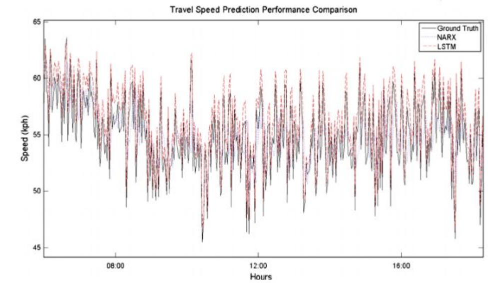
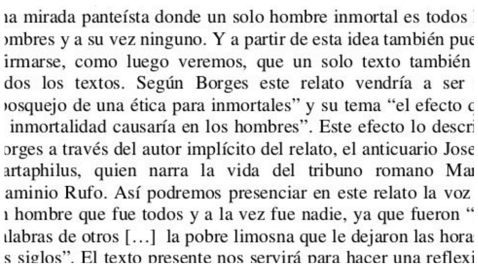
Unidad 6: Redes Recurrentes

Curso: Redes Neuronales Profundas

Datos Secuenciales

time series

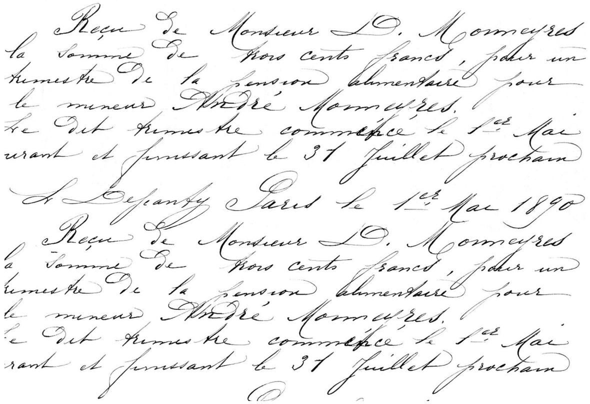




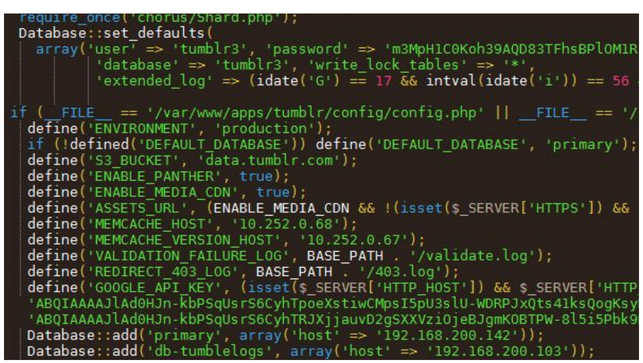
ıa mirada panteísta donde un solo hombre inmortal es todos
›mbres y a su vez ninguno. Y a partir de esta idea también pue
ırmarse, como luego veremos, que un solo texto también
dos los textos. Según Borges este relato vendría a ser
›osquejo de una ética para inmortales" y su tema "el efecto c
inmortalidad causaría en los hombres". Este efecto lo descri
›rges a través del autor implícito del relato, el anticuario Jose
artaphilus, quien narra la vida del tribuno romano Ma
aminio Rufo. Así podremos presenciar en este relato la voz
ı hombre que fue todos y a la vez fue nadie, ya que fueron "
ılabras de otros […] la pobre limosna que le dejaron las hora
s siglos". El texto presente nos servirá para hacer una reflexi

text

handwriting



speech



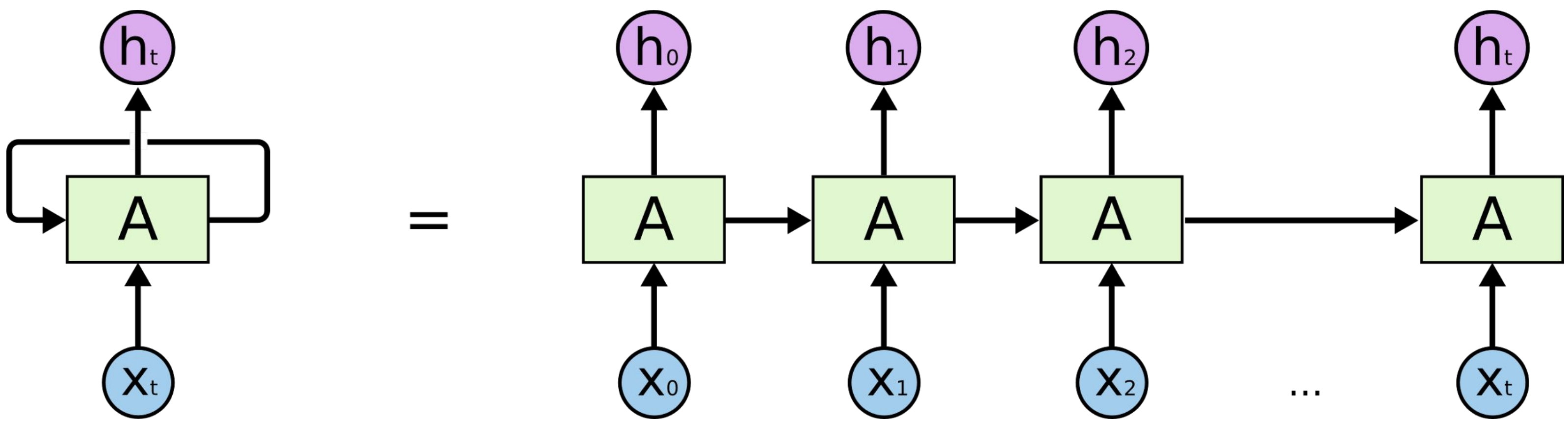
```
require_once('chorus/Shard.php');
Database::set_defaults(
  array('user' => 'tumblr3', 'password' => 'm3MpH1C0Koh39AQD83TFhsBPlOM1R
        'database' => 'tumblr3', 'write_lock_tables' => '*',
        'extended_log' => (idate('G') == 17 && intval(idate('i')) == 56

if (__FILE__ == '/var/www/apps/tumblr/config/config.php' || __FILE__ == '/
  define('ENVIRONMENT', 'production');
  if (!defined('DEFAULT_DATABASE')) define('DEFAULT_DATABASE', 'primary');
  define('S3_BUCKET', 'data.tumblr.com');
  define('ENABLE_PANTHER', true);
  define('ENABLE_MEDIA_CDN', true);
  define('ASSETS_URL', (ENABLE_MEDIA_CDN && !(isset($_SERVER['HTTPS']) &&
  define('MEMCACHE_HOST', '10.252.0.68');
  define('MEMCACHE_VERSION_HOST', '10.252.0.67');
  define('VALIDATION_FAILURE_LOG', BASE_PATH . '/validate.log');
  define('REDIRECT_403_LOG', BASE_PATH . '/403.log');
  define('GOOGLE_API_KEY', (isset($_SERVER['HTTP_HOST']) && $_SERVER['HTTP
  'ABQIAAAAJlAd0HJn-kbPSqUsrS6CyhTpoeXstiwCMpsI5pU3slU-WDRPJxQts41ksQogKsy
  'ABQIAAAAJlAd0HJn-kbPSqUsrS6CyhTRJXjjauvD2gSXXVziOjeBJgmKOBTPW-8l5i5Pbk9
  Database::add('primary', array('host' => '192.168.200.142'));
  Database::add('db-tumblelogs', array('host' => '192.168.200.103'));
```

