors and usage neighbors. Time neighbors refer to metrology values of previous wafers and usage neighbors refer to past runs with similar usage measures of consumables. The model is dynamic which means that the model is trained after each timestep during validation, and new time and usage neighbors are inputted to the data. In addition, they divide the dataset into 3 distinct groups based on material removal rate and train a model for each group separately. With these procedures they achieve a test RMSE of 2.706 with linear regression and test RMSE of 2.606 with their GMDH type polynomial neural networks.

Table 1: CMP virtual metrology publication performances for PHM dataset.

Authors	Feature extraction	Dimensionality reduction	Model	Performance (RMSE)
Li et al. (2019)	Four statistical moments in time-domain and three statistical moments in frequency domain	Feature selection with random forest feature importance	Tree-based ensemble	4.035 (validation), 4.64 (test)
Yu et al. (2019)	Physics-based features	1/3 of the variables used	Physic-informed random forest	16.908 (validation), 15.770 (test)
Jia et al. (2018)	Statistical moments, time and usage neighbors. Dataset divided to three groups	Integrated to prediction model	Group method of data handling type polynomial neural networks	2.606 (validation), result for test dataset not specified
Wang et al. (2017)	Raw data of six manually chosen usage variables	Integrated to prediction model	Deep belief network	2.7 (validation), result for test dataset not specified
Di et al. (2020)	Statistical moments, time and usage neighbors. Dataset divided to three groups	Features selected with student's T-test and out-of-bag feature importance	Ensemble including persistent model, KNN, linear regression, tree bagging and SVR	2.66 (validation), result for test dataset not specified

Wang et al. (2017) develop a static data-driven model based on deep belief network that utilizes only 6 usage features of the PHM dataset that are manually chosen. They claim that even though the pressure and rotational speed settings affect the material removal rate, they do not improve the prediction accuracy as those parameters are controlled in the production environment. Particle swarm optimization determines the final network structure and learning rate. Furthermore, they divide the dataset to two groups based on the processing chamber and reach an average RMSE of 2.7 on the test dataset.

Di et al. (2020) propose a similar data-driven solution as Jia et al. (2018) with the main difference being the feature selection and chosen regression model. They utilize same

feature extraction where hundreds of hand-crafted features are engineered with domain expertise. Di et al. solution incorporates a discrete feature selection step. The selected features are fed into an ensemble regression model that consists of persistent model (predict same material removal rate as last measurement), K-nearest-neighbor regression (KNN), linear regression, tree bagging and SVR. The predictions of all these models are combined with a weighted average based on cross-validation error. Their proposed solution achieved the lowest error in PHM 2016 Data Challenge with an RMSE of 2.66.

In Chapter 3, we introduce the essentials of machine learning that enable these accurate virtual metrology solutions. We cover the basics of deep learning including different types of neural networks. In addition, we describe a specific machine learning task called time series extrinsic regression, which is often present in virtual metrology.

3 Machine learning essentials

In modern manufacturing industry, machine learning predictions are utilized in the automation and operation of machinery and process equipment as well as in mining information from data for manufacturing experts. Enriched information inferred from the data allows process experts to make well-informed decisions and relieves the burden of conducting routine day-to-day tasks that rarely result in concrete actions. This is the case with virtual metrology that frees resources from physical metrology and informs the process operators, if the physical properties of the wafers deviate from normal.

Machine learning allows the computer to learn rules and patterns from data automatically that would otherwise be impossible for a human to infer. Furthermore, machine learning scales to huge data masses with computational resource that are available today and is capable to produce predictions for samples rapidly. These advantages motivate the development of more advanced machine learning solutions.

The performance of virtual metrology is significantly affected by the selection of the machine learning model. Nowadays, there is a wide selection of machine learning algorithms that are suitable for virtual metrology but some perform better in a real production environment. Some of the models are based on cutting edge deep learning research, whereas others are based on decades old studies of statistical modelling.

This chapter introduces the basic concepts of tree-based ensemble models and deep learning – including different neural network types and training of these models. In addition, we review a machine learning research topic called time series extrinsic regression which is the general field of research for a machine learning task that is encountered frequently in industrial machine learning solutions. This chapter provides the background for our experiments that are further described in Chapter 4.

3.1 Tree-based ensemble models

Many of today's state-of-the-art classification and regression models, such as random forest, are based on an ensemble of decision trees (Rokach & Maimon 2008). These ensemble machine learning models utilize a committee of weak learners in learning a machine learning task accurately. A weak learner is a learning algorithm that performs better in a prediction task than random guessing or simple heuristic models, such as sample mean of the response variable (Hastie et al. 2009, p. 383). A weak learner is typically trained with some randomness in order to introduce difference in the resulting set of weak learners. This is achieved by training the weak learners with subsets of the original observations or variables. Furthermore, the predictions of the weak learners are combined with a majority vote (classification) or averaging (regression) to produce the final prediction of the ensemble model. A typical weak learner is a one-level decision tree that is essentially a decision threshold model.

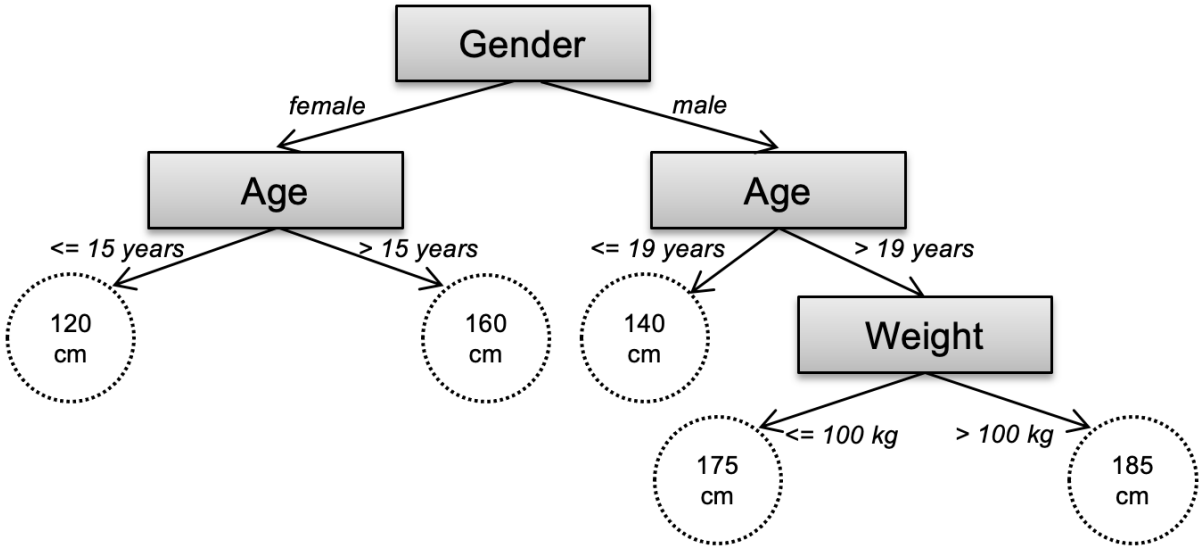

Figure 4: Example of a decision tree for predicting height of a person based on gender, age, and weight.

A decision tree is a divide-and-conquer type of machine learning model that splits the input feature space to finite discrete regions starting from a root node and ending in leaf nodes with different outputs. The average value of the response variable inside the leaf nodes region determines the decision tree output. Decision trees are applicable for both classification and regression tasks with minor modifications on optimized loss functions. For a regression tree, it is computationally infeasible to find a tree that guarantees to minimize the mean-squared-error (MSE) of predictions in the training data. However, it is possible to find a good solution with a greedy approach (Hastie et al. 2009, p. 308):

BuildDecisionTree(X, y, R_0):

$k \in F, where\ F\ is\ variables\ in\ X$
$R_1, R_2 \subset R_0$
$X, y \in R_0$

// Regions R_1 and R_2 inside R_0 divided after a split at point p
$R_1(k,p) = \{X, y \mid X_k \leq p\}, R_2(k,p) = \{X, y \mid X_k > p\}$

// Find the best variable k with the best split point p by minimizing:

$$min_{c_1}\left(\sum_{y_i \in R_1(k,p)} (y_i - c_1)^2\right) + min_{c_2}\left(\sum_{y_i \in R_2(k,p)} (y_i - c_2)^2\right),$$

// where $c_1 = mean(\ y_i | R_1(k,p)\ )$ and $c_2 = mean(\ y_i | R_2(k,p)\ )$.

// Call recursively and split in new regions to smaller regions

IF NOT *stopping_criterion*:
  BuidDecisionTree(X, y, R_1)
  BuidDecisionTree(X, y, R_2)

The stopping criterion indicated in the pseudocode is usually a threshold on the decrease of MSE as a result of the split or a sufficiently small number of observations in the region to be split. To control the size of the decision tree, a method called pruning is commonly used after building the tree. This results to a subtree of the originally built tree which can prevent overfitting the model to the data. The maximum depth of the tree is often utilized as a stopping criterion as well.

Random forest is one of the most encountered tree-based ensemble models. The modern-day version of it was introduced by Leo Breiman in 2001. Random forest combines multiple decision trees – typically hundreds - with bagging[7] and the use of different random subsets of variables for training the individual learners. This results to several uncorrelated decision trees that are on average unbiased and reduce the ensemble model's variance (Hastie et al. 2009, p. 587). A random subset of variables is sampled at each split and the best split is searched from those features. The typical size of the random variable subset of variables is $\sqrt{p}$ where p is the number of original variables. In general, random forest performs better than a single deep decision tree and is fast and reliable to train with little tunning required (Hastie et al. 2009, p. 590).

Another state-of-the-art tree-based ensemble model besides random forest is *gradient boosted trees*. It is based on boosting several decision trees to build an accurate ensemble model. Boosting is an alternative ensemble meta-algorithm for bagging. Boosting is a broad term for training multiple weak learners iteratively and weighting more inaccurate or misclassified predictions in the training of subsequent weak learners that form the ensemble model. Gradient boosted trees incorporate this feature by training the subsequent weak learners $h_m(X)$ with the residuals $(y - F_{m-1}(X))$ of the last iteration's ensemble model where y corresponds to observed responses and F_m to the current ensemble model. In practice, this means that the next weak learner $h_{m+1}(X)$ attempts to fix the errors of the previous ensemble model F_m (Hastie et al. 2009, p. 342). Consequently, this results to iterations that attempt to solve an arbitrary differentiable loss function $\mathcal{L}_{MSE}$. In the case of a regression task, it means that

$$\mathcal{L}_{MSE}(y, F_m(X)) = \frac{1}{2}(y - F_m(X))^2 \qquad (3)$$

$$h_m(X) = y - F_m(X) = -\frac{\partial L_{MSE}}{\partial F_m} \qquad (4)$$

describe the updates throughout iterations and that the updates are towards negative gradients. The size of a new correction step is adjusted after the iteration by optimizing the learner weights γ_m to minimize the loss function $\mathcal{L}$ across all observations. This is expressed as solving an optimization task:

$$\gamma_m = arg\min_{\gamma} \sum_{i=0}^{n} \mathcal{L}\ (y, F_{m-1}(X_i) + \gamma h_m(X_i)) \qquad , \qquad (5)$$

[7] Bagging, also known as bootstrap aggregation, refers to a machine learning ensemble meta-algorithm where the original observations are sampled uniformly with replacement and several bootstrap samples are formed in that way. Each model is then trained with its own bootstrap sample and the model predictions are averaged to produce final predictions.

where n is the number of observations. This is the general structure of training a gradient boosted trees model, but there are a wide variety of implementations, such as Xtreme Gradient Boosting (XGBoost) and Multiple Additive Regression Trees (MART) that include other features as well. XGBoost, for instance, provides a regularized gradient boosting model that is trainable on GPU to reduce training times on bigger datasets.

Although tree-based ensemble models abdicate the interpretability of simple decision trees[8], there are methods to explain the global feature importance in an ensemble tree model. For a regression model, this feature importance score is computed by summing the loss function reduction across all splits in the model by feature. Hence, this gives an interpretation of the magnitude of a feature's effect on the predictions in the ensemble model. Typically, the sum of loss function reduction is normalized for reporting and comparability purposes.

3.2 Deep learning

Deep learning is another sub-field of machine learning that utilizes artificial neural networks in computational modeling. The key idea in deep learning is to learn a representation of the data in a way that makes the actual learning objective easier to reach. Desirably, this removes the necessity for a separate feature extraction step that is prominent in any traditional machine learning system. Manual feature extraction steps require considerable amount of time and domain expertise and do not guarantee generalization in the future. In fact, sometimes finding decent hand-crafted features might prove out to be nearly impossible for human (Goodfellow et al. 2016). Neural network expediate and automate adaptation to changes in the task environment, which reduces the necessity for human intervention. However, training neural networks requires large amounts of data and computation power, and for this reason the selection of an appropriate neural network architecture is crucial in achieving satisfactory modeling results. This section describes some of the most encountered deep learning concepts, such as neural network layer types, architectures and training procedure, and outlines some of the application domains where these solutions are deployed in practice.

3.2.1 Training neural networks

Neural networks, like any machine learning models, require a training procedure to learn the desired machine learning task. Moreover, neural network models include hundreds of parameters that require re-adjusting in order for the model to make accurate predictions. The optimization of neural networks is based on iterative gradient descent similarly to training of some other machine learning models, such as logistic regression. A gradient descent algorithm updates the model weights as

$$\vec{\boldsymbol{w}} \leftarrow \vec{\boldsymbol{w}} - \eta \boldsymbol{g} \quad , \quad (6)$$

where $\vec{\boldsymbol{w}}$ is a model parameter vector, η learning rate also known as the step length, and $\boldsymbol{g}$ the error gradient. Due to the non-linearity of neural networks, the weights of the model

[8] For instance, predictions are clearly explainable in the Figure 4.

cannot be estimated explicitly the same manner as with linear regression model where the model weights are solvable, for example with an ordinary least squares estimation method:

$$\vec{\boldsymbol{w}} \leftarrow (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y} \qquad (7)$$

The optimization of neural network is non-convex which means that the initial weights of the network affect the final learning result and convergence to globally optimal weights is not guaranteed. However, appropriate optimization methods allow the neural networks to reach closer to the global optimal region.

Stochastic Gradient Descent (SGD) or some variation of it is commonly used as an optimization algorithm for minimizing the cost function, such as mean squared error (MSE) for regression, or cross-entropy for classification task, during training of neural networks. SGD updates the network parameters towards an estimated negative gradient and closer to a locally optimal region. The contribution of weights and biases on the error is distributed with an algorithm called backpropagation, which is based on computing the gradient with respect to each weight in the neural network by the chain rule. For long periods in the mid 20th-century, training neural networks with backpropagation was considered computationally too heavy, but when the amount of computational resources became sufficient - later in 2000s - backpropagation was widely adopted to use.

3.2.2 Neural network types

Deep learning models consist of several layers that learn to extract a useful representation from the data. Different neural network types are good for specific application domains and thus the architecture of the deep learning model must be chosen according to the task the network is desired to learn. Generally, more complex models require more data and training time, whereas simpler models might not learn difficult tasks due to limited capacity but are easier to train. The most common neural network types are feedforward, convolutional, and recurrent networks. A neural network model may consist of several different types of networks, and therefore the division is not strict but rather descriptive in its nature.

Feedforward networks or multilayer perceptrons (MLPs) consist of several linear layers that are stacked after each other. The layers between input and output layers are called hidden layers. Neurons on the same layer are not connected to each other, but each neuron is connected to each neuron on the subsequent layer. For this reason, MLPs are considered fully connected neural networks. An activation function is commonly used after each neuron to introduce non-linear features to the next layer. Activation functions are scalar-to-scalar functions that map the outputs of hidden neurons to a reasonable range. A good example of this type of function is Rectified Linear Unit (ReLU), defined as

$$f(x) = \max(0, x) \quad , \qquad (8)$$

which is utilized frequently in MLPs. Activation functions and the depth of the MLP allows the last linear layer to produce accurate predictions when the information is fed forward in the network (Patterson & Gibson 2017). Information in the network moves only forward or backwards and does not form a cycle through feedback loops. Feedforward networks attempt to estimate a function f that maps inputs $\mathbf{x}$ to output y. In theory, MLPs can learn to

estimate any function f, if the number of layers and neurons is sufficient in the model (Goodfellow et al. 2016). MLPs are prevalently utilized for tabular data that has no spatial structure and consists of a fixed number of feature columns, but recently they have attracted attention again in image recognition as well (Tolstikhin et al. 2021).

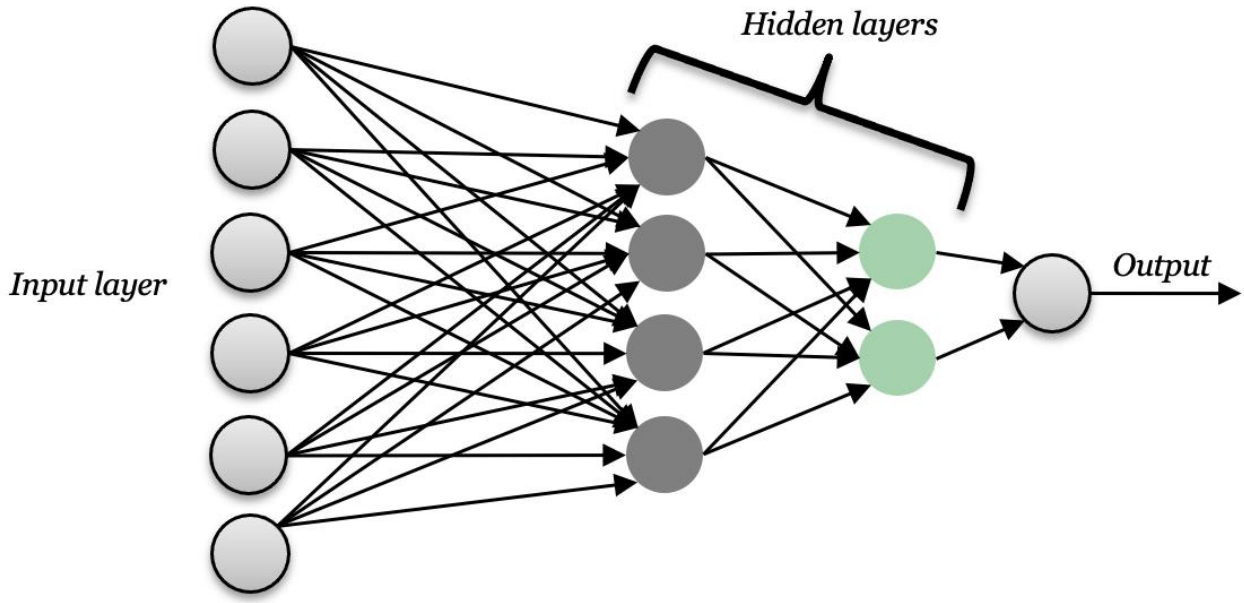

Figure 5: Simplified example of a feedforward network.

Convolutional neural networks (CNNs) are a special type of feedforward networks that are not fully connected as MLPs. CNNs utilize mathematical operations called convolutions at least on some layer of the entire network. Convolution is a linear operation between two functions or matrices and can be thought as a filtering operation of a signal. In CNNs, the operation is usually a dot product between the specified convolution kernel with specific size and different parts of the input matrix which then form the output of the layer (Figure 6). CNNs are especially useful in applications using image or sequential data. This type of grid data typically has a topological structure which allows convolutional layers to extract features or patterns that appear in certain area of the data. For example, CNN may filter out the color of the background of an image but still retain information on the character that is classified. Convolutional layers are usually stacked on top of each other to extract different types of features in different granularity. Typically, each convolutional layer is followed by an activation function and a pooling layer. Pooling layers compute the maximum value from each segment in the representation based on the kernel size of the pooling layer. This reduces the spatial size of the representation, and consequently, the likelihood of overfitting to the data (Goodfellow, et al. 2016, p. 339). Pooling layers do not contain trainable parameters but are instead fixed functions with configuration parameters, such as kernel size and stride. After several convolutional layers, CNNs typically incorporate a fully connected linear layer to compute the final output of the model.

