Feature identification of MANTIS data

Aymen Ayadi, Haitham Hammami, Rami Azouz
Machine Learning Project 2, EPFL, Switzerland

***Abstract*—Machine learning is brand new trend that allowed researchers and scientists to solve almost any complicated problems in reasonable time. In this project, we are working on identifying divertor legs from a set of images token from the inside of fusion reactor (TCV tokamak). We implemented CNN algorithm which proved itself as a good contender to analyse visual data. We ended up having a model that reaches 95.43% of accuracy when identifying divertor legs given an input dataset of images.**

I. Introduction

In the previous decades, global energy consumption grew exponentially due to the continuous development of countries around the world. However, most of the current energy production energies are unreliable (renewable energies) or produce undesirable and dangerous wastes (Nuclear wastes, of CO_2). In the late 20^{th} century, scientists turned their focus

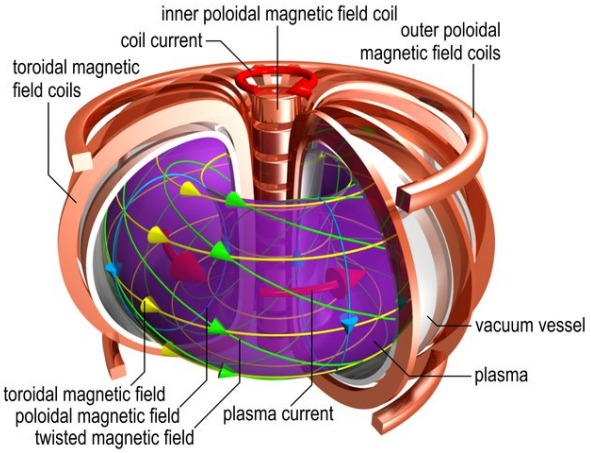


Fig. 1: *Schematic Tokamak [1]*

on the promising nuclear fusion reactors. This energy source distinguishes with its high efficiency, no resulting nuclear waste and especially, the reaction stops autonomously in a non confined environment. A fusion nuclear reactor fuses two light atoms to produce a heavier one while releasing energy. To make it work, very high temperature is required (tens of millions of degrees, similar to the temperature in the core of the sun). This presents a major challenge for researchers to design a successfully working reactor because in such temperature, atoms cannot even exist and everything is melt down into gluons and quarks. However, when atoms are heated up, they loose their electrons and become ionised and form a charged hot cloud which is also called a plasma [2]. The plasma shape and position can be controlled with magnetic fields. It is crucial to prevent the core of the plasma from touching the side walls of the reactor to avoid its meld down (Fig. 1).

A perfect fusion plasma must contain only deuterium, tritium and electrons. Any additional species affects the performance of the reactor [3]. Impurities can be introduced in the core by the reaction itself, such as the helium which is the product of the fusion of deuterium and tritium or the interaction with carbon tiles covering the walls of the reactor, or injected on purpose (Argon) to reduce the temperature of the plasma by radiation and thus the reactivity of the reactor. To be able to evacuate these impurities without shutting down the reactor, the divertor is introduced. This device help extracting heavy ions from the plasma when the latter approaches it (Fig.2).

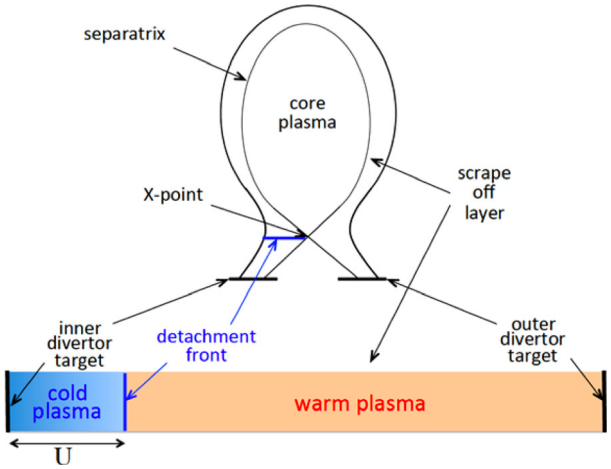
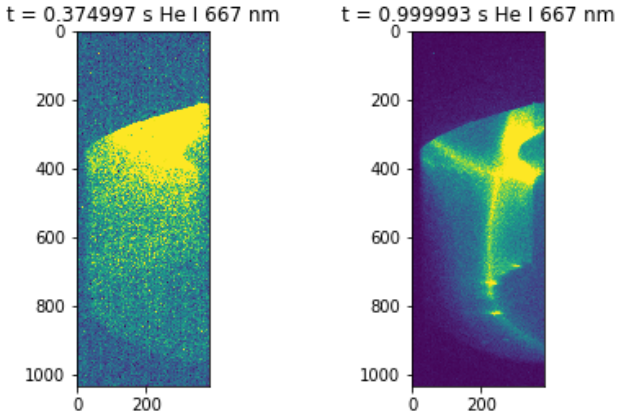


