A Machine Learning Approach to Characterizing Particle Morphology in Nuclear Forensics

Daniel A. Gum

A MACHINE LEARNING APPROACH TO CHARACTERIZING PARTICLE MORPHOLOGY IN NUCLEAR FORENSICS

THESIS

Daniel A. Gum, First Lieutenant, USAF

AFIT-ENP-MS-20-M-099

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENP-MS-20-M-099

A Machine Learning Approach to Characterizing Particle Morphology in Nuclear Forensics

THESIS

Presented to the Faculty

Department of Engineering Physics

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Nuclear Engineering

Daniel A. Gum, BS

First Lieutenant, USAF

March 2020

AFIT-ENP-MS-20-M-099

A Machine Learning Approach to Characterizing Particle Morphology in Nuclear Forensics

THESIS

Daniel A. Gum, BS
First Lieutenant, USAF

Committee Membership:

Dr. Abigail A. Bickley
Chair

Dr. Ashley D. Holland
Member

Dr. John W. McClory
Member

AFIT-ENP-MS-20-M-099

Abstract

A machine learning approach is taken to characterizing a group of synthetic uranium bearing particles. Scanning electron microscope images of these lab-created particles were converted into a binary string representation that captured morphological features in accordance with a guide established by Los Alamos National Laboratory. Each individual particle in the dataset contains an association with chemical creation conditions: processing method, precipitation temperature and pH, calcination temperature, source materials, additive and final products. Machine learning classifiers take binary strings of morphology data and train on them, in order to predict the creation conditions of a test particle. Additionally, the results show that the final products, source materials, and calcination temperature are most closely tied to particle morphology. In addition, trained classifiers are able to successfully relate final products between two particles, implying that morphological features are shared between particles with similar chemical composition. The results show that machine learning classifiers can still successfully differentiate these creation conditions when combined into groups and considered as a whole, a result which could serve as a valuable utility to analysts working on large forensic datasets. Of the classifiers tested, modern techniques such as Gradient Boosting performed the best, with other algorithms such as Bernoulli Naive Bayes and K Nearest Neighbors achieving high performance levels as well.

Table of Contents

Page

Abstract iv

List of Figures vii

List of Tables xiii

1. Introduction 1

1.1 Motivation 1
1.2 Background 3
1.3 Problem 8
1.4 Questions & Hypothesis 8
1.5 Approach 9
1.6 Assumptions & Limitations 11

2. Theory 13

2.1 Uranium Bearing Particle Formation 13
2.2 Nuclear Forensic Analysis 16
2.3 Machine Learning Foundations 17
2.4 Decision Trees 18
2.5 Bootstrapping and Random Forests 20
2.6 Random Forests 21
2.7 Boosted Trees 22
2.8 Imbalanced Datasets 24
2.9 Over-Sampling Techniques 25
2.10 Balanced Accuracy 27
2.11 A Brief Introduction To Neural Networks 28

3. Methodology 31

3.1 Machine Learning Process 31
3.2 Nuclear Forensics Image Lexicon 32
3.3 Data Processing 33
3.4 A Neural Network Approach 36
3.5 Neural Network Methodology 37

4. Results 41

4.1 Predicting Q-value from binary morphology strings 41
4.2 Creation Condition Analysis 45
Mixed Oxide Prediction Using A Pure Final Product Classifier 49

Page

4.3 Hyperparameter Tuning . . . 50
4.4 Classifier Performance Across All Creation Conditions . . . 53
4.5 Failure Analysis . . . 56
4.6 Neural Network Results . . . 59

5. Conclusions . . . 61

5.1 Summary . . . 61
5.2 Initial Goals . . . 62
5.3 Lexicon Steps . . . 63
5.4 Future Work . . . 64

Appendix A. Morphology Flowchart . . . 66

Appendix B. Creation Conditions . . . 68

Appendix C. Glossary . . . 93

Appendix D. GitHub Repository . . . 94

Bibliography . . . 95

List of Figures

Figure Page

1 Overview of the nuclear fuel cycle from beginning to end, with the inclusion of optional reprocessing steps [1]. 5

2 Simplified flowchart for the reprocessing of spent fuel. 7

3 Example synthetic particle with unique project identifier on upper left. 10

4 Example two-class datasets that could be used for classification tasks. 18

5 Representation of how a decision tree may segment multi-class data sets [2]. Figure reproduced from the Python Data Science Handbook under the Creative Commons License. 19

6 An example of naive random over-sampling. Here, the blue circles are the majority class, and random-oversampling is used to duplicate arbitrary points from the minority class, red triangle, until the classes have equal amounts of data points. 25

7 An example of SMOTE over-sampling. The blue circles are the majority class, and SMOTE is used to generate synthetic data points in between observations from the minority class, red triangle, until the classes have equal amounts of data points. 26

8 An example of ADASYN over-sampling. The blue circles are the majority class, and ADASYN is used to generate synthetic data points in between difficult to classify observations from the minority class, red triangle, until the classes have equal amounts of data points. 27

9 An example ANN. Input data is passed to the first layer of nodes, which act on the data and pass it to successive layers. Image provided by Stanford CS class CS231n under under the MIT use license [3]. 29

Figure Page

10 A representation of a convolutional neural network. Image tensor data is passed to the first layer of nodes, which act on the data and pass it to successive layers. Image provided by Stanford CS class CS231n under the MIT use license [3]. 30

11 Correlation plot of the features used from the LANL lexicon. It is shown that several particle traits are not observed in the dataset. These are represented as white bars. Dark squares imply strongly anti-correlated features, while light blue denotes strong correlation. 35

12 Model architecture that was used throughout the project. Final output layer used two nodes with a softmax activation. A Nadam optimizer was used for all tests. 40

13 Distribution of particles for each Q passed into the machine learning algorithms. 41

14 An ordered representation of the normalized feature importance on Q-value classification from the GradientBoostingClassifier. 44

15 An example prediction array. After training, machine learning classifiers output the probability that a particle belongs to each Q. The first element in the array is the chance that the test particle belongs to the first Q-value, and so on. 45

16 A step-wise comparison of particle morphology occurrence between creation condition parameters. In this case precipitation temperature [°C] is measured against particle angularity. 46

17 Particle class distribution for final product groups. 47

18 An ordered representation of the normalized feature importance from the GradientBoostingClassifier for classification on pure final products. 49

19 Confusion matrix for the pure final product test set resulting from a trained GradientBoostingClassifier. 50

Figure Page

20 Accuracy results from using the scikit-learn predict function to correlate the mixed oxide final products to their pure counterparts. A random guess threshold line is included for ease of performance comparisons. 51

21 Adjusted, balanced accuracy results on the test set data for each classifier used. 54

22 A GradientBoostingClassifier misclassification. Here similar particles are displayed to explain possible confusion between particles. 56

23 A GradientBoostingClassifier misclassification. Here different particles are displayed to present possible limitations in the truncated lexicon. 57

24 A GradientBoostingClassifier misclassification. Here similar particles are displayed to explain possible confusion between particles and their creation conditions. 57

25 A GradientBoostingClassifier misclassification. Here similar particles are displayed to explain possible confusion between particles and their creation conditions. 58

26 A GradientBoostingClassifier misclassification. Here different particles are displayed to explore limitations of the truncated lexicon. 59

27 Model accuracy over 400 training epochs. 60

28 Model loss after training. Binary crossentropy was used for the step two prediction, as particles in the dataset were classified as either rounded/blocky or mixture, and never crystalline. 60

29 Flowchart for the particle nature classification. This first step allows for large particles with sub–components to be classified separately from their sub–particles [4]. 66

30 Continuation of the flowchart classification steps [4]. 66

31 The last step that has currently been applied to the particle dataset [4]. 67

32 First portion of the encoded creation conditions table. 68

Figure Page

33 Second portion of the encoded creation conditions table. 68

34 Last group in the encoded creation conditions table. 68

35 Processing method morphology feature occurrence for step 1 of the lexicon chart. 71

36 Processing method morphology feature occurrence for step 2 of the lexicon chart. 72

37 Processing method morphology feature occurrence for step 3 of the lexicon chart. 72

38 Processing method morphology feature occurrence for step 4 of the lexicon chart. 73

39 Processing method morphology feature occurrence for shape portion of step 4 in the lexicon chart. 74

40 Source material morphology feature occurrence for step 1 of the lexicon chart. 75

41 Source material morphology feature occurrence for step 2 of the lexicon chart. 75

42 Source material morphology feature occurrence for step 3 of the lexicon chart. 76

43 Source material morphology feature occurrence for step 4 of the lexicon chart. 76

44 Source material morphology feature occurrence for shape portion of step 4 in the lexicon chart. 77

45 Precipitation temperature morphology feature occurrence for step 1 of the lexicon chart. 78

46 Precipitation temperature morphology feature occurrence for step 2 of the lexicon chart. 78

47 Precipitation temperature morphology feature occurrence for step 3 of the lexicon chart. 79

48 Precipitation temperature morphology feature occurrence for step 4 of the lexicon chart. 79

Figure Page

49 Precipitation temperature morphology feature occurrence for shape portion of step 4 in the lexicon chart. 80

50 Precipitation pH morphology feature occurrence for step 1 of the lexicon chart. 81

51 Precipitation pH morphology feature occurrence for step 2 of the lexicon chart. 81

52 Precipitation pH morphology feature occurrence for step 3 of the lexicon chart. 82

53 Precipitation pH morphology feature occurrence for step 4 of the lexicon chart. 82

54 Precipitation pH morphology feature occurrence for shape portion of step 4 in the lexicon chart. 83

55 Additive morphology feature occurrence for step 1 of the lexicon chart. 84

56 Additive morphology feature occurrence for step 2 of the lexicon chart. 84

57 Additive morphology feature occurrence for step 3 of the lexicon chart. 85

58 Additive morphology feature occurrence for step 4 of the lexicon chart. 85

59 Additive morphology feature occurrence for shape portion of step 4 in the lexicon chart. 86

60 Calcination temperature morphology feature occurrence for step 1 of the lexicon chart. 87

61 Calcination temperature morphology feature occurrence for step 2 of the lexicon chart. 87

62 Calcination temperature morphology feature occurrence for step 3 of the lexicon chart. 88

63 Calcination temperature morphology feature occurrence for step 4 of the lexicon chart. 88

Figure Page

64 Calcination temperature morphology feature occurrence for shape portion of step 4 in the lexicon chart. 89

65 Final product morphology feature occurrence for step 1 of the lexicon chart. 90

66 Final product morphology feature occurrence for step 2 of the lexicon chart. 90

67 Final product morphology feature occurrence for step 3 of the lexicon chart. 91

68 Final product morphology feature occurrence for step 4 of the lexicon chart. 91

69 Final product morphology feature occurrence for shape portion of step 4 in the lexicon chart. 92

List of Tables

Table Page

1 Uranium enrichment levels and their various uses. 6

2 Example representation of the one-hot encoded particle dataset. Particle identifiers are in the leftmost column with the numbers following the Q denoting the project, and the numbers following the C describing the particle. 11

3 A variety of machine learning classifiers were used throughout the project to see which performed best on the particle morphology dataset. 31

4 Example representation of the one-hot encoded particle dataset after pre-processing techniques have been applied. 34

5 SMOTE classification performance. 42

6 ADASYN classification performance. 42

7 Performance comparison of validation set accuracy between oversampled and non-oversampled datasets. 43

8 SMOTE classification performance on pure final products. 47

9 A survey of the complete final products in the synthetic particle creation process. 48

10 An overview of the values used to hyperparameter tune the GradientBoostingClassifier on the final products dataset. 52

11 An overview of the values used to hyperparameter tune the KneighborsClassifier on the final products dataset. 53

12 Final product classification results from the standard dataset compared to variance reduced datasets. Accuracy values are averaged over 5 random splits. 55

13 Particle statistics for the various Qs. 69

14 The class distribution for the source materials creation condition. 69

Table Page

15 The class distribution for the precipitation temperature creation condition. 70

16 The class distribution for the precipitation pH creation condition. 70

17 The class distribution for the additive creation condition. 70

18 The class distribution for the calcination temperature creation condition. 70

19 The class distribution for the final product creation condition. 71

20 Commonly used terms throughout this thesis and their definitions. 93

A Machine Learning Approach to Characterizing Particle Morphology in Nuclear Forensics

1. Introduction

1.1 Motivation

The first nuclear weapon detonated forever changed the way that war was conducted. Political superpowers around the world were spurred into pursuing their own nuclear weapons programs. Over the next few decades following the Trinity Test in 1945, countries began proving their own technological prowess, with the Soviet Union testing their nuclear weapons in 1949, followed shortly thereafter by the United Kingdom. The 1960s saw France and China testing kiloton class nuclear weapons, and India followed in their steps by proving weapons capabilities in the 1970s. With most major world powers having obtained offensive nuclear capabilities, a new status quo emerged with these weapons exerting enormous political influence. As worldwide technological capabilities progressed, additional nation-states sought to expand their offensive portfolio by acquiring nuclear weapons. Continual research and development has expanded not only the field directly involved in creating said weapons, but has also created new fields of research entirely.

One such field that emerged alongside the advent of nuclear weapons was that of nuclear forensics, which encompasses the branch of science that provides critical analysis of nuclear materials and process signatures before or after detonation. Through these techniques, it is possible to identify signatures indicative of the material's processing history and origin. Nuclear forensic analysis can result in a deep characterization of the level of logistic and technological sophistication that gave rise to the material in question. The field itself has existed since the inception of the nuclear age, and is a major cornerstone of the International Atomic Energy Agency (IAEA), which aims to promote the safe and peaceful use of nuclear energy. A full forensics investigation into nuclear activities requires aggregating data points from many different sources and experiments, so as to combine them to produce a cohesive theory of the sample's origin and potential uses.

The study of nuclear forensics expanded after the collapse of the Soviet Union, when a number of cases of illicit trafficking of nuclear materials came to light, including materials directly usable in nuclear weapons [5]. The discovery of an undeclared nuclear weapons program in Iraq after the 1990 - 91 Gulf War further reinforced the need for more knowledge about the nuclear capabilities of other countries throughout the world. Nuclear forensic results allowed policy makers and state officials to be informed of the status of nuclear developments both at home and abroad. Recognizing the importance of verifying the proper use and storage of fissile materials created the basis of several international treaties. This has shaped policy in a particular area known as nuclear non-proliferation. Nuclear facilities across the globe are owned and managed by a mix of government and private organizations, but most have additional oversight as a result of the rules and regulations agreed upon from landmark non-proliferation treaties [6]. The Treaty on the Non-Proliferation of Nuclear Weapons, commonly known as the NPT, was the driving force behind many of these changes, and directed its participant nations to follow the safeguards and additional protocols created by the IAEA.

1.2 Background

The IAEA and many other forensics related organizations can use several different approaches to characterize a sample during a typical analysis process. These methods can be broadly placed into two categories: destructive and non-destructive. Non-destructive tests are often surface level experiments that preserve the integrity of the sample. Destructive tests are those that probe the material at the cost of consuming the sample itself. Of the two, destructive testing is typically regarded as the most powerful category of forensics experiments, as it provides more in-depth information about the basic composition of the materials under scrutiny. However, there is a trade-off that occurs, as destructive tests typically require longer sample preparation times than non-destructive tests, and remove the sample from any possible future use. As such, a common forensics pipeline involves conducting multiple non-destructive tests first, in order to extract as much information about the sample before its consumption in a destructive test [7].

One such non-destructive method is Scanning Electron Microscopy (SEM). An SEM is an instrument that uses a focused beam of high-energy electrons to image the surface of a sample at small distance scales [8]. The device works by directing an electron beam onto a material in a raster pattern, and capturing the resultant secondary and backscattered electrons. These are then collected and converted to gray-scale values to form an image. The difference in energy and yield of captured electrons allow the SEM to operate in BSE mode for backscattered electrons, or SE mode for secondary electrons. Of the two, SE mode is most often used due to its increased resolution over BSE mode. SEMs feature a magnification in the range of 100 - 1,000,000 which allows them to image down to the nanometer scale, a useful range that allows analysts to capture important details on the smallest of forensic samples [9].

Of all the elements on the periodic table, uranium is one of the strongest indicators of a nuclear weapons program. It can be found all throughout the globe, with average uranium concentrations in the Earth's crust of roughly 3 parts per million (ppm) by weight. Substantial uranium mining and extraction facilities can be found in many countries, including Australia, Canada, Kazakhstan, Namibia, Niger, Russia, South Africa, the United States, and Uzbekistan [10]. As such, it is important to be able to distinguish between uranium production intended for peaceful and non-peaceful purposes. The isotope of uranium that is useful in controlled nuclear reactions is far more rare than what occurs in nature: ^{235}U forms just 0.72% of mass of natural uranium worldwide. Converting natural uranium extracted from the earth into a useful form is not an easy endeavor; the need to efficiently produce uranium fuel created a large and resource intensive process known as the nuclear fuel cycle. The first step is to mine and extract the material, as shown in Figure 1. However, the uranium is not immediately useful upon mining, as it is most commonly found in the form of U_3O_8, which must be separated from rock via crushing and acidic leaching. This isolates the U_3O_8 and allows it to be stored in a dry powdered form known as yellowcake.

The next step in the process involves preparing the uranium for enrichment so as to accommodate the design needs of the particular reactor it will be used in. The Canada Deuturium Uranium (CANDU) reactor is of note here, because all operating reactors currently use natural uranium fuel. This design is very flexible and also allows for the use of slightly enriched uranium, recovered uranium, mixed oxide fuel, thorium fuels, and others. These different types of fuel can be introduced to the CANDU with few or no hardware changes, and allows the managing party to tailor the reactor to their own needs. CANDUs are of interest in the field of nuclear non-proliferation due to their ability to conduct on-line refueling. In this manner, Pu enriched nuclear material

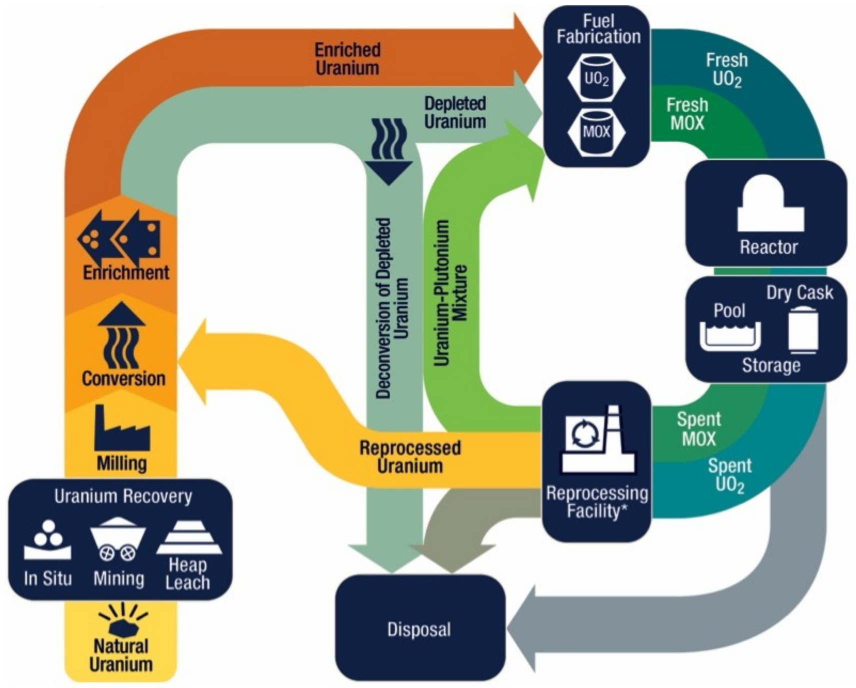


Figure 1. Overview of the nuclear fuel cycle from beginning to end, with the inclusion of optional reprocessing steps [1].

may be collected without interrupting reactor burn cycles. However, the IAEA and cooperating nations have included various safeguard measures for CANDU reactor use, as fuel can proceed directly from conversion to UO_2 fuel fabrication. These precautions have served to make each generation of CANDU safer than the last [11].

For the rest of the reactor types, uranium must be enriched in ^{235}U to a certain level. There are a myriad of ways to do this, but most of these processes rely on a gaseous form of uranium to allow for isotope separation of ^{235}U and ^{238}U. One widely used conversion route is to transform solid U_3O_8 into gaseous UF_6. Its widespread use as an enrichment precursor is attributed to the fact that UF_6 forms as a gas at lower temperatures than any other uranium compound [12]. In addition, fluorine has only a single stable isotope, which is important because many enrichment techniques rely on mass based processes. This is not to say that it is without downsides, as UF_6 is also highly toxic and corrosive to most metals. Once converted into a gaseous form, uranium can be passed through several different separation methods in order to achieve a specific level of enrichment. The intended use of the final uranium product is strongly correlated with the level of enrichment, so this step of the fuel cycle is closely monitored both in-house and by outside agencies like the IAEA, when necessary.

The first technology developed for the enrichment of uranium was the gaseous diffusion method. This technique was the workhorse of the U.S.'s nuclear program during the second World War, but has since fallen out of favor due to its immense demands on the electrical grid. Worldwide, most programs have moved to using centrifuge technology. This involves feeding UF_6 into a gas centrifuge and spinning the device at high speeds, thus separating the isotopes via the centrifugal force applied to the molecules. The uranium hexaflouride molecules containing the lighter ^{235}U atoms collect closer to the center of the chamber, and are siphoned off and fed into further centrifuge stages. In this manner, a large, interconnected cascade of centrifuges can enrich uranium up to the required percentages.

Table 1. Uranium enrichment levels and their various uses.

Enrichment Level (Weight % U-235)	Use
Depleted < 0.7%	Counter-weights, radiation shielding armor plating, armor-piercing projectiles
Natural 0.7%	Fuel for graphite-moderated reactors and heavy water-moderated reactors such as the CANDU
Low Enriched 3-5%	Light water reactor fuel
High Enriched >20%	Research reactor fuel, nuclear weapons
Weapons Grade >90%	Nuclear weapons

Uranium that is enriched up to 20% ^{235}U is known as low enriched uranium (LEU), and forms the category of material used as fuel for nuclear power reactors [13]. Typical enrichment levels for power generating nuclear reactors hover around 5%. Reactors used for research purposes have ^{235}U assays ranging from 20% all the way up to 93% which places them in the category of highly enriched uranium (HEU). At the highest levels of enrichment, uranium with a ^{235}U assay greater than 90% is considered to be weapons-grade (WGU) [13]. Of these different varieties, the highest risk for proliferation concern belongs to HEU. The IAEA has outlined significant quantities (SQ) of various actinides, which represent the approximate amount of material for which the manufacture of a nuclear device cannot be ignored [14]. The SQ for HEU is 25kg, and takes into account the unavoidable losses experienced in conversion and manufacturing processes. HEU is not the only enrichment level listed as a SQ, however. LEU is classified as an indirect use material, for it can still be converted to WGU after additional processing. This demonstrates that large quantities of uranium at any enrichment level can serve as serious security threats and must be closely monitored in some way. But it is not just fresh uranium that must be considered, as nuclear materials have a unique trait that allows for their use after being spent.

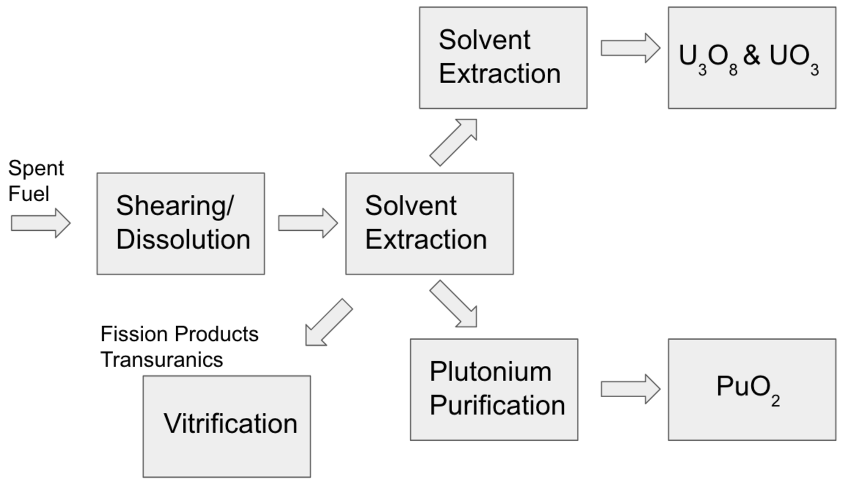


Figure 2. Simplified flowchart for the reprocessing of spent fuel.

A technique called reprocessing allows for recovery of fissile and fertile material from depleted fuel so as to make a new batch for future use. Russia, China, Japan, and several countries in Europe have reprocessing policies or programs in place. Notably absent from this list is the United States, which ended its reprocessing program in 1977 despite inventing the most commonly used reprocessing technique. Originally designed to recover plutonium from spent fuel, the Plutonium Uranium Recovery by EXtraction (PUREX) process achieves the separation of plutonium from uranium while also removing active fission products [15]. Spent uranium oxide fuel from thermal reactors contains around 81 to 83 weight % uranium and roughly 4 to 6 weight % plutonium. These quantities can vary depending on reactor burn times and flux profiles, and can be changed to maximize uranium or plutonium production. As such, it is important to verify whether a plant is reprocessing plutonium for mixed oxide (MOX) fuel use, or as base material for nuclear weapons. Reprocessed uranium (RepU) is typically held in intermediate storage in the form of UO_3 or uranyl nitrate hexahydrate (UNH) which is then converted into U_3O_8 through use of ammonium diuranate (ADU). An overview of the steps in reprocessing can be seen in Figure 2.

1.3 Problem

The broad research objective for this work is: *Given a set of SEM images of uranium bearing particulate matter created under well known conditions, can machine learning techniques be used to answer nuclear forensic questions about the dataset?* This research effort represents an attempt at using modern computational and statistical techniques to discover underlying trends in a synthetic reference dataset. The research question seeks to address several different problem areas related to the analysis of small–scale particles in a typical nuclear forensics process.

1.4 Questions & Hypothesis

The main research question contains a variety of sub–problems, which are detailed below:

- **Is it possible to distinguish among the project numbers in the syn-**

thetic dataset? Project numbers denote a specific grouping of creation conditions belonging to a program, location, or group. Being able to tell samples apart based on project number association implies that the morphology of a particle can tie it back to unique creation conditions.

- **What are the most useful morphological features of a sample particle?** There are many different ways a particle can be classified, but if only a small portion of features can serve to tie a particle with its creation conditions, then time spent analyzing a particle can be greatly reduced. This could also imply that certain features are more strongly associated with specific creation conditions.
- **Can this dataset form an adequate reference for real life forensics data?** The idea behind the synthetic particles was to create samples under well controlled laboratory conditions. Whether or not these conditions can be associated with real–world data remains an open question.
- **Can any correlation be determined to associate a particle's morphological features with the chemistry conditions under which it was formed?** If the physical features of a particle can be associated with the complex processes that created it, then a deep characterization of a sample's history would be possible exclusively through the use of non-destructive imaging techniques.

1.5 Approach

Experimental data is stored in the form of high-resolution grayscale tiff files, as shown in Figure 3. In total the dataset forms a catalog of 3,000 SEM images with over 700 unique synthetic particles. The dataset was created over multiple months, and uses a unique project number for each batch of particles created. AFIT researchers have converted these images into binary strings via a process laid out in a paper from Los Alamos National Labs (LANL) [4]. These binary strings represent a flow chart encoding of the descriptive features of each particle, and form a standardized way to compare particles against each other. In machine learning jargon, this is known as one-hot encoding: a process where observed object features are given values of 1, and absent features are assigned 0. Not all the available images could be converted

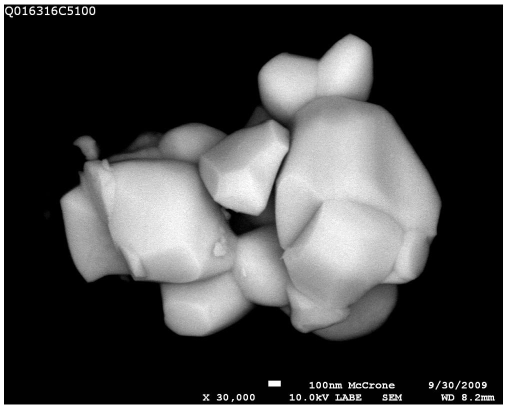


Figure 3. Example synthetic particle with unique project identifier on upper left.

to one-hot encoded strings, however. Some images were blurry and low quality, while others contained too many varieties of particles to consistently classify. These entries were listed as such in the data files that contained the strings, and were simply left as rows of all zeros. These entries were ignored for the entire machine learning process. As a result, the dataset fell to a total of 659 unique particles. A full list of the creation conditions for the samples can be found in Appendix B in Tables 32 through 34. After being passed through the LANL encoding process, the final dataset resembled Table 2, with each particle being assigned a string consisting of ones for features observed and zeros for features not present on the particle.

Table 2. Example representation of the one-hot encoded particle dataset. Particle identifiers are in the leftmost column with the numbers following the Q denoting the project, and the numbers following the C describing the particle.

Q-Identifier	Individual Particle	Congolomerate	Agglomerate
Q016316C10001	0	1	0
Q016316C10002	0	0	1
Q016316C10003	1	0	0

1.6 Assumptions & Limitations

Like many scientific experiments, a large portion of the analysis process involves understanding the data: how it was collected, where the data is sufficient, where it is lacking, and so forth. Every nuclear forensics investigation attempts to use all of the information at hand to produce a well thought out interpretation of the sample's origin, significance, or composition. This research project is no different, and any results derived from the dataset are limited by the variety of samples provided. This first limitation diminished the full extent of the dataset, but not severely enough that statistical learning could not be carried out. Particle one-hot encoding was carried out according to the process established by the LANL paper which outlined 11 total steps for particle morphology classification. In the interests of time, only the first four steps of morphological characterization have been applied. It is assumed that machine learning models can correctly distinguish particles based on their creation conditions with only four out of eleven total steps of the LANL lexicon. This assumption will be tested during the course of the data analysis. Despite having less than half of the steps possible, the amount of different inputs for each sample with just four morphological categories is still very high and the amount of difference from sample to sample may still allow for a machine learning model to distinguish between two samples of interest. If additional morphological lexicon steps are to be added in a future extension of the project, another limitation will be introduced. Increasing classification parameters of a particle could lead to decreasing prediction accuracy.

This is a common problem in the field of machine learning known as the curse of dimensionality. First coined by Richard E. Bellman, it refers to the problems that arise when attempting to analyze data in high dimensional spaces [16]. For a dataset of fixed size, increasing the parameter space, in this case the classification steps, results in an increasingly higher dimensional dataset which tends towards high sparsity due to the non-continuous binary nature of the data, and thus introduces a more difficult parameter space operate on.