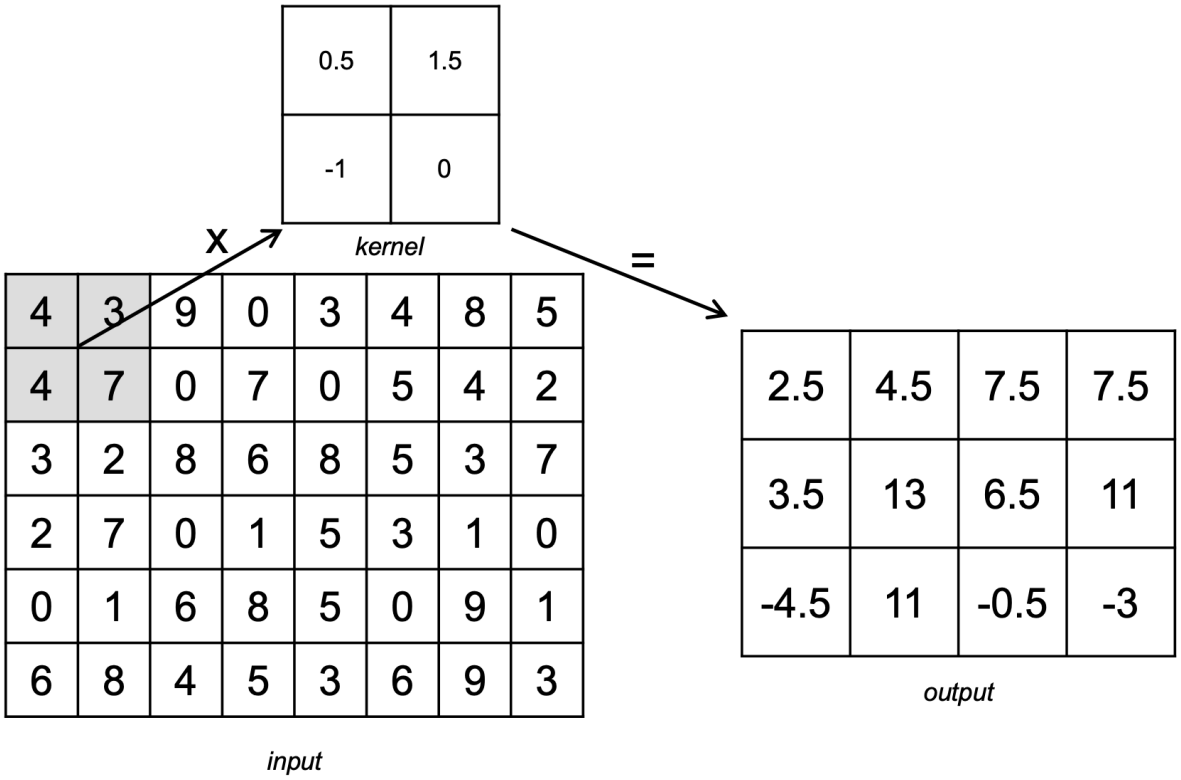

Figure 6: Simplified illustration of convolutional layer's computations.

Another neural network type specialized for handling sequential data (x_0, x_1,.., x_T) is recurrent neural network (RNN). RNN, as its name suggests, utilizes recurrent architecture in processing each step in a sequence which allows it to share parameters for different parts of the sequences and thus process variable length sequences. An RNN cell usually receives a piece (x_t) of the sequence data and a hidden state (h_t) as inputs, and the cell outputs a hidden state (h_{t+1}) after each step. This hidden state is recursively fed into the next computation with the next value of the sequence (x_{t+1}). The recurrent architecture of RNN allows it to process longer sequences without increased model complexity which is inevitable when using an MLP for sequential data. Unfortunately, longer sequences result in a RNN specific issue called vanishing gradient problem which refers to a phenomenon during the training of the model where the gradients of the backward pass become vanishingly small (Hochreiter 1991). Training models using gradient descent with small gradients is inefficient and results in bad learning performance (Bengio et al. 1994).

As a solution to the vanishing gradient problem, Hochreiter & Schmidhuber (1997) proposed a variation of RNN called Long Short-Term Memory (LSTM). LSTM consists of four neural layers and three gates that control the information flow. It includes a “self-loop” and cell state that allow the error gradients to flow in longer sequences. The self-loop is gated which means that the unit learns to control information memorization optimally.

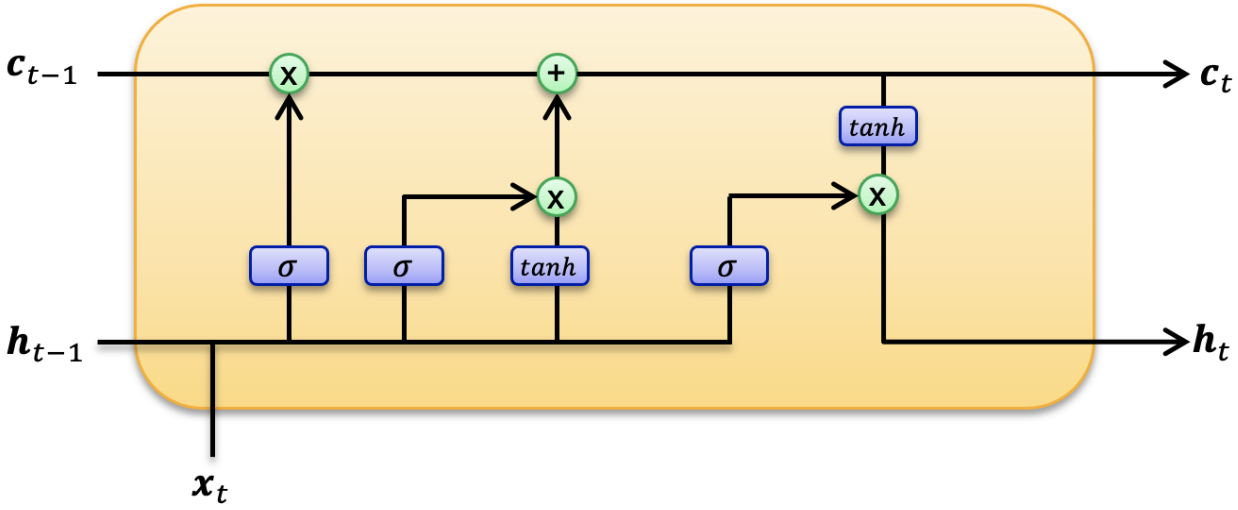

Figure 7: Visualization of computations in a LSTM unit.

This structure and the hidden state allow LSTM to remember information and operate well with longer sequences. LSTM outputs two states: observed ($\boldsymbol{h}_t$) and hidden ($\boldsymbol{c}_t$). These states are computed for each step in the sequence with the following functions:

$$\boldsymbol{h}_t = \boldsymbol{o}_t \cdot \tanh \boldsymbol{c}_t \tag{9}$$

$$\boldsymbol{c}_t = \boldsymbol{f}_t \cdot \boldsymbol{c}_{t-1} + (1 - \boldsymbol{f}_t) \cdot \boldsymbol{i}_t \cdot \tanh(\vec{\boldsymbol{w}}_c \boldsymbol{h}_{t-1} + \vec{\boldsymbol{u}}_c \boldsymbol{x}_t) \tag{10}$$

$$\boldsymbol{o}_t = \sigma(\vec{\boldsymbol{w}}_o \boldsymbol{h}_{t-1} + \vec{\boldsymbol{u}}_o \boldsymbol{x}_t) \tag{11}$$

$$\boldsymbol{f}_t = \sigma(\vec{\boldsymbol{w}}_f \boldsymbol{h}_{t-1} + \vec{\boldsymbol{u}}_f \boldsymbol{x}_t) \tag{12}$$

$$\boldsymbol{i}_t = \sigma(\vec{\boldsymbol{w}}_i \boldsymbol{h}_{t-1} + \vec{\boldsymbol{u}}_i \boldsymbol{x}_t) \quad , \tag{13}$$

where $\vec{\boldsymbol{w}}_c, \vec{\boldsymbol{u}}_c, \vec{\boldsymbol{w}}_o, \vec{\boldsymbol{u}}_o, \vec{\boldsymbol{w}}_f, \vec{\boldsymbol{u}}_f, \vec{\boldsymbol{w}}_i$ *and* $\vec{\boldsymbol{u}}_i$ are all trainable weight vectors, and σ denotes a sigmoid function:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad . \tag{14}$$

LSTM has performed well in industry-leading benchmarks and is one of the most used RNN models (Patterson & Gibson 2017).

Gated Recurrent Unit (GRU) is a another RNN model that does not suffer from the vanishing gradient problem. It was proposed by Cho et al. (2014) as an alternative that is simpler to compute and implement than LSTM. Unlike LSTM, GRU exposes its memory content completely through its single update gate, which works as combined input and forget gate. Instead of deciding separately what information to remember and what to forget, GRU decides these two things at the same time. Furthermore, GRU has less parameters than LSTM

and thus is faster to train, whereas LSTM may beat GRU in performance with large training sets (Chung et al. 2014). GRU contains only one state as opposed to two that are found in LSTM. The state computation is conducted in a GRU cell as follows:

$$\boldsymbol{h}_t = (1 - \boldsymbol{z}_t) \cdot \boldsymbol{h}_{t-1} + \boldsymbol{z}_t \cdot \widetilde{\boldsymbol{h}_t} \tag{15}$$

$$\boldsymbol{z}_t = \sigma(\vec{\boldsymbol{w}}_z \boldsymbol{h}_{t-1} + \vec{\boldsymbol{u}}_z \boldsymbol{x}_t) \tag{16}$$

$$\widetilde{\boldsymbol{h}_t} = \tanh(\vec{\boldsymbol{w}}_h \boldsymbol{x}_t + \vec{\boldsymbol{u}}_h(\boldsymbol{r}_t \cdot \boldsymbol{h}_{t-1})) \tag{17}$$

$$\boldsymbol{r}_t = \sigma(\vec{\boldsymbol{w}}_r \boldsymbol{h}_{t-1} + \vec{\boldsymbol{u}}_r \boldsymbol{x}_t) \quad , \tag{18}$$

where $\vec{\boldsymbol{w}}_z, \vec{\boldsymbol{u}}_z, \vec{\boldsymbol{w}}_h, \vec{\boldsymbol{u}}_h, \vec{\boldsymbol{w}}_r,$ *and* $\vec{\boldsymbol{u}}_r$ are all trainable weight vectors.

In addition to vanishing gradients, exploding gradients can appear with RNNs as well. Pascanu et al. (2013) propose a simple yet effective solution for this problem called gradient clipping. When clipping the gradient, the norm of the gradient is limited to a certain threshold, and if exceeded then adjusted as in Equation (19 describes. This prevents excessive updates to the model parameters and keeps the optimization in control.

$$if\ \|\boldsymbol{g}\| > a, a > 0: \boldsymbol{g} \leftarrow a\frac{\boldsymbol{g}}{\|\boldsymbol{g}\|} \tag{19}$$

This method also ensures that the update is towards the gradient even if it is reduced. The threshold a is generally determined with trial and error.

3.2.3 Deep autoencoders

The previous sections have focused mainly on supervised deep learning tasks for learning a representation of the data to make accurate predictions. An alternative approach, for learning a representation of the data in a supervised manner, is to use unsupervised neural networks called deep autoencoders. An autoencoder attempts to learn a lower dimensional representation of the input data, $\boldsymbol{z} = f(\boldsymbol{x})$, that it can reconstruct back to the original dimensional representation $g(\boldsymbol{z}) = \hat{\boldsymbol{x}}$ with minimal loss $\mathcal{L}(\boldsymbol{x}, \hat{\boldsymbol{x}})$. An autoencoder consist of an encoder $\boldsymbol{f}$, bottleneck layer z, and decoder $\boldsymbol{g}$. The bottleneck structure forces the latent representation to contain informative features and properties of the input data, which is why autoencoders are utilized for feature learning and dimensionality reduction. If not prevented with sufficient bottlenecking or regularization, autoencoders may learn the direct mapping (copying input to output without extracting informative features) $\hat{\boldsymbol{x}} = \boldsymbol{x} = g(f(\boldsymbol{x}))$ which is in most cases not wanted (Goodfellow et al. 2016, p. 499). Autoencoder is called *undercomplete* when the dimensionality of the latent representation is smaller than of the input and *overcomplete* when the relation is the other way around. Typically, overfitting of autoencoders is not desirable - except when training autoencoders for anomaly detection in which case the autoencoders should fail to reconstruct the input only when encountering samples not similar to the training data (Malhotra et al. 2016).

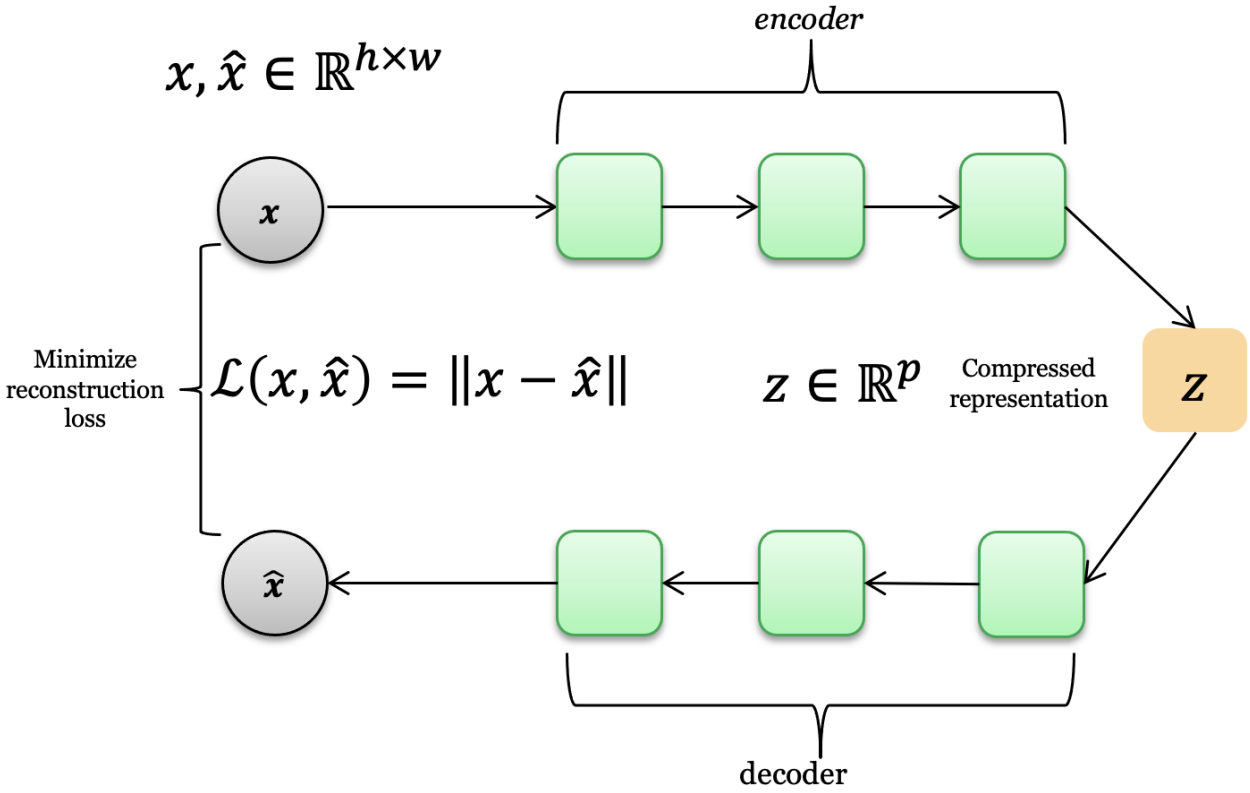

Figure 8: Basic autoencoder consisting of encoder and decoder.

Another reason for using autoencoders instead of supervised neural networks is their ability to learn features from unlabeled data, and therefore they are highly beneficial in a semi-supervised learning context (Goodfellow et al. 2016, p. 421). Semi-supervised learning refers to a machine learning task where only a small subset of samples is labeled, and a vast majority is unlabeled. This is the case in many application domains where physical measuring or manual labeling is impossible or expensive (Kingma et al. 2014), such as in virtual metrology or image recognition. In a semi-supervised setting, an autoencoder is trained with a high amount of unlabeled data, and a subsequent classifier or regressor is trained with the smaller labeled dataset.

Different variations of autoencoders, such as denoising (Vincent et al. 2008) and variational (Kingma & Welling 2014) autoencoders, have emerged to enhance the representation performance of vanilla autoencoders. A denoising autoencoder attempts to denoise an input

$$\tilde{\boldsymbol{x}} = \boldsymbol{x} + \ \boldsymbol{\varepsilon}, \boldsymbol{\varepsilon} \sim \mathcal{N}(0, \sigma^2) \tag{20}$$

by minimizing the error $\mathcal{L}(\boldsymbol{x},\ g(f(\tilde{\boldsymbol{x}})))$. As a result, it learns optimal denoising and learns the structure of $p(\boldsymbol{x})$ (Alain 2014). Denoising autoencoders are often overcomplete and have high capacity. They can be thought also as a regularized version of a vanilla autoencoder. Variational autoencoders differ from denoising and vanilla autoencoders since they do not attempt to only minimize the reconstruction error. Instead, variational autoencoders try to learn the latent probabilistic features generated by a prior distribution $z_i \sim p(\boldsymbol{z})$ that with maximum likelihood generated the data $x_i \sim p(\boldsymbol{x} \mid \boldsymbol{z})$. The encoder part attempts to

approximate the posterior distribution $p(\boldsymbol{z} \mid \boldsymbol{x})$, whereas the decoder part is trained to approximate the distribution $p(\boldsymbol{x} \mid \boldsymbol{z})$. During training this is achieved by two output heads for both encoder and decoder, where one is approximating mean and other variance for gaussian distribution. Variational autoencoders typically minimize a cost function consisting of negative Kullback-Leibler divergence

$$-\int q(z_i) \log \frac{q(z_i)}{p(z_i)}\, dz_i \tag{21}$$

and expected log-likelihood

$$\int q(z_i) \log p(x_i \mid z_i)\, dz_i. \tag{22}$$

The prior distribution is often assumed multivariate gaussian $\mathcal{N}(0, \mathbf{I})$ although other distributions have been studied as well (Partaourides & Chatzis 2017). Variational autoencoders are generative in their nature which means that they learn meaningful features from the data, and for that reason they can achieve better or comparable performance to vanilla autoencoders in most cases (Goodfellow et al. 2016, p. 502)

3.3 Time series extrinsic regression

Time series extrinsic regression is defined as a regression task to learn the relationship between a time series and a continuous scalar value that is extrinsic to the time series (Tan et al. 2021). It is similar to time series classification that has attracted lot of research interest (Bagnall, et al. 2015). Besides Tan et al. (2021), there has been very little research on specifically time series extrinsic regression models and most of the state-of-the-art methods for the task are derived or adapted from time series classification research. Time series extrinsic regression is a dominant machine learning task in industrial settings where a measurement value is predicted based on multivariate sensor data - much like is the case with virtual metrology. Other examples of time series extrinsic regression task include heart rate estimation based on sensor data (Reiss et al. 2019). This section introduces time series extrinsic regression and some of the current state-of-the-art models for the task.

3.3.1 Definition

Tan et al. (2021) defines time series regression as learning the function $\mathcal{T} \rightarrow \mathcal{R}$:

$$\mathcal{T} = \{(\mathrm{t}_0, \mathbf{s}_0), (\mathrm{t}_1, \mathbf{s}_1), \ldots, (\mathrm{t}_T, \mathbf{s}_T)\} \tag{23}$$

$$\mathcal{R} \in \mathbb{R} \quad , \tag{24}$$

where $\mathbf{s}_\mathrm{i} \in \mathbb{R}^\mathrm{D}$ are measurements and t_i are timestamps for those measurements. The measurements describe the same process and are ordered in respect to time. For each time series

$\mathcal{T}$, there is a corresponding scalar value $\mathcal{R}$ which have a relation for the regression model to learn. This definition differs substantially from a typical machine learning regression where the tabular feature space has no relation to time, and instead all samples are expected independent and identically distributed (IID). There is also a significant difference to time series forecasting where an autoregressive model is fitted on univariate scalar values to forecast the future values for the same scalar variable.

3.3.2 Classical regression models

A naïve approach - arguably the easiest - is to train a classical regression model, such as random forest, XGBoost or SVR, on a time series extrinsic regression task. These algorithms can only learn from tabular data, so the multivariate time series data is flattened to $D \times T$ feature columns. Therefore, each time series is considered a row in the tabular data set. When a model is trained in this manner, it cannot consider the temporal structure of the data and may become vulnerable to small variations in the input data. Regardless of the simplicity of this approach, it works well on time series extrinsic regression tasks and the model training times are short compared to deep learning models (Tan et al. 2021). However, in the case of variable length time series, the sequences shorter than the maximum length need to be padded with a placeholder value such as -1. This affects the model performance because the model is not any way informed of the padded values and separating them from the actual data consumes the capacity of the model.

Another method to cast the multivariate time series data to tabular format is to manually extract features, such as statistical moments as described in Section 2.3.1, from the process data (Kang et al. 2017). This approach takes into account the temporal nature of the data and variable length sequences are handled without a need for padding. However, the manually extracted features are somewhat domain specific and result in numerous features which might not describe the response variable well. Additionally, feature selection and dimensionality reduction become important for various regression models when the number of features increases.

