Machine Learning for Beam Profile Analysis

Leo Brockhuis, AGH University of Science and Technology in Cracow, Poland

September 4, 2019

Abstract

The free-electron laser in Hamburg (FLASH) operating in the soft x-ray to extreme ultra-violet regime allows for unprecedented experiments in materials science, biotechnology and other fields. Running such a facility produces large amounts of data that may hold important information about the underlying physics or the inner workings of the machine itself. Unfortunately the sheer amount of data is too large for classical analysis methods. Here another way is shown, analysis using machine learning. This approach gained popularity because of its usefulness in the consumer market, where large amounts of data are acquired on a scale never seen before. This paper shows that research facilities can also benefit from machine learning, particularly the need for the interpretation of large amounts of data is addressed. Uses in beam profile analysis from the FLASH facility are shown. A convolutional neural network is introduced to label beam-profiles and an autoencoder + principal component analysis algorithm is used to aid in data analysis using dimensionality reduction. This work is intended for researchers and engineers working at facilities where large and difficult to interpret data is acquired. No previous knowledge of machine learning is assumed. Python with Tensorflow were used, the code is available at `https://github.com/Korfeusz/DESY_photon_pulse_property_predictor`.

Contents

1 Introduction **3**
1.1 Convolutional neural networks for image classification 3
1.2 Convolutional autoencoders and principal component analysis for dimensionality reduction 4

2 Dataset **5**

3 Application of machine learning to the experimental results **6**
3.1 0- and higher- order mode beam profile labeling 6
3.1.1 Automatic label creation 6
3.1.2 Convolutional Neural Network based labeling 7
3.1.3 Results 7
3.2 Dimensionality reduction for data visualization and analysis 7
3.2.1 Results 12

4 Final remarks and future work **13**

1 Introduction

This work is concerned with exploring the possibilities of using machine learning for data analysis at the FLASH facility at Desy in Hamburg. But the tackled problems extend beyond this specific use and can be useful for any research center that produces large, multidimensional datasets. This report hopes to show that machine learning can be helpful as a tool for data interpretation.
The Free-electron LASer in Hamburg (FLASH) can provide users with intense ultra-short pulses in the extreme ultra-violet to soft x-ray regime. The combination of brilliance, coherence and short pulse length allows this facility to be used in exciting experiments in materials science, biotechnology and other areas of research [1]. Such a facility provides users with large amounts of data for each experiment, but owing to the difficulties in analyzing such datasets most of it is not used. Similar problems have already been discussed for other large-scale research facilities and the scale of the acquired information seems only to increase [2] . Therefore this is an area uniquely suited for machine learning approaches that thrive in vast multi-dimensional difficult to interpret datasets [3].
One can think of machine learning as the process of gaining useful information from the datasets themselves. This means a statistical information inference method where little to no prior information is needed regarding the structure of the data. Machine learning methods are often based on finding relationships between some set of inputs (called features) and an output [4]. Machine learning can be divided into two groups: supervised and unsupervised learning [5]. The first one learns by updating the internal machine learning model by evaluating the loss between the input and a known desired output. The second group consists of methods that create models of the data without any explicit information of what the correct answer may be. An example of the first type of algorithm is a convolutional neural network used for image classification. An example of the second type is a k-nearest neighbors clustering algorithm [6]. There is also a gray area between them, for example an autoencoder algorithm can be seen as a quasi-unsupervised algorithm. A description of classifying convolutional neural networks and compression autoencoders is provided here as both have been used in this work. The main problems that can be solved by machine learning, that are useful here, are classification and dimensionality reduction.

1.1 Convolutional neural networks for image classification

Let us imagine we have a dataset where for each item there exists a known corresponding label, and the set of all possible labels is finite. Now the classification problem is predicting the label of yet unseen examples. One way of solving this is using a multilayer perceptron network. In such an algorithm the input is forward propagated through layers of non-linear transformations with many parameters, called weights, that can be tuned to alter the output of the last layer. After this the output is compared to the known correct answer, the loss is evaluated using a loss function and the weights are updated using a gradient descent derived algorithm. This is used to teach the model in a supervised way, to use this model we save the structure and weights and apply the same