2. Theory

2.1 Uranium Bearing Particle Formation

A common line of thought in the field of nuclear forensics is that several different processing parameters such as chemical composition of feed material, calcination temperature and duration, and precipitation conditions can all affect the resultant morphology of product particles. If these effects can be studied and well understood, then physical characteristics of samples such as shape, size, and surface characteristics could possibly be associated with unique processing conditions. This allows for a means of uncovering the techniques and capabilities of the process that produced the samples in question. In order to understand how morphological features may represent sample creation conditions, it is useful to review the wide variety of techniques used in uranium refining.

A refinery commonly accepts uranium ores made of uranium oxides or uranium diuranate. The first step in the refining process is the dissolution of uranium in nitric acid. This produces an aqueous solution of uranyl nitrate hexahydrate: $UO_2(NO_3)_2 \cdot 6H_2O$, which contains excess nitric acid and various metallic impurities. The next step in the purification process is the separation of uranyl nitrate from other metallic impurities via solvent extraction. Most uranium refineries use tributyl phosphate (TBP) dissolved in an inert hydrocarbon diluent as a solvent. This is due to its useful properties: it is nonvolatile, chemically stable, and selective for uranium [17]. From here a selected ore may go through a variety of processes depending on the particular method used by a uranium refinery, but main steps involve passing the sample through a dissolution stage using 40% HNO_3, then onto a scrubbing section, and finally a uranium stripping column, which leaves a uranyl nitrate (UNH) solution. This aqueous solution then is converted to UO_3 in two steps: concentration and denitration. First, UNH is evaporated in a boil-down tank to a liquid with a composition similar to hexahydrate. Next, the nitrate is removed through the use of a heated pot, a fluidized bed, or a stirred and heated trough, all of which results in UO_3 as a final product. UO_3 can be converted to UO_2 through reduction with cracked ammonia gas ($3H_2 : 1N_2$) around 590° C in two fluidized reactors with counter-current flowing solids and gases. Any exhaust gases are filtered to remove dust that has become incorporated into the flow. The gases are then cooled to condense steam as in $UO_3 + H_2 \rightarrow UO_2 + H_2O$. This step must be performed carefully to prevent sintering effects on the oxides, in order to obtain a product that will react well with HF in preparation for enrichment. If the UO_2 is to be used as fuel in a CANDU type reactor, then the reduction is done at a higher temperature, so as to make a denser fuel [17].

Hydroflourination is the next part of the process, which converts UO_2 to UF_4 via the exothermic reaction $UO_2 + 4HF \rightleftharpoons UF_4 + 2H_2O$. The United States' uranium plants achieve this through use of two stirred fluidized-bed reactors in series, with a counter-current flow of solids and gases [17]. A hot bed which runs at 300°C is fed UO_2 and partially converts it to UF_4. A hotter bed running at 500°C is fed anhydrous HF and partially converted UO_2, converting more the 95% of the UO_2 to UF_4. Any effluent gases are filtered and cooled, so as to remove entrained solids and to condense aqueous HF. The final step in the process is to fluorinate UF_4 to UF_6. The U.S. Department of Energy (DOE) has two plants that do this through fluorine reactions in tower reactors [17]. Solid UF_4 and a small excess of fluorine gas are fed into the top of a monel reactor with walls cooled to 500°C. A majority of the UF_4 reacts instantly with a flame temperature of approximately 1600°C. Small portions of unreacted UF_4 and uranium oxides are removed from the tower and passed to the previous step. Effluent gases containing UF_6, fluorine, and diluent gases are cooled to 150°C and passed through filters to remove any trapped solids. Cold traps cooled at -10°C then condense most of the UF_6 into a solid. Any residual fluorine gas leaving the cold trap is then removed by an additional reaction with UF_4 in a fluid bed reactor. This forms UF_6 and intermediate fluorides like UF_5. Exhaust gases from this reactor go into another cold trap kept at -50°C which condenses most of the UF_6. The very last traces of UF_6 are removed by a second fluid bed reactor, which reduces the presence of UF_6 in exhaust gases to less than 10 ppm. UF_6 produced with these techniques is very pure, with total UF_6 output exceeding 99.5% purity [17].

Enriched or depleted uranium is usually produced in the form of UF_6, but is used as metallic uranium or UO_2 in power reactors. This requires the conversion of UF_6 to either UF_4 or UO_2. For the former, UF_6 is converted to UF_4 through vapor-phase reduction with hydrogen. Because the heat of reaction is small, the mixture must be heated by some other process. In small chemical reactors, this is commonly provided by reacting fluorine with hydrogen. In contrast, the conversion of UF_6 to UO_2 has three different possible routes. In the first, UF_6 is reduced to UF_4 and then hydrolyzed by steam:

$$UF_4 + 2H_2O \rightarrow UO_2 + 4HF, \tag{1}$$

which is the reverse of the reaction used to make UF_4. In the second process, UF_6 is hydrolyzed to UO_2F_2 by solution in water. Ammonia is then added to precipitate ammonium diuranate as shown below. The diuranate is reduced to UO_2 with hydrogen as a final step:

$$2UO_2F_2 + 6NH_4OH \rightarrow 4NH_4F + (NH_4)_2U_2O_7 + 3H_2O. \tag{2}$$

In the third method, known as the ammonium uranyl carbonate (AUC) process, streams of gaseous UF_6, CO_2 and NH_3 are fed batchwise into water that has been de-mineralized. This precipitates $(NH_4)_4UO_2(CO_3)_3$, which is then converted to UO_2 through contact with steam and hydrogen in a fluidized bed at 500°C. This allows for the recovery of UO_2 after separation from CO_2 and NH_3 [17].

2.2 Nuclear Forensic Analysis

A sample that is under investigation in a typical nuclear forensics process may have gone through any of the aforementioned processes to arrive at its final state. As such, a wide variety of chemical, physical and analytical techniques are required to properly discern the methods used to create a sample of interest. Specifically, the materials are searched for "signatures", which describe material characteristics like isotopic abundances, elemental concentration and physical morphology. For nuclear materials, signatures can be created, destroyed or modified at each step in the fuel cycle. Despite this, each stage of the process generally results in materials with unique sample traits, known as process signatures [18]. A multitude of instruments, employing both destructive and non–destructive techniques are used to extract these signatures. In reference to the dataset used for analysis, the particulate samples underwent a few different processes before undergoing SEM analysis. For small–scale particulate, one of four methods is generally employed to process samples: direct pick, sonication, ashing and fission track. In fission track analysis, samples are detached from the originating medium and dispersed onto a thin film of Lexan polycarbonate substrate. The films are then neutron irradiated, with fissile materials leaving behind fission tracks in the surrounding media [19]. This clues analysts into the presence of fissile material, which allows specific particulate to be selected based upon the quantity of fission tracks emanating from the sample.

In sonication, the sample is deposited into a tube containing a solvent. It is then placed into an ultrasonicator for some amount of time. Material that is released from the sample media falls to the bottom and is extracted as microscopic particulate. Ashing involves placing a sample into a crucible and burning off carbon based material. The bottom of the crucible contains the remaining material, which is then washed and dispersed for further analysis. Lastly, the most basic of the techniques: direct pick. Double sided carbon sticky-tape on a SEM stub is used to randomly sample the surface of sample media. Samples are then dispersed on a surface to decrease the particle density before further analysis.

2.3 Machine Learning Foundations

The scope of this project involves the use of machine learning classification algorithms over standard regression models. This is because our intended response variables are qualitative instead of quantitative, or more simply stated as labels instead of numerical quantities. As an example, eye color is qualitative, as it takes on non-numerical values such as green, brown or blue. Qualitative variables are often interchangeably referred to as categorical variables. The process behind predicting a categorical response is known as classification, of which there are many different techniques. As with linear regression, a classification algorithm takes a set of training observations $(x_1, y_1), ..., (x_n, y_n)$ to build parameters for the purpose of creating a response prediction. In a two class setting, the machine learning algorithm predicts $p(X) = p(Y = 1|X)$ or $p(X) = p(Y = 0|X)$ where 0 and 1 correspond to numerically converted representations of the input classes, and variable Y denotes the output labels of each data point. An example classification process would be to take a single data point from a plot in Figure 4, with coordinates (x_1, x_2) and label (y), pass it through a classification model, and predict the probability that it belonged to each class. The class with the highest probability is then assigned as the predicted response, and is compared to the actual class, y for purposes of error calculations.

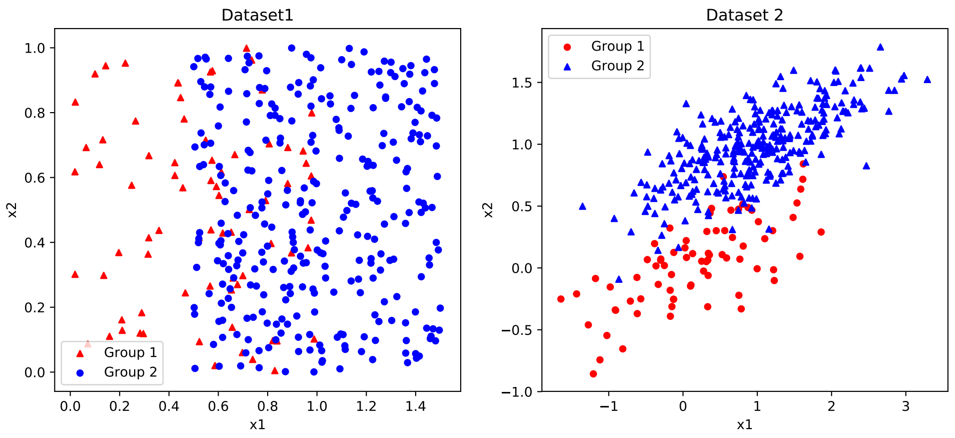


Figure 4. Example two-class datasets that could be used for classification tasks.

2.4 Decision Trees

Tree-based methods are a subset of machine learning techniques that are used to segment the data-space into a number of smaller regions. The mean or the mode of the observations in these regions are then used to make predictions about the dataset, such as assigning classes or probabilities [20]. Standard tree-based methods are simple and allow for ease of interpretation, but as a result are not able to compete with other popular techniques in the field. Additional techniques such as bagging, random forests, and boosting have been developed that have greatly improved the performance of tree-based methods and allow them to be a viable candidate for an accurate machine learning algorithm. Each of these methods creates multiple decision trees which can be combined to yield a single, averaged, consensus prediction.

The regions shown in Figure 5 denote terminal nodes or leaves of the decision tree. These refer to areas where splits are no longer occurring, and are the final step in the decision making process [21]. For a classification tree the predicted response, y, is simply the most commonly occurring class of observations within any given region

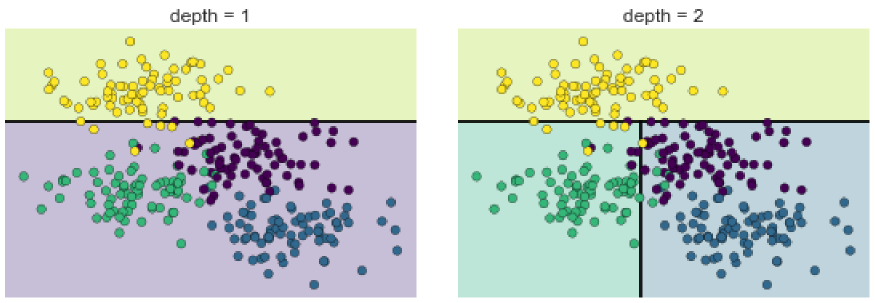


Figure 5. Representation of how a decision tree may segment multi-class data sets [2]. Figure reproduced from the Python Data Science Handbook under the Creative Commons License.

of interest. In order to arrive at this point, the tree must first be built, in accordance with broad tree–building rules:

1. Start with the full dataset X with features p and output labels y, call this the parent node.
2. Split the parent node at the feature p_i to minimize the sum of the child node impurities (maximize homogeneity).
3. Assign samples to new child nodes.
4. Stop if child nodes are homogeneous or some early stopping criteria is satisfied. If not, repeat steps 2 and 3 for each new child node.

For every observation that falls into a region of interest, R, the same prediction is made, which is the most commonly occurring class for the training observations contained in R. The splits that form segmented regions seek to minimize some error parameter in order to increase the overall accuracy. For classification trees, the metric used is the classification error rate which is defined as the fraction of the training observations in that region that do not belong to the most common class.

$$E_m = 1 - \underset{k}{max}(\hat{p}_{mk}) \tag{3}$$

Where $\hat{p}_{mk}$ represents the proportion of training observations in the mth region that are from the kth class. In practice, the classification error is not a useful condition for growing trees, and has been replaced by a metric known as the Gini index:

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}), \tag{4}$$

which is a measure of total variance across K classes [20]. The Gini index takes on a small value when all of the $\hat{p}_{mk}$'s are close to zero or one, which means that the Gini index is minimized when all or most the observations in a region R_m belong to a single class.

2.5 Bootstrapping and Random Forests

Decision trees feature a drawback of tending to high variance or overfitting. This means that the tree models are very sensitive to small changes in the dataset, which could potentially lead to poor performance on new datasets or inputs. Bootstrap aggregation, also known as bagging, is a variance reduction technique used to great effect in decision trees.

Given a set of n independent observations $Z_1, Z_2, ..., Z_n$, each with variance σ^2, the variance of the mean $\bar{Z}$ is given by $\sigma^2 n$. Thus it is seen that averaging a set of observations reduces the variance. This allows for an increase in prediction accuracy via the process of taking many training sets from the dataset, building a new prediction model for each set and then averaging the many predictions:

$$\hat{f}_{ave}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}^b(x). \tag{5}$$

Where $\hat{f}^1(x), \hat{f}^2(x), ..., \hat{f}^B(x)$ are the B different predicted models that are averaged. Given the limits of acquiring data in a real world scenario, this is not often the most practical method. Instead, a technique called bootstrapping can be applied which involves taking repeated samples from a single training datatset, which generates B bootstrapped datasets. It is then possible to train a machine learning model on the bth bootstrapped training set in order to get $\hat{f}^{*b}(x)$ and then average all the predictions to obtain a single bootstrapped prediction:

$$\hat{f}_{bag}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}^{*b}(x). \tag{6}$$

Bootstrap aggregating (bagging) is applied to decision trees by the construction of B trees using B bootstrapped training sets, and then averaging the resulting predictions. This creates an effect that helps mitigate overfitting, as each individual tree has high variance, but through the process of averaging the trees, variance is reduced. Bagging can be applied to classification problems by recording the class predicted by each of the B different trees, and taking the majority vote [20]. The overall prediction of the tree is the most commonly occurring majority class among the B different predictions. The key takeaway is that bagging improves prediction accuracy at the expense of interpretability, as a single tree is no longer responsible for the outcome predictions.

2.6 Random Forests

Random forests provide a performance increase over decision trees via the introduction of a random element with the construction of every tree. As with the bagging process, a large amount of decision trees are built on bootstrapped training samples. The main difference arises when building the trees, as each time a split occurs, a random sample of m predictors from the full set of p predictors is chosen as the candidates to split on. Each split is restricted to choosing only one of the m randomly chosen predictors, which helps decorrelate the decision trees from one another. Every time a split is made, a new sample of m predictors is taken, with values of m typically around $m = \sqrt{p}$ [20]. What this means for the decision tree is that at each split made, the tree-building algorithm is not allowed to consider a majority of the dataset's predictors. Such a restriction allows the decision tree to avoid any overly strong predictors that may overwhelm other predictors in the dataset. This forces variety into the construction of the trees, which prevents predictions from the many bagged trees from being over-correlated. This is an effective technique because on average, $(p-m)/p$ of the splits will not consider the strong predictor, which gives the rest of the dataset a chance to influence the resulting tree architecture. The introduction of splits on random subsets of the predictor-space allows for reduced variance to be achieved through a combination of diverse tree models. The original application of this technique allowed each classifier to vote for a single class [22]. In contrast, Python's sci-kit learn implementation of random forests combines the many different classifiers by averaging their probabilistic predictions [23].

2.7 Boosted Trees

Decision trees and random forests can be improved upon through the use of boosting, a technique which aggregates weak learners, classifiers which have relatively poor performance levels, into a single classifier with arbitrarily high accuracy [24]. This differs from bagging, which builds many trees, all independent of each other. Boosting also does not involve bootstrap sampling; each tree is sequentially grown using information from previous trees, each on a modified version of the original dataset. This approach helps avoid the overfitting problem by learning the data slowly through the combination of a large number of decision trees $\hat{f}^1, \hat{f}^2, ..., \hat{f}^B$. Given a current model, the newest decision tree update is fit to the residuals of the current model instead of the outcome Y. The new decision tree is added to the fitted function and the residuals are then updated. Each individual tree can be rather small, with just a few terminal nodes, as controlled by the split parameter d. By fitting many small trees to the residual of the function, $\hat{f}$ is slowly improved in areas of poor performance. An additional parameter, λ, controls the parameter shrinkage, slowing the fitting process down even more to allow for a variety of trees to fit the residuals. In essence, a boosted tree algorithm operates in the following sequence [20]:

1. Set $\hat{f}(x) = 0$ and $r_i = y_i$ for all i in the training set.

2. For $b = 1, 2, ...B$ repeat:

   - Fit a tree $\hat{f}^b$ with d splits ($d+1$ terminal nodes) to the training data (X, r).
   - Update $\hat{f}$ by adding a shrunken version of the new tree:

$$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x). \tag{7}$$

   - Update the residuals:

$$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i). \tag{8}$$

3. Output the boosted model:

$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x). \tag{9}$$

The above describes the foundations for a boosted regression tree. A boosted classification tree is similar, but uses a negative multinomial log-likelihood loss function with N mutually exclusive classes, and constructs K least squares trees in each iteration [25]. Each tree is fit to its respective negative gradient vector g_{kb}:

$$g_{kb} = -[\frac{\partial L(y_i, f_1(x_i), ..., f_k(x_i))}{\partial f_k(x_i)}], \tag{10}$$

and is used to update the residuals in each step. A boosted classification tree differs from a regression tree in that it repeats step 2 above a K number of times at each iteration b, once for each data class. This results in an output at step 3 which contains K different tree expansions $f_{kB}, b = 1, 2, ...K$. These final trees produce classifications on the input data.

2.8 Imbalanced Datasets

Many different machine learning algorithms operate on two broad assumptions in order to make sensible predictions on a given dataset:

- Maximizing predictive accuracy is the goal of the algorithm
- During use, the algorithm will draw data from the same distribution as the training data

Because of the assumptions made, if a machine learning classifier is trained on an imbalanced datatset, predictions will be heavily skewed towards the majority class. As an example, if 99% of data belongs to a single class in a two-category classification problem, a learning algorithm will max out at 99% accuracy, as it will label every output as the majority class unless instructed otherwise [26]. In order to successfully learn on an imbalanced dataset, one of the previous assumptions will almost certainly be ignored. A common method for dealing with imbalanced datasets involves rebalancing the class representations. This can be done through upsampling techniques via duplication of minority samples, or downsampling via exclusion of majority samples. Through one or more of these techniques, a machine learning algorithm can be trained on a dataset with equal representations of all classes. This is of particular use when the minority classes contain too few data points to fully characterize the class of interest.

2.9 Over-Sampling Techniques

The first, and most basic over-sampling technique is naive random over-sampling. This generates new samples in under-represented classes by randomly sampling with replacement until the minority class has as many data points as the majority class. This is shown in Figure 6, where two classes are balanced by repeated draws to the minority class. This approach gives the machine learning model more examples to train on, ideally boosting predictive accuracy. A drawback to this technique is that the model's generalization abilities do not increase as much when compared to other methods because the new data points are simply repeated samples drawn from the minority class.

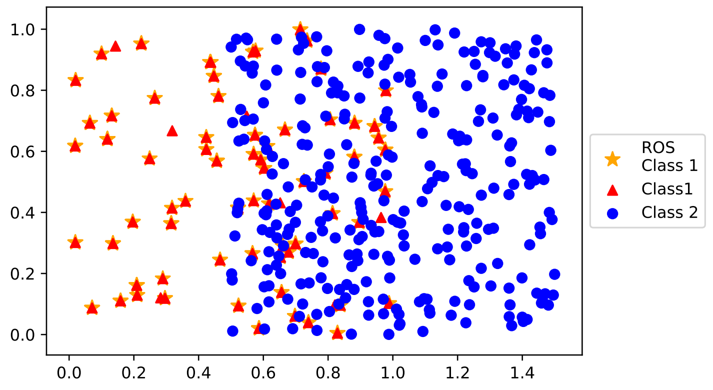


Figure 6. An example of naive random over-sampling. Here, the blue circles are the majority class, and random-oversampling is used to duplicate arbitrary points from the minority class, red triangle, until the classes have equal amounts of data points.

A more popular technique for oversampling is known as the Synthetic Minority Oversampling TEchnique (SMOTE), which is considered the de facto methodology in the framework of learning from imbalanced data [27]. Here, a minority class is over-sampled by the creation of synthetic data points. In this process, each minority sample is chosen along with its k minority class nearest neighbors. Samples are generated by taking the difference between the sample under consideration and its nearest neighbor. This difference is multiplied by a random number in the range of

(0,1), and then added back to the sample under consideration. Effectively, this causes the new synthetic data point to be randomly selected from the line segment joining the minority point and its k nearest neighbor. This forces the machine learning algorithm to create larger and less specific decision regions. These more general regions are now learned for the minority class samples which allows for a more equal representation against a majority class. The resultant effect is that decision trees are then able to generalize better over the scope of the entire feature-space, which allows for increased performance [28]. An example of how this process creates new data is shown in Figure 7.

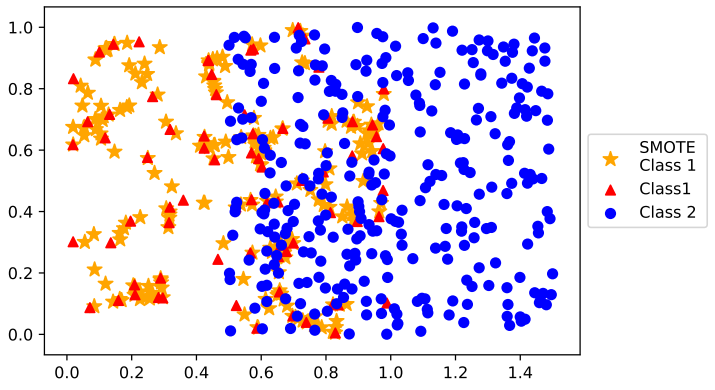


Figure 7. An example of SMOTE over-sampling. The blue circles are the majority class, and SMOTE is used to generate synthetic data points in between observations from the minority class, red triangle, until the classes have equal amounts of data points.

A more recent technique developed for oversampling is ADASYN, which is an ADAptive SYNthetic method for generating samples from a minority class. ADASYN differs from SMOTE in that it takes a weighted distribution for each minority class sample according to the difficulty in learning that sample. In short, more synthetic data points are generated near minority class samples that are harder to learn or frequently mis-classified. The manner in which they are placed is similar to SMOTE; new data points are placed on the line segment joining the minority point and its k nearest neighbor. Creating data in this manner improves the learning process in two ways: reduction of bias introduced by the class imbalance, and adaptively shifting the classification boundary towards difficult examples [29]. An example of how ADASYN generates new data points in the minority class can be seen in Figure 8.

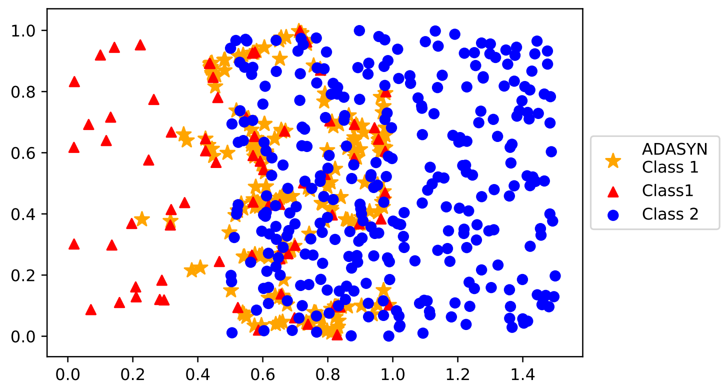


Figure 8. An example of ADASYN over-sampling. The blue circles are the majority class, and ADASYN is used to generate synthetic data points in between difficult to classify observations from the minority class, red triangle, until the classes have equal amounts of data points.

2.10 Balanced Accuracy

In binary and multi-class classification problems, using standard accuracy metrics is a concern when dealing with imbalanced datasets. In particular, the use of standard accuracy metrics may not reflect the full capacity of an algorithm, as a model trained to perform classification on a multi-class problem with an imbalanced dataset may produce a trained model that is biased towards the majority classes [30]. Reporting accuracy metrics on a test dataset that is imbalanced in the same manner may produce overly optimistic accuracy values at best, and exclusive predictions of the majority class at worst. Several efforts have been made to address the problem, namely those by Akbani et al, Chawla et al, and Japkowicz et al [28, 31, 32]. Resampling, for example, may be used to restore class balance by oversampling the minority classes or undersampling the majority classes. Another approach would be to modify the cost of mis-classifications such that no bias is acquired in the process of building a model.

While the aforementioned methods work to prevent model bias, they do not in general protect against optimistic accuracy reporting. One method used to overcome these limitations is the replacement of average accuracy with balanced accuracy, which is defined as the macro-averaged accuracy obtained on each class [33]. Without adjustment of values, the best score is one, and the worst is zero. Using adjusted=True in scikit-learn modifies the value range to [23]:

$$\frac{1}{1-N_c} \text{ to } 1, \tag{11}$$

where N_c is the number of classes. An adjusted balanced accuracy score less than zero indicates worse than random performance. The balanced accuracy values are calculated via the use of sample weights, where y_i is the true value of the i-th sample and w_i is the corresponding sample weight, then the adjusted sample weight is calculated:

$$\hat{w}_i = \frac{w_i}{\sum_j 1(y_j = y_i) w_j}, \tag{12}$$

where 1(x) is the indicator function which is equal to one if $(y_j = y_i)$ and zero otherwise. Given a predicted $\hat{y}_i$ for sample i, the balanced accuracy can be numerically defined as

$$\text{balanced acc}(\hat{y}, y, w) = \frac{1}{\sum \hat{w}_i} \sum_i 1(\hat{y}_i = y_i)\hat{w}_i, \tag{13}$$

with adjusted = True, balanced accuracy reports the relative increase from balanced_accuracy(y,0,w) $= \frac{1}{N_c}$.

2.11 A Brief Introduction To Neural Networks

An additional research effort throughout the course of the project involved taking the SEM images directly as an input dataset. Because neural networks were not the focus of the project, only a very top level introduction will be given. This type of machine learning effort was made possible because of the rapid rise of Artificial Neural Networks (ANNs) in the field of computer science. ANNs are machine learning systems that derive their basis from biological nervous systems, such as the human brain. These models consist of many interconnected computational nodes shown in Figure 9, and referred to as neurons [34]. These neurons contain weights that are collectively learned in order to optimize an output of the network. However, traditional ANNs struggle with the complexity that arises from image-based data. Here a different type of approach is preferred: a Convolutional Neural Network (CNN).

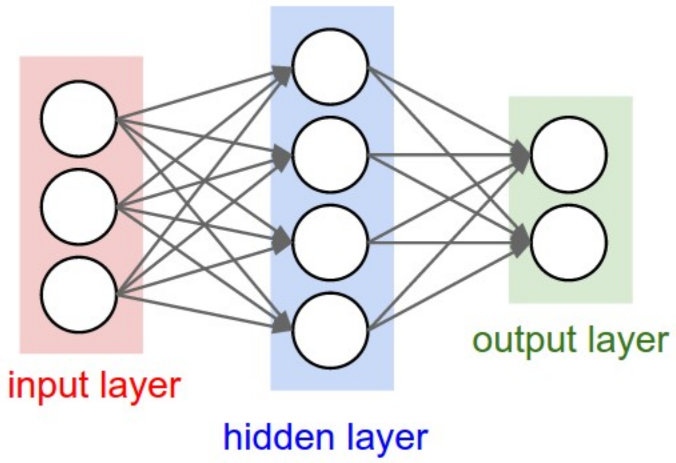


Figure 9. An example ANN. Input data is passed to the first layer of nodes, which act on the data and pass it to successive layers. Image provided by Stanford CS class CS231n under under the MIT use license [3].

CNNs are similar to ANNs in that they are constructed from layers of neurons that are optimized over the course of the learning process. As with ANNs, each neuron receives an input, performs an operation on it, and passes the result to the next layer. The reason that CNNs are much better than traditional ANNs is because of the architecture and style of neurons used. A typical input is a 3-dimensional image tensor with dimensions denoted by H rows, W columns, and 3 color channels. This input is fed to the network, which then considers small segments of the image at a time, in a similar fashion to the human eye. Within the CNN, there are three main types of layers used to process image data: convolutional layers, pooling layers, and fully connected layers. The convolutional layer, shown in Figure 10, modifies inputs via matrix multiplication and pass the resultant output to further layers. Pooling layers perform various forms of downsampling operations, thus reducing the amount of parameters the network needs to process. Finally, CNNs frequently terminate with fully connected layers, exactly like those used in ANNs and shown in Figure 9. Through use of the aforementioned layers, CNNs can take image-based data and transform the inputs layer by layer in order to produce outputs for regression and classification purposes.

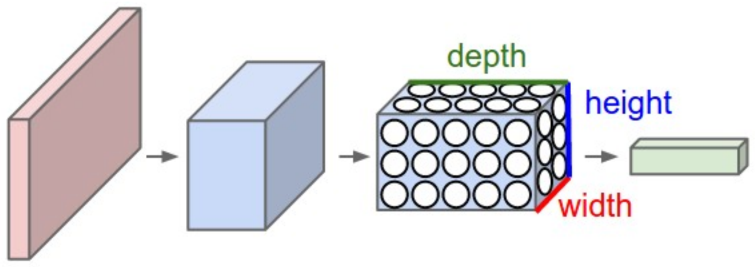


Figure 10. A representation of a convolutional neural network. Image tensor data is passed to the first layer of nodes, which act on the data and pass it to successive layers. Image provided by Stanford CS class CS231n under the MIT use license [3].

3. Methodology

3.1 Machine Learning Process

A standard machine learning task involves taking aggregated data and using it to make a prediction or classification on the dataset. This is possible through the application of statistical learning methods, which build a mathematical model from the input data. There are certain guidelines to be followed in order to ensure sound, scientific decisions can be made which lead to maximal model utility. As such, datasets in machine learning experiments are usually split into three portions: training, validation, and testing. Each describes different steps in the analysis process, and effective utilization of the three datasets is key to delivering a well thought out model. The training set is the first step in the process, and is usually the largest split of the three datasets. Here, machine learning techniques are applied to the training set in order to learn or train the parameters that allow for later predictive use. In this step, algorithms shown in Table 3 were applied in tandem so as to compare performance in later steps. This collection of algorithms was chosen due to its variety in classification methods, as well as its use of modern machine learning techniques. The relative strengths and weaknesses of certain classifiers could be covered by others on the list, which gave a good basis for choosing a main classifier later throughout the project.