3.3.3 Deep learning models

An advantage of neural networks is their adaptability to different data structures with minor changes on the architecture. For this reason, neural networks are utilized on diverse fields varying from NLP to image recognition. Many deep learning models for time series extrinsic regression take influence from other domains not specifically related to time series data. However, many of these algorithms have achieved great results in time series extrinsic regression benchmarks (Tan et al. 2021).

For example, RNNs work well on a task called sentiment classification (Tang 2015) where the goal is to learn to identify sentimental polarities such as positivity or negativity from a sequence of words. A typical RNN sentiment classification model consists of an embedding layer, an RNN unit, such as GRU or LSTM, and a consequent linear layer with SoftMax activation. When the sequence is embedded, it resembles time series data because the data is multidimensional and sequential. In fact, modifying the last linear layer to output scalar values and removing the SoftMax activation transforms the model compatible for time series extrinsic regression tasks (Hüsken & Stagge 2003). Yu et al. (2021) shows that such RNN models transfer to autoencoder architecture and perform well on time series classification

as well. Besides good performance, RNNs can handle variable length sequences without wasting computation power on padding values that are necessary for fixed size batches during training. The reason for this is RNNs recursive nature that allows stopping computation when the end of the sequence is reached without modifying the model architecture (Che et al. 2018).

Another popular time series extrinsic regression model, residual network (ResNet), was adopted from image recognition research. He et al. (2016) proposed ResNet in 2015 for image recognition as an improvement over other image recognition models of that time. ResNet is a convolutional neural network with skip connections that allow easier optimization for deeper networks. Skip connections prevent occurrence of the vanishing gradient problem that was described in Section 3.2.2. Wang et al. (2017) experimented ResNet's applicability on time series classification and found its performance competitive compared to state-of-the-art algorithms. The ResNet utilized in the experiments consists of 3 residual blocks followed by a global average pooling layer and linear layer. These residual blocks consist of 3 convolutional layers that all have different kernel sizes.

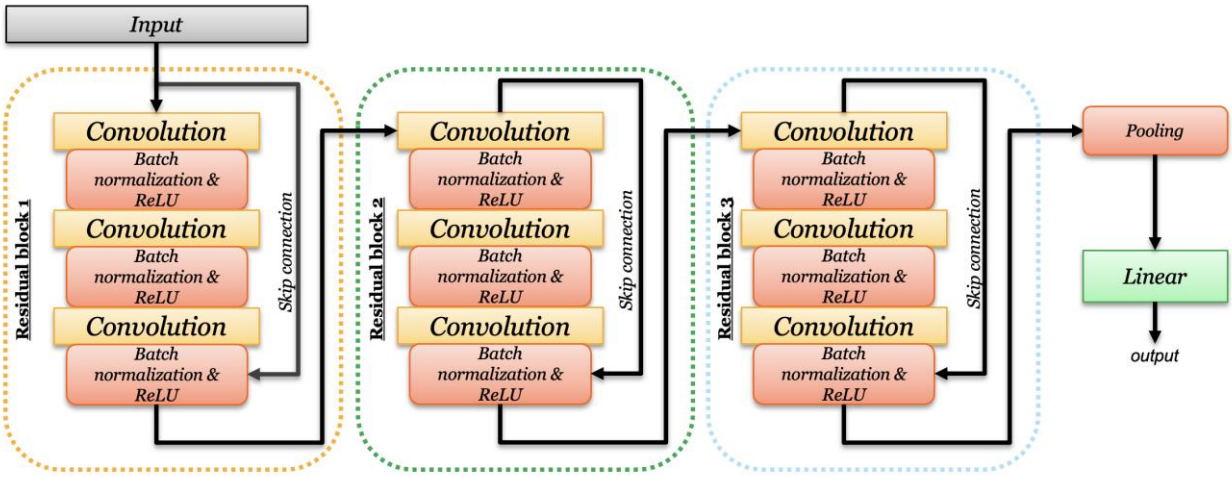

Figure 9: A ResNet with three residual blocks for time series extrinsic regression.

In addition to ResNet, Wang et al. (2017) first proposed the use of fully convolutional neural networks (FCNs) for time series classification. FCN was introduced by Sun & Wang (2018) for semantic segmentation of images. Wang et al. utilize FCN in feature extraction for time series classification and produce the predictions with a global average pooling layer and a linear layer. FCN consists of 3 convolutional layers that all have different kernel sizes and number of channels. Wang et al. state that the FCN achieves "premium performance to other state-of-the-art approaches" including ResNet. FCN and ResNet perform similarly in 20 time series extrinsic regression benchmark tasks conducted by Tan et al. (2021).

InceptionTime is another time series classification algorithm proposed by Ismail Fawaz et al. (2020) which utilizes an ensemble of five deep learning models that are created from multiple Inception modules. The original Inception model is an image recognition model that consists of multiple stacked convolutional layers and residual connections between the convolutional layers (Szegedy et al. 2017). The prediction is produced by a global average pooling layer and linear layer. Tan et al. (2021) finds in their experiments the performance of InceptionTime in time series extrinsic regression comparable to FCN and ResNet.

However, InceptionTime has much higher variation in performance in benchmark datasets than other time series extrinsic models that were experimented in the paper. Similarly to other convolutional models, InceptionTime requires padding of variable length time series data, and the model does not transfer to for example datasets with longer sequences, without changing the architecture of the model and re-training the network.

4 Data and implementation

In this chapter, we first review the CMP dataset utilized in the experiments and conduct comprehensive exploratory data analysis on it. Then, we introduce the architecture of our virtual metrology system and describe how it is deployed in production. Next, we define the standard tree-based machine learning solutions for our metrology prediction task and outline the alternative model candidates that the tree-based models are compared to. Lastly, we introduce the evaluation metrics and validation methods that allow us to compare the overall performance and generalization error of the candidate machine learning models.

4.1 Process data

The process data we use in our experiments was collected by the Prognostics and Health Management (PHM) society from a CMP process tool located in a real semiconductor manufacturing facility during approximately a two-month period. The dataset contains 25 process variables and one response variable. In total, the dataset consists of 2829 runs, which were split in the 2016 PHM Data Challenge to training, validation and testing sets with 1981, 424, and 424 runs respectively. We use precisely the same split to allow easy benchmarking and comparison to other publications and show visualization in this section based on the same split. Furthermore, we conduct model development, hyperparameter tuning, and initial testing with the validation set and validate the results with the testing set. The evaluation procedure is discussed in detail in Section 4.4.

4.1.1 Variable descriptions

The 25 process variables in the dataset are categorized to identification, status, usage, pressure, flow rate, and rotation rate types (Table 2). Identification variables are not useful as inputs in the metrology prediction task but allow separation and identification of specific process runs from each other with the combination of variable *Stage* that separates the runs of the same wafer in different process stages. The stages consist of A and B stages that are both conducted for each wafer. The wafers are processed in either chambers 1, 2, 3 or 4, 5, 6, which refer to a location inside the CMP tool. Six usage measures indicate the wear of different process tool components, which affect the tool operating condition. Another six variables contain the pressure measurements that are set according to process recipe and controlled during the process run in order to achieve good polishing results even for uneven wafer surfaces. Three variables indicate the flow rate of slurry that is dispensed on the polishing table during runs from different flow lines, and three variables describe the rotational speed of different kinetic components of the CMP tool that are set to operate according to the specifications in the process recipe.

Table 2: Process data variable descriptions and types.

Variable name	Type	Description (adopted from[9])
Machine data	Identification	Numeric ID of wafer ring location in machine
Machine ID		Numeric ID of machine (matches Chamber)
Wafer ID		Number representing ID of wafer
Chamber	Status	Chamber in CMP tool for wafer processing (123 or 456)
Dressing water status		Status of dressing water
Stage		A or B representing a different type of processing stage
Timestamp		Timestamp in seconds
Usage of backing film	Usage	A usage measure of polishing pad backing film
Usage of dresser		A usage measure of dresser that is used after runs to condition the polishing pad
Usage of dresser table		A usage measure of dresser table
Usage of membrane		A usage measure of polishing membrane
Usage of polishing table		A usage measure of polishing table
Usage of pressurized sheet		A usage measure of wafer carrier flexible sheet
Edge air bag pressure	Pressure	Pressure of bag on edge of wafer
Center air bag pressure		Pressure related to wafer placement
Main outer air bag pressure		
Retainer ring pressure		
Ripple air bag pressure		
Pressurized chamber pressure		Chamber pressure
Slurry flow line A	Flow rate	Flow rate of slurry type A
Slurry flow line B		Flow rate of slurry type C
Slurry flow line C		Flow rate of slurry type B
Head rotation	Rotation rate	Rotation rate of head
Stage rotation		Rotation rate of stage
Wafer rotation		Rotation rate of wafer
Average material removal rate	Response variable	The average rate of material removal in nm/min

The response variable or the metrology value to predict is the average material removal rate from the wafer surface in nanometers per minute (nm/min) during the CMP processing. The rate of the polishing is physically measured and consequently predicted because it is an essential metric for determining the endpoint for the process and thus accurate predictions of polishing rate enhance the quality and reliability of the polishing process.

4.1.2 Prediction task description

The prediction task that the machine learning models attempt to learn is a time series extrinsic regression task (see Section 3.3.1 for definition). This means that the machine learning solution receives the process data, described in Table 2, as input and predicts single scalar value for each single time series of process data. A single time series, called run, contains multivariate time series process data for one processed wafer. Each run is linked to a scalar metrology value that the model attempts to predict.

[9] https://phmsociety.org/conference/annual-conference-of-the-phm-society/annual-conference-of-the-prognostics-and-health-management-society-2016/phm-data-challenge-4/

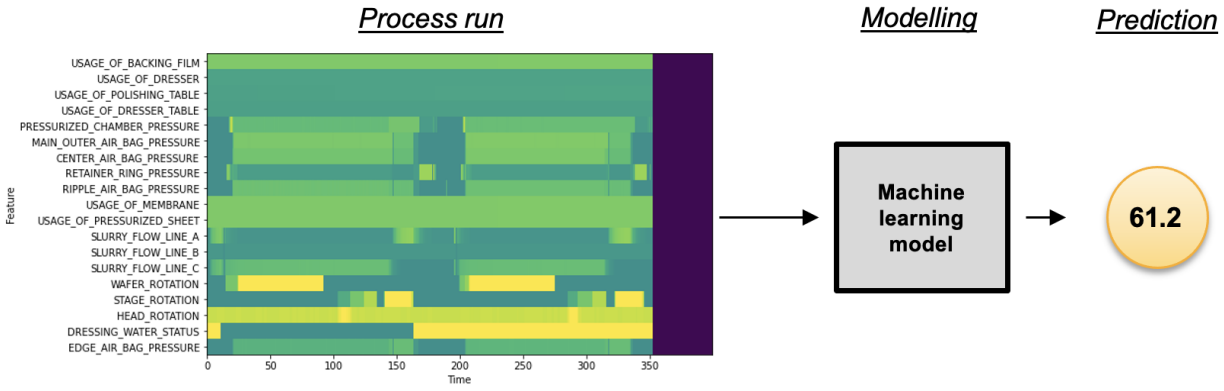

Figure 10: Visualization of the machine learning task.

Figure 10 shows a visualization of a run (a sample) that is inputted to the machine learning model and a scalar value that it outputs consequently. In the next section, we further analyze the process data and explain characteristic of the data based on statistical analysis and visualizations.

4.1.3 Exploratory analysis

The dataset consists of multivariate time series or runs with a metrology value for each of these runs. This exploratory data analysis shows the data in various formats, such as in time series and histogram format, which provides necessary information on the structure of the data. The dataset contains process runs from two different polishing modes (low/high-speed) where the process variables exhibit distinct characteristics for each mode, which is visible in Figure 11 that shows heatmaps of four individual process runs. Each pixel represents a measurement of a single variable at a timestep in the heatmap where the time is represented on the x-axis and the process variables on the y-axis. The runs in the first row of the figure are from the high-speed mode and the second-row runs are from the low-speed mode. Heatmap provides an overall illustration of the data structure and relative variance of the variables. Immediately, it is evident that the two modes are clearly separable in the heatmaps, and that the high-speed mode runs appear to have lower variance in several variables. It is also somewhat obvious that the high-speed mode runs are generally shorter than the low-speed mode runs because it takes less time to polish the wafer with high rotational speeds and hence the higher material removal rates.

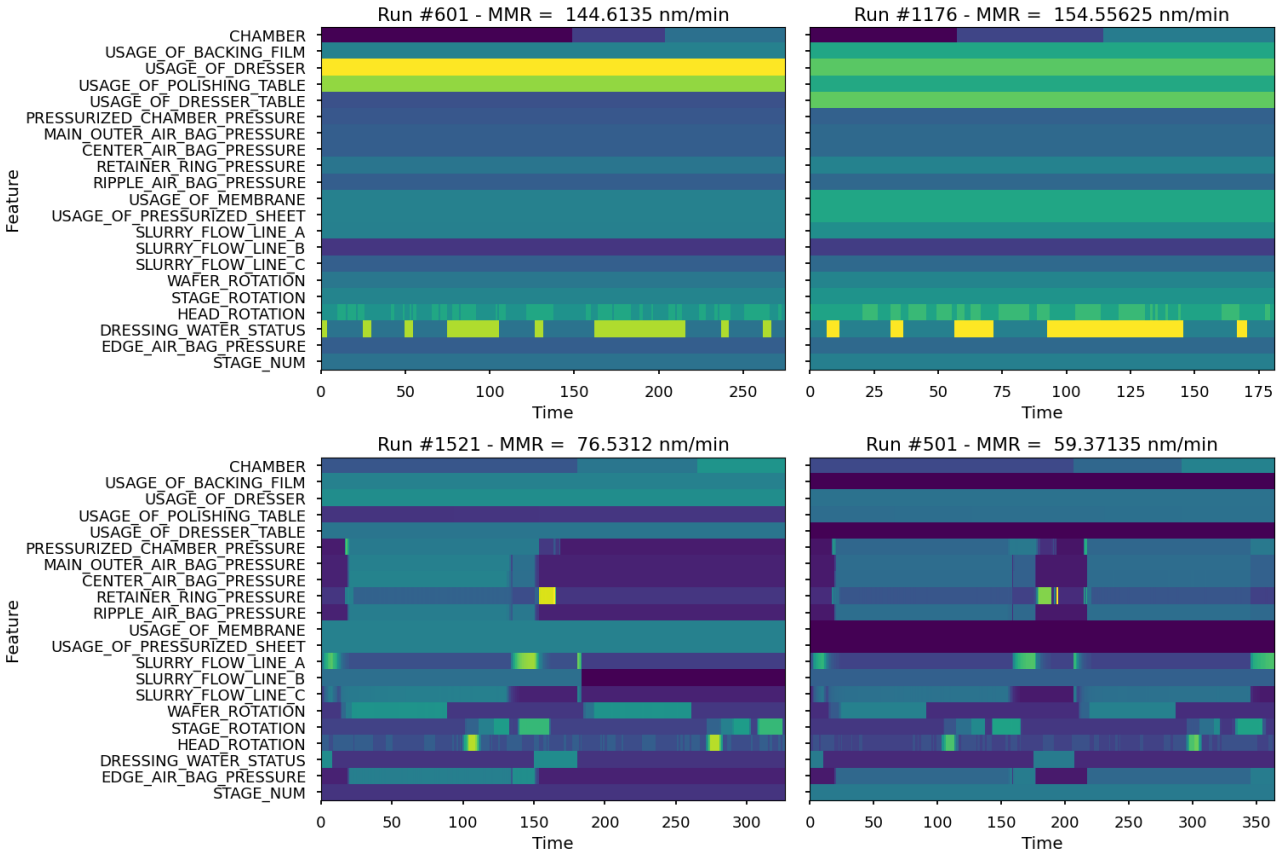

Figure 11: Heatmaps of four process runs. First row is from the high-speed and second row from the low-speed polishing mode.

Next, a histogram of the length of the runs (Figure 12) shows that the lengths are divided into high- and low-speed groups as well, and that most of the runs are around 300 timesteps long.

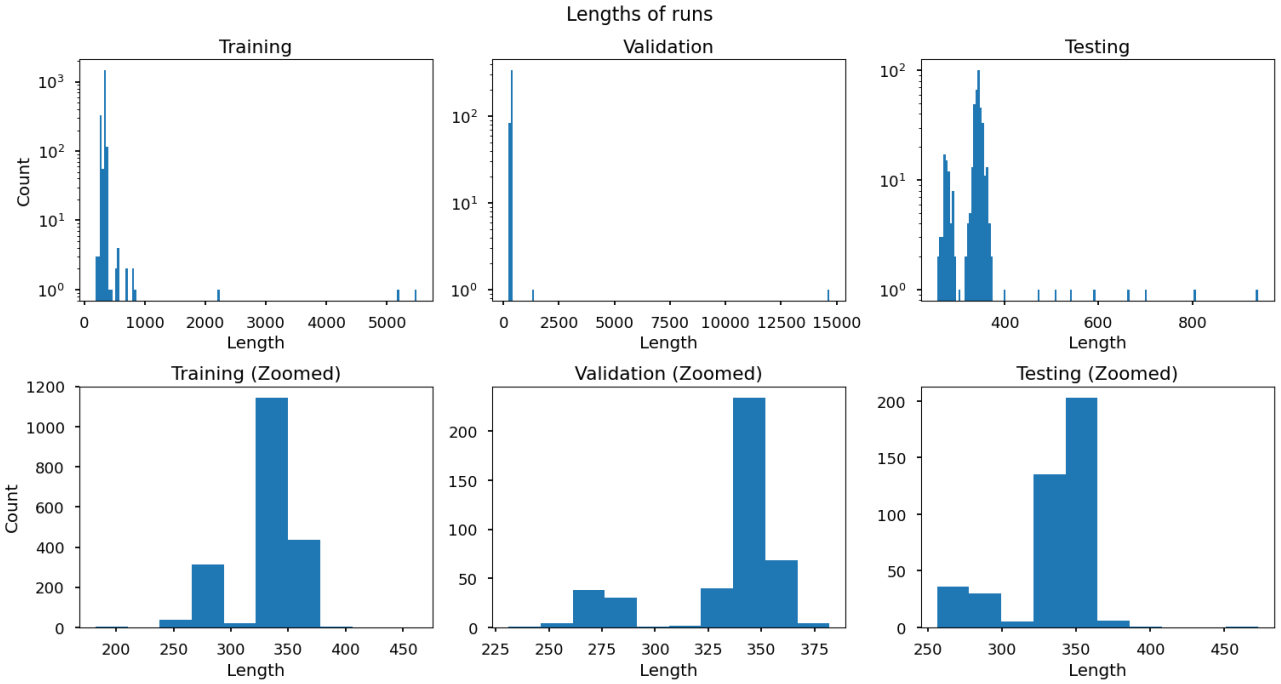

Figure 12: Histogram of process run lengths.

However, there are several outliers in all splits of the dataset which are most likely caused by process runs that have stopped but data logging has nevertheless continued. This is visible in graphs (Appendix A) that show the rotational speeds during these outlier runs. After around 300 timesteps, all the rotation variables go to zero (0) which indicates stopping of the process run. The first row of Figure 12 shows histograms of sequence lengths in the entire dataset, whereas the second row represents a zoomed-in image of the distributions in the normal run length range.

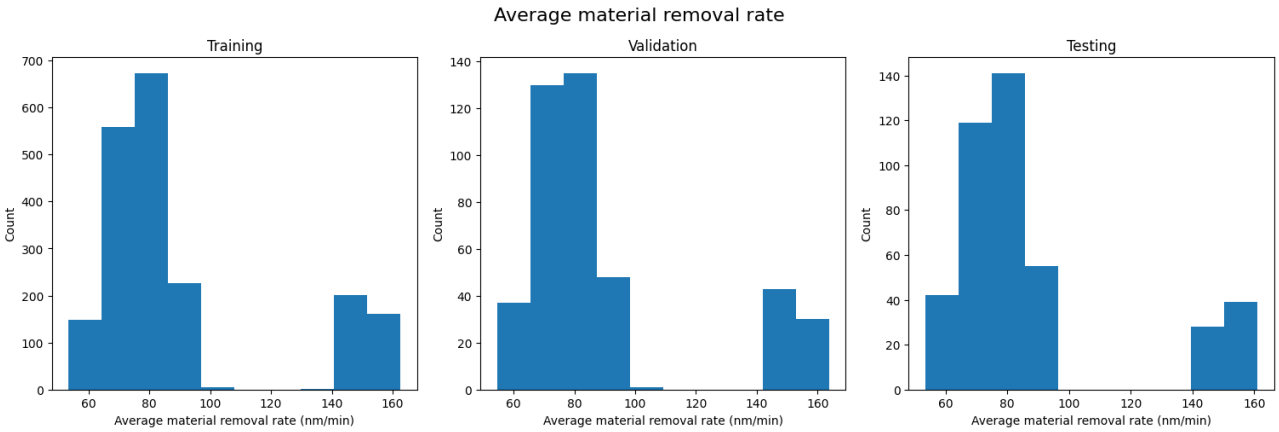

Figure 13: Histogram of average material removal rate by nm/min with outliers removed.

