1D-Convolutional Neural Network Architecture for Generalized Time-Segmentation Tasks

Koushik Chennakesavan *
University of Texas at Austin, Austin TX, 78702

Magnus A. Haw†,
Analytical Mechanics Associates, Moffett Field CA, 94035

Alexandre Quintart ‡
Flying Squirrel, Brussels, 1150, Belgium

Time segmentation of experimental data is a common and often difficult task. Consequently, it is of interest to automate this type of segmentation to reduce manual inputs, which are labor intensive and less consistent. However, simple thresholding algorithms are often insufficiently robust due either to noise or inconsistent data. This paper proposes a simple 1D convolutional neural net (CNN) architecture as a generalized solution for typical time segmentation tasks. The layer architecture, training methods, and methods for simple customization are described as well as the results of application to three separate arc jet data streams: facility condition segmentation, video highlight segmentation, and calorimeter time-series segmentation.

I. Nomenclature

CNN	=	convolutional neural network
RNN	=	recurrent neural network
RCNN	=	recurrent convolutional neural network
ML	=	machine learning
1D	=	one dimensional
NASA	=	National Aeronautics and Space Administration
ARC	=	Ames Research Center
arc jet	=	facility which generates high enthalpy supersonic flows (plasma wind tunnel)
AHF	=	Aerodynamic Heating Facility at NASA Ames
IHF	=	Interaction Heating Facility at NASA Ames
PTF	=	Panel Test Facility at NASA Ames

II. Introduction

The continual improvement of digital data acquisition capacity (higher resolution, faster data rates, smaller sensors, higher channel throughput, etc.) means that an ever larger percentage of measured data will be in the form of time series. This advent of "Big Time Data" has both benefits and caveats. The benefit is that more data allows for more complete analysis and improved statistics. The caveat is that additional tools are often needed to process the mountains of data. In the case of time series data, segmentation or labeling of important intervals, is a common processing need. This typically consists of finding transitions between states or behaviours and then labeling intervals associated with each state. For many systems, inconsistent values, intermittent noise, and system complexity make identifying these transitions robustly a challenge for basic thresholding algorithms (i.e., finding edges based on ranges of proportional values, derivatives, and/or integrals). Consequently, many time segmentation tasks are performed by humans which is

*Computer Engineering Undergraduate Student at the University of Texas at Austin
†Plasma physicist, Thermophysics Materials Branch, NASA Ames Research Center, MS 234-4, Member AIAA, magnus.haw@nasa.gov
‡Aerospace Engineer, Flying Squirrel, Belgium, alex@flying-squirrel.space