Table 3. A variety of machine learning classifiers were used throughout the project to see which performed best on the particle morphology dataset.

Type of classifier	Classifiers
Nearest-neighbors based	KNeighborsClassifier
Support Vector	SupportVectorClassifier (SVC)
Tree-based	Decision Tree
Tree-based	RandomForest
Naive Bayes statistics	Bernoulli NaiveBayes
Bayes statistics	LinearDiscriminantAnalysis
Tree-based	GradientBoostingClassifier

The validation set is used to tune and compare performance of the algorithms that were trained in the previous step. Here, model hyperparameters may be changed and iterated over, in order to provide the best possible predictive performance on the validation set. The validation set allows for an unbiased look at model performance that results from being fit on the training set. Finally, the test set, sometimes called the holdout set, is the last step which is used to measure final model performance. It is kept hidden (held out), and decisions are not allowed to be made based on the performance of the model on the test set, as it is made to resemble real world data that the model must interpret. If a model has been properly trained to avoid issues with overfitting, and the test set is independently drawn from the same probability distribution as the training set, favorable results can be expected, presuming the dataset provided contains distinguishable patterns and thus provides favorable results.

Decision tree type algorithms were used as the main machine learning tools for much of the research project, as they are simple and useful for interpretation [20]. In addition, many additional techniques such as bagging, random forests and boosting make these algorithms flexible and capable of reaching high levels of accuracy. In particular, scikit-learn's GradientBoostingClassifier was used because it provided an accurate and effective off-the-shelf model capable of handling multi-class problems [23].

3.2 Nuclear Forensics Image Lexicon

The physical characteristics of nuclear materials can provide a preliminary method of determining the process history behind the sample. A commonly used tool in the forensics process is the SEM, which allows for characterization of sample texture, surface features, structure, size, and grain boundaries [35]. However, methods that allow for a standard comparison between experiments are few and far between. As a result, comparisons between images are often subjective, with each researcher having to establish their own methodology in order to draw conclusions [36–40].

A team from the Los Alamos National Laboratory designed a flowchart based methodology in order to overcome the limitations of subjective sample comparisons. A well laid out lexicon of defined terminology for characterization of morphological features is presented, which allows for a consistent, thorough description of a sample under scrutiny [4]. The lexicon itself is built upon terms commonly used in the study of geology, powder science, mineralogy, and crystallography [41, 42]. The flow chart structure begins with more general descriptors of particle morphology, and becomes increasingly specific as the steps progress. The process begins with classifying the types and quantities of particles, and ends with surface and texture features.

There are 11 total steps in the flow chart. The first step gives an overall assessment of the particle in the SEM image: it is categorized as individual, complex, sub-grains, and clumped or massive material. The following steps classify the overall particle morphology. Due to time limitations, only the first four steps in the lexicon flowchart were classified for each of the particles in the dataset. The charts for each step can be found in Appendix A.

3.3 Data Processing

For ease of analysis, the columns in the encoded dataset were renamed to a consistent naming scheme, such as S1F1 for “Step One, Feature 1”. The particle excel file contained a column of Q-string identifiers, which were initially listed in strings such as ‘Q019907C8020’. In order to classify this, the columns were stripped of the particle numbers so that only the project numbers were left, ex: Q019907. The project numbers were then passed through scikit-learn’s LabelEncoder which created 56 different classes of particle, 0 through 55. These numerical classes were treated as the truth-values and were used in the machine learning process in place of the project numbers. In addition, features that did not occcur in the dataset were identified from Figure 11 and had their columns removed from the input dataset. These consisted of S2F3 (crystalline), S4FE (fibrous), S4FF (ribbon-like), S4FH (straight), and S4FK (twisted). This allowed for dimensionality reduction without the need for extensive feature engineering. As a result of the pre-processing changes, the morphology dataset appeared as shown in Table 4 with a total of 659 rows and 26 columns of binary encoded data, each representing a unique isolated particle.

Table 4. Example representation of the one-hot encoded particle dataset after pre-processing techniques have been applied.

Class	S1F1	S1F2	S1F3	S2F1	S2F2	S3F1
1	0	1	0	0	1	0
1	0	0	1	0	1	0
1	1	0	0	1	0	0
2	1	0	0	0	0	1
2	0	1	0	0	0	1
			.			
			.			
			.			
55	0	0	1	0	1	0

A separate csv file was read in that contained columns of data for each Q: project number, source material, processing method, precipitation temperature, precipitation pH, additive, calcination temperature, and final product. These values were compared to the original data matrix and each particle was assigned its specific creation conditions. Processing method required additional steps, as there were often multiple processing methods within a single Q, so an alternate routine was developed to match individual particles to their respective processing method. Having all particles matched to individual creation conditions enabled analysis of particle statistics as well as majority and minority classes for each creation condition, the results of which can be found in Tables 13 - 18 from Appendix B. Each group of creation con-

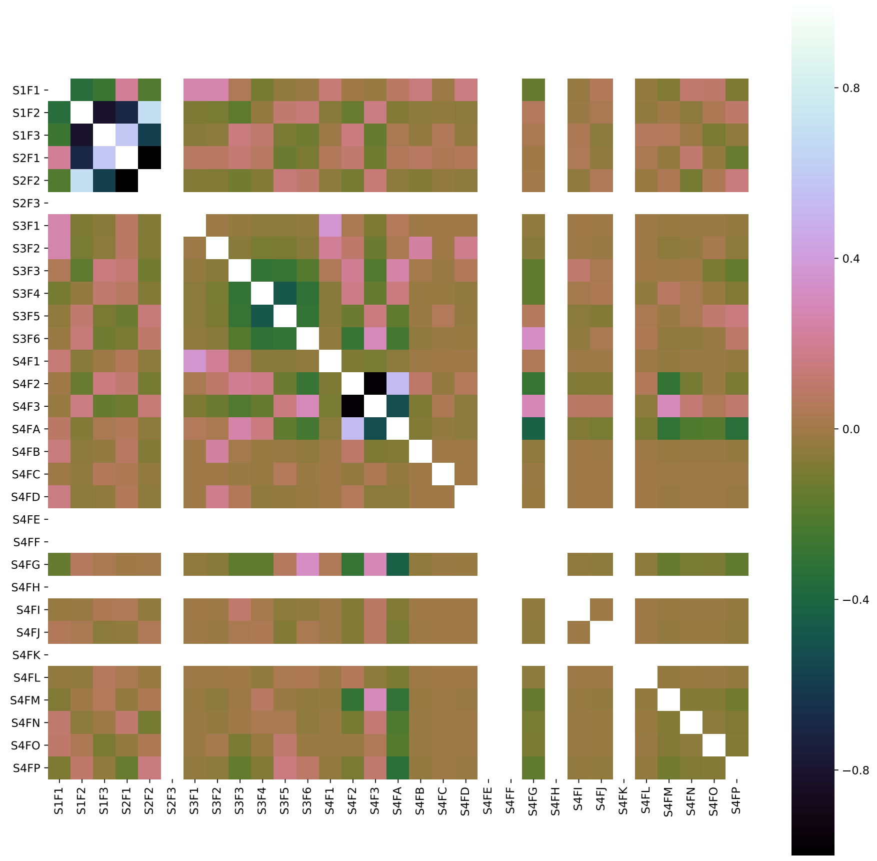


Figure 11. Correlation plot of the features used from the LANL lexicon. It is shown that several particle traits are not observed in the dataset. These are represented as white bars. Dark squares imply strongly anti-correlated features, while light blue denotes strong correlation.

ditions was passed through scikit-learn's LabelEncoder to convert the data into the appropriate format for machine learning algorithms. Now the encoded creation conditions were used as the input truth values for the learning process, and were passed into the algorithms along with the encoded binary strings. A variety of classifiers were trained on the data, and accuracy metrics were compared amongst algorithms to measure relative performance levels.

Feature engineering was not widely pursued throughout the project, as the aim was to preserve the integrity of the LANL lexicon and all the steps it contained. One method that was used briefly was variance reduction of columns. Scikit-learn's VarianceThreshold routine allows for a simple approach to feature selection, and works by removing all columns of data with variance levels that do not meet some threshold. By default it removes all zero-variance columns, more easily expressed as columns with the same value in all samples. This is useful on the one-hot encoded data, as features that are all zeros or all ones can easily be pruned from the dataset. One-hot datasets use Bernoulli random variables, which have a variance given by:

$$Var[X] = p\,(1 - p), \tag{14}$$

where p measures the ratio of zeros or ones to the total amount of data in a column. The new dataset with less columns could now be used in place of the original for training machine learning models. Several different variance thresholds were tested to check for possible accuracy gains through the project. To do so, for a given probability level, models were trained 5 times, with different splits of the dataset in each iteration. Results were averaged and compared between non-variance reduced datasets and those with variance reduction.

3.4 A Neural Network Approach

A large portion of the particle classification process involves a human in the loop, requiring each particle to be carefully analyzed by hand before being converted to a binary string format that may be useful to a machine learning classifier. A possible method of encoding the morphological features of each particle without requiring the constant presence of a human would be to use a neural network to directly read the image and output a representative binary string. This idea is not unique, a group lead by Daigo Shoji built a convolutional neural network (CNN) for a similar endeavor [43]. Shoji et al used a CNN to classify the morphological features of volcanic ash, so as to trace the samples back to their creation conditions. CNNs were chosen because of their use in processing data with grid-like topology, such as with images. CNNs are described by neural networks that use convolutions in place of general matrix multiplication in at least one of their layers [44]. This project represents a strong parallel to the task at hand, as it involves the use of CNNs to classify small-scale particles based on their morphological parameters. Much like with the nuclear forensics samples, it is believed that the shape of volcanic ash particles is heavily influenced by the creation conditions. Classification of these samples has traditionally been performed visually, and the inclusion of CNNs helps eliminate the user bias associated with visual analysis, which is another possible objective within the scope of this project. The provided dataset formed a catalog of 2972 SEM grayscale images with over 700 unique synthetic particles. Each image measured 1280x1024 pixels and contained a footer with information about the SEM setup, as well as the date and other parameters from the originating laboratory. These images were taken as the input data for the CNNs, but the truth values had to be established in order to carry out the supervised learning classification task. The aforementioned string encodings became the target truth values for the CNN to learn.

3.5 Neural Network Methodology

Data pre-processing steps included cropping every image before passing into the CNN through use of the Th-MakerX tool recommended by Shoji et al [45]. The crop was necessary to remove the image footer and project identifier string seen in Figure 3.

In certain images, the project string was located in the top left of the images, so a crop was made on both the top and bottom of each picture. The cv2 module in python was used to resize every image read in to be 300x250, with cubic interpolation. Because of the small size of the dataset and the inability to obtain more data, various image augmentation methods were used to present a greater variety of sample imagery to the CNN. A rotation range of 10 degrees was added, as the orientation of the sample in the SEM images is arbitrary to begin with, so reorienting them should not change classification. Random brightening and darkening effects in a range of 0.8-1.2 of the original brightness levels were applied. Random horizontal flipping was added as well. In addition, the image intensity values were rescaled to a range of 0 to 1.

Several variations on a CNN type model were tested for this project. The first type was a shallower network based on the model presented by Shoji et al [43]. that uses a single convolutional layer and max pooling layer, before passing to a hidden layer and then an output with the desired number of classes. Further testing presented better results with a network that used more convolutional, max pooling, and hidden dense layers. A specific step in the LANL morphology chart was chosen as an exploratory effort for the project. Step two was picked as it presented a wide variety of morphological traits without having a large amount of classes to choose from. Incidentally, the third class given in the second lexicon step did not exist in the dataset as the chemical characteristics of the uranium in the samples did not allow for crystalline growth. As such, the deep learning task turned into a binary classification problem, which aimed to predict if a given sample was rounded/blocky or a mixture of rounded and crystalline. If it is possible to achieve this, then results could feasibly be extended to the other three remaining labelled steps in the LANL flowchart in order to augment or improve a nuclear forensics analysis capability.

A pipeline had to be established in order to properly read and process the data.

The actual images were unlabelled with regards to the truth values and had to be associated with their respective steps and morphological characteristics. This was obtained by reading in the images and checking their file name for project and particle number (ex: Q019907C5102.png), and then searching a data matrix for the morphological traits associated with that particle. Several images could not be processed by the cv2 module, so they were thrown as exceptions and not read in. Once the images were stored as arrays they were checked for possible class imbalances. The first category in step two was the majority class, with 1005 images in the training dataset, while the mixture category contained 664. A class weight of 1.51 for class two was used to give the weights of the mixture images 1.51 times more importance than the weights of the rounded images, thus equalizing the categories during training. Several model parameters were tested by choosing from a parameter list and training with several possible combinations.

Figure 12 shows the overall model architecture, which contained parameters of 0.5, 0.5, 0.6 for the dropout layers, a l2 kernel regularization of 0.0005, and gradient clipping at 0.5 to prevent exploding gradients and improve behavior in the vicinity of steep cliffs in the loss-function space. Experiments were run to check for accuracy metrics, as it measured the ultimate goal of whether or not a CNN could classify particles as well as a human analyst. A preliminary test in this area was to run the network without regularization techniques and see if the training set accuracy could reach 100%. This was indeed the case; after nearly 300 epochs the network was able to overfit and learn the training set. This indicated that the network had sufficient capacity for the task, and was an ideal starting point for validation testing. Three-fold cross-validation was used to balance training times and statistically significant metric reporting. Finally, area under the curve (AUC) values were produced that outlined the performance of the classifier over each epoch. All code was written with python 3.7.3, keras 2.2.4 and tensorflow version 1.13.1.

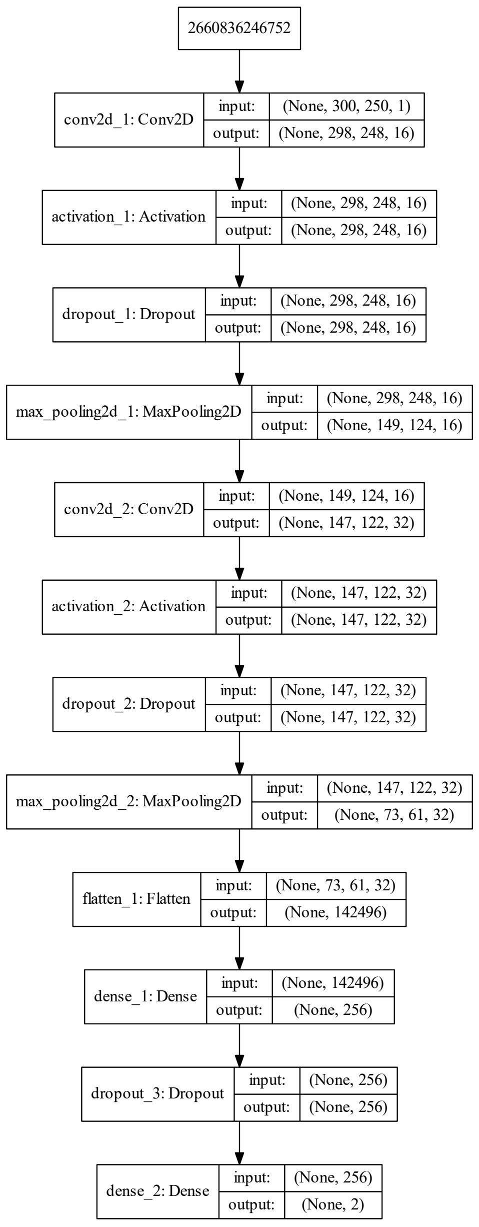

2660836246752
conv2d_1: Conv2D
input: (None, 300, 250, 1)
output: (None, 298, 248, 16)
activation_1: Activation
input: (None, 298, 248, 16)
output: (None, 298, 248, 16)
dropout_1: Dropout
input: (None, 298, 248, 16)
output: (None, 298, 248, 16)
max_pooling2d_1: MaxPooling2D
input: (None, 298, 248, 16)
output: (None, 149, 124, 16)
conv2d_2: Conv2D
input: (None, 149, 124, 16)
output: (None, 147, 122, 32)
activation_2: Activation
input: (None, 147, 122, 32)
output: (None, 147, 122, 32)
dropout_2: Dropout
input: (None, 147, 122, 32)
output: (None, 147, 122, 32)
max_pooling2d_2: MaxPooling2D
input: (None, 147, 122, 32)
output: (None, 73, 61, 32)
flatten_1: Flatten
input: (None, 73, 61, 32)
output: (None, 142496)
dense_1: Dense
input: (None, 142496)
output: (None, 256)
dropout_3: Dropout
input: (None, 256)
output: (None, 256)
dense_2: Dense
input: (None, 256)
output: (None, 2)

Figure 12. Model architecture that was used throughout the project. Final output layer used two nodes with a softmax activation. A Nadam optimizer was used for all tests.

4. Results

4.1 Predicting Q-value from binary morphology strings

Various classification models were fit against the binary morphology data in an attempt to predict which project a hypothetical particle resembled most. The predictive capability of these models was examined through use of accuracy reporting metrics in order to assess the confidence in said prediction. In particular, the random forest classifier was used to investigate classification accuracy, feature importance, and predictive capabilities. This was done by splitting the main dataset into a two groups of 90% non-Test and 10% Test or Holdout. A common split in machine learning processes is a 70/30 split, but the provided data contains too few points in the minority classes, as seen in Figure 13, thus to accurately train a classifier algorithm a 90/10 split is used instead. Total class distribution counts can be seen displayed numerically in Table 13 in Appendix B.

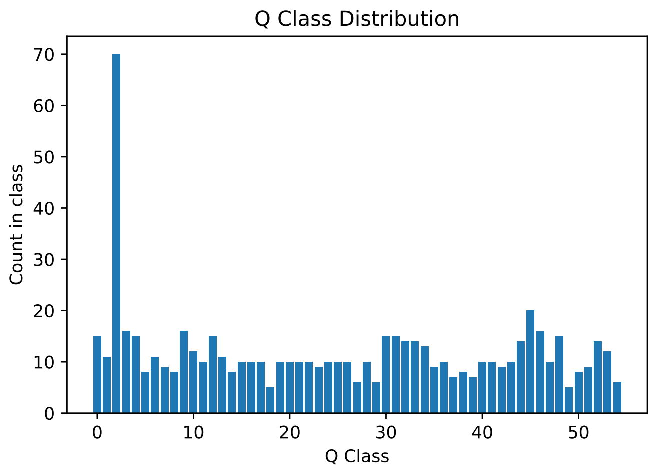


Figure 13. Distribution of particles for each Q passed into the machine learning algorithms.

The non-Test dataset is then upsampled to make the minority classes have equal amounts of data as the majority class. This is done through the use of synthetic data independently with SMOTE and ADASYN, or non-synthetic with random sampling with replacement. Once the new non-Test dataset contains equal amounts of points in all classes, it is split into training and validation datasets, this time with a more common 70/30 split. Several different classification models were then trained and tested against the dataset to check for relative performance metrics.

Table 5. SMOTE classification performance.

Validation Set Accuracy %	TestSet Accuracy %	Classifier
50.29	18.18	KNeighbors
3.51	3.84	SVC
54.97	19.67	Decision Tree
55.67	21.21	RandomForest
28.89	19.67	Bernoulli NB
32.63	15.15	LDA
53.34	18.18	GradientBoostingClassifier

Table 6. ADASYN classification performance.

Validation Set Accuracy %	TestSet Accuracy %	Classifier
45.83	16.67	KNeighbors
2.5	1.51	SVC
51.94	16.67	Decision Tree
54.07	18.18	RandomForest
39.07	19.69	Bernoulli NB
35.27	13.63	LDA
49.09	16.67	GradientBoostingClassifier

At first glance, the values from Tables 5 and 6 seem low compared to many of the high 90% accuracy metrics reported in the literature. However, it is important to keep in mind that overall accuracy is reported, which compares the amount of predicted labels that exactly match the true output labels. In a multiclass classification problem, a random guess has a $100 \cdot 1/\text{classes}$ % chance of making a correct

Table 7. Performance comparison of validation set accuracy between oversampled and non-oversampled datasets.

Unmodified Validation Acc %	SMOTE Validation Acc %	ADASYN Validation Acc %	Classifier
15.16	53.66	48.09	KNeighbors
16.85	36.44	31.19	SVC
21.91	59.43	56.54	Decision Tree
21.91	58.79	56.91	RandomForest
23.03	43.86	40.2	Bernoulli NB
24.71	40.48	32.96	LDA
21.34	53.02	52.18	GradientBoosting

guess. In this case, a random guess has a $1/55 \approx 1.8\%$ chance of correctly predicting the output class. Thus, the classifiers which report accuracy metrics above 10% are roughly 5 times more effective than a random guess, which indicates a consistent prediction capability. Validation set results are higher than those of the test set because the validation set had been included in the original oversampling, which meant it more closely resembled the dataset used for training. Both synthetic oversampling techniques performed better than the unmodified dataset as seen in Table 7. In each scenario, the data was split into three groups: training, validation and test. The ADASYN and SMOTE results were trained and tested on oversampled data, thereby boosting the amount of particles that could be used for reporting of accuracy metrics.

In an attempt to understand the machine learning algorithm's decision process, the feature_importances_ module of scikit-learn's decision tree classifiers was used to determine which steps of the LANL lexicon were the most important in classifying particles according to the Q values. When used on the GradientBoostingClassifier, the top five input features that returned were: S4FA (Blocky), S3F4 (Sub-angular), S3F6 (Very Angular), S1F3 (Agglomerate), S3F3 (Sub-rounded), demonstrated in Figure 14. Once the model was trained, each classification technique could use scikit-

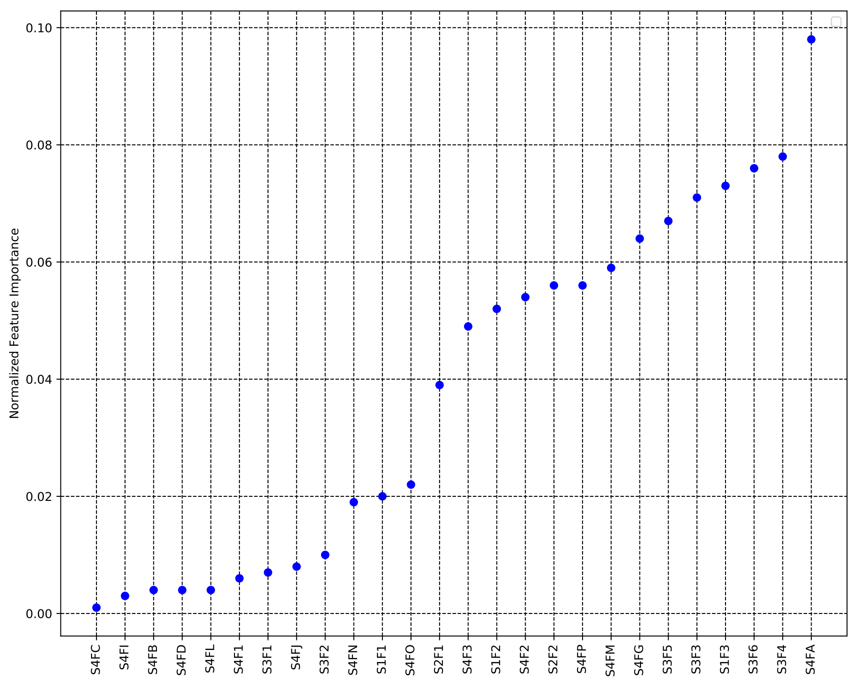
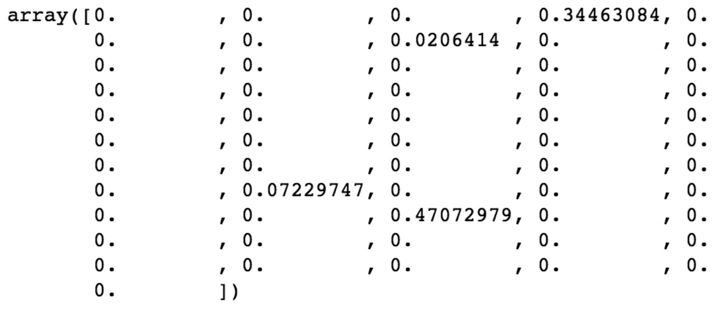


Figure 14. An ordered representation of the normalized feature importance on Q-value classification from the GradientBoostingClassifier.

learn's predict capability on the test datatset, which returns the probability that a test sample belongs to each Q-class as seen in Figure 15. This method allows a user to check and see which creation conditions a particle most closely resembles. In this manner, creation conditions can be compared for similarity. The predict functionality would allow a user to compare a new test particle against the synthetic reference particle dataset. Another extremely useful aspect of this functionality is the reduction in creation conditions that a test particle has to be compared against. Figure 15 shows that a test particle has the highest chance of belonging to 4 out of the total 56 Qs, something that could help analysts quickly narrow down comparisons when working on very large datasets. As an example, Figure 15 shows that a sample particle has the highest probability of belonging to Q019908, Q019914, and Q016317, with a low chance of belonging to Q016322. These Qs with high probabilities are linked by similar creation conditions in the additive and final product categories, and could be a link that is explored by an analyst in the forensic process.



```
array([0.        , 0.        , 0.        , 0.34463084, 0.
       0.        , 0.        , 0.0206414 , 0.        , 0.
       0.        , 0.        , 0.        , 0.        , 0.
       0.        , 0.        , 0.        , 0.        , 0.
       0.        , 0.        , 0.        , 0.        , 0.
       0.        , 0.        , 0.        , 0.        , 0.
       0.        , 0.        , 0.        , 0.        , 0.
       0.        , 0.07229747, 0.        , 0.        , 0.
       0.        , 0.        , 0.47072979, 0.        , 0.
       0.        , 0.        , 0.        , 0.        , 0.
       0.        , 0.        , 0.        , 0.        , 0.
       0.        ])
```

Figure 15. An example prediction array. After training, machine learning classifiers output the probability that a particle belongs to each Q. The first element in the array is the chance that the test particle belongs to the first Q-value, and so on.

4.2 Creation Condition Analysis

Comparisons between particle morphology and specific creation conditions were also pursued through exploration of possible trends in morphological characteristics between creation conditions. Plots that displayed the normalized amount of particles in each lexicon step for given creation conditions allowed for a first-pass look at possible categories of interest. For example, in Figure 16 the percentage of particles in each part of step 3 of the lexicon are shown and grouped according to their precipitation temperature. Error bars are represent trait occurrence uncertainty and are calculated via

$$\Delta x = x\sqrt{\left(\frac{\Delta y}{y}\right)^2 + \left(\frac{\Delta z}{z}\right)^2}, \tag{15}$$

where y and z are obtained from the ratio of morphological traits observed to total particle counts, and x is the resulting ratio of y and z [46]. As such, the formula is more simply stated as:

$$\Delta x = \frac{\text{traits observed}}{\text{total particles}}\sqrt{\left(\frac{\Delta\text{traits observed}}{\text{traits observed}}\right)^2 + \left(\frac{\Delta\text{total particles}}{\text{total particles}}\right)^2}. \tag{16}$$

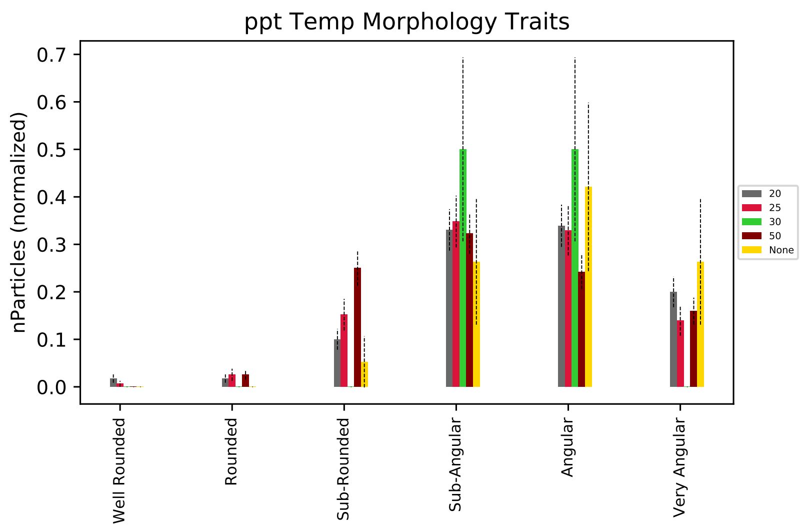


Figure 16. A step-wise comparison of particle morphology occurrence between creation condition parameters. In this case precipitation temperature [°C] is measured against particle angularity.

A total of seven creation conditions and 4 lexicon steps were plotted, which created 35 plots of particle morphology creation conditions, all of which can be found in Appendix B. Of the listed creation conditions, the final product was found to be the most useful discriminant by visual inspection, as particles had visibly different morphologies from final product to final product. The fifteen final products were broken into two different categories: pure final products, and mixed phase final products. These categories allowed for an investigation into the correlation between creation process chemistry and resultant particle morphological features. Instead of training machine learning classifiers to predict Qs which represented collections of creation condition parameters, machine learning classifiers were trained to predict final products given input binary morphology strings. As with particle counts in the Q classes, the problem-space featured strong majority classes, and many minority classes, as seen in Figure 17.

The balance of particles in the pure final product classes compared to the mixed final products was roughly 50% in each. First the pure final product classes were trained and tested on to see if classifiers could learn to distinguish between morpho-

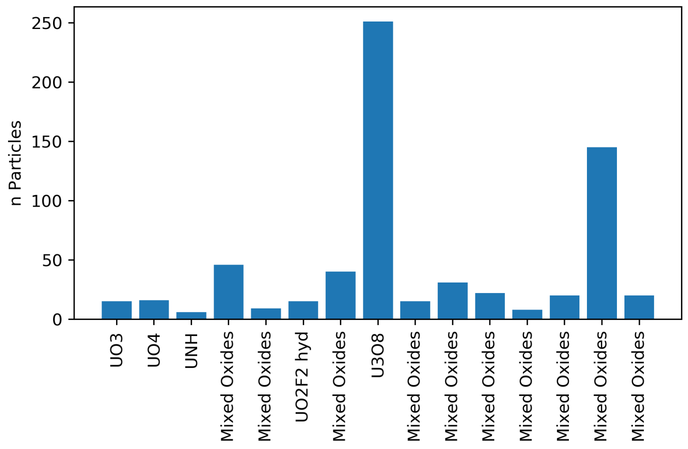


Figure 17. Particle class distribution for final product groups.

Table 8. SMOTE classification performance on pure final products.