In Figure 13, the histogram of material removal rate is shown for all the split datasets. Four clear outlier values are omitted from the training set histogram to enable clearer visualization. The process of defining and removing outliers is discussed more in Section 4.2.1. Figure 13 shows the two separate material removal rate distributions in each split of the dataset, which are a result of the two polishing modes in the tool: high-speed and low-speed. Furthermore, the metrology distributions appear similar in all datasets, and it is clear that the high-speed polishing mode is underrepresented in the dataset compared to the low-speed mode. This may affect the difficulty of modeling the high-speed mode.

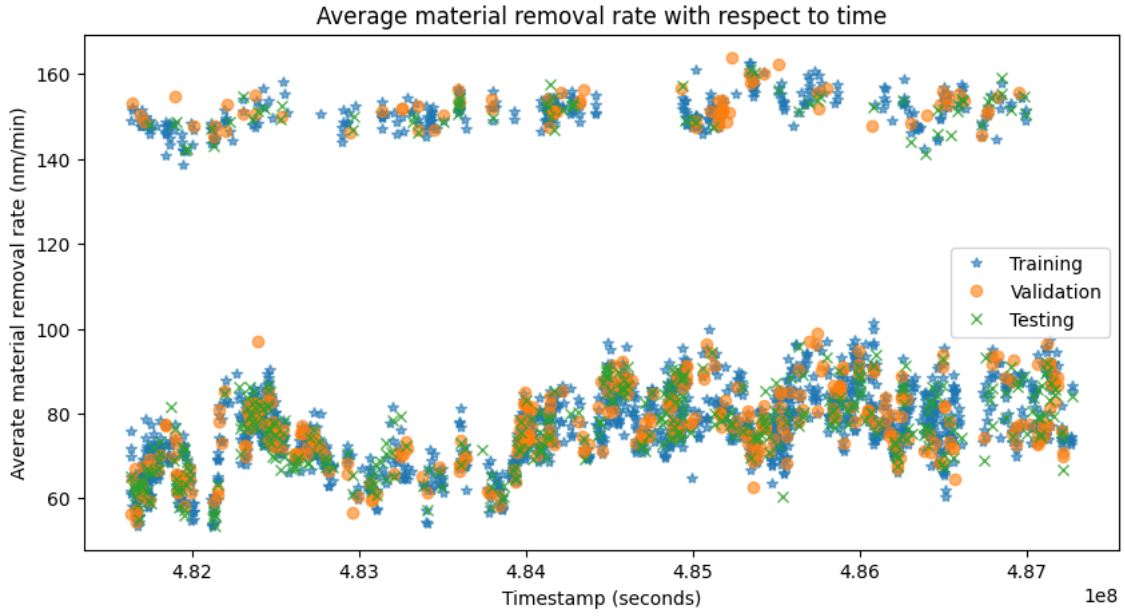

Figure 14: Average material removal rates in time series format.

It is also important to note that the average material removal rate is univariate data with time dimension, which means that it may have temporal structure itself that is not quantifiable from the other process variables. Figure 14 presents the average material removal rate measurements in a temporally ordered time series. We can see that the response variables have some temporal dependence which is expectable in industrial process data. It is also visible from the Figure 14 that the training, validation and testing dataset splits are overlapping and not split in regard to time.

The different modes of polishing speed are also apparent when dividing the data by chambers. The chamber measurements for each run vary between either 1, 2 and 3, or 4, 5, and 6, which is why we denote these as chambers *123* and *456*. Evidently, the different modes are operated with different chamber with different process recipes. Figure 15 shows this in a boxplot where the average material removal rate is divided discretely by the chamber to show the two different distributions that are apparent in the dataset. The standard deviations in these distributions are small and there is no significant skew. Some outlier values appear outside the whiskers of the boxplot, which is to be expected.

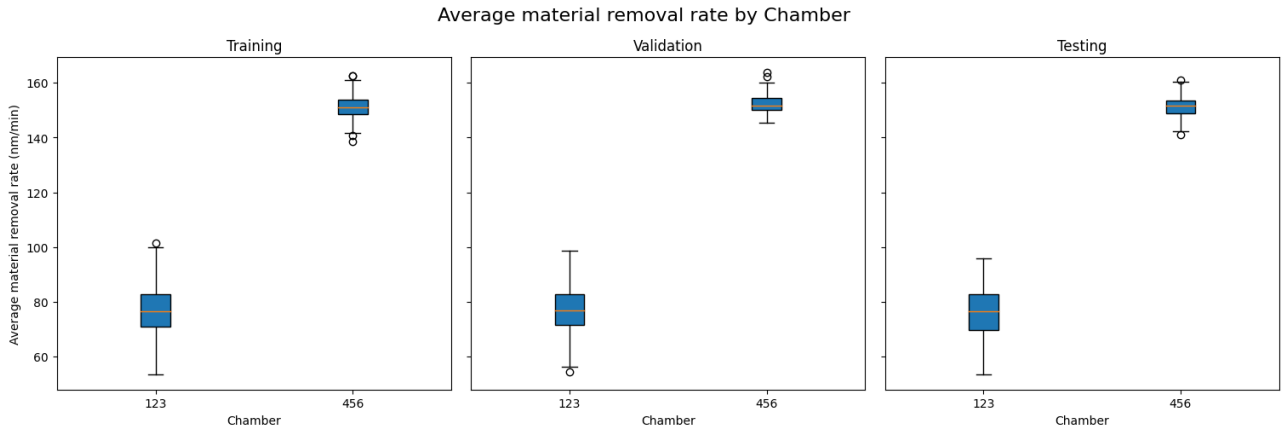

Figure 15: Boxplot of the average material removal rate by the two different chambers.

The structure of the data is rather complex and manual feature engineering for CMP requires lot of extra manhours. In addition, the data contains outliers, erroneously logged measurement values and missing values which set demanding standards for the virtual metrology system that handles the data. The next section covers the implemented architecture of our virtual metrology system that is utilized in facilitating different predictive machine learning models. The virtual metrology system enables successful and stable system operation in an industrial environment.

4.2 Virtual metrology system architecture

The design of the virtual metrology system aims to maximize accuracy and reliability of the metrology predictions. This results to finer control and monitoring of the process in wafer-to-wafer basis which leads to higher yield and production output in the end. Our virtual metrology system consists of data preparation, modeling, and deployment modules. During the data preparation, the data is gathered, stored, and preprocessed in a format that the machine learning models can utilize. This is an important step because the data from a semiconductor process tool is usually low-quality containing outliers, missing values, and redundant variables. After data preparation, a model is trained and evaluated before it is deployed into

production. Production model quality is crucial because bad virtual metrology predictions can cause unnecessary interruptions and unnecessary loss of wafers in the production line.

The architecture of the metrology system is shown in Figure 16 where components of data preparation, modeling and deployment modules are included. Next, we introduce these components of the pipeline in detail and describe the implementational aspects of them. Later, we further review the modeling methodologies utilized in the system.

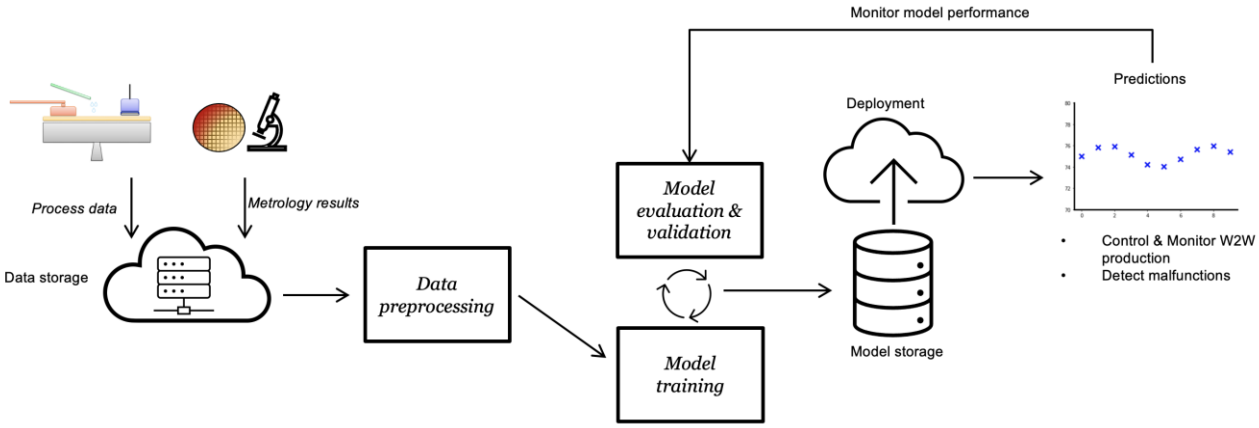

Figure 16: The architecture of our virtual metrology system.

4.2.1 Data preparation

Before the data is utilized in virtual metrology, it needs to be prepared accordingly. The preparation begins with combining the process data from the CMP tool to the physical metrology results that are obtained only after measuring the wafer. This is conducted by joining the datasets with the necessary identifiers - in this case with the *Wafer ID* and *Stage* variables. After the mapping is done, the matched data is stored in a database where it is accessible for the data preprocessing component. The raw PHM data is initially provided in several files with average material removal rate measurement results in separate files. This reflects the real-life virtual metrology scenario where the process tool and physical metrology generate the flows of data that are utilized in combination ultimately.

Preprocessing of the data begins with encoding of variables to a format interpretable to the machine learning models. Firstly, the *Chamber* is encoded to 0/1 variable where 0 represents wafer that was processed in chamber 1,2, and 3, whereas 1 represents wafer processed in chamber 4, 5, and 6. This encoding procedure results to one exogenous variable not varying in time during individual runs. Autoencoder models utilize the exogenous variables in conjunction with the latent vectors in the virtual metrology task, whereas in the supervised deep learning models, the exogenous variables are not separated from the rest of the time-varying data but rather retained with the rest of the time-varying variables. Furthermore, variables such as *Stage* that are in char format are converted to integers. Encoded stage is utilized in the same manner in different models as the chamber encodings.

Semiconductor manufacturing tool data often contains several variables, such as additional identifiers or status variables, that have no variance or significance in virtual metrology. These are removed in this stage of preprocessing to remove redundant data that is further processed and utilized in modelling the process. Next, all the measurements inside the runs are ordered in time and truncated to length 400. We chose 400 as the maximum length

because most of the runs are shorter than that and mostly outlier runs succeed it, which is visible in Figure 12 that shows a histogram of the run lengths. The maximum sequence length affects the model architectures and longer sequences cause increased computational complexity, which is why truncating sequences to reasonable length is justifiable. This is a common and accepted procedure in the field of machine learning (Goodfellow 2016).

Missing values are common in industrial data, and therefore our preprocessing component is prepared to handle missing values by either removal or imputation based on the amount of missing values. After all the missing data is handled, the data is stored in tensor format by padding the sequences that are shorter than the maximum sequence length (400 timesteps) with Nan-values which results to sequences with equal array size. This is a procedure that allows to store the data in a single tensor and train the neural networks. Later in the pipeline, the Nan-values are replaced with a padding value such as (0 or -1) which indicates to the model that the sequence has ended at that time point.

Physical metrology is typically a slow but reliable method to determine properties of the wafer that was processed. However, sometimes, the metrology tools fail, and erroneous data is stored in the database. In the PHM dataset, there are four metrology outputs that exceed the normal operating range by a factor of 40. These overshooting values outliers are removed automatically based on Z-score, which is defined as

$$z = \frac{y - \bar{y}}{S_y} \quad , \qquad (25)$$

where y is a sample of measured average material removal rate, $\bar{y}$ is sample mean, and S_y is the sample standard deviation. All samples with a Z-score higher than 3.0 are removed, which in the training dataset corresponds to an average material removal rate higher than 660.77 nm/min. With this threshold, four outlier samples are omitted from the training dataset so that the model training would not suffer from them. Figure 17 shows these physical metrology outliers in a histogram.

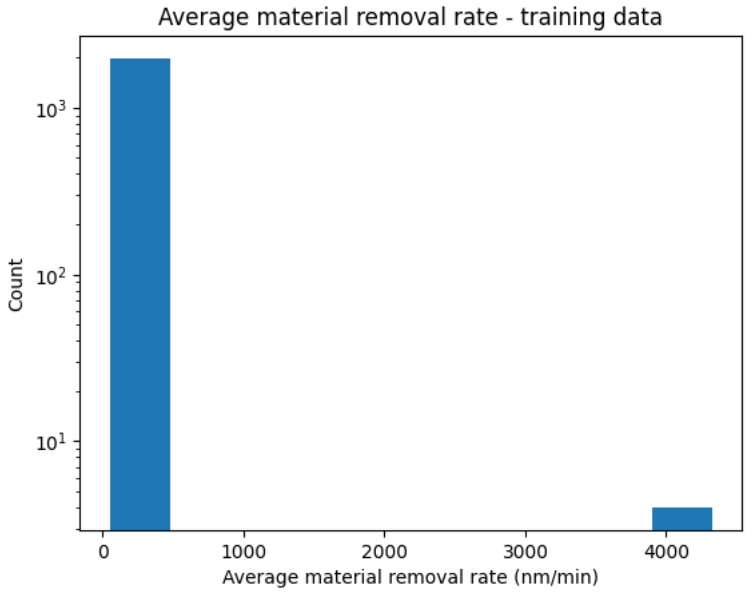

Figure 17: Histogram of training data set outliers in physical metrology results.

Many machine learning algorithms such as neural networks require data that is scaled to normalized scale in order to learn useful features from the data. Unscaled data can result in exploding gradients and instability of learning. Hence, we scale the process data of the runs sensor-wise to zero (0) mean and unit (1) variance. Sensor-wise scaling is utilized in most of the virtual metrology publications, and it is preferred because it retains the temporal structure of a single run but scales the runs and process variables independently to the same scale. Standardization to 0-1 range was also trialed but it did not affect the learning results significantly when compared to the chosen 0-mean-unit-variance scaling. The scaled data is computed as:

$$X_{sensor\_i\_scaled} = \frac{X_{sensor\_i} - \bar{X}_{sensor\_i}}{S_{X\_sensor\_i}} \qquad \forall i \in 0,1,\dots,D \quad , \qquad (26)$$

where X is the original data with D variables and T timesteps a run. The scaling is conducted when the data still contains Nan-values inputted earlier, but after scaling the Nan-values are replaced with a padding value (-1). The Nan-values are replaced only after the scaling so that the padding values do not affect the scaling by changing the mean and standard deviation of shorter runs that are padded.

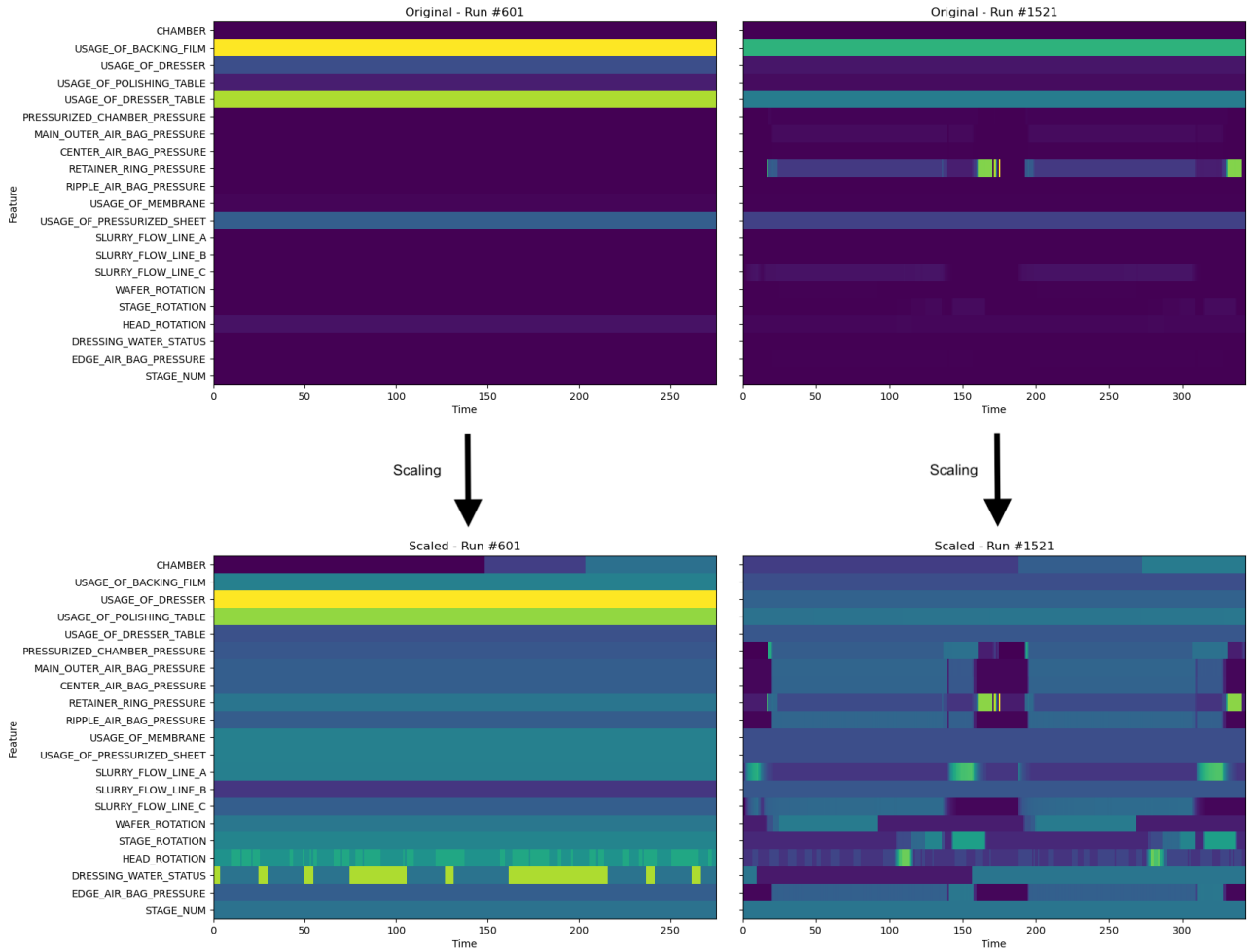

Figure 18: Visualization of two example process runs before and after conducting the scaling as described in Equation (26.

Looking at Figure 18, we can see that the described scaling procedure brings visible the characteristics of single runs in the visualization because the sensors then vary in the same scale, and thus variation in a sensor inside a run becomes visible more clearly. This scaling procedure thus retains the temporal structure of single variable time series and allows neural networks to learn useful features from the data.

We compute the mean and standard deviation of process data for scaling only from the training data and utilize it in scaling both validation and testing datasets. This practice prevents undesirable peaking to the parts of the dataset that are utilized in evaluating the performance of the model, and thus eventually leads to better modeling of the regression task. This is a good practice, even if in practice the training, validation and testing dataset rarely have drastically different mean and standard deviations.

After the data is preprocessed, it is in a format that the neural networks and other machine learning models can utilize efficiently. The preprocessing is conducted on all new data, and the preprocessed data is saved to a permanent storage to avoid unnecessary re-computation. Next, we review the modeling and deployment module that supersedes the data preparation module.

4.2.2 Modelling and deployment

In the modelling module, the machine learning model for the virtual metrology is trained on the prepared data. The model is trained and tuned with the validation dataset split from the entire dataset and then the modeling results are validated with the testing dataset. Only a model that achieves similar validation and testing performance is deployed. Training time of the model needs to be reasonable as the computational resources in semiconductor factories are typically limited. After the model is trained, it is stored in a compact format to a model database with a unique training id that allows other services or modules to utilize it later.

Deployment of the model begins with fetching the trained model from the model database and deploying it to use through an inference pipeline. When the model is in use, it predicts the metrology results for every wafer and the predictions are compared to the available physical metrology results. If the model accuracy fluctuates or is insufficient, re-training of the model is initiated. The predictions are utilized in monitoring the process behavior through all processed wafers which means that malfunctions of the process tool are detected early. Additionally, the predictions enable controlling of the process run parameters on wafer-to-wafer basis rather than only lot-to-lot basis.

