Disruption prediction in nuclear fusion with WaveNet architecheture on JET dataset

Emilien Seiler, Master in Computational Science Engineering, EPFL
Supervisor: Pascal Fua, Computer Vision Laboratory and Alessandro Pau, Swiss Plasma Center
Master Semester Project, Spring 2022 semester, 8 credits

***Abstract*—In this report, we discuss advances in deep convolutional neural networks (CNNs) for sequence learning, which allow identifying long range, multi-scale phenomena in long sequences, such as those found in fusion plasmas. We used an architecture derived from known Wavenet network. We apply this neural network architecture to the popular problem of disruption prediction in fusion energy tokamaks, using raw data from the JET tokamak. We reach an accuracy of 96% and an F1 score of 84% on individual time-slices.**

I. Introduction

Avoiding plasma disruptions will be one of the major concerns for the next generation of tokamaks such as ITER. A tokamak is a device which uses a powerful magnetic field to confine plasma in the shape of a torus. This is one of several types of magnetic confinement devices being developed to produce controlled thermonuclear fusion power. Disruptions can cause severe damage to the structural integrity of the machines, forcing unexpected and eventually long maintenance interventions, which significantly reduce the availability of the device. Avoiding disruptions is a great challenge for tokamak fusion devices on the road to fusion energy. For avoidance, the chain of events preceding the disruption has to be detected and suitable interventions have to be performed either to fully recover the nominal plasma parameters. For mitigation, massive amount of gas or pellets can be injected into the plasma to increase the radiation, with the aim of reducing thermal and electromagnetic loads. The plasma control system on ITER, as well as the one on JET, will necessitate the availability of reliable triggers that have to satisfy different requirements, depending on whether they are intended for avoidance or mitigation purposes. The reliability and the effectiveness of such triggers can be defined in terms of warning time (that is the time between the trigger and the time of disruption), correct predictions, missed and false alarms.

In fusion energy plasmas, many disparate diagnostic instruments are simultaneously used in order to capture various spatiotemporal measurements such as radiated power, electron temperature... In addition, fusion experiments are increasingly built to run longer pulses, with the goal of eventually running a reactor continuously. The confluence of these facts leads to large, complex datasets with phenomena manifest over long sequences. A key challenge is enabling scientists/engineers to utilize these long sequence datasets to, for example, automatically catalog events of interest or predict the onset of phenomena.

Machine learning, and specifically the variant deep learning, has been proven to be highly successful in automating a number of tasks, such as identifying objects in images, language translation... Many deep learning architectures have been created and successfully applied to sequence learning problems, in areas of time-series analysis or natural language processing. However, many of the typical architectures used for learning from sequences [e.g., recurrent neural networks (RNNs) and their most popular variant long short-term memory (LSTM) networks] suffer from memory loss; long-range dependencies in sequences are difficult for these architectures to track.

In this report, we discuss recent advances in neural networks, specifically an architecture that uses dilated convolutions in a convolutional neural network (CNN), which was designed to overcome these problems of learning on long sequences. We use this architecture to predict oncoming disruptions in fusion plasma discharges of the Joint European Torus tokamak (JET).

The outline for the rest of this paper is as followed: Sec. II discusses about the JET dataset and its differents dignostic, Sec. III discusses about the model and its speceficity for sequence learning, Sec. IV discusses about the experiment and the trainng procedure done to predict distruption with this kind of model. V present the results . VI discusses the results and improvements that can be made.

II. Dataset

The Joint European Torus, or JET 1, is an operational magnetically confined plasma physics experiment, located at Culham Centre for Fusion Energy in Oxfordshire, UK. Based on a tokamak design, the fusion research facility is a joint European project with a main purpose of opening the way to future nuclear fusion grid energy. At the design stage JET was larger than any such machine then in production.

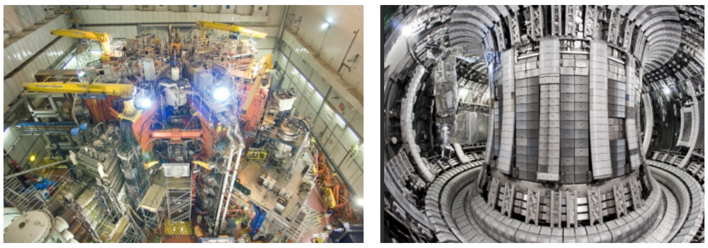

Fig. 1. JET tokamak

During the experiments on JET tokamak several measurements are performed to track the behavior of the plasma. It is on these times series data that we will perform machine learning. These are the several 0D signals measured during an experiment and provided in the data:

- the electron cyclotron emission (ECE on Fig. 2 from [1]) consists of 96 channels over four data acquisition bands, allowing either the first- harmonic measurements (O-mode) or the second-harmonic measurements (X-mode).