Validation Set Accuracy %	TestSet Accuracy %	Classifier
74.164	44.231	KNeighbors
78.067	51.923	Decision Tree
79.926	55.769	RandomForest
78.439	51.923	GradientBoosting
58.736	19.231	GaussianNB
63.197	25.0	Bernoulli NB
64.870	26.923	LDA

logical features that resulted from pure final products. The results of this process can be seen in Table 8.

Accuracy values were markedly higher than prediction efforts on Qs, in part because of the dramatic reduction in categories to be guessed, but also as a result of tying particle morphology to individual creation conditions that may have influence on their final topology. The feature_importances_ module of scikit-learn's decision tree classifiers was once again used to determine which steps of the lexicon were the most important in classifying particles; this time according to their pure final products. When used on the GradientBoostingClassifier, the top five input features that

Table 9. A survey of the complete final products in the synthetic particle creation process.

Encoded Final Product	Unencoded Final Product
F1	UO3
F2	U3O8
F3	UO3, UO4, U3O8
F4	U3O8, UO3, UO2(OH)2 hydrate, UO2F2
F5	U3O8 with very minor UO2F2
F6	U3O8 with minor UO2F2
F7	U3O8 with UO2F2
F8	ADU
F9	AUC
F10	UO4
F11	N/A (UNH)
F12	Schoepite
F13	U Oxide
F14	UO2F2 hydrate
F15	U3O8, UO3

returned were: S3F6 (Very Angular), S4FG (Irregular), S1F1 (Individual Particle), S1F3 (Agglomerate) and S4FN (cracked) as shown in Figure 18. The least important features involved many of the step 4 characterizations such as S4FC (Lenticular), S4FD (Prolate), S4FI (Curved), S4FJ (Bent), and S4FL (Flattened).

Figure 19 graphically demonstrates the model's classification performance by displaying true positives, false positives, true negatives and false negatives for each pure final product. In a scikit-learn confusion matrix, entries in row i, column j, are the number of observations actually in group i but predicted to be in group j. For example, Figure 19 shows 15 correct predictions of F2 and 5 erroneous predictions of F2 particles as F8. F2 and F8, U_3O_8 and ADU respectively, are the classes with the strongest representation in the test set and are also the classes with the most overlap in prediction by the classifier. This points towards an avenue that may be explored during failure analysis efforts. Once trained, the machine learning models were used to associate mixed oxide final products with pure final products in an attempt to

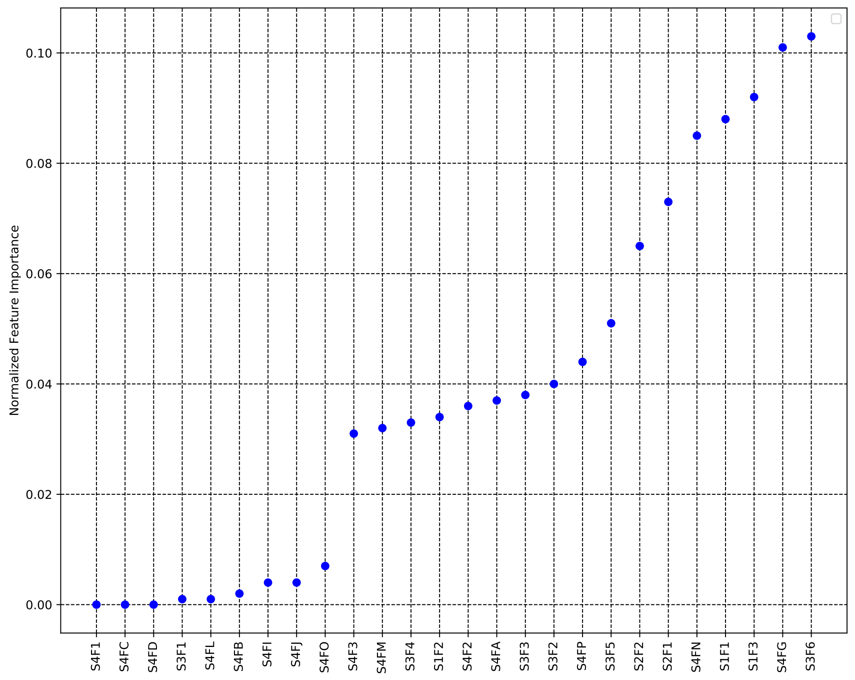


Figure 18. An ordered representation of the normalized feature importance from the GradientBoostingClassifier for classification on pure final products.

replicate a more realistic forensics process. The predict functionality of scikit-learn was used to take binary strings of particles with mixed oxide final products and predict which pure final product they most resembled. Final products F3 through F7, as well as F15 were of interest here, and a classifier that could successfully relate the two categories would imply a consistency of morphological traits across final products.

Mixed Oxide Prediction Using A Pure Final Product Classifier

After being trained on pure final products, machine learning classifiers were given particles from Qs with mixed phase final products. Scikit-learn's predict functionality was used as a test to see if mixed oxide final products could be related to their pure counterparts. Accuracy metrics were then reported, by counting the amount of correct comparisons (i.e. if the predict function outputs a final product of U_3O_8 or UO_4 for a mixed oxide particle with a final product of U_3O_8, UO_4) versus the number of

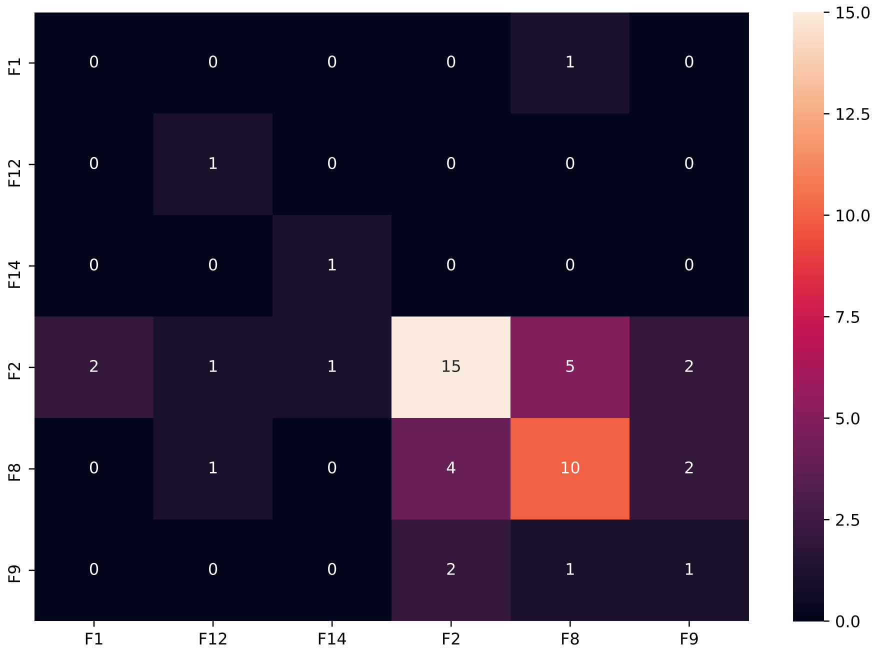


Figure 19. Confusion matrix for the pure final product test set resulting from a trained GradientBoostingClassifier.

incorrect comparisons. Figure 20 shows the results for each class of mixed oxide, with varying accuracy levels across the board. These results show that the morphology of pure final products is consistent, and persists even when added to other materials to form a mixed oxide. As such, the pure final products may serve as a discriminator across a wide range of mixed final products.

4.3 Hyperparameter Tuning

Machine learning models of all types are parametrized by a set of hyperparameters that may be controlled by the user to maximize the utility of the model [47]. These controls are used to configure different aspects of the learning algorithms and can have a wide range of effects on the resulting model's performance. The process

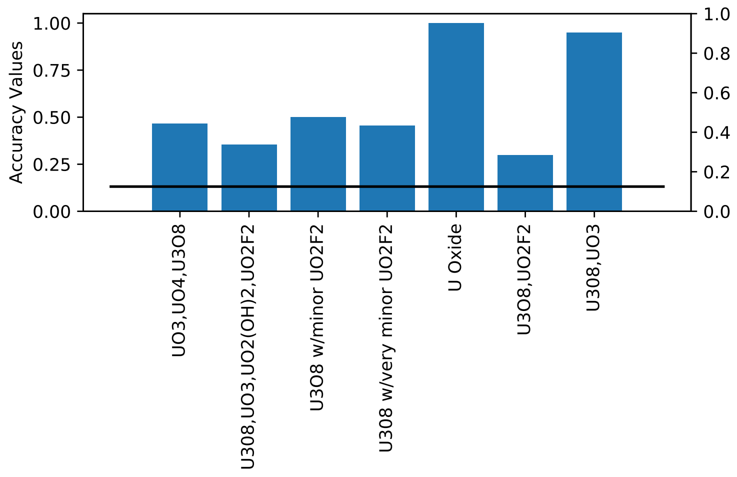


Figure 20. Accuracy results from using the scikit-learn predict function to correlate the mixed oxide final products to their pure counterparts. A random guess threshold line is included for ease of performance comparisons.

of finding the best hyperparameters for each model given a dataset is called hyperparameter search and is either performed manually [48,49] or through searching a grid of input parameters [23]. These methods of finding ideal hyperameters take real-world considerations into account, as a perfect search would be exhaustive of most if not all possible parameters, something that is not feasible under current computational performance levels. As such, models are tested on the validation set with a variety of hyperparameters, and the model settings that result in the highest reporting metric are typically selected for the final model.

An effort was made to increase the accuracy of prediction on the final product creation condition. The GradientBoostingClassifier method was used because of its high accuracy marks throughout creation conditions, and its plethora of hyperparameters that allowed for fine-tuning. Scikit-learn's GridSearchCV routine was used, which exhaustively searches over a range of user defined parameter values for a given model. Four different hyperparameters were chosen to tune due to their relative impact on final model performance: n_estimators, max_depth, min_samples_split, and subsample. The first parameter, n_estimators, controls the number of boosting stages to be used.

Gradient boosting is robust against over-fitting, so a large number of boosting stages can usually lead to better performance [23]. Max_depth sets the depth limit, and thus the number of nodes for the individual trees. Min_samples_split is the minimum amount of samples required to split an internal tree node. Machine learning rules of thumb commonly advise for using a value that is 0.5-1% of the amount of data trained upon. The final parameter, subsample, denotes the fraction of samples used for fitting the individual base trees. If smaller than unity, this results in stochastic gradient boosting. A value smaller than one also leads to a reduction in variance, and an increase in bias [23]. After running GridSearchCV with the values listed in Table 10,

Table 10. An overview of the values used to hyperparameter tune the GradientBoostingClassifier on the final products dataset.

Hyperparameter	Values
n_estimators	20, 40, 60, 80, 100, 200
min_samples_split	3, 4, 5, 6
max_depth	3, 4, 5, 6, 7, 8, 9, 10
subsample	0.7, 0.75, 0.8, 0.85, 0.9, 1

an output of the tuned model gave parameters of max_depth of 6, min_samples_split of 4, n_estimators of 200, and subsample equal to 0.85. This resulted in an accuracy gain of roughly 2% in the validation and test sets, when averaged over five instances of the GradientBoostingClassifier fitted with the optimized hyperparameters.

A KNeighborsClassifier was also used for hyperparamter tuning, to see if larger increases in accuracy were possible with a different machine learning model. The parameters that could be tuned were different in this case, with the three total parameters that consisted of: n_neighbors, p, and weights. Controlling n_neighbors changes the number of neighboring points that the KNeighborsClassifier considers when calculating class boundaries. By default this is held at five neighboring points. Changing p had the effect of choosing the power parameter for the Minkowski metric. When $p = 1$, the algorithm calculates the also known as the l_1 norm, also known as the manhattan distance, where the space between two points is measured along axes at right angles. With $p = 2$ the euclidean distance is used, known as the l_2 norm. For arbitrary p values, the Minkowski distance l_p norm is used [23]. Weights controlled the weight function used in prediction, with two main choices of Uniform or Distance. Uniform weighing made all points in each neighborhood equally weighted. Distance changed the weight of points in accordance with the inverse of their distance. In this setting, closer neighbors to a selected data point will have a greater influence than those further away [23]. Table 11 shows the total parameter space under consideration.

Table 11. An overview of the values used to hyperparameter tune the KneighborsClassifier on the final products dataset.

Hyperparameter	Values
n_neighbors	3, 4, 5, 6
p	1, 2, 3
Weights	Uniform, Distance

After using the same GridSearchCV and input dataset as the previous hyperparameter effort, the best parameters resulted in n_neighbors = 4, $p = 1$, and weights = distance. This resulted in a 1.68% increase in accuracy when averaged over 5 different random splits of the dataset.

4.4 Classifier Performance Across All Creation Conditions

Scikit-learn classifiers were trained on the dataset in order to predict each category of final product when given a binary morphology string. In this manner, a comparison could be made between creation conditions to see which ones were the easiest to distinguish between, given their class balances or resultant impact on particle morphology. Each model was trained on a dataset that had been expanded through SMOTE oversampling, and later tested on held out non-oversampled data.

SMOTE was chosen over ADASYN because it provided higher accuracy values in the validation set with greater frequency than ADASYN, as seen in Table 7. Reported balanced accuracy values in Figure 21 are from an average of 10 different runs, where SMOTE and train_test_split routines were given randomly chosen seeds from a list of values. An additional group of creation conditions was tested: pure final products, which consisted of particles from the F1, F2, F8, F9, F10, F11, F12, and F14 final products. Hyper-parameter tuning was not included in the following results.

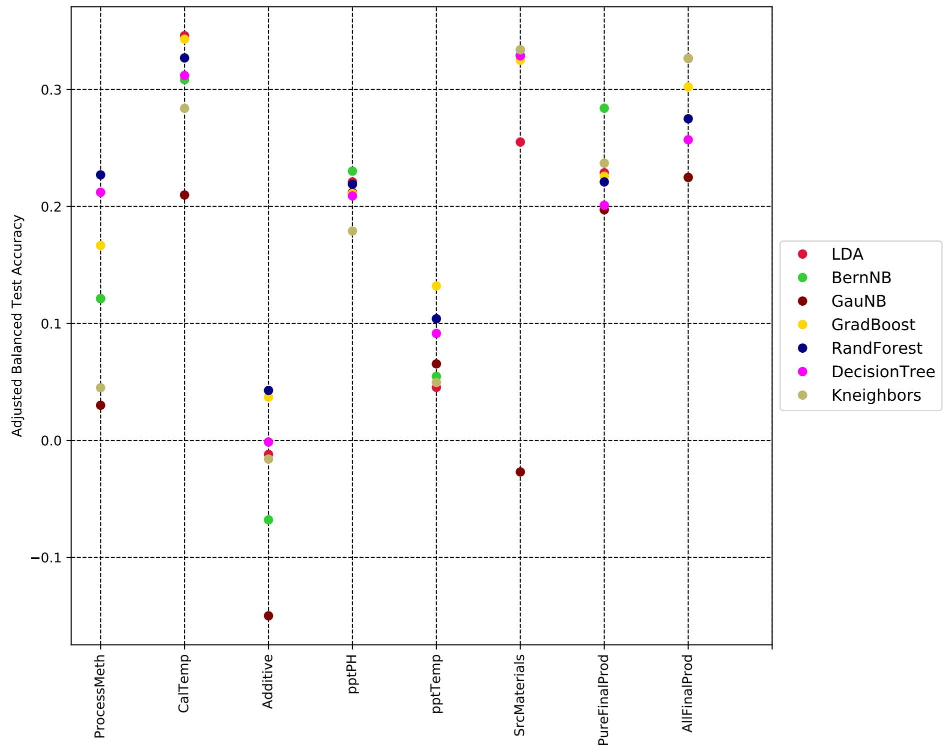


Figure 21. Adjusted, balanced accuracy results on the test set data for each classifier used.

Because the class count is taken into consideration with the adjusted balanced accuracy, the results of Figure 21 allow for a look into the relative impact of each creation condition on a particle's morphology. Calcination temperature, followed by source materials and final products, all have the highest balanced accuracy score. This shows that of the available parameters, these creation conditions are most closely tied to particle morphology and serve as the best discriminators in the creation process. Narrowing down which creation conditions matter can help clue analysts into specific processes when observing test datasets. In addition, this serves to inform future studies on which creation conditions to look into at greater depth. Of the remaining parameters, additive classification results fall below zero and thus consist of models that perform worse than random guessing, implying minimal association with particle morphology.

Some feature engineering was conducted in this step to check for possible accuracy gains. The VarianceThreshold function was used to perform dimensionality reduction on the original dataset with 26 columns. Two variance levels were tested: 0.16 resulting from an input of p=0.8 and a variance of 0.09 which corresponded to p=0.9. These probability values created datasets that had 9 columns and 15 columns, respectively. Lower probability values were not extensively tested, as the dataset lost too many columns and lower accuracy values became increasingly common. The variance reduced dataset was compared to a non-hyperparameter tuned GradientBoosting-Classifier on classification of particles into final products, the results of which can be seen in Table 12. Results indicate that removal of columns consisting of either 90%

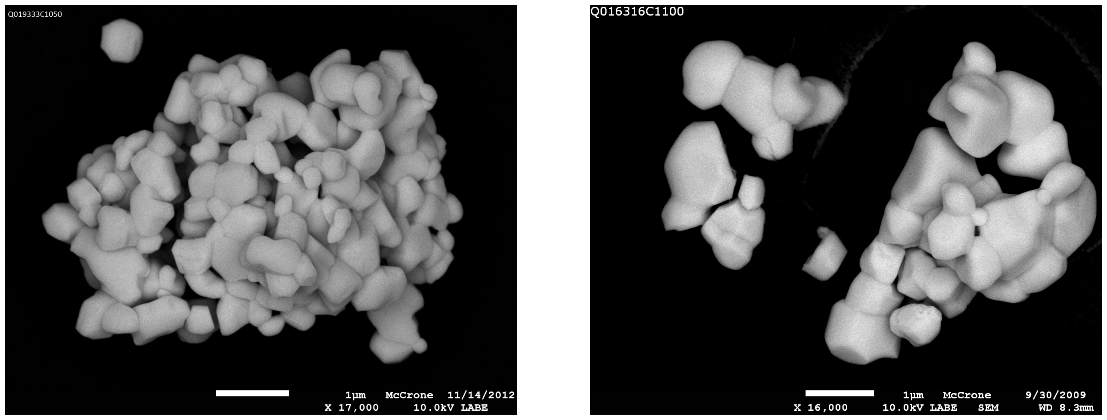
Table 12. Final product classification results from the standard dataset compared to variance reduced datasets. Accuracy values are averaged over 5 random splits.

# of Columns	Probability value	Balanced Test Acc
26	Unmodified Dataset	20.89
9	p = 0.8	20.01
15	p = 0.9	25.23

ones or 90% zeros yields an ideal amount of dimensionality reduction in the dataset by removing the columns that do not communicate useful information to the machine learning model. For probability levels lower than this, useful columns begin to be removed, and thus accuracy gains can no longer be expected.

4.5 Failure Analysis

After training a GradientBoostingClassifier to distinguish between pure final products, a precursory exploration into the misclassifications was conducted to understand why particles were being incorrectly labeled. This comparison was carried out both visually on the SEM images themselves, and feature-wise on the input binary data. One of the first misclassifications arose from particle Q16316C1100 with final product U_3O_8 being labelled as ADU. Figure 22 compares the misclassified particle with a particle consisting of ADU as a final product. An examination of the two shows that their topological features are quite similar and could certainly appear identical to a human analyst. However, because the machine learning classifier only relates input particles to creation conditions as a whole, other instances of ADU were inspected to find additional related particles.

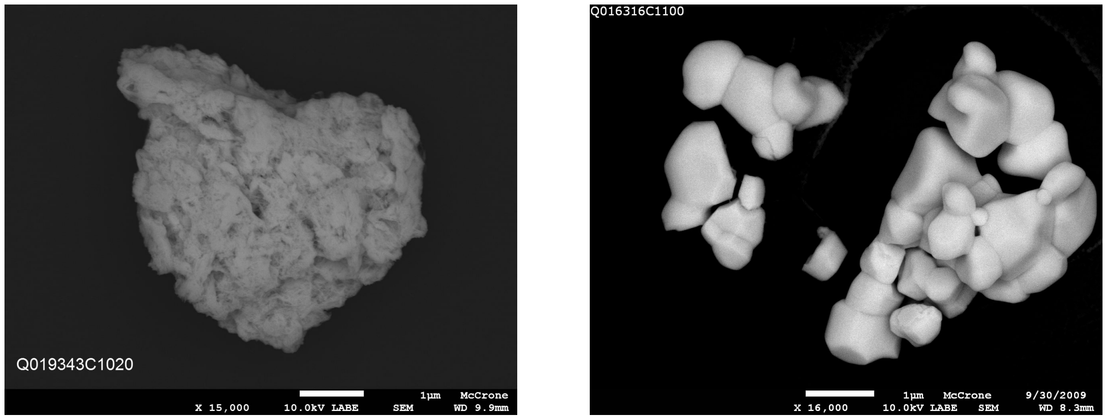


(a) ADU particle example

(b) U_3O_8 Particle that was incorrectly classified as ADU

Figure 22. A GradientBoostingClassifier misclassification. Here similar particles are displayed to explain possible confusion between particles.

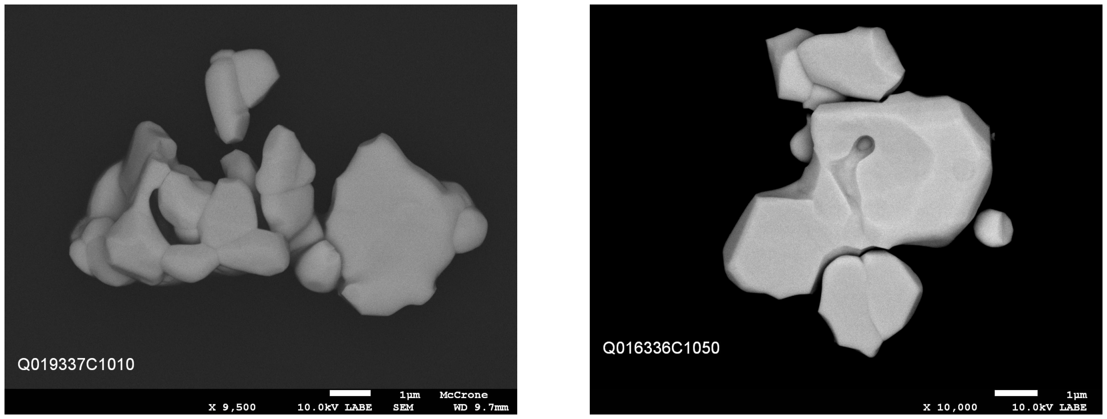
Figure 23 again displays the misclassified U_3O_8 particle on the right, but this time compares against an example ADU which is vastly different. Particles with the jagged surface features seen in Figure 23 (a) are not successfully differentiated from the sample in Figure 23 (b) due to the lack of topological micro-features in the first four steps of the lexicon. The macro-features: overall shape, angularity, and sphericity appear similar between images and may contribute to the misclassifications that a machine learning algorithm could make when analyzing particles with a truncated lexicon.


Q019343C1020
x 15,000 10.0kV LABE SEM 1µm McCrone WD 9.9mm
Q016316C1100
x 16,000 10.0kV LABE SEM 1µm McCrone 9/30/2009 WD 8.3mm

(a) ADU particle example

(b) U_3O_8 Particle that was incorrectly classified as ADU

Figure 23. A GradientBoostingClassifier misclassification. Here different particles are displayed to present possible limitations in the truncated lexicon.



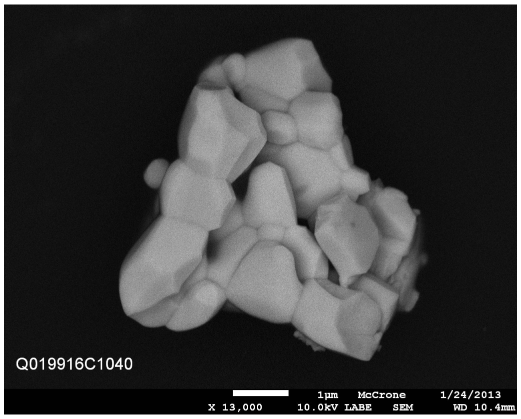
(a) AUC particle example

(b) U_3O_8 particle that was incorrectly classified as AUC

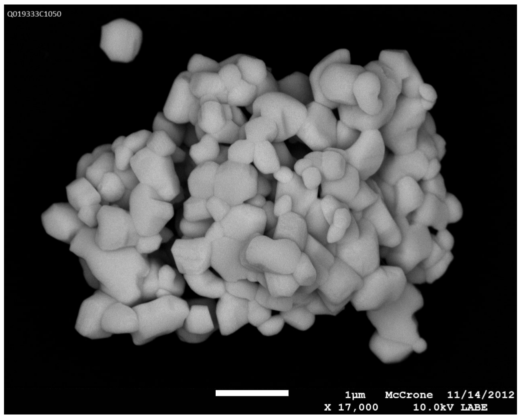
Figure 24. A GradientBoostingClassifier misclassification. Here similar particles are displayed to explain possible confusion between particles and their creation conditions.

Another incorrect prediction involves particle Q163361050 of final product U_3O_8 which was incorrectly classified as AUC. This misclassification was more of a direct comparison than the previous event, as there were only 20 particles of final product

AUC in the entire dataset, and all displayed similar morphological features. Figure 24 shows two particles that are visually similar but differ in their final products. The near-identical features of the two particles may have required more steps of the LANL lexicon to successfully distinguish. A final comparison of the classification



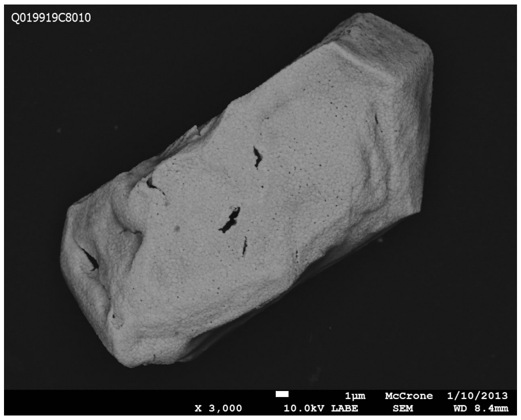
(a) U_3O_8 particle example



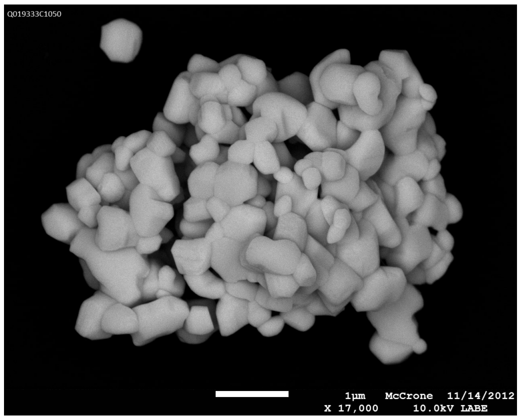
(b) ADU particle that was incorrectly classified as U_3O_8

Figure 25. A GradientBoostingClassifier misclassification. Here similar particles are displayed to explain possible confusion between particles and their creation conditions.

errors shows an ADU particle Q19333C1020 being incorrectly labeled as U_3O_8. Visual inspection reveals similarities to particles in the Q19916 series as shown in Figure 25. As with other cases before, comparison to other Qs with U_3O_8 as a final product also showed some dramatic differences, evidenced in Figure 26. Additional samples and steps are likely required to raise the consistency of correct classifications between creation conditions.



(a) U_3O_8 particle example



(b) ADU particle that was incorrectly classified as U_3O_8

Figure 26. A GradientBoostingClassifier misclassification. Here different particles are displayed to explore limitations of the truncated lexicon.

4.6 Neural Network Results

When put to the task of assigning correct lexicon step classifications, the network's performance increased over the course of the 400 epoch training time, as seen in Figure 27. Model loss results in Figure 28 show neural network overfitting on the data around the 250th training iteration, thus future efforts on a similar dataset would benefit from training for similar timescales. Accuracy results in the three-fold cross validation test were reported as 0.7510, 0.7615, and 0.7824, averaging 76.5% classification ability after training. A final test set classification of 74% was obtained on the held-out images, which means that 74% of the test images were correctly classified as either Rounded/Blocky or Mixture from step two of the lexicon. A custom callback was created that produced training and validation area-under-the-curve (AUC) values after each epoch of training. Final results output a value between 0 and 1, where a model with 100% incorrect predictions has an AUC of 0, and a model with 100% correct predictions has an AUC of 1. This was used for two reasons: AUC values are scale invariant, and measure how well predictions are made regardless of class balance. Second, the AUC is classification threshold invariant, as it measures the quality of the model's predictions regardless of the probability threshold used. After training, the test set AUC halted close to 0.77, which is above the random guessing baseline of 0.5 and shows that a discriminative ability between rounded/blocky and mixture exists.

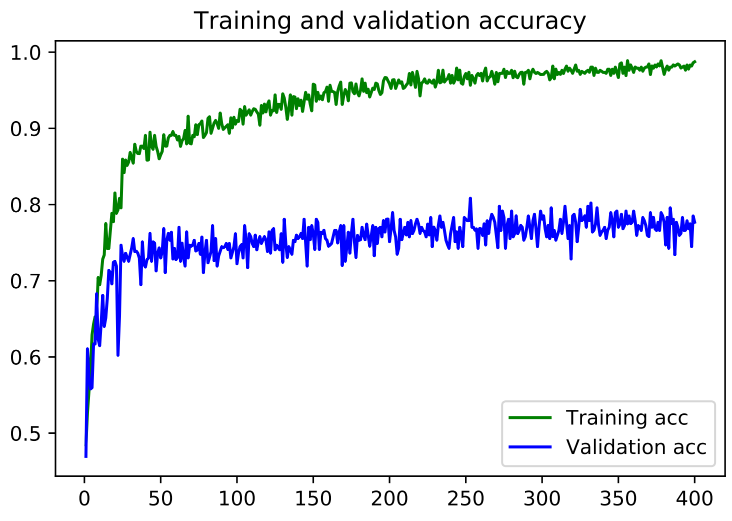


Figure 27. Model accuracy over 400 training epochs.

