






Deep feature learning of in-cylinder flow fields to analyze cycle-to-cycle variations in an SI engine

International J of Engine Research
2021, Vol. 22(11) 3263–3285
© IMechE 2020
Article reuse guidelines:
sagepub.com/journals-permissions
DOI: 10.1177/1468087420974148
journals.sagepub.com/home/jer


Daniel Dreher¹, Marius Schmidt², Cooper Welch², Sara Ourza^{1,2},
Samuel Zündorf¹, Johannes Maucher³, Steven Peters¹,
Andreas Dreizler², Benjamin Böhm² and Alexander Hanuschkin¹

Abstract

Machine learning (ML) models based on a large data set of in-cylinder flow fields of an IC engine obtained by high-speed particle image velocimetry allow the identification of relevant flow structures underlying cycle-to-cycle variations of engine performance. To this end, deep feature learning is employed to train ML models that predict cycles of high and low in-cylinder maximum pressure. Deep convolutional autoencoders are self-supervised-trained to encode flow field features in low dimensional latent space. Without the limitations ascribable to manual feature engineering, ML models based on these learned features are able to classify high energy cycles already from the flow field during late intake and the compression stroke as early as 290 crank angle degrees before top dead center (-290°CA) with a mean accuracy above chance level. The prediction accuracy from -290°CA to -10°CA is comparable to baseline ML approaches utilizing an extensive set of engineered features. Relevant flow structures in the compression stroke are revealed by feature analysis of ML models and are interpreted using conditional averaged flow quantities. This analysis unveils the importance of the horizontal velocity component of in-cylinder flows in predicting engine performance. Combining deep learning and conventional flow analysis techniques promises to be a powerful tool for ultimately revealing high-level flow features relevant to the prediction of cycle-to-cycle variations and further engine optimization.

Keywords

Deep learning, machine learning, feature analysis, particle image velocimetry, in-cylinder flow, cycle-to-cycle variations, IC engine

Date received: 15 August 2020; accepted: 21 October 2020

Introduction

Cycle-to-cycle variations (CCVs) decrease the performance of spark-ignition (SI) internal combustion (IC) engines and increase their emissions. Rigorous experimental research is required to provide detailed data for the development of accurate models of engine phenomena and the reduction of CCVs.¹ Over the last decades, advancements in laser and imaging technology have enabled IC engine research to expand to high-speed, crank angle-resolved^{2–4} and high-resolution boundary layer velocity and fuel film measurements^{5–8} as well as multi-parameter measurements to examine the propagation of the early flame kernel.⁹ With the increase in technology, also comes the ability of generating better statistics for flow and flame data obtained from advanced laser diagnostics in IC engines and the subsequent use of machine learning (ML)-based processing

and analysis techniques to better understand the phenomena involved in engine CCV.

Variations of in-cylinder flows have a profound effect on CCVs and have been investigated with conditional statistics,^{10,11} proper orthogonal decomposition,^{12–14} and analysis of engineered features.^{11,15,16} With larger experimental and simulated flow field data sets available, ML has become a promising approach to identify relevant non-linear characteristics of the

¹Group Research, Mercedes-Benz AG, Stuttgart, Germany

²Department of Mechanical Engineering, Reactive Flows and Diagnostics, Technical University of Darmstadt, Darmstadt, Germany

³Hochschule der Medien, Stuttgart, Germany

Corresponding author:

Alexander Hanuschkin, Group Research, Mercedes-Benz AG, HPC H515, Sindelfingen 71059, Germany.

Email: alexander.hanuschkin@daimler.com

turbulent flow responsible for CCVs,^{17–20} which will ultimately enable engine design optimization.

Deep learning (DL), a sub-field of ML, allows automatic feature learning and facilitates models with unmatched predictive power,²¹ but requires large data sets for training complex models. Feature learning, sometimes referred to as representation learning, describes the process of extracting discriminative features of given input data into a high-level representation, that is, compressing important features of a data set into dense encodings.²² Examples of traditional, shallow feature learning approaches include Pearson's principal component analysis or Fisher's linear discriminant analysis.²³ For deep feature learning, the so-called pretext task can be separated from the final (downstream) task. Training models on the pretext task can be self-supervised, that is, without dependence on labeled training data, and enables them to extract general purpose representations of the training data, for example, abstract hierarchical representations or feature space representations with separate feature clusters. A good representation of such data is valuable even for unrelated downstream tasks.²² Alternatively, deep neural networks (DNNs) can also be trained end-to-end, that is, the features are learned while the network is trained to solve the final task, but a large, labeled, and task-specific data set is required.

Autoencoders (AEs) are one way of realizing self-supervised feature learning. Under-complete or regularized AEs are trained to encode a compressed representation of the input in latent space and decode the information to match the input.²⁴ Convolutional autoencoders (CAEs) are DNNs with convolutional layers, reducing the number of weights to be trained compared with fully connected layers. Tschannen et al.²⁵ showed that AEs can generate good general purpose feature representations. However, AEs might not necessarily store salient features in latent space encodings, but in the decoder. Even if salient features are encoded in latent space, it is not given that these features are stored in a format favored by the downstream task. CAEs have been used before to investigate combustion instabilities in gas turbine engines,²⁶ but so far, they have not been utilized for the prediction of cyclic variability in IC engines.

In previous works by the authors, engineered feature sets were used to successfully predict engine performance based on in-cylinder flow field data obtained from particle image velocimetry (PIV) measurements¹⁹ and based on the 2D cross-section of the early flame kernel using the extracted flame contour from a PIV setup.²⁷ However, since finding good feature representations is a critical step in CCV analysis to reduce the dimension of the input space,²⁸ it was hypothesized that with a sufficiently large data set and by using DL methods to learn features, the prediction accuracy may be increased. To this end, a large experimental data set of in-cylinder flow fields obtained from PIV in an optical SI engine is used to first test ML models with

engineered features as a baseline to predict cycles of high (HC) and low (LC) maximum in-cylinder pressure (P_{\max}) values using binary classification. Then deep feature learning using CAEs is employed and the learned features are tested against the baseline.

Methods

PIV measurements and flame extraction

A detailed description of the experimental setup is given in Hanuschkin et al.²⁷ and only necessary details are summarized in the following. Experiments are conducted in the optical research engine at TU Darmstadt.^{2,29} The single-cylinder SI engine is employed using a spray-guided cylinder head configuration with a compression ratio of 8.7:1. At an average intake pressure of 0.95 bar, engine speed of 1500 rpm, and an average intake temperature of 301 K, a stoichiometric port fuel injected mixture (iso-octane) is ignited with a spark timing of -22.2°CA (22.2 crank angle degrees (CADs) before compression top dead center). A piezoelectric transducer (GU22C, AVL) measures the in-cylinder pressure. The exhaust temperature T is measured with a sampling frequency of 10 Hz.

Through optical access granted via a fused-silica cylinder liner and flat piston window, a PIV experiment is realized in the mid-cylinder plane for in-cylinder flow measurements and flame visualizations. A pair of laser sheets from frequency-doubled Nd:YAG laser cavities with variable time separations ($dt = 4.8\text{--}21\ \mu\text{s}$) are optimized on a CAD basis and used to illuminate silicone oil droplets (Dow Corning DOWSIL 510 Fluid, $\sim 0.5\ \mu\text{m}$) issued well upstream of the intake valves near the port fuel injector. A Phantom v1610 (1280×800 pixel, 12-bit) high-speed CMOS camera equipped with a 532 nm band-pass filter to suppress the flame luminosity is used to record image pairs for flow field calculations. For the flame visualizations, one image from each pair of the sample set is considered and the lack of silicone oil droplets is used to define the cross-sectional burned gas structure. A schematic of the optical setup and an example of such an extracted flame contour can be found in Figure 1 of Hanuschkin et al.²⁷ Likewise, a detailed description of the early flame kernel characterization is subsequently provided.

Flow fields are calculated using the commercial software DaVis 8.4.0 (LaVision). First, image preprocessing is conducted by means of a sliding Gaussian background subtraction (8×8 pixel) and a particle intensity normalization (5×5 pixel) to improve the particle contrast. Then, a unique geometric mask for each CAD is manually generated to eliminate the intake valves and piston from view. Additionally, to yield as close to the same vector grid for each cycle and experimental run as possible, all vectors with their position outside of the mask are eliminated and only pixels inside the mask are considered. For the calculation of

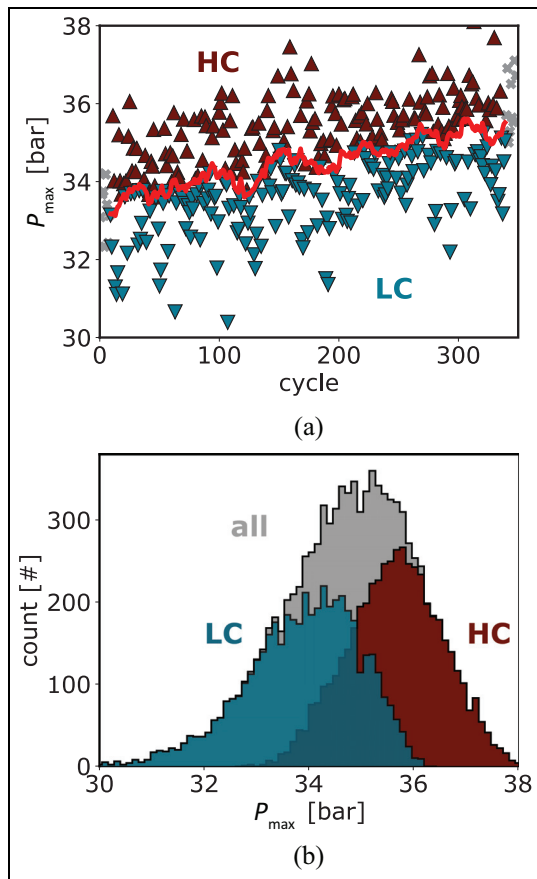


Figure 1. Cycle-to-cycle variations: (a) recorded P_{\max} values during a single experimental run. The recorded P_{\max} values shift due to the thermal engine load and subsequently shift the decision boundary $P_{\max,m.a.}$ (red line). HC (deep red, upward triangles) and LC (petrol, downward triangles) are cycles with P_{\max} above and below $P_{\max,m.a.}$, respectively. (b) histogram of all (gray), HC (deep red), and LC (petrol) P_{\max} values.

vectors, a multi-pass cross-correlation with decreasing interrogation window size (2 initial passes with 64×64 pixel, no weighting function, and 50% overlap; 2 final passes with 32×32 pixel, adaptive Gaussian weighting function, and 75% overlap) is used. Finally, each iteration in the multi-pass cross-correlation as well as the resulting flow field underwent post-processing consisting of a peak ratio criterion, in which vectors with peak ratio < 1.3 are deleted, a universal outlier detection median filter of size 5×5 pixel, and a vector group removal criterion of a minimum of 5 vectors, that is, if there are less than 5 viable neighboring vectors, the vector of interest is removed.

For this study, 30 experiments with 350 consecutive cycles are recorded and evaluated for 66 different CADs (-330°CA to -5°CA with a resolution of 5°CA), resulting in 10,500 total cycles, each with a measured maximum cycle pressure and 693,000 total flow fields. The 30 experiments of interest are each conducted in a skip-fire sequence to improve the thermal stability of the engine with 200, 200, and 400 fired

cycles each interrupted by 300 motored cycles. The measurements took place during the final 350 fired cycles and at fixed engine operation parameters (OP1, see Hanuschkin et al.²⁷). Notably the intake valves (IV) opened at 325°CA after top dead center (aTDC) and closed at -125°CA . Additionally, 7 more experiments (2450 cycles and 161,700 flow fields) are conducted with slightly different but fixed engine operation parameters (OP2: IV opened at 310°CA aTDC and closed at -140°CA) and used for pre-training the DNNs and fine-tuning the feature extractor.

Machine learning approach

Machine learning concerns the design and implementation of computer algorithms capable of learning how to solve problems without the need to program them explicitly. Hence, ML allows the exploitation and revelation of implicit knowledge and unknown relations. Applied ML employs ML algorithms to train an ML model on a training set. A well-trained and regularized model is capable of definitive inference or prediction on separate, previously unseen test sets. Supervised learning is a common way to train ML models. During training, the model's parameters are adjusted to map experienced input samples to corresponding output labels. Depending on the ML method, the input can either be a set of pre-processed input feature values or unprocessed data, from which features can be learned (see sections Feature Engineering and Feature Learning).