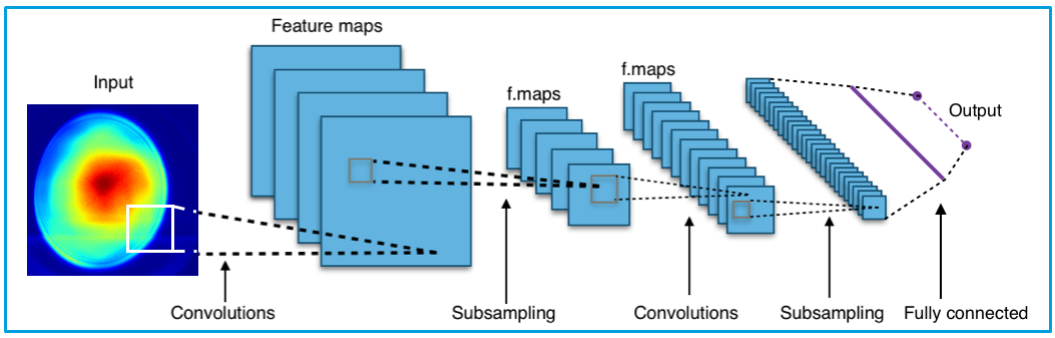


Figure 1: Diagram of the structure of a convolutional neural network. The convolutions create a stack of feature maps, also called channels. Sub-sampling reduces the dimensionality of each map between layers condensing the information. (Image adapted from [7])

feed-forward transformations to a new example obtaining the label as the output of the network. This algorithm works well, but loses information about the initial structure of the example, as it accepts only input in the form of a vector. For images there exists another, similar algorithm that uses 2D convolutions in the feed-forward pass to infer useful information from the image. Each convolutional layer can consist of a stack of 2D convolutions, a dimensionality reduction method (sub-sampling) and a nonlinear output function. The weights to be updated are the weights in the convolutional kernels. Figure 1 schematically shows the feed forward structure of a convolutional neural network. More can be learned about convolutional neural networks (or CNNs) in [9].

1.2 Convolutional autoencoders and principal component analysis for dimensionality reduction

If our each example in our dataset has some number of dimensions and we want to reduce the dimensionality while loosing as little information as possible then we have posed a dimensionality reduction problem. This can be used to lower the stored size of the data (compression) or for data analysis in lower, easier to interpret, dimensions. The autoencoder can be divided into the encoder and decoder. The encoder is a neural network that lowers the dimensionality of the input, then the decoder tries to recreate the input. A loss evaluation function calculates some distance measure between the input and the autoencoder output and updates the encoder and decoder weights accordingly. Data in the bottleneck is a lossy compressed representation of the input. The final dimensionality reduction is done using a principal component analysis algorithm (PCA). This algorithm transforms the input data into a new set of uncorrelated variables ordered in such a way that the first ones account for most of the variance. This transformation can be thought of as fitting a mutlidimensional ellipsoid to mean centered data and ordering the ellipsoid axes by their size. Taking the largest axes makes sure that most of the variance is accounted for [8]. This procedure ensures orthogonality of the output features providing an orthogonal low dimensional output space. Figure 2 shows this

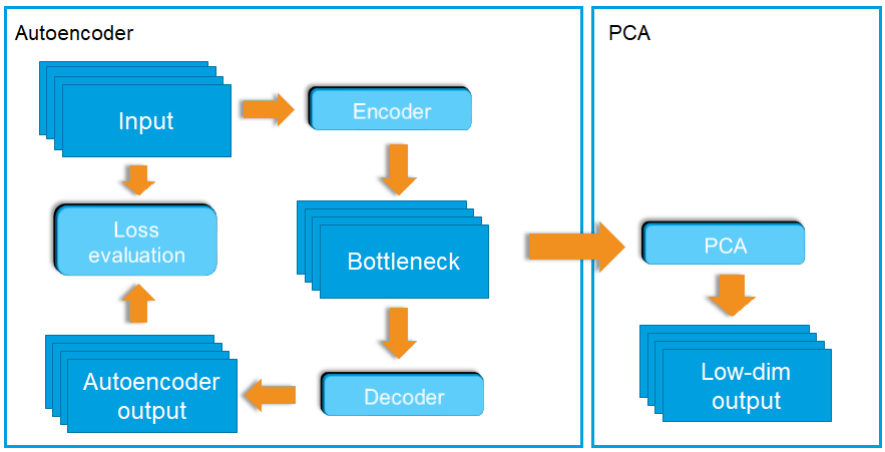


Figure 2: Diagram of an autoencoder and principal component analysis dimensionality reduction algorithm.