4.3 Machine learning models

The choice of the machine learning model is a fundamental part of designing a virtual metrology system. The choice of the model affects multiple design aspects of the virtual metrology pipeline, such as the preprocessing and dimensionality reduction. Hence, we propose, experiment, and compare several deep learning model candidates for use in the CMP virtual metrology system. These models include convolutional, recurrent and MLP neural networks as well as autoencoder-based semi-supervised neural network models. In order to benchmark these models, we define two standard tree-based model approaches that we consider the obvious and simple choices for conducting virtual metrology. In this section, we review the design and implementation of all model candidates.

4.3.1 Tree-based

The first tree-based model approach - called raw approach - consists of a tree ensemble model, either random forest or XGBoost regressor, that receives the flattened process run data with padded values included as input. This results to 8 400 variables for each sample with the maximum sequence length of 400 and 21 original variables in the dataset. We use the vanilla implementation of random forest with tree maximum depth of 100 from the latest Scikit-learn[10] library version 0.24.2, and XGBoost[11] Python library version 1.4.2 with GPU-acceleration enabled and number of estimators set to 100. Tree ensemble regressors are typically utilized for tabular data regression, but with flattened time series data, they have achieved surprisingly good results in time series extrinsic regression benchmarks (Tan et al. 2021). For this reason, the raw approach sets a good baseline for more complex approaches that require more effort in feature engineering or model tuning.

The second tree-based model approach – called *Hand-crafted-features approach* – is an approach that is typical to a virtual metrology system in academic publications. The fundamental difference to the raw approach is that the time series data is first converted to tabular data by hand-crafted features. The statistical moments this approach utilizes are mean, standard deviation, median, minimum, maximum, kurtosis, and skew from the time-domain. Furthermore, three exogenous variables, *Chamber-123, Chamber-456* and *Stage*, without time variance are extracted from the time series data before aggregation. This results to

$$D_{variables} \times F_{statistical\ moments} = 19 \times 8 + 3 = 155 \quad (27)$$

variables in total. Typically, after aggregation, virtual metrology systems reduce the dimensionality of the data with either a feature selection or feature extraction method. PCA-based dimensionality reduction performed worse than feature selection methods in our initial tests, and hence we chose to utilize 120 best variables based on the F-score that was computed from the Pearson correlation coefficient between the average material removal rate and input variable. Only 120 best variables were chosen based on empirical testing to filter the worst input variables from reaching the machine learning model – not because tree ensemble regressor could not handle high number of variables successfully. For this hand-crafted-features approach, we chose to utilize random forest and XGBoost regressor as they are the state-of-the-art algorithms for this type of tabular data in both general machine learning and virtual metrology. The model implementations are the same as the ones utilized in the first tree-based approach with the exception of random forest having a maximum depth of 10 for a tree instead. Another difference to the first tree-based approach is that the varying-length runs do not require padding because of the aggregation step that casts the data to tabular data format.

[10] https://scikit-learn.org/stable/about.html
[11] https://xgboost.readthedocs.io/en/latest/index.html

4.3.2 Supervised deep learning

Supervised deep learning models that we benchmark to the two tree-based approaches are various RNN models, ResNet, FCN, and InceptionTime. The RNN models are implemented with PyTorch library version 1.8.1[12], whereas the other ones with tsai[13] library that is a collection of state-of-the-art time series deep learning models implemented with PyTorch. All of the deep learning models are trained on Microsoft Azure Machine learning compute instance with a NVIDIA Tesla K80 GPU.

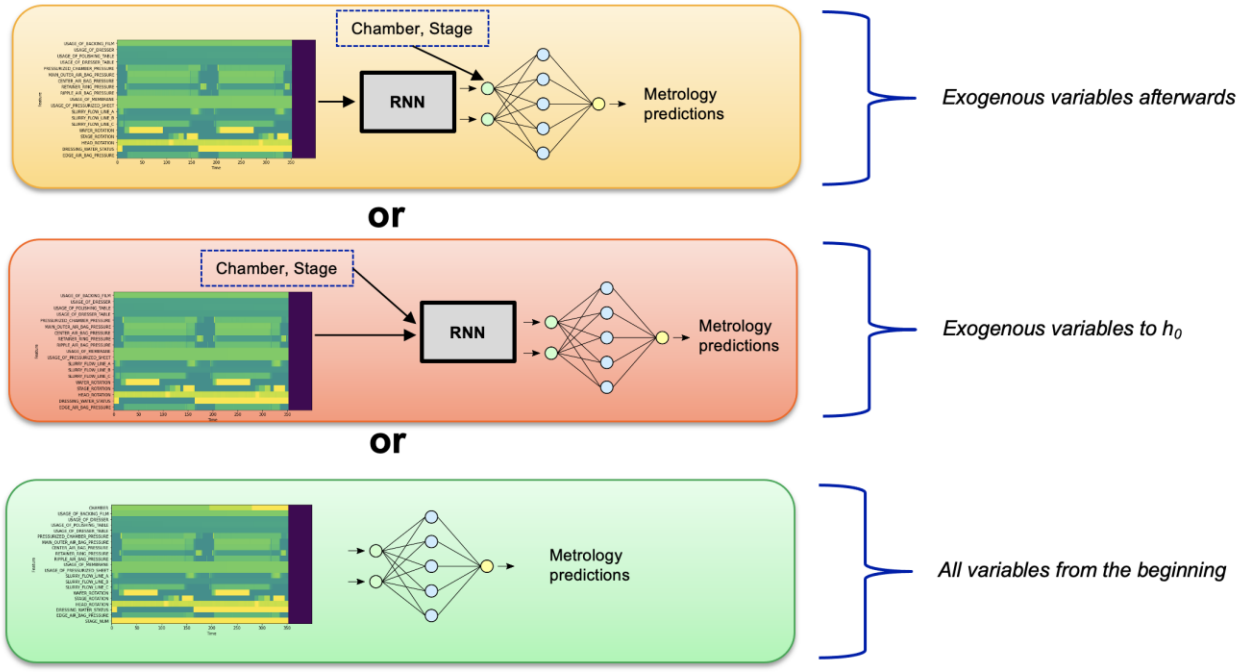

Figure 19: Different approaches for including exogenous variables in the RNN model.

We benchmark both LSTM and GRU based RNN models with three different approaches (Figure 19). The first approach consists of a RNN cell which last hidden state is concatenated with the three exogenous variables, and that concatenation is fed to a two-layer MLP network. In the second approach, the initial hidden state of the RNN cell is initialized and filled with repetitions of the exogenous variables, and a single linear layer is fed with the last hidden state of the RNN cell. The third approach is the simplest as the exogenous variables are just left in the sequential data regardless of their invariability during runs. Then, the last hidden state of the RNN is fed to a single linear layer which outputs the predictions. The hidden size of the RNN cell for all the models is 150. We utilize the PyTorch's built-in mechanism, pack_padded_sequence[14], which causes the PyTorch implementation of RNN cell to stop the computation of the sequence when the process run ends without considering the padding values described in Section 4.2.1. However, we conduct experiments on this functionality and report on how the use of it affects the model's performance and training time.

[12] https://pytorch.org/
[13] https://github.com/timeseriesAI/tsai
[14] https://pytorch.org/docs/stable/generated/torch.nn.utils.rnn.pack_padded_sequence.html

The ResNet implementation we use in our tests is from the tsai library, and it is realized from the proposal of Wang et al. (2017). The model consists of 3 residual blocks with three convolutional layers (with 7, 5 and 3 as kernel size) each as described in Section 3.3.3. Contradict to the RNN models, the ResNet is not informed of the real lengths of the sequences, but rather it computes the padded values as if they were actual measurements. This means that the model needs to learn the significance of the padded values, and changes to the data sequence maximum length affects the model architecture and thus necessitates model retraining. The source code for the model is available on tsai's GitHub repository[15].

The FCN we utilize in our tests is implemented as described in Wang et al. (2017) with three convolutional blocks that all have different kernel sizes (7, 5, and 3) and number of output channels (128, 256, and 128). The implementation of the FCN is similar to ResNet but lacks the residual connections. The source code for the model is available from[16].

Another supervised deep learning model we test for CMP virtual metrology is InceptionTime that was also introduced in Section 3.3.3. We utilize the tsai implementation of InceptionTime[17] that is adopted from Ismail Fawaz et al. (2020). As with the other deep learning models in our tests, the InceptionTime is adopted to regression task by excluding the activation function that would otherwise follow the last linear layer of network. Furthermore, the architecture of InceptionTime is dependent on the maximum length of sequence in the dataset.

4.3.3 Autoencoders and semi-supervised approach

We review the performance of several deep autoencoder types for the virtual metrology of CMP. Firstly, we develop a *vanilla multi-layer perceptron autoencoder* (MLP-AE) that consists of 4 encoding and 4 decoding layers and utilize the latent representation in the regression task with a random forest regressor (Figure 20).

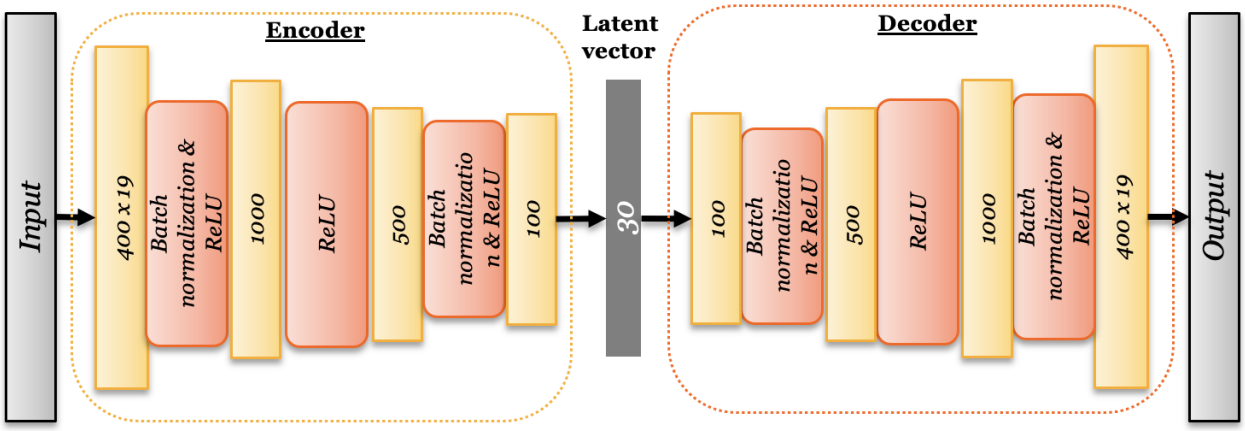

Figure 20: Outline of MLP-AE architecture.

The encoder part of the network gets the padded time-varying process data as input and compresses it to a latent representation of size 30, and that latent representation and

[15] https://github.com/timeseriesAI/tsai/blob/main/tsai/models/ResNet.py
[16] https://github.com/timeseriesAI/tsai/blob/main/tsai/models/FCN.py
[17] https://github.com/timeseriesAI/tsai/blob/main/tsai/models/InceptionTime.py

exogenous variables are then utilized in the subsequent virtual metrology task. During training, the decoder part of the network attempts to reconstruct the time series process data and minimize the MSE. The MSE is computed without including padding values in the computation because reconstruction of the padded values would only consume the capacity of the autoencoder.

Another autoencoder we experiment on is vanilla *RNN autoencoder* (GRU/LSTM-AE) with either a GRU or LSTM cell. During encoding, the RNN cell first compresses the non-padded variable-length time series data into its last hidden state with size 50 which is the latent vector that is utilized in the subsequent virtual metrology prediction. These parameters performed the best in our initial hyperparameter tuning with the validation dataset. Furthermore, it is worth noting that by default the RNN cell is informed of the time series lengths similarly as the supervised RNN model introduced earlier. During decoding, the latent representation is fed to the RNN cell each time step and the output of the RNN cell reconstructs the prediction of the original process data directly by minimizing the MSE error. To avoid exploding gradients, the gradients are clipped by a 0.25 threshold as described in Equation (19 with PyTorch built-in function called *clip_grad_norm_*[18].

In addition to the vanilla autoencoder, we develop a *variational autoencoder* (VAE) model which consists of an encoder and decoder, each with 3 layers and 2 heads. During training, the network minimizes the sum of negative KL-divergence and log-likelihood of the reconstruction. Similarly to MLP-AE, the design of VAE is dependent on the maximum sequence length in the dataset and its complexity increases as the maximum sequence length increases.

An interesting feature of autoencoders is their ability to learn features from the unlabeled data. Therefore, our autoencoder models utilize all of the process data available regardless of if it was labeled or not. We call this the semi-supervised approach where we train the autoencoder with abundant unlabeled data and the subsequent regression model with scarce labeled data. In theory, this improves performance as opposed to just training a supervised deep learning model with a small labeled dataset. We experiment how the amount and ratio of labeled and unlabeled data affects the learning results of different model candidate. We consider this as an important aspect of virtual metrology system because acquiring labels with physical metrology is fundamentally expensive and time-consuming in semiconductor manufacturing.

4.4 Validation and evaluation

We evaluate the performance of our candidate models with the validation and testing datasets described earlier. The data is from the same time interval for all of the split datasets and therefore there is no time-related distribution change between training, validation and testing sets (Figure 14). This is representative because in manufacturing the actual metrology is not conducted on every wafer, and therefore the testing and validation sets provide us ground truth for those wafers that are not normally measured. Furthermore, it is worth noting that physical metrology is only conducted approximately on every 25th wafer which means that the ratio of labeled to unlabeled data is different from the ratio of training to validation and testing set. For this reason, we experiment on the model performance on

[18] https://pytorch.org/docs/stable/generated/torch.nn.utils.clip_grad_norm_.html

different amount of labeled data and show which models perform the best with a realistic labeled to unlabeled data ratio.

We use RMSE error as the main metric in performance evaluation and show residual plots of the predictions in a format that is common in virtual metrology publications. Training time of the models is another important metric that we want to assess because in semiconductor manufacturing the training times of models cannot prolong endlessly as the production is continuous. Finally, we analyze feature importance of our tree ensemble models to evaluate variables that affect their predictions the most.

5 Results

This chapter presents the results of our virtual metrology experiments and discusses them in detail. First, we show a general overview of the model candidates' performance on the CMP virtual metrology task. Secondly, we present results for various RNN model approaches and compare them to formulate a recommended approach for including exogenous variables to RNN models. Thirdly, we analyze the performance of the best model of each approach type in a limited labeled data setting, which reflects the realistic labeled to unlabeled data ratio in the operation environment of the virtual metrology. Lastly, we conduct feature importance analysis on the raw approach tree models to review their operation and performance. This analysis presents us variables and factors that affect the virtual metrology predictions the most in the raw approach model types.

5.1 General overview

A general performance test is conducted on every model candidate where each model is trained with the training dataset consisting of 1977 wafers and tuned with the validation dataset. The model performance is evaluated with the testing dataset only after the development of the model is seized. This provides us an unbiased estimate of the generalization error of the model.

Table 3 presents RMSE and R^2 for each model with the training time included grouped by the model type. All the models converged well, but we acknowledge that the model performance varies slightly from training to training due to the indeterministic nature of the optimization algorithms and the models. Despite this, the Table 3 shows performance of the models on a single training instance.

Table 3: Performance of different virtual metrology models.

Type	Method	No. inputs / sample	Training time (sec)	Validation		Testing	
				RMSE	R^2	RMSE	R^2
Raw	Random forest	8400	377	3.80	0.98	3.58	0.98
	XGBoost	8400	24	3.85	0.98	3.50	0.99
Hand-crafted	Random forest	155	7	**3.52**	0.98	3.42	0.98
	XGBoost	155	1	3.64	0.98	**3.36**	0.99
Supervised deep learning	LSTM	8400	257	4.37	0.98	4.46	0.98
	GRU	8400	245	4.11	0.98	3.98	0.98
	ResNet	8400	230	3.67	0.98	3.90	0.98
	FCN	8400	213	3.66	0.98	3.93	0.98
	InceptionTime	8400	401	3.79	0.98	3.93	0.98
Autoencoder	MLP-AE	7602	144	4.47	0.98	4.20	0.98
	MLP-VAE	7602	231	4.10	0.98	3.82	0.98
	LSTM-AE	7602	576	4.34	0.98	4.26	0.98
	GRU-AE	7602	580	4.37	0.98	4.10	0.98

Based on the R^2 coefficient, both tree-based approaches (raw and hand-crafted model types) perform sufficiently well when trained with the entire training dataset. This suggests that prediction of the CMP tool's average material removal rate is not a particularly difficult task for tree ensemble models. Both raw and hand-crafted approach models perform slightly

better in the testing than in the validation dataset[19]. The training time of the hand-crafted approaches is short because of the relatively small feature space compared to the raw approaches that consider each measurement of each variable as a feature. The training time of both tree-based model approaches could further decrease with the use of GPU in training.

The supervised deep learning methods do not outperform the tree-based model approaches in general. Surprisingly, the supervised RNN based models that are informed of the varying sequence lengths are performing the worst and are slow to train. The RNN models perform equally well for both validation and testing set which could indicate that the RNN architecture models the phenomenon of CMP well but lacks absolute accuracy compared to other models. The most accurate model of the supervised deep learning approaches is ResNet which performs substantially well in both validation and testing sets. FCN performs similarly to ResNet which is expected as the model architectures share similar components. Both of these CNN based models are fast to train because of good computational parallelization of convolutional layers with GPU.

The autoencoder based models are trained with the unlabeled process data and the learned latent representation is utilized in the subsequent predictions task that is conducted with a random forest regressor. We can notice that the autoencoder models perform by average worse than other models as can be seen in the

Table 3. This is expectable because the training of autoencoders aims to learn a representative latent space of the original process data which might miss some of the information necessary to predict the material removal rate accurately. Therefore, the use of autoencoders on a fully supervised machine learning task as opposed to supervised deep learning models is not advisable based on the results. Later, we compare the performance of the models with limited labeled process data where the use of autoencoders is arguably more justified. The RNN based autoencoders are the slowest to train out of all the model candidates, whereas the MLP based autoencoders are relatively fast to train even compared to CNN based models.

Next, we further examine the virtual metrology performance of the models with the smallest prediction error from each group (raw, hand-crafted, supervised deep learning, autoencoder). We illustrate the models' performance with three visualizations of the residuals that are typically used in virtual metrology publications. These visualizations include a residual plot (true vs. predicted), residual histogram, and time series of true and predicted values. The predictions are shown for the testing dataset because it reflects the unbiased real-life performance of the models better than for example the validation dataset.

[19] NOTE: Commonly, the machine learning model error is lower for the validation set than for the testing set due to hyperparameter tuning, but unexpectedly the raw approaches perform better in the testing set. This could be due to a random split where the validation set contains, for example, more outliers than the testing set.

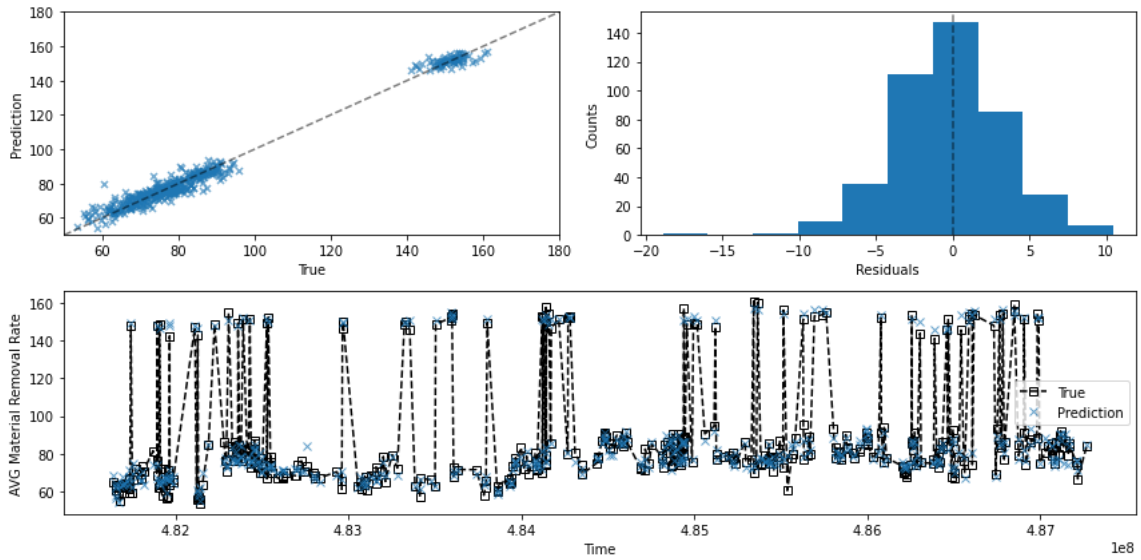

Figure 21: Raw approach with XGBoost regressor. The illustration shows performance on unseen testing dataset.