Machine learning methods can be used for classification, where the output (label) is categorical. In the case of only two classes (categories), the problem is called binary classification. Machine learning methods can also be used for regression, where the output (label) is continuous. The simplest ML method for regression is called univariate linear regression in which there is only one dependent variable and a single independent (explanatory) variable. A linear coefficient is fitted, reducing the mean squared error of the regression. Multivariate linear regression has several independent variables and accordingly several coefficients to be fitted. Additional regularization terms can be used to avoid over-fitting of the regression by penalizing, for example, the absolute sum of coefficients (L1-regularization; Lasso), the sum of the squared coefficients (L2-regularization; Ridge), or a combination of both (elastic net). Neural networks, like multilayer perceptrons (MLPs), can also be used for regression and promise to fit any function given a sufficient number of neurons in the hidden layer.^{30,31}

ML Models. To investigate CCVs in the Darmstadt engine, a binary classification task is formulated to predict high or low classes of P_{\max} (HC or LC) and identify important feature patterns. Therefore, for each cycle the corresponding P_{\max} value is labeled, depending on the class boundary given by the moving average P_{\max} value ($P_{\max,m.a.}$) for the particular cycle

(Figure 1(a)). The $P_{\max, m.a.}$ slightly increases during the course of each experiment due to the increasing engine temperature induced by the thermal load of consecutively fired cycles. Transient boundary conditions are unavoidable in realistic operation of optical engines, but the adaptive decision boundary counteracts this inherent thermal effect. In Figure 1(a), P_{\max} values of HC are colored deep red, LC petrol blue, and $P_{\max, m.a.}$ is shown by the red line. The moving average at cycle n is calculated by the average P_{\max} values of the cycle set $[n-10, n+9]$. At the edges, for example $n \leq 10$, the set is appropriately reduced. The classes are balanced by sub-sampling the majority class, that is, by randomly selecting an equal number of cycles to match the minority class, resulting in 9638 cycles for training. Boosted decision trees (AdaBoost)⁵⁶ and logistic regression models are trained on the training sets and evaluated on the test sets of the 10-fold stratified cross-validation (CV), that is, the data set is split into 90% training and 10% test sets 10 times while preserving the ratio of samples per class. Reported accuracies and standard deviations (std) are derived from the accuracies of the ML models' inference on the 10 test sets. Hyper-parameters are fitted once at -30°CA and are used for all other CADs to avoid information leakage to the ML models. Hyper-parameters of the AdaBoost include 85 estimators and a learning rate of 0.1. A regularization strength of $C = 100$ and a maximum of 1000 iterations are used for the logistic regression. Flame features are preprocessed with a standard scaler (mean subtraction and scaling to unit variance) and applying a principle component analysis (PCA)⁶³ explaining at least 97% of the variance. The best hyper-parameters for the MLP used in the downstream task are $\alpha = 0.8$, a batch size of 64, and a maximum of 1000 iteration steps with 5 neurons in the first layer. All other parameters are the default values defined in Scikit-learn v. 0.19. Reported feature importance is derived from the AdaBoost model, since decision tree methods allow direct feature importance calculations, for example, using the Gini importance.³²

Furthermore, to investigate CCVs in this study, different ML models (linear regression (LR) and MLP) are trained to predict the cyclic P_{\max} value for given features. The LR minimizes the mean squared error; the quality of the regression was quantified by the mean absolute error between predictions and labels. Hyper-parameters of the LR with elastic net regularization (LR_{EN}) are $\alpha = 0.01$, a L1 ratio of 0.99, and a maximum of 500 iteration steps. The best hyper-parameters for MLPs are $\alpha = 100$, $lr = 0.01$, and a batch size of 512 with 2 and 5 neurons in the first and second hidden layer, respectively. They are fitted once at -80°CA . All other parameters are the default values defined in Scikit-learn v. 0.19. Feature importance cannot be extracted from the coefficient of a LR or the weights and biases of a neural network directly. Instead, indirect and iterative methods like LOCO (leave-one-covariate-out) can be applied.

Table 1. Engineered flow features as in Hanuschkin et al.¹⁹: The set $\{v_{x, \max}, v_{x, \min}, v_{x, \text{mean}}\}$ is abbreviated by v_x : (max/min/mean). Sections defined in Figure 6(a).

Feature	Description	Region
1 to 9	$v_x, v_y, v $: (max/min/mean)	Global
10 to 18	$v_x, v_y, v $: (max/min/mean)	Section 1
19 to 27	$v_x, v_y, v $: (max/min/mean)	Section 2
28 to 36	$v_x, v_y, v $: (max/min/mean)	Section 3
37 to 45	$v_x, v_y, v $: (max/min/mean)	Section 4
46 to 54	$v_x, v_y, v $: (max/min/mean)	Section 5
55 to 63	$v_x, v_y, v $: (max/min/mean)	Section 6
64 to 72	$v_x, v_y, v $: (max/min/mean)	Section 7
73 to 81	$v_x, v_y, v $: (max/min/mean)	Section 8
82 to 90	$v_x, v_y, v $: (max/min/mean)	Section 9

Table 2. Engineered flame features inspired by Hanuschkin et al.²⁷

Feature	Description	Feature	Description
1	Total area	8	Rightmost x
2	Perimeter	9	Distance bottommost-piston
3, 4	Centroid x,y	10	Contour height
5	Bottommost y	11	Contour width
6	Topmost y	12, 13	Distance contour-piston (mean, std)
7	Leftmost x	14, 15	Contour x-values (mean, std)

Feature engineering

To solve supervised learning tasks, ML models are trained with data samples and their corresponding output labels. Samples use either manually engineered features for the given problem, or raw data from which ML algorithms have automatically extracted features. In the following section, engineered features of flow fields and flame contours are described. In the succeeding section, DNNs are employed for automatic feature learning.

Flow field features. Features derived from basic flow field statistics have been shown to be sufficient for predicting HC and LC and are not inferior to high-level engineered features like the in-plane tumble flow.¹⁹ For the mid-cylinder symmetry plane, 90 distinct features are extracted for each CAD (Table 1), that is, for each of the 9 sections (illustrated in Figure 6(a)) and the global field of view (FOV), the minimum, maximum, and mean value of the in-plane velocity components (x and y), and in-plane magnitude are evaluated.

Flame features. The development of the early flame kernel at -15°CA is quantified by a defined set of features (see Table 2). In a cross-section of a flame, discrete contours of the flame can emerge since the flame is a 3D structure forming and propagating around the spark

plug, and is imaged in a 2D plane. Engineered features describe global quantities such as the total area (first feature), summed perimeters (second feature) position of the center of mass (features 3 and 4), and extreme points of the flame contour or contours (e.g. features 5 to 8). Features 9 to 11 are derived from these global features and hence might be correlated to them (see Appendix). The contour height and width is given by the distance between features 5 and 6, and 7 and 8, respectively. The distribution (mean and std) of the contour points in the y-direction are given by features 12 and 13. Features 14 and 15 are used for the distribution in the x-direction. In a previous study,²⁷ engineered flame features were shape (and orientation) and position related, which is possible for single contours in the flame cross-section. However, this restriction to only single contours reduces the amount of available samples and most certainly biases the model, thus having an effect on the accuracy of the models' prediction. Single and multiple flame contours of experiments (OP1, $P_{\max} \in [-34, 35]$ bar) with a high particle contrast are extracted (1724 samples). After class balancing, that is, matching the number of samples for each class, by subsampling the majority class, 1280 samples remain.

Feature learning

While DNNs are often trained end-to-end, it can be advantageous to train the feature extractor separately (transfer learning). A general (pretext) task can be defined on a large data set for feature learning, which is different from the specific classification or regression (downstream) task.^{22,25} In-between these two steps, an optional fine-tuning step can be introduced to adjust the learned features to the downstream task. Figure 2 illustrates how a pretext task, fine-tuning, and a downstream task build on each other.

In the following, autoencoding²⁴ is used as a self-supervised pretext task to encode, or to compress the flow field to features in latent space, and decode, or to expand the compressed features back into flow field samples. Optionally, the obtained encoder is then extended with an MLP and fine-tuned in the downstream task. In the final step, the obtained encoder extracts feature values from the flow field to be used in an AdaBoost classifier to achieve comparable results to AdaBoost classifiers trained on hand-engineered features. In this process, the decoder is only used for the training, that is, evaluation of the CAE.

Deep neural networks allow automated feature learning but require manual pre-processing steps to enable or enhance the learning process. To process the flow fields, the 2D velocity vectors \vec{v} with components v_x and v_y , are transformed into 3D vectors $(\frac{v_x}{|\vec{v}|}, \frac{v_y}{|\vec{v}|}, \frac{|\vec{v}|}{C})$ to separate the vectors' directions from their magnitudes. The constant C was chosen as 120 m/s to normalize the velocity magnitude ($\frac{|\vec{v}|}{C} \in [0, 1]$). Normalizing input values enhances learning performance

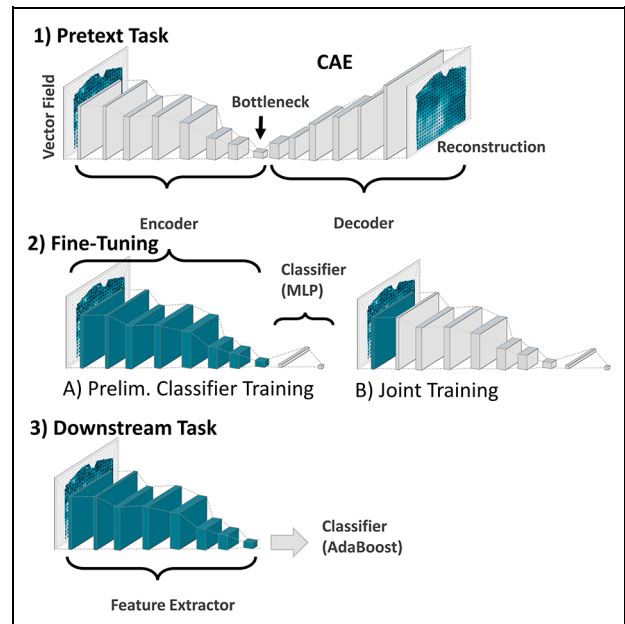


Figure 2. Feature learning: first a convolutional autoencoder (CAE) is trained to compress flow fields in a lower dimensional feature space (bottleneck) before decoding a reconstruction of the input. Secondly, the weights of the encoder are fine-tuned to be suitable for the downstream task. In the preliminary classifier training step, the decoder is replaced by an MLP which is trained on a similar task to the downstream task. The layers of the encoder are frozen during this step (petrol color). Then the encoder and the MLP are trained jointly, while only keeping the first layer frozen. Finally, the fine-tuned encoder is fixed (layers frozen) and used for feature extraction in the downstream task.

of DNNs because it reduces vanishing or exploding gradient problems during training.^{33,34}

Autoencoding (pretext task). Feature representations can be extracted in a self-supervised fashion using AE (see Figure 2; pretext task). The task of undercomplete or regularized AEs is to compress the input information and reconstruct it, by encoding the input in a lower dimensional space (latent space, representing simple or abstract features) into a bottleneck layer, followed by decoding this compressed representation. The ML algorithm is trained on samples of pre-processed (see above) in-cylinder flow fields, which at the same time serve as the output target. A bottleneck size of 96 neurons is chosen (see section Investigated CAE architectures in the Appendix) to approximately match the size of the engineered feature set dimension of 90 (see section Flow Field Features).

Convolutional autoencoders use convolution layers to detect local low-level features in shallow layers and combine them to increasingly higher-level features with increasing vicinity to the bottleneck layer. By defining a set of relatively small convolution filters (kernels) of size 3×3 with 9 weights and 1 bias (see Figure 3), the number of trainable parameters is significantly reduced compared to AE with fully connected layers. A small

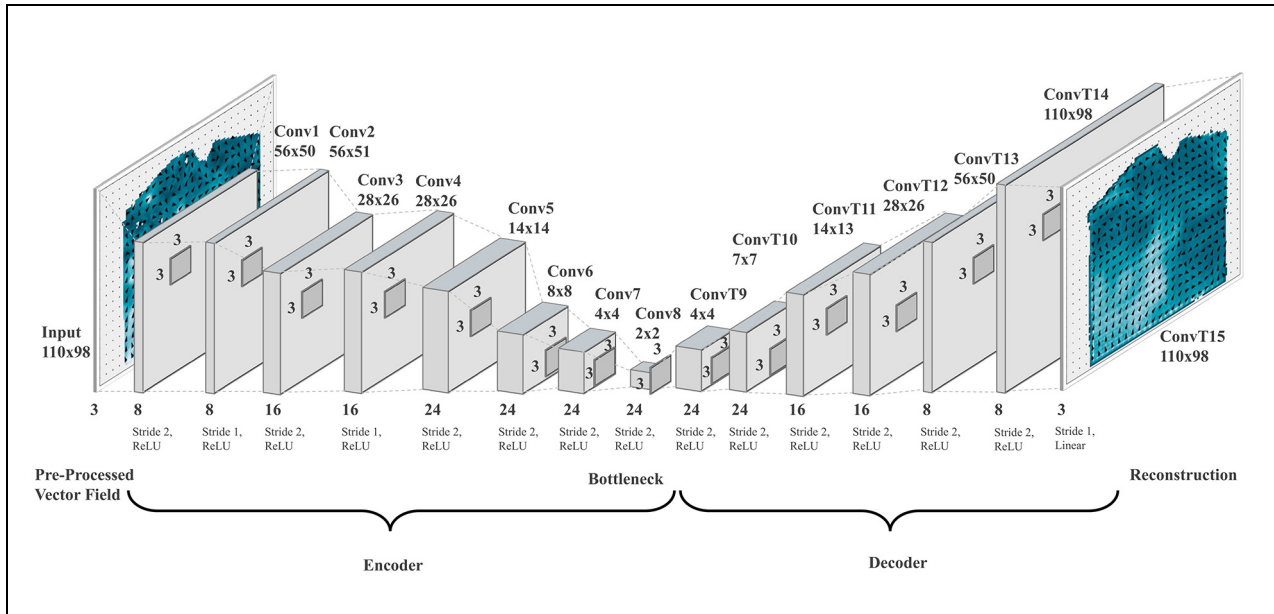


Figure 3. Autoencoder architecture: the flow field input is compressed in the encoder by convolutional layers with increasing number of feature maps and increasing vicinity to the bottleneck. The flow field is finally encoded in the latent space of the bottleneck layer. A set of transposed convolutional layers with decreasing number of feature maps build the decoder. A fixed kernel size of 3×3 for each layer is used. Depending on the layer, a stride of 1 or 2 and ReLU or linear activation functions are applied. The same padding is used for convolution-based layers except the final layer. Cropping is omitted.

kernel size further allows resolving small and local features in lower levels. In practice, smaller kernel sizes with deeper NNs have shown to be superior to larger kernel sizes with shallower NNs.^{35,36} This leads to the need of increasingly deeper NNs to achieve performance increases.

To perform a compression from the 110 (vectors) \times 98 (vectors) \times 3 (components) dimensional input to the 96 dimensional bottleneck, a stride of 2 is used in the encoding convolutional layers. The stride defines how the kernel convolves the input. A stride of 2 shifts the receptive field of a kernel by 2 units, downsampling the kernel's input. The number of feature maps is increased with the depth of the convolutional encoder. This enables the network to learn more distinct high-level features in deeper layers. The lossy compression ratio between input and latent space coding is 336.9. For decoding, transposed convolutions are used. The "same padding" is used for all convolutional-based layers, that is, the minimal padding size p is chosen such that $i + 2p - k$ is divisible by the stride of 2 or equals zero in the case of a stride of 1, where i is the input size and $k = 3$ the filter size. Rectified linear activation units (ReLU)³⁷ are used as the activation function in all except the final convolutional-based layers to induce nonlinearity. A linear activation unit is employed in the final layer. The structure of the CAE is sketched in Figure 3.

