NMT-Keras: a Very Flexible Toolkit with a Focus on Interactive NMT and Online Learning

Álvaro Peris, Francisco Casacuberta

Pattern Recognition and Human Language Technology Research Center, Universitat Politècnica de València

Abstract

We present NMT-Keras, a flexible toolkit for training deep learning models, which puts a particular emphasis on the development of advanced applications of neural machine translation systems, such as interactive-predictive translation protocols and long-term adaptation of the translation system via continuous learning. NMT-Keras is based on an extended version of the popular Keras library, and it runs on Theano and Tensorflow. State-of-the-art neural machine translation models are deployed and used following the high-level framework provided by Keras. Given its high modularity and flexibility, it also has been extended to tackle different problems, such as image and video captioning, sentence classification and visual question answering.

1. Introduction

To easily develop new deep learning models is a key feature in this fast-moving field. We introduce NMT-Keras[1], a flexible toolkit for neural machine translation (NMT), based on the Keras library for deep learning (Chollet et al., 2015). Keras is an API written in Python which provides high-level interfaces to numerical computation libraries, such as Theano (Theano Development Team, 2016) or Tensorflow (Abadi et al., 2016). Keras is well-structured and documented, with designing principles that make it modular and extensible, being easy to construct complex models.

Following the spirit of the Keras library, we developed NMT-Keras, released under MIT license, that aims to provide a highly-modular and extensible framework to

[1] https://github.com/lvapeab/nmt-keras

NMT. NMT-Keras supports advanced features, such as support to interactive-predictive NMT (INMT) (Barrachina et al., 2009), continuous learning and active learning strategies. An additional goal, is to ease the usage of the library, but allowing the user to configure most of the options involving the NMT process.

Since its introduction (Sutskever et al., 2014; Cho et al., 2014), NMT has advanced by leaps and bounds. Several toolkits currently offer fully-fledged NMT systems. Among them, we can highlight OpenNMT (Klein et al., 2017), Tensor2Tensor (Vaswani et al., 2018) or Nematus (Sennrich et al., 2017). NMT-Keras differentiates from them by offering interactive-predictive and long-term learning functionalities, with the final goal of fostering a more productive usage of the NMT system.

This document describes the main design and features of NMT-Keras. First, we review the deep learning framework which is the basis of the toolkit. Next, we summarize the principal features and functionality of the library. We conclude by showing translation and INMT results.

2. Design

NMT-Keras relies on two main libraries: a modified version of Keras[2], which provides the framework for training the neural models; and a wrapper around it, named Multimodal Keras Wrapper[3], designed to ease the usage of Keras and the management of data. These tools represent a general deep learning framework, able to tackle different problems, involving several data modalities. The problem of NMT is an instantiation of the sequence-to-sequence task, applied to textual inputs and outputs. NMT-Keras is designed to tackle this particular task. In this section, we describe these components and their relationships in our framework.

2.1. Keras

As mentioned in Section 1, Keras is a high-level deep learning API, which provides a neat interface to numerical computation libraries. Keras allows to easily implement complex deep learning models by defining the layers as building blocks. This simplicity, together with the quality of its code, has made Keras to be one of the most popular deep learning frameworks. It is also well-documented, and supported by a large community, which fosters its usage.

In Keras, a model is defined as a directed graph of layers or operations, containing one or more inputs and one or more outputs. Once a model has been defined, it is compiled for training, aiming to minimize a loss function. The optimization process is carried out, via stochastic gradient descent (SGD), by means of an optimizer.

[2] https://github.com/MarcBS/keras

[3] https://github.com/lvapeab/multimodal_keras_wrapper

Once the model is compiled, we feed it with data, training it as desired. Once a model is trained, it is ready to be applied on new input data.

2.2. `Multimodal Keras Wrapper`

The `Multimodal Keras Wrapper` allows to handle the training and application of complex `Keras` models, data management (including multimodal data) and application of additional callbacks during training. The wrapper defines two main objects and includes a number of extra features:

Dataset: A `Dataset` object is a database adapted for `Keras`, which acts as data provider. It manages the data splits (training, validation, test). It accepts several data types, including text, images, videos and categorical labels. In the case of text, the `Dataset` object builds the vocabularies, loads and encodes text into a numerical representation and also decodes the outputs of the network into natural language. In the case of images or videos, it also normalizes and equalizes the images; and can apply data augmentation.