Fig. 2: *Schematic view of the poloidal cross-section of a tokamak and a scrape off layer shown in a straight flux tube approximation [4]*

With this configuration, the magnetic flux lines separates the plasma into two regions: a closed field lines, which is also called the core (the confined plasma), and open field ones where the evacuation of the impurities occurs also know as the scrap-off layer (SOL). The separatrix delimits the two regions. The SOL's legs (outer and inner legs) must touch the divertor's plates without touching any other material. Moreover, it is important to set the length of legs in order to avoid cooling the core (if the latter is too long) or melting the divertor plates (if they are too short). The magnetic flux measurements allow the scientists to identify with certitude the existence of the divertor legs. However, when there is an indication of their presence, there is only a high probability that they really do exist because their presence depends also on their temperature. Due to the heat produced inside the reactor, it is very difficult to introduce sensors that can resist such high



(a) *Non detectable legs* ***(b)*** *Detectable legs*

Fig. 3: *Images token from shot* $N°64005$ *at different times. In (a) during ramp up, the divertor legs are not detectable. In (b) during the burning phase, we can distinguish the outer and inner divertor legs*

temperature. In the TCV tokamak at EPFL, 10 cameras are mounted with each one sensitive to a given wavelength. These cameras can record the evolution of the plasma, by capturing the atomic radiation emitted from its edges, and can therefore identify with certitude the divertor legs existence, position, length and their detachment. Previously, researchers have to check manually the videos recorded after each experiment which is time consuming.

In this project, we are applying machine learning techniques, to train an algorithm on identifying divertor legs in a given set of images. This will greatly speed up analysis since researchers will not need to label the data manually.

II. Models and Methods

A. *MANTIS Data*

Multispectral Advanced Narrowband Tokamak Imaging System (MANTIS) is a 10 cameras system developed to assist the studies on the nuclear fusion reactor [5]. Swiss Plasma Center (SPC) provided us with access to their server via an ssh tunnel in order to download the data we need for our project. The data consist of different shots each one is characterised with a shot_ID and corresponds to a specific experiment. Every shot is composed of 10 video channels. These videos capture different wavelength (atomic_line) of the same experiment. As discussed in the introduction, our goal is to identify the presence of divertor legs in a given sequence of images. To train our Convolutional Neural Network (CNN), we downloaded. To be efficient in labeling our data, we picked experiments composed of only 3 phases in the following order: Ramp-up, burning and ramp-down. The legs are present only during the intermediate phase. To label our data, our supervisor provided us with a csv file containing the shot_ID, the starting and finishing time of the burning phase. We can therefore label the images where the divertor legs are present. In Fig.3 we can see two labeled pictures. The one on the left takes 0 as label, since there is no detectable legs, while we assign 1 as label for the one on the left because we can easily distinguish the outer and inner divertor legs.

Before defining our model, we first downloaded our dataset that is composed of shots from 26 experiments, each with up to 10 camera footage and from each video we sampled every 10th frame so that we have an interval of 50ms between two consecutive frames .

B. *Convolutional Neural Network*

Convolutional Neural Networks (CNNs) is a type of artificial neural network most commonly applied to analyzing visual imagery. It is a very popular neural network model for image classification problem, operating like a multilayer-perceptron (Linear Binary Classifier). The layers of a CNN consist of an input layer, an output layer and hidden layers that may include convolutional layers, pooling layers, fully connected layers and normalization layers.

B.1 Implementation

To implement our CNN, we chose to work with tensorflow.keras This open source library operates over TensorFlow to build and train deeplearning models. It has the advantage of being modular and composable which makes building layers and blocks easy and intuitive. We trained the model over google colab with a NVIDIA (R) Cuda compiler driver for faster computation.