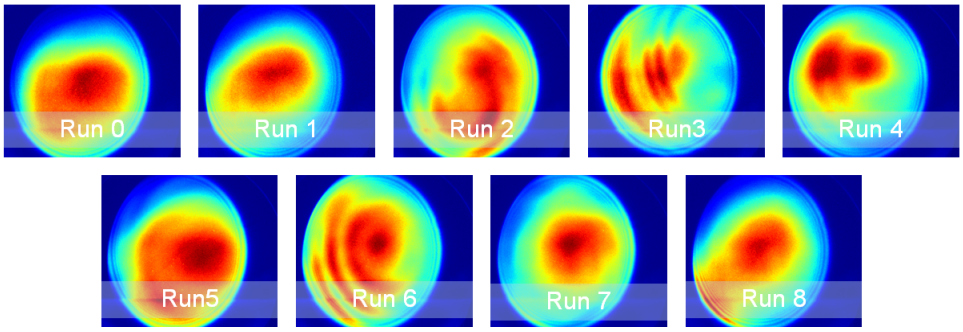


Figure 3: Example input beam profiles for each run.

relation. More can be once again learned using [9].

2 Dataset

The data were 72000 beam profiles acquired at the FLASH2 facility. There were nine runs where the undulator number and the electron bunch charge were changed. The wavelength was 21.4 nm for all runs. The runs and some of their parameters are in table 1. other parameters were saved during the experiment and stored in .h5 files format available under the directory /asap3/flash/gpfs/fl21/2019/data/11007852/ internally at DESY. The original images are cropped to 259×259, but not downsized. Example images can be seen in figure 3. Some beam profiles contained noise or were empty and therefore had to be removed form the used set, this reduced the number of examples to 64129.

Table 1: Information on the experimental runs used to obtain the data

Run number	Number of undulators	e-bunch charge [nC]	Tapering	Energy [μJ]	comment
0	9	0.3	Yes	131	
1	7	0.3	No	100	
2	6	0.3	Yes	90	
3	6	0.4	No	46	
4	7	0.4	Yes	202	
5	9	0.4	Yes	238	Donut intensity profile
6	9	0.5	Yes	88	
7	7	0.5	Yes	34	
8	6	0.5	Yes	22	

3 Application of machine learning to the experimental results

This section will be divided into the classification problem of the profiles as 0-order or higher order mode. And the dimensionality reduction problem for data analysis.

3.1 0- and higher- order mode beam profile labeling

The first part of this work concerned itself with the determination of the order of the mode of the beam profiles. As there were over 64129 non-corrupted images their manual labeling would be tedious, therefore a different approach was taken. First some measure of 0-order vs higher-order mode was created for automatic label creation. Later the automatically created labels were inserted into a convolutional neural network, so that a more robust measure of mode order may be abstracted from the beam profiles and their labels.

3.1.1 Automatic label creation

Two independent algorithms were created for label creation. Both based on the fact that a symmetric 0-order mode profile will be circular in appearance. One measures the perimeter squared over area of the profile, noticing that for a circle this should be 4π and so by subtracting this constant and taking the absolute value of the result a distance measure is created. This process is repeated at different binarization levels and a mean value of the distances is taken. The second method is based on the fact that a centered ring-shaped section of a circle has a fixed ratio of area inside the ring to the rest of the image. These two measures have been applied to all images and a threshold was applied to both of them in order to get a mode order label. The threshold was set experimentally

through the inspection of profiles and their mode order measures. Finally a logical OR was applied to both measures to obtain the label that will be used to train the CNN. It is worth mentioning that the reason why a two-dimensional Gaussian was not simply fit to the profiles and a goodness-of-fit measure used for label creation is that fitting a two-dimensional function to such an amount of images was computationally expensive and therefore time consuming. In total 2302 images were labeled as 0-order mode.

