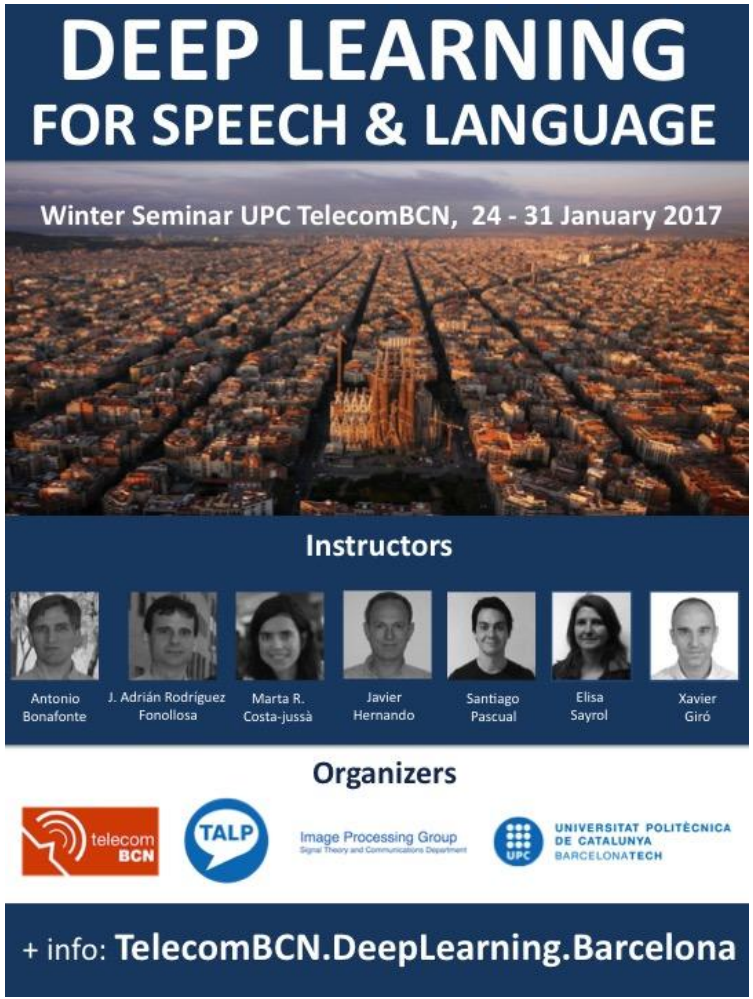


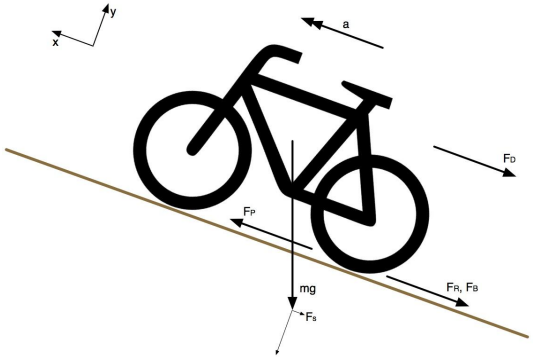
[course site]

Deep Learning with Keras

José Adrián Rodríguez Fonollosa

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Department of Signal Theory and Communications
Image Processing Group

Deep Learnig for Speech and Language



Theory sessions

Keras sessions

Simple projects

Risky projects

Deep Learning Frameworks

Hardware	API	Libraries	Frameworks	High-level Libraries

CPU

GPU (Nvidia) — CUDA — cuDNN cuBLAS

Nvidia stock chart

Caffe (Berkeley Vision)

CNTK (Microsoft)

Tensorflow (Google)

Theano (Univ. de Montréal)

Torch (Facebook et al.)

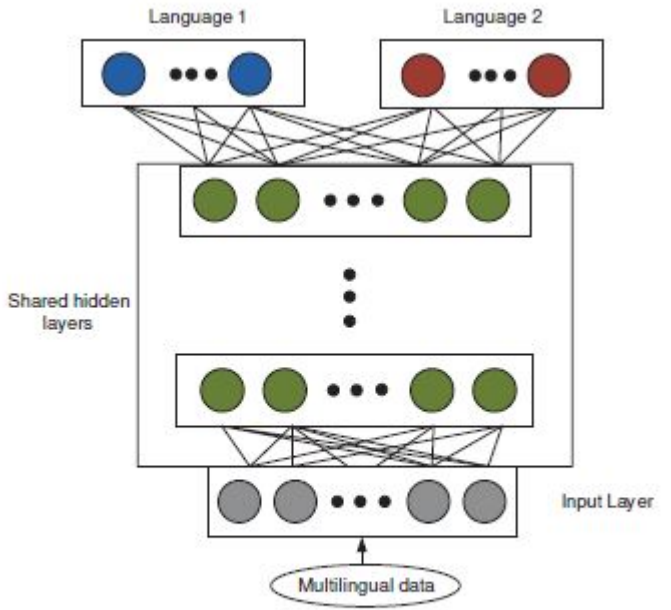
MXNet (DMLC)

Chainer

Keras

Specialized Frameworks: Speech Recognition with DNN

KALDI



(a) Multilingual DNN with language dependent output layer.

Keras: Deep Learning library for TensorFlow and Theano

High-level neural networks library, written in **Python** and capable of running on top of either TensorFlow or Theano.

- Allows for easy and fast prototyping (through total modularity, minimalism, and extensibility).
- Supports convolutional networks and recurrent networks.
- Supports arbitrary connectivity schemes (including multi-input and multi-output training).
- Runs seamlessly on CPU and GPU.
- Out-of-the-box standard layers, optimization algorithms, loss functions.

Read the documentation at Keras.io.

