Deep Learning Final Project: Powder/Particle Classification

Aidan Horn [1] **Eri Kim** [1] **Adeline Evans** [1]

Abstract

This paper examines multiple techniques for improving multi-instance classification on imbalanced classes using deep neural networks. It compares statistical methods of dimensionality reduction like T-distributed Stochastic Neighbor Embedding with deep learning approaches like auto-encoders and graph-convolutional neural networks in improving classification accuracy. We demonstrate that a combination of statistical and deep dimensionality reduction achieves high precision and recall. We seek to contribute these findings to the Worcester Polytechnic Institute Army Research Lab to expedite powder testing and discovery.

1. Introduction

The purpose of our research is to explore deep learning approaches to predict the flowability and flow rate for cold spray powders. Each powder is constituted of hundreds to thousands of particles, whose individual physical properties are measured. We seek to infer the properties of these powders based on particle measurements. Of interest is a classification of powders by *flowability*, whether the powder is able to be cold-sprayed. Previous experiments have yielded 33 samples of powders, each composed of their constituent particle measurements. This can be formulated as a supervised, multi-instance classification problem, wherein each powder has a binary flow class that must be applied to each particle instance. Previous research done at WPI has looked into many simple machine-learning models and methods with linear regression, simple clustering, correlation matrices, and some rudimentary neural networks. Neural networks had the most success, but results were limited, indicating it is a complex problem and that further manipulation and technique are needed to achieve better results. Success in this research enables scientists to evaluate candidates for cold spray additive manufacturing without needing to physically test the powder.

[1]Worcester Polytechnic Institute. Correspondence to: Aidan Horn <ahorn@wpi.edu>, Eri Kim <ekim4@wpi.edu>, Adeline Evans <amevans@wpi.edu>.

CS/DS 541 Deep Learning Final Project, Worcester Polytechnic Institute. Copyright 2022 by the author(s).

2. Related Work

Within the scope of particle flow classification, previous methods have exploited the central tendency of the particle features within each powder, describing each powder by the mean or median of its constituent particle features. This approach was tested using leave-one-out cross-validation, wherein feed-froward neural networks were trained on a subset of *N - 1* powder means, then tested on the excluded powder. This approach of training using statistically derived particles validates the applicability dimensionality reduction to this classification task, however it is limited in its capability to categorize powder mixtures whose features are multimodally distributed whereby a mean transformation would destroy relevant information and potentially misrepresent the particle.

There are a number of applicable techniques to the problem of dimensionality reduction for classification. Within the domain of statistical machine learning, approaches like T-distributed Stochastic Neighbor Embedding (T-SNE) can be applied as prepossessing step before passing the transformed data to a neural network. Non-linear dimensionality reduction can also be achieved solely using a neural network structure, such as an auto-encoder, which attempts to embed the data within a smaller latent space, then reconstruct the original input from this latent embedding. The ability of an auto-encoder to reconstruct its input has been used to great effect in anomaly detection (Sakurada, Mayu and Yairi, Takehisa, 2014) wherein a class instance not present in the training data will be poorly encoded resulting in a high reconstruction error which signals an outlier.

Instead of reduction, some methods attempt to extend the dimensionality of the data by encoding relationships between the instances within a graph structure, adding a spacial continuity which can be exploited for inference. Graph-based neural network approaches have been thoroughly studied for prediction tasks, such as in Design Space for Graph Neural Networks (You, Jiaxuan and Ying, Rex and Leskovec, Jure), which proposes a graph-convolutional neural-network architecture which uses neighboring node properties to create low dimensional node embeddings. This paper recommends different variations of their proposed structure depending on similarity to a given type of GNN prediction task. In the case of particle flow class classification, we referred to architectures for node classification.

3. Proposed Method

3.1. Flow Class Classification

To improve flow classification (0 for no flow and 1 for flow) of a particle, we will test numerous techniques for dimensionality reduction for preprocessing. As a baseline, we developed a multi-layer perceptron (MLP) model due to its ability to classify data that is not linearly separable. Given this baseline, we will test different methods of preprocessing, namely statistical dimensionality reduction using T-SNE in comparison to an auto-encoded embedding.

We will also employ autoencoders as naïve classifiers using the principle of anomaly detection by reconstruction error. Positive instances will be separated into equally sized training and testing sets for an auto-encoder. Once trained, negative instances can be fed to the auto-encoder, the reconstructions of which will be compared to the original input. We hypothesize that negative instances will have a noticeably higher average reconstruction loss than positive instances used during testing.