code

stock market

Recurrent Neural Networks

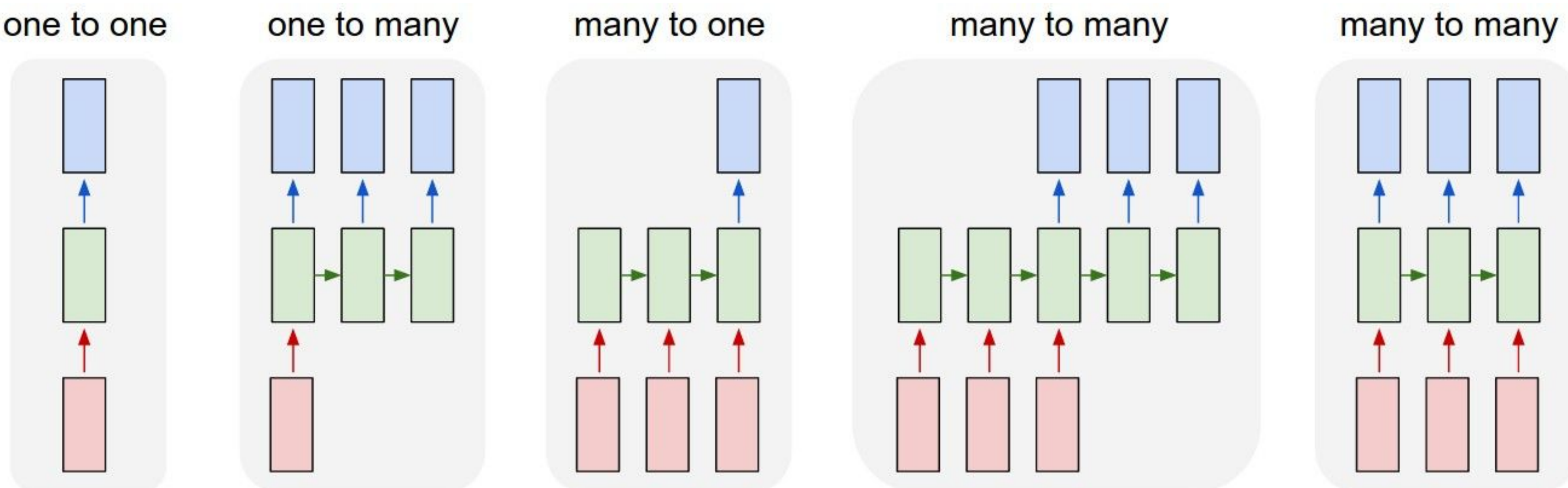
from [colah's blog](http://colah.github.io/posts/2015-08-Understanding-LSTMs/) http://colah.github.io/posts/2015-08-Understanding-LSTMs/



