FUSION SCIENCE AND TECHNOLOGY
© 2020 EUROFusion. Published with license by Taylor & Francis Group, LLC.
DOI: https://doi.org/10.1080/15361055.2020.1820805

Detection of Alfvén Eigenmodes on COMPASS with Generative Neural Networks

Vít Škvára,[a,b]* Václav Šmídl,[c] Tomáš Pevný,[d] Jakub Seidl,[a] Aleš Havránek,[a,d] and David Tskhakaya[a]

[a]*Czech Academy of Sciences, Institute of Plasma Physics, Za Slovankou 1782/3, Prague, Czech Republic, 18200*
[b]*Czech Technical University, Faculty of Nuclear Sciences and Physical Engineering, Břehová 7, Prague, Czech Republic, 11519*
[c]*Czech Academy of Sciences, Institute of Theory of Information and Automation, Pod Vodárenskou Věží 4, Prague, Czech Republic, 18200*
[d]*Czech Technical University, Faculty of Electrical Engineering, Technická 2, Prague, Czech Republic, 16627*

Received November 29, 2019
Accepted for Publication September 2, 2020

Abstract — *Chirping Alfvén eigenmodes were observed at the COMPASS tokamak. They are believed to be driven by runaway electrons (REs), and as such, they provide a unique opportunity to study the physics of nonlinear interaction between REs and electromagnetic instabilities, including important topics of RE mitigation and losses. On COMPASS, they can be detected from spectrograms of certain magnetic probes. So far, their detection has required much manual effort since they occur rarely. We strive to automate this process using machine learning techniques based on generative neural networks. We present two different models that are trained using a smaller, manually labeled database and a larger unlabeled database from COMPASS experiments. In a number of experiments, we demonstrate that our approach is a viable option for automated detection of rare instabilities in tokamak plasma.*

Keywords — *Tokamak, generative models, neural networks, Alfvén eigenmodes.*

Note — *Some figures may be in color only in the electronic version.*

I. INTRODUCTION

Alfvén eigenmodes (AEs) present a magnetic instability that appears during the experimental operation of tokamaks. The presence of fast energetic particles in tokamak plasmas can lead to destabilization of the shear Alfvén waves by resonance of particle velocity with Alfvén velocity of the plasma. These Alfvénic instabilities can, in turn, degrade confinement of the energetic particles and thus lower plasma performance and possibly endanger plasma-facing components of the tokamak.[1] Alfvén waves are typically excited by fast ions generated by auxiliary plasma heating or fusion reactions; however, magnetic measurements of the COMPASS tokamak[2] and DIII-D tokamak[3] have recently revealed Alfvénic modes that are driven by runaway electrons (REs). Since the presence of REs constitutes a high risk for a fusion reactor, nonlinear interaction of REs and AEs may be of high interest for topics of RE mitigation and study of their losses. Moreover, the presence of AEs in the plasma offers a diagnostic opportunity. On JET, measurement of AEs is used to compute equilibrium parameters such as the safety factor.[4]

There are multiple types of AEs with different characteristics. A specific type of AE, driven by REs and with chirping characteristics, is believed to be present at COMPASS (Refs. 2, 5, and 6), and similar chirping modes

*E-mail: skvara@ipp.cas.cz

were also observed on the DIII-D machine.[3] In COMPASS (Ref. 7), the frequency of the detected modes is in the range 0.5 to 2 MHz, scaling with the plasma density and its profile, safety factor, magnetic field, and shape of the plasma.[2] The mode has a bursty character. It typically appears after a sawtooth crash, together with increased RE losses detected by measurement of hard-X-ray radiation.[2] The frequency chirp itself takes ~1 ms, and the frequency can chirp both up and down by ~0.1 MHz. A change of the RE distribution function during the frequency chirp was directly measured in DIII-D (Ref. 3).

Up to now, spectrograms of COMPASS were labeled manually by experts. However, the rate of occurrence of chirping modes in COMPASS is relatively low (estimated to be on the order of 10^{-3} in terms of shots), and there are multiple measurement probes from which a spectrogram can be computed. Therefore, many spectrograms have to be combed through to find an experiment during which a chirping mode has occurred. This means that to collect enough experimental data for further analysis, a large amount of manual labor is required. In this work, we will try to use the available labeled spectrograms (both with and without a chirping mode) and the large unlabeled database (coming from over 15 000 COMPASS discharges) to train models that would enable automatic identification of spectrograms that contain a chirping AE.

To this end, two approaches based on generative neural networks have been implemented. Generative models based on the variational autoencoder (VAE) paradigm[8] have been used because they are powerful estimators of high-dimensional distributions, suitable for modeling image data. They do not require labels for training, which is a limiting factor for classification neural networks that overfit when not supplied with enough labeled data. Also, VAEs possess an ability to produce a low-dimensional representation of target data, which proved to be useful for our task as well.

In the following section, the basics of probabilistic autoencoders will be given. Then, the details of experimental data, implementation, and network architecture will be described together with the details concerning the experimental setup. Finally, the experimental results will be shown, and their implications will be discussed.