- ECE Michelson Interferometers KK1 (ECM1) temperature profile measurements along 51 channels with a resolution of 60Hz
- the high resolution Thomson scattering (HRTS on Fig. 2 from [1]) measuring electron temperature (Te) and electron density (Ne), providing 63 data points per profile with a repetition rate of 20 light pulses per second (20 Hz).

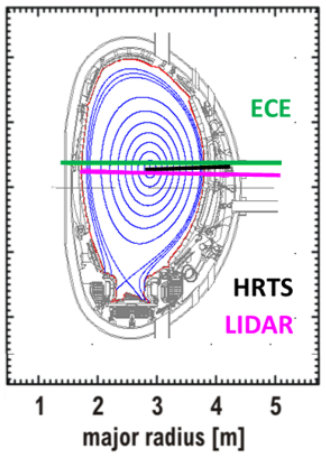

Fig. 2. Line of sight in the poloidal section for ECE, HRTS, and LIDAR diagnostics

- KB5H (Fig. 3 from [1]) cameras that collects the horizontal radiation along 24 chords, 8 channel cover the divertor, 16 channel cover the “bulk” plasma.
- KB5V (Fig. 3 from [1]) cameras that collects the vertical radiation along 23 chords, 5 channel cover the divertor, 15 channel cover the “bulk” plasma.

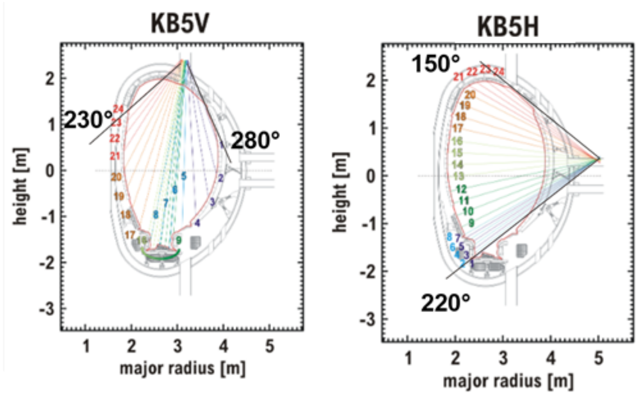

Fig. 3. Horizontal and vertical view bolometer KB5H and KB5V at JET

All these measurements are 0D diagnostics, so for each channel, we are provided with the x-position and z-value of measurements. This gives us a total of :

- 192 channels for ECE diagnostic
- 102 channels for ECM1 diagnostic
- 126 channels for HRTS_Te diagnostic
- 126 channels for HRTS_Ne diagnostic
- 48 channels for KB5H diagnostic
- 46 channels for KB5V diagnostic

In addition to these 0D signals we are provided with 1D signals like the peaking factor [1] of these 0D diagnostic BOLO_HV, ECE_PF... Or even the internal inductance LI, the plasma current IPLA, Greenwald fractio GW... A total of 70 different 1D signals are measured and give information about the plasma behavior.

In the end when we combine 0D and 1D signals we are provided a total of 710 channels for each experiments. Hence, the final database is composed of 520 experiments that we will call shots with 62% of disrupted shot. So in terms of disruptive/non disruptive shot the dataset is quite balanced. A shot is therefore composed of 710 times series channels with a resolution of 2 Hz (a value each 2ms).

III. Model

A. Deep Learning

Deep learning has been tremendously successful in recent years in achieving state-of-the-art results for many machine learning tasks, including image classification, language translation, and speech recognition. Deep learning refers to neural networks with many hidden layers. Each hidden layer provides a linear transform (with a number of weights, Wj , along with a bias term bj), followed by a nonlinear transform (called the activation). By stacking up several layers, a deep neural network can potentially learn complicated non-linear functions. A task specific loss function is defined and produces a measure of the error between predictions of a network and the user labeled targeted outcomes. Using this measure of error from the loss function, neural networks use a method called backpropagation to update the weights of the neural network, with the goal of making the predictions match the targets. One of the reasons for deep learning’s great success is the ability to learn multiple filters for high-dimensional data, avoiding the need for humans to do feature extraction. This allows the deep learning algorithms to learn directly from raw data.