B.2 Image Preprocessing

- **Image Resizing:** The dataset is composed of images of size 1032x428. We resize them by applying a downsampling to get square images of size 128x128. This is done in the purpose of reducing the dataset size and later on for reducing the computational costs .
- **Image augmentation:** Despite having a considerable amount of data, and since we reduced information held when resizing images as described in the previous point, we need to increase the diversity of the data and hence its robustness (to avoid bias). This will later allow a good simulation of all possible states when training the model. We generate new data by randomly applying the following transformations:
  - Rotation up to 30 degrees.
  - Translation up to 10 percent in width and height. This process allows recognizing the object in any part of the image while training our network.
  - Horizontal and vertical shearing up to 10 percent.
  - Zooming up to 10 percent.
  - Horizontal flipping.
- **Normalization:** Image pixel values lie between 0 and 4096. We need to apply normalization method which decreases sensitivity to the scale of features, and hence improves the convergence while training. The new pixel intensity x stands between 0 and 1 and is calculated using Equ.(1)

$$x = (I - I_{min})/(I_{max} - I_{min}) \quad (1)$$

where I is the current intensity of the pixel, I_{Min} and I_{Max} is the minimum and maximum intensities of the image

B.3 Defining the model

The task consists of classifying blocks of 128x128 pixels, considering that associated labels correspond to 1 if we detect the presence of the outer leg, 0 otherwise. As for the neural network architecture, we focus on the choice of layers and filters that will define our neural network model. It is an important step allowing the optimization of the model predictions on classifying the images. The following layers and functions were implemented in our model :

- **2D Convolutional Layer:** Consists of 2D filters that the algorithm needs in order to identify specific features. Each filter is convoluted with the input image and results into a new one in the output. The Convolutional Layer is the major building block where CNN learns the most. It takes the following hyper-parameters:
  - *filters*: we call them also kernels that the convolution layer will learn via training.
  - *kernel_size*: a 2-tuple specifying the width and height of the convolution kernel. If only one integer is specified, the same window length will be used for both dimensions.
  - *activation function*: is applied during forward propagation.

  Convolutional layer takes on input a tensor of the following shape: (number of images x image width x image height x image depth). The number of trainable parameters issued from convolutional layer is: ((shape of width of the filter*shape of height of the filter+1)*number of filters).
- **Max Pooling Layer:** Also known as downsampling layer, it is another building block of a CNN that usually follows the convolutional layer. It down samples the convoluted images and allows the algorithm do identify simpler features that can be crucial in classifying the images. There are no learnable parameters in pooling layer. This layer is just used to reduce the image dimension size.
- **Flattening Layer:** Flattens the convoluted and downsampled images in order to feed them to the hidden layers. It helps increasing the learning parameters of the model. There are no learnable parameters in flattening layer.
- **Fully Connected Layer:** also known as Dense Layer in which the results of the convolutional and pooling layers are input through one or more neural layers to generate a prediction. It takes as hyper-parameters:
  - *filters*: a 2-tuple specifying the factors by which to downscale (vertical, horizontal).
  - *activation function*: it may vary from a layer to another.

  The number of trainable parameters by this layer is: (output_size x (input_size + 1)). It is basically a fully connected layer of neurons. A linear operation in which every input is connected to every output by a weight and is followed by an activation function (in our case ReLU for the hidden layer and sigmoid for the classification layer)
- **Dropout Layer** aims at regularizing the neural network and avoiding risk of over-fitting. It takes *rate* as hyper-parameter in order to fraction of the input units to drop.
- **Activation function** Defines the signal that a neuron propagates for a given input. Several functions can be used but the most popular one is the Rectified Linear Unitreferred (ReLU)

C. Training

The accuracy of the classification predictions depends mainly on the choice of the model's architecture and its parameters. Here we try to optimize the accuracy by finding the best trade off between variance and bias while reducing the computational cost.

Following the CNN model architecture described above, we started by implementing a baseline model taking on input the dataset of images as it is.

A brief summary of the structure of the model is in (TABLE I):

CNN baseline model	
Layer type	**Configuration**
Input Layer	input_shape=(128, 128 ,1)
Convolution Layer	filters=16, kernel_size=6, padding='same', activation='relu'
Max Pooling Layer	_
Convolution Layer	filters=32, kernel_size=6, padding='same', activation='relu'
Max Pooling Layer	_
Flatten Layer	_
Dense Layer	filters=128, activation='relu'
Dense Layer	filters=1, activation='sigmoid'

TABLE I: *CNN configuration*

- **Training settings:**
  - *epochs*: The number of complete passes (forward and backward) through the training dataset. We chose to work with 20 epochs
  - *batch size*: The number of training examples in one forward/backward pass. The higher the batch size, the more memory will be needed. We chose 128 as batch size.
  - *Optimization algorithm* : The algorithm helps the model weights to converge to optimal values that minimise the loss. As a baseline we chose to work with Stochastic Gradient Descent and from there we try to apply better algorithms such as Adam.

- *Loss function* : Since we are dealing with a binary classification problem, we chose binary cross entropy as a loss function.
- *Metrics*: To evaluate the quality of our model, we chose accuracy as metric.

