The Journal of Open Source Software

captcha: An R package to read, visualize and solve Captchas using torch

Julio Treceni[1] **and Victor Fossaluza**[1]

1 Institute of Mathematics and Statistics, University of São Paulo

DOI:

Software

- Review
- Repository
- Archive

Submitted:
Published:

Summary

The {captcha} package is a toolbox for researches who need to solve text-on-image Captchas (Completely Automated Public Turing test to tell Computers and Humans Apart) for their academic work. The package includes fundamental operations for reading, visualizing, and annotating Captchas. Additionally, it offers modeling capabilities using the torch backend (Falbel & Luraschi, 2023) to fit new models, load pre-trained models, share fitted models, and solve Captchas. Finally, the package provides a streamlined workflow for solving new Captchas by enabling the creation of a new repository with a step-by-step guide.

Statement of need

Captcha (Completely Automated Public Turing test to tell Computers and Humans Apart) is a challenge aimed to identify whether access to a webpage is performed by a human or a robot (Von Ahn, Blum, & Langford, 2004). The challenge is designed to be easy for humans to solve, but difficult for machines.

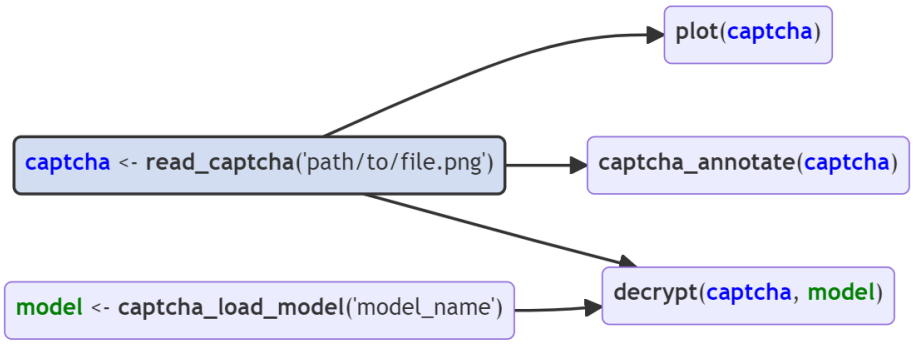
In principle, the use of Captchas increases the security of individuals accessing the internet and protect web systems from abusive use. For human website users, Captchas represent a minor inconvenience, whereas for those performing massive accesses, they pose a significant difficulty.

However, the presence of Captchas on websites is detrimental to society when automated access is necessary to conduct scientific research on publicly available, but not openly accessible data (Molloy, 2011). Several sites storing public data use Captchas and do not provide data in an open format, increasing significantly the effort needed to produce scientific research.

This package is relevant to science for two reasons: technical feasibility and practical importance. With regard to technical feasibility, the package provides the necessary tools to develop solutions for new Captchas, including tasks of reading, annotating and fitting models using Torch framework. With regard to practical importance, the package provides a list of Captchas that have already been solved using deep learning techniques (LeCun, Bengio, & Hinton, 2015). The package release page provides open access to datasets and fitted models for various Captchas.

Basic usage

The basic usage of {captcha} involves the functions `read_captcha()`, `plot()`, `captcha_annotate()`, `captcha_load_model()` and `decrypt()`. The diagram below summarizes the relationships between these functions. The arrows indicate the dependency of functions on objects generated by other functions.



The `read_captcha()` function reads a character vector of image files and stores them in memory. Behind the scenes, the function uses the {magick} (Ooms, 2021) package to deal with the types of files that may appear (JPEG, PNG, among others).

```
library(captcha)
example <- "dados_tjmg.jpeg"
captcha <- read_captcha(example)
captcha
```

```
##   format width height colorspace matte filesize density
## 1   JPEG   100     50       sRGB FALSE     4530   72x72
```

The function returns an object of class `captcha`, which can be used by other methods. Internally, is a list with three elements: `$img`, which contains the image read from the {magick} package; `$lab`, which contains the image label (by default, `NULL`); and `$path`, which contains the path of the image.