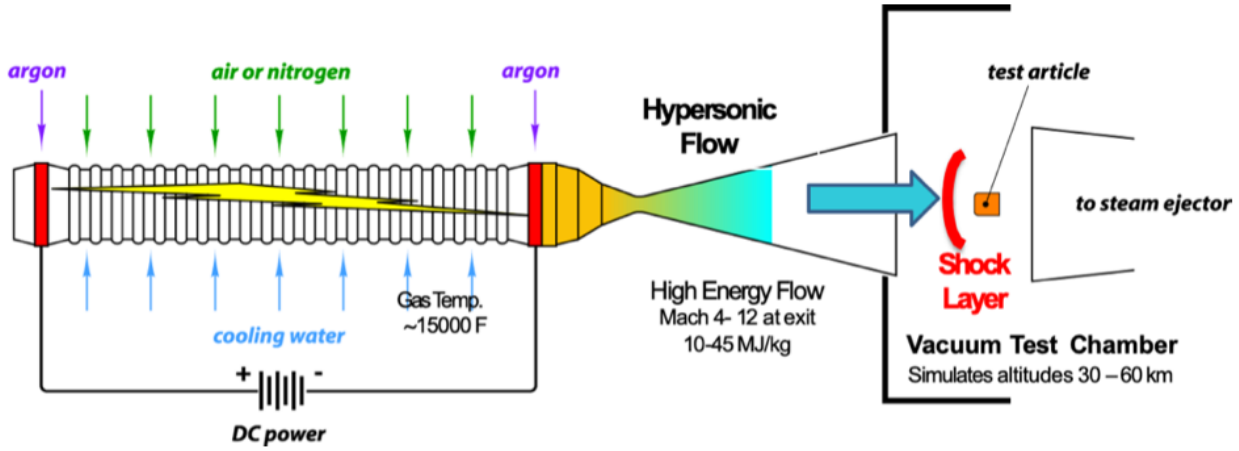


Fig. 1 A schematic of the 60 MW segmented arc jet used in the Interacting Heating Facility (IHF) at NASA Ames Research Center [1]

labor intensive, slow, and often inconsistent. For any repetitive segmentation task, manual input is nearly always a processing bottleneck and as such, automation of these tasks is of significant interest.

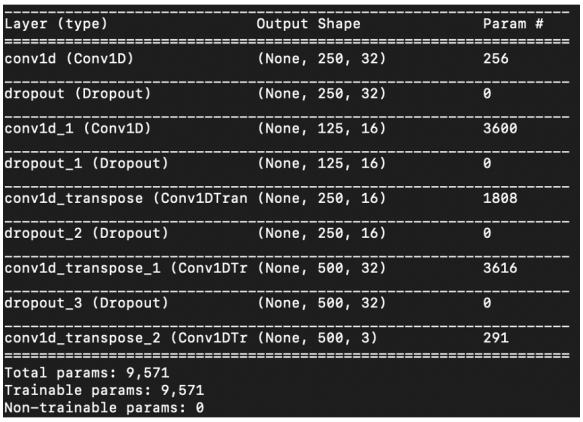
In this paper we propose a new 1-dimensional convolutional neural net (1D CNN) architecture as a generalized solution to automate many time segmentation tasks. The network architecture (Section III.B) is fully convolutional (can handle arbitrary input lengths), performs semantic segmentation (continuous classification of points rather than a single classification of a given time series), and can be easily extended to multiple input time series ($n \times m$). We believe this provides a general solution for most time segmentation problems, provided that sufficient training data can be obtained or generated. The architecture is also attractive compared with other segmentation algorithms (e.g., thresholding, recurrent neural networks, hidden Monte Carlo models) since the model has fast processing speed (<1 sec for 10^4 points), small memory ($\ll$ 1 MB), equivalent or better accuracy, and is fast to train (< 5 min for 6000 samples).

The principal test cases and results (Section IV) are on data from the NASA Ames Research Center (ARC) arc jet facilities so a brief description is provided for context. The Ames arc jets are multi-million dollar facilities built for ground testing thermal protection materials (heat shields) using high enthalpy flows (5-47 MJ/kg). These high enthalpy flows are similar to the high heat fluxes and pressures experienced by a spacecraft during atmospheric entry (See Figure 1). These facilities have numerous customers including NASA projects (e.g., Mars Sample Return Project), Department of Defense, and various aerospace companies. During testing, the facilities measure hundreds of data channels from valves to thermocouples to video cameras to pressure sensors. These data are then parsed and analyzed by facility staff before delivering to the customer. The facility data processing pipeline currently involves a number of manual input steps. Automating these steps was the primary motivation behind the development of the neural network architecture presented in the paper. Section IV describes the successful application of the CNN to three different segmentation problems: arc jet condition segmentation, video clip segmentation, and calorimeter insertion segmentation.

III. CNN Implementation

A. Time segmentation problem statement

As mentioned before, the use case of the CNN is to automate segmentation tasks within three cases: video clip, calorimeter, and condition segmentation. In all three of these scenarios, we deal with a single time series distributed across unequal intervals. Each interval is classified with a specific label or class depending on the context. This time series comprises of the input for our CNN. Our output is a continuous classification for an arbitrarily determined time series. An important note is that our CNN should be capable of classification with noise interference, which should also be appropriately addressed in training. Another important feature that is important to address would be the scalability of the CNN, mainly to address multiple inputs and more complex time series.