Keras is compatible with: Python 2.7-3.5.

Setup

- Install python pip if it is not already installed:

```
# Ubuntu/Linux 64-bit
        $ sudo apt-get install python-pip python-dev
# Mac OS X
        $ sudo easy_install pip
        $ sudo easy_install --upgrade six
# Mac OS X Homebrew version (requires Xcode)
        $ /usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"
        $ brew install python
# Windows
```

- Install tensorflow
  pip install tensorflow

- Get Keras pyhton package from github
  git clone https://github.com/fchollet/keras.git
  export PYTHONPATH=`pwd`/keras

- Get Keras examples for this tutorial
  git clone https://github.com/jarfo/dlsl.git

Keras models

Two types of Keras models:

- Sequential model (linear stack of layers)

```
model = Sequential([
    Dense(64, activation='relu', input_dim=784),
    Dense(10, activation='softmax')
])
```

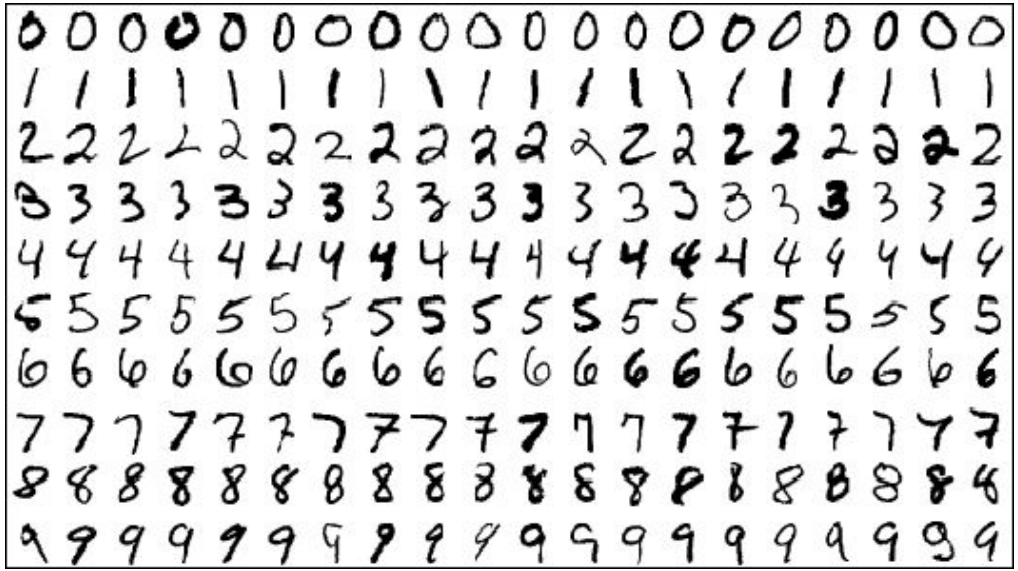
- Functional API (complex models, multi-output models, shared layers, directed acyclic graph)

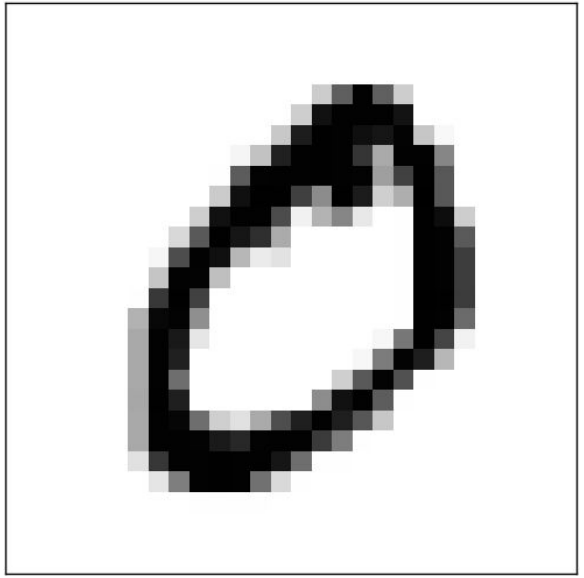
```
inputs = Input(shape=(784,))
x = Dense(64, activation='relu')(inputs)
predictions = Dense(10, activation='softmax')(x)
model = Model(input=inputs, output=predictions)


# Layers as functions
```

Example 1: MNIST

A database of handwritten digits,
Training set of 60,000 examples. Test set of 10,000 examples.
The digits have been size-normalized and centered in a fixed-size 28x28 image





Example 1: Getting started

Example1. Select Sequential model

```
from keras.models import Sequential
model = Sequential()
```

Stacking layers using `.add()`:

```
from keras.layers import Dense, Activation
model.add(Dense(output_dim=64, input_dim=784))
model.add(Activation("relu"))
model.add(Dense(output_dim=10))
model.add(Activation("softmax"))
```

Configure its learning process with `.compile()`:

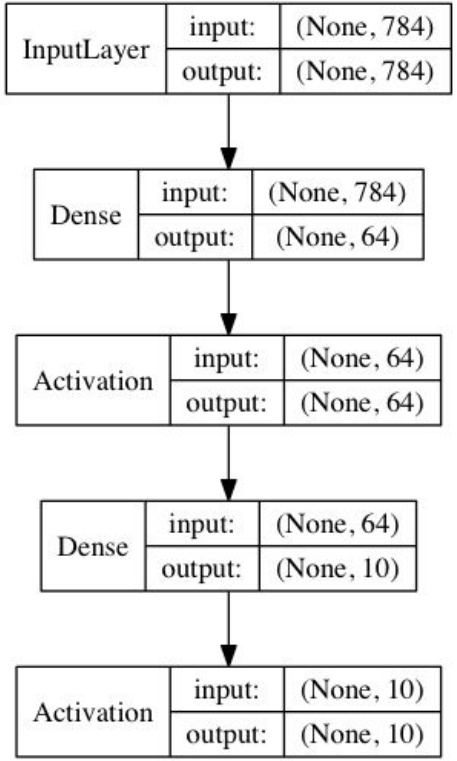
```
model.compile(loss='categorical_crossentropy', optimizer='sgd', metrics=['accuracy'])
```