3.1.2 Convolutional Neural Network based labeling

Using convolutional neural networks in order to classify data as 0-order or higher order modes has been done before [10]. In order to get a more robust measure of 0-order vs higher order modes a convolutional neural network was trained using the automatically created labels. If all images would have been taken to train the model the high bias towards higher-order mode profiles would induce a bias in the model or even make it a viable strategy to have a constant output of higher-order mode for all inputs. Therefore a subset of all the uncorrupted images was taken so that the 2302 profiles labeled 0-order mode made up 20% of this subset. This subset was then split $80 - 20$ into a train and test set for the CNN. In total the train set had 8000 profiles and the test set 2000, both having 20% of profiles labeled as 0-order mode. The profiles were then downsized to 32×32 in order to make the learning faster. This size was chosen so that the the main features and shape of the profiles are still visible. The architecture of the CNN was two convolutional layers, one hidden dense layer and an output layer. The first convolutional layer had a kernel size of 3 with 12 output channels, a ReLU output function and a dropout of 0.5 was used. The second layer also had a dropout of 0.5 and the same output function, but had 24 output channels and a kernel size of 6. The fully-connected layer had 10 neurons and a ReLU output activation with a dropout of 0.5. The output was fed through a softmax activation. The high dropout rates are used to not over fit the data, as the goal is to teach the network the main features of a 0-order mode profile. The layer structure is in table 2.

3.1.3 Results

Example output images from the test set can be seen in figure 4. The accuracies and loss evaluations for the training and test set are in figure 5. These images confirm that the algorithm converged. Confusion tables for train and test data are in tables 3 and 4. These show the accuracy on the train and test set, but also the composition of false positives and false negatives. What can be seen in table 4 is that the number of false positives is similar to the number of false negatives, which may mean that there is no bias towards labeling as 0-order or higher-order mode.

3.2 Dimensionality reduction for data visualization and analysis

In order to be able to interpret the profiles in terms of FLASH2 machine settings and the created 0- vs higher-order mode measures a dimensionality reduction scheme was

Table 2: Convolutional neural network structure. The Param number column shows the number of trainable parameters for each layer. The None in the output shape depends on the number of images used during one forward pass. The last number in the output shape is the number of channels in that layer. A detailed description of the layer types can be found in the Tensorflow documentation: [12]

Layer (type)	Output Shape	Param #
reshape (Reshape)	(None, 33, 33, 1)	0
conv2d (Conv2D)	(None, 33, 33, 12)	108
batch_normalization	(None, 33, 33, 12)	36
activation (Activation)	(None, 33, 33, 12)	0
dropout (Dropout)	(None, 33, 33, 12)	0
conv2d_1 (Conv2D)	(None, 17, 17, 24)	10368
batch_normalization_1	(None, 17, 17, 24)	72
activation_1 (Activation)	(None, 17, 17, 24)	0
dropout_1 (Dropout)	(None, 17, 17, 24)	0
flatten (Flatten)	(None, 6936)	0
dense (Dense)	(None, 10)	69360
batch_normalization_2	(None, 10)	30
activation_2 (Activation)	(None, 10)	0
dropout_2 (Dropout)	(None, 10)	0
dense_1 (Dense)	(None, 2)	22