B. Causal Dilated Convolution

Recently, there has been much research into deep learning architectures, which can overcome the deficiencies of RNN/LSTM networks and handle long, multi-scale sequences. A seminal paper presented by DeepMind presented such an architecture, WaveNET [2] which is a convolutional neural network (CNN) focused on generating realistic audio. One of the key insights of this paper was to use causal dilated convolutions to increase the receptive field of the network (i.e., the number of sequence points used by a neural network to make a prediction at a single time point). This overcomes the dilemma faced with using normal convolutions in causal networks, where to be sensitive to long sequences, you must increase the convolutional kernel size and/or the number of layers in the network. Dilated convolutions have a dilation factor (d) that represents the number of input points skipped between filter parameters. e.g., the sequence output y[n] from a dilated convolution with dilation d is

$$y[n] = \sum_{i=0}^{k-1} w[i] * x[n - d * i] \tag{1}$$

where w represents the weights of the 1D dilated convolution kernel of size k and x[n] is the input sequence. A normal convolution results by setting d = 1. By stacking layers of dilated convolutions and increasing the dilation factor in each layer, the receptive field of the network can be increased while maintaining a tractable number of model parameters. The difference between normal and dilated convolution architectures is shown in Fig. 4.

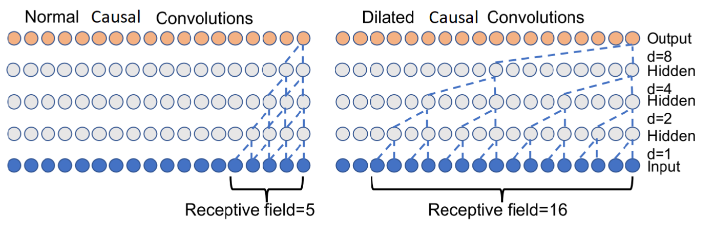

Fig. 4. Normal convolution vs dilated convolution

C. Wavenet

The Wavenet [2] is generative model introduce by DeepMind operating directly on the raw data. It perform state of the art performance for audio generative task like text-to-speech.
The joint probability of a sample $x = x_1, ..., x_T$ is factorised as a product of conditional probabilities as follows:

$$p(x) = \prod_{t=1}^{T} p(x_t|x_1, ..., x_{t-1}) \tag{2}$$

Each sample x_t is therefore conditioned on the samples at all previous timesteps. The conditional probability distribution is modelled by a stack of convolutional layers. There are no pooling layers in the network. The model outputs a categorical distribution over the next value x_t with a softmax layer and it is optimized to maximize the loglikelihood of the data w.r.t. the parameters.

By using causal convolutions, we make sure the model cannot violate the ordering in which we model the data: the prediction $p(x_{t+1}|x_1, ..., x_t)$ emitted by the model at timestep t cannot depend on any of the future timesteps $x_{t+1}, x_{t+2}, ..., x_T$.

Stacked dilated convolutions enable networks to have very large receptive fields with just a few layers, while preserving the input resolution throughout the network as well as computational efficiency. The dilation is doubled for every layer $d = 1, 2, 4, ..., 512$. The intuition behind this configuration is two-fold. First, exponentially increasing the dilation factor results in exponential receptive field growth with depth. For example each 1, 2, 4, . . . , 512 block has receptive field of size 1024, and can be seen as a more efficient and discriminative (non-linear) counterpart of a 1×1024 convolution. Second, stacking these blocks further increases the model capacity and the receptive field size.

The activation function used is :

$$z = tanh(W_k * x) \times \sigma(W_k * x) \tag{3}$$

where $*$ denotes a convolution operator, $\times$ denotes an element-wise multiplication operator, $\sigma()$ is a sigmoid function, k is the layer index, and W is the learnable convolution filter.

Both residual and parameterised skip connections are used throughout the network, to speed up convergence and enable training of much deeper models. In Fig. 5 we show a residual block of the model, which is stacked many times in the network.