II. CHIRPING MODES ON THE COMPASS TOKAMAK

Chirping modes on the COMPASS tokamak can be observed indirectly in spectrograms of a magnetic U-probe.[9] The raw probe signal and its spectrogram are plotted in Fig. 1. A single spectrogram covers one experiment of average length of 0.3 s and frequency range of 0 to 2.5 MHz. The spectrograms are of uneven size because every experiment has a different length. Also, the size of

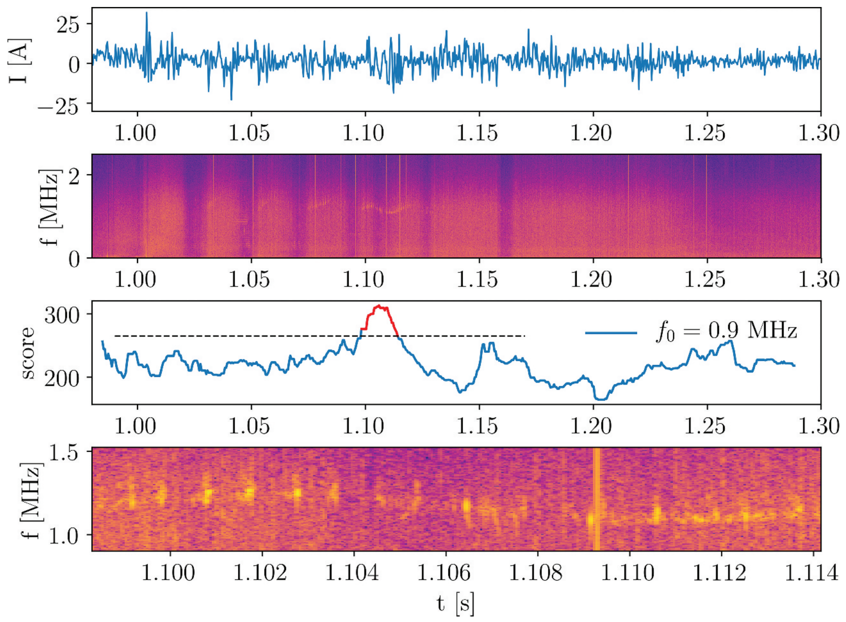

Fig. 1. COMPASS shot 10870. (a) Raw U-probe signal. (b) The corresponding spectrogram. (c) The score is the output of the proposed algorithm over the spectrogram at $f_0 = 0.9$ MHz. The highest peak around 1.1 s corresponds to a detected chirping mode. (d) A close-up of the spectrogram part containing the chirping mode as detected from the red part of the score plot. The size of the close-up is 128 × 311 pixels.

the whole spectrogram is too large for practical training of a neural network. Therefore, for labeling, training, and validation purposes, we have split spectrograms into square patches of the same size. We have chosen the size of a single spectrogram patch to be 128×128 pixels, which covers 6.54 ms in the time axis and 0.62 MHz in the frequency axis and which is a feasible input size for current convolutional neural network architectures. It is also enough to capture most of a typical chirping mode as can be seen in Fig. 1d. For all future purposes, positively labeled data are those spectrogram patches that were labeled as containing a chirping mode while negatively labeled patches do not.

III. MODEL STRUCTURE

This section contains a brief theoretical background on VAEs and their variants. This theory will be used in the construction of two types of models, i.e., one class and two stage, that differ in the way the VAE model is trained and used. In the former, the VAE log likelihood is used to detect out-of-distribution samples while in the latter, a classifier is trained on the latent space of the autoencoder.

The VAE neural network is a generative model. The purpose of a generative model is to enable sampling from a data distribution of interest $p(x)$, where $x \in \mathcal{X}$ is a sample from the data distribution. Since the data space $\mathcal{X}$ is usually high-dimensional (e.g., it represents images of high definition) and all that is available for training is a finite set of samples from $p(x)$, we cannot express it in a closed form and sample from it directly. Therefore, the problem of sampling is moved to a latent space $\mathcal{Z}$ on which we define a prior distribution $p(z)$ from which we can easily sample [e.g., $\mathcal{N}(0,1)$]. A generative model then serves the purpose of approximating the mapping $f : \mathcal{Z} \to \mathcal{X}$ such that $f(p(z)) \approx p(x)$; i.e., sampling in the latent space is equivalent to that in the data space. Also, it provides us with an approximation of $p(x)$ that can be used to test for out-of-distribution samples. In our case, $p(x)$ that is to be approximated is the distribution of the 128×128 spectrogram patches.

III.A. Generative Autoencoders

A generative autoencoder consists of two main parts, i.e., an encoder and a decoder, that are two neural networks that can be denoted as mappings $e_\phi : \mathcal{X} \to \mathcal{Z}$ and $d_\theta : \mathcal{Z} \to \mathcal{X}$, where $\{\phi, \theta\}$ are trainable parameters (weights) of the neural network. The decoder parameterizes the generative distribution $p_\theta(x|z)$, which means that it is used to compute the parameters (e.g., mean and variance) of $p_\theta(x|z)$ as a function of z. Likewise, the encoder parameterizes the encoding distribution $q_\phi(z|x)$.