Samples of both engine operation points (OP 1 and OP 2), but either only during the compression stroke or during the intake and compression strokes are considered for training of the CAE. Restricting training data

to the compression stroke reduces the amount of samples with piston-reduced FOV and allows an analysis of learned feature generalization to data in the intake stroke. The CAE's reconstruction performance of the original input is measured by the mean squared error (MSE) and is used in the loss function (compare Appendix). The CAE is trained jointly, that is, the CAE is trained as a whole and not layer-wise (stacked training). The loss function is optimized with the adaptive moment estimation (ADAM) algorithm with the learning rate set to 10^{-3} . Models are trained in mini-batches of size 512, for a maximum of 100 epochs. Early stopping is used to reduce iteration times and regularize the model. If the minimum change of the model's validation loss does not exceed 5×10^{-4} for 10 epochs, it is assumed that the learning algorithm has converged. Finally, the best models of a training run are determined by the lowest validation loss. All tested hyper-parameters are selected empirically.

Fine-tuning. To optimize the learned features for the downstream task, that is, the classification of HC and LC, the pre-trained encoder is fine-tuned by training on the same task with a different but similar data set. All samples of the compression stroke of OP2 are used and a 70:30 train-test split is applied. Since the fine-tuning and downstream tasks are the same, samples from OP1 are not used to avoid information leakage into the model.

For fine-tuning, the pre-trained encoder is separated from the rest of the AE (see Figure 2; fine-tuning) and the existing decoder is replaced with fully connected

layers for classification. To this end, the encodings of the bottleneck are flattened and passed to a two-layer MLP consisting of 16 neurons in the hidden layer and 1 neuron in the output layer. A MLP is chosen because, in contrast to other classifier method (e.g. AdaBoost), it simply extends the neural network structure and hence allows fast training. The hidden layer of the classifier applies a ReLU activation function.³⁷ A sigmoid activation function ($1/(1 + e^{-x})$) is applied at the MLP's output layer and the binary cross-entropy serves as the models' loss metric (compare Appendix). The fine-tuning procedure is further subdivided into two steps: first, the newly attached classifiers are pre-trained in isolation until convergence, that is, all trainable parameters of the feature extractor are frozen and only the classifier layers remain trainable. This can prevent catastrophic forgetting due to weight changes in the feature extractor that are too drastic. Second, both the feature extractor and binary classifier are trained in unison. In order to maintain filters trained to extract low-level features, weights in the first convolution layer of the feature extractor remain frozen.

During fine-tuning, the models are trained with mini-batches of size 512 for a maximum of 10^4 epochs. Once again, early stopping is employed to reduce iteration times and regularize the model. The patience is set to 200 epochs with a delta of 10^{-3} . The loss function is optimized with the ADAM algorithm. For the pre-training of the classifier, the learning rate is set to 10^{-3} . For the joint training of the feature extractor and the classifier, the learning rate is reduced to a value of 10^{-4} to retain already learned features in the feature extractor. The specific values of hyper-parameters are also selected empirically.

Downstream task. The first two previous steps, training a CAE on the pretext task and optionally fine-tuning its encoder part (see Figure 2), generate ML models which are able to encode flow fields into a suitable low dimensional representation (latent encodings). These latent encodings are learned feature representations and can be used, in an additional final step, to train subsequent ML models on the specific downstream task of classifying HC and LC based on the given flow fields. To this end, the generated feature representations are fed into subsequent AdaBoost or MLP classifiers and trained for each CAD individually to classify HC and LC (see Figure 2; downstream task). The AdaBoost algorithm evaluates the Gini impurity to train individual weak learners (single trees) and the misclassification rate to calculate their stage value (see Appendix). Binary cross-entropy is used as loss function in MLP training (see Appendix). Hyper-parameters are fitted at -30°CA and their values are given in the Results and Discussion section. The results are evaluated with 10-fold CV training on all OP1 samples.

Feature analysis. Class activation maps³⁸ reveal spatial regions of interest. These allow the interpretation of the classifier's decisions but do not expose the specifics of learned features directly. Activation maps are generated class-wise with the gradient-weighted class activation mapping (CAM) algorithm³⁸ and are averaged over 250 cycles. To narrow down the most informative regions, a threshold of 0.5 is applied to the activation map. Values below the threshold are zeroed out. Finally, the averaged centers of mass are calculated for all time points to visualize temporal and spatial attention shifts over the course of a cycle.

Python 3.6 is used throughout the study with the AdaBoost, logistic regression, MLP, and LR implementations in Scikit-learn³⁹ (v. 0.19). DNNs are implemented in TensorFlow (Tf v. 2.0, CUDA v. 10.0.130). Tf-Explain (v. 0.2.1; Sicara SAS, France) and Tf-Keras-Vis (v. 0.4.1)⁴⁰ are used to interpret classification decisions with class activation maps (GradCAM)³⁸ and GradCAM++, respectively. DNN models are trained on a workstation outfitted with 16 Intel Xeon Silver 4112 quad core CPUs and three Nvidia V100 GPUs. Results of the DNN inference are visualized with OpenCV (v. 4.2.0.32). Data plots are generated with matplotlib (v. 2.0.0 or 3.1.2) and post-processed with Inkscape (v. 1.0).

Results and discussion

Two different ML approaches are applied and compared in the following. ML methods using engineered input features are investigated first. This approach has been successfully applied to smaller data sets of flow fields and flame contours.^{18,19,27} In the second ML approach presented in this work, features are learned by a CAE. Results of the first approach serve as a baseline for comparison with the second ML approach.

Machine learning with engineered features

In this section, different ML approaches, namely classification and regression, are employed using engineered input features. To disentangle the features' influences on CCV predictions, different engineered feature sets (see section Feature Engineering) are tested. First, 90 flow field-derived features (f_{vf}) and 90 flow field features combined with the exhaust temperature T ($f_{vf,T}$) are used as feature sets. The first features set is extended with 15 flame contour-derived features ($f_{vf,flame}$). Finally, a feature importance analysis for the velocity-derived features is performed.

Classification. A binary classification task is defined by splitting the measured engine cycles into high and low P_{max} cycles (HC and LC, respectively), where the class boundary is given by the moving average P_{max} value (Figure 1(a)). Results for classifier models (AdaBoost) with the feature set f_{vf} are shown in Figure 4(a). After

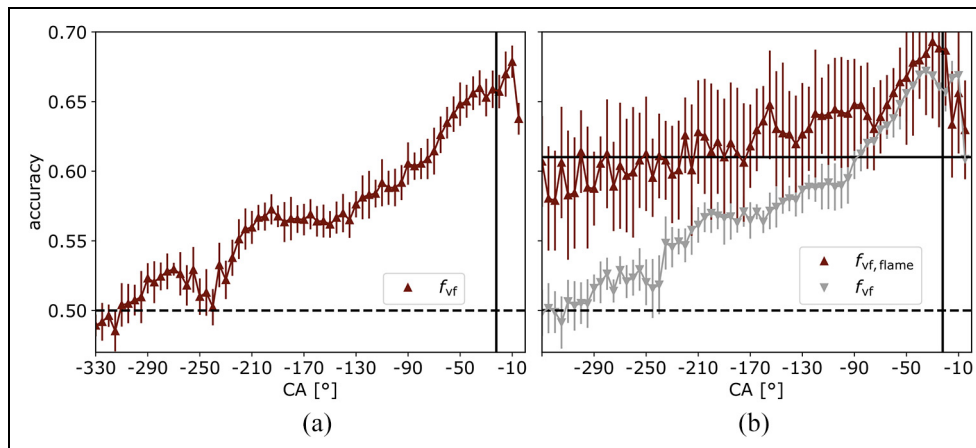


Figure 4. Classification results based on engineered features: (a) binary classifier (Adaboost) trained to distinguish HC and LC based on f_{vf} (deep red color). Chance level is indicated by the dashed line at the accuracy of 0.5. Time of ignition is given by the solid vertical line at -22.2°CA . (b) binary classifier (logistic regression) trained on additional flame features ($f_{vf, \text{flame}}$; deep red color). Accuracy based on 15 flame features alone at -15°CA is given by the horizontal solid black line. The binary classifier (logistic regression; f_{vf}) is given as the baseline (gray color).

-290°CA , that is, already during the intake phase, the model is able to classify HC and LC with mean accuracies at least one std above the baseline (chance level at 0.5) with few exceptions (-250°CA to -240°CA). The accuracies increase during the compression and the highest accuracies are achieved after ignition at -10°CA (0.68 ± 0.01). The increase in accuracy after -240°CA coincides with a transition of the flow from a strong intake-dominated structure to an emerging tumble structure with high velocity magnitudes from the impinging wall jet near the exhaust side (see Figure 6(a)). Accuracies then reach a plateau and increase again after -130°CA . At this CAD, on average, the tumble center moves into the FOV. Both observations might indicate the relevance of the tumble flow and the coherent structures that define it for the early combustion processes around the spark plug. On a similar, though smaller data set, it has been previously shown that such a binary classification approach is capable of revealing spatial flow field features important for high IMEP values at individual time points during the compression phase.¹⁹ The results shown in Figure 4(a) are consistent with these findings and hence the applied methods seem to be robust in different experimental setups and specific engine types. However, in the present study, the ML models already distinguish HC and LC during the intake phase, pushing the boundary of this ML approach beyond what was previously possible to earlier time points during the combustion cycle. This might be explained by the larger data set (9638 sample cycles after pre-processing) compared to the previous study (544 sample cycles) even though it cannot be ruled out that it is a result of the different experimental engine types and operational conditions used. For example, a port fuel injection engine configuration is used in this study, which might explain why early intake flows might be more informative than in a direct injection configuration.

The logistic regression classifier (Figure 4(b), gray color) shows qualitative and quantitative similar behavior compared with the results of the AdaBoost classifier (average accuracy and pooled variance: 0.571 ± 0.013 AdaBoost, 0.575 ± 0.014 logistic regression). Adding flame features obtained at -15°CA ($f_{vf, \text{flame}}$; deep red color) remarkably increases the mean accuracies of the ML models during intake and early compression. Their mean accuracies are at least one std above the flame-features-only baseline between -80° and -15°CA with the highest accuracies at -30°CA (0.69 ± 0.06). The baseline is given by the accuracy of a ML model trained on flame features alone (logistic regression; 0.61 ± 0.04 ; black horizontal line). Note that non-causal models before -15°CA are generated because at each evaluated CAD before ignition, information about the flame contour at -15°CA is present. $f_{vf, \text{flame}}$ has a smaller total sample size of 1280 cycles because flame contours are only extractable in a subset of experiments, and the majority class was sub-sampled to balance the classes. This leads to much larger std compared with models trained with feature set f_{vf} (9638 cycles; Figure 4(a) and (b)). Due to the large std, a direct comparison to the classification with only flow field-derived features is difficult to make. Nonetheless, a comparison shows that the highest mean accuracies are slightly higher for feature sets augmented with features of the flame contour in the range of $[-80^{\circ}; -15^{\circ}]\text{CA}$.

Regression. Various regression methods are tested to build an ML model that can predict individual P_{max} values given the feature sets f_{vf} , $f_{vf, T}$, and $f_{vf\text{-global}, T}$ (9 global flow field features and T). Since the P_{max} value correlates with T , a univariate linear regression (LR) model of P_{max} given T serves as the baseline in the following. The mean absolute error (MAE) of this baseline is 1.04 bar. The best regression models, namely

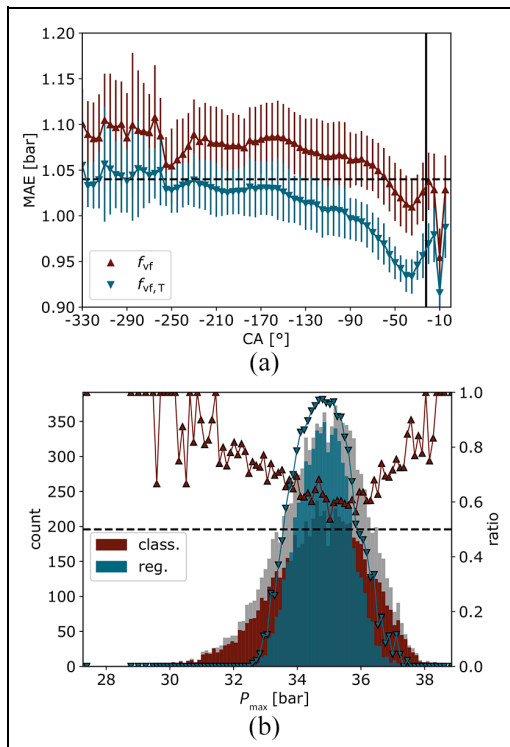


Figure 5. Regression results based on engineered features: (a) linear regression (elastic net) trained to predict P_{max} values using f_{vf} (deep red color) or $f_{vf,T}$ (petrol color). Baseline of linear regression (elastic net) with only the temperature feature is given by the horizontal dashed line. The vertical solid line marks the ignition. Error bars indicate one std. (b) distribution of experimental (gray), correctly classified (deep red), and well-predicted (MAE below baseline, petrol) P_{max} values at -80°CA . The percentage of correctly classified (deep red, upward triangles) and well-predicted (MAE below baseline, petrol, downward triangles) samples.

LR with elastic net regularization (LR_{EN}) and MLP, achieve an MAE of 0.99 ± 0.02 bar and 1.00 ± 0.02 bar, respectively, on the CV test sets for $f_{vf,T}$ at -80°CA (Figure 5(a)). Interestingly, models trained with $f_{vf\text{-global},T}$ are only slightly inferior (not shown, LR_{EN} 1.02 ± 0.02 bar; MLP 1.02 ± 0.02 bar) at -80°CA . The average MAEs of ML models trained with feature set $f_{vf,T}$ (see Figure 5(a)) are at least one std below the baseline after -80°CA and reach the best performance at -40°CA (0.93 ± 0.02 bar). Without the temperature feature (f_{vf}), the average MAEs are at least one std below baseline solely at -10°CA (0.95 ± 0.03 bar) and exhibit higher std.