The receptive field of this architecture is calculated with the 4 hyperparameters of the model:

- K, the kernel size

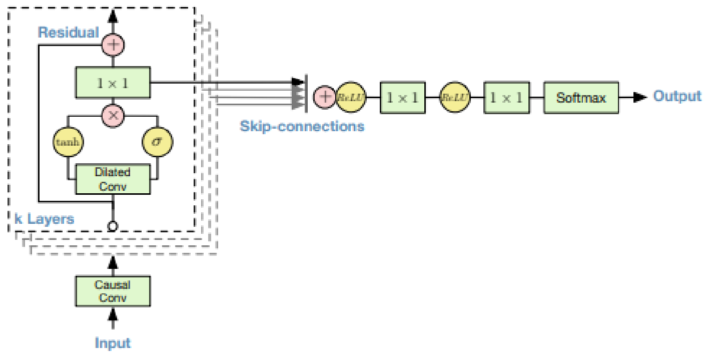

Fig. 5. Overview of the residual block and the entire architecture

- D, the dilated coefficient
- L, the layer size
- S, the stack size

$$N_{rec} = K + S \times \sum_{i=0}^{L-1} (K-1) \times D^i \tag{4}$$

IV. Experiment

A. Classification task

The idea of the implementation is to use the WaveNet architecture to build a WaveNet classifier. It will take the JET channel data as input and used N_{rec} time-step to predict whether there will be a disruption or not for the next time-step. The objective isn't to classify each shot but to classify each time step.

1) Binary classification by time step: We treat the problem of disruption prediction as a binary classification problem, where we predict whether each time slice corresponds to a "non-disruptive" 0 or "disruptive" 1 class.
For a non-disruptive shot all time step are label trivialy as non-disruptive 0.
For disruptive shot we define a "pre-disruptive" time from which we consider time step as disruptive until the time of disruption. The pre-disruptive time is define with one of these events:

- Radiation collapse
- Impurity accumulation

Fig. 6 is an exemple of how we labeled a disruptive shot

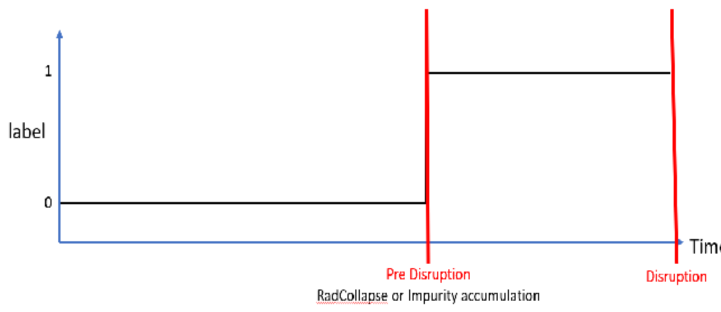

Fig. 6. Label for a disruptive shot

2) Receptive field: The label of each time step is predict with the past measurement. Then for a shot a length N_{shot} a receptive field of size N_{rec} we predict N_{pred} time step:

$$N_{pred} = N_{shot} - N_{rec} \tag{5}$$

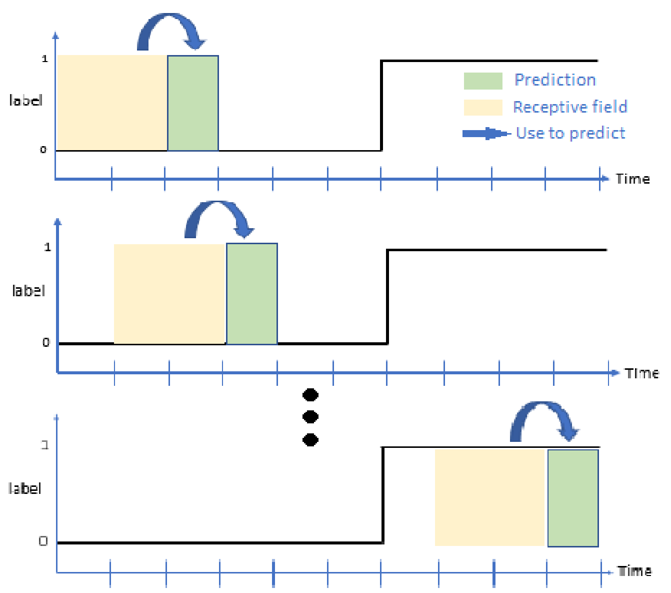

Fig. 7. Toy example of how label are predicted

because we don't have access to the past for the N_{rec} first time step. Fig. 6 is a toy example to better understand the notion of receptive field.

B. Data preprocessing

1) Data cleaning: Given the experimental conditions, raw data coming from the diagnostics are not ready to be used as they are but need some post processing. Even if the diagnostics employed are state of the art, the difficult experimental conditions cause the instruments to not be completely efficient all the time.