Iterate on your training data in batches:

```
model.fit(X_train, Y_train, nb_epoch=5, batch_size=32)
```

Evaluate your performance in one line:

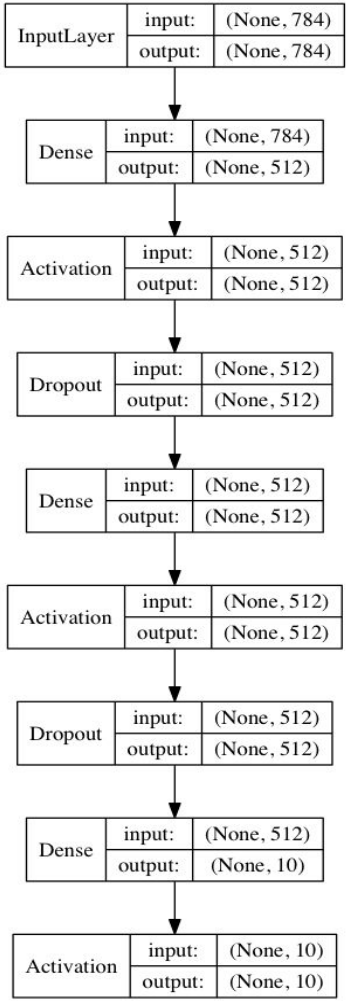
```
loss_and_metrics = model.evaluate(X_test, Y_test, batch_size=32)
```



Example 2: mnist_mlp.py

```
# Sequential model for the MNIST database of handwritten digits
model = Sequential()
model.add(Dense(512, input_shape=(784,)))
model.add(Activation('relu'))
model.add(Dropout(0.2))
model.add(Dense(512))
model.add(Activation('relu'))
model.add(Dropout(0.2))
model.add(Dense(10))
model.add(Activation('softmax'))
plot(model, show_shapes=True, to_file='mnist_mlp.png', show_layer_names=False)
model.compile(loss='categorical_crossentropy',
              optimizer=RMSprop(),
              metrics=['accuracy'])
```

mnist_mlp.py



Example 2: mnist_mlp.py (summary)

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 512)               401920
_________________________________________________________________
activation_1 (Activation)    (None, 512)               0
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 512)               262656
_________________________________________________________________
activation_2 (Activation)    (None, 512)               0
_________________________________________________________________
dropout_2 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_3 (Dense)              (None, 10)                5130
_________________________________________________________________
activation_3 (Activation)    (None, 10)                0
=================================================================
Total params: 669,706
Trainable params: 669,706
Non-trainable params: 0
_________________________________________________________________
```

Example 2: mnist_mlp.py (val_acc)

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/20
60000/60000 [==============================] - 9s - loss: 0.2418 - acc: 0.9247 - val_loss: 0.0960 - val_acc: 0.9711
Epoch 2/20
60000/60000 [==============================] - 9s - loss: 0.1011 - acc: 0.9698 - val_loss: 0.0809 - val_acc: 0.9764
Epoch 3/20
60000/60000 [==============================] - 10s - loss: 0.0753 - acc: 0.9775 - val_loss: 0.0835 - val_acc: 0.9751
Epoch 4/20
60000/60000 [==============================] - 10s - loss: 0.0608 - acc: 0.9817 - val_loss: 0.0782 - val_acc: 0.9783
Epoch 5/20
60000/60000 [==============================] - 10s - loss: 0.0502 - acc: 0.9851 - val_loss: 0.0703 - val_acc: 0.9813

../..

Epoch 17/20
60000/60000 [==============================] - 12s - loss: 0.0212 - acc: 0.9945 - val_loss: 0.1111 - val_acc: 0.9827
Epoch 18/20
60000/60000 [==============================] - 12s - loss: 0.0196 - acc: 0.9946 - val_loss: 0.1182 - val_acc: 0.9822
Epoch 19/20
60000/60000 [==============================] - 12s - loss: 0.0196 - acc: 0.9950 - val_loss: 0.1093 - val_acc: 0.9843
Epoch 20/20
60000/60000 [==============================] - 12s - loss: 0.0179 - acc: 0.9950 - val_loss: 0.1203 - val_acc: 0.9824
Test score: 0.120279448856
Test accuracy: 0.9824
```

K 20 newsgroup data set (bydate version)

Collection of approximately 20,000 newsgroup documents, partitioned (nearly) evenly across 20 different newsgroups.

alt.atheism
comp.graphics
comp.os.ms-windows.misc
comp.sys.ibm.pc.hardware
comp.sys.mac.hardware
comp.windows.x
misc.forsale
rec.autos
rec.motorcycles
rec.sport.baseball
rec.sport.hockey
sci.crypt
sci.electronics
sci.med
sci.space
soc.religion.christian
talk.politics.guns
talk.politics.mideast
talk.politics.misc
talk.religion.misc

From: joth@ersys.edmonton.ab.ca (Joe Tham)
Subject: Where can I find SIPP?
Organization: Edmonton Remote Systems #2, Edmonton, AB, Canada
Lines: 11

I recently got a file describing a library of rendering routines called SIPP (SImple Polygon Processor). Could anyone tell me where I can FTP the source code and which is the newest version around?