Model wrapper: This is the core of the wrapper around `Keras`. It connects the `Keras` library with the `Dataset` object and manages the functions for training and applying the `Keras` models. When dealing with sequence-to-sequence models, it implements a beam search procedure. It also contains a training visualization module and a set of ready-to-use convolutional neural networks (CNN) architectures.

Extra: Additional functionalities include extra callbacks, I/O management and evaluation of the system outputs. For computing the translation quality metrics of the models, we use the `coco-caption` evaluation tool (Chen et al., 2015), which provides common evaluation metrics: BLEU, METEOR, CIDEr, and ROUGE-L. Moreover, we modified it[4] to also include TER.

2.3. `NMT-Keras`

The `NMT-Keras` library makes use of the aforementioned libraries, for building a complete NMT toolkit. The library is compatible with Python 2 and 3. The training of NMT models is done with the `main.py` file. The hyperparameters are set via a configuration file (`config.py`), and can also be set from the command line interface. To train a NMT system with `NMT-Keras` is straightforward:

1. Set the desired configuration in `config.py`.
2. Launch `main.py`.

The training process will then prepare the data, constructing the `Dataset` object and the `Keras` model. The default models implemented in `NMT-Keras` are an attentional RNN encoder–decoder (Bahdanau et al., 2015; Sennrich et al., 2017), and the

[4] `https://github.com/lvapeab/coco-caption`

Transformer model (Vaswani et al., 2017). Once the model is compiled, the training process starts, following the specified configuration. For translating new text with a trained model, we use beam search.

3. Features

As we keep our `Keras` fork constantly up-to-date with the original library, `NMT-Keras` has access to the full `Keras` functionality, including (but not limited to):

Layers: Fully-connected layers, CNN, (bidirectional) recurrent neural networks (RNN) (long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) and gated recurrent units (GRU) (Cho et al., 2014)), embeddings, noise, dropout (Srivastava et al., 2014) and batch normalization (Ioffe and Szegedy, 2015) layers.

Initializers: The weights of a model can be initialized to a constant value, to values drawn from statistical distributions or according to the strategies proposed by Glorot and Bengio (2010), He et al. (2015) and Klambauer et al. (2017).

Optimizers: A number of SGD variants are implemented: vanilla SGD, RMSProp (Tieleman and Hinton, 2012), Adagrad (Duchi et al., 2011), Adadelta (Zeiler, 2012), Adam and Adamax (Kingma and Ba, 2014). The learning rate can be scheduled according to several strategies (linear, exponential, "noam" (Vaswani et al., 2017)).

Our version of `Keras` implements additional layers, useful for sequence-to-sequence problems:

Improved RNNs: All RNN architectures can be used in an autoregressive way, i.e. taking into account the previously generated token. They also integrate attention mechanisms, supporting the *add* (Bahdanau et al., 2015) and *dot* (Luong et al., 2015) models.

Conditional RNNs: Conditional (Sennrich et al., 2017) LSTM/GRU layers, which consist in a cascaded application of LSTM/GRU cells, with an attention model in between.

Multi-input RNNs: LSTM/GRU networks with two and three different inputs and independent attention mechanisms (Bolaños et al., 2018).

Transformer layers: Multi-head attention layers, positional encodings and position-wise feed-forward networks (Vaswani et al., 2017).

Finally, `NMT-Keras` supports a number of additional options. Here we list the most important ones, but we refer the reader to the library documentation[5] for an exhaustive listing of all available options:

Deep models: Deep residual RNN layers, deep output layers and deep fully-connected layers for initializing the state of the RNN decoder.

[5] `https://nmt-keras.readthedocs.io`

Embeddings: Incorporation of pre-trained embeddings in the NMT model and embedding scaling options.

Regularization strategies: Label smoothing (Szegedy et al., 2015), early-stop, weight decay, doubly stochastic attention regularizer (Xu et al., 2015) and a fine-grained application of dropout.

Search options: Normalization of the beam search process by length and coverage penalty. The search can be also constrained according to the length of the input/output sequences.

Unknown word replacement: Replace unknown words according to the attention model (Jean et al., 2015). The replacement may rely on a statistical dictionary.

Integration with other tools: Support for `Spearmint`[6], for Bayesian optimization of the hyperparameters and `Tensorboard`, the visualization tool of `Tensorflow`.