The `plot()` function is a method of class S3 from base R. The function facilitates the visualization of Captchas. The function receives a list of images (obtained with the `read_captcha()` function) and displays the Captcha visually.

```
example <- "dados_tjmg.jpeg"
captcha <- read_captcha(example)
plot(captcha)
```

The `captcha_annotate()` function annotates a Captcha image, either manually or automatically. It modifies the image path and adds the text `_label` to the end of the file path. The function returns a vector with the paths of the modified files. The `labels=` parameter can handle situations where one knows the Captcha label. For example, a workflow that uses an oracle might provide the label automatically. When the label doesn't exist, the `captcha_annotate()` function opens the prompt for classification and shows the image using `plot()`.

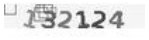
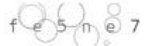
The `decrypt()` function returns a label for an image using a fitted model. The function takes two arguments: `file=`, which can be either the file path or a captcha class object, and `model=`, which contains an object of class luz_module_fitted, fitted using the {luz} package.

```
model <- captcha_load_model("cadesp")
img <- "dados_cadesp.jpg"
captcha <- read_captcha(img)
plot(captcha)
```

```
decrypt(captcha, model)
```

```
## [1] "dwyy"
```

There are several fitted models for several different Captchas accessible through the `{captcha}` package. It is possible to load a trained model using the `captcha_load_model()` function. The `path=` parameter contains either the path for a fitted model or a string with the name of a released model, like `"rfb"`. Fitted models are stored in the `{captcha}` package repository releases, which can be downloaded using the {piggyback} package (Boettiger, 2018). Currently, the Captchas with available fitted models are `trf5`, `tjmg`, `trt`, `esaj`, `jucesp`, `tjpe`, `tjrs`, `cadesp`, `sei` and `rfb`. The table below describes the models and their accuracy.

Name	Example	Description	Accuracy
cadesp		Centro de Apoio ao Desenvolvimento da Saude Publica	96.37%
esaj		Tribunal de Justica da Bahia	94.50%
jucesp		Junta Comercial de Sao Paulo	89.88%
rfb		Receita Federal	95.70%
sei		Sistema Eletronico de Informacoes - ME	77.25%
tjmg		Tribunal de Justica de Minas Gerais	98.35%
tjpe		Tribunal de Justica de Pernambuco	91.88%
tjrs		Tribunal de Justica do Rio Grande do Sul	99.57%
trf5		Tribunal Regional Federal 5	98.77%
trt		Tribunal Regional do Trabalho 3	98.50%

Custom model

The `{captcha}` package provides a basic interface for fitting custom models from a fully labeled data. Annotation can be done manually using the `captcha_annotate()` function presented earlier or with another method developed by the user. The model uses a convolutional neural network architecture, similar to the LeNet-5 model (LeCun et al., 1995).

The `captcha_fit_model()` function fits a model from a folder with annotated images. It also has some parameters related to the neural network. The function returns a fitted model of class `luz_module_fitted` (Falbel, 2022), which can be saved to disk using `luz_save()`.

The training step of neural networks involves many small adaptations, it was decided to export functions in two depth levels. To address that, the `{captcha}` package also provides a **procedural** approach to fit the model, using a step-by-step described in the advanced guide.

Acknowledgements

We acknowledge contributions from Daniel Falbel, Caio Lente, and Beatriz Milz, and support from Athos Damiani and Fernando Corrêa during the genesis of this project.

References

Boettiger, C. (2018). Managing larger data on a GitHub repository. *Journal of Open Source Software*, *3*(29), 971. doi:10.21105/joss.00971

Falbel, D. (2022). *Luz: Higher level 'API' for 'torch'*. Retrieved from https://CRAN.R-project.org/package=luz

Falbel, D., & Luraschi, J. (2023). *Torch: Tensors and neural networks with 'GPU' acceleration*. Retrieved from https://CRAN.R-project.org/package=torch

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, *521*(7553), 436–444.

LeCun, Y., Jackel, L., Bottou, L., Brunot, A., Cortes, C., Denker, J., Drucker, H., et al. (1995). Comparison of learning algorithms for handwritten digit recognition. *International conference on artificial neural networks* (Vol. 60, pp. 53–60). Perth, Australia.

Molloy, J. C. (2011). The open knowledge foundation: Open data means better science. *PLoS biology*, *9*(12), e1001195.

Ooms, J. (2021). *Magick: Advanced graphics and image-processing in r*. Retrieved from https://CRAN.R-project.org/package=magick

Von Ahn, L., Blum, M., & Langford, J. (2004). Telling humans and computers apart automatically. *Communications of the ACM*, *47*(2), 56–60.