Layer (type)	Output Shape	Param #
conv1d (Conv1D)	(None, 250, 32)	256
dropout (Dropout)	(None, 250, 32)	0
conv1d_1 (Conv1D)	(None, 125, 16)	3600
dropout_1 (Dropout)	(None, 125, 16)	0
conv1d_transpose (Conv1DTran	(None, 250, 16)	1808
dropout_2 (Dropout)	(None, 250, 16)	0
conv1d_transpose_1 (Conv1DTr	(None, 500, 32)	3616
dropout_3 (Dropout)	(None, 500, 32)	0
conv1d_transpose_2 (Conv1DTr	(None, 500, 3)	291

Total params: 9,571
Trainable params: 9,571
Non-trainable params: 0

Fig. 2 Convolutional neural network layer structure: diagram of network layers including dropout layers (only used during training).

B. Layer Architecture

The CNN used in the Time Segmentation Tasks use appropriate filters to detect features. Image inputs in Computer Vision and Image Processing applications can be commonly separated into color layers. Take, for example, a RGB image that can be separated into 3 individual layers pertaining to it. This would be rather simple to process but the complexity becomes significantly higher when processing images with larger color schemes and better resolution quality.

The Convolution Layer helps to reduce an image layer through matrix multiplication. A kernel, or filter, matrix can be selected to multiply the portion of the image matrix layer. This smaller filter matrix traverses left to right across the image portions and creates a smaller result matrix. To enforce traversal length, a stride length value can be defined. This process helps to collect important image features across layers and reduce the size of the layer into the kernel matrix.

Similarly, the Pooling Layer reduces image layer size but for the purpose of reducing computational resources. In addition, it also extracts features that are significant to the image irrespective of the position and rotation of the image, giving an extra layer of validation. Two common forms of Pooling are Max Pooling and Averaged Pooling which take the max value of the kernel and the average value of the kernel. Since Max Pooling performs de-noising as well as dimensional reduction, it is largely the preferred form of pooling across the industry.

These two layers are the foundation to the learning of the network and more layers can be added for improved details at the cost of computational power. To prevent non-linearity, an activation function can be added intermittently throughout the process of stacking convolution and pooling layers. To tie it all together, a Fully Connected Layer is used to compute a classification score from the last layer based on the potential classes specified.

C. Training

Training is performed in a three-step process for each of the three applications tested. Initial training and optimization is done with artificially generated data to simplify meta-parameter optimization. After reasonable performance is attained on the artificial data, the model is validated against real data. The final model is trained with a combined set of artificial and real data. This process is greatly facilitated by the relative ease of generating realistic time-series data as compared with 2D or 3D data.

D. Model metrics

To evaluate our CNN, we can use a variety of metrics to garner accurate responses. Primarily, accuracy can be used to get a ratio of correctly classified image subsections. To calculate this, we take a look at the sum of true positives and negatives divided by the sum of all labeled positives and negatives. Another potential metric is prediction which helps to quantify how many positive classifications are from true positive by taking the true prediction rate and dividing by the sum of false and true positive rate. Another tool that can give us information of the performance of our model is a Receiver Operating Characteristic Curve (ROC Curve). A ROC Curve helps to model performance of our model across different classification thresholds. This helps us identify how well our model can distinguish between two different potential labels with a potential max optimization for the ROC Score being 1.0.

E. Caveats and simple extensions

Our CNN is not infallible, however, and there are some factors to be mindful of in larger scale deployment. The size of the inputs are not a multiple of four and thus will not result in the same size output lengths. This is due to the upsampling and downsampling operations happening inside the network. In a simple and rather elegant solution, we can create an 'input shape' parameter when instantiating the model. We also note that in extended tests, the training and test data should have similar time resolution. This is an intended effect and should hold true regardless of the length of the series.

To improve the model, we tried experimenting with more CNN layers, both convolution and pooling. However, with the scope and size of our project, we found that adding more layers did not have a positive effect as we had already spatially merged our input data to the desired size. Lastly, we note that our classification categories can be easily adjusted by using the filters parameters in the last layer. By default, it is set to 2.