Classification and regression results show similar qualitative behavior (Figures 4 and 5(a)). The mean accuracies of the classification models increase while the MAEs of the regression models drop toward the ignition. The approaches exhibit their highest accuracy (classification) or lowest MAE (regression) at -40°CA . However, the classification approach already allows the prediction of HC and LC cycles during the intake phase after -290°CA , while the regression model is on average one std below the baseline only after -130°CA .

Furthermore, the std of classification models are far lower than the std of the regression models, even though both approaches were trained with the same sample size.

To analyze this further, a histogram of the experimental P_{max} values (gray color) is shown in Figure 5(b). Overlaid on this histogram, is a histogram of correctly classified P_{max} values (deep red color) and a histogram of well-predicted P_{max} values, defined by an absolute error below the baseline value (1.04; petrol color). Ratios of correctly predicted classes (upward triangles, deep red color) and well-predicted P_{max} values (downward triangles, petrol color) to the population are superimposed (secondary y-axis). It becomes obvious that the classifier works perfectly (ratio = 1) for extreme samples of either very high or very low P_{max} values, while the regressor is very accurate (ratio ~ 1) close to the average P_{max} value. These findings can be explained by the differences of the underlying ML algorithms. Classification algorithms aim to find decision boundaries, for example, by optimizing model parameters in order to separate the classes. In the case of a binary classifier, extreme samples with very high or low P_{max} will be easily separated, but perform worse in closer proximity to the decision boundary. By contrast, regression algorithms aim to fit with the majority of the samples to reduce the overall fitting error. This leads to regression models that predict samples with average behavior, for example, average P_{max} values, more easily than extreme samples, for example, very high or low P_{max} values.

Feature importance. The feature importance can be directly calculated for each CAD from the AdaBoost classifier (see section Methods). For each CAD the feature importance is section-wise-averaged (sections as marked in Figure 6(a), left column) and presented in Figure 6(b). Regions marked by I and II in Figure 6(b) do not contribute to the ML models (feature importance = 0, white area) because the piston covers the corresponding sections during these CADs and classifiers cannot extract information.

Interestingly, an area of low importance (indicated by the dashed line in Figure 6(b)) is observable during $\sim[-220^{\circ}; -150^{\circ}]\text{CA}$ of sections 7–9, which belong to the upper part of the cylinder. During this period, the intake valves begin to close, the intake mass flow and therefore intake velocities decrease, and most of the cylinder filling has already taken place. After bottom dead center (BDC), the upward moving piston pushes cylinder gases back into the intake duct until the valves fully close. Therefore, the flow field close to the cylinder head during that period provides little or at least less important information useful for predicting HC or LC. The classifier deems global features, that is, features which are obtained from the entire FOV as important all of the time, except for the period $\sim[-140^{\circ}; -60^{\circ}]\text{CA}$.

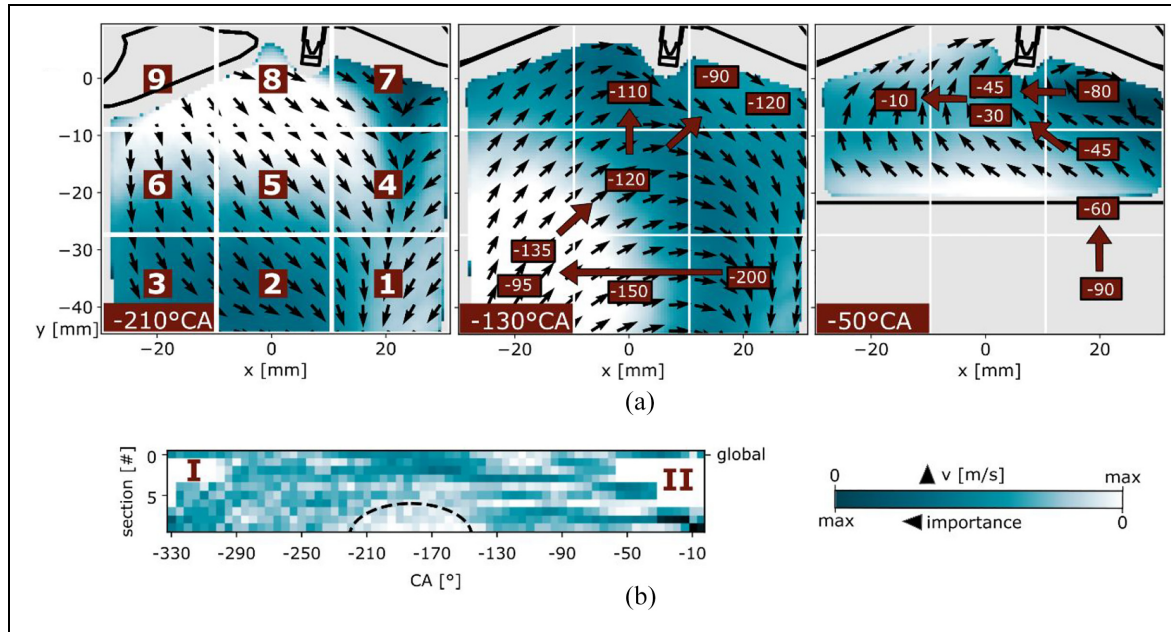


Figure 6. Feature importance analysis: (a) average (single experiment, 350 cycles) flow field at -210°CA (left, velocity range 0–20 m/s) with numbered spatial regions, at -130°CA (middle, $v_{\text{max}}=8$ m/s), and at -50°CA (right, $v_{\text{max}}=10$ m/s). Thick arrows in the middle and right plots (deep red color) illustrate the change of regional importance with CAD (small white numbers, deep red background). (b) Section-averaged velocity feature importance (section 0 = global). Velocity features are calculated in 9 spatial sections as depicted in panel A. Two regions marked by I and II are highlighted. All panels share the same color bar (color gradient from dark petrol ($v=0$ (a); very important (b)) to white ($v=v_{\text{max}}$ (a); not important (b))).

In a similar analysis, Hanuschkin et al.¹⁹ observed a clear spatial-temporal pattern of feature importance evolution during the compression stroke (see Figure 6(e) in Hanuschkin et al.¹⁹). In the results of the present work, however, the spatial-temporal pattern (Figure 6(a) middle and right columns, arrows in deep red color) is comparable but less structured. This might be a result of the different experimental setups and engine types used. Notably, the tumble in the mid-cylinder plane is weaker in the investigated engine, which might be the reason for the less distinct spatial-temporal pattern of importance. Nevertheless, it seems that at the beginning of the compression stroke, sections 1 and 2 are important, which are later replaced by sections 3, 5, and 7. The phase-averaged flow field during this early compression period is dominated by the impinged wall jet, which disappears from the FOV through sections 1 and 2 (Figure 6(a)). When sections 3, 5, and shortly after, 7 become important, the opposing side of the tumble reappears with high velocity magnitudes in sections 3, 5, and 6. Interestingly section 6 is not deemed important at all, which might be a result of correlation and therefore redundant information between different sections. During the second half of the compression stroke the sections of highest importance roughly coincide with the regions below the tumble center. During that period and at the end of the compression stroke $\sim[-35^{\circ}; -15^{\circ}]$ CA section 8 becomes progressively more important. After ignition $\sim[-10^{\circ}; -5^{\circ}]$ CA section 9 takes over. Since the piston is nearly at top dead

center (TDC) and the cylinder content is compressed to a small volume, only sections 7–9 are populated. The dominating importance of section 8 (and 9) is in line with previous studies that analyzed flame-derived features only. It was found that the flame development into the negative x-direction, and in general the centering of the flame inside the cylinder volume, is of major importance.²⁷

Machine learning with learned features

In this section, DNNs are employed for unsupervised feature extraction. An AE is trained on 70% of all flow field samples from either the compression stroke only or intake and compression strokes until the loss parameter (i.e. the MSE between input and reconstruction (compare Appendix)) has converged. Figure 7 shows three exemplary flow fields and their corresponding CAE reconstructions from a single cycle during the early intake stroke (-305°CA), mid-compression stroke (-105°CA), and late-compression stroke (-5°CA) after ignition. The CAE can correctly reproduce the shape of the flow field down to small structures, which are visible close to the flow field borders. Areas of the flow field that are missing data due to, for example, thrown out vectors in the top right corner at -305°CA , are interpolated in the reconstruction. In general, the CAE smooths both the velocity direction and magnitude information. The CAE generalizes in the intake stroke since it was trained on data from the compression

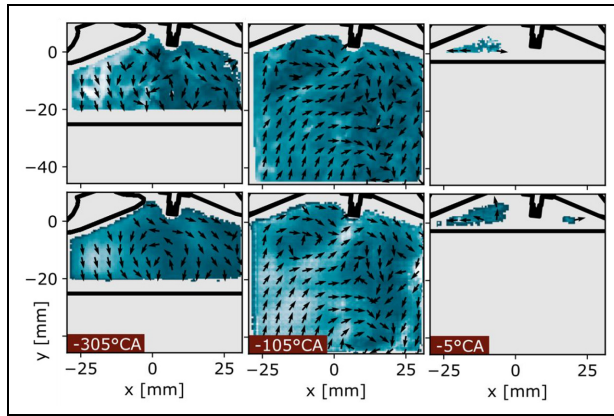


Figure 7. CAE reconstruction examples: recorded flow fields (top row) and CAE's reconstructed flow fields (bottom row; DL_c) for three different times (columns) of a single combustion cycle. Contours of the cylinder head, spark plug, piston, and valves are given for orientation. The color bar is shared with Figure 6: velocity range from 0 to 60 m/s (left), 15 m/s (middle), and 20 m/s (right).

stroke only. At $-5^\circ CA$, many of the seeding particles are burned in the flame region and the CAE is still able to reproduce this partial and irregularly shaped flow field. However, an artifact of this prediction is visible in the top right region of the example where a tiny flow field recording is hallucinated. This artificial creation of data might be explained by a high probability of a flow field in this region at this time, which is utilized in one or more features of the encoding.

Classification. Features extracted from the encoder part of a CAE trained on 70% of all flow field samples are used to train AdaBoost classifiers for each CAD using 10-fold CV (Figure 8; black color; $DL_{i,c,noFT}$; AdaBoost: $lr = 0.2$, $n_{est} = 400$). The results are qualitatively similar to the baseline results of AdaBoost classifiers trained on engineered features (Figure 8; gray color and Figure 4(a)). After $-290^\circ CA$ accuracies are at least one std above chance level throughout the cycle with few exceptions ($-255^\circ CA$ to $-240^\circ CA$) and increase slowly toward the late compression stroke ($> -90^\circ CA$), where accuracies of the models increase further. During this period, the fine-tuning of the encoder, that is, pre-training of the encoder on a task similar to the downstream task, increases the models' accuracies remarkably (Figure 8; deep red color; $DL_{i,c}$; AdaBoost: $lr = 0.3$, $n_{est} = 140$), while for earlier times, there is no performance gain. Flow field information in this period has more predictive power for the downstream task, as shown by the baseline models, and hence might be better suited for fine-tuning the encoder. Interestingly, the late compression stroke is also the period where the piston moves into the FOV. It might be that the fine-tuning process guides the encoding from representing the piston position to flow field-related features important for the downstream

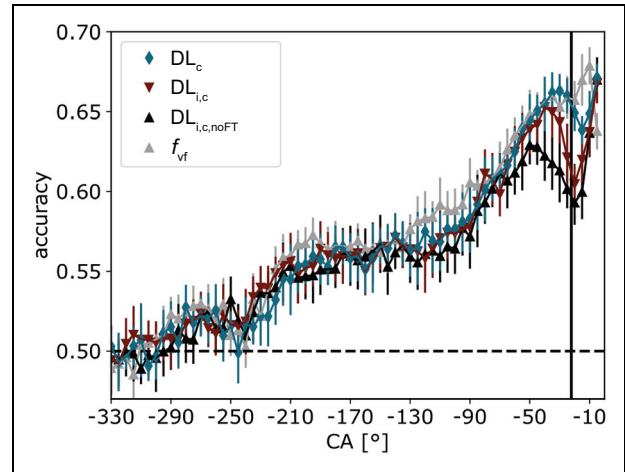


Figure 8. Results of DL: AdaBoost classifier with features extracted from the encoder part of the CAE. For each CAD the mean and std of a 10-fold CV is given for the encoder either with (deep red downward triangles, $DL_{i,c}$) or without fine-tuning (black upward triangles, $DL_{i,c,noFT}$). The encoder with fine-tuning trained in the compression stroke only is shown (petrol diamonds, DL_c). AdaBoost with engineered features (f_{vf}) is given as a baseline (gray upward triangles, compare Figure 4(a)).

task. The performance of the CAE trained on flow field samples from the compression stroke only (Figure 8; petrol color; DL_c ; AdaBoost: $lr = 0.2$, $n_{est} = 400$) is the highest and reaches accuracies near the baseline in late compression ($> -90^\circ CA$). The accuracy gain in the late compression stroke, however, does not sacrifice the accuracy during the intake stroke.

During late compression at $-5^\circ CA$, all CAE-based models' accuracies are better than the baseline. At this CAD, the in-cylinder content is compressed to a very small volume, the experimental flow fields are restricted to the cylinder head region, and the flame evaporates most of the seeding particles, resulting in partial flow field data. The CAEs learned features for this state that are superior to the engineered flow field features, which are not designed to handle either of the two aforementioned factors. In contrast, by accurately encoding flow fields at $-5^\circ CA$, the coding capacity for earlier flow fields might be lowered. This can lead to a lower average accuracy of the model during times where the piston is not in the FOV. However, this effect is not quantified, yet nonetheless reasonably acceptable, since the accuracies of the CAE methods are all nearly the same as that of the baseline throughout the rest of the cycle.