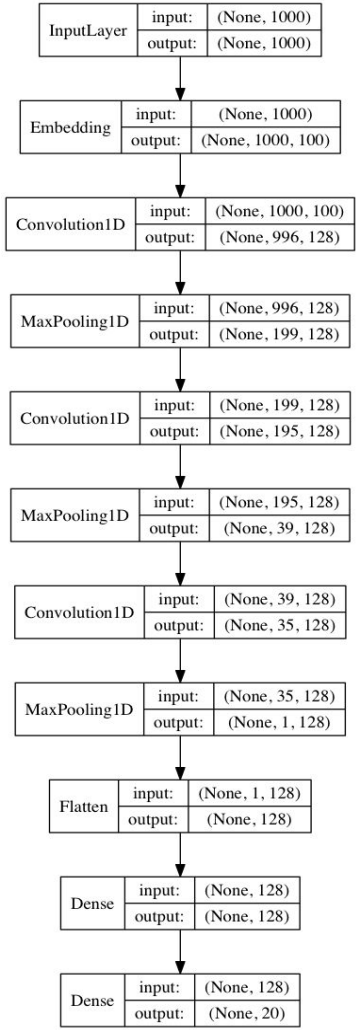
Also, I've never used Renderman so I was wondering if Renderman is like SIPP? ie. a library of rendering routines which one uses to make a program that creates the image...

Thanks, Joe Tham

--
Joe Tham joth@ersys.edmonton.ab.ca

20 newsgroup (model)

```
#  Functional model
# load pre-trained word embeddings into an Embedding layer
# note that we set trainable = False so as to keep the embeddings fixed
embedding_layer = Embedding(embedding_matrix.shape[0], embedding_matrix.shape[1],
                            input_length=MAX_SEQUENCE_LENGTH,
                            weights=[embedding_matrix], trainable=False)
# train a 1D convnet with global maxpooling
sequence_input = Input(shape=(MAX_SEQUENCE_LENGTH,), dtype='int32')
embedded_sequences = embedding_layer(sequence_input)
x = Conv1D(128, 5, activation='relu')(embedded_sequences)
x = MaxPooling1D(5)(x)
x = Conv1D(128, 5, activation='relu')(x)
x = MaxPooling1D(5)(x)
x = Conv1D(128, 5, activation='relu')(x)
x = MaxPooling1D(35)(x)
x = Flatten()(x)
x = Dense(128, activation='relu')(x)
preds = Dense(len(labels_index), activation='softmax')(x)
model = Model(sequence_input, preds)
model.compile(loss='sparse_categorical_crossentropy', optimizer='rmsprop', metrics=['acc'])
```



20 newsgroup (fit)

```
$ python pretrained_word_embeddings.py
Using TensorFlow backend.
Reading word vectors.
Found 400000 word vectors.
Processing text dataset
Found 11314 training texts.
Found 7532 test texts.
Training model.
Train on 11314 samples, validate on 7532 samples
Epoch 1/20
11314/11314 [==============================] - 202s - loss: 2.6423 - acc: 0.1355 - val_loss: 2.4272 - val_acc: 0.1911
Epoch 2/20
11314/11314 [==============================] - 199s - loss: 1.9865 - acc: 0.2836 - val_loss: 1.8056 - val_acc: 0.3585
Epoch 3/20
11314/11314 [==============================] - 197s - loss: 1.6437 - acc: 0.4026 - val_loss: 1.6412 - val_acc: 0.4304
../..
Epoch 7/20
11314/11314 [==============================] - 194s - loss: 0.7226 - acc: 0.7509 - val_loss: 1.0494 - val_acc: 0.6681
Epoch 8/20
11314/11314 [==============================] - 200s - loss: 0.6111 - acc: 0.7914 - val_loss: 0.9850 - val_acc: 0.6864
```

Keras Layers

Core Layers: Dense, Activation, Dropout, Flatten, Reshape, RepeatVector, Merge, Lambda, Highway, …

Convolutional Layers: Convolution1D, Convolution2D, …

Pooling Layers: MaxPooling1D, MaxPooling2D, …

Recurrent Layers: SimpleRNN, GRU and LSTM

Embedding Layers: Embedding

Normalization Layers: BatchNormalization

Layers wrappers: TimeDistributed, Bidirectional

Advance Activation Layers: LeakyReLU, PReLU

Keras objectives

- **mean_squared_error / mse**
- **mean_absolute_error / mae**
- **mean_absolute_percentage_error / mape**
- **mean_squared_logarithmic_error / msle**
- **Squared_hinge, hinge**
- **binary_crossentropy**: Also known as logloss.
- **categorical_crossentropy**: Also known as multiclass logloss. **Note**: using this objective requires that your labels are binary arrays of shape `(nb_samples, nb_classes)`.
- **sparse_categorical_crossentropy**: As above but accepts sparse labels. **Note**: this objective still requires that your labels have the same number of dimensions as your outputs; you may need to add a length-1 dimension to the shape of your labels.
- **kullback_leibler_divergence / kld**: Gives a measure of difference between two distributions.
- **poisson**: Mean of `(predictions - targets * log(predictions))`
- **cosine_proximity**: The opposite (negative) of the mean cosine proximity between predictions and targets.