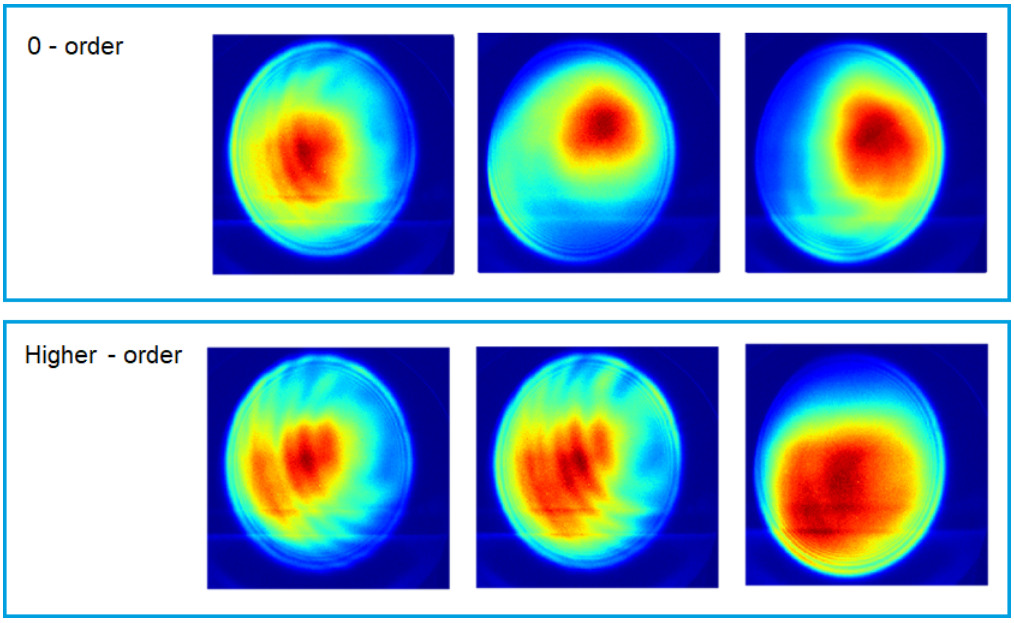
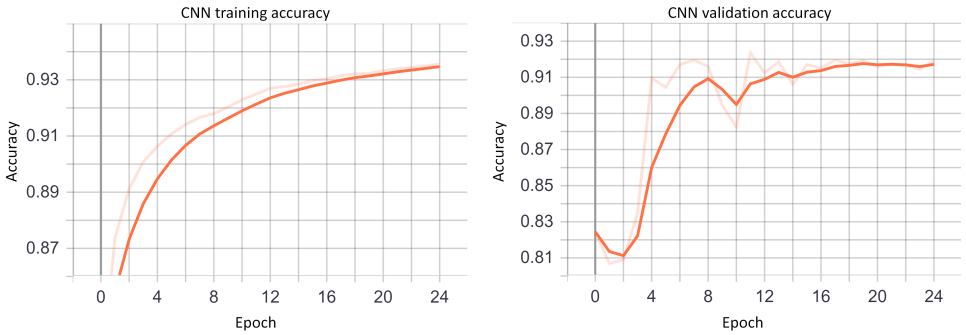
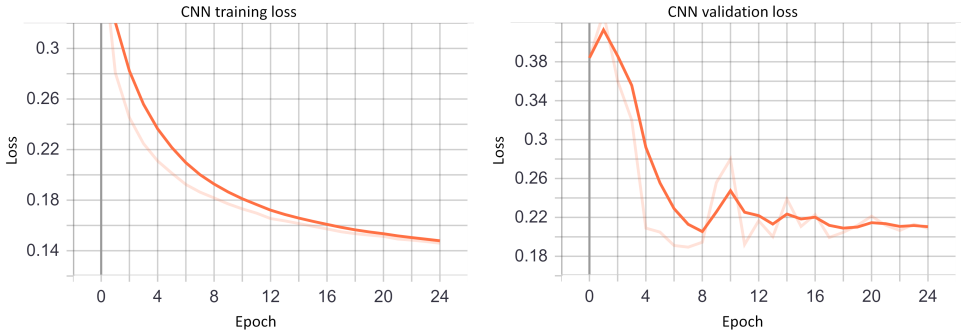


Figure 4: Examples of profiles labeled as 0-order and higher order mode by the CNN algorithm.



(a) Accuracy on the train set over training epoch number.

(b) Accuracy on the test set over training epoch number.



(c) Loss evluation on the train set over training epoch number.

(d) Loss evluation on the train set over training epoch number.

Figure 5: Accuracies and loss evaluations during training.

Table 3: Confusion matrix for the training data. The numbers represent the amount of images in this set that have been labeled correctly or incorrectly, thus showing the number of false positives, false negatives, true positives and true negatives.

Training data		Predicted	
Accuracy: 94.8%		0-order	Higher-order
Correct	0-order	1464	136
	Higher-order	283	6117

Table 4: Confusion matrix for the test data. The numbers represent the amount of images in this set that have been labeled correctly or incorrectly, thus showing the number of false positives, false negatives, true positives and true negatives.

Test data		Predicted	
Accuracy: 92.1%		0-order	Higher-order
Correct	0-order	331	69
	Higher-order	90	1510

created. First a convolutional autoencoder network was used to initially reduce the dimensionality. Later a PCA algorithm was used to further reduce the dimensions to two. PCA was chosen as the final step to ensure orthogonality of the resultant dimensions. In this way a two dimensional space was created in which each image can be represented as a single point. The pipeline of this algorithm can be seen in figure 6. The layer structure of the autoencoder is shown in table 5. The smallest data dimensionality is $8 \times 8 \times 4$, this is the bottleneck and from there the lower dimensional representations are taken to the PCA algorithm. The input and output layer sizes are the same, because the input and correct label are the same image.