The whole process of passing a sample x through the network during training is as follows: z is sampled from encoding distribution $q_\phi(z|x)$ whose parameters are obtained from $e_\phi(x)$. This is then passed to the decoder $d_\theta(z)$, which produces parameters of $p_\theta(x|z)$ from which a reconstruction $\hat{x}$ is sampled. Through this, the generative autoencoder maximizes the probability of each sample obtained through the generative process with respect to the available data:

$$p_\theta(x) = \int_{\mathcal{Z}} p_\theta(x|z)p(z)dz \ . \tag{1}$$

The general expression for the training loss that minimizes Eq. (1) can be expressed in accordance with Refs. 10 and 11 as

$$\begin{aligned}\mathcal{L}(x,\theta,\phi) = \inf_{q_\phi(z|x)} \mathbb{E}_{x\sim p(x)}\mathbb{E}_{z\sim q_\phi(z|x)}[\ln p_\theta(x|z))] \\ + \lambda\Gamma(p(z), q_\phi(z)) \ ,\end{aligned} \tag{2}$$

where

$\mathbb{E}_p[.]$ =	expected value
$\lambda > 0$ =	scaling parameter (or hyper-para-meter)
$\Gamma(.,.)$ =	divergence measure between two probability distributions
$q_\phi(z) = \mathbb{E}_{x\sim p(x)}[q_\phi(z|x)]$ =	encoding distribution marginal.

The first term forces the reconstruction $\hat{x}$ to be as close to x as possible so that the generated samples resemble the training data. The second term pushes the encoder distribution close to the prior. This has the effect that after training, one can generate new data by passing the sample from the prior (instead from the encoder) to the decoder. This generated sample will have a high probability under $p(x)$ (look realistic) if the prior and encoding distributions are indeed close.

In our models, we will use Gaussian encoding and generating distributions, that is, $q_\phi(z|x) = \mathcal{N}(z|\mu_\phi(x), \Sigma_\phi(x))$ and $p_\theta(x|z) = \mathcal{N}(x|\mu_\theta(z), \Sigma_\theta(z))$. Different choices of Γ and the resulting optimization objectives are explored here. During training, objective (2) is minimized with respect to parameters θ, ϕ using backpropagation and a standard optimization algorithm, e.g., the Adam algorithm.[14] A schematic diagram of a convolutional autoencoder is in Fig. 2.

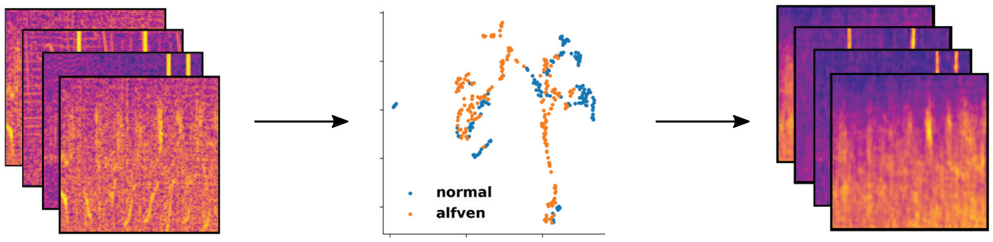

Fig. 2. A schematic diagram of the convolutional autoencoder used for our experiments. Spectrogram patches are encoded through several convolutional,[12] maxpooling,[13] and dense (fully connected) layers into d-dimensional vectors (here $d = 2$) and then decoded back with transposed convolutions and upscaling layers.

Kullback-Leibler Divergence. The use of Kullback-Leibler divergence (KLD) $D_{KL}(q_\phi(z|x)||p(z))$ results in the well-known VAE model,[8] where the prior $p(z) = \mathcal{N}(0,1)$ is used in order to obtain an analytical expression of the optimization objective.

Maximum Mean Discrepancy. The maximum mean discrepancy (MMD) divergence $\mathrm{MMD}_k(q_\phi(z|x), p(z))$ can be also used, which requires a kernel k. For details, see Ref. 11. Its optimization requires sampling from both latent distributions, which means that any prior that can be sampled from can be used, even a mixture prior, whose probability density $p(z)$ consists of several modes. In our work, we have used the Variational Mixture of Posteriors prior[15] (VampPrior), which enables optimization of the parameters of the components of the latent mixture, thus giving more flexibility to the overall model.

Jensen-Shannon Divergence. The Jensen-Shannon divergence (JSD) $D_{JS}(q_\phi(z|x)||p(z))$ is a symmetric form of KLD. Its use was again demonstrated in Ref. 11, where it was shown that it leads to the adversarial loss used in the Adversarial Autoencoder (AAE) model.[16] It requires an additional network, i.e., a discriminator $d_\eta : \mathcal{Z} \rightarrow [0,1]$ that tries to recognize between z sampled from the prior and that sampled from the encoding distribution. The output of the network is a probability that the input comes from the prior $p(z)$. The discriminator is trained in tandem with the encoder so that they both improve in their tasks and the encoder learns to map the data x so that they resemble samples from the prior. Again, here we have used VampPrior.

A form of the adversarial loss is commonly used in generative adversarial networks (GANs), and although it leads to generated samples of better quality, its use is known to destabilize the training process.[17] This holds for AAE models as well. For this reason, we have used a combination of the MMD and the GAN loss in order to stabilize the training of the encoder and the distribution it produces.