Keras activations

- **softmax** (last layer and categorical crossentropy objective)
- **softplus**
- **softsign**
- **relu** (simple default choice)
- **tanh**
- **sigmoid** (single output last layer and binary crossentropy objective)
- **hard_sigmoid**
- **Linear**
- **Advanced activation layers** (learnable or configurable variants)

Other Keras modules

- **Optimizers:** SGD, RMSprop, Adagrad, Adadelta, Adam, Adamax, Nadam, ...
- **Callbacks:** EarlyStopping, LearningRateScheduler, TensorBoard ...
- **Initializations:** uniform, orthogonal, normal and variants
- **Regularizers:** l1, l2, l1l2, activity_l1, activity_l2, activity_l1l2
- **Constraints:** maxnorm, nonneg, unitnorm
- **Visualization:** plot
- **Utils:** data, I/O, layer, numpy
- **Preprocessing:** sequence, text, image
- **Datasets:** IMDB, Reuters, CIFAR10, CIFAR100. MNIST

20 newsgroup (fit)

```
$ python pretrained_word_embeddings.py
Using TensorFlow backend.
Reading word vectors.
Found 400000 word vectors.
Processing text dataset
Found 11314 training texts.
Found 7532 test texts.
Training model.
Train on 11314 samples, validate on 7532 samples
Epoch 1/20
11314/11314 [==============================] - 202s - loss: 2.6423 - acc: 0.1355 - val_loss: 2.4272 - val_acc: 0.1911
Epoch 2/20
11314/11314 [==============================] - 199s - loss: 1.9865 - acc: 0.2836 - val_loss: 1.8056 - val_acc: 0.3585
Epoch 3/20
11314/11314 [==============================] - 197s - loss: 1.6437 - acc: 0.4026 - val_loss: 1.6412 - val_acc: 0.4304
../..
Epoch 7/20
11314/11314 [==============================] - 194s - loss: 0.7226 - acc: 0.7509 - val_loss: 1.0494 - val_acc: 0.6681
Epoch 8/20
11314/11314 [==============================] - 200s - loss: 0.6111 - acc: 0.7914 - val_loss: 0.9850 - val_acc: 0.6864
```

Bachelor Projects

INTRODUCCIÓ A L'APRENENTATGE PROFUND PER A LA PARLA I EL LLENGUATGE

- Small project
- Groups of 2 or 3 students
- Try different parameters, optimizations, layers, etc to simple tasks (pho, PoS, 20news)
- E.g, plot accuracy versus epoch (train/valid)
- 2 page report (including figures and tables) + references + code in github

Master Projects

APRENENTATGE PROFUND PER A LA PARLA I EL LLENGUATGE

- Groups of 3 students
- Tasks: pho, PoS, 20news, imbd, reuters, End of sentence, Language modeling, other
- Public presentation next Tuesday (15 min)
- 4 page report in article format + code in github + presentation slides

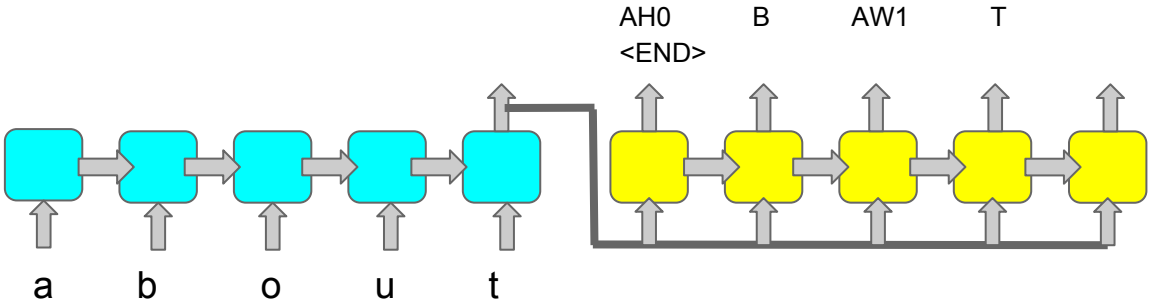
Phonetic transcription (CMUdict)

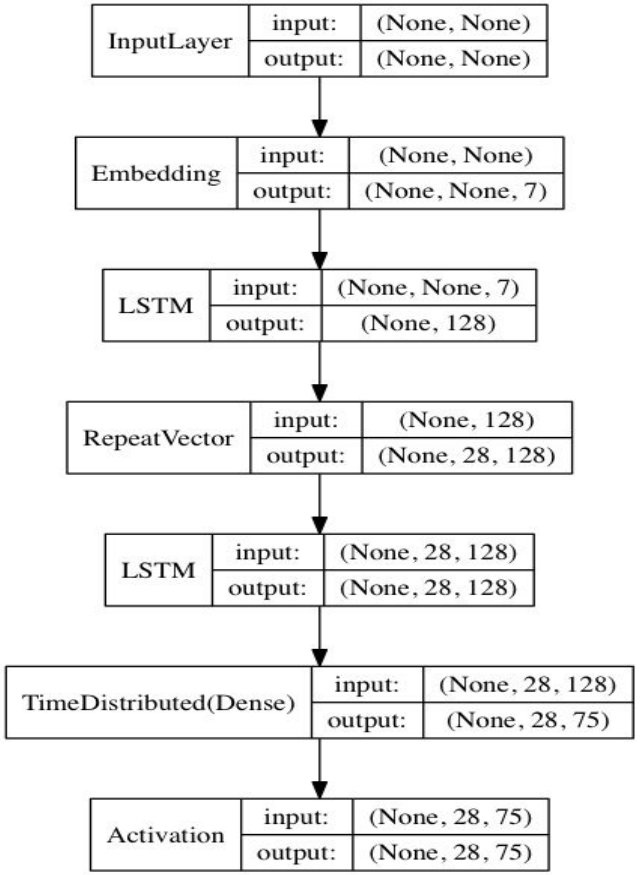
- The Carnegie Mellon University Pronouncing Dictionary is an open-source machine-readable pronunciation dictionary for North American English that contains over 134,000 words and their pronunciations.
- Useful for speech recognition and synthesis
- ARPAbet phoneme set (39 phonemes)
- Vowels carry a lexical stress marker (74 different phonetic labels):
  0 -> No stress, 1 -> Primary stress, 2 ->Secondary stress
- Examples:

```
ORIGINAL VERSION
      wcmudict.train.dict: a d r i a n           EY1 D R IY0 AH0 N
      wcmudict.train.dict: a d r i a n c e       AA0 D R IY1 AH0 N S
      wcmudict.train.dict: t h i n k e r         TH IH1 NG K ER0

ALIGNED VERSION (by UPC):
      wcmudict.train.aligned: a d r i a n        EY1 D R IY0 AH0 N
      wcmudict.train.aligned: a d r i a n c e    AA0 D R IY1 AH0 N S NULL
      wcmudict.train.aligned: t h i n k e r      NULL TH IH1 NG K NULL ER0
```