Figure 21 illustrates the performance of the raw approach with XGBoost as the regressor. We notice that there is one prediction with a high error around 60 nm/min material removal rate that was yet predicted 80 nm/min. Otherwise, the predictions are accurate, and residuals are distributed evenly around zero. The time series plot shows no signs of fluctuated performance around any specific time.

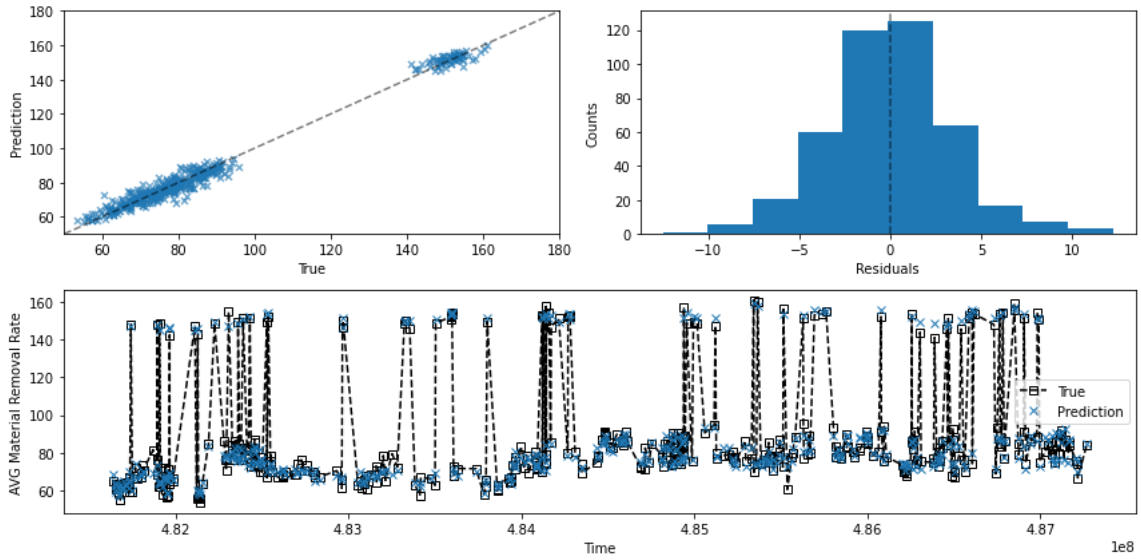

Figure 22: Hand-crafted approach with XGBoost regressor.

The performance of the hand-crafted approach with XGBoost appears comparable to the raw approach as is visible in Figure 22. The low-speed mode is more difficult to model as expected and there are some predictions with significant error (maximum error of slightly over 10 nm/min). The residuals are normally distributed around 0 mean and the model performs equally well through the time period of the testing dataset.

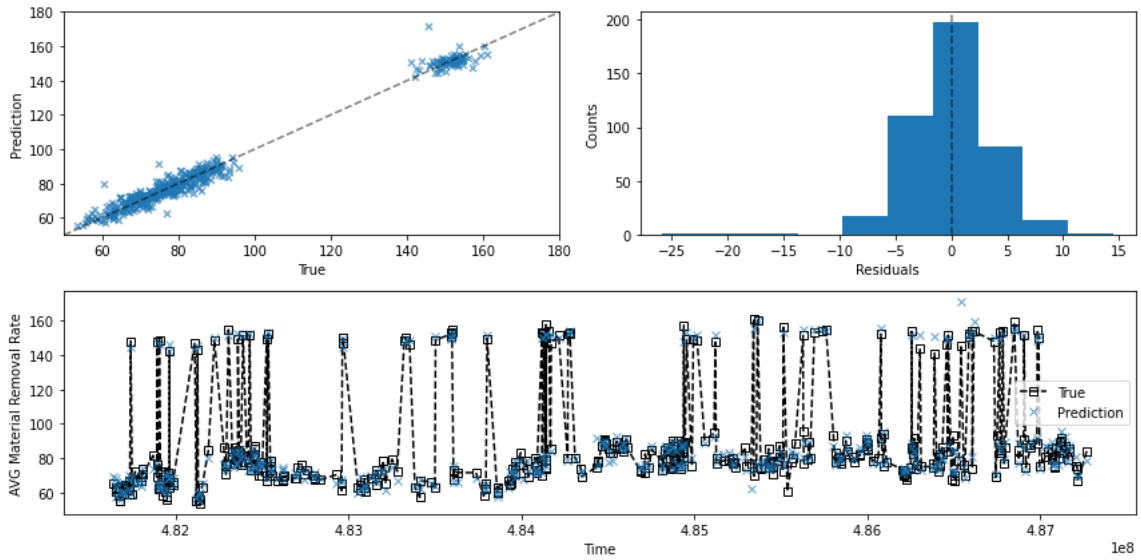

Figure 23: ResNet performs best in the testing dataset out of the supervised deep learning models.

ResNet performs slightly worse compared to both tree-based approaches. This is visible in Figure 23, which shows visualizations of ResNet's predictions on the testing dataset. There are a few predictions with high error in both low- and high-speed modes, and the maximum error reaches 25 nm/min which could affect tool operation in an undesirable way. The highest residuals are somewhat skewed to the left side of the distribution meaning that the average removal rate is predicted too high. From the time series plot we notice that the model performance fluctuates significantly between timestamps 4.68 and 4.87 (x 10^8 sec).

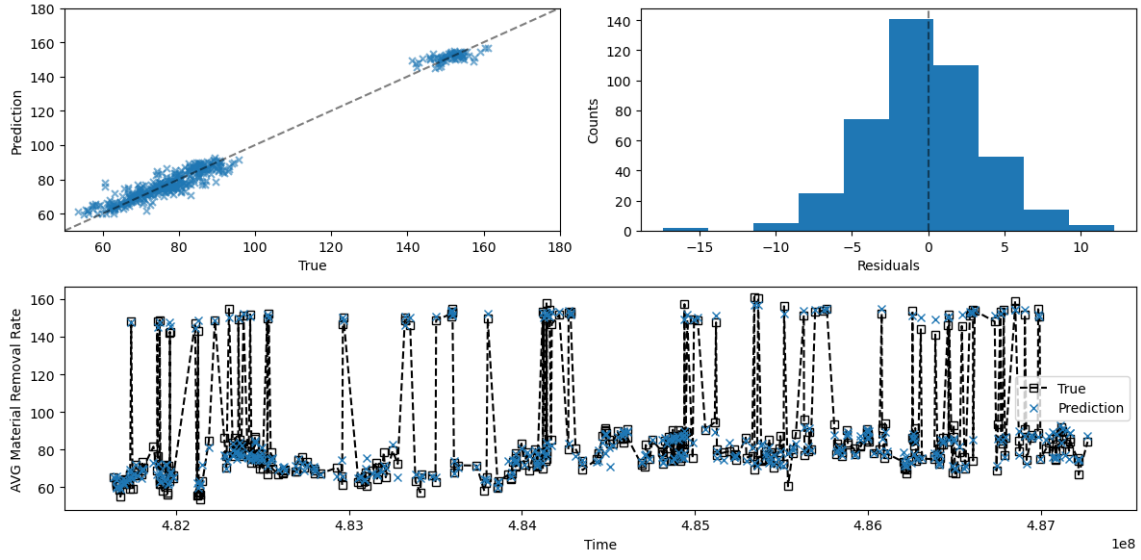

Figure 24: Visualization of MLP-VAE perfomance on the testing dataset.

The autoencoder based virtual metrology models perform slightly worse than other approaches in general. However, MLP-VAE outperforms even the best supervised deep learning model in the testing dataset error and models the metrology with only marginally higher errors than the tree-based models. Figure 24 shows an illustration of the MLP-VAE's

prediction results on the testing dataset. The high-speed mode is somewhat difficult for the MLP-VAE to model, and there are some relatively high error predictions in the low-speed mode as well. The residuals are quite evenly distributed around zero, and the predictions remain equally accurate throughout the time period of the testing dataset as can be seen in the time series plot.

Another significant aspect to compare - besides prediction performance - is the required time for the development and implementation of these models. The implementation of raw approach models took the least time to develop with 5 % of the total time spent on all of the model development. The development of the hand-crafted approaches took the second least time with 10 % of the total time. Both supervised deep learning and autoencoder based model approaches took individually more time than the development of the tree-based approaches. For the development of the supervised deep learning models, we used 30 % of total time, whereas the development of the autoencoder models took 40 % of our time. The consumed time was high because deep learning models contain numerous hyperparameters, optimization methods and architectural choices which may all affect the final model performance unexpectedly. In addition, the training process of deep learning models takes more time than of traditional machine learning models. This increases the time spent on trial-and-error type of development which is necessary in machine learning research. Furthermore, these results reflect that the development of deep learning models might consume more time but as a result automates the feature learning, whereas hand-crafted features are fairly fast to craft but often fail to generalize to other use cases. The raw tree-based approach gave great returns on the invested development time.

5.2 Supervised recurrent neural network variations

We described in Section 4.3.2 the different approaches for implementing supervised RNNs. This section discusses the performance and viability of these approaches for including time-invariable exogenous variables, such as process stage, in supervised RNNs. In addition, we review the applicability of RNN cells and computation of padded values to the virtual metrology prediction task.

Table 4 shows the training time and prediction error on the validation and testing datasets for each RNN model version. It is evident from the table that the best performing model is a GRU model where the exogenous variables are included in the initial hidden state and the model computes the padded values. When the same model is informed of the sequence lengths (computation stops when sequence ends), the model performs significantly worse on the virtual metrology task. Another approach that works well for the GRU cell is the approach where all the variables are inputted to the network simultaneously on a single tensor and the model is not informed of the sequence lengths. This approach reaches testing RMSE of 3.79 that is significantly lower than the average error of all RNN models.

By average, the GRU based models perform better than the LSTM based models. Although, the difference in testing errors is not drastic. This supports the understanding that GRU and LSTM cells are both state-of-the-art RNN solutions that perform similarly when compared to each other. Furthermore, neither RNN type models overfit to the data with hyperparameter tuning which might also suggest that the validation part of the split is easier to predict than the testing part as generally the error increases from validation to testing dataset. This is, however, acceptable as it means that our RNN models react well to completely unseen observations.

Table 4: Performance of different recurrent neural network virtual metrology models.

RNN type	Exogenous variables handling	Padding values computed	Training time (sec)	Validation		Testing	
				RMSE	R^2	RMSE	R^2
GRU	Exogenous variables afterwards	Yes	247	4.39	0.98	4.50	0.98
		No	236	4.26	0.98	4.24	0.98
	Exogenous variables to h_0	Yes	247	**3.68**	0.98	**3.66**	0.98
		No	240	4.19	0.98	4.11	0.98
	All variables from the beginning	Yes	247	3.85	0.98	3.79	0.98
		No	245	4.11	0.98	3.98	0.98
LSTM	Exogenous variables afterwards	Yes	255	7.21	0.94	7.89	0.92
		No	252	4.86	0.97	4.57	0.97
	Exogenous variables to h_0	Yes	256	4.40	0.98	4.20	0.98
		No	252	4.57	0.98	5.29	0.97
	All variables from the beginning	Yes	258	4.67	0.98	5.04	0.97
		No	257	4.37	0.98	4.46	0.98

The least attractive approach for handling exogenous variables seems to be including exogenous variables after the RNN cell. This could be due to the additional layers that are required after the RNN cell to include the non-linear characteristics of exogenous variables on the last hidden state of the RNN cell. In our experiments, these additional linear layers rarely resulted to better performance on the validation dataset. The two other approaches for handling exogenous variables perform similarly in terms of the validation and testing error. Based on these results, including all variables in the same tensor and inputting them to the RNN cell is the best choice because it performs competently and is simple to implement. This suggests that time-invariant features are not a challenge for an RNN cell when combined with time-variable multivariant data.

Generally, informing the RNN cell of the length of the sequence seems to have no significant advantage in our virtual metrology task. Informing the RNN cell of the sequence length results to better performance in 3 out of 6 approaches and 1 out of 3 GRU based models. For LSTM based models, not including padded values in the computation seems advantageous. The low significance of informing the model of sequence lengths is unexpected for us because the use of RNN models is often motivated with the pure computation of the time series values and not the padded values that have no actual meaning. However, the RNN models that are informed of the sequence values are slightly faster to train, and thus we see informing the model of the sequence lengths beneficial because more extreme differences in sequence lengths could result in a more substantial decrease in the model training time.

5.3 Limited labeled data and semi-supervised learning

This section discusses the performance of models in a modern manufacturing setting where only a fraction of wafers is physically measured, and thus fewer labels are acquired. We show results for different ratios of labeled to unlabeled wafers. Our supervised models can only utilize labeled samples, whereas the autoencoder models utilize unlabeled samples as well.

Figure 25 shows the regression error of one model from each approach type introduced earlier with 100, 500, 1000 and 1977 labeled wafers out of all the 1977 wafers. Firstly, it is immediately evident that ResNet, a supervised deep learning model, does not perform well with fewer labeled samples. Its error increases rapidly when approximately half of the labeled data is transmuted unlabeled. This supports the conception that supervised neural networks require plenty of labeled data in order to perform sufficiently. Secondly, we can notice that when the number of labeled wafers exceeds 500 – the raw and hand-crafted XGBoost models perform the best.

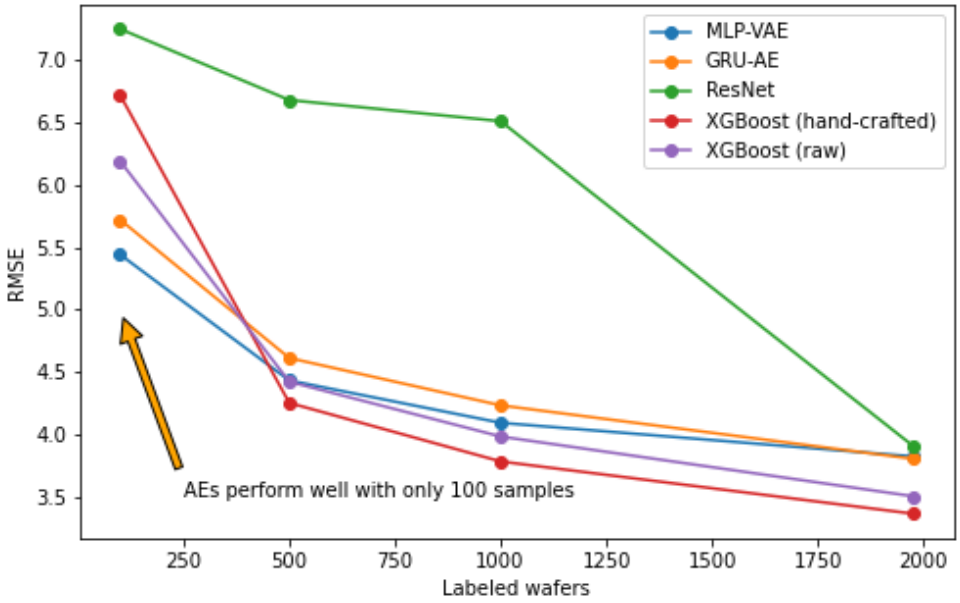

Figure 25: Model performance on the testing dataset with a different number of labeled samples of the 1977 wafers in the training dataset.

However, from the Figure 25, it is also evident that the autoencoder models outperform all other approaches in a restricted labeled data setting where only 100 wafers are labeled. MLP-VAE extracts useful features from unlabeled data that help the subsequent regression model to predict the metrology values accurately. Learning from just 100 labeled wafers, means that the model is capable of predicting the metrology results earlier than other models that require more labeled data to perform as effectively. This is valuable in semiconductor manufacturing where each physical measurement consumes time and data collection for virtual metrology cannot take too much time. Accurate predictions with little labeled data allow control and monitoring of the process sooner after an event such as tool maintenance or product recipe change.

A more detailed visualization of all the autoencoder models' performance is visible in Figure 26. The figure shows that MLP-VAE performs overall best out of the autoencoders and MLP-AE the worst. The RNN autoencoders perform adequately but do not provide significant benefits over MLP autoencoders. LSTM-AE and MLP-VAE perform similarly with only 100 labeled wafers, and overall, the difference in error between autoencoders trained with 100 labeled wafers is minuscule (RMSE from 5.44 to 5.88). The overall performance of the models improves with the number of labeled wafers, but the improvement is small compared to other approaches such as raw and hand-crafted.

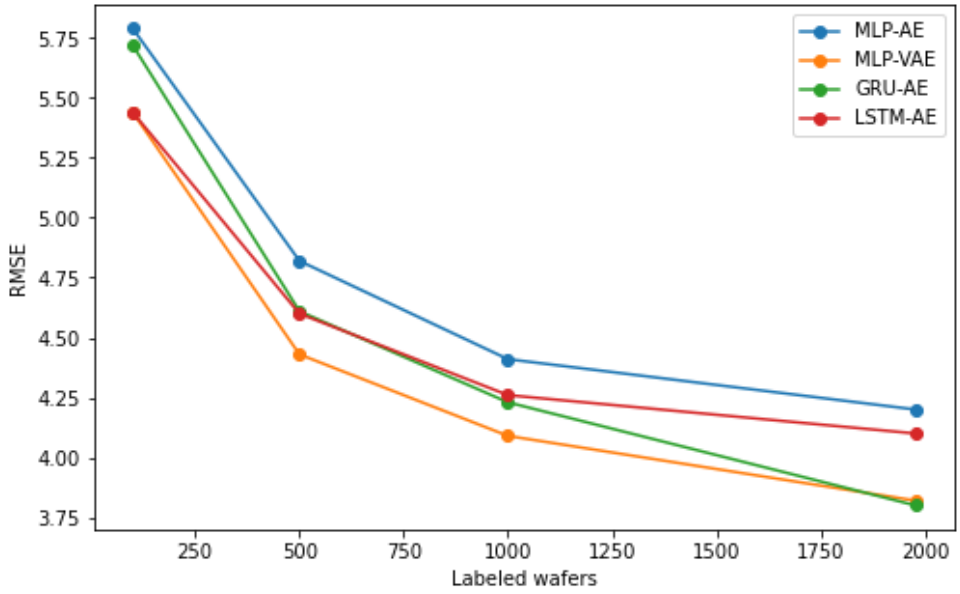

Figure 26: Performance of autoencoder models on the testing dataset with different number of labeled samples of the 1977 wafers in the training dataset.

Figure 27 shows the performance of MLP-VAE trained with only 100 labeled wafers. We notice that the residuals are much higher than in the Figure 24 that shows the performance of the MLP-VAE trained with more labeled samples. Nonetheless, the residuals are quite evenly distributed around zero and there are no outliers. This would suggest that the MLP-VAE is relatively robust in the virtual metrology task with limited labeled data. MLP-VAE performs well throughout the time series without any major sections of the time period with fluctuated performance.

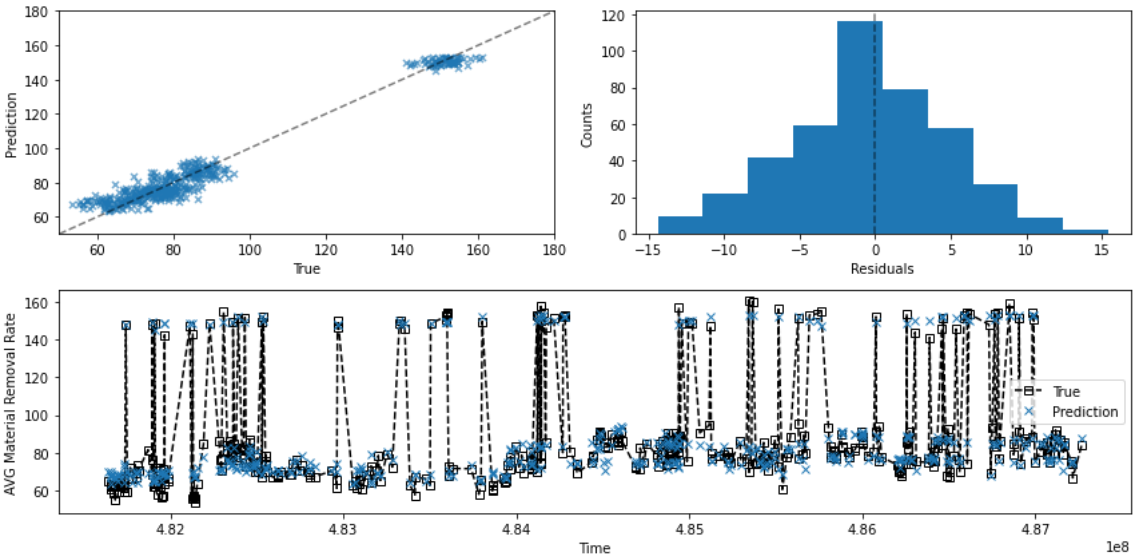

Figure 27: MLP-VAE trained with 100 labeled and 1877 unlabeled wafers.