Apart from these model options, `NMT-Keras` also contains scripts for ensemble decoding and generation of N-best lists; sentence scoring, model averaging and construction of statistical dictionaries (for unknown words replacement). It also contains a client-server architecture, which allows to access to `NMT-Keras` via web. Finally, in order to introduce newcomers to `NMT-Keras`, a set of tutorials are available, explaining step-by-step how to construct a NMT system.

3.1. Interactive-predictive machine translation

The interactive-predictive machine translation (IMT) framework constitutes an efficient alternative to the regular post-editing of machine translation; with the aim of obtaining high-quality translations minimizing the human effort required for this (Barrachina et al., 2009). IMT is a collaborative symbiosis between human and system, consisting in an iterative process in which, at each iteration, the user introduces a correction to the system hypothesis. The system takes into account the correction and provides an alternative hypothesis, considering the feedback from the user. Fig. 1 shows an example of the IMT protocol.

Source:		They are lost forever .
Target:		Ils sont perdus à jamais .
0	**MT**	Ils sont perdus pour toujours .
1	**User**	*Ils sont perdus* **à** pour toujours .
	MT	*Ils sont perdus à* jamais .
2	**User**	*Ils sont perdus à jamais .*

Figure 1. IMT session. The user introduces in iteration 1 a character correction (boxed). The MT system modifies its hypothesis, taking into account this feedback.

This protocol has show to be especially effective when combined with NMT (Knowles and Koehn, 2016; Peris et al., 2017b). `NMT-Keras` implements the interactive protocols described by Peris et al. (2017b). Moreover, they can be combined

[6] `https://github.com/HIPS/Spearmint`

with online learning (OL) methods, which allow to specialize the system into a given domain or to the preferences of a user. This approach was described by Peris and Casacuberta (2018) and it is also included in `NMT-Keras`. We built a demo website[7] of these interactive, adaptive systems using the client-server features of the toolkit. Fig. 2 shows a screenshot of the website frontend.

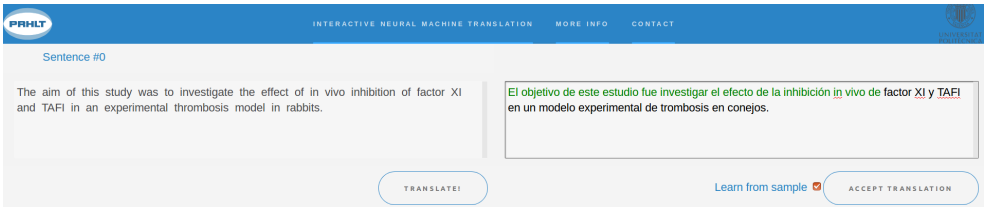

Figure 2. Frontend of the INMT with OL website built with `NMT-Keras`. In the left-side, the source sentence to translate is displayed. The system computes a translation hypothesis, which is located at the right frame. The user can then make changes on this hypothesis. As a keystroke is pressed, the hypothesis is immediately updated, in order to take into account the feedback from the user. In this interaction protocol, the system validates the hypothesis prefix (validated in green), although we expect to support more flexible protocols (Peris et al., 2017b) in a near future. This process is repeated until the desired translation is reached. Then, the user presses the Accept translation button, validating the translation. The system will use this sample to update the model if the "Learn from sample" checkbox is activated. The demo is available at `http://casmacat.prhlt.upv.es/inmt`.

3.2. Tackling other tasks

The modular design of `Keras` and `Multimodal Keras Wrapper` allows to use them to tackle other problems, following the spirit of `NMT-Keras`. These libraries have been used to perform video captioning (Peris et al., 2016), egocentric video captioning considering temporal contexts (Bolaños et al., 2018), text classification (Peris et al., 2017a), food recognition and localization (Bolaños and Radeva, 2016) and visual question answering (Bolaños et al., 2017).

4. Results

We report now results of `NMT-Keras`, assessing translation quality and INMT effectiveness. Due to space restrictions, we report results on two tasks: EU (Barrachina et al., 2009) and Europarl (Koehn, 2005). More extensive results obtained with `NMT-Keras` can be found at Peris and Casacuberta (2018). For the first task, we used the standard partitions. For the Europarl corpus, we used the `newstest2012` and `newstest2013`, as development and test, in order to establish a comparison with other IMT

[7] `http://casmacat.prhlt.upv.es/inmt`.

works (e.g Ortiz-Martínez (2016)). The NMT system was configured as in Peris and Casacuberta (2018). For the sake of comparison, we include results of phrase-based statistical machine translation (PB-SMT), using the standard setup of `Moses` (Koehn et al., 2007). We computed significance tests (95%) via approximate randomization and confidence intervals with bootstrap resampling (Riezler and Maxwell, 2005).