We have used a plain autoencoder[18] as a baseline, which optimizes the mean-squared error (MSE) between input x and the reconstruction $\hat{x}$:

$$\mathcal{L}_{\mathrm{AE}}(x, \theta, \phi) = ||x - d_\theta(e_\phi(x))||^2 \ . \tag{3}$$

III.B. One-Class Model

In the first model, we will use a convolutional generative autoencoder as a one-class estimator. This is an approach well known in the anomaly/outlier detection setting.[19,20] A model of choice learns a representation of one class of data and can, therefore, be used to detect out-of-class samples. It is trained either with labeled data belonging to the class of interest or with unlabeled data that are believed to contain so little out-of-class examples that the model is robust enough to ignore them.

A generative autoencoder can be readily used for this task if we set $p_\theta(x) \approx p(x)$ to be the distribution of the class of our interest. Then, there are two modes of training the autoencoder. In the first mode, we model the distribution of patches that contain a chirping mode. Then, the autoencoder is trained with the positively labeled data. However, this is a bit problematic since there are very few labeled patches available; therefore, the neural network will very likely overfit. In the second mode, we can choose the class of interest to be of the patches that do not contain a chirping mode. This is closer to an anomaly detection formulation of the problem, as the relatively rare chirping modes are considered to be anomalous. Also, the autoencoder can be trained with unlabeled data due to the sparse occurrence of chirping modes and robustness of probabilistic neural networks,[21,22] hugely increasing the number of training samples and thus the representative power of the neural network.

To decide whether a sample x is in class or out of class, we can compute its negative log likelihood under the generative distribution

$$-\mathbb{E}_{q_\phi(z|x)}[\ln p_\theta(x|z)] \tag{4}$$

or its approximation, the MSE between x and its (sampled) reconstruction $\hat{x}$. In our experiments, we use log likelihood (4) since it better captures the uncertainty in the reconstruction.

III.C. Two-Stage Model

The second model is designed to make the most use of both labeled and unlabeled data. It exploits the ability of generative autoencoders to produce a low-dimensional uncorrelated representation of high-dimensional image data. It consists of two stages. The first stage is a convolutional generative autoencoder trained with unlabeled data. Its task is to learn the general topology of the input space and encode input data. The second stage is a classifier that is trained on encoded labeled data. Through the use of MMD or JS_D measures and VampPrior, we can enforce separation of the encoded data into clusters that contain similar inputs, which makes the task of the classifier easier. Two different classifiers were tested:

1. *K-nearest neighbors (kNN)*: The kNN algorithm for classification[23] was trained using the labeled training data. In this setting, an unlabeled sample is given a score based on the average label of its kNN. The more the neighboring training samples are labeled as positive, the higher is the score.
2. *Gaussian mixture model (GMM)*: A GMM (Ref. 24) with M components was fitted on the latent representations of both labeled and unlabeled training data. Afterward, we determined one or more components of the mixture into which the positively labeled training samples are most likely to be projected via the encoder. Then, for a new sample, the score is the (average) log likelihood of the sample in the anomalous components.

IV. EXPERIMENTAL SETUP

IV.A. Data

Every spectrogram was divided into patches of size 128×128 pixels. Out of 40 preprocessed spectrograms, 370 nonoverlapping patches were extracted and labeled. This results in a labeled training data set $\{X_l, Y\}, X_l = \{x_i\}_i, x_i \in \mathbb{R}^{128\times128\times1}, Y = \{y_i\}, y_i \in \{0, 1\}$ of samples X_l and labels Y, where $Y = 1$ if a patch contains a chirping mode. Also, an unlabeled database X_u of 330 000 patches coming from 2000 spectrograms was created.

Training of the one-class model was done both with labeled positive spectrograms and on the large unlabeled data set. In the first case, the 50% of positive spectrograms was used for training, and the rest together with the unlabeled ones was used for testing. Also, training patches were randomly shifted, and noise was added to them so that there was a total of 10^4 training samples. In the second case, all of the labeled data were used only for testing. Ten different training and testing data sets were created this way for cross-validation purposes.

For training of the two-stage model, we have split the labeled data set into training/testing subsets with the ratio 80/20. Again, this splitting was done randomly a total of ten times.

IV.B. Model Architecture and Hyperparameters

Regarding the one-class model, the architecture for the one-class encoder was two or three convolutional layers with (32, 64) or (32, 32, 64) channels and kernel size of 5. Each convolutional layer was followed by a maxpooling layer that downscaled the image by a factor of 2. Then, a dense layer produced the final encoding into a d-dimensional latent space. The decoder mirrored the encoder architecture with transposed convolutions in place of maxpooling layers. Residual nets[25] (ResNets) types of residual blocks were used to speed up and stabilize the training. The hyperparameter values over which we have optimized trained models are in Table I. Parameter γ denotes the scaling parameter of the inverse multiquadratics (IMQ) kernel.[26]

Regarding the two-stage model, the basic encoder architecture was as follows: three convolutional layers with (16, 16, 32) channels and kernel size of 3, (2, 2, 1) downscaling ratios via maxpooling, followed by two dense layers of width (256, d), where d is the dimension of the latent space. The decoder mirrored this architecture. Also, batch normalization[27] was used. The hyperparameter values over which we have optimized are in Table I. The number of components in the used prior is denoted by N.