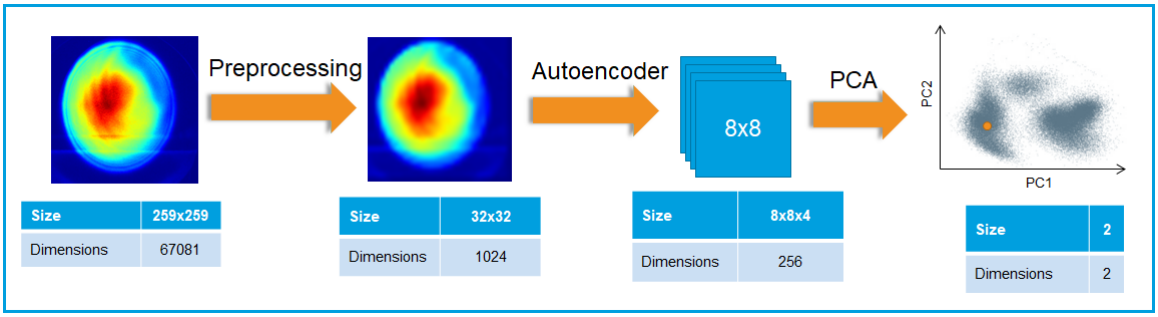
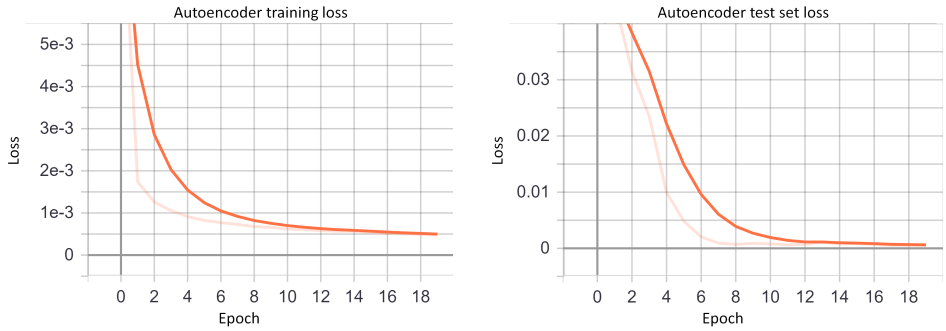


Figure 6: The autoencoder and PCA algorithm diagram showing the dimensions at various steps.

Table 5: Autoencoder structure. The Param number column shows the number of trainable parameters for each layer. The None in the output shape depends on the number of images used during one forward pass. The last number in the output shape is the number of channels in that layer. A detailed description of the layer types can be found in the Tensorflow documentation: [12].

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 32, 32, 1)]	0
conv2d (Conv2D)	(None, 32, 32, 16)	160
max_pooling2d (MaxPooling2D)	(None, 16, 16, 16)	0
batch_normalization	(None, 16, 16, 16)	64
conv2d_1 (Conv2D)	(None, 16, 16, 4)	580
max_pooling2d_1	(None, 8, 8, 4)	0
batch_normalization_1	(None, 8, 8, 4)	16
conv2d_2 (Conv2D)	(None, 8, 8, 4)	148
up_sampling2d (UpSampling2D)	(None, 16, 16, 4)	0
batch_normalization_2	(None, 16, 16, 4)	16
conv2d_3 (Conv2D)	(None, 16, 16, 16)	592
up_sampling2d_1	(None, 32, 32, 16)	0
batch_normalization_3	(None, 32, 32, 16)	64
conv2d_4 (Conv2D)	(None, 32, 32, 1)	145



(a) Loss evluation on the train set over training epoch number.

(b) Loss evluation on the train set over training epoch number.

Figure 7: Loss evaluations during training.