Table 1. Results of translation quality for all tasks in terms of TER [%] and BLEU [%]. Results that are statistically significantly better for each task and metric are boldfaced. Extended results in Peris and Casacuberta (2018).

		TER (↓)		BLEU (↑)	
		PB-SMT	NMT	PB-SMT	NMT
EU	En→De	54.1 ± 1.9	**52.3 ± 1.9**	35.4 ± 2.1	**36.4 ± 2.0**
	En→Fr	41.4 ± 1.6	**38.4 ± 1.5**	47.8 ± 1.7	**50.4 ± 1.6**
Europarl	En→De	**62.2 ± 0.3**	63.1 ± 0.4	19.5 ± 0.3	**20.0 ± 0.3**
	En→Fr	56.1 ± 0.3	**55.0 ± 0.3**	26.5 ± 0.3	**27.8 ± 0.3**

Table 1 shows the translation quality. `NMT-Keras` outperformed `Moses`, obtaining significant TER and BLEU improvements almost every language pair. Only in one case `Moses` obtained better TER than `NMT-Keras`.

4.1. Interactive NMT

We evaluate now the performance of `NMT-Keras` on an IMT scenario. We are interested in measuring the effort that the user must spend in order to achieve the desired translation. Due to the prohibitive cost that an experimentation with real users conveys, the users were simulated. The references of our datasets were considered to be the desired translations. The amount of effort was measured according to keystroke mouse-action ratio (KSMR) (Barrachina et al., 2009), defined as the number of keystrokes plus the number of mouse-actions required for obtaining the desired sentence, divided by the number of characters of such sentence.

Table 2. Effort required by INMT systems compared to the state-of-the-art (SOTA), in terms of KSMR [%]. † refers to Ortiz-Martínez (2016), ‡ to Barrachina et al. (2009). Extended results in Peris and Casacuberta (2018).

		KSMR [%] (↓)	
		INMT	SOTA
EU	En→De	19.8 ± 0.5	30.5 ± 1.1‡
	En→Fr	16.3 ± 0.4	25.5 ± 1.1‡
Europarl	En→De	32.9 ± 0.2	49.2 ± 0.4†
	En→Fr	29.8 ± 0.2	44.4 ± 0.5†

Table 2 shows the performance in KSMR (%) of the INMT systems. We also compare these results with the best results obtained in the literature for each task. All

INMT systems outperformed by large the other ones. Again, we refer the reader to Peris and Casacuberta (2018) for a larger set of experiments, including OL.

5. Conclusions

We introduced `NMT-Keras`, a toolkit built on the top of `Keras`, that aims to ease the deployment of complex NMT systems by having a modular and extensible design. `NMT-Keras` has a strong focus on building adaptive and interactive NMT systems; which leverage the effort reduction of a user willing to obtain high-quality translations. Finally, its flexibility allows the `NMT-Keras` framework to be applied directly to other problems, such as multimedia captioning or sentence classification.

Acknowledgements

Much of our `Keras` fork and the `Multimodal Keras Wrapper` libraries were developed together with Marc Bolaños. We also acknowledge the rest of contributors to these open-source projects. The research leading this work received funding from grants PROMETEO/2018/004 and CoMUN-HaT - TIN2015-70924-C2-1-R. We finally acknowledge NVIDIA Corporation for the donation of GPUs used in this work.

Bibliography

Abadi, Martín, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. TensorFlow: A System for Large-Scale Machine Learning. In *Proceedings of USENIX-SOSDI*, volume 16, pages 265–283, 2016.

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv:1409.0473*, 2015.

Barrachina, Sergio, Oliver Bender, Francisco Casacuberta, Jorge Civera, Elsa Cubel, Shahram Khadivi, Antonio Lagarda, Hermann Ney, Jesús Tomás, Enrique Vidal, and Juan-Miguel Vilar. Statistical Approaches to Computer-Assisted Translation. *Computational Linguistics*, 35(1):3–28, 2009.

Bolaños, Marc and Petia Radeva. Simultaneous food localization and recognition. In *ICPR*, pages 3140–3145, 2016.

Bolaños, Marc, Álvaro Peris, Francisco Casacuberta, and Petia Radeva. VIBIKNet: Visual bidirectional kernelized network for visual question answering. In *Proceedings of IbPRIA*, pages 372–380, 2017.