The base architecture of the two models is slightly different. We have experimented with different architectures prior to the hyperparameter optimization and found that for the different tasks, different architectures provide better results. This is probably because both models have a different objective: The one-class model requires

TABLE I.

Overview of Model Hyperparameters*

Parameter		Values
One-Class Model		
γ		$\{10^0, 10^{-1}, 10^{-2}\}$
$\lambda,\lambda_1,\lambda_2$		$\{10^1, 10^0, 10^{-1}\}$
d		$\{8, 128, 256\}$
Two-Stage Model		
First stage	N	$\{1, 2, 4, 8\}$
	γ	$\{10^0, 10^{-1}, 10^{-2}\}$
	$\lambda,\lambda_1,\lambda_2$	$\{10^1, 10^0, 10^{-1}\}$
	d	$\{2, 16, 32, 64\}$
Second stage	k	$\{1, 3, \ldots, 31\}$
	M	$\{2, 4, 6, 8\}$

*The terms λ_1 and λ_2 are scaling parameters for the combination of MMD and adversarial loss.

precise reconstruction, while the two-stage model is evaluated based on the shape of latent space. Ideally, we would include architectures as a tunable parameter, but that would require a level of computational power that was not available to us.

Both models use rectified linear unit[28] (ReLu) activation and were optimized with the RMSProp optimizer with learning rate 10^{-4}. For a single optimization iteration, batches of 128 patches were used. We have implemented all the models in the Julia language[29] and trained them on TITAN V Nvidia GPU with 12 Gbytes of memory.

V. RESULTS

In this section, the results of the experiments with the one-class and two-stage models are examined and compared. Also, we discuss the importance of an appropriate train/test splitting strategy that was used in our experiments in order not to obtain overly optimistic model performance estimates.

V.A. Output Evaluation

In case our framework was implemented in a production environment, the working scenario would be the following. A set of experiments to be analyzed would be selected. Then, the needed signals would be extracted, and spectrograms would be computed and divided into patches of appropriate size. These would be fed to a trained model that would produce scores to enable ranking of the patches. Since this would produce hundreds, maybe thousands, of patches and scores, the operator would ideally want to go through only a few with the highest score. In Fig. 3 we show the output of such procedure: four patches with the highest score, out of which three contain a chirping mode. It illustrates that even though the neural network encoding might be powerful, it is still basically a black box model, and we need to be very careful in its evaluation. Because of this, we evaluate the model performance by computing the area under the receiver operating curve (AUC), which is a standard measure for binary classification problems and also by precision@k score, which is the precision at the k-highest scoring samples.

V.B. One-Class Model Optimization

The hyperparameter optimization routine resulted in hundreds of trained models. To select the best one, the AUC and precision@50 measures were computed on a testing data set. Then, for a set of fixed hyperparameter values, these were averaged over ten cross-validation test-train splits. The best results for a combination of target class and used divergence based on these measures

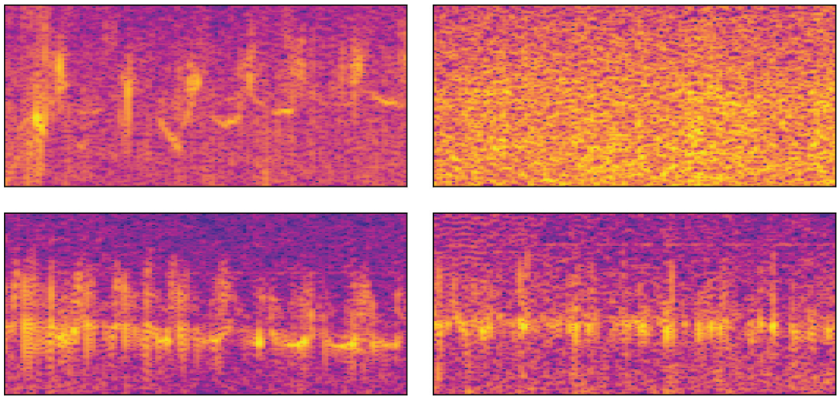

Fig. 3. Examples of spectrogram patches identified as containing a chirping mode.

are reported in Table II. Clearly, it seems that modeling the distribution of chirping mode spectrograms is more difficult than vice versa with the exception of KLD, which completely fails. Also, the precision in the top samples is very low in the Alfvén target class. Surprisingly, a plain autoencoder achieves results almost comparable to the other models. In Fig. 4 are the receiver operating characteristic[30] (ROC) and precision-recall[31] (PR) curves of the single best-performing one-class models as well as the two-stage models.

V.C. Two-Stage Model

Here, we evaluate the performance of the two-stage model. The methodology of hyperparameter optimization via cross-validation is similar to that used for the one-class model. The best average results across ten splits for different combinations of stage one divergences and stage two classifiers are reported in Table III. The simple kNN model is superior to the GMM approach. Also, MMD regularization seems to produce the best results. We might speculate that this might be due to the improved ability to produce a well-separated encoding enforced by the used prior. Again, in Fig. 4, see the ROC and PR curves for the single best two-stage models.