At the time of the initial flame kernel ($-15^\circ CA$) the results of AdaBoost models with engineered flow field features (0.67 ± 0.02), engineered flame features (0.62 ± 0.03), a combination of the former two (0.63 ± 0.04), and learned features (0.62 ± 0.01), can be compared (Figures 4(a), (b), and 8). Models based on hand-engineered flow field features are superior to other approaches at this CAD. The weak performance

of models based on engineered flame features is discouraging, due to the effort needed for the algorithmic extraction of hand-engineered flame features from raw PIV images, which is error-prone and requires tedious verifications.²⁷ The superior performance of the flow-based features may be related to a fundamental characteristic of flow field data: if the temporal resolution is better than the characteristic time scales of coherent flow structures, the flow field carries information about the future trajectory of itself. In comparison to the mere flame position encoded in the flame features derived from particle-void regions, flow-based features therefore seem to correlate better with the cycle performance.

Still, it is surprising that the greatest prediction accuracy at -15°CA is achieved with the engineered flow field feature model, which has spatially unresolved features (compared with the spatially superior resolved features of the learned feature model and the position information incorporated in the engineered flame feature model). Further, it appears that features in section 8 alone have enough predictive power (based on the temporal feature importance analysis) to be superior to the other models. The distinct increase in model accuracy observed already during the late intake indicates a deterministic flow development from the intake to the late compression flow near the spark plug, even in this limited 2D section in the symmetry plane. This highlights the importance of a predictable and repeatable intake flow design in avoiding CCVs¹¹ and allows the optimization of future engine designs by identifying important early flow structures that can be influenced by intake flow phenomena.

Even though the engineered features for the baseline models have a low spatial resolution and use coarse features like average velocity and extreme velocity values during the compression stroke, the accuracies of models with learned features are consistently (except -5°CA) lower than the baseline models' accuracies. A set of features learned during the pre-text task encode information irrelevant to the downstream task, for example, the piston's position. While the fine-tuning elevates the encoder's performance, the baseline accuracy is not reached. Spatial-temporal fine structures, measured by the MSE between input and reconstruction, might be relevant for a high CAE performance. However, they might hinder the learning of abstract or unspecific features, which may be important for the downstream task of investigating CCVs. Learning to encode regular patterns in flow fields with high precision does not increase the performance of classifiers if these patterns do not have predictive power.

The highest accuracies are obtained for the CAEs which are trained and fine-tuned on data from the compression stroke only (DL_c ; CAD-average accuracy and pooled std: 0.564 ± 0.015 (AdaBoost) and 0.558 ± 0.021 (MLP)). By this, the complexity of the feature space is reduced while the model is still able to generalize to the intake stroke. Furthermore, this approach reduces the

effect of learning irrelevant features or features with low predictive power, because CCV analysis is difficult during the intake stroke. When using the whole data set of flow fields from intake and compression stroke, while qualitatively and on average quantitatively similar ($\text{DL}_{i,c}$; 0.563 ± 0.015), a clear accuracy difference is visible in the late compression stroke. Using a sufficiently large data set from the same CAD might be beneficial in reducing the feature space by encoding only the CAD-specific flow shape and might result in higher downstream task accuracy. Without fine-tuning ($\text{DL}_{i,c,\text{noFT}}$), and for the baseline model (f_{vf}) the average accuracies are 0.556 ± 0.015 and 0.571 ± 0.013 , respectively.

One explanation for the generally low performance of the presented ML classification models (accuracy of 0.67, shortly before ignition) is the definition of the class boundary (see Appendix; compare^{19,27}), the shape of the P_{max} -distribution (see Figure 1(b)) combined with the inherently increasing uncertainty a classification algorithm exhibits near the defined decision boundary (see discussion of Figure 5(b)). In addition to the possibility that the patterns used to derive the learned features do not have predictability for the given downstream task, another explanation for the low predictive power of the ML models, is the difficulty of the ML task itself. Despite the significant experimental effort, the extensive data set used in this study is still limited considering it only includes two vector components in a 2D cross-section of the 3D flow field. Other influencing factors that cause CCVs such as the exhaust gas distribution, spark energy and spark movement, turbulent flame development, and flame-wall interactions are neglected by design of this study. Still, some of these influencing factors might be embedded in the flow data. Although the unsupervised nature of the learned feature approach allows for a more abstract, downstream task-specific and higher-level feature generation, which might also incorporate these hidden influences, basic local flow field statistics like the ones used in the engineered feature set seem to be as predictive for the given task or encode less irrelevant information (e.g., piston position) and at the same time need less samples for training. Furthermore, the interpretation of "black-box" models like the CAE is more difficult and discussed in the following section.

Interpretation and understanding of the deep neural network model. The interpretation and understanding of DNN models is a vivid ongoing research topic.⁴¹ While it is challenging to understand decisions and learned features of DNNs due to their non-linear and complex natures, several methods like class activation mapping (CAM)³⁸ can be employed for interpretation.

Class activation mapping. Class activation mapping does not expose the specifics of learned features, but the resulting regions of interest allow a visual explanation

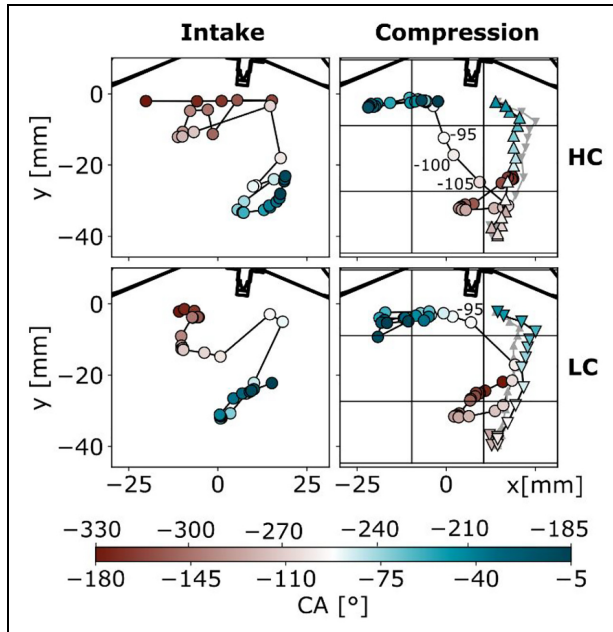


Figure 9. Feature analysis: class-depended mapping of the highest attention (CAM) for HC (upper row) and LC (lower row) cycles. CAMs of 250 cycles are averaged and a threshold of 0.5 is applied. Shown are the center of mass for each time (circles, CAD color-coded). The tumble trajectory is given for the compression stroke (HC: upward pointing triangles; LC: downward pointing triangles; CAD color-coded; gray triangles: LC tumble trajectory in the HC row and vice versa). The black grid marks the regions of engineered features.

of the classifier's decisions. Figure 9 shows the temporal focus of attention results of CAM applied to 250 HC (top row) and 250 LC cycles (bottom row). For each CAD, a threshold of 0.5 is applied to the CAMs and the averaged centers of mass are calculated, which are interpreted as the attention focus for the decision-making of the DNNs. The following qualitative statements are not found to be sensitive to the threshold. At the beginning of the intake stroke, the attention focus of HC, located slightly off-center in the upper region toward the intake valves, is not stable but jumps around. After about -230°CA , a counterclockwise attention trajectory in the mid-bottom-right region of the FOV emerges. During the compression stroke, a more stable pattern emerges. As the FOV shrinks during compression, the attention shifts rapidly toward positions in the top part of the FOV close to the spark plug. While similar regions are of importance, attention shifts of LC (Figure 9, bottom row) during the intake are more structured and the spatio-temporal trajectory is less fluctuates less than in HC. Clear temporal and spatial attention shifts are observed in both the intake and compression strokes. During the intake stroke, the focus of attention shifts from the intake valve downwards, toward the top-center at about -230°CA , and finally settles in the mid-bottom part of the FOV. Next, during the compression stroke, the attention

shifts counterclockwise upwards, first toward the exhaust valves, then toward positions close to the spark plug, and finally to the intake valves.

The tumble trajectories in Figure 9 (signified by colored triangles, gray represents LC in the HC panel and vice versa) both show a tendency of movement from the mid-bottom of the cylinder upwards toward the exhaust valves, then finally toward the spark plug as the piston reaches TDC. At a first glance, the attention movement for HC and LC generally resembles the tumble center trajectories, if the initial downwards movement from around -180°CA to -130°CA is disregarded, since this is induced by the limited FOV and the tumble center visibility. In addition to appearing more leftward (negative x-direction), the positions of the tumble centers for HC also appear closer to the top of the cylinder earlier than cycles of LC. However, after closer inspection, it is clear that the CAM attention centers actually move to higher positions near the spark plug and intake valves much earlier than the tumble centers. This very quick movement to the late flame propagation region (section 8 and 9, Figure 6(a)) during early compression indicates that the flow directions and magnitudes in this region alone may be sufficient for the prediction of HC using the CAE. Since the flow structures in this flame propagation region are tied to the location and strength of the tumble flow, these regions correlate with and are able to predict information about the entire flow field. While the conventional ML model using engineered flow data can only capture local information on a section-by-section basis, the CAE takes the entire flow field into consideration. This is why the feature importance trajectories in Figure 6 vary significantly to the CAM trajectories in Figure 9.

Conditional flow fields. To further conceptualize the CAMs, average 2D CAMs of the HC class are shown in the left column of Figure 10 as well as the conditional averaged flow fields (represented by arrow directions and lengths) of 80 HC cycles. Therefore, all HC cycles of a single experiment with P_{max} values at least 1 bar above the class boundary are selected. The CAMs reveal a broad attention area of the DNN and clear patterns underlying the decision of the DNN are not easily visible when comparing the conditional CAMs and visible average flow features. However, the decision process of the applied ML techniques can be further elucidated by analyzing separate aspect of the conditional (HC: 80 cycles; LC: 79 cycles) averaged flow fields. The middle and right columns of Figure 10 show the differences in magnitude (middle column) and the x-component velocity v_x (right column) for select CADs. Average flows between the two classes of cycles differ as revealed by the overlaid vector arrows (white: HC, black: LC) and the difference in flow magnitude. This analysis reveals patterns that can be used for the differentiation between cycles of high and low P_{max} , while an analysis based solely on instantaneous velocity

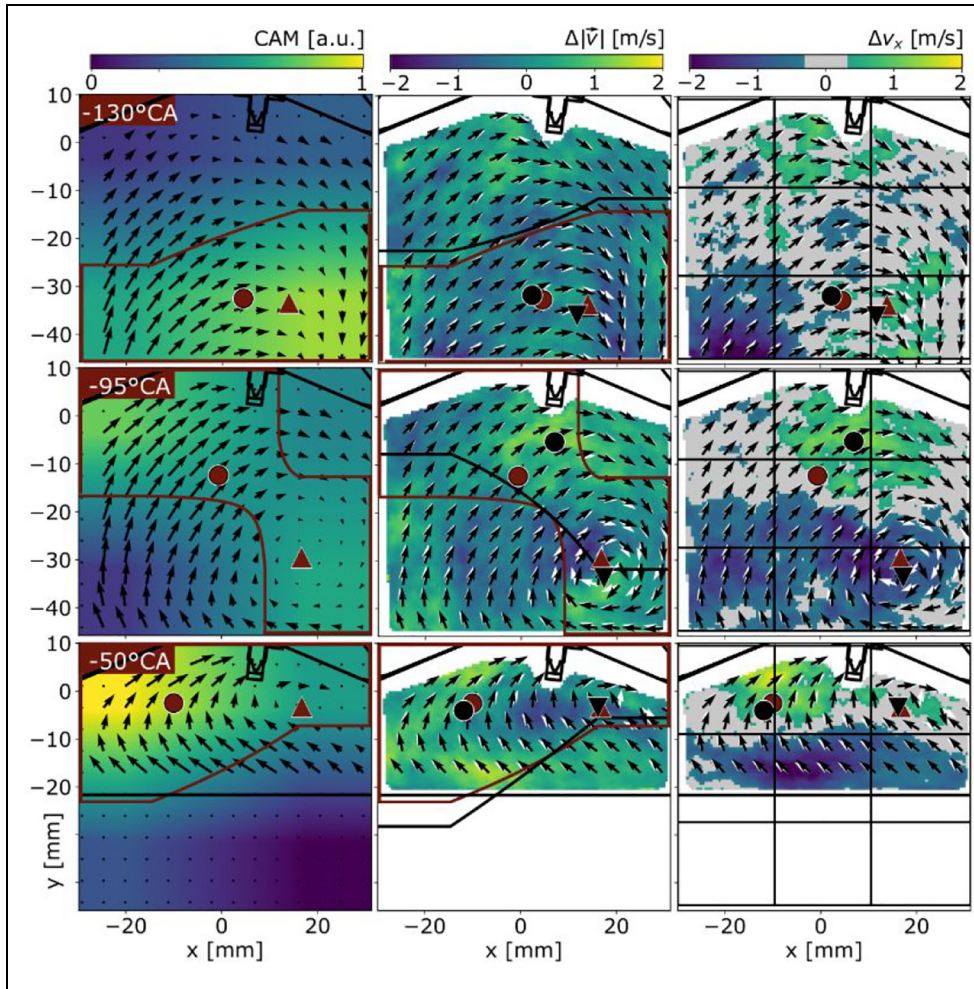


Figure 10. Difference between conditional average flow fields: Left column: CAM for the HC class and their respective 0.5 threshold (deep red contours), center of mass (circles), and tumble center (upward triangles) at -130 , -95 , and -50°CA (rows). The black arrows represent the flow direction and magnitude. Middle column: color-coded average magnitude difference between HC (white arrows) and LC (black) flow fields at the same CADs as in the left column. Positions of HC (deep red; upward triangle) and LC (black; downward triangle) tumble centers are marked. The CAMs' center of mass are again marked by circles and the CAMs' contour lines at 0.5 are given by solid lines (HC: deep red, LC: black). Right column: Color-coded average v_x difference between HC and LC. Symbols and their colors are the same as in the left columns. The black grid marks the regions of engineered features.

magnitude flow fields (e.g. Figure 7, top row) is challenging.