IV. Case studies

The 1D CNN architecture described in previous sections was applied to three separate segmentation tasks for arc jet data: facility condition segmentation, video highlight segmentation, and calorimeter insertion segmentation. For each of these cases, the basic model was tweaked and retrained on data specific for each task.

A. Arc Jet Condition Segmentation

The arc jet facilities typically have four principle input variables: arc current [A], main gas flow [g/s], add gas flow [g/s], and argon gas flow [g/s]. The arc current provides the heating mechanism and the various flows provide a stable gas flow through the facility. However, these inputs are often independently varied during a single test and can be quite noisy (see Figure 3) so the start/stop times for a given input condition are difficult to extract using thresholding algorithms. Consequently, it is an appropriate task to solve using ML segmentation.

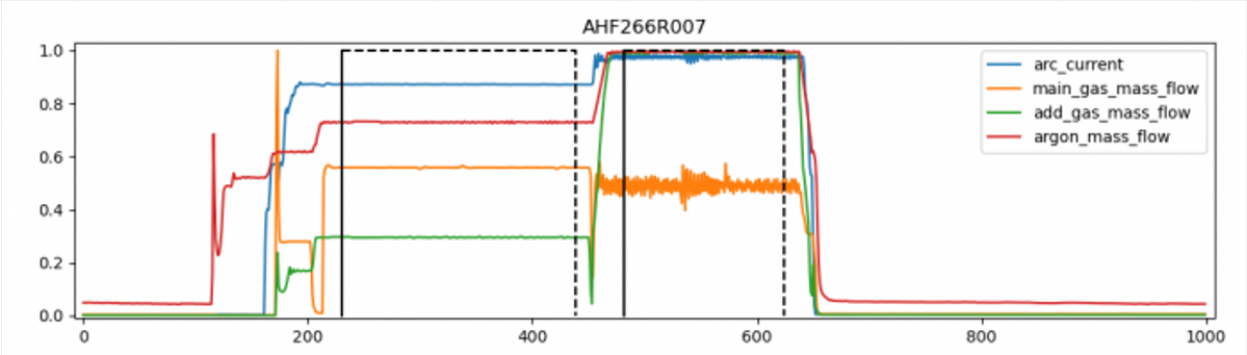


Fig. 3 Condition time-series segmentation

1. CNN initialization

The CNN was initialized with the minimum number of layers, a single normalized time-series input and three classes (on, off, and transition). The condition segmentation is accomplished by classifying each input time series separately using the CNN and then taking the intersection of all the 'on' condition intervals. Three classes were chosen for this context to provide a buffer region between 'on' and 'off' states as the steady state performance is of primary interest for statistical tracking of input conditions.

2. Training

Artificial training data was generated using a sum of one to three step functions with random heights, widths, and start times (1000 points per series, 600 series). These initial traces were then smoothed with a random window length between 5 and 50, and then uniform random noise with a random amplitude between 0.05 and 0.25 was added. 150 additional arc jet time series traces were manually segmented and added to the final training set. The final model was

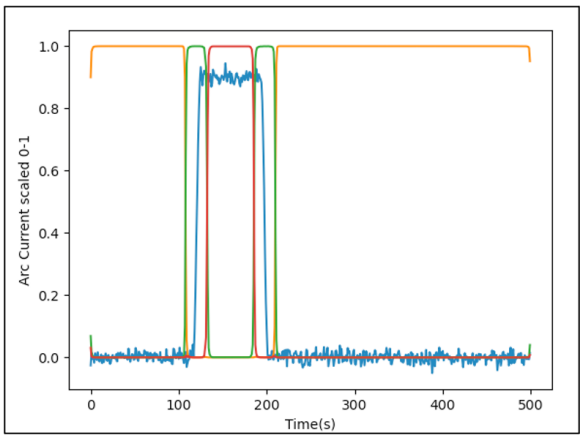


Fig. 4 Artificial data generation (blue) and class labeling (orange, green, red) for condition segmentation. Orange lines indicate 'off' class, green lines indicate 'transition' class, and red lines indicate 'on' class.