- Infinite values are present (the diagnostic may have received a signal too high to associate a finite number to it: this can be due to different mechanisms according to the diagnostic)
- NAN values are present. These can derive from many situations (the diagnostic missed some data, that specific diagnostic wasn't used for that specific shot)
- Nonphysical/non plausible values are present (values of density smaller than 0, safety factor bigger than 10)
- Data are not normalized

To deal with these difficulties the procedure is the following:

- First, we deal with the presence of Inf values, which are substituted everywhere with NaN, since they are effectively missing values.
- For NaNs in the middle of the shot, coming from an error in data acquisition: we linearly interpolate with the adjacent values
- For NaNs at the beginning and end of the shot, we leave them as they are: these are phases in which the diagnostics haven't started/have already stopped working properly
- It is necessary to filter nonphysical and implausible data. This is of paramount importance to correctly visualize the distribution of the data and to make an informed decision on the kind of normalization to apply. For each feature, we plot the distribution over all the available shots:
  - Some distributions have no tails (or really small ones), so we simply exclude nonphysical values and maybe cut the short tails.
  - Others have long tails but simply need a cut (the tails have values that are nonphysical or non plausible)
  - for some, it is necessary to make an initial cut and then apply a transformation (log/power law), followed by a final cut (an exemple on Fig. 8)
- apply a min-max normalization

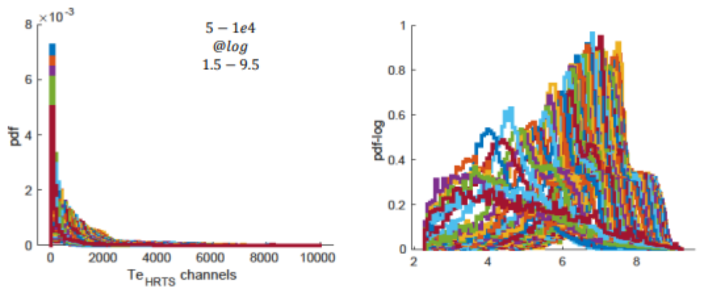

Fig. 8. Example of cut and log transform on HRTS_Te diagnostic

2) Shot to sequence: To perform the task the model needs to take as input time series with the same length. Because shots don't have the same number of time step we need to transform each shot to sequences of the same length. This transformation is also essential to be able to train with batch because if the data don't have the same shape we can't batch them. The Fig. 9 shows how we transform a shot in several sequences by losing as little information as possible.

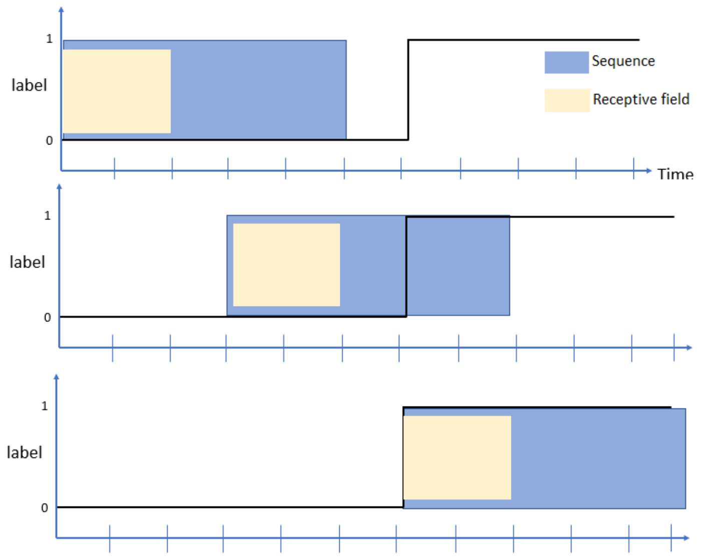

Fig. 9. Toy example of shot to sequence for N_{shot}=15, N_{seq}=5, N_{rec}=2, output 3 sequences

3) Training dataset: Another bottleneck regarding the data is that there are many NAN or missing channels in most of the shots. It is crucial for the model to take input data with the same channels. We made the choice to remove shots with too many missing channels and channels with too many missing shots. This is a compromise between the number of shots and the number of channels to be used for training.