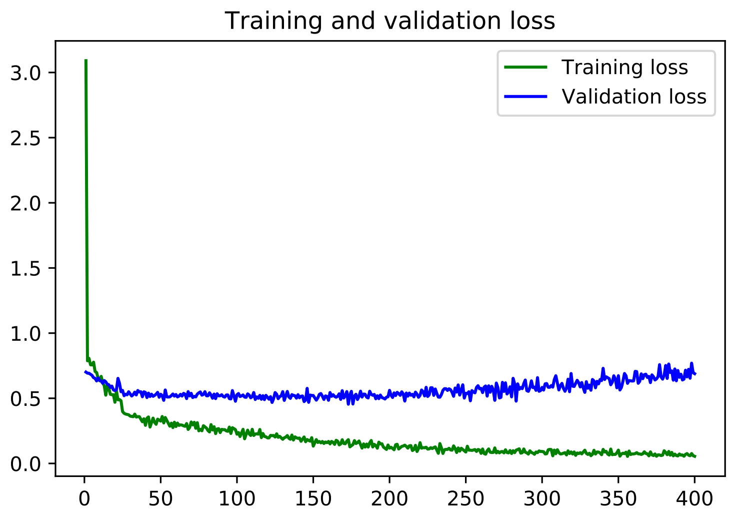


Figure 28. Model loss after training. Binary crossentropy was used for the step two prediction, as particles in the dataset were classified as either rounded/blocky or mixture, and never crystalline.

5. Conclusions

5.1 Summary

This research project encompassed an analysis of synthetic uranium particulate with well documented creation conditions. Previous research had indicated the particulate creation conditions were tied to resultant morphological features [50–53]. However, these approaches had not used a standardized lexicon to make objective comparisons between particles. This research represented a first effort at using a lexicon of particle morphology in tandem with machine learning techniques to derive relationships between creation conditions and final morphology. Results show that trends in morphology can be analyzed using machine learning techniques, and that classification efforts can be applied in a nuclear forensics process to glean additional information that may not have been available before.

Likewise, it was shown that creation conditions do in fact impact the final particle morphology, a result backed up by the forensic literature. The relative distinguishability is based on the individual creation conditions. In particular, source materials, final chemical compound, and calcination temperature all appeared as having the strongest association with particle morphology. The last result ties in well with the work done by Schwerdt et al [50] which shows variations in particle size and sphericity as a function of calcination temperature. The association with final chemical compound implies that trends in morphology are strongly tied with chemical composition of the sample. Adding chemical composition or elemental abundance data in with the one-hot encoded morphology strings could be more successful than using morphological data alone, and may result in higher accuracy values in future studies. Finally, results showed that trained machine learning algorithms demonstrated a potential for informing analytic procedures of material classification. The present work provides the foundation required to justify development of analyst tools, which could serve to reduce analysis times, make quick comparisons between datasets, and provide new insights that may not have been available before.

5.2 Initial Goals

At the start of the project, several questions were outlined and formed the guiding efforts of the research:

1. **Is it possible to distinguish among the project numbers in the synthetic dataset?**
2. **What are the most useful morphological features of a sample particle?**
3. **Can this dataset form an adequate reference for real life forensics data?**
4. **Can any correlation be determined to associate a particle's morphological features with the chemistry conditions under which it was formed?**

Answering the first question indicated that a trained machine learning classifier could, in fact, distinguish between project numbers when given a test particle. Additional particles would be necessary to increase classifier accuracy to levels that would be consistently useful to nuclear forensic analysts, but the results still prove an interesting utility that should be further explored. A full application of this technique could serve to increase the speed at which analysts make connections between test samples and their unknown creation conditions. The second question was answered through use of the FeatureImportances routine of a trained GradientBoostingClassifier. It was found that the most important features commonly included

S1F3 (Agglomerate) and S3F6 (Very Angular), with other features becoming more or less important depending on the type of creation condition that was being classified. Question three is answered as a 'possibly'. If the dataset is to be used as a reference for real life datasets, then additional statistics are needed to ensure that proper comparisons can be made between test particles and the synthetic ones used for reference. The fourth and final question was answered by training a classifier to predict a particle's specific creation condition. Results were reported with the use of balanced, adjusted accuracy metrics, which took class imbalances into account and reported the macro-average of accuracy on each creation condition. As reported above, it was shown that three of the creation conditions were most closely associated with final morphology. Further research is needed to quantify the exact extent that these creation conditions influence particle morphology, but the current results provide excellent building blocks for further efforts.

5.3 Lexicon Steps

Being able to classify the particles according to their shape indicates that the truncated LANL lexicon with just the first four steps was able to replicate morphology classes. This shows that the lexicon is relevant and useful for efforts involving comparison of particulate against one another, and hopefully provides a starting ground for widespread adoption of the method. Of the steps used, 1 - 4 focus on large scale features: step one covers particle nature, step two captures overall morphology, step three measures edge characteristics and step four covers a range of sphericity values. The addition of steps 5 - 11 would allow for a selection of most descriptive steps from the entire lexicon. Step 8 is suspected to be one of the more useful steps, as it covers particle surface features; many of these descriptors are unique and cover a wide variety of indentations and patterns. Efforts in the realm of error analysis showed that addition of surface features would serve to increase particle distinguishability, reduce misclassifications, and improve accuracy scores of the models.

5.4 Future Work

A first priority in any future work on the SEM dataset would involve fully converting every particle into a binary string in accordance with the LANL lexicon. Once all steps had been completed, machine learning techniques could be applied to discover which steps were the most useful for associating particles with creation conditions, an important result in it of itself. Narrowing down which aspects of particle morphology mattered most would help clue analysts into possible ties between specific chemical creation processes and resultant particle shape.

A future effort with binary encoded data being fed into machine learning classifiers would greatly benefit from the presence of balanced classes. This is not a cheap endeavor, so if not realistic, then a dataset with classes that display a lesser degree of imbalance than the current dataset would still be of service. The main issue throughout the project was the multitude of minority classes with extremely low representation. Regardless of machine learning algorithm used, it is difficult to learn on a multiclass datatset where the minority classes have a single digit amount of data points.

Another tool that would be useful to forensics analysts would be the ability to conduct classifications on groups of particles. In essence, an analyst would give the algorithm several particles at once, and the already-trained classifier would report back the most likely Q (or Qs) that this batch came from. This technique would allow analysts to view samples as an overarching whole, and see what characteristics they collectively displayed.

In regards to the neural network approach, a future effort would see improvement from additional SEM imagery to train on. A dataset with images free of text or other labels would eliminate the need to crop labels out of the picture. This would allow for a full view of the particles as captured by the SEM, which would help add morphology for the neural networks to learn. Because the work done in this area was exploratory, additional steps in the lexicon could be run through the neural network to see if they were more or less accurate than the step two example conducted here.

Appendix A. Morphology Flowchart

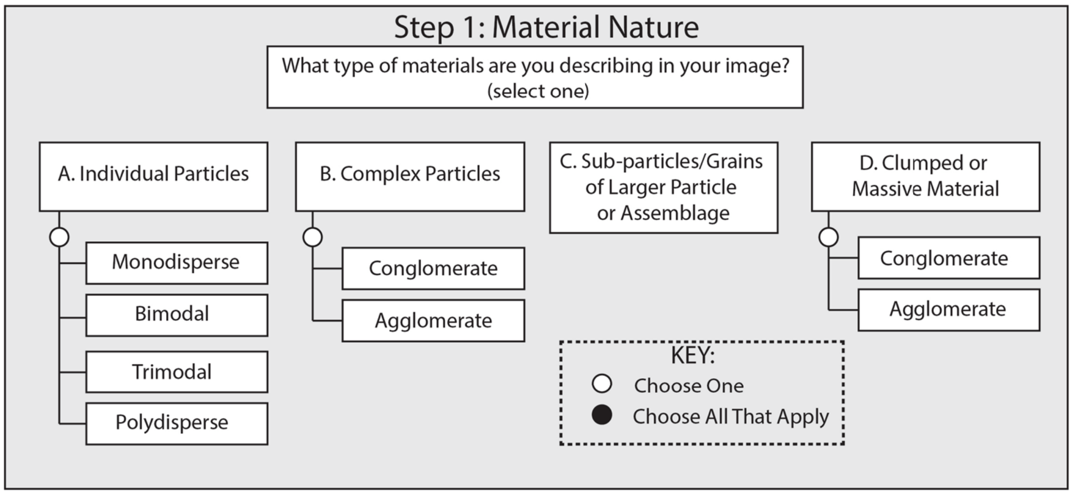


Figure 29. Flowchart for the particle nature classification. This first step allows for large particles with sub–components to be classified separately from their sub–particles [4].

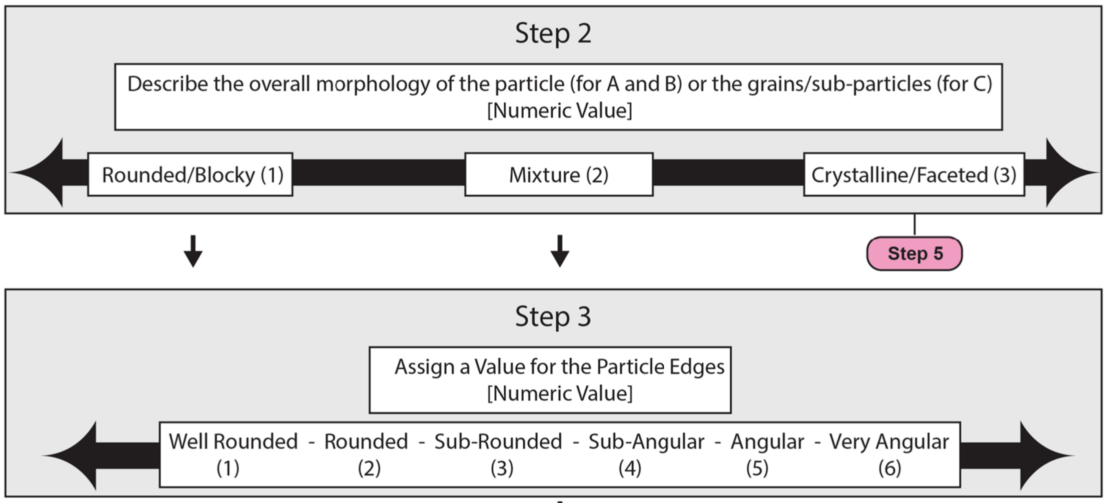

Step 2
Describe the overall morphology of the particle (for A and B) or the grains/sub-particles (for C) [Numeric Value]
Rounded/Blocky (1)
Mixture (2)
Crystalline/Faceted (3)
Step 5
Step 3
Assign a Value for the Particle Edges [Numeric Value]
Well Rounded (1) - Rounded (2) - Sub-Rounded (3) - Sub-Angular (4) - Angular (5) - Very Angular (6)

Figure 30. Continuation of the flowchart classification steps [4].

Reprinted by permission from Springer Nature: Springer Netherlands, Journal of Radioanalytical and Nuclear Chemistry, A lexicon for consistent description of material images for nuclear forensics, Alison L. Tamasi et al, 2016

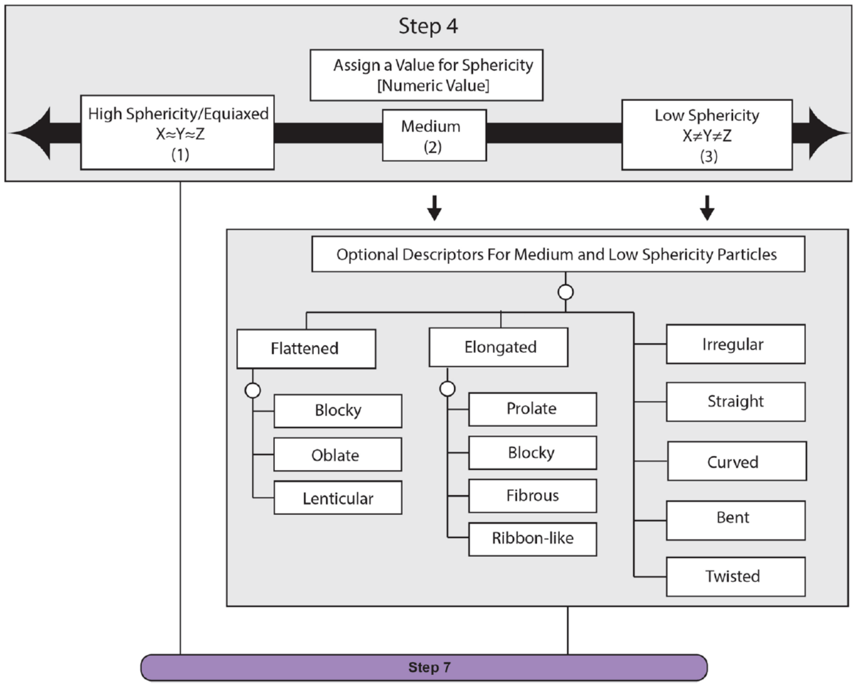
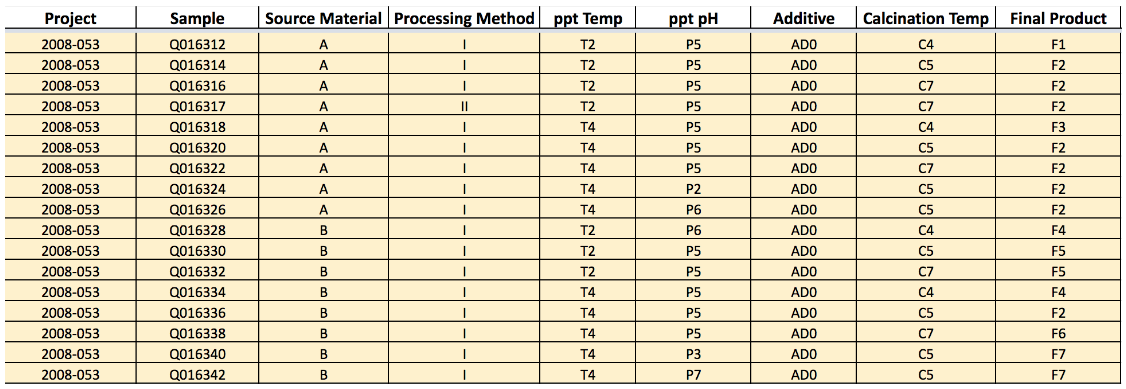


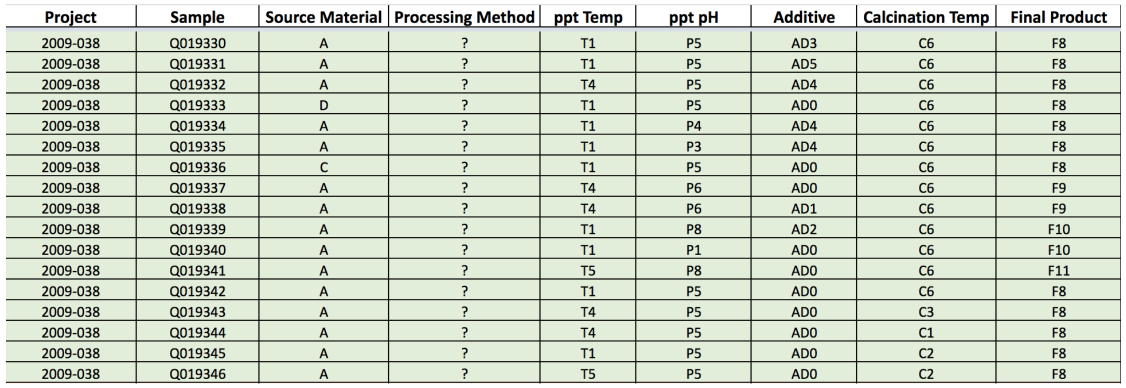
Figure 31. The last step that has currently been applied to the particle dataset [4].

Appendix B. Creation Conditions



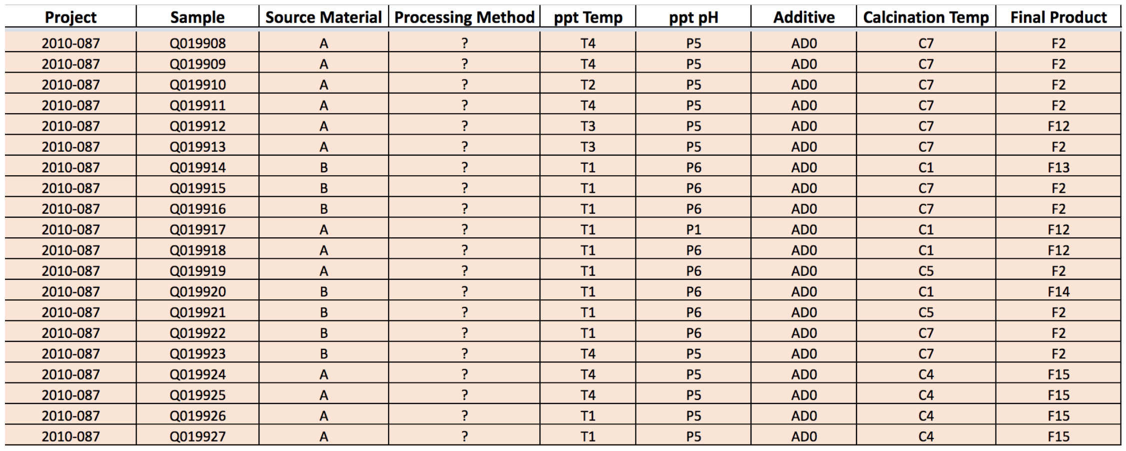
Project	Sample	Source Material	Processing Method	ppt Temp	ppt pH	Additive	Calcination Temp	Final Product
2008-053	Q016312	A	I	T2	P5	AD0	C4	F1
2008-053	Q016314	A	I	T2	P5	AD0	C5	F2
2008-053	Q016316	A	I	T2	P5	AD0	C7	F2
2008-053	Q016317	A	II	T2	P5	AD0	C7	F2
2008-053	Q016318	A	I	T4	P5	AD0	C4	F3
2008-053	Q016320	A	I	T4	P5	AD0	C5	F2
2008-053	Q016322	A	I	T4	P5	AD0	C7	F2
2008-053	Q016324	A	I	T4	P2	AD0	C5	F2
2008-053	Q016326	A	I	T4	P6	AD0	C5	F2
2008-053	Q016328	B	I	T2	P6	AD0	C4	F4
2008-053	Q016330	B	I	T2	P5	AD0	C5	F5
2008-053	Q016332	B	I	T2	P5	AD0	C7	F5
2008-053	Q016334	B	I	T4	P5	AD0	C4	F4
2008-053	Q016336	B	I	T4	P5	AD0	C5	F2
2008-053	Q016338	B	I	T4	P5	AD0	C7	F6
2008-053	Q016340	B	I	T4	P3	AD0	C5	F7
2008-053	Q016342	B	I	T4	P7	AD0	C5	F7

Figure 32. First portion of the encoded creation conditions table.



Project	Sample	Source Material	Processing Method	ppt Temp	ppt pH	Additive	Calcination Temp	Final Product
2009-038	Q019330	A	?	T1	P5	AD3	C6	F8
2009-038	Q019331	A	?	T1	P5	AD5	C6	F8
2009-038	Q019332	A	?	T4	P5	AD4	C6	F8
2009-038	Q019333	D	?	T1	P5	AD0	C6	F8
2009-038	Q019334	A	?	T1	P4	AD4	C6	F8
2009-038	Q019335	A	?	T1	P3	AD4	C6	F8
2009-038	Q019336	C	?	T1	P5	AD0	C6	F8
2009-038	Q019337	A	?	T4	P6	AD0	C6	F9
2009-038	Q019338	A	?	T4	P6	AD1	C6	F9
2009-038	Q019339	A	?	T1	P8	AD2	C6	F10
2009-038	Q019340	A	?	T1	P1	AD0	C6	F10
2009-038	Q019341	A	?	T5	P8	AD0	C6	F11
2009-038	Q019342	A	?	T1	P5	AD0	C6	F8
2009-038	Q019343	A	?	T4	P5	AD0	C3	F8
2009-038	Q019344	A	?	T4	P5	AD0	C1	F8
2009-038	Q019345	A	?	T1	P5	AD0	C2	F8
2009-038	Q019346	A	?	T5	P5	AD0	C2	F8

Figure 33. Second portion of the encoded creation conditions table.



Project	Sample	Source Material	Processing Method	ppt Temp	ppt pH	Additive	Calcination Temp	Final Product
2010-087	Q019908	A	?	T4	P5	AD0	C7	F2
2010-087	Q019909	A	?	T4	P5	AD0	C7	F2
2010-087	Q019910	A	?	T2	P5	AD0	C7	F2
2010-087	Q019911	A	?	T4	P5	AD0	C7	F2
2010-087	Q019912	A	?	T3	P5	AD0	C7	F12
2010-087	Q019913	A	?	T3	P5	AD0	C7	F2
2010-087	Q019914	B	?	T1	P6	AD0	C1	F13
2010-087	Q019915	B	?	T1	P6	AD0	C7	F2
2010-087	Q019916	B	?	T1	P6	AD0	C7	F2
2010-087	Q019917	A	?	T1	P1	AD0	C1	F12
2010-087	Q019918	A	?	T1	P6	AD0	C1	F12
2010-087	Q019919	A	?	T1	P6	AD0	C5	F2
2010-087	Q019920	B	?	T1	P6	AD0	C1	F14
2010-087	Q019921	B	?	T1	P6	AD0	C5	F2
2010-087	Q019922	B	?	T1	P6	AD0	C7	F2
2010-087	Q019923	B	?	T4	P5	AD0	C7	F2
2010-087	Q019924	A	?	T4	P5	AD0	C4	F15
2010-087	Q019925	A	?	T4	P5	AD0	C4	F15
2010-087	Q019926	A	?	T1	P5	AD0	C4	F15
2010-087	Q019927	A	?	T1	P5	AD0	C4	F15

Figure 34. Last group in the encoded creation conditions table.

Table 13. Particle statistics for the various Qs.

Q-class	Count in Class	Q-class	Count in Class
Q016316	70	Q019338	10
Q016317	16	Q019339	6
Q016312	15	Q019340	10
Q016314	11	Q019341	6
Q016318	15	Q019342	15
Q016320	8	Q019343	15
Q016322	11	Q019344	14
Q016324	9	Q019345	14
Q016326	8	Q019346	13
Q016328	16	Q019907	9
Q016330	12	Q019908	10
Q016332	10	Q019909	7
Q016334	15	Q019910	8
Q016336	11	Q019911	7
Q016338	8	Q019912	10
Q016340	10	Q019913	10
Q016342	10	Q019914	9
Q019329	10	Q019915	10
Q019330	5	Q019916	14
Q019331	10	Q019917	20
Q019332	10	Q019918	16
Q019333	10	Q019919	10
Q019334	10	Q019920	15
Q019335	9	Q019921	5
Q019336	10	Q019922	8
Q019337	10	Q019923	9
Q019924	14	Q019926	6
Q019925	12	Q019927	8

Table 14. The class distribution for the source materials creation condition.

Source materials	Particle count in class
UNH	477
UO2F2	162
UNH stripped from 30% TBP, 70% hexachlorobutadiene	10
UNH stripped from 30% TBP, 70% n-dodecane	10

Table 15. The class distribution for the precipitation temperature creation condition.

Precipitation temperature [°C]	Particle count in class
20	230
25	158
30	20
50	245
None	6

Table 16. The class distribution for the precipitation pH creation condition.

Precipitation pH	Particle count in class
1	30
5	9
6	28
6.5	10
8	429
9	131
11	10
None	12

Table 17. The class distribution for the additive creation condition.

Additive	Particle count in class
None	589
Gaseous Reactants	10
1 Mol HNO3	6
2 Mol Urea	5
4 Mol Urea	39
6 Mol Urea	10

Table 18. The class distribution for the calcination temperature creation condition.

Calcination temperature [°C]	Particle count in class
300	89
350	27
400	15
500	101
800	94
900	110
975	217
None	6

Table 19. The class distribution for the final product creation condition.

Final Product	Particle count in class
UO3	15
U308	251
UO3, UO4, U308	15
U308, U03, UO2(OH)2 hydrate, UO2F2	31
U3O8 w/ very minor UO2F2	22
U3O8 w/ minor UO2F2	8
U3O8 and UO2F2	20
ADU	145
AUC	20
UO4	16
N/A (UNH)	6
Schoepite	46
U Oxide	9
UO2F2 hydrate	15
U3O8, UO3	40

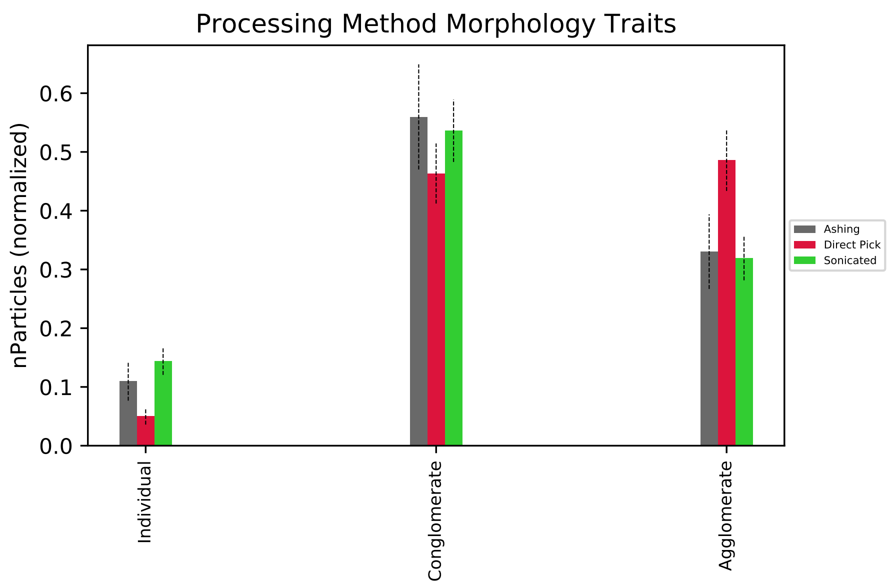


Figure 35. Processing method morphology feature occurrence for step 1 of the lexicon chart.

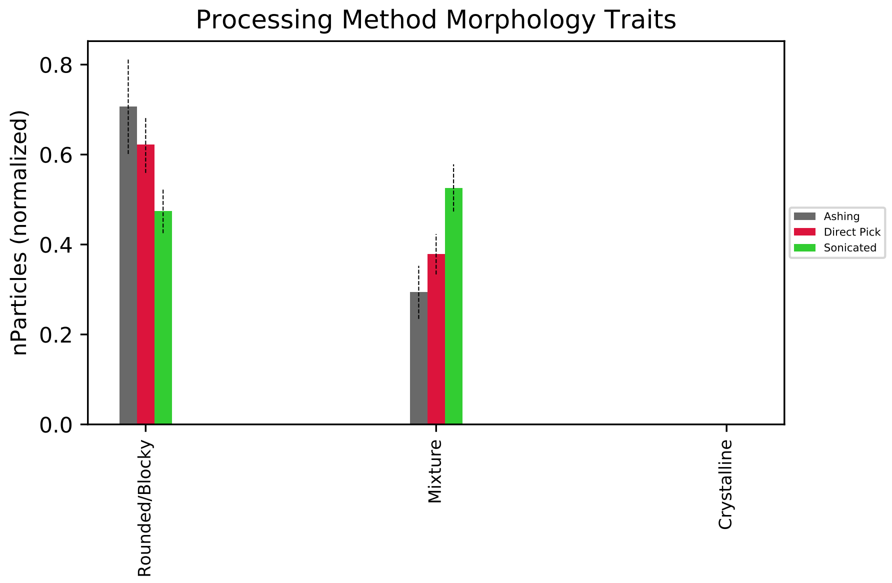


Figure 36. Processing method morphology feature occurrence for step 2 of the lexicon chart.

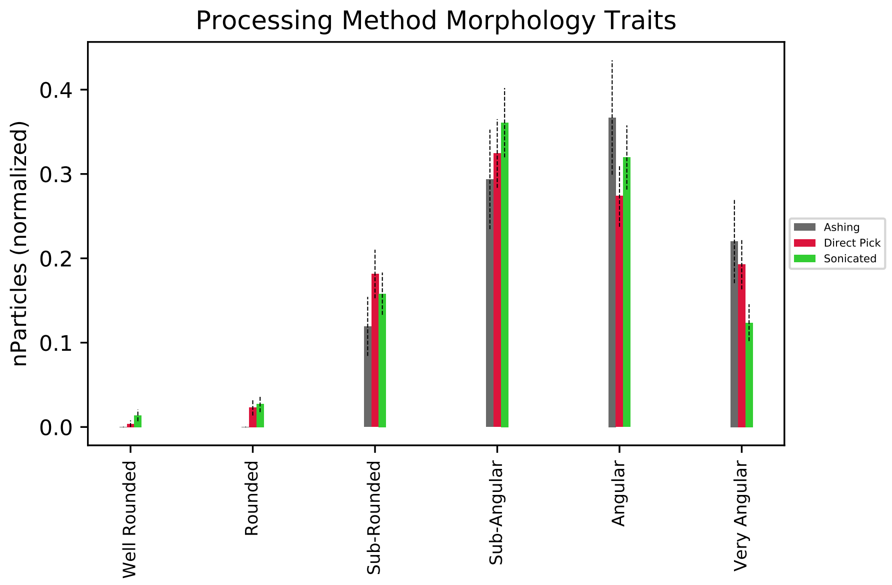


Figure 37. Processing method morphology feature occurrence for step 3 of the lexicon chart.

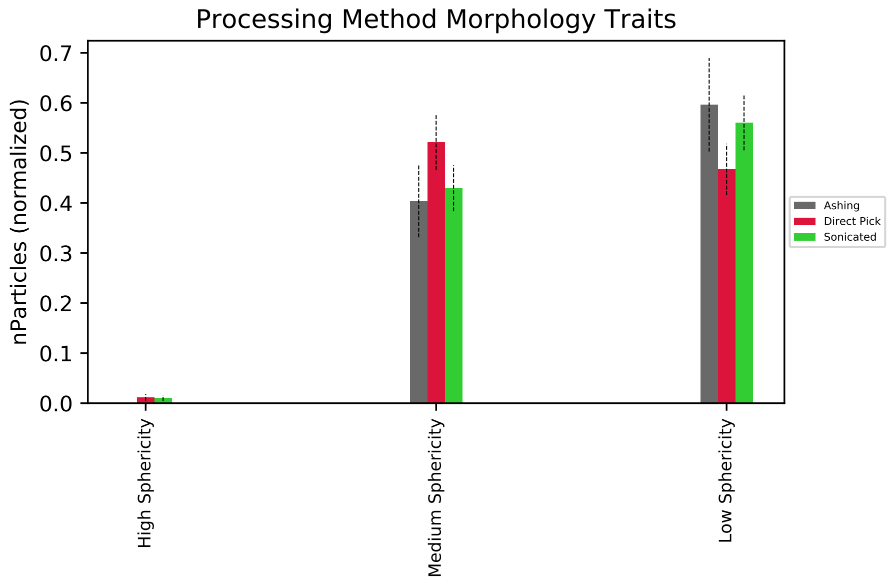


Figure 38. Processing method morphology feature occurrence for step 4 of the lexicon chart.