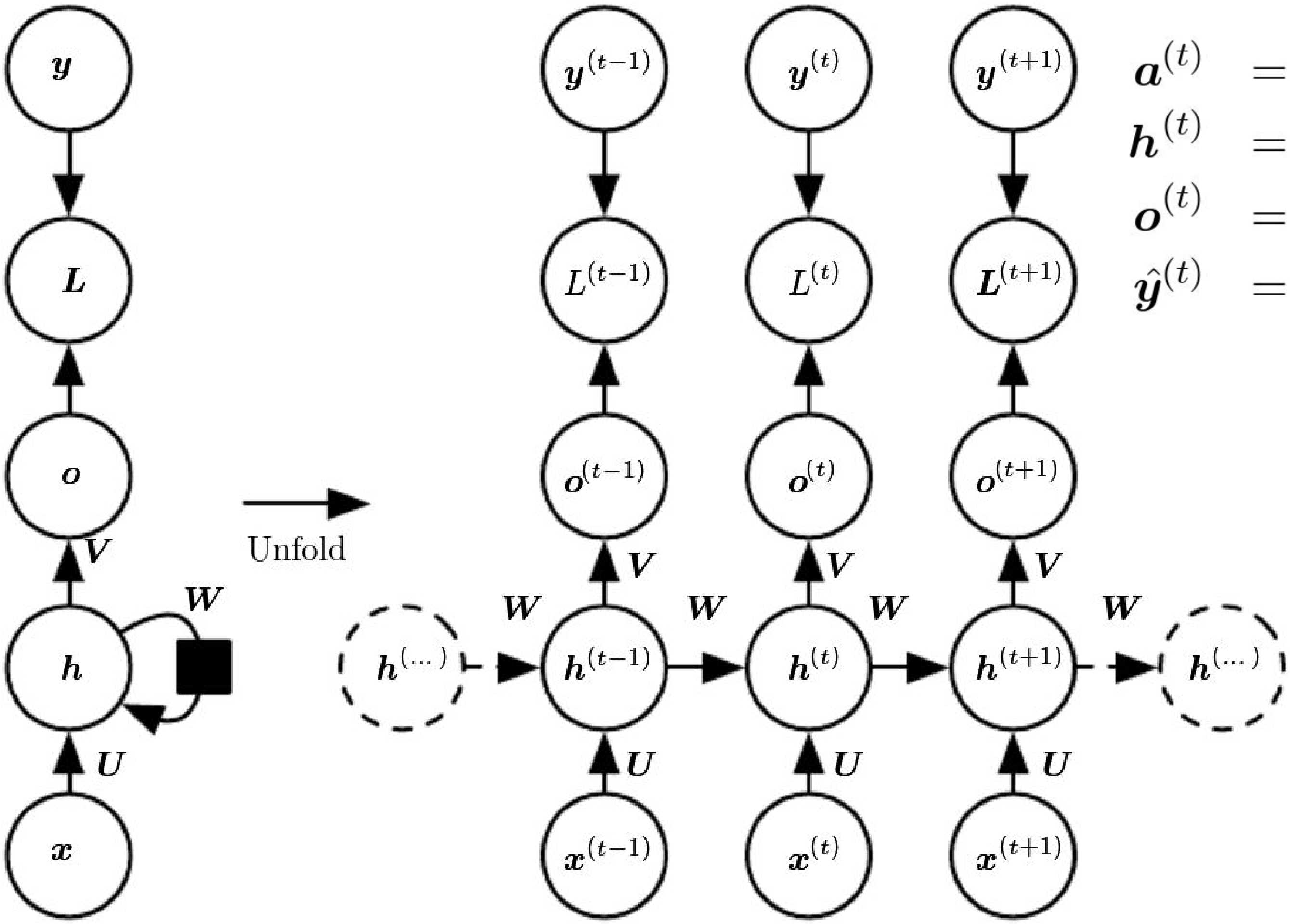
Neural Network
with a loop

Unfolded Computational Graph

Recurrent Neural Networks



Recurrent Neural Networks

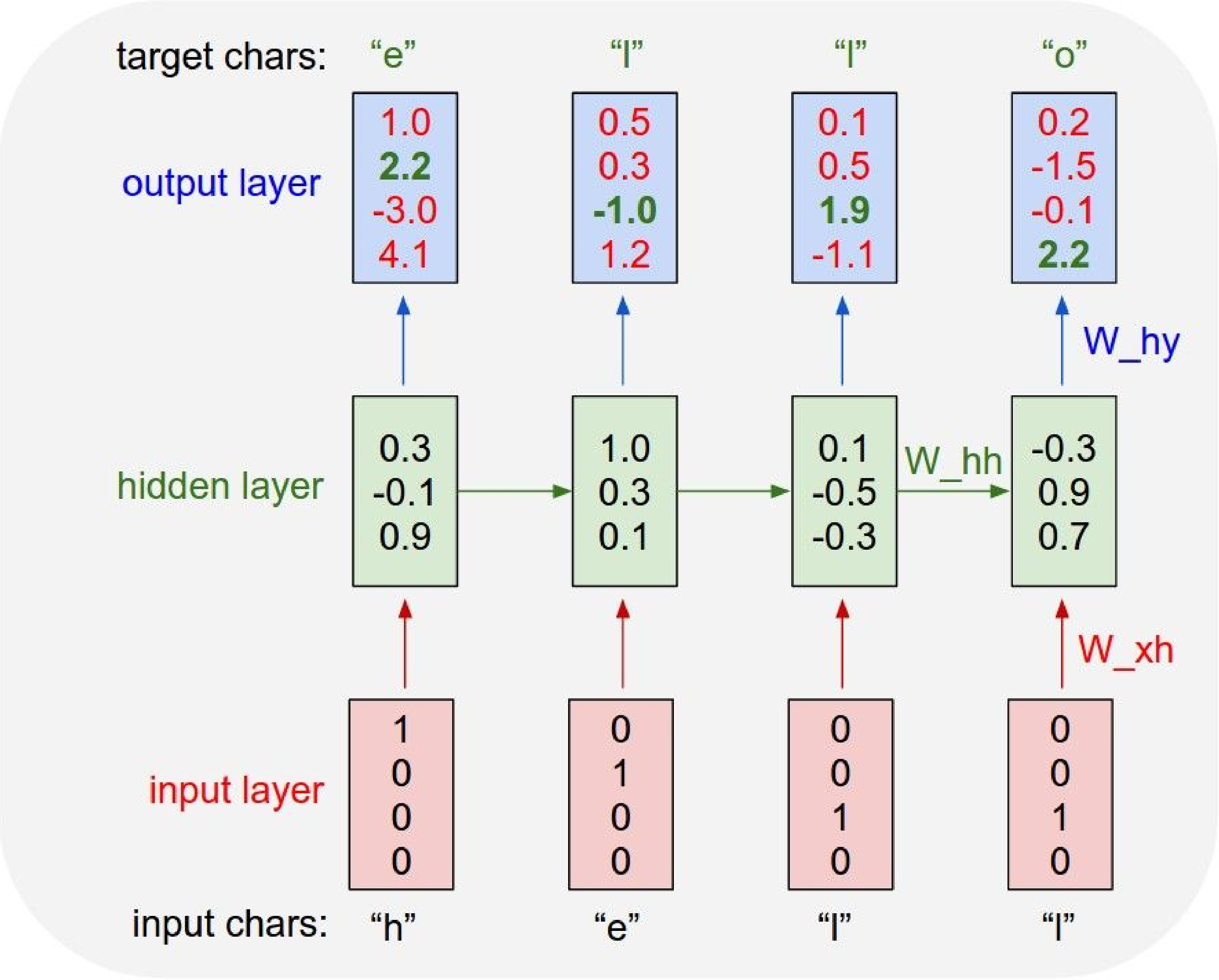


$$
\begin{aligned}
\boldsymbol{a}^{(t)} &= \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)} \\
\boldsymbol{h}^{(t)} &= \tanh(\boldsymbol{a}^{(t)}) \\
\boldsymbol{o}^{(t)} &= \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)} \\
\hat{\boldsymbol{y}}^{(t)} &= \operatorname{softmax}(\boldsymbol{o}^{(t)})
\end{aligned}
$$