Then we construct 2 different trainable dataset:

- dataset1 with 398 shot and 150 channels, mostly 1D channels and 0D ECM1 channels
- dataset2 with 431 shot and 373 channels, 1D channels and 0D ECM1, KB5H, KB5V, HRTS_Te, HRTS_Ne channels

After transforming each shot in several sequence of length $Nseq = 2000$ the two dataset have the following balance characteristic (see Tab. I) Because we classify timesteps and not

Dataset	$nb_{sequence}$	$\%Disr_{shot}$	$\%Disr_{timestep}$
1	687	46	6.5
2	767	55	8

TABLE I
Balance label characteristic of the 2 training dataset

shots it's important to note that the two datasets are imbalanced in a timestep point of view.

C. *Training parameters*

1) Model: The hyperparameters of the Wavenet model define the size of the receptive field.

- kernel size, K=3
- layer size, L=6
- stack size, S=4

These parameters lead to a receptive field of size 507 (see Eq. 4), the data resolution is 2 Hz, so there are 2ms between each time step. The Wavenet model therefore uses 1014ms in the past to predict whether the next time step is disruptive or not.

2) Loss function: Typical binary cross-entropy loss was used for the loss function during the neural network training:

$$L = -\frac{1}{n}\sum_{n} w_n[y_n log(\bar{y_n}) + (1 - y_n) log(\bar{y_n})] \qquad (6)$$

where n is the number of time step in a batch, y_n the target (binary, discrete, either 0 or 1), $\bar{y_n}$ the network prediction (continuous, ranging from 0 to 1), and w_n a constant class weight applied to help balance between disruptive and non-disruptive time step.
The weight w_n (see Tab. II) are calculated with the label proportion in the dataset.

$$w_{pos} = 0.5\frac{N_{timestep}}{N_{disruptive\_timestep}} \qquad (7)$$

$$w_{neg} = 0.5\frac{N_{timestep}}{N_{non\_disruptive\_timestep}} \qquad (8)$$

Dataset	w_{pos}	w_{neg}
1	7.72	0.53
2	6.15	0.54

TABLE II
w_n value for the two dataset

3) Optimizer: As a stochastic optimizer, we chose Adam. [3] This method is computationally efficient, has little memory requirements and is well suited for problems that are large in terms of data and/or parameters.
For dataset 1 $lr_{inital} = 0.001$ and for dataset 2 $lr_{inital} = 0.0001$ The learning rate is reduced by a factor 0.1 each 150 epoch.

4) Validation: For the validation the dataset is split in a training set and a test set with a split factor of 0.2. Accuraccy and f1 score are calculated on the test set.

V. RESULT

Fig. 10 presents the evolution of the loss during the training on dataset1 without validation. The training loss decreases over the time, thus indicating that the model learns the task. The figure also provides the evolution of metrics during the training. The accuracy and the f1 score increase during the training.

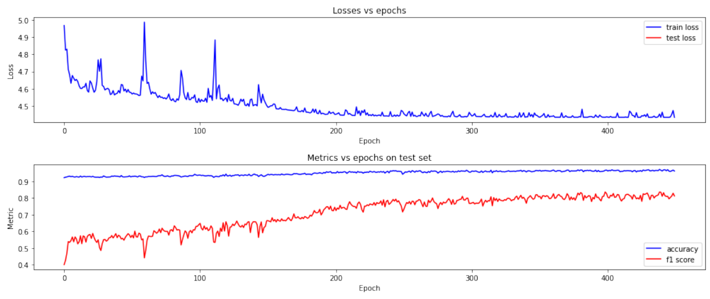

Fig. 10. Loss and metric on dataset1 with no validation set

Fig. 11 provides the confusion matrix of the trained model on dataset1. We can see that the model is more sensitive to false positive.

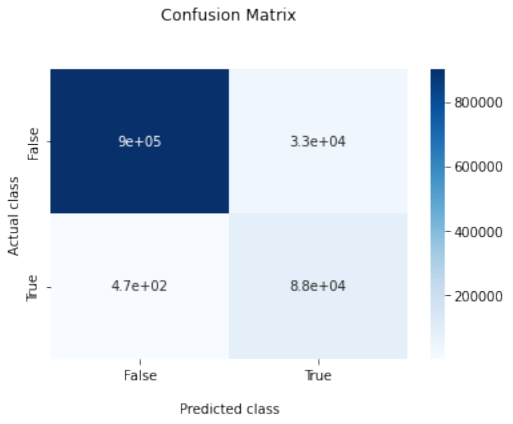

Fig. 11. Confusion matrix on dataset1

Fig. 12 presents the evolution of the loss during the training on dataset2 with validation. The training loss decreases over the time, but the test loss increase. However we can see that even if the test loss increase the metrics continue to increase.