3.2.1 Results

The loss evaualions over epoch number is presented in figure 7, the values converge during training. The PCA output space itself can be seen in figure 8. Here the usefulness of this method comes from adding a third dimension to this output space in order to be able to interpret the data in terms of various other parameters. The PCA output space with with points color coded by different parameters can be seen in figure 9. Several conclusions can be made from the principal component analysis plots shown in figure 9. Looking at plot 9a containing the PCA output space color coded by the run numbers we can notice that images contained in a single run cluster together implying similarity between examples from a single run. On the same plot it can also be seen that three distinct clusters form, one containing runs $0, 2$ and 5, one containing just run number 7 and one containing the rest of the runs. This implies an inherent similarity between beam profiles from within one cluster when we take the distance in the principal components space to be a measure of similarity due to some unknown features. This features can be found by analyzing the PCA output space color coded by different parameters and looking for relations between the clustering and the distribution of those features in the principal component space. Another thing to note is the bunching of 0-order mode labeled images in plot 9b. This shows, that there is a relationship between the found principal components and the predicted label. This information is especially useful for machine operation, as it may help find the settings that provide a higher likelihood of producing 0-order mode beam profiles. Finally from plot 9c where profile intensities are shown in the PCA space we can see that a structure formed where the intensities fall from left to right in each of the discussed clusters. There is more interpretation that can be done here and also more data that could be chosen as the color axis to aid in the understanding of the underlying physics. Here just some examples of what could be inferred were given.

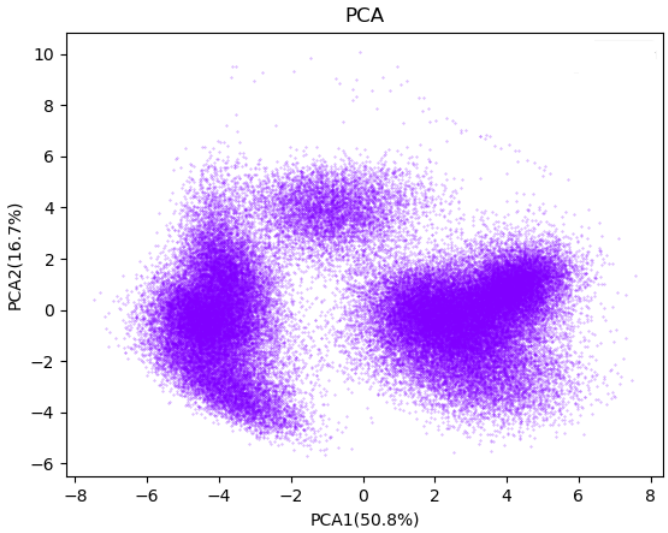
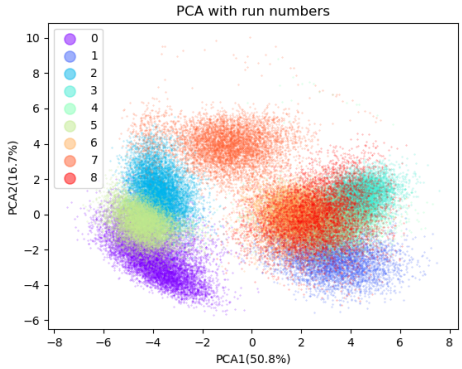


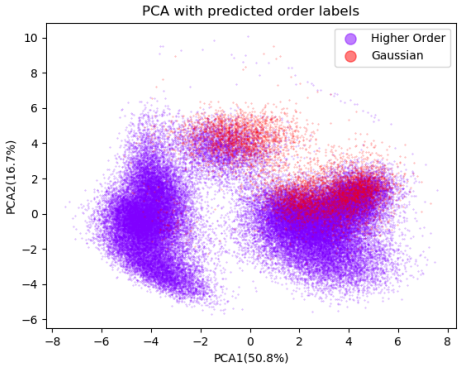
Figure 8: The PCA output space. The percentages in parentheses on the axes are the explained variances by each axis.

4 Final remarks and future work

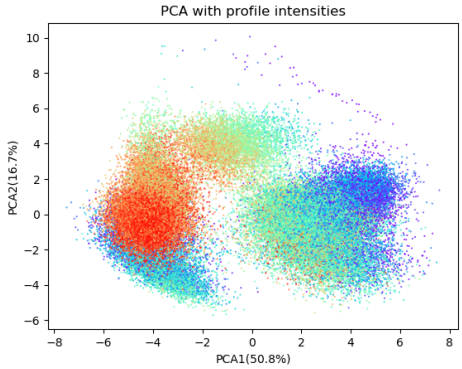
This work should be considered preliminary. It was argued that using machine learning to gain insight into large datasets is a viable strategy. Also because the FLASH facility produces large amounts of multidimensional data it is well suited for the use of machine learning. Information gained through ML can be used both for machine control and for getting a better understanding of the underlying physics. Further insights into the relations between the FLASH2 beam line settings and the resultant beam profiles can be researched and may be of importance for further operation and for the beam quality that can be provided to the user. An example of this would be the prediction of pulse properties from machine settings and easy-to-do diagnostics [11]. Another direction of further development could be a split of the higher-order modes into more specific free electron laser modes.