Deep feature learning can directly exploit the average differences between HC and LC flow fields in magnitude and flow direction, for example, by learning features representing large patches of higher or lower velocities and regions of different flow directions. Furthermore, by observing the conditional differences, it becomes obvious why ML models based on simple and spatially coarse engineered features are able to achieve high prediction accuracies. Regions of consistent differences of the conditional average flow fields are large and span one or multiple sections of the engineered features shown in Figure 10 (black grid; right column). Hence, averages as well as maximum and minimum flow field quantities have potentially high predictive power even in the case of noisy instantaneous flow fields.

A detailed analysis reveals the link between the important engineered features and observed differences

in the conditional averages. To this end, the contours of the 0.5 threshold of the CAMs (deep red: HC, black: LC) as well as their centers of mass, the calculated tumble centers and sections of engineered features are shown in Figure 10. Beginning from -130°CA , it is clear that the tumble center is outside of the FOV and not identified correctly by the algorithmic detection. Locations of the center of mass of the CAMs and the tumble centers for HC and LC are therefore only similar by chance. But the magnitude flow field reveals a sporadic spread of velocity differences throughout as well as pronounced vector angle differences near the calculated tumble center, the intake side bottom of the FOV, and near the spark plug. Evaluating the average feature importance of the AdaBoost classifier (not shown), v_x -related features are superior to v_y - and magnitude-related features in the late compression stroke. For this reason, it is helpful to analyze the difference of the x-component velocity Δv_x (Figure 10;

right column), where one can observe bigger cohesive regions of absolute differences of up to -20% . At -130°CA , Δv_x matches well with the sectional feature importance (Figure 6 and overlaid sections in Figure 10 right column), where section 3 (bottom-left) is the most important in predicting HC. Further in the compression stroke at -95°CA , section 3 (bottom-left), 7 (top-right), and 1 (bottom-right) are deemed important by the AdaBoost classifier and show notable difference in v_x for HC and LC. Yet, the CAMs at -95°CA differ in their morphology. From -105°CA to -95°CA the center of mass of HC's CAM transitions diagonally across the FOV with section 1 (bottom right) losing and 9 (top right) gaining attention. This indicates that during this period, two regions with high attention are present. This manifests in an intermediate center of mass and explains the difference in the center of mass trajectory between LC and HC (see Figure 9), since such a transition is not observed in the case of LC, where only one region of importance is present. Finally, at -50°CA , the highest feature importance of sections 4 (middle-right), 5 (middle-center), and 8 (top-center) also correspond well to the regions of large absolute Δv_x . These regions are also encompassed by the attention of the CAMs, though these are more focused toward the top-left of the FOV. The high importance of v_x -components is in line with previous findings, which related a high P_{\max} to the horizontal position of the early flame kernel.²⁷

The differences of the velocity magnitude (Figure 10; middle column) are more difficult to interpret since they take into account the y-component of the velocity, which shows a more granular distribution and may be less indicative in the cyclic prediction since the flame mainly propagates in the x-direction in a 2D plane. Since v_x -based features have high predictive power, the feature importance of magnitude- and v_y -based features is low in the AdaBoost classifier. On the other hand, as previously mentioned, the decision of the deep feature learning-based model takes into account the flow magnitude and direction as the whole field, instead of component-wise parts as defined in the engineered feature set.

Overall, a qualitative correlation between differences in the conditional flow fields and the sectional importance seems to be present for most of the compression stroke, but is harder to deduce for the CAMs of the DNN. Still, the conditional average flow field analysis might further help to explain the differences in prediction accuracy and the difference between the center of mass trajectory of the CAM to the spatio-temporal feature importance trajectory of the AdaBoost classifier. The input to the DNN has high dimensionality ($133 \times 100 \times 3$; the entire flow field at full resolution, including local information about both x- and y-components) and separated into normalized direction and magnitude information, while the engineered features already include unnormalized v_x information averaged over large areas which turned out to be useful for classifying HC based on the conditional averaged flow field analysis. The deep learning approaches lack these direct

features and might be hindered by the unimportant features, such as the y-component of the velocity, to find an optimal solution given the limited training set. While a sectional analysis of ML is much more spatially coarse, the resulting breakdown on a section and component basis yields slightly higher prediction accuracy than the complex CAE model. Finally, CAE models allow resolving fine structures of the flow which seem to be less informative for the given task, maybe due to a lack of robustness.

Conclusion and outlook

To our knowledge, this study is the first to have successfully applied self-supervised deep feature learning of in-cylinder flow fields during the intake and compression stroke of IC engine operation. These learned features are used in an ML approach for binary classification of cycles with high and low maximum pressure to investigate CCVs. Features learned with CAEs from compression stroke data generalize to the intake stroke and allow the prediction of HC already during the early intake stroke at -290°CA with a mean accuracy above chance level. From then on, the mean accuracy is always at least one std above chance level with a few exceptions. This indicates a distinct deterministic flow development from the intake to the late compression flow near the spark plug, which influences the combustion and enables future engine optimizations. By analyzing the encoder of the CAE, the focus of spatio-temporal attention is investigated with high resolution. Revealed attention-shifts differ for HC and LC during the compression stroke, where LC attention and tumble trajectories are observed to be closer to the cylinder wall. We show that models based on learned features achieve the same qualitative and comparable quantitative results as models based on hand-engineered features with coarse spatial resolution.¹⁹ Deep neural networks require larger data sets for training, more resources, for example computing time and power, and their learned features are harder to explain compared to interpretable models using engineered features. Given that the deep feature learning produces similar but not superior prediction accuracies for the investigated application, latter approach may be preferred in practice.^{42,43}

Conditional average flow field analysis reveals CAD-specific large spatial regions of differences between HC and LC flow fields, which can be exploited for CCV classification of individual cycles. These differences are well-captured by the engineered features, allowing the ML models high prediction accuracies. The spatio-temporal feature importance calculated by these models corresponds with the difference of the x-velocity component Δv_x . In the future, it might be beneficial to explore differently structured spatial feature regions and additional engineered features such as histograms of oriented gradients to further improve the accuracies. Our analysis suggests that a classification

approach is favorable to a regression approach to investigate CCVs, since classifiers are strongest in detecting outliers far from the class boundary. In addition, combining engineered flame and flow features allows models with slightly higher mean accuracies in the later compression stroke, but such models might require tedious manual verification of the automatically extracted flame contours.

The trained CAEs are able to encode and decode input flow fields with small reconstruction errors. Spatial-temporal fine structures might be relevant for high CAE performance. However, they might also hinder the learning of abstract or unspecific features, which may be important for the downstream task of investigating CCVs. Learning to encode regular patterns in flow fields with high precision does not increase the performance of classifiers if these patterns themselves do not have predictive power. Alternative DNN architectures preserving spatial information (e.g., Ronneberger et al.⁴⁴) or other pretext tasks for deep self-supervised learning of flow field features might be explored in the future.⁴⁵ Examples of the latter are generation-based methods (e.g., variational AE),⁴⁶ generative adversarial networks,⁴⁷ image inpainting,⁴⁸ and image super resolution,⁴⁹ context-based methods (e.g. solving of a jigsaw puzzle),⁵⁰ free semantic label-based methods (e.g., semantic segmentation)⁵¹ or contour detection,⁵² and cross modal-based methods (e.g., optical flow estimation).⁵³ In general, larger consistent data sets result in better ML models and reduce over-fitting.⁵⁴ Due to the fixed recording angle and flow orientation of the current data set, conventional data augmentation techniques cannot be applied to increase training variability. Specific flow field-related augmentations are difficult to define and might introduce artifacts in the artificially augmented samples.

Techniques to visualize the attention of deep neural networks allow the investigation of the spatio-temporal importance focus during intake and compression stroke with high spatial resolution.^{38,41} While the CAM centers of mass and tumble centers during the late intake stroke seem to have similar trajectories in that they begin in the bottom-right section of the FOV and move upward along the y-axis into the near-spark plug region, they have different meanings. As soon as the tumble center begins to take prominent form, the CAM center of mass shifts away and remains in the flame propagation region because other features like v_x are more informative than the tumble center to classify HC and LC. This finding is consistent with previous results, where simple flow field features are superior to derived, high-level tumble features.¹⁹

ML methods are powerful in exploring high dimensional data. An extension of our approach to 3D flow fields with time-dependent features is possible in the future (e.g., scanning PIV, tomographic PIV, and LES/DNS), despite the immense challenges and required resources associated with obtaining an experimental or simulation data set large enough for such an analysis.

To this end, a CAE with 3D convolutions and recurrent neural network could be used and alternative pretext tasks for deep self-supervised learning could be applied. Such a feature learning approach has the potential to identify deterministic high-level flow components that appear at any point in a cycle which may look stochastic and are hard or impossible to find using conventional analysis in order to predict the cycle's performance. Furthermore, the presented method of flow field analysis is not limited to only in-cylinder IC engine data. For example, deep feature learning might reveal high-level flow features of coolants to reduce the complexity of heat transfer problems.

The presented results of ML models with learned and hand-engineered features are qualitatively similar to previous studies using smaller sample sizes^{19,27} and show very high precision (low std of the accuracy). Applying our approach of combining deep learning and conventional flow field analysis not only to the 2D cross-section of the mid-cylinder plane, but also to the whole flow field of the 3D cylinder volume could ultimately reveal unknown high-level flow features relevant to the prediction of engine CCV and engine optimization in the future.

Acknowledgement

Daniel Dreher and Alexander Hanuschkin would like to thank Carola Krug for discussing the learning of flow field features with generative adversarial networks.


Declaration of conflicting interests


The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.


Funding


The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: Deutsche Forschungsgemeinschaft through FOR 2687 "Cyclic variations in highly optimized spark-ignition engines: experiment and simulation of a multiscale causal chain" – project number 423224402 – is kindly acknowledged by the authors from TU Darmstadt.

ORCID iDs

Marius Schmidt  <https://orcid.org/0000-0002-5424-1251>

Cooper Welch  <https://orcid.org/0000-0001-9067-9405>

Steven Peters  <https://orcid.org/0000-0003-3131-1664>

Alexander Hanuschkin  <https://orcid.org/0000-0001-9643-8987>

References

1. Reitz RD, Ogawa H, Payri R, et al. IJER editorial: The future of the internal combustion engine. *Int J Eng Res* 2020; 21(1): 3–10.

2. Baum E, Peterson B, Böhm B and Dreizler A. On the validation of LES applied to internal combustion engine flows: part 1: comprehensive experimental database. *Flow Turbul Combust* 2014; 92: 269–297.
3. Geschwindner C, Kranz P, Welch C, et al. Analysis of the interaction of Spray G and in-cylinder flow in two optical engines for late gasoline direct injection. *Int J Engine Res* 2020; 21(1): 169–184.
4. Matsuda M, Yokomori T, Shimura M, Minamoto Y, Tanahashi M and Iida N. Development of cycle-to-cycle variation of the tumble flow motion in a cylinder of a spark ignition internal combustion engine with Miller cycle. *Int J Eng Res*. 16 April 2020. DOI: 10.1177/1468087420912136.
5. Jainski C, Lu L, Dreizler A and Sick V. High-speed micro particle image velocimetry studies of boundary-layer flows in a direct-injection engine. *Int J Eng Res* 2013; 14: 247–259.
6. Renaud A, Ding C-P, Jakirlic S, Dreizler A and Böhma B. Experimental characterization of the velocity boundary layer in a motored IC engine. *Int J Heat Fluid Flow* 2018; 71: 366–377.
7. Schmidt M, Ding C-P, Peterson B, Dreizler A and Böhm B. Near-wall flame and flow measurements in an optically accessible SI engine. *Flow Turbulence Combust*. Epub ahead of print 21 August 2020. DOI: 10.1007/s10494-020-00147-9.
8. Ding C-P, Vuilleumier D, Kim N, Reuss DL, Sjöberg M and Böhm B. Effect of engine conditions and injection timing on piston-top fuel films for stratified direct-injection spark-ignition operation using E30. *Int J Eng Res* 2020; 21(2): 302–318.
9. Peterson B, Baum E, Böhm B and Dreizler A. Early flame propagation in a spark-ignition engine measured with quasi 4D-diagnostics. *Proc Combust Inst* 2015; 35(3): 3829–3837.
10. Krüger C, Schorr J, Nicollet F, Bode J, Dreizler A and Böhm B. Cause-and-effect chain from flow and spray to heat release during lean gasoline combustion operation using conditional statistics. *Int J Eng Res* 2017; 18(1–2):143–154.
11. Stiehl R, Bode J, Schorr J, Krüger C, Dreizler A and Böhm B. Influence of intake geometry variations on in-cylinder flow and flow–spray interactions in a stratified direct-injection sparkignition engine captured by time-resolved particle image velocimetry. *Int J Eng Res* 2016; 17(9): 983–997.
12. Bizon K, Continillo G, Leistner KC, Mancaruso E and Vaglieco BM. POD-based analysis of cycle-to-cycle variations in an optically accessible diesel engine. *Proc Combust Inst* 2009; 32(2): 2809–2816.
13. Fogleman M, Lumley J, Rempfer D and Haworth D. Application of the proper orthogonal decomposition to datasets of internal combustion engine flows. *J Turbul* 2004; 5(23): 1–3.
14. Roudnitzky S, Druault P and Guibert P. Proper orthogonal decomposition of in-cylinder engine flow into mean component, coherent structures and random Gaussian fluctuations. *J Turbul* 2006; 7: N70.
15. Graftieux L, Michard M and Grosjean N. Combining PIV, POD and vortex identification algorithms for the study of unsteady turbulent swirling flows. *Meas Sci Technol* 2001; 12(9): 1422.
16. Stiehl R, Schorr J, Krüger C, Dreizler A and Böhm B. In-cylinder flow and fuel spray interactions in a stratified spray-guided gasoline engine investigated by high-speed laser imaging techniques. *Flow Turbul Combust* 2013; 91(3): 431–450.
17. Truffin K, Angelberger C, Richard S and Pera C. Using large-eddy simulation and multivariate analysis to understand the sources of combustion cyclic variability in a spark-ignition engine. *Combust Flame* 2015; 162(12): 4371–4390.
18. Kodavasal J, Moiz AA, Ameen M and Som S. Using machine learning to analyze factors determining cycle-to-cycle variation in a spark-ignited gasoline engine. *J Energy Resour Technol* 2018; 140(10): 102204.
19. Hanuschkin A, Schober S, Bode J, et al. Machine learning based analysis of in-cylinder flow fields to predict Combustion Engine Performance. *Int J Eng Res*. Epub ahead of print 14 March 2019. DOI: 10.1177/1468087419833269.
20. DiMauro A, Chen H and Sick V. Neural network prediction of cycle-to-cycle power variability in a spark-ignited internal combustion engine. *Proc Combust Inst* 2019; 37(4): 4937–4944.
21. McKinney S, Sieniek M, Godbole V, et al. International evaluation of an AI system for breast cancer screening. *Nature* 2020; 577: 89–94.
22. Bengio Y, Courville A and Vincent P. Representation learning: a review and new perspectives. *IEEE Trans Pattern Anal Mach Intell* 2012; 35: 1798–1828.
23. Zhong G, Ling X and L.-N. W., 2018. From Shallow Feature Learning to Deep Learning: Benefits from the Width and Depth of Deep Architectures. *Wiley Interdiscip Rev Data Min Knowl Discov* 2018; 9: e1255.
24. Kramer MA. Nonlinear principal component analysis using autoassociative neural networks. *AIChE J* 1991; 37(2): 233–243.
25. Tschannen M, Bachem O and Lucic M. Recent advances in autoencoder-based representation learning. arXiv: 1812.05069 [cs.LG], 2018.
26. Akintayo A, Lore K, Soumalya S and Sarkar S. Prognostics of combustion instabilities from hi-speed flame video using a deep convolutional selective autoencoder. *Int J Progn Health Manag* 2016; 7(23): 1–14.
27. Hanuschkin A, Zündorf S, Schmidt M, et al. Investigation of cycle-to-cycle variations in a spark-ignition engine based on a machine learning analysis of the early flame kernel. *Proc Combust Inst*. 21 August 2020. DOI: 10.1016/j.proci.2020.05.030.
28. Keogh E and Mueen A. Curse of dimensionality. In: Sammut C and Webb G (eds) *Encyclopedia of machine learning and data mining*. New York: Springer, 2017, pp.314–315.
29. Freudenhammer D, Peterson B, Ding C-P and Böhm B. The influence of cylinder head geometry variations on the volumetric intake flow captured by magnetic resonance velocimetry. *SAE Int J Eng* 2015; 8(4): 1826–1836.
30. Hornik K, Stinchcombe M and White H. Multilayer feed-forward networks are universal approximators. *Neural Netw* 1989; 2: 359–366.
31. Cybenko G. Approximation by superpositions of a sigmoidal function. *Math Control Signals Syst* 1989; 2(4): 303–314.