A question one might ask is whether the autoencoding is truly necessary. In the end, we are doing a projection from $d = 128 \times 128 = 16384$ dimensional picture space into at most $d = 64$ dimensional latent space, which must naturally lead to a loss of information. As shown in Fig. 5, where $d = 8$, the autoencoder is able to identify the difficult nonlinear correlations and improve the performance of a subsequent second-stage kNN model. The compression is clearly necessary for overcoming the curse of dimensionality, which implies that L_2 distance degenerates in large dimensions. An alternative approach to overcoming the issue of large input dimension might be to train a classification convolutional neural network, which does the compression by its nature. However, we have not chosen to go this path since we believe that such a network would be

TABLE II

Results of Optimization of the One-Class Model*

Divergence	Target Class	AUC	precision@50
—	Alfvén	0.57 ± 0.04	0.24 ± 0.05
KLD	Alfvén	0.74 ± 0.06	0.44 ± 0.13
MMD	Alfvén	0.77 ± 0.03	0.49 ± 0.06
JSD	Alfvén	0.69 ± 0.07	0.42 ± 0.08
MMD + JSD	Alfvén	0.72 ± 0.09	0.37 ± 0.03
—	Non-Alfvén	0.82 ± 0.03	0.86 ± 0.06
KLD	Non-Alfvén	0.46 ± 0.05	0.50 ± 0.14
MMD	Non-Alfvén	0.84 ± 0.03	0.90 ± 0.06
JSD	Non-Alfvén	0.84 ± 0.05	0.83 ± 0.10
MMD + JSD	Non-Alfvén	0.84 ± 0.01	0.87 ± 0.01

*Target class differences are described in Sec. IV, "Experimental Setup." No divergence is a plain autoencoder with MSE training objective.

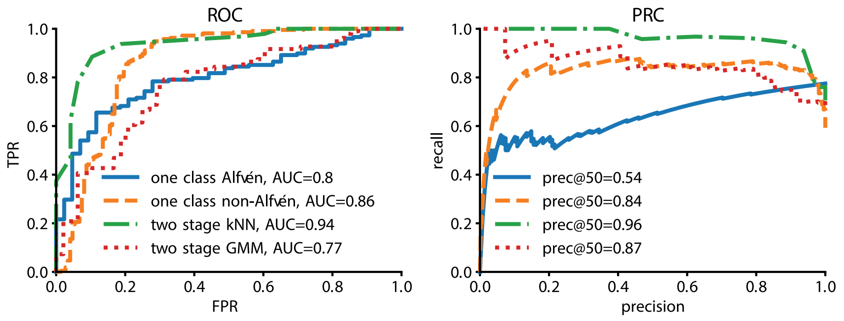

Fig. 4. The ROC and PR curves of selected models.

TABLE III

Results of Hyperparameter Tuning of the Two-Stage Model Across Ten Cross-Validation Splits

S1 Divergence	S2 Model	AUC	precision@50
—	kNN	0.80 ± 0.07	0.88 ± 0.10
KLD	kNN	0.80 ± 0.08	0.85 ± 0.11
MMD	kNN	0.91 ± 0.06	0.94 ± 0.05
JSD	kNN	0.83 ± 0.07	0.87 ± 0.10
MMD + JSD	kNN	0.86 ± 0.07	0.91 ± 0.10
—	GMM	0.75 ± 0.06	0.80 ± 0.10
KLD	GMM	0.74 ± 0.06	0.83 ± 0.11
MMD	GMM	0.66 ± 0.12	0.72 ± 0.12
JSD	GMM	0.74 ± 0.06	0.82 ± 0.11
MMD + JSD	GMM	0.76 ± 0.06	0.84 ± 0.10