3.2. Powder Classification by Binned Flow Rate

Additionally, we sought to classify powders within a discretized range of possible flow rates represented in the dataset. There were seven flow rate classes, [0, 4, 16, 24, 32, 40, 54]. This transforms the problem into a multi-instance, multi-class classification problem. Our approach employed a graph-based convolutional neural network, using an architecture recommended by You et al. First, the data is encoded as bidirectional graph, with each node corresponding an individual particle vector. A subset of all possible edges are then instantiated randomly. This is done both out of run-time consideration and adherence to the sparse-networks used in the reference paper. The edges are weighted using the cosine similarity between the two particle vectors. Next, we implemented the architecture displayed in *Figure 1*. Initially, an edge-agnostic multi-layer-perceptron with non-linear activations creates node representations. These representations are then fed to a series of graph-convolutional networks which create embeddings through aggregation of neighboring node representations. These embeddings are finally interpreted by a final layer of similar MLP blocks to generate a final set of class predictions for a given node.

This approach is highly configurable by design in order to accommodate a wide range of graph structures. Given the

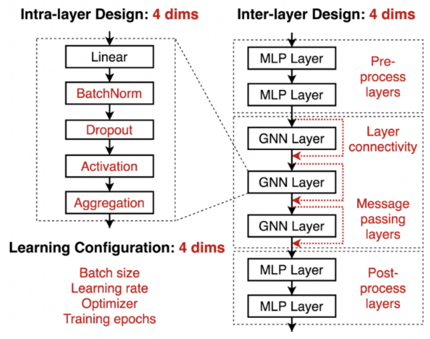


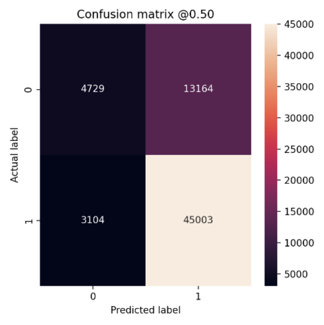
Figure 1. GNN Architecture and Design Space (You et al. 2021)

novelty of this dataset, we will experiment with different configurations within the intra-layer design, the inter-layer designs, and the learning configurations.

4. Experiment

4.1. Embedding for Flow Classification

In order to classify the flow class of a particle, we first implemented a Multilayer Perceptron (MLP) model using Keras. With raw data without scaling or sampling, the accuracy was about 62.5%. To increase the accuracy, we decided to balance and scale our data. Since one of the reasons for the first model was the imbalance in data, we balanced our data so that an equal number of particles represent each flow class as well as the powder class. Additionally, using MinMaxScaler, we scaled and translated each feature individually such that it is in the range between zero and one. Then, we used the MLP model with scaled and balanced data, and although the accuracy improved, the model tended to predict flow class 1 for flow class 0 particles (Figure 2).



```
No Flow Detected (True Negatives):  4729
Flow Incorrectly Detected (False Positives):  13164
No Flow Incorrectly Detected (False Negatives):  3104
Flow Detected (True Positives):  45003
Total Flow Detected:  48107
0.7535151515151515
```

Figure 2. Confusion matrix for MLP model with scaled and balanced data

To achieve a better-performing model, we decided to visualize our data using t-distributed stochastic neighbor embedding (T-SNE), which is a dataset decomposition technique that reduces the dimensions of data. As shown in Figure 3, particles in flow classes 0 and 1 overlap, which made the model difficult to accurately classify its flow class.

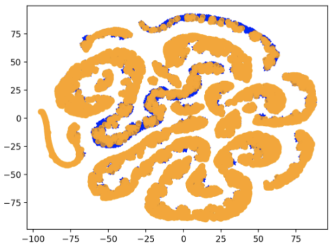


Figure 3. Overlap between two flow classes

To address this issue, we attempted a new approach by creating an autoencoder model that learns the best representation of flow class 0. The latent representation of the autoencoder is shown in Figure 4.

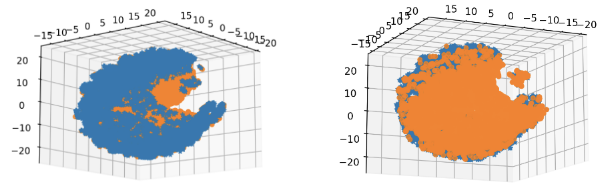


Figure 4. 3D latent representation of particle embeddings

Using this embedding, we performed logistic regression, which improved our accuracy. Finally, we used the MLP Classifier we implemented as our first step with the embedding, and we were able to obtain the best-performing model with hyperparameter tuning using kerastuner.