32. Breiman L, Friedman JH, Olshen RA and Stone CJ. *Classification and regression trees*. Monterey, CA: Wadsworth & Brooks/Cole Advanced Books & Software, 1984.
33. Hochreiter S. *Untersuchungen zu dynamischen neuronalen Netzen*. Munich: Technische Univ, Institut f. Informatik, 1991.
34. Hochreiter S, Bengio Y, Frasconi P and Schmidhuber J. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In: Kremer SC and Kolen JF (eds) *A field guide to dynamical recurrent neural networks*. IEEE Press, 2001.
35. Bengio Y and LeCun Y. Scaling learning algorithms towards AI. In: Bottou L, Chapelle O, DeCoste D and Weston J (eds) *Large-scale kernel machines*. Cambridge, MA: The MIT Press, 2007, pp.321–360.
36. Simonyan K and Zisserman A. Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556, 2014.
37. Hahnloser RHR, Sarpeshkar R, Mahowald MA, Douglas RJ and Sebastian Seung H. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature* 2000; 405: 947.
38. Selvaraju RR, Cogswell M, Das A, Vedantam R, Parikh D and Batra D. GradCAM: visual explanations from deep networks via gradient-based localization. *Int J Comput Vis* 2019; 128(2): 1573–1405.
39. Pedregosa F, Varoquaux G, Gramfort A, et al., 2011. Scikit-learn: machine learning in python. *J Mach Learn Res* 2011; 12: 2825–2830.
40. Kotikalapudi R. Keras-vis. GitHub, 2017, <https://github.com/raghakot/keras-vis>
41. Montavon G, Samek W and Müller K-R. Methods for interpreting and understanding deep neural networks. *Digit Signal Process* 2018; 73: 1–15.
42. Rudin C. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nat Mach Intell* 2019; 1: 206–215.
43. Rudin C and Carlson D. The secrets of machine learning: ten things you wish you had known earlier to be more effective at data analysis. arXiv:1906.01998 [cs.LG], 2019.
44. Ronneberger O, Fischer P and Brox T. U-Net: convolutional networks for biomedical image segmentation. *Med Image Comput Comput Assist Interv* 2015; 9351: 234–241.
45. Jing L and Tian Y. Self-supervised visual feature learning with deep neural networks: a survey. arXiv:1902.06162, 2019.
46. Kingma DP and Welling M. Auto-encoding variational bayes. arXiv:1312.6114 [stat.ML], 2013.
47. Donahue J, Krähenbühl P and Darrell T. Adversarial feature learning. arXiv:1605.09782 [cs.LG], 2016.
48. Pathak D, et al. Context encoders: feature learning by inpainting. arXiv:1604.07379 [cs.CV], 2016.
49. Ledig C. Photo-realistic single image super-resolution using a generative adversarial network. arXiv:1609.04802 [cs.CV], 2016.
50. Wei C, Xie L, Ren X, et al. Iterative reorganization with weak spatial constraints: solving arbitrary jigsaw puzzles for unsupervised representation learning. arXiv:1812.00329 [cs.CV], 2018.
51. Pathak D, et al. Learning features by watching objects move. arXiv:1612.06370 [cs.CV], 2016.
52. Li Y, Paluri M, Rehg JM and Dollár P. Unsupervised learning of edges. arXiv:1511.04166 [cs.CV], 2015.
53. Ilg E, et al. FlowNet 2.0: evolution of optical flow estimation with deep networks. arXiv:1612.01925 [cs.CV], 2016.
54. Studer S, Bui B, Drescher C, Hanuschkin A., Winkler L, Peters S and Mueller KR. Towards CRISP-ML(Q): a machine learning process model with quality assurance methodology. arXiv:2003.05155, 2020.
55. Das A and Rad P. Opportunities and challenges in explainable artificial intelligence (XAI): a survey. arXiv:2006.11371, 2020.
56. Freund Y and Schapire RE. A decision-theoretic generalization of on-line learning and an application to boosting. *J Comput Syst Sci* 1997; 55(1): 119–139.
57. Rosenblatt F. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychol Rev* 1958; 65: 386–408.
58. Rumelhart D, Hinton G and Williams R. Learning representations by back-propagating errors. *Nature* 1986; 323: 533–536.
59. Bengio Y. Learning deep architectures for AI. *Found Trends Mach Learn* 2009; 2(1): 1–127.
60. Fukushima K and Miyake S. Neocognitron: a new algorithm for pattern recognition tolerant of deformations and shifts in position. *Pattern Recognit* 1982; 15(6): 455–469.
61. Krizhevsky A, Sutskever I and Hinton GE. ImageNet classification with deep convolutional neural networks. *Commun ACM* 2017; 60(6): 84–90.
62. Ciresan D, Meier U and Schmidhuber J. Multi-column deep neural networks for image classification. In: *Proceedings of the 2012 IEEE conference on computer vision and pattern recognition*, Providence, RI, 16–21 June 2012. New York: IEEE.
63. Hotelling H. Analysis of a complex of statistical variables into principal components. *J Educ Psychol* 1933; 24(6): 417–441.

Appendix

Introduction to machine learning and terminology

Machine learning describes the method of applying algorithms to learn a mapping from input to output. A machine learning model is trained for this mapping. Mapping an input by the machine learning model is called model inference or prediction. If the output is continuous, a regression model is trained while for discrete output a classification model is trained. The task is called supervised learning, if for each input the corresponding output value, called label, is given during training. The task is called self-supervised learning if for each input the corresponding output label is not given explicitly, but if such a label can be generated by the machine learning algorithm itself. For example, autoencoders (AEs) are trained by mapping the input to itself, that is, each input serves as its own label. Each training example, simply called input sample, consists of a set of variables called features or covariates. Inputs can either be engineered features, that is, manually designed and selected depending on the task and based on domain knowledge, or raw features, for example, an array of image pixel values.

The more features needed to represent an input sample, the higher the feature space dimension, that is, the space spanned by the feature axes. The number of samples needed to sufficiently cover the feature space increases exponentially with the number of dimensions. Hence, reducing the feature space, known as a feature selection, can lead to higher model accuracies. For example, highly inter-feature-correlated features can be removed. Alternatively, principle component analysis (PCA)⁶³ can be applied, projecting the feature space to a lower dimensional orthogonal space, where the new basis set is derived from the features' covariance matrix. By this, correlated features are projected onto a new common axis and individual dimensions of the transformed data are linearly uncorrelated. Depending on the data set and machine learning algorithm, further data pre-processing methods are recommended to improve the model performance. For neural networks, the input features are commonly mean subtracted and scaled to unit variance. This procedure is also called standard scaling. Another way of limiting the features' values between negative one and positive one is feature normalization, where individual values are divided by the maximal absolute value of this feature encountered in the data set.

Machine learning models have to be trained on a training set and their performances have to be evaluated on a separate test set. Thereby, training and test sets are disjoint subsets of the available data set. This is important because a good model should generalize. Over-fitting occurs if a trained model is too attuned to the training set and therefore unable to generalize on the test set. For classification, the model's accuracy is given by the number of correctly classified test samples divided by the total number of test samples; but a variety of different specific performance measures exists. The training set can be further split into a training and validation set, where the latter is used to evaluate the model performance during training. A common method to automatically generate multiple training and validation sets is cross-validation (CV). The training set is split into k -folds and for each instance $k - 1$ splits are used for training and the remaining split for validation. In total, k different training and validation sets result from such a CV. Cross-validation can be used for hyper-parameter optimization, where the machine learning algorithm with different hyper-parameter settings train on the training set and its performance is evaluated on the validation set. Hyper-parameters are parameters of the algorithm, describing the machine learning model and how it is trained; for example the maximum number of training iterations. The CV allows a statistical evaluation of the trained machine learning model performance.

Machine learning algorithms optimize parameters of a machine learning model to minimize a cost function, for example, during training the algorithm minimizes the summed losses or errors that the model produces by mapping the input to the output. Therefore, a loss

function of a sample's prediction and label is applied measuring a penalty. A common cost function is the mean squared error (MSE), where the cost is given by the average squared difference $\frac{1}{n} \sum_i^n (y_i - \hat{y}_i)^2$ between predicted output \hat{y}_i and label y_i . For binary classification tasks, that is, where only two different output class labels are present, binary cross entropy can be used, where the loss is given by $-(y \log(p) + (1 - y) \log(1 - p))$ for p being the probability of predicting one of the classes and y the corresponding label in $\{0,1\}$. Different machine learning algorithms and consequently different machine learning models exist for binary classification. The algorithms can be linear (e.g., linear regression or logistic regression), non-linear (e.g., classification and regression trees (CART)³²), or ensemble methods (e.g., bagging and boosting).

Inherently interpretable models, which are loosely defined as providing their own explanations of the model's mapping,⁴² allow the understanding of the models' mapping functions. For example, a decision tree model can be directly transferred to a set of distinct rules. Deep neural networks are not inherently interpretable and are sometimes considered black box models because humans cannot understand the complicated and non-linear mapping directly. Current research on explainable artificial intelligence (XAI), where machine learning is a subfield of AI, focuses on deriving explanations of the black box models' inferences.⁵⁵ For example, class dependent activation mapping (CAM) is an explanation method for deep neural networks revealing spatial input regions of interest that are important for a specific classification task.³⁸ Additionally, the feature importance can be evaluated. Feature importance is a measure of how important an individual feature is for the performance of the model for a given training set. From some machine learning models it can be derived directly. The feature importance of a decision tree can be calculated from the node importance,³² for example, taking into account the Gini impurity (depending on the tree model) weighted by the normalized probability (derived from the training samples) of reaching the node, of the node itself, and its child nodes. Conceptually in this case, feature importance is derived from the training sample distribution and the model's shape, which in turn is also determined by the training sample distribution and the specific hyper-parameter setting. Indirect and iterative methods like LOCO (leave-one-covariate-out) can be applied if the feature importance cannot be derived directly.