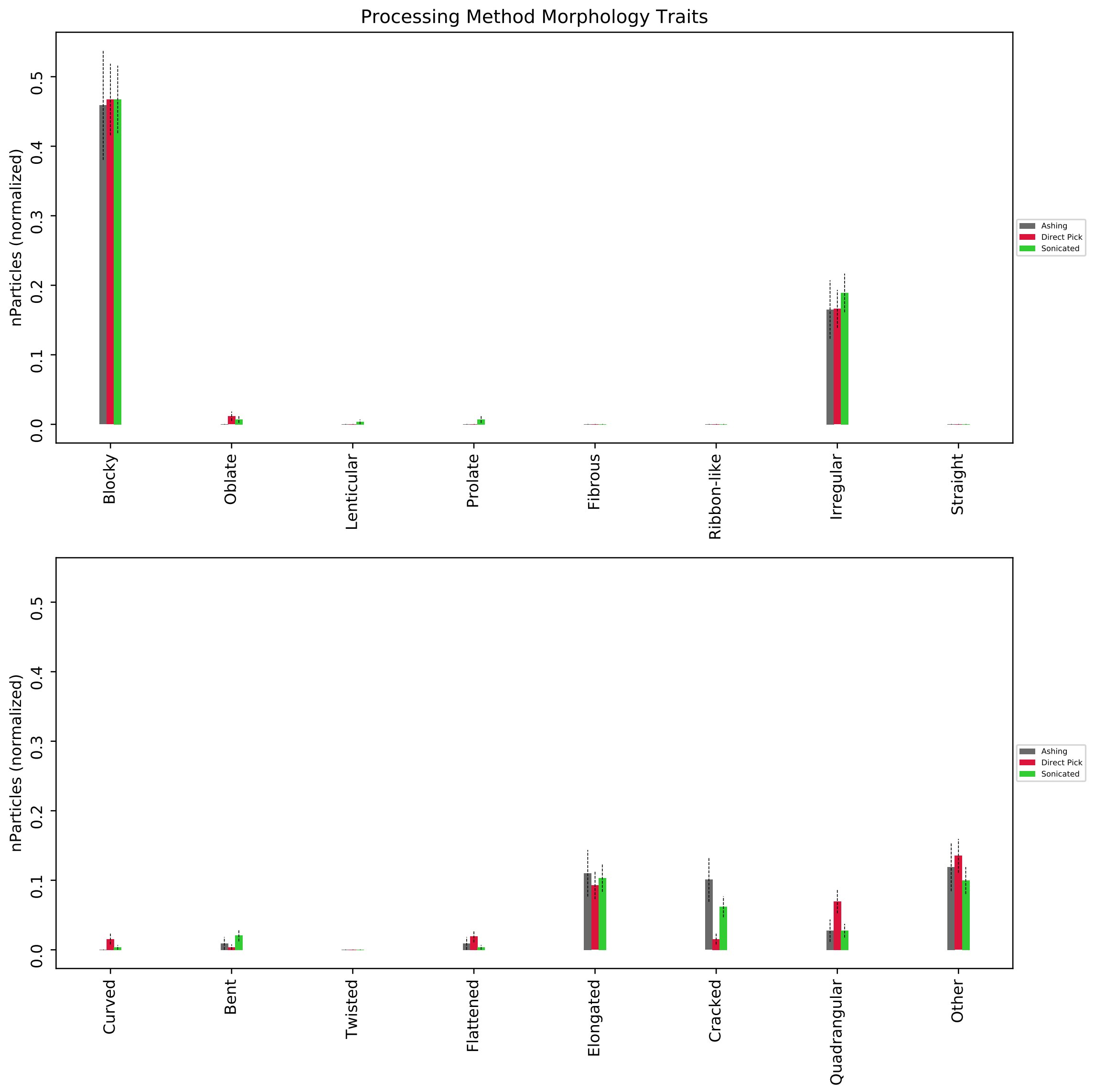


Figure 39. Processing method morphology feature occurrence for shape portion of step 4 in the lexicon chart.

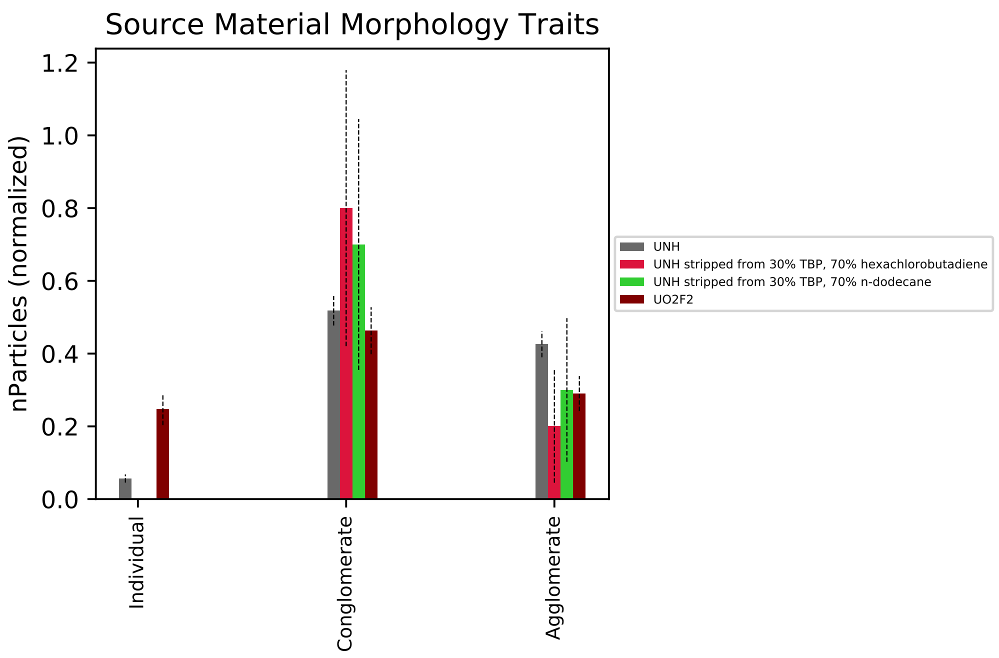


Figure 40. Source material morphology feature occurrence for step 1 of the lexicon chart.

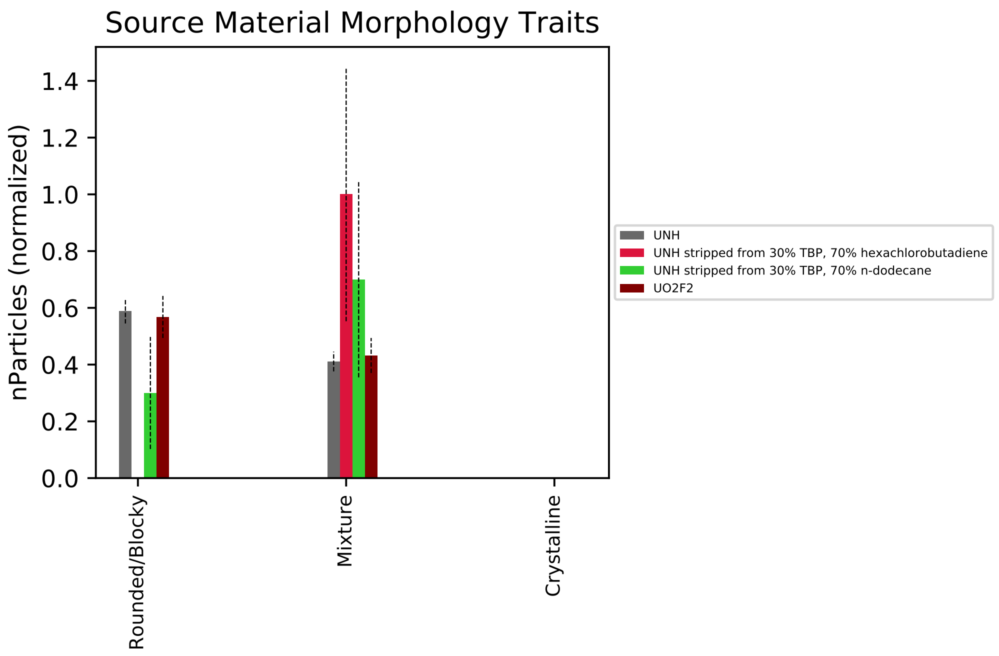


Figure 41. Source material morphology feature occurrence for step 2 of the lexicon chart.

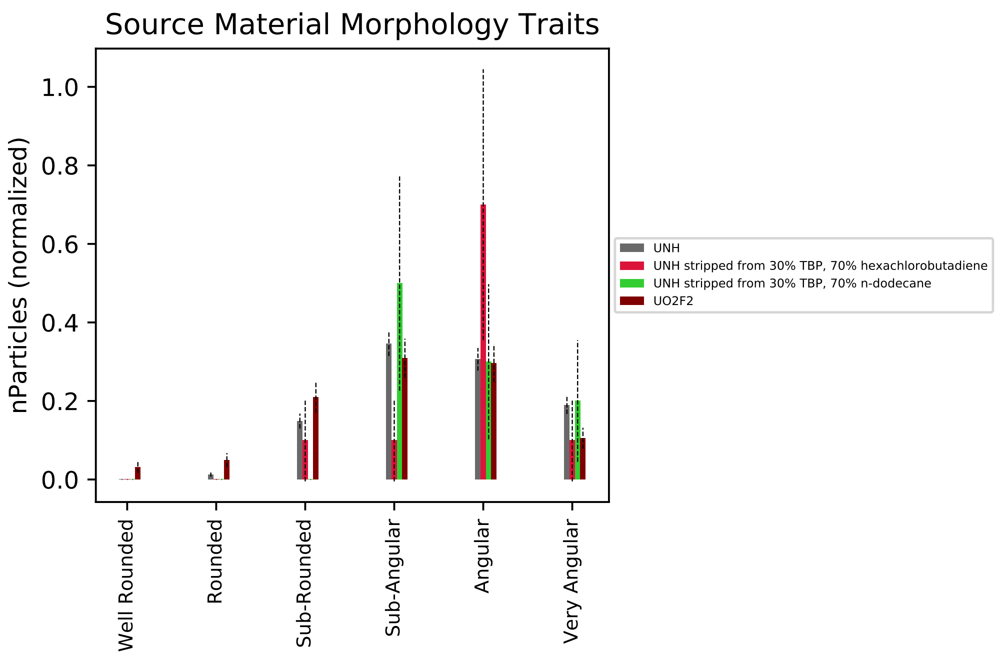


Figure 42. Source material morphology feature occurrence for step 3 of the lexicon chart.

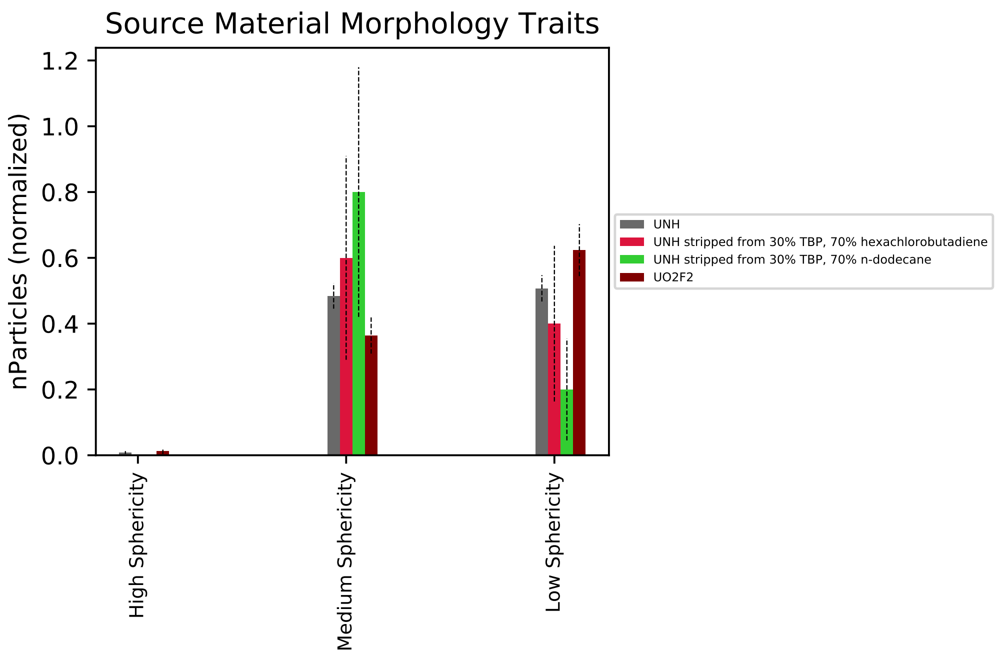


Figure 43. Source material morphology feature occurrence for step 4 of the lexicon chart.

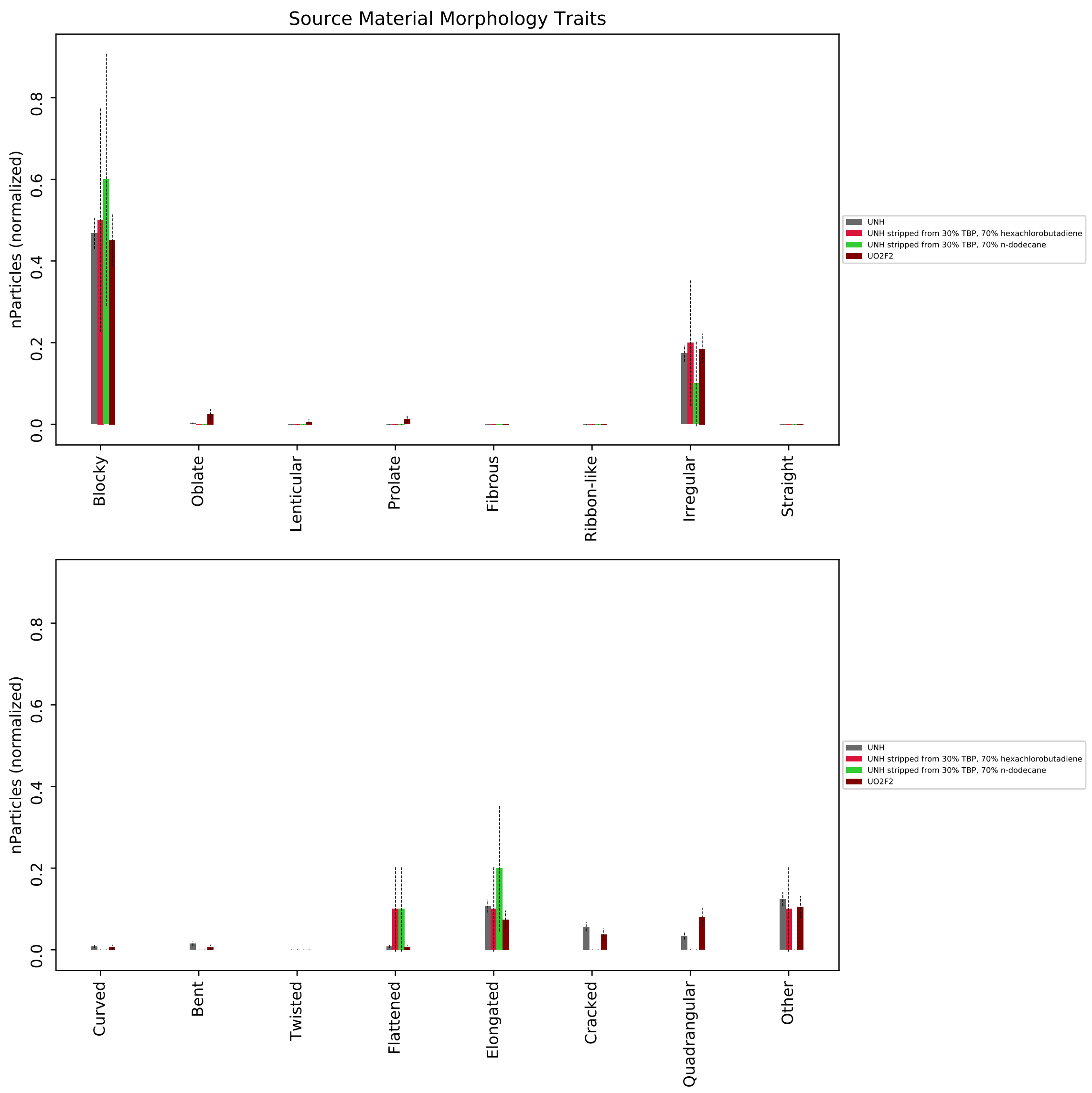


Figure 44. Source material morphology feature occurrence for shape portion of step 4 in the lexicon chart.

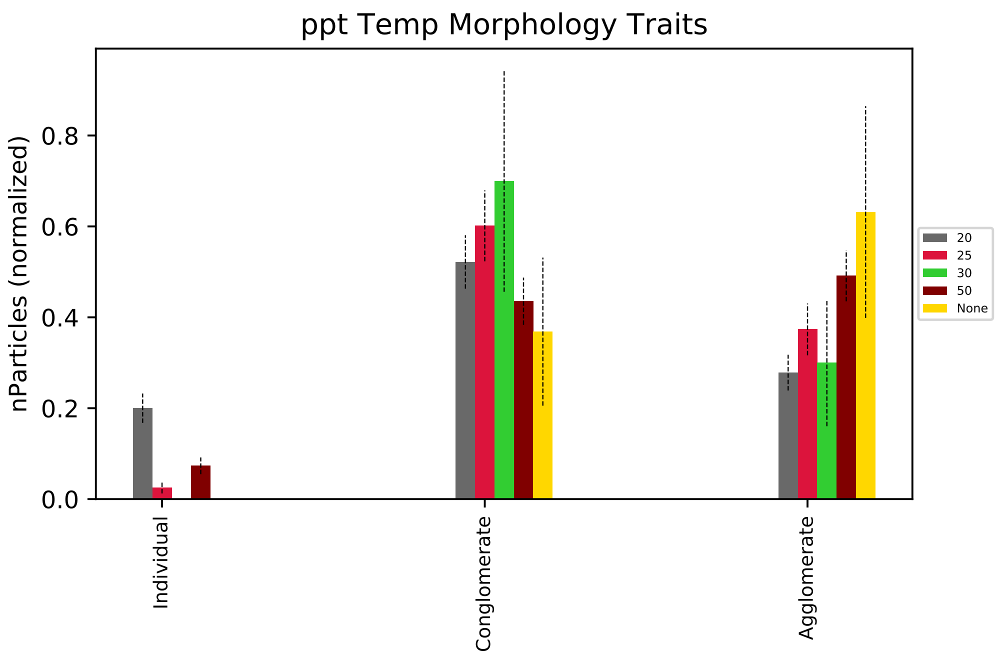


Figure 45. Precipitation temperature morphology feature occurrence for step 1 of the lexicon chart.

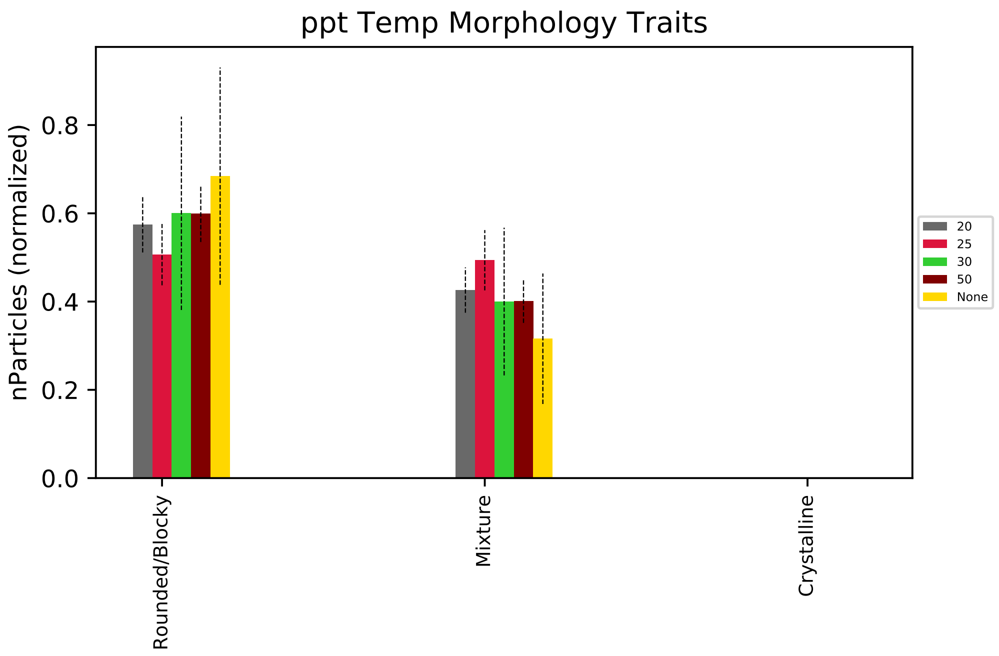


Figure 46. Precipitation temperature morphology feature occurrence for step 2 of the lexicon chart.

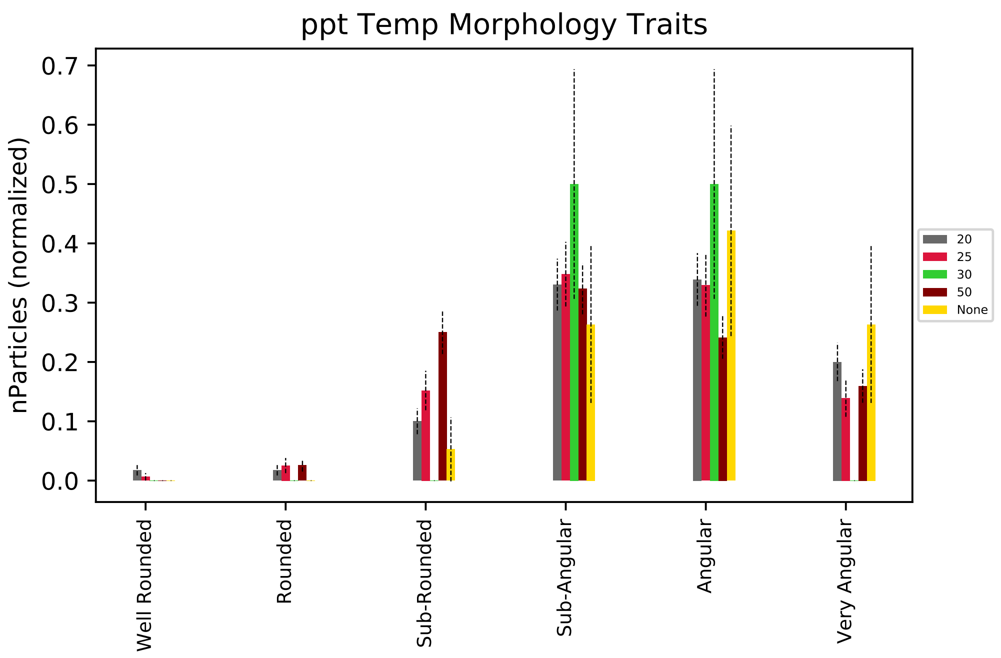


Figure 47. Precipitation temperature morphology feature occurrence for step 3 of the lexicon chart.

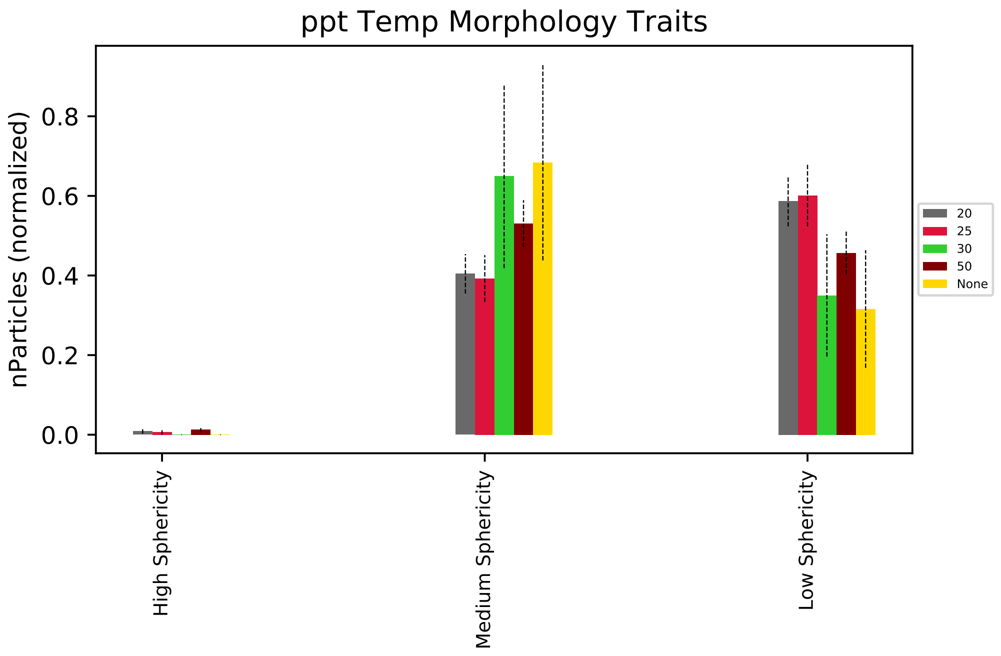


Figure 48. Precipitation temperature morphology feature occurrence for step 4 of the lexicon chart.

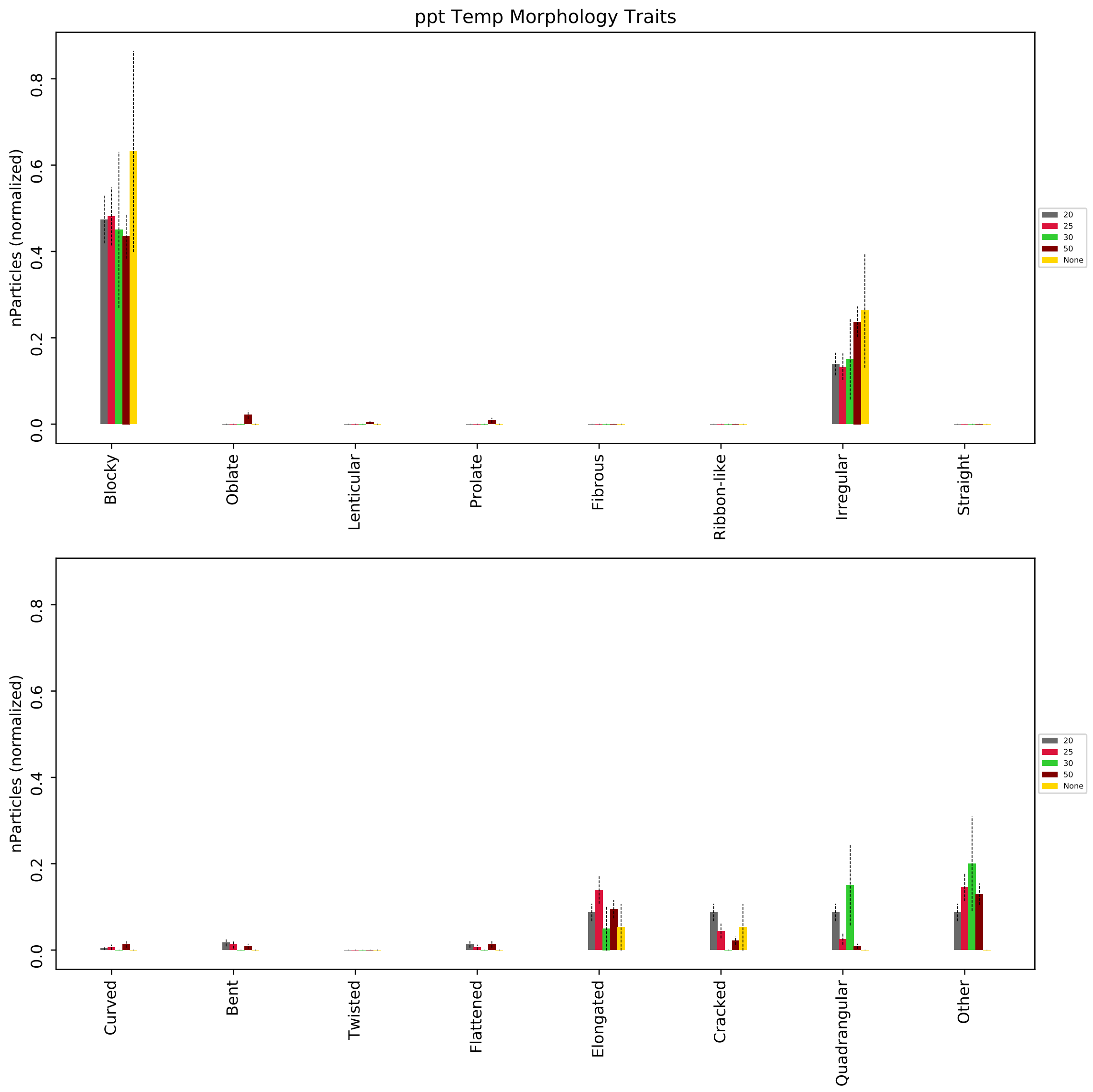


Figure 49. Precipitation temperature morphology feature occurrence for shape portion of step 4 in the lexicon chart.

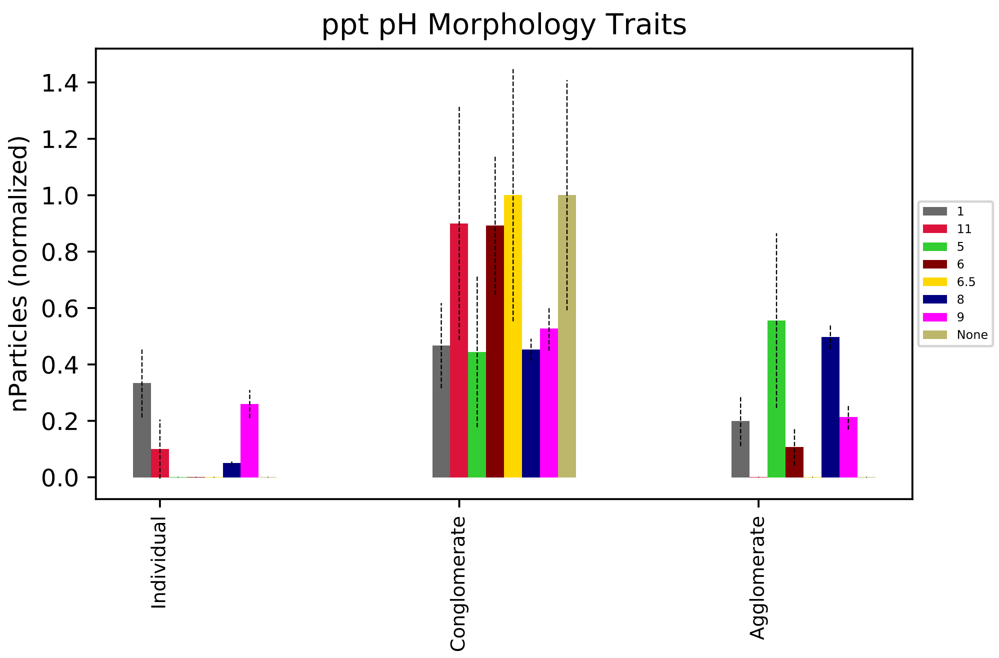


Figure 50. Precipitation pH morphology feature occurrence for step 1 of the lexicon chart.