Deep Learning (Book in preparation) Ian Goodfellow, Yoshua Bengio and Aaron Courville, 2016

Recurrent Neural Networks

from Andrej Karpathy blog http://karpathy.github.io/2015/05/21/rnn-effectiveness/



RNN: Problems

Ejemplo artificial: Reconocer secuencias de la forma

$$A^n B^n$$

```
AAAAAABBBBBB
AABB
AAAAAAAAABBBBBBBB
```

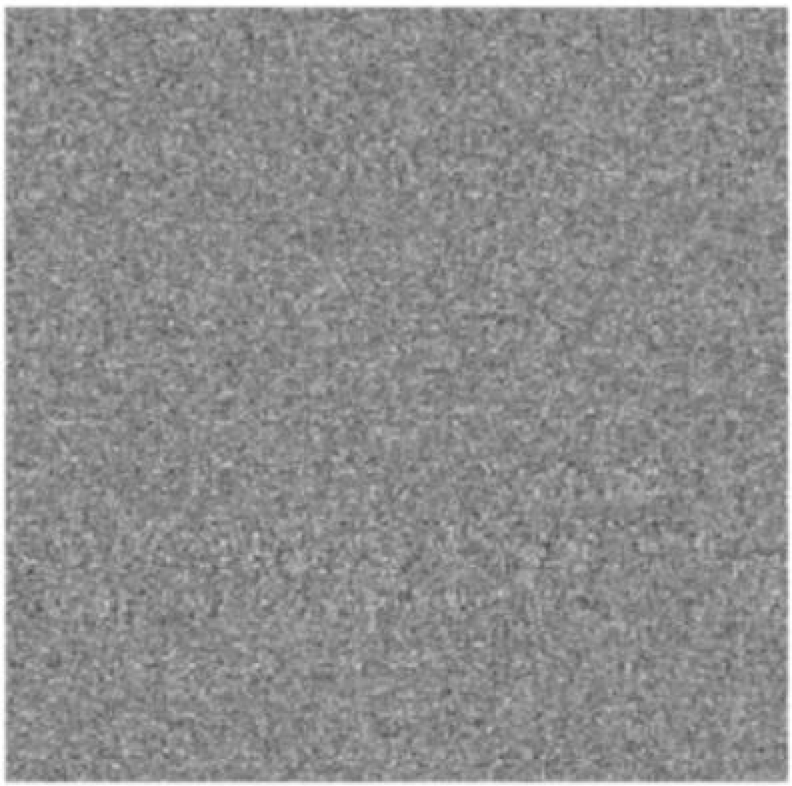
Train: n hasta 11. Test: n hasta 18.
Resultado: Calamitoso, 20% accuracy

¿POR QUÉ?

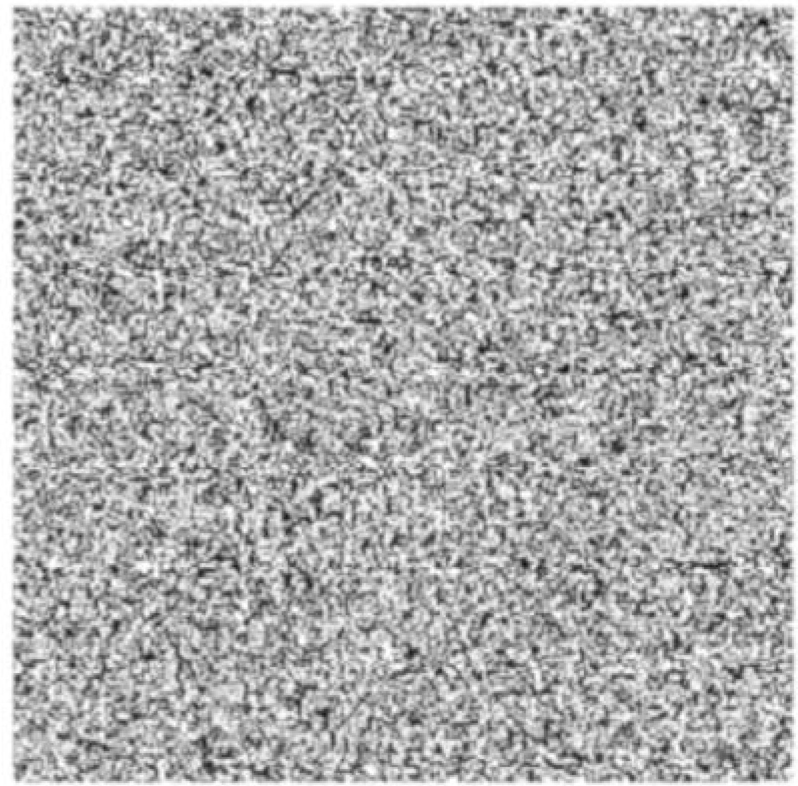
Veremos un modelo que logra 100% accuracy con
$n = 1000$