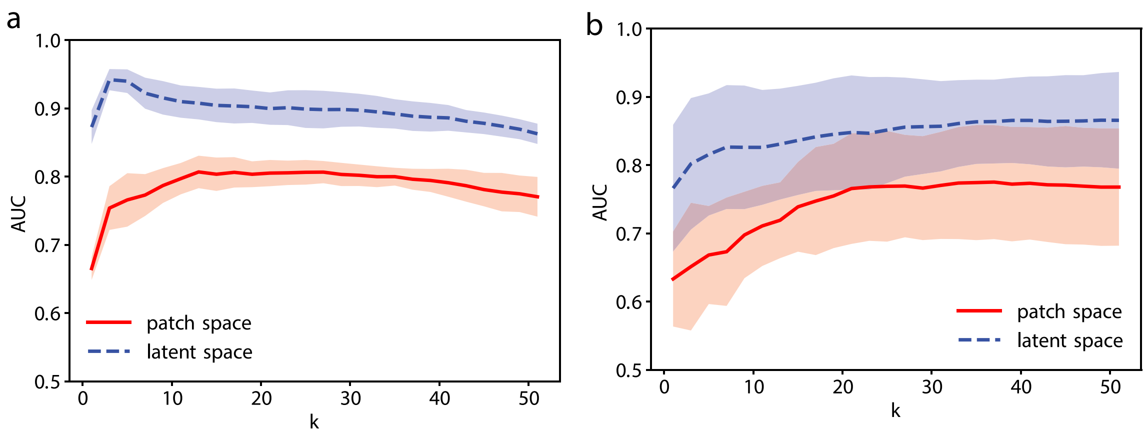

Fig. 5. The kNN fits for different values of k. The red line and band show the mean and 1σ bands of the resulting AUC values when kNN is fitted to the original vectorized images. The input space dimensionality is $d = 16384$. The blue dashed line and band are the same quantities for a $d = 8$ dimensional representation by a first-stage model. (a) The training and testing splits were done on the level of individual patches, leading to better performance and less variance. (b) The split was done on the level of the original spectrograms, which is a more realistic scenario. The standard deviation and mean were computed from ten random splits.

highly susceptible to overfitting since it requires much labeled data that are not available to us. Instead, we have tried to overcome this in the two-stage model by learning the compression from all the available unlabeled data.

V.D. Influence of the Train/Test Splitting Methodology

At first, the splitting of testing and training labeled patches was done on the level of patches, without any regard for the spectrogram/experiment that the patch came from. It was assumed that the labeled chirping modes are homogeneous across the spectrograms. However, this turned out not to be true. Therefore, the train/test splits were done on the level of spectrograms, which were then subsequently divided into patches. See Fig. 5a, where the AUC curves for different values of k of the kNN model are for the case when the data split was done on the level of patches. The blue line that is the result of kNN fit peaks at $k = 3$. On the other hand, there is no such peak in Fig. 5b, where splitting was done on the level of spectrograms. This indicates that the positively labeled patches in a single spectrogram are much more similar to each other than to those in different spectrograms, as only a relatively low number of neighbors are sufficient for optimal performance. Also, the variance of Fig. 5a is much higher, again indicating larger differences across spectrograms. If we continued with the splitting on the level of patches, we would have a biased and too optimistic estimate of performance before putting the framework into production.

VI. CONCLUSION

Our task was identification of anomalous phenomena, i.e., chirping AEs, in graphical representations of signals measured during the operation of a tokamak. To this end, we have proposed two models based on generative autoencoders. The first model learned the distribution of normal data and identified chirping modes as out-of-class samples of this distribution. The second model implemented a two-stage learning approach. A regularized convolutional VAE trained on unlabeled data was successfully combined with a classifier trained with a smaller labeled data set. It has been shown that both models are viable options in chirping mode identification, although the latter one proved to be superior.

We have also shown the need for proper cross-validation splitting of data in the evaluation phase and outlined the need for careful evaluation in order for the model to be useful in real-world application. However, this is still work in progress. We mentioned the need to use a more appropriate evaluation measure that reflects the operational conditions. Furthermore, so far we have used spectrograms only from a single U-probe, but there are about 40 more magnetic diagnostics that could be potentially used for this task; e.g., their spectrograms/correlograms could be added as an additional input channel. Finally, a more thorough evaluation of the contemporary experimental results is needed for the understanding of the framework behavior and to be applicable to COMPASS operation, most likely through the expansion of the labeled data set.

Acknowledgments

Research presented in this work has been supported by the grant SGS18/188/OHK4/3T/14 provided by Czech Technical University in Prague and by GAČR project 18-21409S. This work has been carried out within the framework of the project COMPASS-U: Tokamak for Cutting-Edge Fusion Research (No. CZ.02.1.01/0.0/0.0/16_019/0000768) and cofunded from European structural and investment funds. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan V GPU used for this research.

This work has been carried out within the framework of the EUROfusion Consortium and has received funding from the Euratom research and training programme 2014–2018 under grant agreement No. 633053. The views and opinions expressed herein do not necessarily reflect those of the European Commission.

ORCID

Vít Škvára http://orcid.org/0000-0003-4165-7124
Václav Šmídl http://orcid.org/0000-0003-3027-6174