Based on the results, autoencoder models work better than other approaches in the limited labeled data setting for virtual metrology. When the number of measured wafers increases significantly, other approaches become more accurate. This might, however, take a long time and result in low quality wafers that wind up scrapped. It must also be noted that the hand-

crafted features with XGBoost regressor outperform other methods when the labeled data reaches 1000 wafers. This suggests that the hand-crafted features are representative of the runs.

5.4 Raw approach analysis

Tree-based ensemble regressor's criterion reduction-based feature importance gives an overview of the variables' impact on model predictions. This provides insight on the factors or variables that have dependency with the model's predictions and can help to analyze the process's behavior. The raw approaches consider each measurement of each sensor and process information as a variable. A heatmap of the measurement's importance shows the normalized MSE reduction for each variable in the model.

Figure 28 shows the heatmap and histogram of XGBoost regressor's feature importance in the virtual metrology task. The most important feature in the model is the first measurement of the *Chamber* variable. It varies only a little during runs and therefore the splits often target the first measurement of that variable. This is expectable because the different modes of polishing speed are processed in different chambers (Figure 15). Nevertheless, this high feature importance value is clearly distinct in the histogram where all of the other variables have significantly lower feature importance values[20].

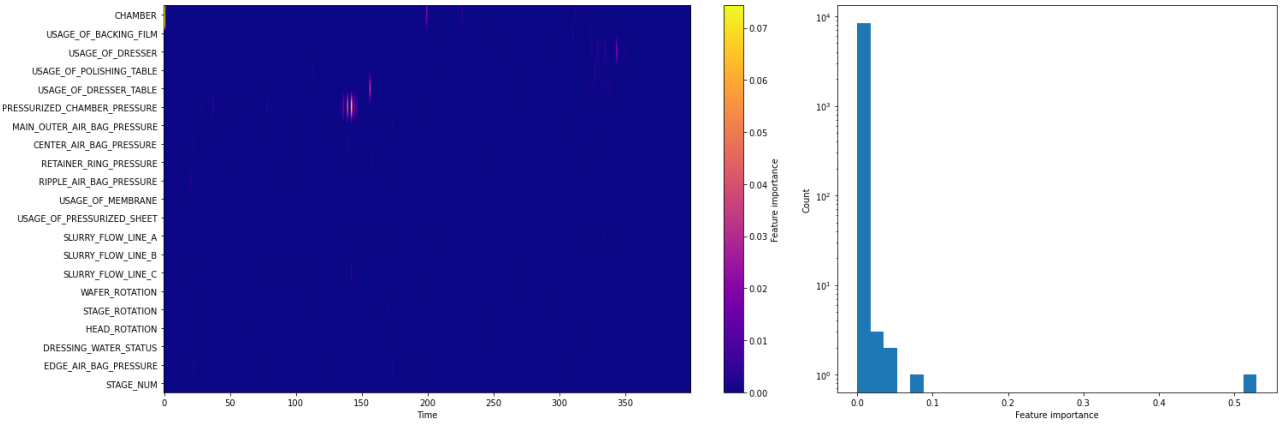

Figure 28: Feature importance of raw approach with XGBoost.

Another prominent sensor based on the feature importance is *Pressurized chamber pressure* at just before the timestep 150. The conspicuous group in the middle of the heatmap is interesting as it is probably caused by a certain event that happens often approximately on that period of the runs. For example, process operator's conditional actions could cause the importance of the time period in Pressurized chamber pressure. Another noteworthy aspect in the figure is the slight increase in feature importance of *Usage of dresser* just before the timestep 350. This might indicate a wearing event that often occurs in the end of the run. Furthermore, only 19 % of the measurement variables have non-zero feature importance and only a few are distinct from the heatmap where higher values dominate the visualization color scale.

The feature importance of the random forest regressor variables are shown in Figure 29. The results differ in some parts significantly from the XGBoost feature importance values.

[20] NOTE: Figure 25 heatmap color scale does not consider the single high (0.52) feature importance of Chamber variable in the color scale for visualization purposes.

For instance, in the *Chamber* variable, the importance is divided between several timesteps even though they are fully correlated. This happens because the SciKit-learn library implementation of the random forest selects subset of features randomly at each split[21] and therefore the first split on Chamber variable occurs on random measurement of the variable. The *Pressurized chamber pressure* is also prominent in the random forest even though the time period is shorter than in the XGBoost regressor. In addition, with the random forest regressor, approximately 90% of the variables have non-zero feature importance which is significantly higher than with the XGBoost regressor. This is due to the random subset of features that is sampled at each split. This is also visible in the histogram of the feature importance values where the values are quantified in certain values and distributed with even gaps.

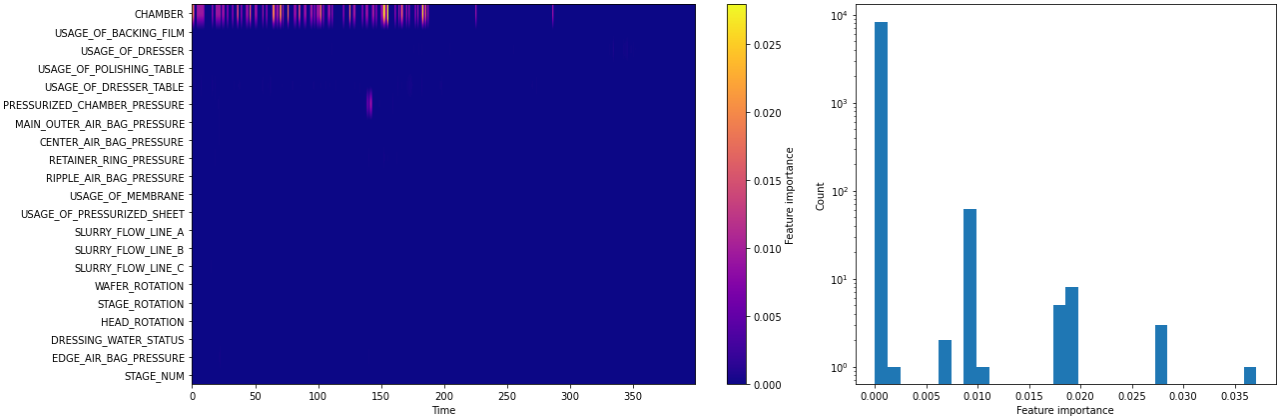

Figure 29: Random forest feature importance.

The feature importance values of both XGBoost and random forest regressor are similar with slight differences that are caused by each model's individual splitting and subsampling methods. However, with the feature importance values, we discovered that the characteristics of Pressurized chamber pressure have an effect on the material removal rate in the CMP tool or at least the behavior of the sensor is a proxy for another event that affects the polishing. We also discovered that only a fraction of the measurements is required to predict the material removal rate with a tree ensemble model.

[21] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html

6 Conclusions and discussion

In this thesis, we implemented and reviewed multiple state-of-the-art machine learning models for virtual metrology of chemical-mechanical polishing. We compared typical virtual metrology machine learning pipelines consisting of hand-crafted features and tree ensemble models to the current state-of-the-art time series extrinsic regression models, such as InceptionTime and recurrent neural networks. In addition, we proposed and evaluated several approaches for including time-invariant extrinsic process variables to recurrent neural networks. Furthermore, we implemented semi-supervised autoencoder models for a prediction scenario where only a fraction of process runs are labeled. These autoencoders were compared to other machine learning methods in a limited labeled data setting which is common in semiconductor manufacturing processes, such as chemical-mechanical polishing, where physical measurements are rarely conducted on the wafer. In addition, we evaluated the performance of tree ensemble models by analyzing the feature importance of the models.

Our extensive experiments and comparison of different models show that hand-crafted features and tree ensemble models are a good choice for the virtual metrology of chemical-mechanical polishing. These tree ensemble methods are relatively simple to implement and perform best in a setting where there are a high number of labeled samples. Hand-crafted features improve the performance compared to the raw values approach but require more domain knowledge and development time. Our experiments on automating the feature extraction with deep learning shows that supervised deep learning models are able to extract useful features from the process data. They perform decently with a high number of labeled samples, but as the amount of training data decreases - their performance decline rapidly. This is the case for both RNN and time series extrinsic models, such as InceptionTime and ResNet. However, these deep learning models automate the feature extraction process and thus adapt to new process environments without additional effort on handcrafting features from the data with domain knowledge. Nevertheless, initial development of deep learning models may consume a considerable amount of time based on our experiments.

In our experiments on RNNs, we found that including time-invariant exogenous variables with the rest of the variables - is a viable approach for supervised RNN models, since it is a simple technique to implement, and it performs well compared to other approaches. Another viable approach according to our results, is to fill the initial hidden state of the RNN cell with the exogenous variables to “condition” the model for different process stages or chambers as an example. Including the exogenous variables after the RNN cell, at an intermediate part of the neural network, was found to work the worst in our experiments. This is in line with our discovery that additional layers after the RNN cell often result in overfitting rather than improved performance. Furthermore, padding of variable length sequences did not appear to affect the RNN models’ performance significantly which evident because the models performed similarly when the padding values were not included in the computation as they did when the padding values were included.

In our limited data setting, we discovered that autoencoder models can predict metrology results accurately when trained with just 100 physically measured wafers. The semi-supervised approach allows the model to learn the structure and features of the data that a subsequent regression model can utilize in making predictions. Supervised regression models did not achieve as good results on the same task but after 500 labeled wafers their performance was comparable to that of autoencoders. This data collection may, however, take weeks in semiconductor manufacturing and result in decreased yield and quality before accurate

predictions are available. Therefore, learning from fewer labeled samples with a semi-supervised approach brings significant benefits to the semiconductor manufacturing. In our experiments, the variational autoencoder performed the best with both fully labeled and partially labeled datasets. Recurrent autoencoders performed similarly to MLP-based models but were considerably slower to train.

Our tree ensemble model feature importance analysis revealed that only a fraction of the measurements in the process data are sufficient for an accurate prediction of average material removal rate in a chemical-mechanical polishing tool. In fact, XGBoost utilized only 19% of the measurements and achieved an R^2 score of 0.99 in the testing set. The most important features according to our analysis were the chamber the wafer was processed in and pressurized chamber pressure at a certain time period. The average removal rate is highly dependent on the chamber because different polishing speed modes are operated in specific chambers. In addition, we believe that the pressurize chamber pressure received high importance because it acts as a proxy for an event that occurs during that time, which is not visible otherwise in the sensor data. Based on the feature importance analysis, we believe that the raw approaches perform well on the PHM dataset because the prediction problem is fairly easy, and the data does not inherit complex temporal structure, such as autocorrelations. The performance of the raw approaches would likely decline in more complex prediction tasks.

The main contribution of our work to the virtual metrology research field is the comparison of state-of-the-art time series deep learning algorithms on the chemical-mechanical polishing tool. Especially, our experiments on time series extrinsic regression models contribute to the knowledge of their performance in semiconductor manufacturing which has not been included in the benchmark datasets in recent publications (Tan et al. 2021). In addition, our experiments on autoencoders are the first we have encountered that consider a semi-supervised setting where the autoencoder utilizes unlabeled process data alongside labeled data. These results extend the previous research on autoencoder use in virtual metrology (Maggipinto et al. 2018) (Choi & Jeong 2019). Furthermore, our standard tree-based models outperform some of the more complex solutions (Li et al. 2019) encountered in publications that utilize the same dataset in their tests. This shows that even a simple solution may prove out efficient in virtual metrology of chemical-mechanical polishing when implemented correctly.

The most significant limitation of our work is the PHM dataset which was used in evaluating the performance of our models. Firstly, the dataset portraits the operation of a single process tool during a fixed time period which limits the experiments that would compare the predictions on, for example, different tools or after tool maintenance. Secondly, it does not contain unlabeled data from process runs where the physical metrology is not conducted, which is why we need to synthetically limit the ratio of labeled to unlabeled data in our experiments. Ideally, the data would contain unlabeled process data from the ~24 wafers that were processed between physical measurements. This would result to data from ~50 000 processed wafers for training data in the case of PHM data that consists of ~2 000 measured wafers. Moreover, this would allow training autoencoders with more data from a shorter time period. Thirdly, our dataset contains only one measured response variable (average material removal rate) that is inevitably restricted in its nature because it is an average of a single variable across a two-dimensional surface. It would be beneficial for root-cause and other analysis purposes to predict, for instance, the topology of the wafer from multiple measured points. Lastly, we did not find any public datasets from different process tools that

we could use as an additional validation for general applicability of our models and implementations.

Future work could include the use of virtual metrology predictions on run-to-run control of the process tool. This would allow to evaluate the actual benefits of accurate virtual metrology predictions because the virtual metrology model would affect the process parameters and thus the yield and quality of manufacturing. Another interesting future topic would be to study how to dynamically control the frequency of physical metrology based on certainty estimates of a machine learning model and study the accuracy of the certainty estimates. This way the models could reduce the need for physical metrology even further. Additionally, the frequency of the model re-training and utilization of certainty estimates on triggering the re-training procedure would be an interesting research topic. In addition to our feature importance analysis, we could also research interpretable machine learning methods, such as LIME[22] or SHAP[23], and try to find explanations for the predictions. These explanations could be then further utilized in root cause analysis of the models and the process.

[22] Local Interpretable Model-agnostic Explanations

[23] Shapley Additive exPlanations

References

Alain, G., and Bengio, Y. (2014). What Regularized Auto-Encoders Learn from the Data-Generating Distribution. *The Journal of Machine Learning Research*, [online] Volume 15(1), pp. 3563–3593. Available at: https://dl.acm.org/doi/10.5555/2627435.2750359 [Accessed 10 Jun. 2021].

Babu, S. V. (2016). *Advances in chemical mechanical planarization (CMP)*. Waltham, MA: Woodhead Publishing.

Bagnall, A., Lines, J., Hills, J., and Bostrom, A. (2015) Time-series classification with COTE: the collective of transformation-based ensembles. IEEE Transactions on Knowledge and Data Engineering, [online] Volume 27(9), pp. 2522–2535. Available at: https://ieeexplore.ieee.org/document/7069254 [Accessed 15 May 2021].

Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, [online] Volume 5(2), pp. 157–166. Available at: https://ieeexplore.ieee.org/document/279181 [Accessed 27 May 2021].

Breiman, L. (2001). Random Forests. *Machine Learning*, [online] Volume 45, pp. 5–32. Available at: https://link.springer.com/article/10.1023%2FA%3A1010933404324 [Accessed 10 Sep. 2021].

Che, Z., Purushotham, S., Cho, K., Sontag, D., and Liu, Y. (2018). Recurrent Neural Networks for Multivariate Time Series with Missing Values. *Scientific Reports*, [online] Volume 8, pp. 6085. Available at: https://www.nature.com/articles/s41598-018-24271-9 [Accessed 7 Jun. 2021].

Chen, P., Wu, S., Lin, J., Ko, F., Lo, H., Wang, J., Yu, C. H., and Liang, M. S. (2005). Virtual metrology: A solution for wafer to wafer advanced process control. In: *Proceedings of IEEE international symposium on semiconductor manufacturing (ISSM 2005)*. [online] San Jose, CA: IEEE. Available at: https://ieeexplore.ieee.org/document/1513322 [Accessed 1 Mar. 2021].

Cho, K., Merrienboer, B. V., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y.. (2014). Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. [online] Doha: Association for Computational Linguistics, pp. 1724-1734. Available at: http://emnlp2014.org/papers/pdf/EMNLP2014179.pdf [Accessed 21 May 2021].

Choi, J., and Jeong, M. K. (2019). Deep Autoencoder With Clipping Fusion Regularization on Multistep Process Signals for Virtual Metrology. In: *IEEE Sensors Letters*. [online] Available at: https://ieeexplore.ieee.org/document/8556469 [Accessed 10 May 2021].

Chung, J., Caglar, G., Cho, K. H., and Bengio, Y. (2014). Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. [online] arXiv. Available at: https://arxiv.org/pdf/1412.3555.pdf [Accessed 20 May 2021].

Di, Y., Jia, X., and Lee, J. (2020). Enhanced Virtual Metrology on Chemical Mechanical Planarization Process using an Integrated Model and Data-Driven Approach. *International Journal of Prognostics and Health Management*, [online] Volume 8(2). Available at: https://papers.phmsociety.org/index.php/ijphm/article/view/2641 [Accessed 6 Apr. 2021].

Garcevic, H., and Lidberg, E. (2021). *COVID-19 effects on supply chain risk management in the Swedish automotive industry*. Chalmers University of Technology.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. 1st ed. [ebook]. MIT Press. Available at: https://www.deeplearningbook.org/ [Accessed 25 Mar. 2021].

Grayson, K. (2018). Photolithography – The Role and Properties of Photosensitive Glass. [online] Mo-Sci. Available at: https://mo-sci.com/photolithography-photosensitive-glass/ [Accessed 21 Jun. 2021].

Hastie, T., Tibhirani, R., and Friedman, J. *The elements of statistical learning: data mining, inference, and prediction*. 2nd ed. [eBook]. Springer. Available at: https://web.stanford.edu/~hastie/ElemStatLearn/printings/ESLII_print12_toc.pdf [Accessed 10 Sep. 2021].

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep Residual Learning for Image Recognition. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [online] Las Vegas: IEEE, pp. 770-778. Available at: https://ieeexplore.ieee.org/document/7780459 [Accessed 8 Jun. 2021].

Hochreiter, S. (1991). *Untersuchungen zu dynamischen neuronalen Netzen*. Technische Universität München.

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, [online] Volume 9(8), pp. 1735–1780. Available at: https://pubmed.ncbi.nlm.nih.gov/9377276/ [Accessed 27 May 2021].

Hüsken, M., and Stagge, P. (2003). Recurrent neural networks for time series classification. *Neurocomputing*, [online] Volume 50, pp. 223-235. Available at: https://www.sciencedirect.com/science/article/abs/pii/S0925231201007068 [Accessed 4 Jun. 2021].

Ismail Fawaz, H., Lucas, B., Forestier, G., Pelletier, C., Schmidt, D.F., Weber, J., Webb, G.I., Idoumghar, L., Muller, P. A., and Petitjean, F. (2020). InceptionTime: Finding AlexNet for time series classification. *Data Mining and Knowledge Discovery,* [online] Volume 34(6), pp. 1936–1962. Available at: https://link.springer.com/article/10.1007/s10618-020-00710-y [Accessed 3 Jun. 2021].

Jebri, M., El Adel, E., Graton, G., Ouladsine, M., and Pinaton, J. (2017). Virtual Metrology applied in Run-to-Run Control for a Chemical Mechanical Planarization process. *Journal of Physics: Conference Series*, [online] Volume 783(1), p. 12042. Available at: https://iopscience.iop.org/article/10.1088/1742-6596/783/1/012042 [Accessed 23 Jun. 2021].

Jia, X., Di, Y., Feng, J., Yang, Q., Dai, H., and Lee, J. (2018). Adaptive virtual metrology for semiconductor chemical mechanical planarization process using GMDH-type polynomial

neural networks. *Journal of Process Control*, [online] Volume 62, pp. 44–54. Available at: https://www.sciencedirect.com/science/article/pii/S0959152417302263 [Accessed 16 May 2021].

Kang, P., Lee, H., Cho, S., Kim, D., Park, J., Park, C., and Doh, S. (2009). A virtual metrology system for semiconductor manufacturing. *Expert Systems with Applications*, [online] Volume 36(10), pp. 12554–12561. Available at: https://www.sciencedirect.com/science/article/abs/pii/S0957417409004746 [Accessed 4 May 2021].

Kang, Y., Hyndman, R. J., and Smith-Miles, K. (2017). Visualising forecasting algorithm performance using time series instance spaces. *International Journal of Forecasting*, [online] Volume 33(2), pp. 345–358. Available at: https://www.sciencedirect.com/science/article/abs/pii/S0169207016301030 [Accessed 24 May 2021].

Khan, A. A., Moyne, J. R., and Tilbury, D. M. (2007). An Approach for Factory-Wide Control Utilizing Virtual Metrology. *IEEE Transactions on Semiconductor Manufacturing*, [online] Volume 20(4), pp. 364-375. Available at: https://ieeexplore.ieee.org/document/4369341 [Accessed 10 May 2021].

Kingma, D. P., and Welling, M. (2014). Auto-Encoding Variational Bayes. [online] arXiv. Available at: https://arxiv.org/abs/1312.6114 [Accessed 12 Jun. 2021].