RNN: Vanishing Gradient Problem

127

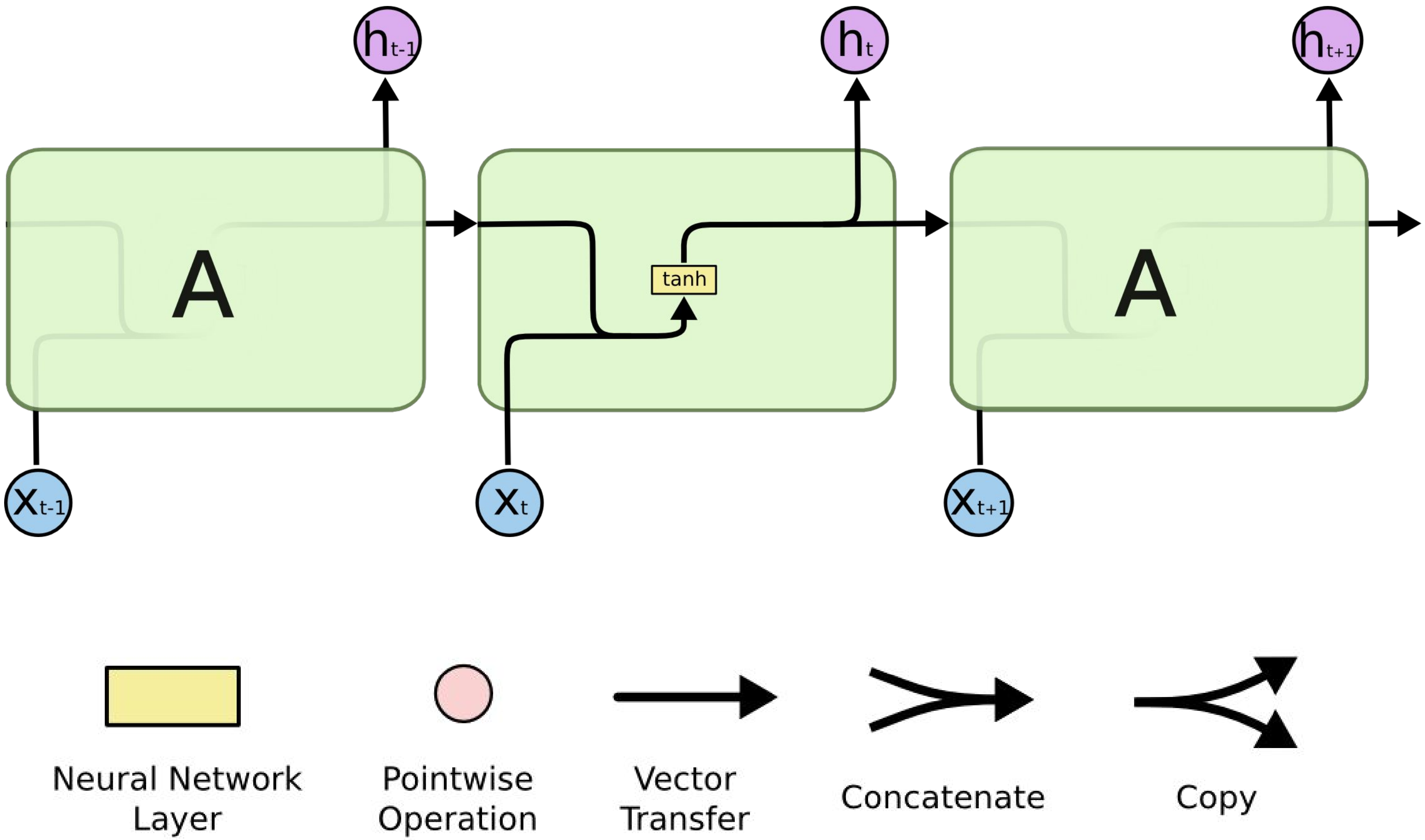


127



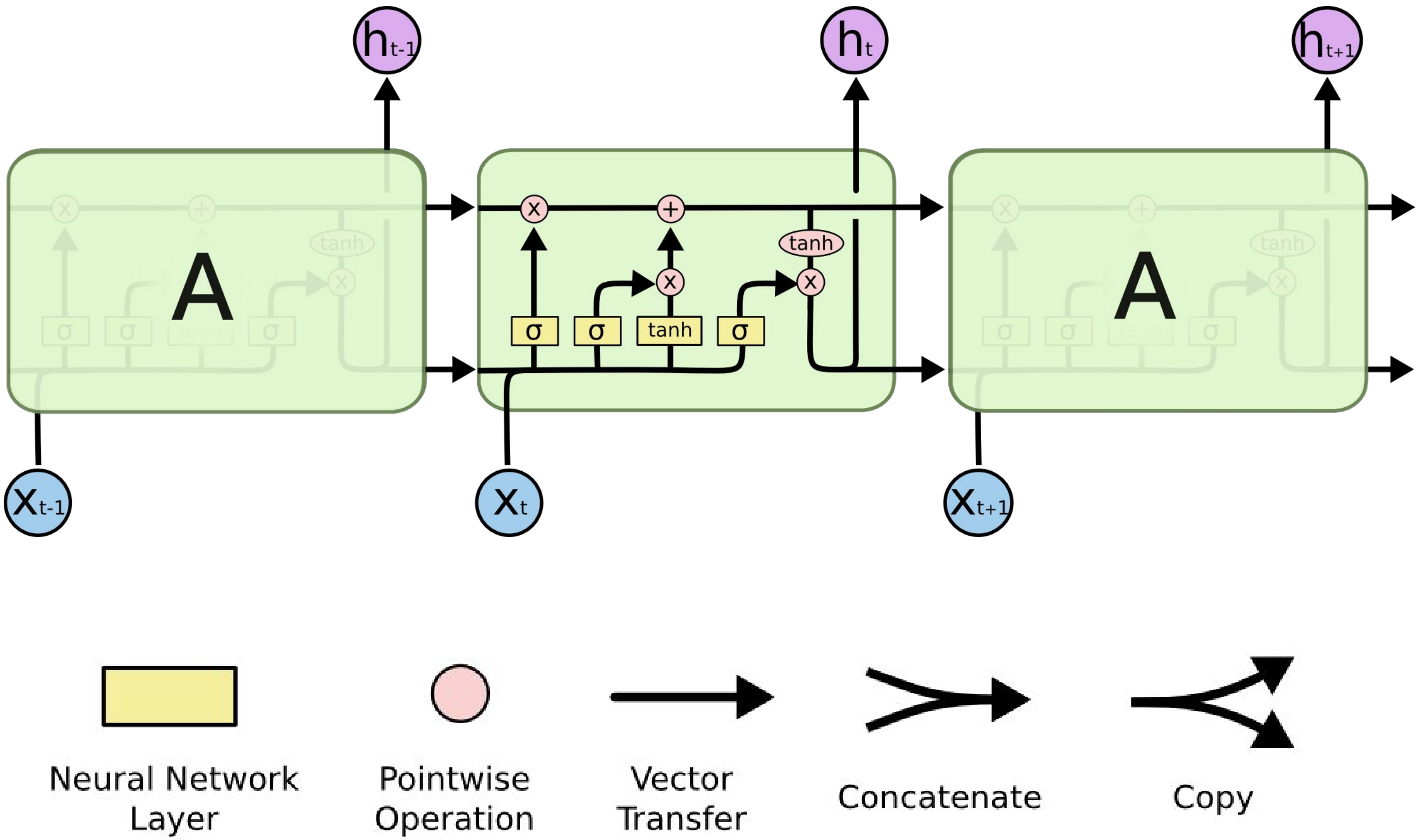
Recurrent Neural Networks

from colah's blog http://colah.github.io/posts/2015-08-Understanding-LSTMs/



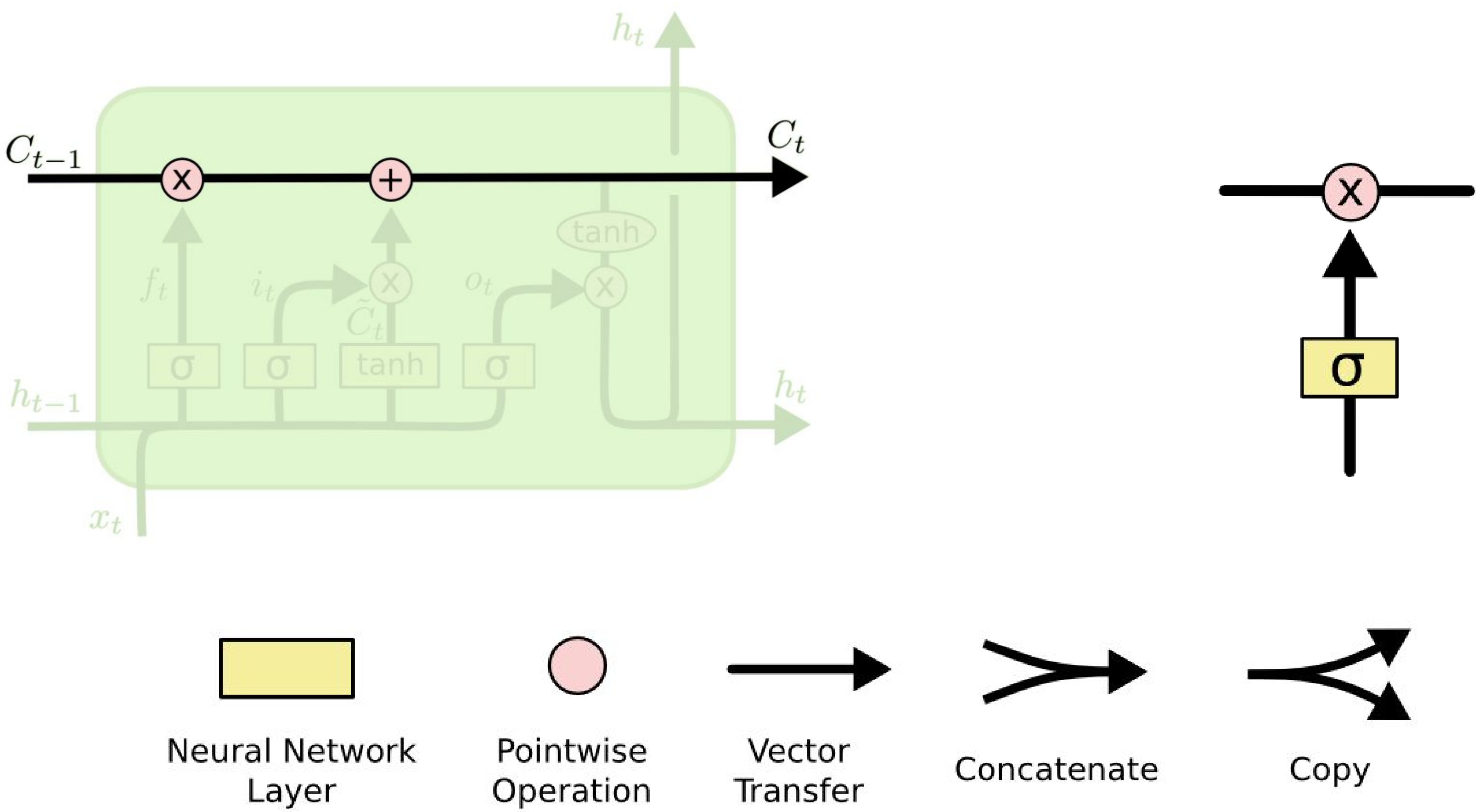