References

1. R. METT and S. MAHAJAN, “Kinetic Theory of Toroidicity-Induced Alfvén Eigenmodes,” *Phys. Fluids B*, **4**, *9*, 2885 (1992); https://doi.org/10.1063/1.860459.
2. T. MARKOVIC et al., “Alfvén-Character Oscillations in Ohmic Plasmas Observed on the COMPASS Tokamak,” *Proc. 44th EPS Conf. Plasma Physics*, Belfast, Northern Ireland, June 26–30, 2017, Vol. 5, European Physical Society (2017).
3. A. LVOVSKIY et al., “Observation of Rapid Frequency Chirping Instabilities Driven by Runaway Electrons in a Tokamak,” *Nucl. Fusion*, **59**, *12*, 124004 (2019); https://doi.org/10.1088/1741-4326/ab4405.
4. S. SHARAPOV et al., “MHD Spectroscopy Through Detecting Toroidal Alfvén Eigenmodes and Alfvén Wave Cascades,” *Phys. Lett. A*, **289**, *3*, 127 (2001); https://doi.org/10.1016/S0375-9601(01)00588-6.
5. T. MARKOVIC et al., “Alfvén-Wave Character Oscillations in Tokamak COMPASS Plasma,” *Proc. 42nd EPS Conf. Plasma Physics*, Lisbon, Portugal, June 22–26, 2015, p. P4.104, European Physical Society (2015).
6. A. MELNIKOV et al., “Quasicoherent Modes on the COMPASS Tokamak,” *Plasma Phys. Control. Fusion*, **57**, *6*, 065006 (2015); https://doi.org/10.1088/0741-3335/57/6/065006.
7. R. PANEK et al., “Status of the COMPASS Tokamak and Characterization of the First H-Mode,” *Plasma Phys. Control. Fusion*, **58**, *1*, 014015 (2015); https://doi.org/10.1088/0741-3335/58/1/014015.
8. D. P. KINGMA and M. WELLING, “Auto-Encoding Variational Bayes,” arXiv preprint arXiv:1312.6114 (2013).
9. K. KOVAŘÍK et al., “U-Probe for the COMPASS Tokamak,” *Proc. 20th Annual Conf. Doctoral Students*, Prague, Czech Republic, May 31–June 3, 2011, Part II, p. 227 (2011).
10. L. MESCHEDER, S. NOWOZIN, and A. GEIGER, “Adversarial Variational Bayes: Unifying Variational Autoencoders and Generative Adversarial Networks,” *Proc. 34th Int. Conf. Machine Learning*, Sydney, Australia, August 6–11, 2017, Vol. 70, p. 2391 (2017).
11. I. TOLSTIKHIN et al., “Wasserstein Auto-Encoders,” arXiv preprint arXiv:1711.01558 (2017).
12. Y. LeCUN et al., “Backpropagation Applied to Handwritten Zip Code Recognition,” *Neural Comput.*, **1**, *4*, 541 (1989).
13. M. RANZATO et al., “Efficient Learning of Sparse Representations with an Energy-Based Model,” *Proc. 21st Annual Conf. Advances in Neural Information Processing Systems*, Vancouver, British Columbia, Canada, December 3–6, 2007, p. 1137 (2007).

14. D. P. KINGMA and J. BA, "Adam: A Method for Stochastic Optimization," arXiv preprint arXiv:1412.6980 (2014).

15. J. M. TOMCZAK and M. WELLING, "VAE with a VampPrior," arXiv preprint arXiv:1705.07120 (2017).

16. A. MAKHZANI et al., "Adversarial Autoencoders," arXiv preprint arXiv:1511.05644 (2015).

17. I. GOODFELLOW, "NIPS 2016 Tutorial: Generative Adversarial Networks," arXiv preprint arXiv:1701.00160 (2016).

18. P. VINCENT et al., "Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion," *J. Mach. Learn. Res.*, **11**, 3371 (Dec. 2010).

19. V. CHANDOLA, A. BANERJEE, and V. KUMAR, "Anomaly Detection: A Survey," *ACM Computing Surveys*, **41**, *3*, 15 (2009); https://doi.org/10.1145/1541880.1541882.

20. B. SCHÖLKOPF et al., "Estimating the Support of a High-Dimensional Distribution," *Neural Comput.*, **13**, *7*, 1443 (2001).

21. J. AN and S. CHO, "Variational Autoencoder Based Anomaly Detection Using Reconstruction Probability," Seoul National University Data Mining Center (2015).

22. V. LEVEAU and A. JOLY, "Adversarial Autoencoders for Novelty Detection," Inria - Sophia Antipolis (2017).

23. Z. DENG et al., "Efficient *k*NN Classification Algorithm for Big Data," *Neurocomputing*, **195**, 143 (2016); https://doi.org/10.1016/j.neucom.2015.08.112.

24. Y. HUANG et al., "A Gaussian Mixture Model Based Classification Scheme for Myoelectric Control of Powered Upper Limb Prostheses," *IEEE Trans. Biomed. Eng.*, **52**, *11*, 1801 (2005); https://doi.org/10.1109/TBME.2005.856295.

25. K. HE et al., "Deep Residual Learning for Image Recognition," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Las Vegas, Nevada, June 27–30, 2016, p. 770, IEEE (2016).

26. J. GORHAM and L. MACKEY, "Measuring Sample Quality with Kernels," arXiv preprint arXiv:1703.01717 (2017).

27. S. IOFFE and C. SZEGEDY, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," arXiv preprint arXiv:1502.03167 (2015).

28. R. H. HAHNLOSER et al., "Digital Selection and Analogue Amplification Coexist in a Cortex-Inspired Silicon Circuit," *Nature*, **405**, *6789*, 947 (2000).

29. J. BEZANSON et al., "Julia: A Fresh Approach to Numerical Computing," *SIAM Rev.*, **59**, *1*, 65 (2017); https://doi.org/10.1137/141000671.

30. T. FAWCETT, "An Introduction to ROC Analysis," *Pattern Recognit. Lett.*, **27**, *8*, 861 (2006); https://doi.org/10.1016/j.patrec.2005.10.010.

31. K. BOYD, K. H. ENG, and C. D. PAGE, "Area Under the Precision-Recall Curve: Point Estimates and Confidence Intervals," *Proc. Joint European Conf. Machine Learning and Knowledge Discovery in Databases*, Prague, Czech Republic, September 23–27, 2013, p. 451, Springer (2013).