Phonetic transcription (CMUdict)



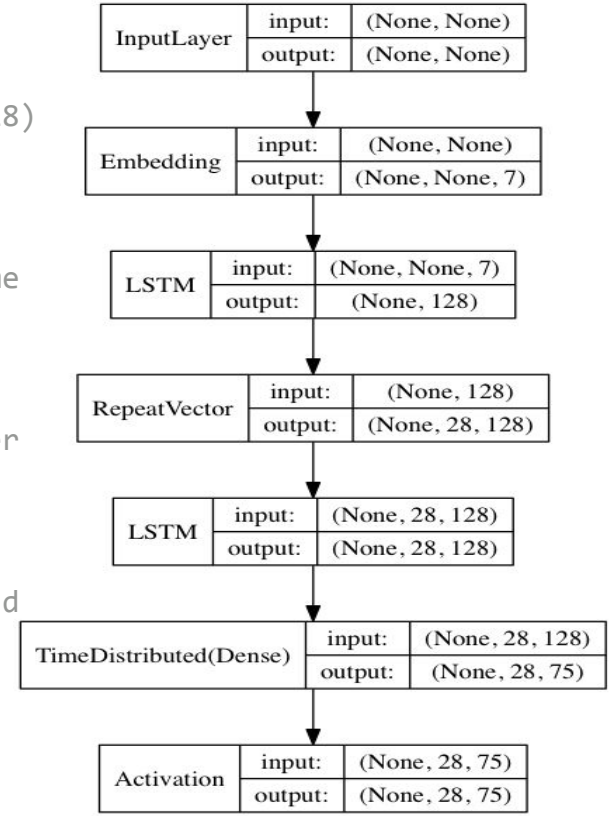


- It does not need the aligned training dictionary
- We need to code and from a single vector

More Keras sequence to sequence models in arizrahman4u/seq2seq

Phonetic transcription (CMUdict)

```
model = Sequential()
# Encode the input sequence using an RNN, output of HIDDEN_SIZE (128)
model.add(Embedding(ctable.size, VSIZE, input_dtype='int32'))
model.add(RNN(HIDDEN_SIZE))
# For the decoder's input, we repeat the encoded input for each time
step
model.add(RepeatVector(trans_maxlen))
# The decoder RNN could be multiple layers stacked or a single layer
for _ in range(LAYERS):
   model.add(RNN(HIDDEN_SIZE, return_sequences=True))
# For each of step of the output sequence, decide which phone should
be chosen
model.add(TimeDistributed(Dense(ptable.size)))
model.add(Activation('softmax'))
```