A binary decision tree model essentially asks a set of "if - else" questions until a leaf node with an associated (class) value is reached. Different machine learning algorithms exist for decision tree, such as classification and regression tree (CART)³² or iterative dichotomizer 3 (ID3), defining the splitting rules at each node based on certain criteria like Gini impurity (CART) or information gain (ID3). Decision trees are interpretable, since they provide simple rules which are easy to

understand and visualize. An ensemble of weak decision tree classifiers can create a strong classifier, commonly referred to as a boosted decision tree. Thereby, a single decision tree is trained and the succeeding tree is trained to correct for its errors, for example, by fitting the second tree to a weighted version of the original data set. To obtain a strong classifier, additional decision trees are added in the same way. AdaBoost⁵⁶ is a boosting method which performs very well on binary classification problems by using decision trees with only one level, which are also called decision stumps. The quality (stage value) of each weak learner is calculated from the weighted sum of the misclassification rate, where training samples have a weight and the misclassification rate is given by the ratio between incorrectly classified and all training samples. The sample weights of misclassified samples are changed depending on the stage value. For inference, the predictions of each tree are summed and weighted by the stage values of each tree.⁵⁶

Multi-layer perceptrons (MLPs)⁵⁷ are feed-forward connected networks of artificial neurons (units). This type of neural network consists of an input layer, an output layer, and at least one hidden layer, where the layers are fully connected, that is, all neurons of one layer are connected to all neurons of the next layer, and non-linear activation functions are applied on the summed and weighted input for each neuron. The weights are optimized using the backpropagation algorithm.⁵⁸ For classification, the neurons in the output layer represent the different classes and a softmax function can be applied. With a sufficient amount of hidden neurons, MLPs have been proven to be universal approximators^{30,31}

A key feature of deep learning is the ability to learn how to form feature hierarchies by combining low-level features of shallow layers to increasingly more abstract high-level features in deeper layers⁵⁹ of a neural network. While MLPs are capable of learning abstract features, their reliance on fully connected layers leads to a high number of trainable parameters, which in turn renders the models harder to train and prone to over-fitting. Additionally, MLPs require a flattened feature vector as their input and therefore rely on a global view of multidimensional data, possibly leading to redundant features, to handle spatial variances in the input data. Convolutional neural networks (CNNs) alleviate these problems to some degree by learning how to detect local features while at the same time reducing the number of trainable parameters by weight sharing. Their design was inspired by the biological visual cortex. While first usage of CNNs can be traced back to the 1980s,⁶⁰ recent advances in computer hardware and increased availability of large data sets allowed training CNNs with groundbreaking predictive power.⁶¹ For the implementation of CNNs two new network layer types were introduced: convolution layers and pooling layers. The convolution layer is named after the eponymous mathematical operation. Each convolution layer

consists of a customizable set of filters (kernels) with fixed dimensions convolved over the input volume. Sliding the filters over the width and height of the input volume results in two-dimensional feature maps which maintain spatial information about the filters' activations. Each filter has its own set of weights, which are shared for all possible spatial positions of the filter. This in turn drastically reduces the number of trainable parameters in the neural network. Pooling layers are used for downsampling the feature maps. The stride describes how many elements the filter is moved when it is slid over the input volume. The padding describes the convolution layer's behavior when the input volume's boundaries are reached. The input can be artificially padded to allow the filters to convolve the whole input volume (same padding) or the convolution operation can be constrained to remain inside the input volume's bounds (valid padding). The values used for the padding is customizable. A common way to pad edges is to fill additional space with constant values, for example, zeroes (zero padding).

Class boundary definition

Due to the increasing thermal load during the course of one experimental run of 350 consecutive fired cycles, a single dynamic class boundary, the moving average P_{\max} value (see Methods), is used. Thereby, the class boundary is given by the moving average P_{\max} value, for example, the n th cycle of a recorded sequence with a P_{\max} value above the moving average P_{\max} value at position n would be assigned HC. This approach compensates for the experimentally observed drift in moving average P_{\max} value and allows the use of the entire recorded data set. The latter is a crucial requirement for deep learning methods requiring substantially large data sets for training.

Alternatively, a fixed P_{\max} value can be used for the class boundary definition, if the data set is constrained to a smaller range of allowed mean P_{\max} values.²⁷ However, the data set available for the ML method is substantially reduced, hindering a deep learning approach. By further defining a margin around the class boundary, removing samples with P_{\max} values within this margin around the class boundary, the data set can be further constrained to only include cycles with extremely high and low combustion energies. In Figure 5(b) the accuracy of a classifier dependent on the sample's P_{\max} value is shown. The classifier reaches perfect accuracies for either extremely high or low P_{\max} values, and hence, an ML model based on data obtained with such a class boundary margin would allow perfect classification results.

In the case of an experimental setup with higher thermal stability¹⁹ or smaller numbers of consecutive fired cycles, the median P_{\max} can be chosen for the class boundary resulting in a data set with class-balanced labels.

Reduced flame feature set

By definition, numerous flame features are highly correlated, because features 9 to 11 are derived from features 1 to 8. Hence, they contain similar information or even redundant information, for example, feature 5 (bottom most point) and feature 9 (distance between bottom most point and piston). In an additional test, highly correlated features ($r > 0.95$) are neglected. These features are 9 ($r_{5,9} = 1$), 10 ($r_{5,10} = 0.95$), 11 ($r_{11,15} = 0.97$), 12 ($r_{4,12} = 0.95$) and 14 ($r_{3,14} = 0.98$). A CAD-averaged accuracy of 0.607 ± 0.036 results, which is similar to the CAD-averaged accuracy of 0.614 ± 0.039 obtained after applying PCA (see Results).

Investigated CAE architectures

A set of 7 different CAE architectures is defined (Table 3). Dense or fully-connected autoencoders (abbr.: dense) with increasing depth are investigated, serving as baseline models. Different CAE architectures are tested; Amax: convolution layer and subsequent downsampling via max pooling in the encoder and nearest-neighbor upsampling, with optional edge cropping, followed by a convolution layer for the decoder; AMaxI: as before but with increasing number of feature maps in the vicinity of the bottleneck, allowing the network to learn more high-level features in deeper layers of the encoder; AMaxIFc: as AMaxI but with an MLP substituting the convolutional layer in the bottleneck; ASI: as AMaxI but exploiting the strides for downsampling and likewise transposed convolutions for up-sampling. Additionally, residual AE (Residual) architectures are investigated, where blocks consisting of convolution layers were replaced with residual blocks. Finally, minimal models (Minimal) with a single convolution layer with large kernel size and stride with a transposed convolution in the decoder have been tested.

Different model variants are described using an adapted notation by Ciresan et al.⁶²: the first input layer for vector fields has a shape of $110 \times 98 \times 3$. 8C3S2 denotes a convolution layer with 8 feature maps, filter size of 3×3 and a stride of 2. 8TC3S2 is the notation for a transposed convolution with 8 feature maps, a filter size of 3×3 , and a stride of 2. A stride of 1 is to be

assumed, if the stride has not otherwise been specified. Padding is specified as follows: with same padding, the input is padded in a way that ensures that the filter can be moved across the complete original input; valid padding only uses valid input data, no additional padding is added to the borders. For example, if the filter would require more data to be applied on the input's edge, the operation would be dropped. The employed padding type defaults to same padding. The P(v) token indicates the usage of valid padding, for example, 16C3P(v). MP2 denotes a non-overlapping 2×2 MaxPooling2D layer and UP2 nearest-neighbor upsampling operation with size 2×2 . 8Res3 denotes a residual block skipping 2 convolution layers with 8 feature maps, filter size of 3×3 , and stride of 1. FC90 describes a fully connected layer with 90 neurons. Model names are composed of architecture name and number of neurons in the bottleneck.

In general, the validation loss (MSE) on the pretext task of the (C)AE decreases with increasing bottleneck size and corresponding decreasing compression rate. The CAE performs better than the dense AE while having significantly smaller numbers of trainable parameters in the encoder of the model. The minimal models perform worse than the CAE and dense AE while having more trainable parameters than the CAE of the same bottleneck size.

Four different architectures (Dense-90, AMaxI-96, ASI-96, Minimal-90; Table 4; see Figure 11(a)) with similar bottleneck sizes of around 90 neurons are fine-tuned and tested on the downstream task to systematically evaluate the influence of different architecture. All models reach similar CAD-averaged accuracies of 0.554 ± 0.015 (Dense-90), 0.567 ± 0.016 (AMaxI-96), 0.564 ± 0.015 (ASI-96), and 0.559 ± 0.016 (Minimal-90).

Next, different bottleneck sizes of 64, 96, 320 and 392 are fine-tuned and tested on the downstream task. Since no difference in architecture choice was observed before, a subset of 4 different architectures (AMaxIFc-64, ASI-96, AMaxI-320, Amax-392; Table 4; see Figure 11(b)) was chosen to simultaneously further sample this hyperparameter dimension. Neither larger nor smaller bottleneck sizes, with the former reducing the MSE of the AE, substantially improved the model accuracy of the downstream task: CAD-averaged accuracies 0.557 ± 0.015

Table 3. Autoencoder architectures.

Name	Feature maps	Encoder	Decoder	Bottleneck
Dense	–	–	–	Dense
Amax	Fixed	Max pooling	2D upsampling	Conv.
AMaxI	i/d	Max pooling	2D upsampling	Conv.
ASI	i/d	Conv. & stride	Transposed conv.	Conv.
AMaxIFc	i/d	Max pooling	2D upsampling	Dense
Residual	i/d	Conv. & stride	Transposed conv.	Conv.
Minimal	Fixed	Conv. & stride	Transposed conv.	Conv.

conv.: convolution; i/d: encoder: increasing with depth, decoder: decreasing with depth; the dense architecture only employs a dense layer.

Table 4. Model descriptions. MSE on the pretext task is evaluated on the validation set. Models trained and evaluated on intake and compression data.

Name	Description	pretext task (MSE)	downstream task (Accuracy)
Dense-90	FC512-FC128-FC90-FC128-FC512-FC32340	0.038	0.554 ± 0.015
Dense-765	FC1024-FC768-FC768-FC1024-FC32340	0.028	–
Dense-4096	FC4096-FC4096-FC32340	0.033	–
Dense-2048	FC2048-FC2048-FC32340	0.028	–
Amax-21560	8C3-MP2-UP2-8C3-3C3	0.002	–
Amax-5600	8C3-MP2-8C3-MP2-UP2-8C3-UP2-8C3-3C3	0.006	–
Amax-392	8C3-MP2-8C3-MP2-8C3-MP2-8C3-MP2-UP2-8C3-UP2-8C3-UP2-8C3-3C3	0.029	0.570 ± 0.016
AMaxI-11648	8C3-MP2-16C3-MP2-32C3-MP2-64C3-64C3-UP2-32C3-UP2-16C3-UP2-8C3-3C3	0.009	–
AMaxI-4096	16C3-MP2-32C3-MP2-64C3-MP2-128C3-MP2-256C3-MP2-UP2-256C3-UP2-128C3-UP2-64C3-UP2-32C3-UP2-16C3-3C3	0.011	–
AMaxI-5824	8C3-MP2-16C3-MP2-32C3-MP2-UP2-32C3-UP2-16C3-UP2-16C3-3C3	0.008	–
AMaxI-96	8C3-MP2-8C3-MP2-16C3-MP2-16C3-MP2-24C3-MP2-24C3-MP2-UP2-24C3-UP2-24C3-UP2-16C3-UP2-16C3-UP2-8C3-UP2-8C3-3C3	0.035	0.567 ± 0.016
AMaxI-320	Input-16C3-MP2-24C3-MP2-32C3-MP2-48C3-MP2-64C3-MP2-80C3-MP2-UP2-80C3-UP2-64C3-UP2-48C3-UP2-32C3-UP2-24C3-UP2-16C3-3C3	0.021	0.561 ± 0.015
AMaxIFc-64	8C3-MP2-8C3-MP2-16C3-MP2-16C3-FC64-FC2912-16C3-UP2-16C3-UP2-8C3-UP2-3C3	0.032	0.557 ± 0.015
ASI-96	8C3S2-8C3-16C3S2-16C3-24C3S2-24C3S2-24C3S2-24C3S2-24C3S2-24C3S2-16C3S2-16C3S2-8C3S2-3C3	0.029	0.564 ± 0.015
ASI-768	8C3S2-8C3-16C3S2-16C3-32C3S2-32C3S2-48C3S2-48C3-48C3-32C3S2-32C3S2-16C3S2-8C3S2-8C3S2-3C3	0.015	–
ASI-1024	8C3S2-8C3-16C3S2-16C3-32C3S2-32C3S2-64C3S2-64C3	0.013	–
ASI-2048	16C3S2-16C3-32C3S2-32C3-64C3S2-64C3S2-128C3S2-128C3	0.008	–
Residual	8Res3-8C3S2-8Res3-16C3S2-16Res3-16C3S2-16Res3-24C3S2-24Res3-24C3S2-24Res3-24C3S2-24TC3S2-24Res3-16TC3S2-16Res3-16TC3S2-16Res3-8TC3S2-8Res3-8TC3S2-8Res3-3C3	0.032	–
Minimal-90	10C(37,33)S(37,33)P(v)-3C(37,33)S(37,33)P(v)	0.056	0.559 ± 0.016

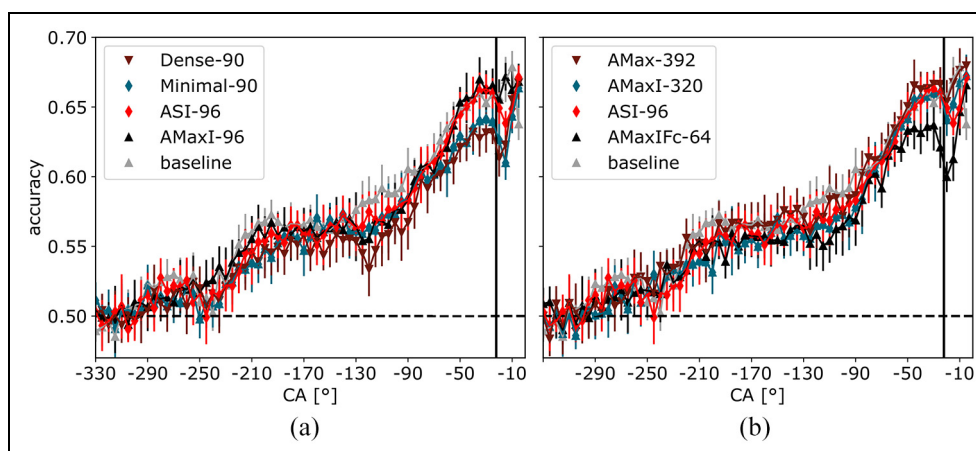


Figure 11. Comparing the performance of different autoencoder architectures for the downstream task. (a) different architectures with bottleneck sizes comparable to the baseline model with 90 features. (b) different bottleneck sizes and different architectures. The baseline model with engineered features (gray color; upwards triangles) and selected deep neural network (red color; diamond, ASI-96) are the same in both panels.

(AMaxIFc-64), 0.564 ± 0.015 (ASI-96), 0.561 ± 0.015 (AMaxI-320), and 0.570 ± 0.016 (Amax-392).

Dimension reduction with PCA is tested on AMaxIFc-64 and ASI-96 but does not improve the results: 0.558 ± 0.015 (AMaxIFc-64; 60 principle

components (PCs)) and 0.557 ± 0.016 (AMaxIFc-64; 45 PCs); 0.559 ± 0.016 (ASI-96; 90 PCs), 0.560 ± 0.016 (ASI-96; 60 PCs), and 0.560 ± 0.016 (ASI-96; 45 PCs). The baseline model with engineered features reaches 0.571 ± 0.013 .