Bolaños, Marc, Álvaro Peris, Francisco Casacuberta, Sergi Soler, and Petia Radeva. Egocentric video description based on temporally-linked sequences. *Journal of Visual Communication and Image Representation*, 50:205–216, 2018.

Chen, Xinlei, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO captions: Data collection and evaluation server. *arXiv:1504.00325*, 2015.

Cho, Kyunghyun, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. In *Proceedings of the Workshop on SSST*, pages 103–111, 2014.

Chollet, François et al. Keras. `https://github.com/keras-team/keras`, 2015. GitHub repository.

Duchi, John, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011.

Glorot, Xavier and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of AISTATS*, pages 249–256, 2010.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of ICCV*, pages 1026–1034, 2015.

Hochreiter, Sepp and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9 (8):1735–1780, 1997.

Ioffe, Sergey and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv:1502.03167*, 2015.

Jean, Sébastien, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. On Using Very Large Target Vocabulary for Neural Machine Translation. In *Proceedings of ACL*, pages 1–10, 2015.

Kingma, Diederik and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.

Klambauer, Günter, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In *NIPS*, pages 971–980, 2017.

Klein, Guillaume, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. OpenNMT: Open-Source Toolkit for Neural Machine Translation. *Proceedings of ACL 2017, System Demonstrations*, pages 67–72, 2017.

Knowles, Rebecca and Philipp Koehn. Neural Interactive Translation Prediction. In *Proceedings of the AMTA*, pages 107–120, 2016.

Koehn, Philipp. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of the MT Summit*, pages 79–86, 2005.

Koehn, Philipp, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of ACL*, pages 177–180, 2007.

Luong, Thang, Hieu Pham, and Christopher D. Manning. Effective Approaches to Attention-Based Neural Machine Translation. In *Proceedings of EMNLP*, pages 1412–1421, 2015.

Ortiz-Martínez, Daniel. Online Learning for Statistical Machine Translation. *Computational Linguistics*, 42(1):121–161, 2016.

Peris, Álvaro and Francisco Casacuberta. Online Learning for Effort Reduction in Interactive Neural Machine Translation. *arXiv:1802.03594*, 2018.

Peris, Álvaro, Marc Bolaños, Petia Radeva, and Francisco Casacuberta. Video Description using Bidirectional Recurrent Neural Networks. In *Proceedings of ICANN*, pages 3–11, 2016.

Peris, Álvaro, Mara Chinea-Ríos, and Francisco Casacuberta. Neural Networks Classifier for Data Selection in Statistical Machine Translation. *The Prague Bulletin of Mathematical Linguistics*, 1(108):283–294, 2017a.

Peris, Álvaro, Miguel Domingo, and Francisco Casacuberta. Interactive neural machine translation. *Computer Speech & Language*, 45:201–220, 2017b.

Riezler, Stefan and John T Maxwell. On some pitfalls in automatic evaluation and significance testing for MT. In *Proceedings of the workshop on MTSE*, pages 57–64, 2005.

Sennrich, Rico, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry, and Maria Nadejde. Nematus: a Toolkit for Neural Machine Translation. In *Proceedings of the Software Demonstrations at EACL*, pages 65–68, 2017.

Srivastava, Nitish, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le. Sequence to Sequence Learning with Neural Networks. In *Proceedings of NIPS*, volume 27, pages 3104–3112. 2014.

Szegedy, Christian, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of CVPR*, pages 1–9, 2015.

Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv:1605.02688*, 2016.

Tieleman, Tijmen and Geoffrey Hinton. Lecture 6.5-RMSProp: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. In *Proceedings of NIPS*, volume 30, pages 5998–6008, 2017.

Vaswani, Ashish, Samy Bengio, Eugene Brevdo, Francois Chollet, Aidan N Gomez, Stephan Gouws, Llion Jones, Łukasz Kaiser, Nal Kalchbrenner, Niki Parmar, et al. Tensor2tensor for neural machine translation. *arXiv:1803.07416*, 2018.

Xu, Kelvin, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. In *Proceedings of ICLR*, pages 2048–2057, 2015.

Zeiler, Matthew D. ADADELTA: An Adaptive Learning Rate Method. *arXiv:1212.5701*, 2012.

Address for correspondence:
Álvaro Peris
lvapeab@prhlt.upv.es
Pattern Recognition and Human Language Technology Research Center,
Universitat Politècnica de València,
Camino de Vera s/n, 46022 Valencia, SPAIN.