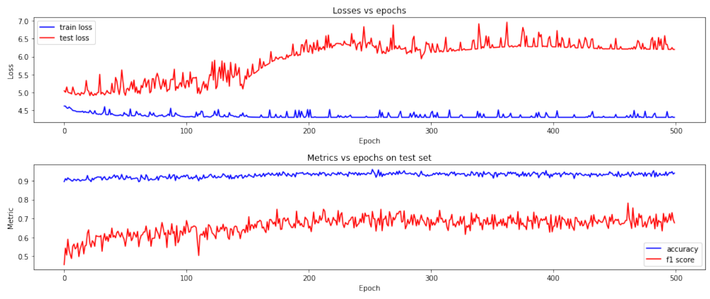

Fig. 12. Loss and metric on dataset2 with validation set

The Tab. III provides the final metrics result for the two datasets. Fig. 13 provides some shot visualization of the results. We can see that for disruptive shots the model predict well the disruptive event, there are just a few miss classification. The model predict the disruptive event 3 seconds before the disruption. For the non

Dataset	Validation	accuracy	f1 score
1	NO	0.96	0.84
2	YES	0.96	0.78

TABLE III
Result of training

disruptive event the result are quite good there is a bit false positive. But these miss classification are punctual and we can imagine that the alarm will be triggered after some continuous disruptive prediction (some disruptive prediction in a row).

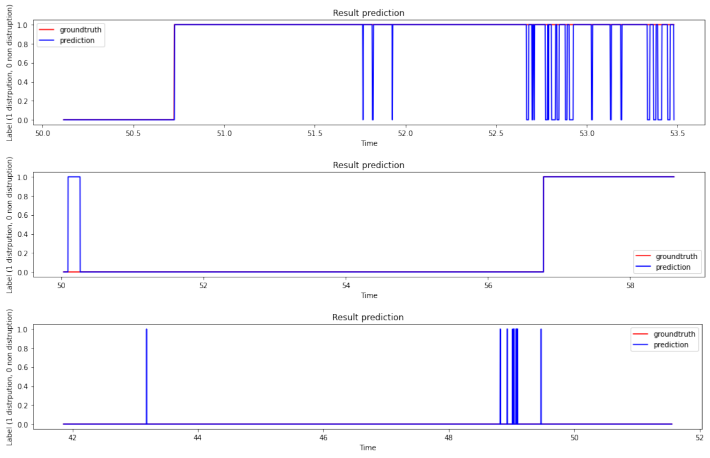

Fig. 13. Shot prediction

VI. Discussion

It is important to note that due to processor access issues and the semester project schedule, we started training the model in the last week of the project. The training parameters are therefore not optimal and we could probably obtain better results by trying other training procedures. The points on which it would be necessary to look closer at are:

- try other size of the receptive field and hyperparmeters for the wavenet (see Eq. 4)
- try other weights for the loss function (see Eq. 8)
- try to subsample the ”non-disruptive” class to avoid imbalance problems (see Tab. I)
- try other datasets with more/less channels (select the channels that we are sure provide information about the disruption).
- try to see in which condition the model has difficulty to classify (for exemple for the false positive)
- train on gpu

The code is provided on git.

VII. Conclusion

These results show that deep convolutional neural networks with dilated convolutions can be used for fusion problems where the multi-scale, multi-physics nature mandates capturing long-range dependencies in time series. In deed the result displayed on Fig. 13 show that the model can learn to prevent disruption event with the procedure explain in the report. Nevertheless, many things still need to be experimented with, especially in terms of training procedure, in order to have more optimal and robust results.

References

[1] A. Pau, A. Fanni, B. Cannas, S. Carcangiu, G. Pisano, G. Sias, P. Sparapani, M. Baruzzo, A. Murari, F. Rimini, M. Tsalas, and P. C. de Vries, “A first analysis of jet plasma profile-based indicators for disruption prediction and avoidance,” *IEEE Transactions on Plasma Science*, vol. 46, no. 7, pp. 2691–2698, 2018.

[2] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, “Wavenet: A generative model for raw audio,” 2016. [Online]. Available: https://arxiv.org/abs/1609.03499

[3] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2014. [Online]. Available: https://arxiv.org/abs/1412.6980