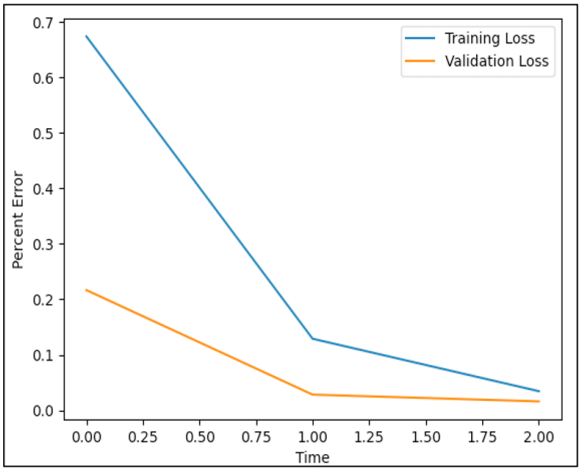


Fig. 5 Training and validation loss over first two epochs

trained on this combined set of 750 traces for 5 epochs where the artificial data was randomly regenerated for each epoch. Model training error was roughly evaluated to around 1-1.5 percent.

B. Video Highlight Segmentation

Arc jets typically have several video feeds for each test which comprise a majority of the total data collected (> 10 GB per run). This footage is currently parsed by hand to extract the relevant clips from a continuous recording of the whole test. Since the video from a single test can last for hours, this can amount to a large number of clips which need to be manually processed, cropped, and labeled: 1 test x 2-8 runs x 3-10 model insertions per run x 3-4 camera views (18-320 clips). Consequently, automating this process can save a large amount of manual labor.

To approach this problem, video sequences were first parsed using the Python API for OpenCV [2] to extract the average pixel intensity each frame. These 1D time series can then be easily classified using the 1D CNN architecture. Figure 6 shows an initial classification of a many-insertion time series.

1. CNN modifications

The basic CNN described above was modified to classify only two states ('on', 'off') instead of the original three states. This better identified very short insertions (e.g., calorimeter insertions) in the same time series as longer insertions (material samples).

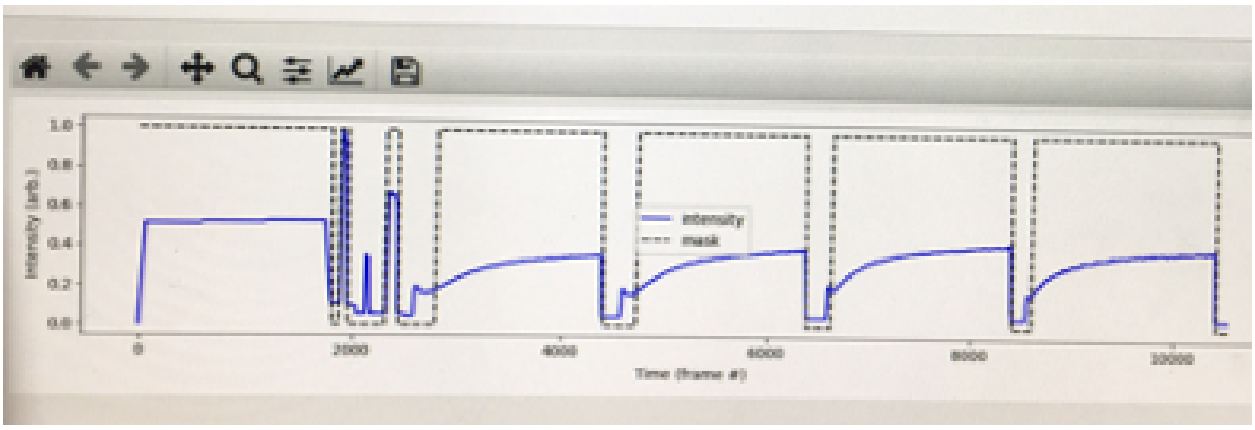


Fig. 6 Video highlight segmentation: typical time series of video mean pixel brightness (blue) with clip segmentation indicators (black dotted).