- **Model optimization:**
  Starting with our initial model, we will try to improve the accuracy by sequentially tuning the following parameters in our model:
  - *Image augmentation*: As our baseline dealt only with re-scaling, now we try to apply the image transformations which we discussed in the image preprocessing part
  - *Optimization algorithm* : We will change the algorithm from sgd to Adam
  - *Adding dropout*: As this method penalizes overfitting, applying dropout reduces bias.
  - *Kernel size*: We will try a range of 3 to 15
  - *Number of convolutional layers* : We will try a range of 1 to 4
  - *Number of neurons in the fully connected layer*: We will set them as 128, 256 and 512
- **Splitting the data:** The data we collected consists of almost 9000 labeled frames from 26 experiments. First, we isolate frames from 4 different selected videos that differ the most from the rest. This will be our validation set. For the optimization part, we run a 5-fold cross validation on the remaining set without shuffling to avoid data leakage as we are working with consecutive frames from the same videos. After finding the optimal parameters, we train the model on the entire training set and we test it on the validation set.

III. Results and Discussion

A. Model Tuning

By going through the analysis of the data with a focus on CNN architecture, we managed to build up performing models. We started from a simple baseline model that ran with validation accuracy of 80.57%. When applying Image transformation, the accuracy of the training model decreased (61%) which is normal due to the enhanced diversity of the data. In order to adapt, we set up Adam as Optimizer to allow faster convergence with validation accuracy of 89.5%. All final cross-validation results are shown in (TABLE II), with the validation accuracy of the other hyper-parameters optimizations in (Fig.4):

B. Additional application: Transfer learning

Transfer Learning is a method used in machine learning that consists of using an already trained model and applying it to solve a similar task to what it was trained for. In our case, we would like to take the model trained for detecting the presence of an outer leg in an image and using it to detect the X point in the same dataset. This can be useful since there are only 3 videos out of 26 in our dataset that contain images where we see the outer leg without the X point, and transfer learning is a useful tool when we have a limited set. Then we can compare it to a completely new model that is trained to detect the X point only. The new model will have the same architecture as before, whereas the transfer learning model will have the old model with the same weights set as non trainable except for the top layer that is used for the classification, in which we reinitialize the weights and set them as trainable. From the transfer model we get 94.54% accuracy and from the new model we get 94.92%. The results are very close.

CNN baseline model	
Model	**Accuracy**
Baseline	80.57%
Image transformation	61%
Setting Adam as optimizer	89.5%
Adding Dropout layer	90.6%
Setting kernel_size to 11	91.19%
Keeping number of convolutional layers	_
Keeping number of neurons in the fully connected layer	_

TABLE II: *Results of optimization*

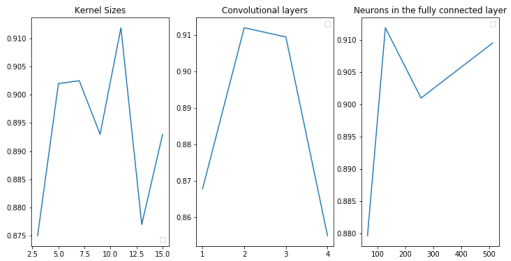


Fig. 4: *Validation Accuracy for different optimizations*

IV. Conclusion

During this project, we had the chance to work on a real world dataset. We collaborated with SPC laboratory which gave us access to their data. We first tried to understand the challenge behind their project and how machine learning can improve their work. We decided that CNN is the most convenient algorithm for our application. We tested multiple configurations and parameters in order to increase our accuracy. As a result we kept a final model with validation accuracy 95.43%. It is thus a very satisfactory performance for identifying divertor legs from a set of images.

Nevertheless, our model could have been more performing if we had a wider range of experiment settings collected in the data. We detected bias in our dataset, which got more relevant when training our CNN model. Therefore we applied data transformations and optimized our model on the augmented dataset in order to cover all possible input settings.

REFERENCES

[1] "Josefine Henriette Elise Proll". *"Trapped-particle instabilities in quasi-isodynamic stellarators"*. PhD thesis, "Eindhoven University of Technology", 2013.

[2] Nuclear power. https://en.wikipedia.org/wiki/Fusion_power#Mechanism. Accessed: 2019-12-05.

[3] "Peter Donnel.". *"Impurity transport in tokamak plasmas: gyrokinetic study of neoclassical and turbulent transport"*. PhD thesis, "Aix Marseille Université", 2018.

[4] S. I. Krasheninnikov and A. I. Smolyakov. Current convective instability in detached divertor plasma. *Phys. Plasmas*, 23(9):092505, 2016.

[5] Nuclear fusion reactor monitored by ten-camera multispectral imaging system. https://www.vision-systems.com/non-factory/article/14034794/nuclear-fusion-reactor-tokamak-ximea-cameras-differ-research. Accessed: 2019-12-15.