4.2. Anomaly Detection

Using the same auto-encoder architecture as the previous MLP classification, we trained model using a subset of positive instances. Standard scaling was fit to the training sample and applied to the positive instance validation set and negative instance testing set. Mean squared error (MSE) was used to compute the reconstruction loss across each reconstructed feature. Each particle is reconstructed individually, then grouped by powder class. The average MSE is then computed across the group. This value will serve as an indicator of flow class, with a low reconstruction error indicating a positive class label and a high reconstruction error indicating a negative class label.

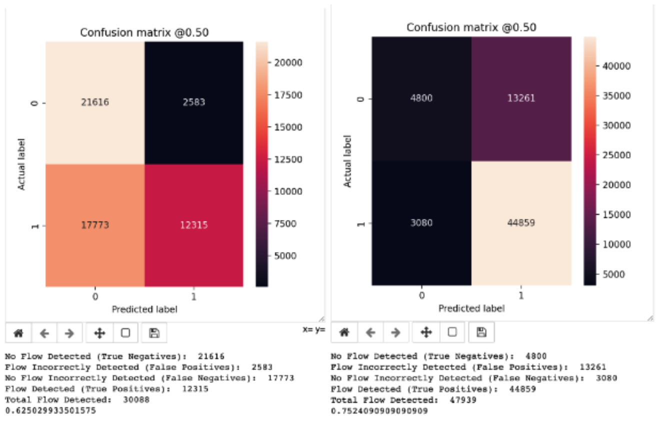
4.3. Graph Convolution

Given the graph encoding described in section 3.2, we performed multiple tests with different configurations. Specifically, we tested 18 total configurations. Each configuration consisted of an aggregation method (sum, mean, or max), a combination method (GRU, concatenate, or add) and the choice to use skip connections or not. To train this model, we used ADAM optimization, sparse categorical cross-entropy loss, and measured spare categorical accuracy for our metric.

5. Results

5.1. Flow Class Classification

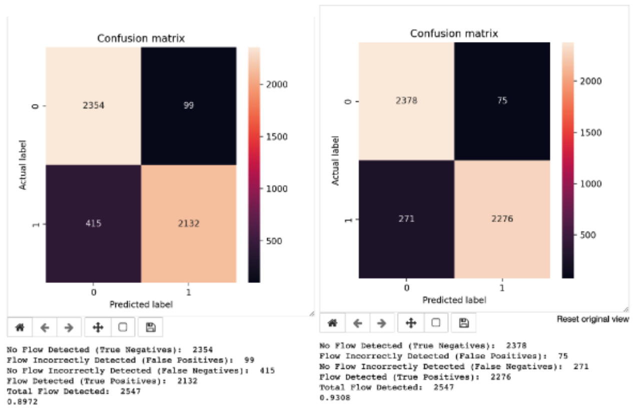
Each model's architecture and performance are shown in Table 1.



```
No Flow Detected (True Negatives):  21616
Flow Incorrectly Detected (False Positives):  2583
No Flow Incorrectly Detected (False Negatives):  17773
Flow Detected (True Positives):  12315
Total Flow Detected:  30088
0.625029933501575
```

```
No Flow Detected (True Negatives):  4800
Flow Incorrectly Detected (False Positives):  13261
No Flow Incorrectly Detected (False Negatives):  3080
Flow Detected (True Positives):  44859
Total Flow Detected:  47939
0.7524090909090909
```

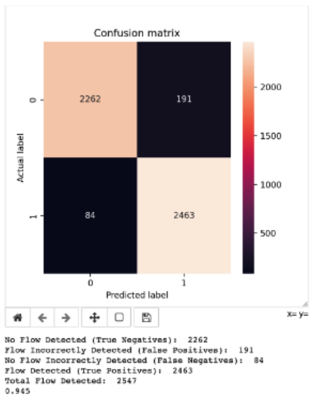
Figure 5. Confusion matrix of MLP (left) and MLP with scaled and balanced data (right)



```
No Flow Detected (True Negatives):  2354
Flow Incorrectly Detected (False Positives):  99
No Flow Incorrectly Detected (False Negatives):  415
Flow Detected (True Positives):  2132
Total Flow Detected:  2547
0.8972
```

```
No Flow Detected (True Negatives):  2378
Flow Incorrectly Detected (False Positives):  75
No Flow Incorrectly Detected (False Negatives):  271
Flow Detected (True Positives):  2276
Total Flow Detected:  2547
0.9308
```