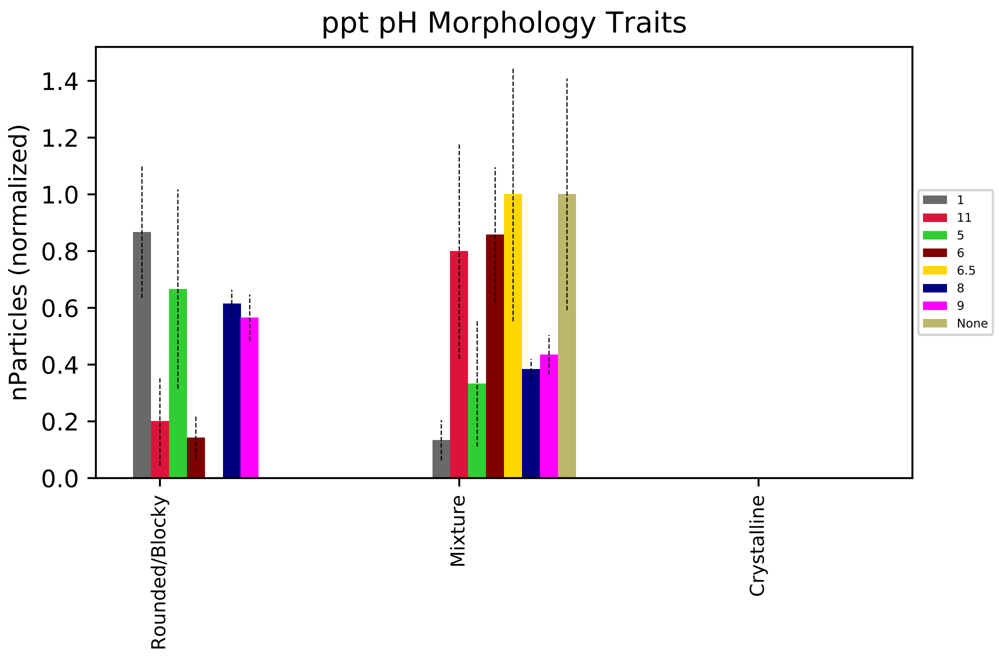


Figure 51. Precipitation pH morphology feature occurrence for step 2 of the lexicon chart.

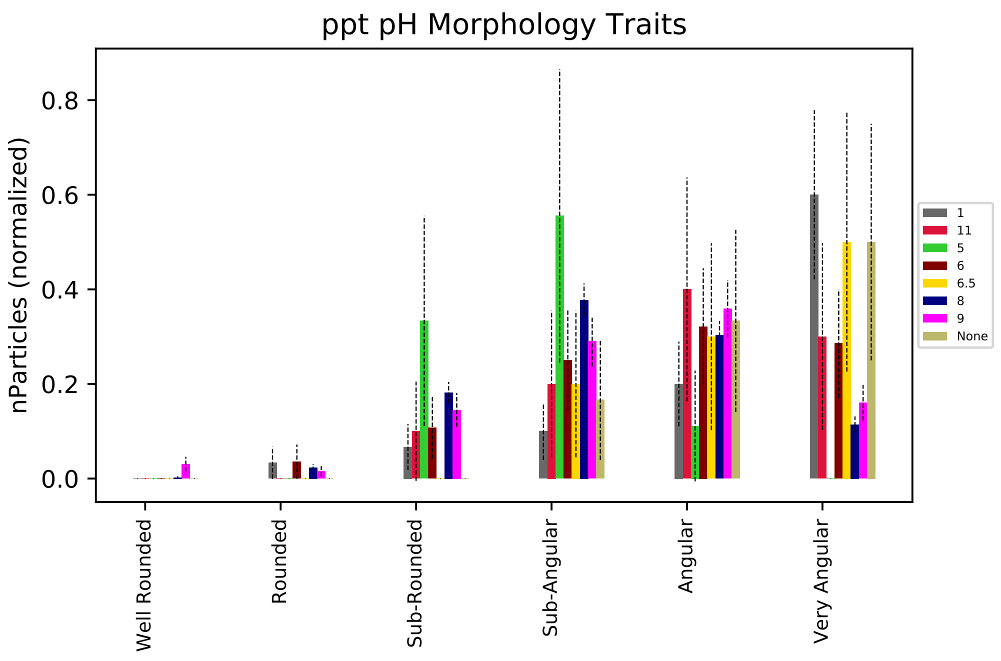


Figure 52. Precipitation pH morphology feature occurrence for step 3 of the lexicon chart.

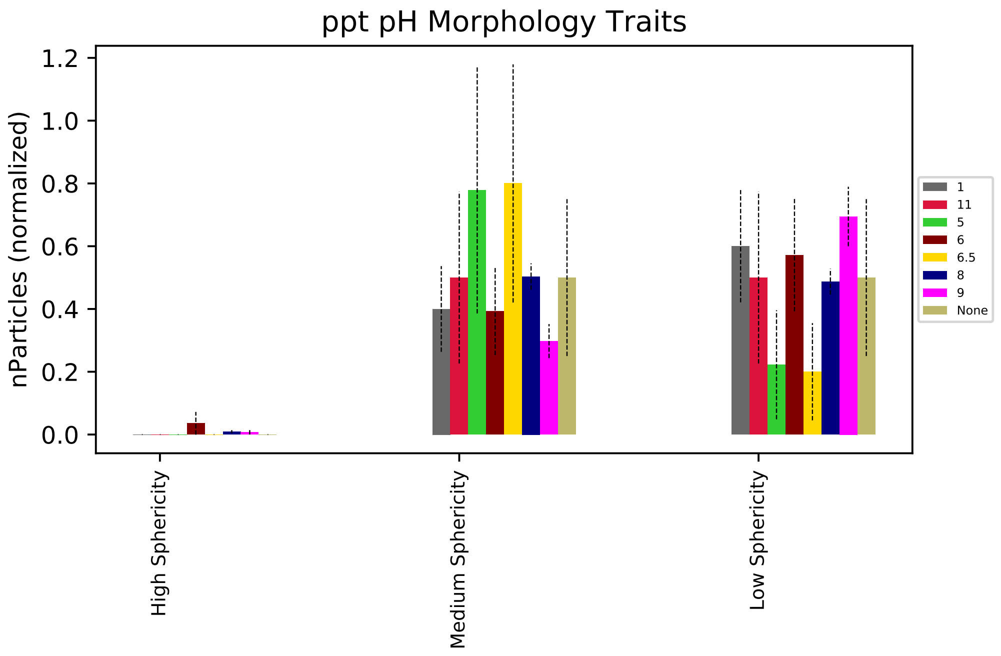


Figure 53. Precipitation pH morphology feature occurrence for step 4 of the lexicon chart.

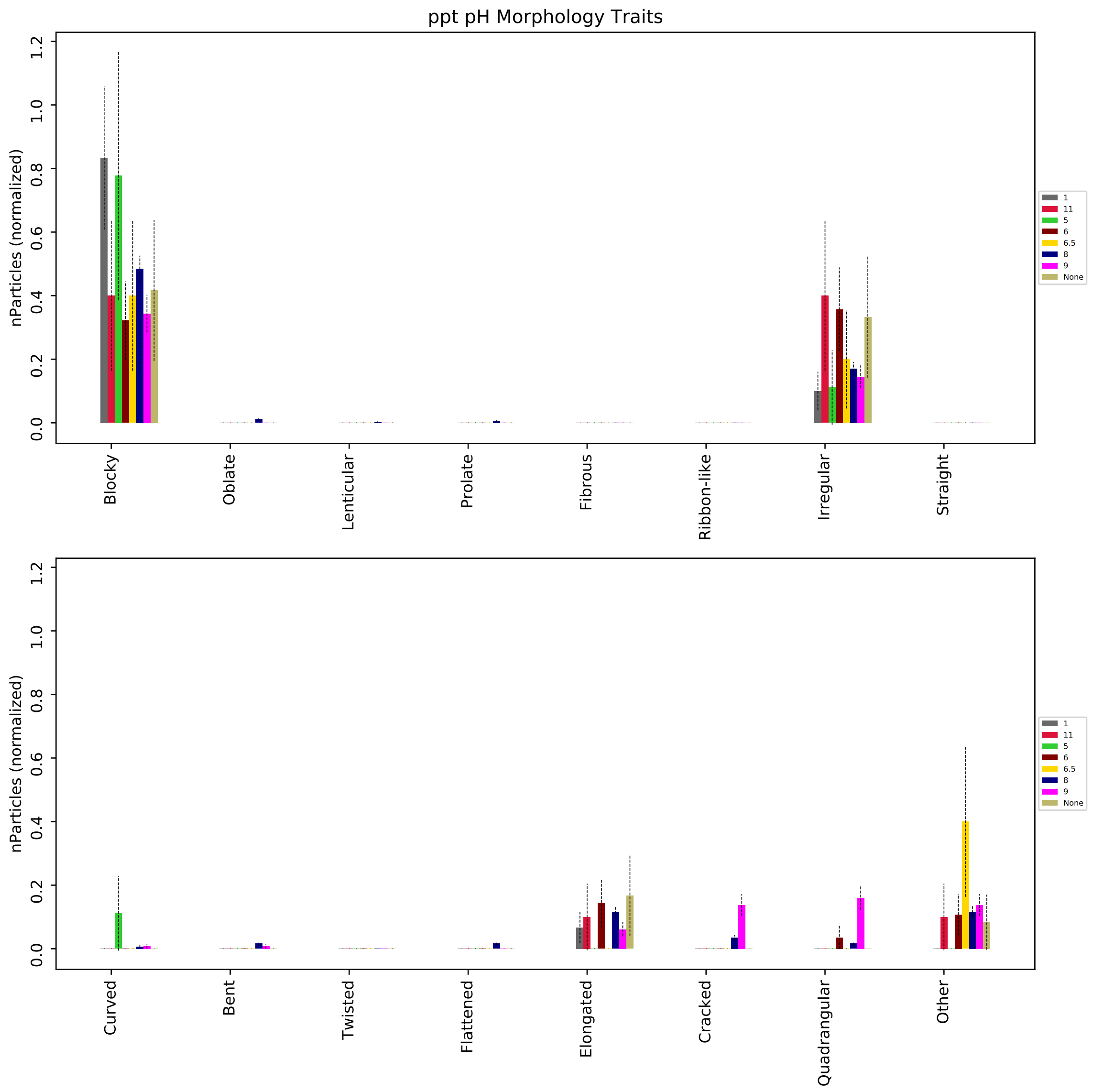


Figure 54. Precipitation pH morphology feature occurrence for shape portion of step 4 in the lexicon chart.

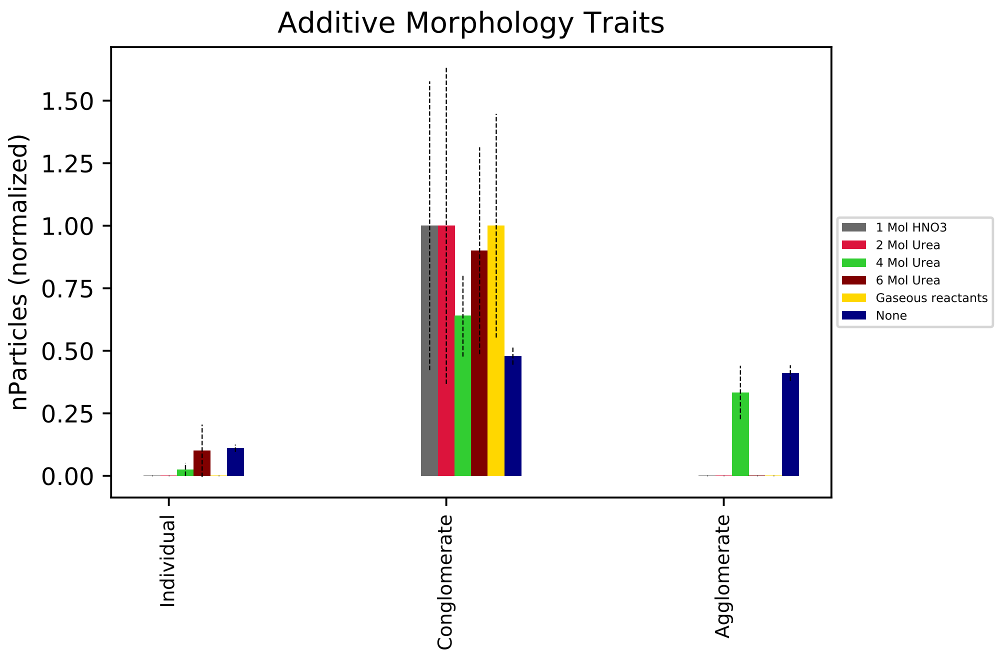


Figure 55. Additive morphology feature occurrence for step 1 of the lexicon chart.

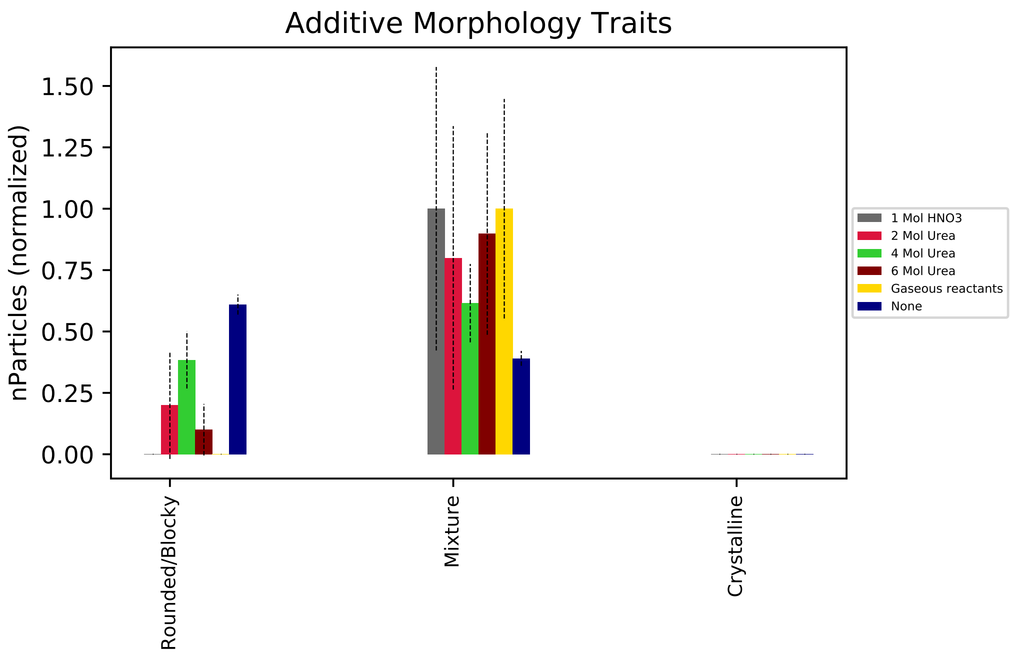


Figure 56. Additive morphology feature occurrence for step 2 of the lexicon chart.

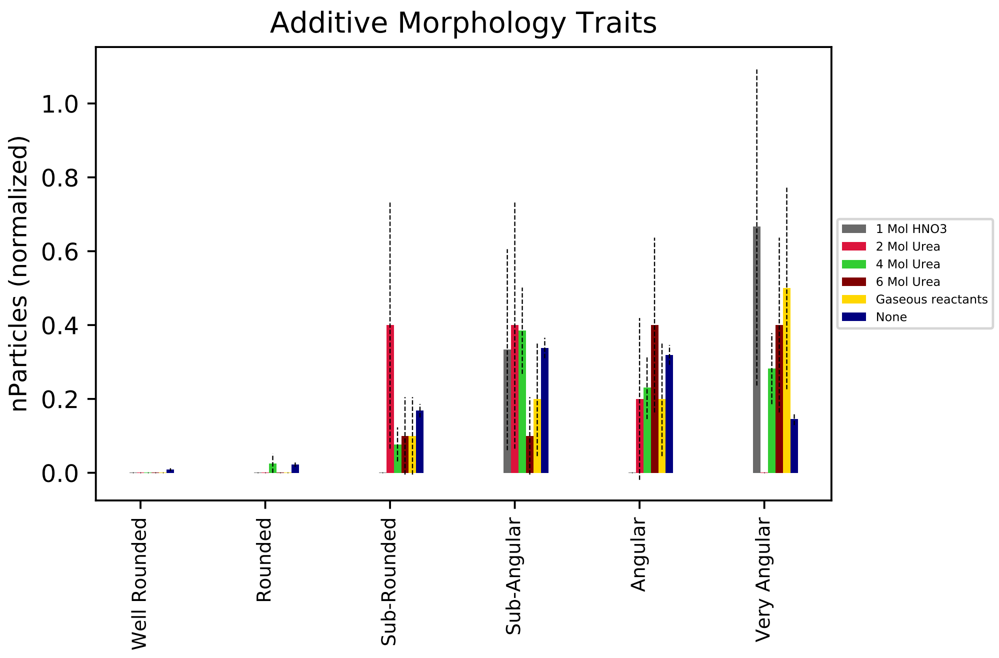


Figure 57. Additive morphology feature occurrence for step 3 of the lexicon chart.

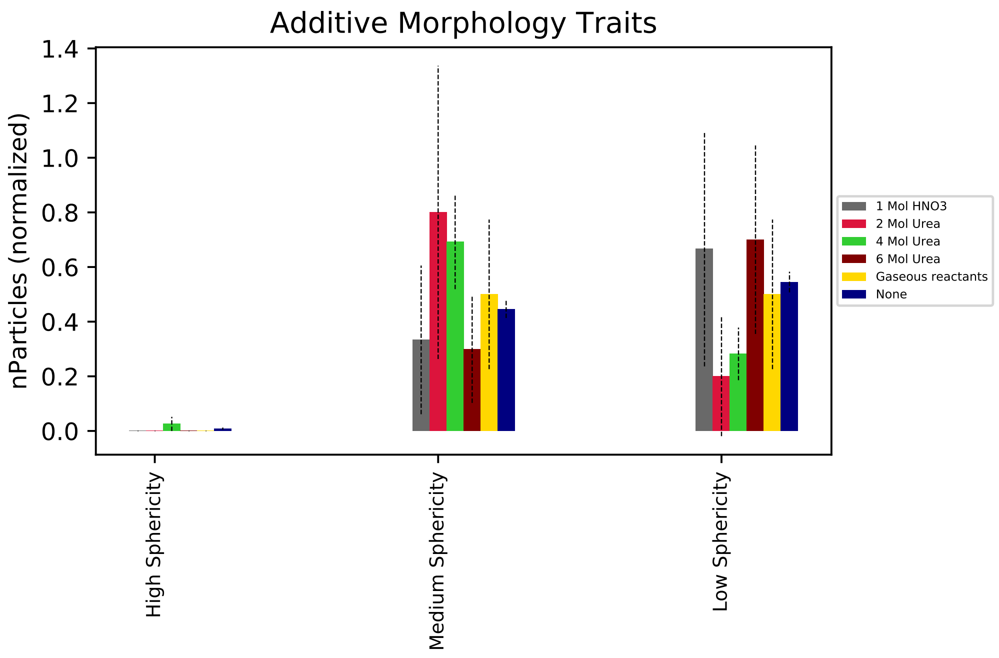


Figure 58. Additive morphology feature occurrence for step 4 of the lexicon chart.

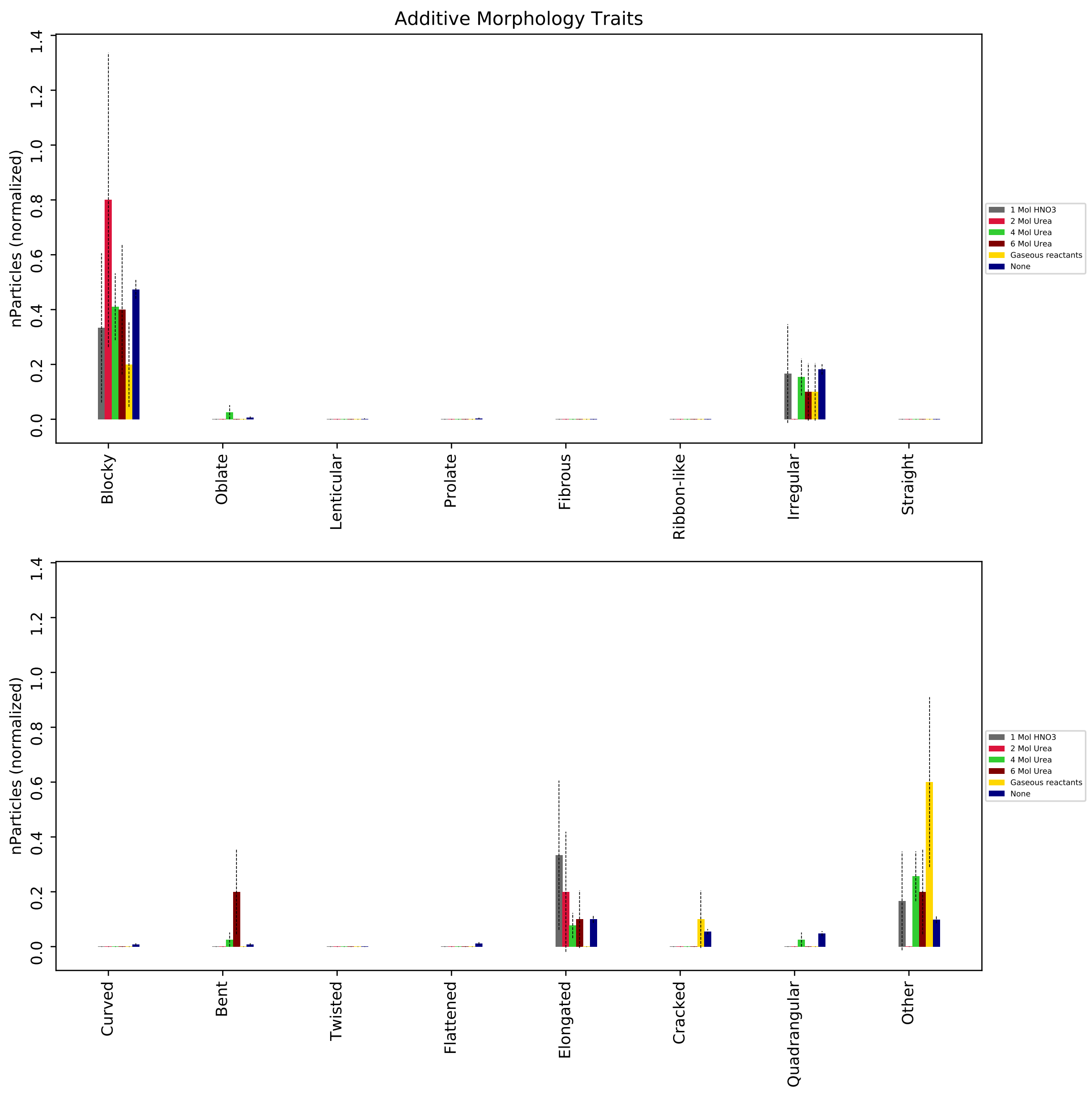


Figure 59. Additive morphology feature occurrence for shape portion of step 4 in the lexicon chart.

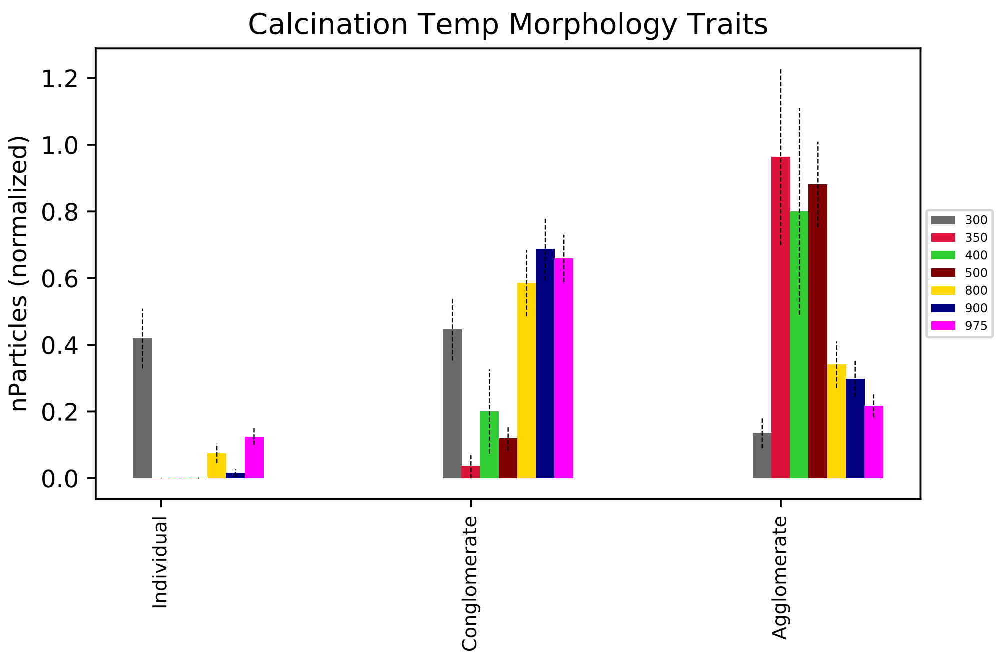


Figure 60. Calcination temperature morphology feature occurrence for step 1 of the lexicon chart.

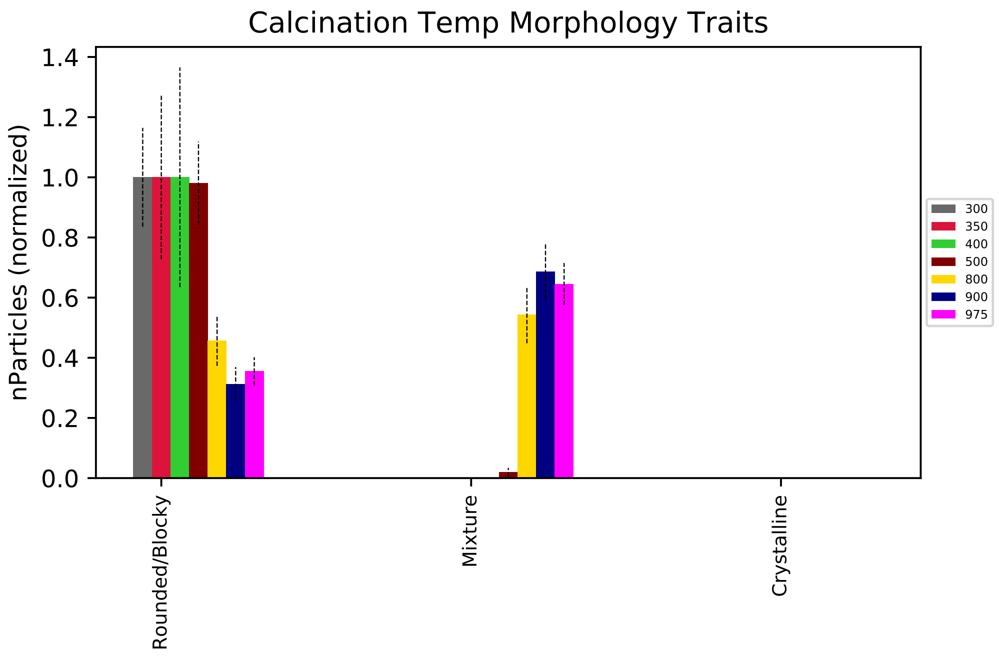


Figure 61. Calcination temperature morphology feature occurrence for step 2 of the lexicon chart.

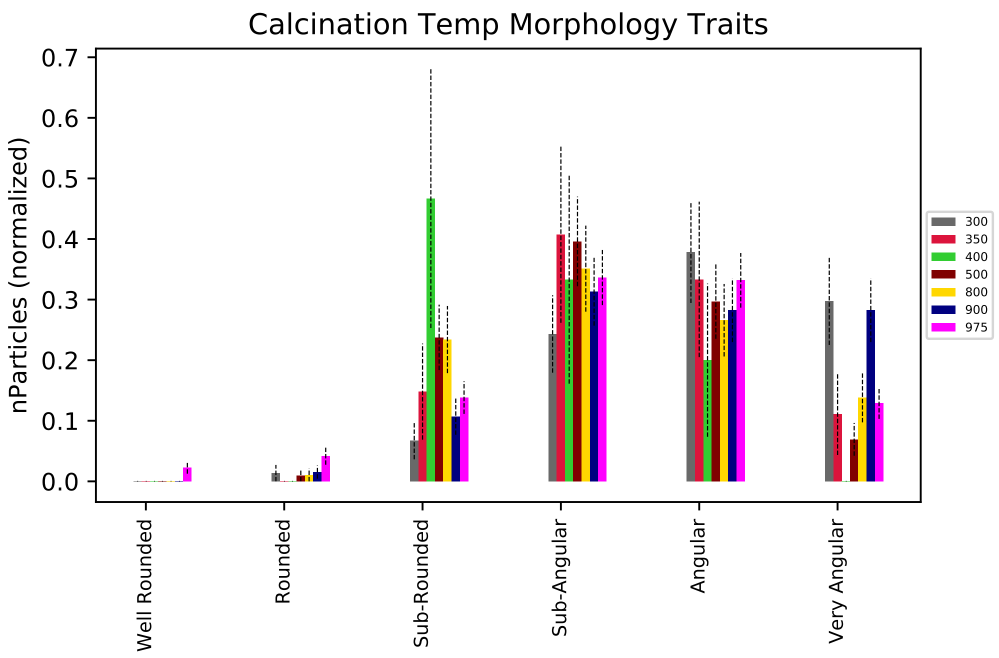


Figure 62. Calcination temperature morphology feature occurrence for step 3 of the lexicon chart.