2. Training

Artificial training data was generated using similar methods to the condition segmentation: randomized step functions were generated with added noise. In addition, several starting edges were randomly added to the step functions: logistic rise, starting spike, slow linear rise, fast linear rise. This variety was sufficient to give fairly good performance on the 20 or so videos available.

C. Calorimeter Insertion Segmentation

Copper slug calorimeters are used to calibrate the heat flux and stagnation enthalpy of facility conditions and are typically inserted at least once on most runs. Consequently, it's can be onerous to manually identify the start and stop times for each calorimeter insertion. In this case however, the regions of interest are just the rising edges instead of top hat windows.

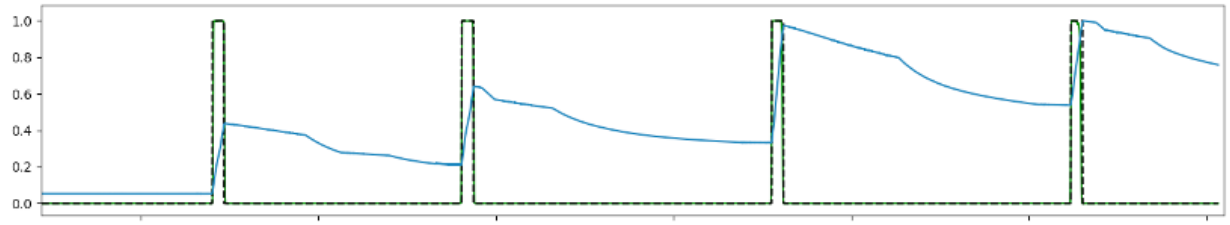


Fig. 7 Calorimeter time-series segmentation: plot showing segmentation of calorimeter insertions.

1. CNN modifications

No modifications were needed to the base CNN described in earlier sections.

2. Training

The artificial training data was created assuming a set of 3 insertions each with a linearly rising edge and an exponential decay profile post-insertion. The amplitudes, slopes, decay rates, and start times were all randomized for these generated time series.

3. Model metrics

Listed below are the metrics used for the CNN:

1. Model Structure:

- 1D CNN Layer with input size (500, 1)
- Dropout Layer with rate of 0.25
- 1D CNN Layer
- Dropout Layer with rate of 0.25
- 1D Transpose CNN Layer
- Dropout Layer with rate of 0.25
- 1D Transpose CNN Layer
- Dropout Layer with rate of 0.25
- 1D Transpose CNN Layer

2. Model Fit:
- 3 epochs
- Batch Size of 128
- Validation Split of 0.1
- Monitor val loss using Keras EarlyStopping

V. Conclusions

Based on the aforementioned research, we can summarily say that 1D fully Convolutional Neural Network (CNN) provide a general framework for our time-segmentation tasks with strong advantages. In regards to memory, size, and complexity, our CNN model is better in all three metrics when compared to similar models like RNN's and RCNN. The model is also dynamic enough to be used across three different scenarios: Calorimeters, Video Feeds, and Arc Jet condition. While some features were slightly modified for the case at hand, we see great success with the model being able to detect different segments across all these use cases. With data being a pivotal part of the twenty-first century, the use cases for models similar to our Convolutional Neural Network only continue to grow within the Aerospace industry and beyond.

Acknowledgments

This work was funded by the NASA Ames Center Innovation Fund (supported by NASA's Space Technology Mission Directorate), and internal research and development funds from the TSM, TSF, and TSS branches of the Thermophysics Division at NASA Ames.

References

[1] Calomino, A., Bruce, W., Gage, P., Horn, D., Mastaler, M., Rigali, D., Robey, J., Voss, L., Wahlberg, J., and Williams, C., "Evaluation of the NASA Arc Jet Capabilities to Support Mission Requirements," *NASA Technical Reports Server*, , No. 20110007355, 2010. URL https://ntrs.nasa.gov/citations/20110007355.

[2] Itseez, "Open Source Computer Vision Library," https://github.com/itseez/opencv, 2015.