Thessaloniki Machine Learning Meetup

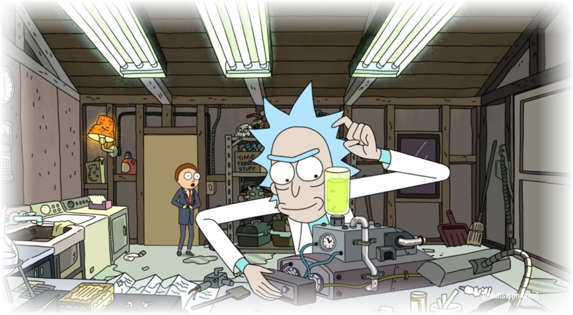
Keras

# DEEP LEARNING MADE EASY WITH KERAS: TEXT, IMAGE AND TIME-SERIES ANALYSIS

Nikolaos Passalis
Aristotle University of Thessaloniki
Department of Informatics
passalis@csd.auth.gr
http://users.auth.gr/passalis

# YET ANOTHER PRESENTATION

A few boring stuff before getting our hands dirty!



# DEEP LEARNING IN A SLIDE

- **Define a model** (usually a neural network)!
- **Gather the data**!
- **Define an objective** that measures how well the model performs a task!
- **Optimize** the model according to this objective!
- **Deploy your model**!

# CASE STUDIES

- Datasets:
  - MNIST dataset (image)
  - Fashion MNIST dataset (image)
  - IMDB Movie reviews dataset (text)
  - Reuters newswire dataset (text)
  - EEG Database (time-series)
- Topics that we will cover:
  - Image classification (MLP, CNNs)
  - Text classification (MLP, CNNs, RNNs)
  - Time-series classification (MLP, RNNs)
  - Using deep features for visualization/data exploration
  - Under-fitting, over-fitting, regularization, ...
  - Practical skills: Monitoring progress, visualizing the data/activations, troubleshooting, ...

# KERAS: DEEP LEARNING MADE EASY

# PREREQUISITES

- Step 0: Install Linux (you can also use Windows, but ...)
- Step 0.5: Install CUDA (if you want GPU acceleration)
- Step 1: Installing the required libraries
- **Install tensorflow and keras**
  - If a CUDA-enabled GPU is available (and CUDA is installed):

    ```
    pip install tensorflow-gpu
    ```
  - Otherwise:

    ```
    pip install tensorflow
    ```
  - Install keras:

    ```
    pip install keras
    ```
- Optionally:

  ```
  pip install h5py jupyter  scikit-learn matplotlib
  ```

# KERAS CONFIGURATION

- Probably you want to leave it as is!
- Located at `~/.keras/keras.json`

  ```
  {
  ”backend”: ”tensorflow”,
  ”epsilon”: 1e-07,
  ”floatx”: ”float32”,
  ”image_data_format”: ”channels_last”
  }
  ```

- You can switch between the three available backends.

# KERAS STRUCTURE

- Keras provides:
  - **Deep Learning building blocks**: fully connected layers, convolutional layers, pooling layers, dropout, recurrent layers
  - **Pretrained models for many well-known networks**
  - **Many different optimizers, loss functions, etc.**
  - **Dataset and preprocessing tools**
  - **Tools for training/evaluation/saving and loading models**
  - **...**
- More or less everything one needs for **getting started with deep learning**
- It is even **used for research purposes**!

# KERAS WORKFLOW

1. Define your model
    - Two different ways: sequential and functional APIs
2. Compile your model (define the optimizer, loss, learning rate schedule, callbacks, etc.)
3. Load the data
4. Train the model
5. Evaluate the model

# SEQUENTIAL MODEL

- The easiest way to define a keras model
- Some more complex models cannot be expressed using the sequential API

```
from keras.models import Sequential
from keras.layers import Dense, Activation

model = Sequential()
model.add(Dense(32, input_dim=784))
model.add(Activation(’relu’))
model.add(Dense(10))
model.add(Activation(’softmax’))
```

# FUNCTIONAL API

Allows for developing more complex models (shared layers, DAGs, …)

```
from keras.layers import Input, Dense
from keras.models import Model

# This returns a tensor
inputs = Input(shape=(784,))

# Define the layers
dense_layer_1 = Dense(32, activation=’relu’)
dense_layer_2 = Dense(64, activation=’relu’)
dense_layer_3 = Dense(10, activation=’softmax’)

# Use them to define the model
x = dense_layer_1(inputs)
x = dense_layer_2(x)
predictions = dense_layer_3(x)

model = Model(inputs=inputs, outputs=predictions)
```

# A JUPYTER NOTEBOOK WORTHS A THOUSAND SLIDES

```
git clone http://github.com/passalis/keras_meetup
```