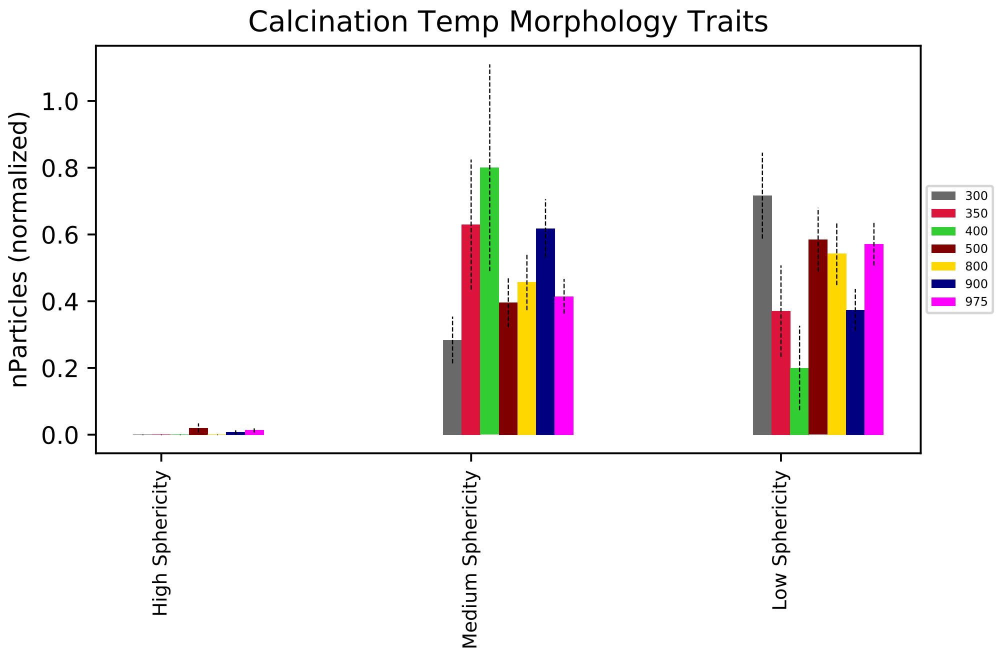


Figure 63. Calcination temperature morphology feature occurrence for step 4 of the lexicon chart.

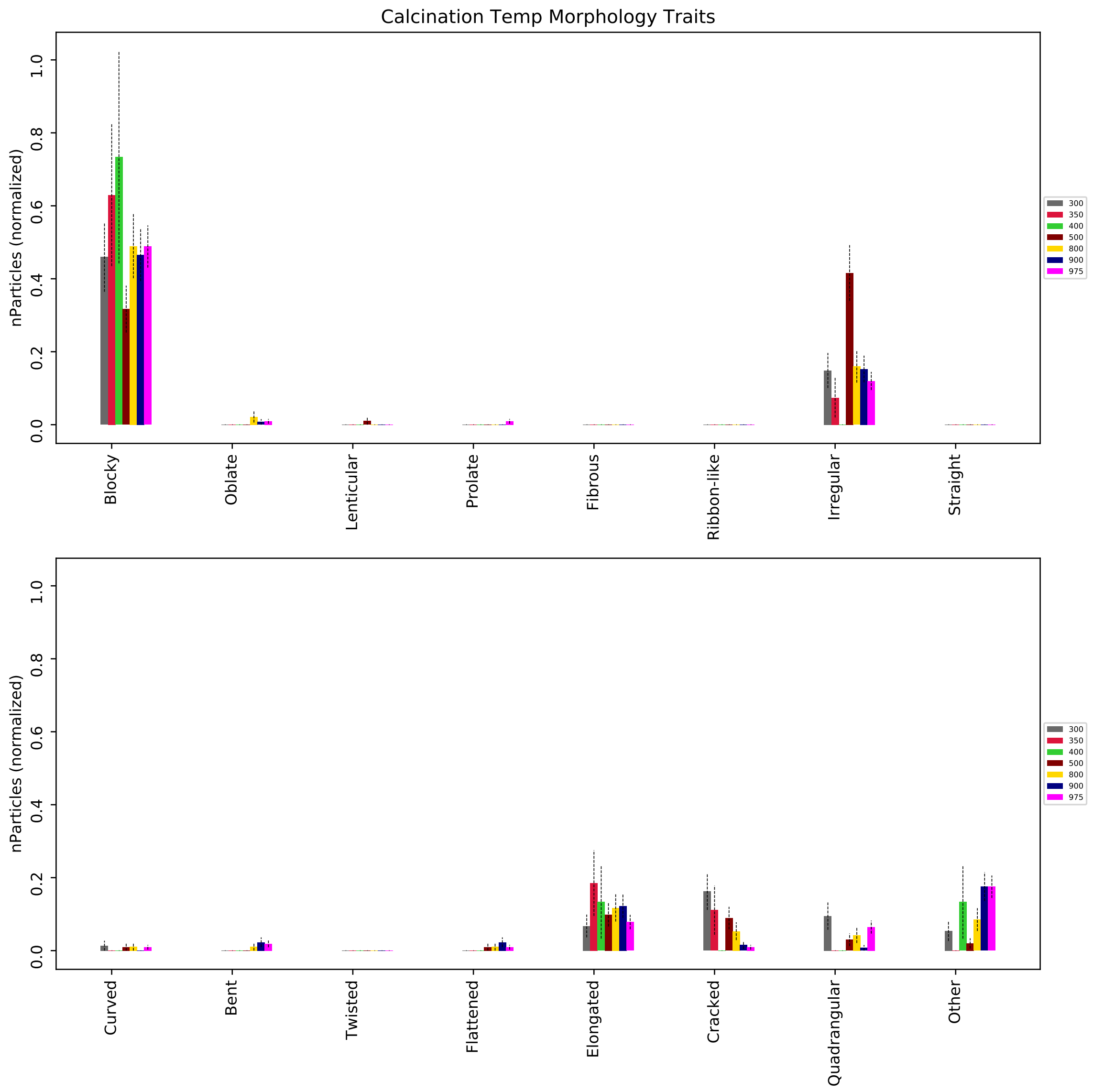


Figure 64. Calcination temperature morphology feature occurrence for shape portion of step 4 in the lexicon chart.

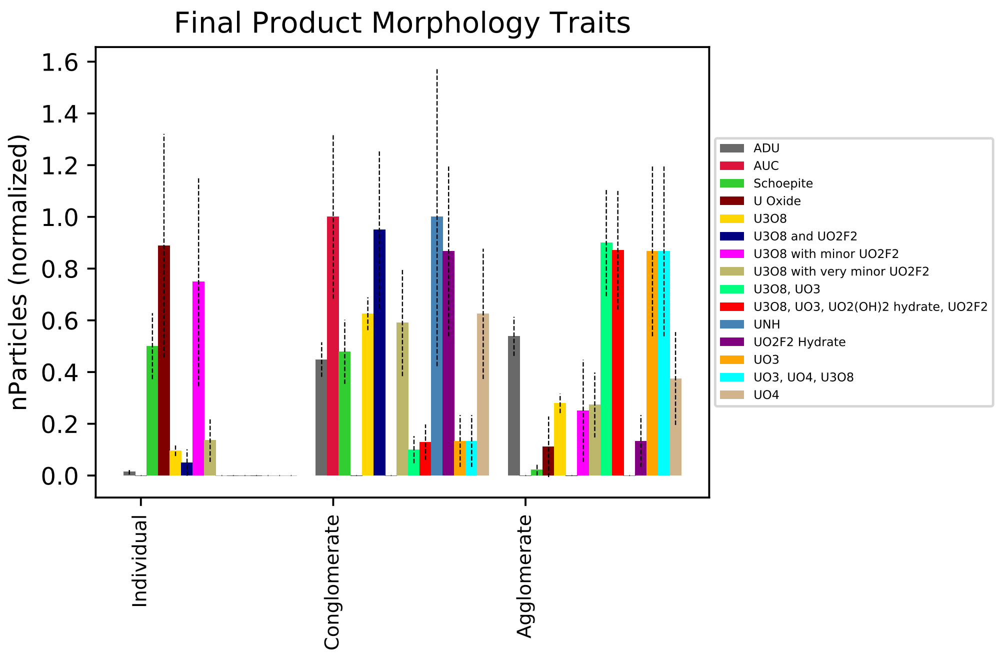


Figure 65. Final product morphology feature occurrence for step 1 of the lexicon chart.

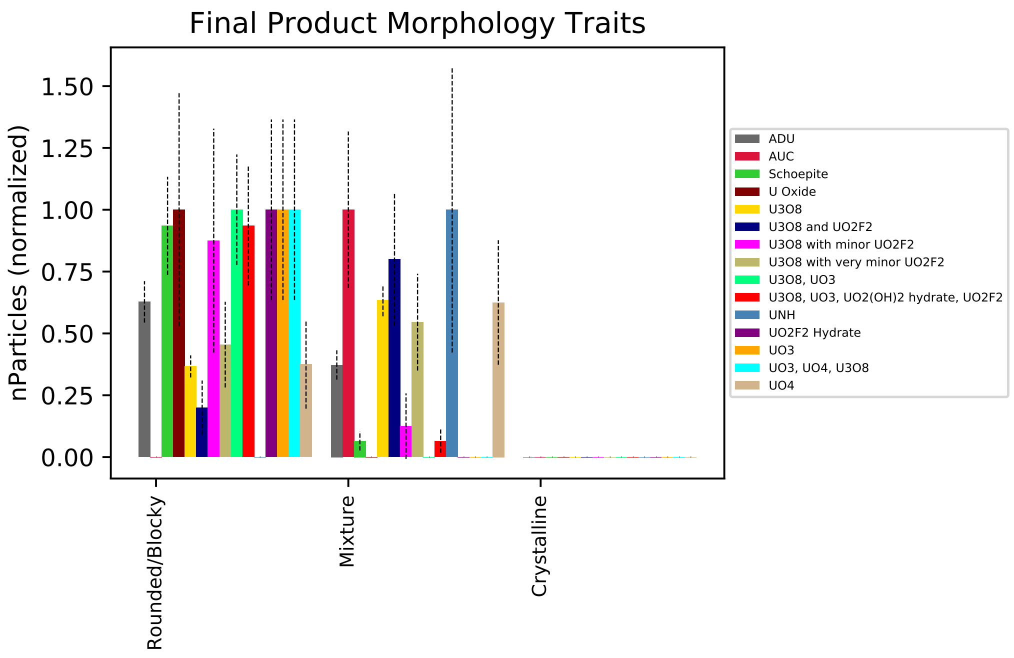


Figure 66. Final product morphology feature occurrence for step 2 of the lexicon chart.

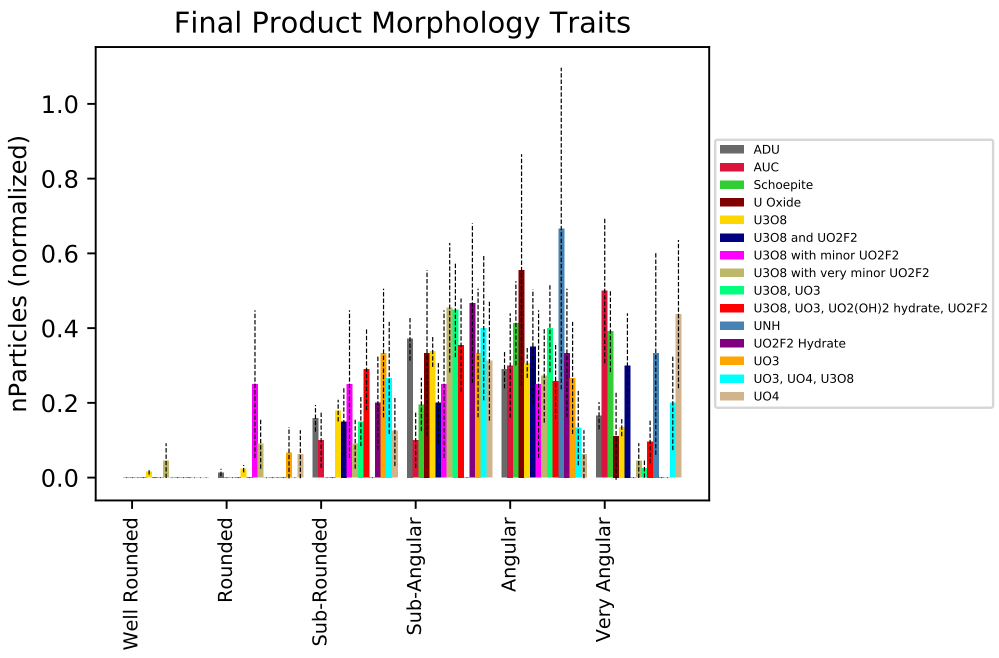


Figure 67. Final product morphology feature occurrence for step 3 of the lexicon chart.

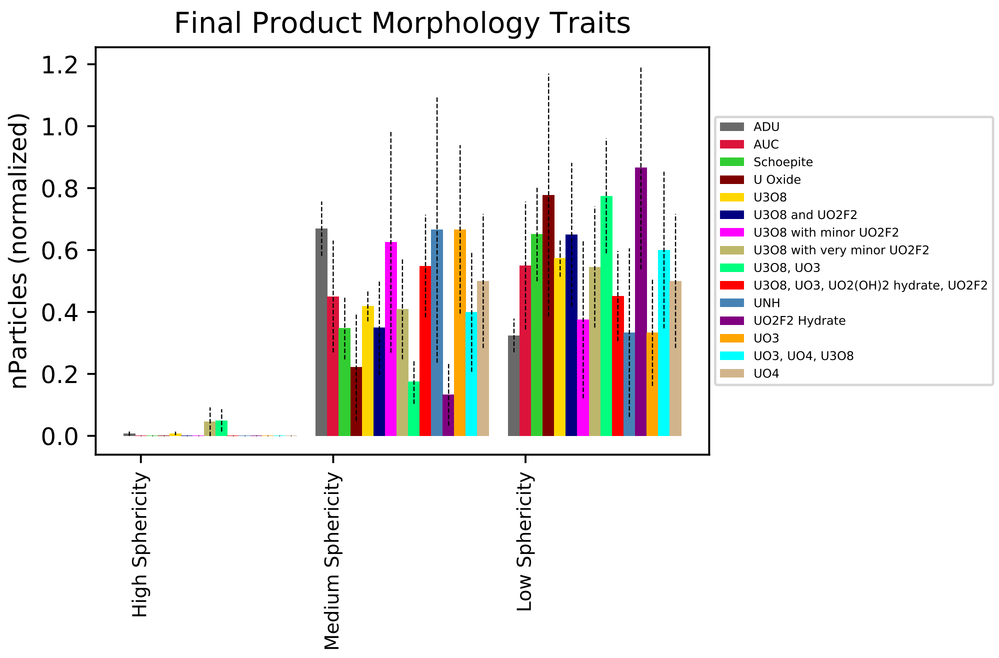


Figure 68. Final product morphology feature occurrence for step 4 of the lexicon chart.

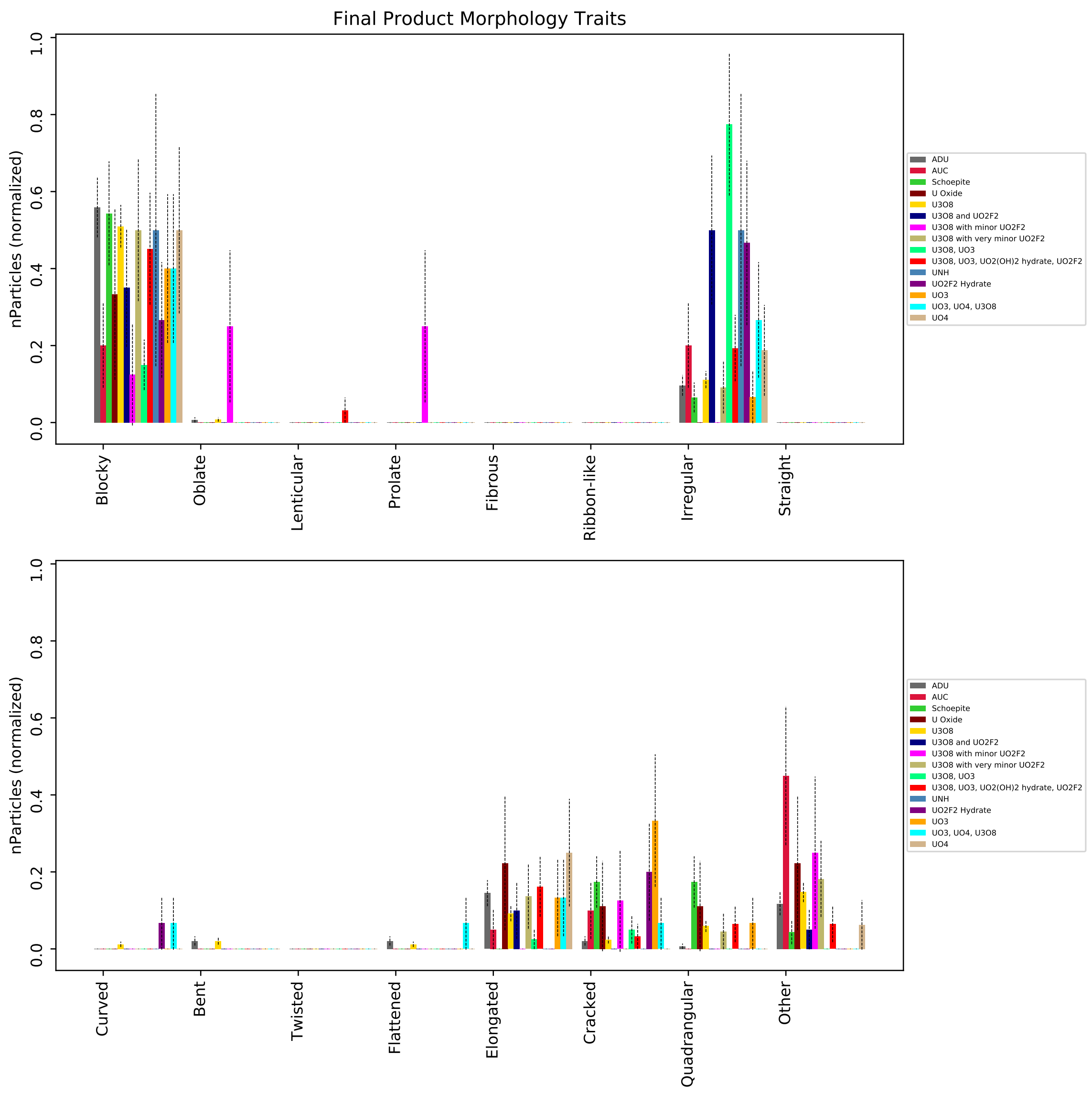


Figure 69. Final product morphology feature occurrence for shape portion of step 4 in the lexicon chart.

Appendix C. Glossary

Table 20. Commonly used terms throughout this thesis and their definitions.

Term/Abbreviation	Definition
Project Number	ex: 2010-87
Sample ID	ex: Q019907
NIT ID	ex: C8020
Class	machine learning category that data belongs to ex: dog, cat, bird
Feature/Predictor	characteristics that the data may take on ex: cost, size, weight
Random Forest	a group of decision trees with random tweaks in each tree
Conglomerate	a particle consisting of many dissimilar sub-particles
Agglomerate	a mostly homogeneous mass of particles held together
ADU	Ammonium diuranate or $(NH_4)_2U_2O_7$
AUC	Ammonium uranyl carbonate or $UO_2CO_3 \cdot 2(NH_4)_2CO_3$
UNH	Uranyl nitrate hexahydrate or $UO_2(NO_3)_2 \cdot 6H_2O$
Schoepite	$(UO_2)_8O_2(OH)_{12} \cdot 12(H_2O)$

Appendix D. GitHub Repository

The machine learning python scripts used in the research can be found in a private GitHub repository located at:

https://github.com/dgum/Research/tree/master/Code

To access the repository and the files held therein, contact the author for access. The main **Research** directory contains the following subdirectories of note:

- **Code:** containing the python files for analysis and use of machine learning techniques on the lexicon encoded dataset.
- **Readings:** containing many different journal articles referenced in the course of creating this thesis document.
- **ExcelFiles:** containing csv and txt files that hold many of the datasets imported into the python scripts.
- **Images:** containing all the graphics used throughout the thesis.

Bibliography

1. U. N. R. Commission, *Information Digest, 2018–2019*, Aug 2018.
2. J. T. Vanderplas, *Python data science handbook: essential tools for working with data.* OReilly, 2017.
3. A. Karpathy, "CS231n: Convolutional Neural Networks for Visual Recognition." Dec. 2019. [Online]. Available: https://github.com/cs231n/cs231n.github.io
4. A. L. Tamasi, L. J. Cash, C. Eley, R. B. Porter, D. L. Pugmire, A. R. Ross, C. E. Ruggiero, L. Tandon, G. L. Wagner, and J. R. Walensky, "A lexicon for consistent description of material images for nuclear forensics," *Journal of Radioanalytical and Nuclear Chemistry*, vol. 307, no. 3, pp. 1611–1619, 2015.
5. V. Fedchenko, *The new nuclear forensics: analysis of nuclear materials for security purposes.* Oxford university press, 2015.
6. Statistics Denmark, "SIPRI Yearbook 2016," Tech. Rep., 2016.
7. K. Mayer, M. Wallenius, and I. Ray, "Nuclear forensics-a methodology providing clues on the origin of illicitly trafficked nuclear materials," *The Analyst*, Dec 2004.
8. R. Schmitt, "Scanning electron microscope," *CIRP Encyclopedia of Production Engineering*, Jan 1970.
9. H. Hansen, K. da Costa Carneiro, H. Haitjema, and L. De Chiffre, "Dimensional micro and nano metrology," vol. 55, no. 2. Elsevier B.V., 2006, pp. 721–743.
10. *Uranium 2018.* Nuclear Energy Agency and International Atomic Energy Agency. [Online]. Available: https://www.oecd-ilibrary.org/content/publication/uranium-2018-en
11. J. Whitlock and A. Lee, "Candu®: Setting the standard for proliferation resistance of generation iii and iii+ reactors," 2009.
12. J. J. Duderstadt and L. J. Hamilton, *Nuclear Reactor Analysis.* Wiley & Sons, 1976.
13. *Management of High Enriched Uranium for Peaceful Purposes: Status and Trends*, ser. TECDOC Series. Vienna: International Atomic Energy Agency, 2005, no. 1452.
14. *IAEA Safeguards Glossary*, ser. International Nuclear Verification Series. Vienna: International Atomic Energy Agency, 2003, no. 3.
15. *Management of Reprocessed Uranium Current Status and Future Prospects*, ser. TECDOC Series. Vienna: International Atomic Energy Agency, 2007, no. 1529.

16. R. Bellman, *Dynamic Programming.* Princeton University Press, 1984.

17. M. Benedict, T. H. Pigford, and H. W. Levi, *Nuclear chemical engineering.* McGraw-Hill, 1981.

18. E. Keegan, M. J. Kristo, K. Toole, R. Kips, and E. Young, "Nuclear forensics: Scientific analysis supporting law enforcement and nuclear security investigations," *Analytical Chemistry*, vol. 88, no. 3, pp. 1496–1505, 2016.

19. Z. Karpas, *Analytical Chemistry of Uranium: Environmental, Forensic, Nuclear, and Toxicological Applications.* Taylor & Francis, Nov 2014.

20. G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning: With Applications in R.* Springer, 2017.

21. Y.-y. Song, "Decision tree methods: applications for classification and prediction," *Shanghai Archives of Psychiatry*, vol. 27, no. 2, pp. 130–135, Apr 2015.

22. L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct 2001.

23. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

24. Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, 1997.

25. T. Hastie, J. Friedman, and R. Tisbshirani, *The Elements of statistical learning: data mining, inference, and prediction.* Springer, 2017.

26. F. Provost, "Machine learning from imbalanced data sets 101," vol. 68. Proceedings of the AAAI 2000 workshop on imbalanced data sets., 2000.

27. A. Fernandez, F. Herrera, and N. V. Chawla, "Smote for learning from imbalanced data: Progress and challenges, marking the 15-year anniversary," *Journal of Artificial Intelligence Research*, Apr 2018.

28. N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, Jun 2002.

29. H. He, Y. Bai, E. A. Garcia, and S. Li, "Adasyn: Adaptive synthetic sampling approach for imbalanced learning," *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pp. 1322–1328, 2008.

30. K. H. Brodersen, C. S. Ong, K. E. Stephan, and J. M. Buhmann, "The balanced accuracy and its posterior distribution," *20th International Conference on Pattern Recognition*, pp. 3121–3124, 2010.

31. R. Akbani, S. Kwek, and N. Japkowicz, "Applying support vector machines to imbalanced datasets," in *Machine Learning: ECML 2004*, J.-F. Boulicaut, F. Esposito, F. Giannotti, and D. Pedreschi, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 39–50.

32. N. Japkowicz and S. Stephen, "The class imbalance problem: A systematic study," *Intelligent Data Analysis*, pp. 429–449, 2002.

33. R. J. Urbanowicz and J. H. Moore, "Exstracs 2.0: description and evaluation of a scalable learning classifier system," *Evolutionary Intelligence*, vol. 8, no. 2-3, p. 89–116, Mar 2015.

34. F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain." *Psychological review*, vol. 65, no. 6, p. 386, 1958.

35. S. Shaban, N. M. Ibrahiem, S. El-Mongy, and E. E. Elshereafy, "Validation of scanning electron microscope (sem), energy dispersive x-ray (edx) and gamma spectrometry to verify source nuclear material for safeguards purposes," *Journal of Radioanalytical and Nuclear Chemistry*, vol. 296, 2012.

36. J. Bowen, S. Glover, and H. Spitz, "Morphology of actinide-rich particles released from the bomarc accident and collected from soil post remediation," *Journal of Radioanalytical and Nuclear Chemistry*, vol. 296, no. 2, pp. 853–857, 2012.

37. T. Tsuey-Lin, T.-Y. Lin, T.-Y. Su, T.-J. Wen, L.-C. Men, and C.-C. Lin, "Chemical and radiochemical analysis of corrosion products on bwr fuel surfaces," *Journal of Radioanalytical and Nuclear Chemistry*, vol. 295, no. 2, 2012.

38. S. Paik, S. Biswas, S. Bhattacharya, and S. Roy, "Effect of ammonium nitrate on precipitation of ammonium di-uranate (adu) and its characteristics," *Journal of Nuclear Materials*, vol. 440, no. 1-3, pp. 34–38, 2013.

39. S. Manna, P. Karthik, A. Mukherjee, J. Banerjee, S. B. Roy, and J. B. Joshi, "Study of calcinations of ammonium diuranate at different temperatures," *Journal of Nuclear Materials*, vol. 426, no. 1-3, pp. 229–232, 2012.

40. S. Manna, S. B. Roy, and J. B. Joshi, "Study of crystallization and morphology of ammonium diuranate and uranium oxide," *Journal of Nuclear Materials*, vol. 424, no. 1-3, pp. 94–100, 2012.

41. K. M. Krupka, M. A. Parkhurst, K. Gold, B. W. Arey, E. D. Jenson, and R. A. Guilmette, "Physicochemical characterization of capstone depleted uranium aerosols iii: Morphologic and chemical oxide analyses," *Health Physics*, vol. 96, no. 3, pp. 276–291, 2009.

42. N. A. Miller and J. J. Henderson, “Quantifying sand particle shape complexity using a dynamic, digital imaging technique,” *Agronomy Journal*, vol. 102, no. 5, pp. 1407–1414, 2010.

43. D. Shoji, R. Noguchi, S. Otsuki, and H. Hino, “Classification of volcanic ash particles using a convolutional neural network and probability,” *Scientific Reports*, vol. 8, no. 1, May 2018.

44. I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning.* MIT Press, 2017.

45. Mizkozawa Akira, “Th-makerx.” [Online]. Available: http://www5.wind.ne.jp/miko/index-en.html

46. J. R. Taylor, *An Introduction to Error Analysis: The Study of The Uncertainties in Physical Measurements*, 2nd ed. University Science, 1997.

47. M. Claesen and B. D. Moor, “Hyperparameter search in machine learning,” 2015.

48. G. E. Hinton, *A Practical Guide to Training Restricted Boltzmann Machines.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 599–619.

49. C.-w. Hsu, C.-c. Chang, and C.-J. Lin, “A practical guide to support vector classification,” 2003.

50. I. J. Schwerdt, A. Olsen, R. Lusk, S. Heffernan, M. Klosterman, B. Collins, S. Martinson, T. Kirkham, and L. W. McDonald, “Nuclear forensics investigation of morphological signatures in the thermal decomposition of uranyl peroxide,” *Talanta*, vol. 176, pp. 284 – 292, 2018.

51. K.-W. Kim, J.-T. Hyun, K.-Y. Lee, E.-H. Lee, K.-W. Lee, K.-C. Song, and J.-K. Moon, “Effects of the different conditions of uranyl and hydrogen peroxide solutions on the behavior of the uranium peroxide precipitation,” *Journal of Hazardous Materials*, vol. 193, pp. 52 – 58, 2011.

52. A. M. Olsen, B. Richards, I. Schwerdt, S. Heffernan, R. Lusk, B. Smith, E. Jurrus, C. Ruggiero, and L. W. Mcdonald, “Quantifying morphological features of α-U_3O_8 with image analysis for nuclear forensics,” *Analytical Chemistry*, vol. 89, no. 5, pp. 3177–3183, Oct 2017.

53. E. Cordfunke and A. van der Giessen, “Particle properties and sintering behaviour of uranium dioxide,” *Journal of Nuclear Materials*, vol. 24, no. 2, pp. 141 – 149, 1967.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)* 03-26-2020	**2. REPORT TYPE** Master's Thesis	**3. DATES COVERED** *(From — To)* Sept 2018 - March 2020

4. TITLE AND SUBTITLE A Machine Learning Approach to Characterizing Particle Morphology in Nuclear Forensics.	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Gum, Daniel, A, 1st Lt	**5d. PROJECT NUMBER**
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology, Graduate School of Engineering and Management (AFIT/EN), 2950 Hobson Way, WPAFB OH 45433-7765	**8. PERFORMING ORGANIZATION REPORT NUMBER** AFIT-ENP-MS-20-M-099
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Intentionally Left Blank	**10. SPONSOR/MONITOR'S ACRONYM(S)**
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

A machine learning approach is taken to characterizing a group of synthetic uranium bearing particles. SEM images of these lab-created particles were converted into a binary representation that captured morphological features in accordance with a guide established by Los Alamos National Laboratory. Each particle in the dataset contains an association with chemical creation conditions: processing method, precipitation temperature and pH, calcination temperature are most closely tied to particle morphology. Additionally, trained classifiers are able to relate final products between particles, implying that morphological features are shared between particles with similar composition. The results show that machine learning classifiers can still successfully differentiate these creation conditions when combined into groups and considered as a whole, a result which could serve as a valuable utility to analysts working on large forensic datasets.

15. SUBJECT TERMS

Nuclear Forensics, Machine Learning, Particle Morphology, Classification, Oversampling, Actinide Chemistry

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Dr. Abigail A. Bickley, AFIT/ENP
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			**19b. TELEPHONE NUMBER** *(include area code)*
U	U	U	U	99	(937) 255 3636 x4555 abigail.bickley@afit.edu