LSTM: Long Short-Term Memory

from colah's blog http://colah.github.io/posts/2015-08-Understanding-LSTMs/



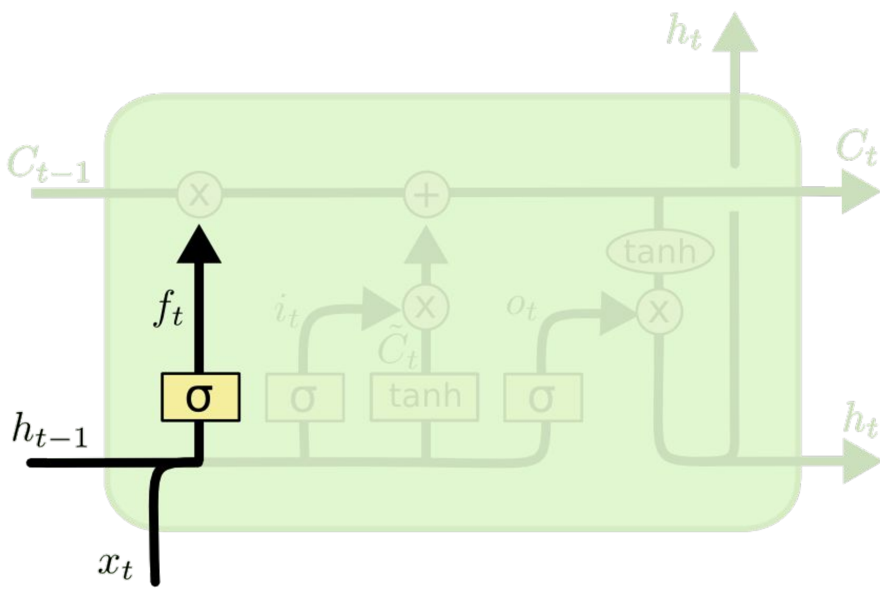
LSTM: Long Short-Term Memory

from [colah's blog](http://colah.github.io/posts/2015-08-Understanding-LSTMs/) http://colah.github.io/posts/2015-08-Understanding-LSTMs/

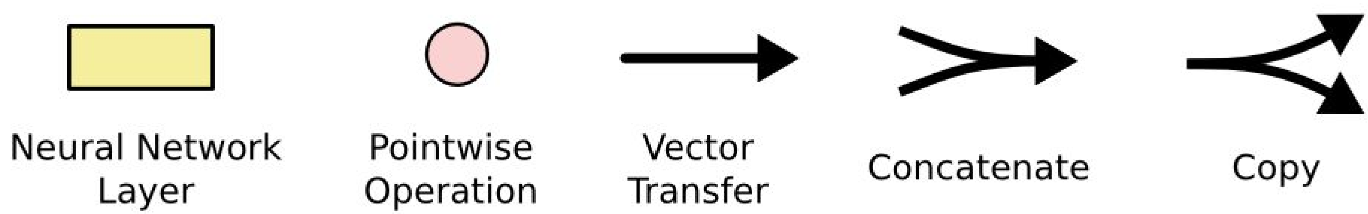


LSTM: Long Short-Term Memory

from [colah's blog](http://colah.github.io/posts/2015-08-Understanding-LSTMs/) http://colah.github.io/posts/2015-08-Understanding-LSTMs/