Kingma, D. P., Rezende, D. J., Mohamed, S., and Welling, M. (2014). Semi-supervised Learning with Deep Generative Models. In: *Advances in Neural Information Processing Systems 27 (NIPS 2014)*. [online] Montreal: Curran Associates, Inc., pp. 3581-3589. Available at: https://papers.nips.cc/paper/2014/hash/d523773c6b194f37b938d340d5d02232-Abstract.html [Accessed 12 Jun. 2021].

Kong, Z., Oztekin, A., Beyca, O., Phatak, U., Bukkapatnam, S., and Komanduri, R. (2010). Process Performance Prediction for Chemical Mechanical Planarization (CMP) by Integration of Nonlinear Bayesian Analysis and Statistical Modeling. *IEEE Transactions on Semiconductor Manufacturing*, [online] Volume 23(2), pp. 316-327. Available at: https://ieeexplore.ieee.org/document/5433049 [Accessed 5 Apr. 2021].

Lee, K. B., and Kim, C. O. (2020). Recurrent feature-incorporated convolutional neural network for virtual metrology of the chemical mechanical planarization process. *Journal of Intelligent Manufacturing*, [online] Volume 31(1), pp. 73-86. Available at: https://link.springer.com/article/10.1007/s10845-018-1437-4 [Accessed Apr. 2021].

Li, Z., Wu, D., and Yu, T. (2019). Prediction of Material Removal Rate for Chemical Mechanical Planarization Using Decision Tree-Based Ensemble Learning. *Journal of Manufacturing Science and Engineering*, [online] Volume 141(3). Available at: https://doi.org/10.1115/1.4042051 [Accessed 21 Apr. 2021].

Lin, T., Hung, M., Lin, R., and Cheng F. (2006). A virtual metrology scheme for predicting CVD thickness in semiconductor manufacturing. In: *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006*. [online] Orlando, FL: IEEE. Available at: https://ieeexplore.ieee.org/document/1641849 [Accessed 17 May 2021].

Lynn, S. A., MacGearailt, N., and Ringwood, J.V. (2012a). Real-time virtual metrology and control for plasma etch. *Journal of Process Control*, [online] Volume 22(4), pp. 666–676.

Available at: https://www.sciencedirect.com/science/article/pii/S0959152412000157 [Accessed 12 May 2021].

Lynn, S. A., Ringwood, J., and MacGearailt, N. (2012b). Global and Local Virtual Metrology Models for a Plasma Etch Process. *IEEE Transactions on Semiconductor Manufacturing*, [online] Volume 25(1), pp. 94-103. Available at: https://ieeexplore.ieee.org/document/6084764 [Accessed 26 Jun. 2021].

Maggipinto, M., Beghi, A., McLoone, S., and Susto, G.A. (2019). DeepVM: A Deep Learning-based approach with automatic feature extraction for 2D input data Virtual Metrology. *Journal of Process Control*, [online] Volume 84, pp. 24–34. Available at: https://www.sciencedirect.com/science/article/pii/S095915241930191X [Accessed 19 Mar. 2021].

Maggipinto, M., Masiero, C., Beghi, A., and Susto, G. A. (2018). A Convolutional Autoencoder Approach for Feature Extraction in Virtual Metrology. In: *Procedia Manufacturing, Volume 17*. [online] Columbus, OH: Elsevier B.V, pp. 126-133. Available at: https://www.sciencedirect.com/science/article/pii/S2351978918311399 [Accessed 1 May 2021].

Malhotra, P., Ramakrishnan, A., Anand, G., Vig, L., Agarwal, P., and Shroff, G. (2016). LSTM-based Encoder-Decoder for Multi-sensor Anomaly Detection. [online] arXiv. Available at: https://arxiv.org/pdf/1607.00148.pdf [Accessed 23 May 2021].

Nanz, G., and Camilletti, L. E. (1995). Modeling of chemical-mechanical polishing: a review. *IEEE Transactions on Semiconductor Manufacturing*, [online] Volume 8(4), pp. 382-389. Available at: https://ieeexplore.ieee.org/abstract/document/475179 [Accessed 3 Mar. 2021].

Orji, N., Obeng, Y., Beitia, C., Mashiro, S., & Moyne, J. (2018). Virtual Metrology White Paper. IEEE-International Roadmap for Devices and Systems (IRDS). [Whitepaper]. Available at: https://www.nist.gov/publications/virtual-metrology-white-paper-international-roadmap-devices-and-systemsirds [Accessed 24 Jun. 2021].

Partaourides, H. and Chatzis, S. P. (2017). Asymmetric deep generative models. *Neurocomputing*, [online] Volume 241, pp. 90–96. Available at: https://www.sciencedirect.com/science/article/abs/pii/S0925231217302989?via%3Dihub [Accessed at 6 Jun. 2021].

Pascanu, R., Mikolov, T., and Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In: *Proceedings of the 30th International Conference on Machine Learning*. [online] Atlanta: JMLR.org, pp. 1310–1318. Available at: https://dl.acm.org/doi/10.5555/3042817.3043083 [Accessed 15 May 2021].

Patterson, J., and Gibson, A. (2017). *Deep Learning*. 1st ed. [ebook]. O'Reilly Media, Inc. Available at: https://www.oreilly.com/library/view/deep-learning/9781491924570/ [Accessed 25 Mar 2021].

Qin, S. J., Cherry, G., Good, R., Wang, J., and Harrison, C. A. (2006). Semiconductor manufacturing process control and monitoring: A fab-wide framework. *Journal of Process Control*, [online] Volume 16(3), pp. 179-191. Available at: https://www.sciencedirect.com/science/article/pii/S0959152405000600 [Accessed 3 Mar. 2021].

Ragnoli, E., McLoone, S., Lynn, S., Ringwood, J., and Macgearailt, N. (2009). Identifying key process characteristics and predicting etch rate from high-dimension datasets. In: *2009 IEEE/SEMI Advanced Semiconductor Manufacturing Conference*. [online] Berlin: IEEE, pp. 106-111. Available at: https://ieeexplore.ieee.org/document/5155966 [Accessed 25 Jun. 2021].

Rao, P., Bhushan, M., Bukkapatnam, S., Kong, Z., Byalal, S., Beyca, O., Fields, A., and Komanduri, R. (2014). Process-Machine Interaction (PMI) Modeling and Monitoring of Chemical Mechanical Planarization (CMP) Process Using Wireless Vibration Sensors. *IEEE Transactions on Semiconductor Manufacturing*, [online] Volume 27(1), pp. 1-15. Available at: https://ieeexplore.ieee.org/document/6675842 [Accessed 20 Apr. 2021].

Reiss, A., Indlekofer, I., Schmidt, P., and Van Laerhoven, K. (2019). Deep PPG: Large-Scale Heart Rate Estimation with Convolutional Neural Networks. *Sensors*, [online] Volume 19(14), p. 3079. Available at: https://www.mdpi.com/1424-8220/19/14/3079/html [Accessed 23 May 2021].

Rokach, L., and Maimon, O. (2008). *Data mining with decision trees*. 1st ed. [eBook]. World Scientific Publication Co. Available at: https://doc.lagout.org/Others/Data%20Mining/Data%20Mining%20with%20Decision%20Trees_%20Theory%20and%20Applications%20%5bRokach%20%26%20Maimon%202008-04-01%5d.pdf [Accessed 9 Sep. 2021].

Steigerwald, J.M., Murarka, S.P., Gutmann, R.J., and Duquette, D.J. (1995). Chemical processes in the chemical mechanical polishing of copper. *Materials Chemistry and Physics,* [online] Volume 41(3), pp. 217–228. Available at: https://www.sciencedirect.com/science/article/abs/pii/0254058495015167 [Accessed 26 Mar. 2021].

Sun, W., and Wang, R. (2018). Fully Convolutional Networks for Semantic Segmentation of Very High Resolution Remotely Sensed Images Combined With DSM. I*EEE Geoscience and Remote Sensing Letters,* [online] Volume 15(3), pp. 474-478. Available at: https://ieeexplore.ieee.org/document/8281008 [Accessed 7 Jun. 2021].

Susto, G. (2017). A dynamic sampling strategy based on confidence level of virtual metrology predictions. In: *2017 28th Annual SEMI Advanced Semiconductor Manufacturing Conference (ASMC)*. [online] Saratoga Springs: IEEE, pp. 78-83. Available at: https://ieeexplore.ieee.org/document/7969203 [Accessed 24 Jun. 2021].

Szegedy, C., Ioffe, S., Vanhoucke, V., and Alemi, A. A. (2017). Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. In: *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*. [online] San Francisco: AAAI Press, pp. 4278-4284. Available at: https://dl.acm.org/doi/abs/10.5555/3298023.3298188 [Accessed 21 Jul. 2021].

Tan, C.W., Bergmeir, C., Petitjean, F., and Webb, G. (2021). Time series extrinsic regression. *Data Mining and Knowledge Discovery*, [online] Volume 35, pp. 1032–1060. Available at: https://link.springer.com/article/10.1007/s10618-021-00745-9 [Accessed 15 May 2021].

Tang, T. (2015). Document Modeling with Gated Recurrent Neural Network for Sentiment Classification. In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. [online] Lisbon: Association for Computational Linguistics, pp.

1422–1432. Available at: https://www.aclweb.org/old_anthology/D/D15/D15-1167.pdf [Accessed 3 Jun. 2021].

Terzi, M., Masiero, C., Beghi, A., Maggipinto, M., and Susto, G. A. (2017). Deep learning for virtual metrology: Modeling with optical emission spectroscopy data. In: *2017 IEEE 3rd International Forum on Research and Technologies for Society and Industry (RTSI)*. [online] Modena: IEEE, pp. 1-6. Available at: https://ieeexplore.ieee.org/document/8065905 [Accessed 21 May 2021].

Tolstikhin, I., Houlsby, N., Kolesnikov, A., Beyer, L., Zhai, X., Unterthiner, T., Yung, J., Steiner, A., Keysers, D., Uszkoreit, J., Lucic, M., and Dosovitskiy, A. (2021). MLP-Mixer: An all-MLP Architecture for Vision. [online] arXiv. Available at: https://arxiv.org/abs/2105.01601v1 [Accessed 14 Jun. 2021].

Tu, Y. M., and Lu, C. W. (2017). The Influence of Lot Size on Production Performance in Wafer Fabrication Based on Simulation. *Procedia Engineering*, [online] Volume 174, pp. 135–144. Available at: https://www.sciencedirect.com/science/article/pii/S1877705817301807 [Accessed 22 Jun. 2021].

Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.A. (2008). Extracting and Composing Robust Features with Denoising Autoencoders. In: *Proceedings of the 25th International Conference on Machine Learning*. [online] New York: Association for Computing Machinery, pp. 1096–1103. Available at: https://dl.acm.org/doi/10.1145/1390156.1390294 [Accessed 13 Jun. 2021].

Wang, P., Gao, R. X., and Yan, R. (2017). A deep learning-based approach to material removal rate prediction in polishing. *CIRP Annals*, [online] Volume 66(1), pp. 429–432. Available at: https://www.sciencedirect.com/science/article/abs/pii/S0007850617300136 [Accessed 6 Apr. 2021].

Wang, Y.L., Wu, J., Liu, C.W., Wang, T.C., and Dun, J. (1998). Material characteristics and chemical–mechanical polishing of aluminum alloy thin films. *Thin Solid Films*, [online] Volume 332(1–2), pp. 397–403. Available at: https://www.sciencedirect.com/science/article/abs/pii/S0040609098012000 [Accessed 27 Mar. 2021].

Wang, Z., Yan, W., and Oates, T. (2017). Time series classification from scratch with deep neural networks: A strong baseline. In: *2017 International Joint Conference on Neural Networks (IJCNN)*. [online] Anchorage: IEEE, pp. 1578-1585. Available at: https://ieeexplore.ieee.org/document/7966039 [Accessed 5 Jun. 2021].

Wong, H.S., Akarvardar, K., Antoniadis, D., Bokor, J., Hu, C., King-Liu, T.J., Mitra, S., Plummer, J., and Salahuddin, S. (2020). A Density Metric for Semiconductor Technology [Point of View]. In: *Proceedings of the IEEE*. [online] IEEE, pp. 478-482. Available at: https://ieeexplore.ieee.org/document/9063714 [Accessed 6 Jun. 2021].

Xiao, H. (2012). *Introduction to Semiconductor Manufacturing Technology*. 2nd Edition. [eBook]. SPIE. Available at: https://doi.org/10.1117/3.924283 [Accessed 20 Jun. 2021].

Yu, T., Li, Z. and Wu, D. (2019). Predictive modeling of material removal rate in chemical mechanical planarization with physics-informed machine learning. *Wear*, [online] Volume 426–427, pp. 1430–1438. Available at: https://www.sciencedirect.com/science/article/abs/pii/S0043164819303229 [Accessed 20 Mar. 2021].

Yu, W., Kim, I.Y., and Mechefske, C. (2021). Analysis of different RNN autoencoder variants for time series classification and machine prognostics. *Mechanical Systems and Signal Processing*, [online] Volume 149, p. 107322. Available at: https://www.sciencedirect.com/science/article/abs/pii/S0888327020307081 [Accessed 6 Jun. 2021].

Yung-Cheng, J. C., and Cheng, F. (2005). Application development of virtual metrology in semiconductor industry. In: *31st Annual Conference of IEEE Industrial Electronics Society, 2005. IECON 2005*. [online] Raleigh, NC: IEEE. Available at: https://ieeexplore.ieee.org/document/1568891 [Accessed 20 Mar. 2021].

A. **Interrupted and absurdly long process runs**

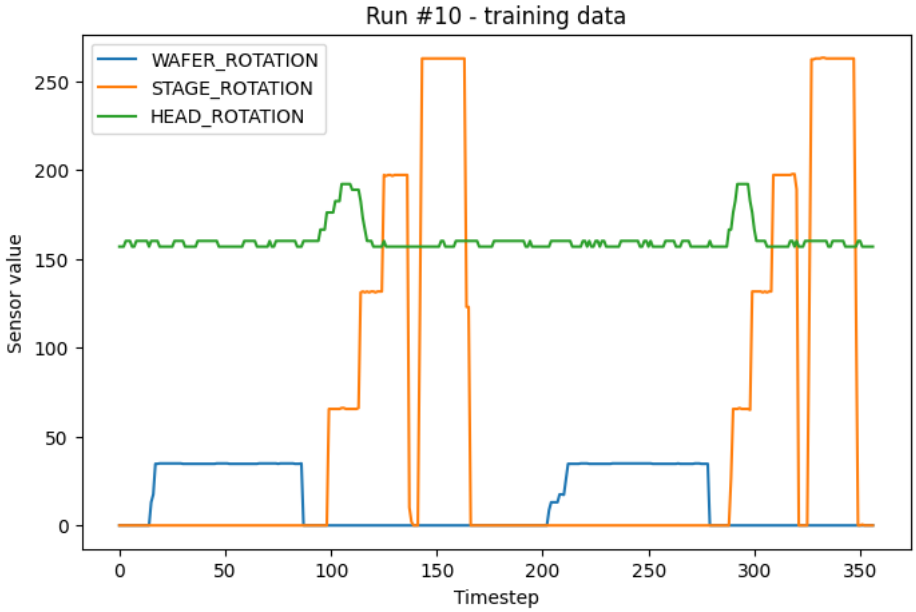

Figure A1: A typical “normal” run that is 357 timesteps long.

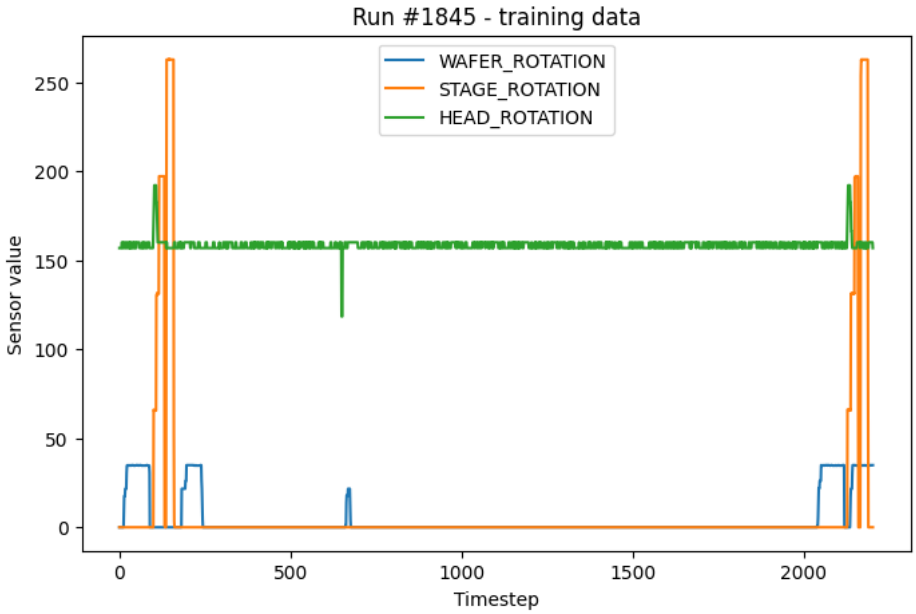

Figure A2: An instance of an absurdly long sequence where the rotational speeds are zero (0) most of the time.

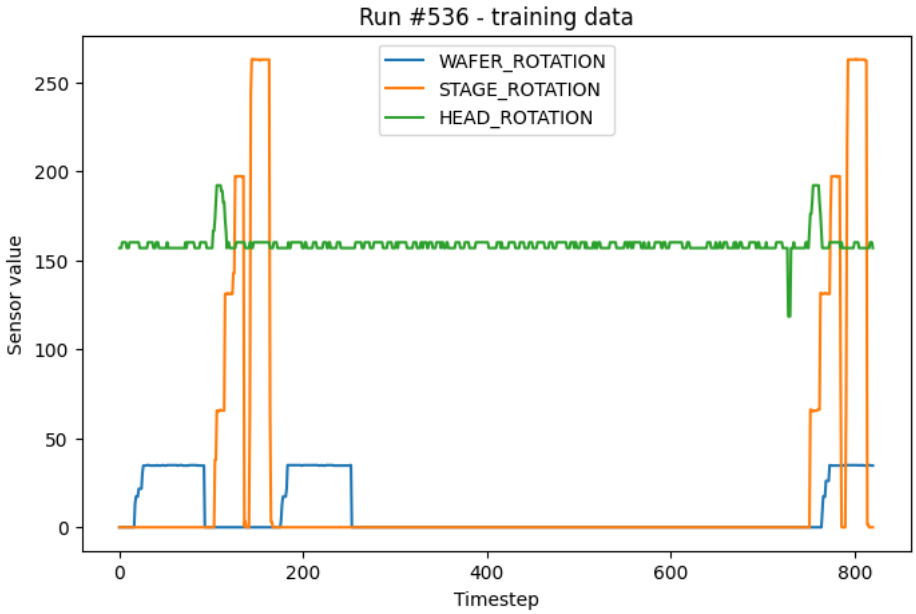

Figure A3: Another instance of a long sequence where the rotational speeds are zero (0) most of the time.

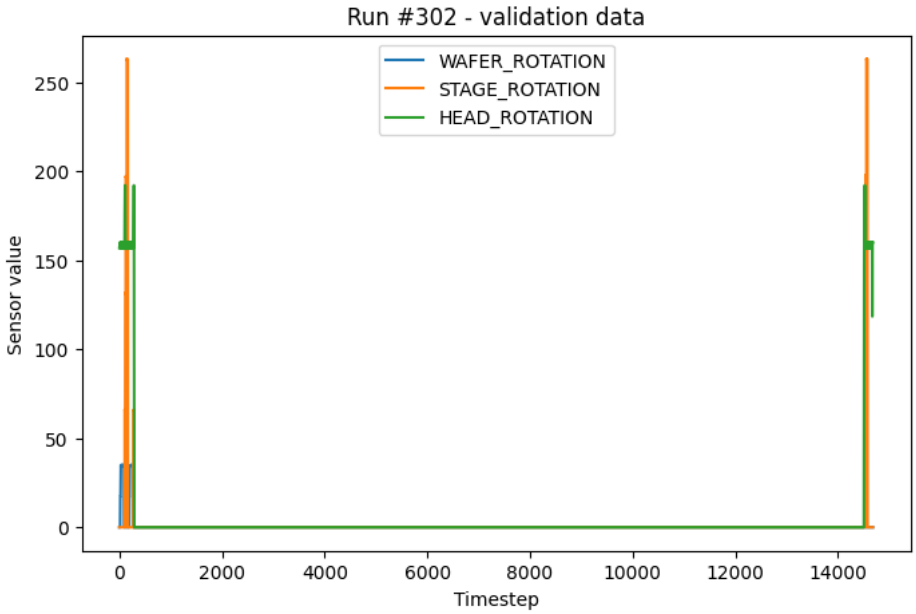

Figure A4: The longest sequence in the entire dataset is 14 678 timestep long.