Language modeling with RNN

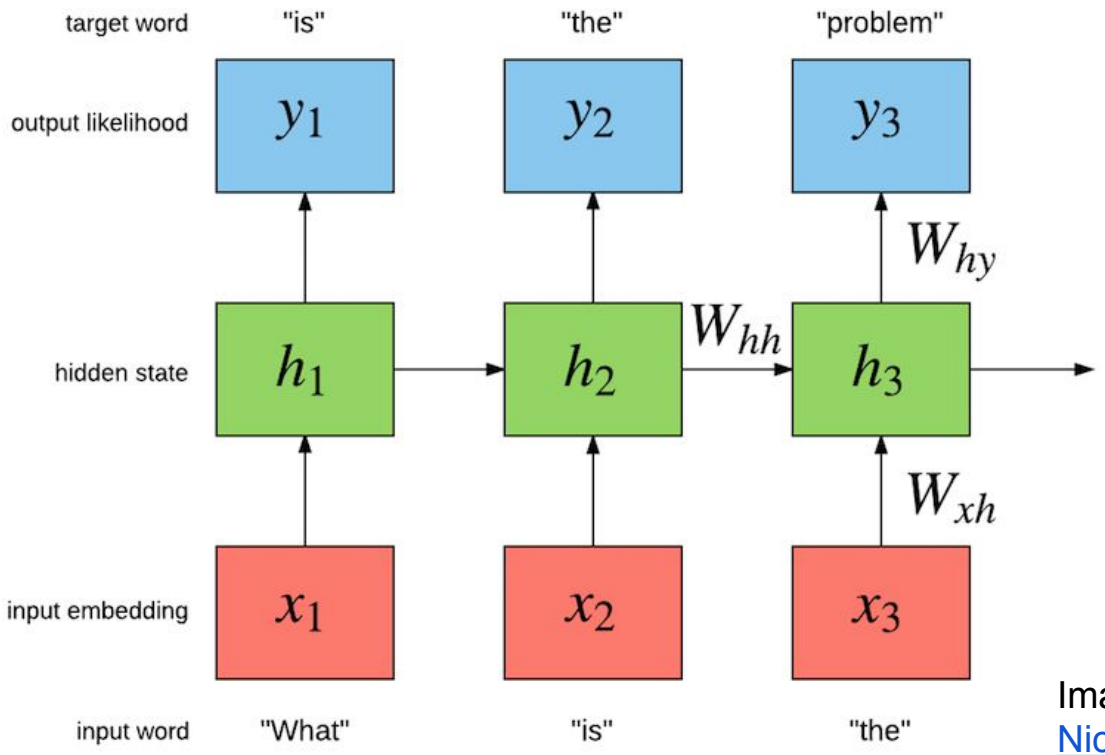


Image by
Nicholas Léonard

Part of Speech tagging

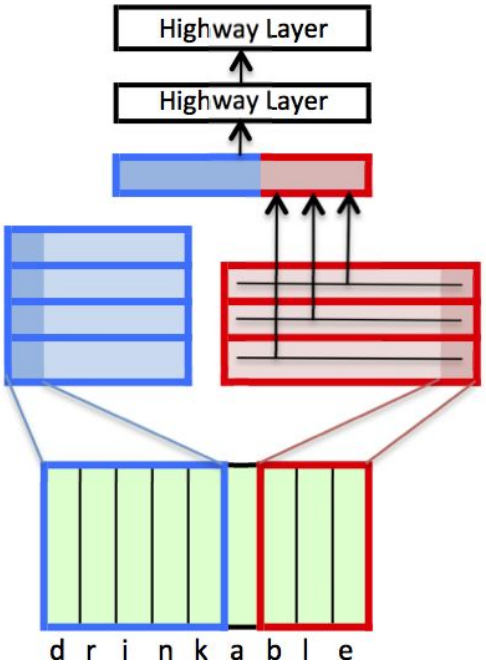
Training Windows (winSize of 5 words)	Target (Plural, Singular, Neutral)
['VMIP3[exhortar]', 'a', 'DI00[todo]', 'DA00[el]', 'NC000[estado]']	[1.0, 0.0, 0.0]
['a', 'DI00[todo]', 'DA00[el]', 'NC000[estado]', 'a']	[1.0, 0.0, 0.0]
['DI00[todo]', 'DA00[el]', 'NC000[estado]', 'a', 'que']	[1.0, 0.0, 0.0]
['a', 'que', 'VMSP3[permitir]', ',', 'de']	[1.0, 0.0, 0.0]
[',', 'de', 'NC000[conformidad]', 'con', 'DA00[el]']	[0.0, 1.0, 0.0]

Keras baseline

```
model.add(Embedding(vocabSize + 3, VSIZE, input_length=winSize, input_dtype='int32'))
model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(Dense(trainTargets.shape[1], activation='softmax'))
```

Char-based Language modeling

The vector representation of each word is computed by a character-based convolutional network



Highway Layers

Max output of each filter

Multiple convolution filters of different lenghts

Sequence of character embeddings

	PPL	Size
LSTM-Word-Small	97.6	5 m
LSTM-Char-Small	92.3	5 m
LSTM-Word-Large	85.4	20 m
LSTM-Char-Large	78.9	19 m
KN-5 (Mikolov et al. 2012)	141.2	2 m
RNN† (Mikolov et al. 2012)	124.7	6 m
RNN-LDA† (Mikolov et al. 2012)	113.7	7 m
genCNN† (Wang et al. 2015)	116.4	8 m
FOFE-FNNLM† (Zhang et al. 2015)	108.0	6 m
Deep RNN (Pascanu et al. 2013)	107.5	6 m
Sum-Prod Net† (Cheng et al. 2014)	100.0	5 m
LSTM-1† (Zaremba et al. 2014)	82.7	20 m
LSTM-2† (Zaremba et al. 2014)	78.4	52 m

Char-based Language modeling

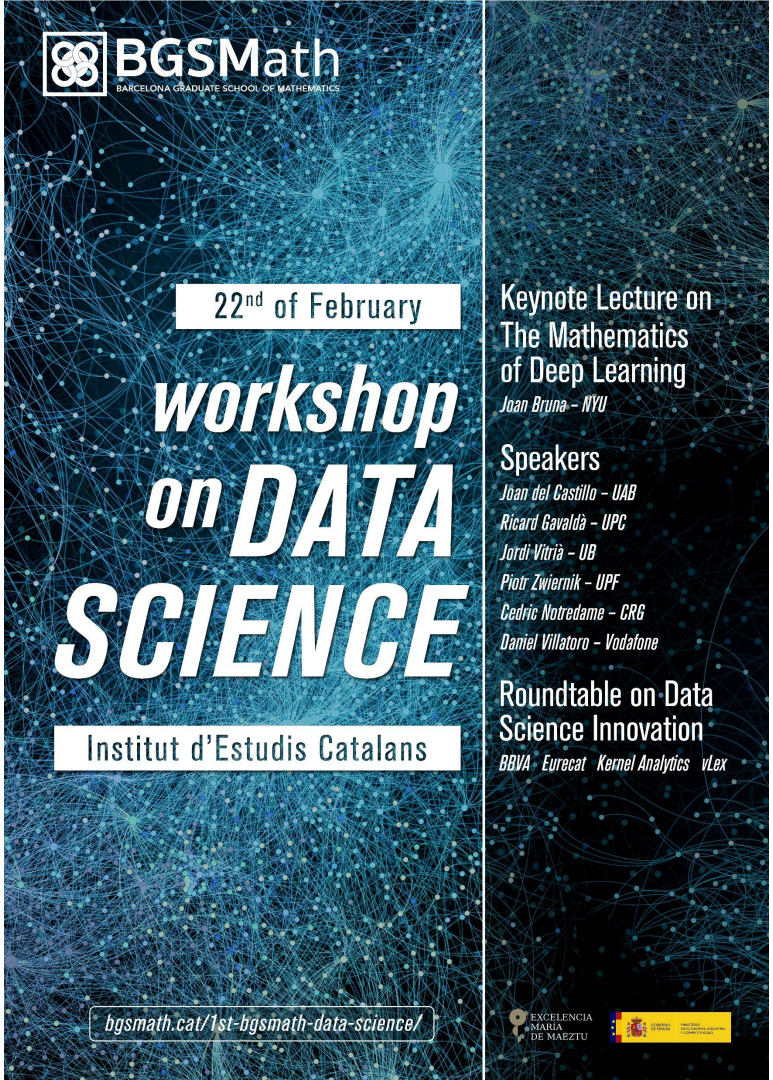
```
def CNN(seq_length, length, input_size, feature_maps, kernels, x):
    concat_input = []
    for feature_map, kernel in zip(feature_maps, kernels):
        reduced_l = length - kernel + 1
        conv = Convolution2D(feature_map, 1, kernel, activation='tanh', dim_ordering='tf')(x)
        maxp = MaxPooling2D((1, reduced_l), dim_ordering='tf')(conv)
        concat_input.append(maxp)
    x = Merge(mode='concat')(concat_input)
    x = Reshape((seq_length, sum(feature_maps)))(x)

chars = Input(batch_shape=(opt.batch_size, opt.seq_length, opt.max_word_l), dtype='int32', name='chars')
chars_embedding = TimeDistributed(Embedding(opt.char_vocab_size, opt.char_vec_size, name='chars_embedding'))(chars)
cnn = CNN(opt.seq_length, opt.max_word_l, opt.char_vec_size, opt.feature_maps, opt.kernels, chars_embedding)
for l in range(opt.highway_layers):
    x = TimeDistributed(Highway(activation='relu'))(x)
for l in range(opt.num_layers):
    x = LSTM(opt.rnn_size, activation='tanh', inner_activation='sigmoid', return_sequences=True, stateful=True)(x)
if opt.dropout > 0:
    x = Dropout(opt.dropout)(x)
output = TimeDistributed(Dense(opt.word_vocab_size, activation='softmax'))(x)
```