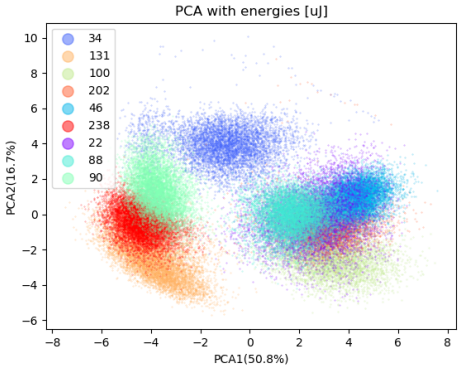
(a) Run numbers, see table 1



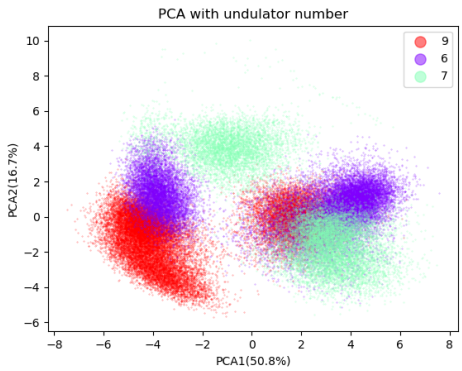
(b) Predicted 0-order vs higher-order mode label



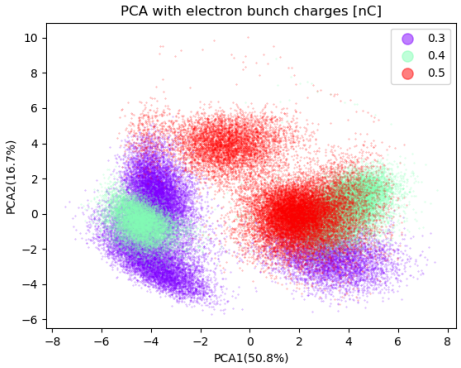
(c) Profile intensities



(d) Total run energies, see table 1



(e) Undulator number, see table 1



(f) Electron bunch charge, see table 1

Figure 9: PCA plots with various parameters used as the color dimension.

References

[1] The soft x-ray free-electron lases FLASH at DESY: beamlines, diagnostics and end-stations, *K. Tiedtke et al.*, New Journal of Physics 11 (2009) 023029

[2] BigData and computing challenges in high energy and nuclear physics, *A. Klimentov et al.*, Journal of Instrumentation 12 C06044, (2017)

[3] Machine Learning Algorithms in Big data Analytics, *K. Sree Divya et al.*, International Journal of Computer Sciences and Engineering 6(1):63-70 (2018)

[4] Introduction to Machine Learning, *A. Smola et al.*, Cambridge University Press (2008)

[5] Understanding Machine Learning: From Theory to Algorithms, *S. Shalev-Shwartz and S. Ben-David*, Cambridge University press (2014)

[6] Nearest neighbor pattern classification. *Thomas M Cover and Peter E Hart.* IEEE Transactions on Information Theory, 13(1):2127, 1967.

[7] A typical convolutional neural network (image) `commons.wikimedia.org/wiki/File:Typical_cnn.png`

[8] Principal Component Analysis, Second Edition, *I.T. Jolliffe*, Springer (2002)

[9] Deep Learning, *I. Goodfellow et al.*, MIT Press (2016),

[10] Seeking Higher-Order Modes in Free Electron Lasers, *P. Sun, Y. Sun*

[11] Accurate prediction of X-ray pulse properties from a free-electron laser using machine learning, *A. Sanchez-Gonzales et al.*, Nature communications 8:15461 (2017)

[12] Tensorflow documentation, (accessed 09.2019), `https://www.tensorflow.org/api_docs`