Figure 6. Confusion matrix of Logistic Regression with embedding (left) and MLP with embedding (right)



```
No Flow Detected (True Negatives):  2262
Flow Incorrectly Detected (False Positives):  191
No Flow Incorrectly Detected (False Negatives):  84
Flow Detected (True Positives):  2463
Total Flow Detected:  2547
0.945
```

Figure 7. Confusion matrix of MLP with embedding + hyperparameter tuning

Table 1. Flow Class Classification Models

Model	**Accuracy(%)**
MLP	62.5 (Figure 6)
MLP with scaled and balanced data	75.3 (Figure 6)
Logistic Regression with embedding	89.7 (Figure 7)
MLP with embedding	93.1 (Figure 7)
MLP with embedding + hyperparameter tuning	94.5 (Figure 8)

5.2. Anomaly Detection Classification

The results of the auto-encoder anomaly detection classification are show in Figure 8. With a reconstruction error cutoff of 0.2, this model has a precision of 93.3%, a recall of 87.5%, and an accuracy of 88.0%.

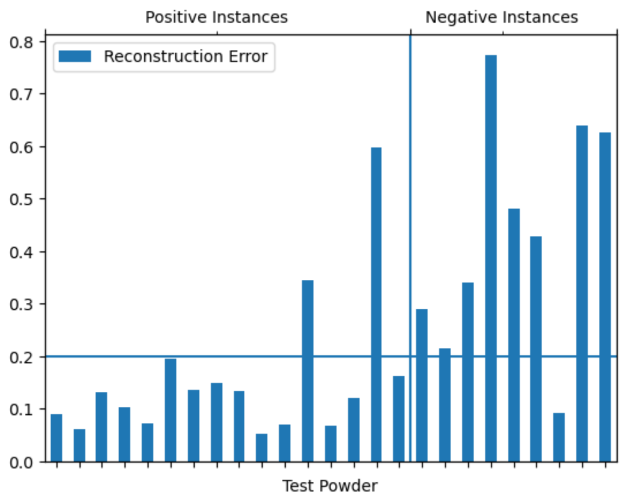


Figure 8. Average reconstruction error of powders (y ¡ 0.2)

5.3. Graph Convolution Classification

The results of the Graph CNN configuration experiments are documented in Table 2. Across these configurations, mean aggregation with concatenation and skip connections achieves the highest accuracy.

Table 2. Model Configuration Comparisons for Graph-CNN Flow Rate Classification

Aggregation	**Combination**	**Test Accuracy(%) with/without Skip Connection**
Sum	GRU	69.3/68.9
Sum	Concat	71.1/70.0
Sum	Add	66.5/66.9
Mean	GRU	68.2/64.6
Mean	**Concat**	**74.5**/72.8
Mean	Add	62.5/62.3
Max	GRU	68.3/68.0
Max	Concat	73.4/72.5
Max	Add	64.4/64.2

6. Discussion

While previous research showed that the mean and standard deviation for each particle feature was useful enough in predicting flow class, this approach fails to appropriately describe particle mixtures that have multimodal distributions.

The Graph Neural Network approaches did not immediately yield useful results across the entire dataset, though the use of regularization through dropout and smoothing through skip connections did improve the model somewhat. Ultimately, the largest setback was the limitation on training time due to the costs of generating a graph with a large number of nodes and edges. The utility of a graph was highly dependent on the random initialization as many samplings failed to adequately differentiate between classes.

7. Conclusions and Future Work

The use of autoencoders for powder mixture embedding could be enhanced by using a variational auto-encoder with a Gaussian mixture as a prior distribution instead of a standard normal which leads to better clustering outcomes, especially for multimodal data. There was also an issue of overfitting in our experiment due to the use of a small sample size to address the computational complexity.

Regarding the graph neural network approach, improvements can be made in two aspects; Firstly, the graphbuilding stage, using locality-sensitive hashing achieves drastically faster linear computation of distances between particles when compared to cosine similarity. Secondly, graph pruning to reduce the number of redundant elements, either by applying the lottery ticket hypothesis to identify useful subnetworks within the GNN, or by applying Kruskal's Minimum Spanning Tree algorithm to prune the graph itself.

References

Sakurada, Mayu and Yairi, Takehisa. Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*, MLSDA'14, pp. 4–11, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450331593. doi: 10.1145/2689746.2689747. URL https://doi.org/10.1145/2689746.2689747.

You, Jiaxuan, Ying, Rex, and Leskovec, Jure. Design space for graph neural networks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

(Sakurada & Yairi, 2014) (You et al., 2020)