Char-based Language modeling

Layer (type)	Output Shape	Param #	Connected to
chars (InputLayer)	(20, 35, 21)	0	
timedistributed_1 (TimeDistribut	(20, 35, 21, 15)	765	chars[0][0]
convolution2d_1 (Convolution2D)	(20, 35, 21, 50)	800	timedistributed_1[0][0]
convolution2d_2 (Convolution2D)	(20, 35, 20, 100)	3100	timedistributed_1[0][0]
convolution2d_3 (Convolution2D)	(20, 35, 19, 150)	6900	timedistributed_1[0][0]
convolution2d_4 (Convolution2D)	(20, 35, 18, 200)	12200	timedistributed_1[0][0]
convolution2d_5 (Convolution2D)	(20, 35, 17, 200)	15200	timedistributed_1[0][0]
convolution2d_6 (Convolution2D)	(20, 35, 16, 200)	18200	timedistributed_1[0][0]
convolution2d_7 (Convolution2D)	(20, 35, 15, 200)	21200	timedistributed_1[0][0]
maxpooling2d_1 (MaxPooling2D)	(20, 35, 1, 50)	0	convolution2d_1[0][0]
maxpooling2d_2 (MaxPooling2D)	(20, 35, 1, 100)	0	convolution2d_2[0][0]
maxpooling2d_3 (MaxPooling2D)	(20, 35, 1, 150)	0	convolution2d_3[0][0]
maxpooling2d_4 (MaxPooling2D)	(20, 35, 1, 200)	0	convolution2d_4[0][0]
maxpooling2d_5 (MaxPooling2D)	(20, 35, 1, 200)	0	convolution2d_5[0][0]
maxpooling2d_6 (MaxPooling2D)	(20, 35, 1, 200)	0	convolution2d_6[0][0]
maxpooling2d_7 (MaxPooling2D)	(20, 35, 1, 200)	0	convolution2d_7[0][0]
merge_1 (Merge)	(20, 35, 1, 1100)	0	maxpooling2d_1[0][0] maxpooling2d_2[0][0] maxpooling2d_3[0][0] maxpooling2d_4[0][0] maxpooling2d_5[0][0] maxpooling2d_6[0][0] maxpooling2d_7[0][0]
reshape_1 (Reshape)	(20, 35, 1100)	0	merge_1[0][0]
timedistributed_2 (TimeDistribut	(20, 35, 1100)	2422200	reshape_1[0][0]
timedistributed_3 (TimeDistribut	(20, 35, 1100)	2422200	timedistributed_2[0][0]
lstm_1 (LSTM)	(20, 35, 650)	4552600	timedistributed_3[0][0]
dropout_1 (Dropout)	(20, 35, 650)	0	lstm_1[0][0]
lstm_2 (LSTM)	(20, 35, 650)	3382600	dropout_1[0][0]
dropout_2 (Dropout)	(20, 35, 650)	0	lstm_2[0][0]
timedistributed_4 (TimeDistribut	(20, 35, 10000)	6510000	dropout_2[0][0]

Char-based Language modeling



REGISTRATION NOW OPEN!

The 1st BGSMath Data Science Workshop aims to bring together the internal practitioners in the area across the Barcelona Graduate School of Mathematics, as well as those working in related fields and/or other local organizations, to discuss ongoing research and strengthen the links of a growing network.
World-class guest speakers will also join the programme, to review their latest results, discuss with colleagues and inspire early-career researchers.
The workshop will become a regular staple in the BGSMath Events calendar. In this edition, it will be complemented by an "users" session, in which companies and research centres from a variety of sectors (Bio/health, finance, communications) will put forward their data challenges and how they are trying to solve them to the BGSMath experts. The session aims fostering collaborations and improve talent attraction schemes.
There is no registration fee but registration is mandatory for participating in the workshop.
There will a poster session. Participants willing to present a poster should provide a title and abstract in the registration page.
Program

Location: IEC, carrer del Carme, 47. Barcelona.Program