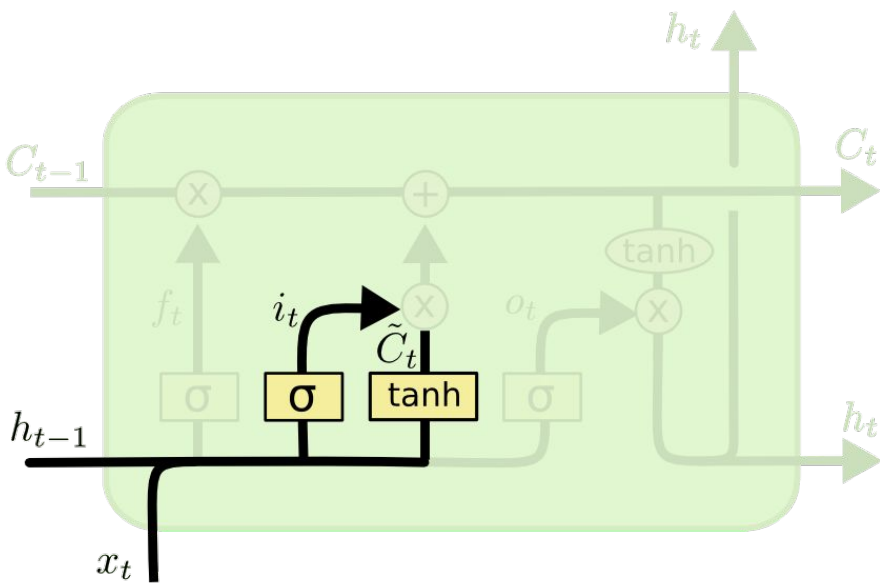


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$



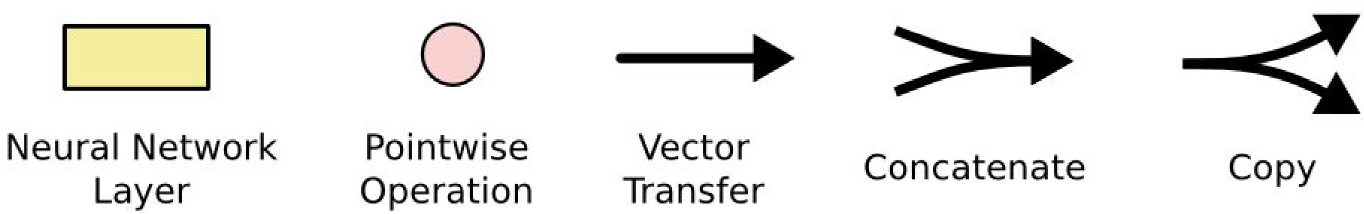
LSTM: Long Short-Term Memory

from colah's blog http://colah.github.io/posts/2015-08-Understanding-LSTMs/



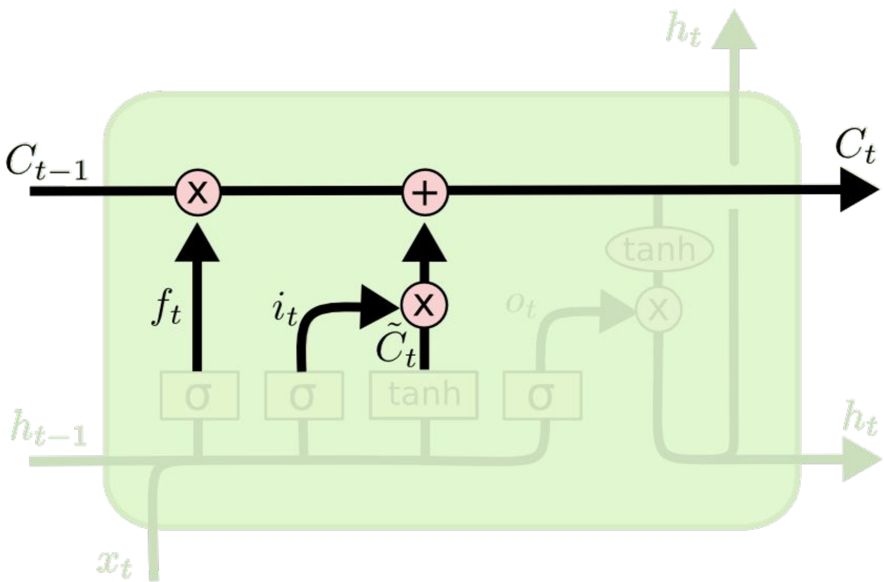
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

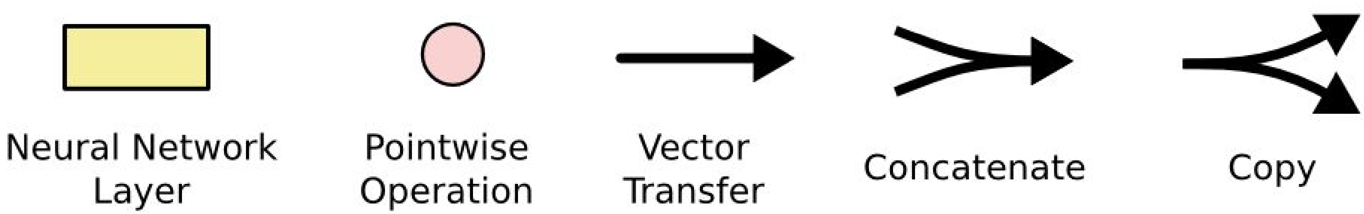


LSTM: Long Short-Term Memory

from [colah's blog](http://colah.github.io/posts/2015-08-Understanding-LSTMs/) http://colah.github.io/posts/2015-08-Understanding-LSTMs/

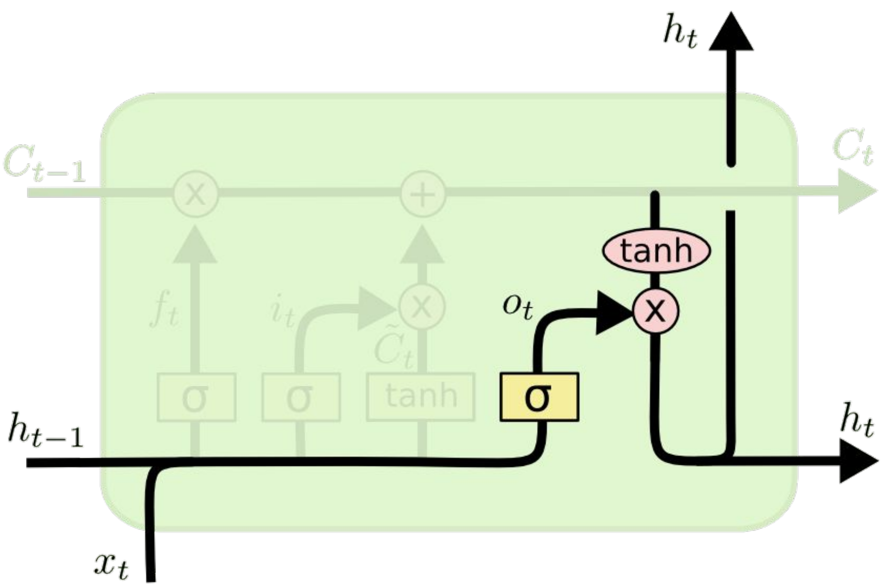


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$



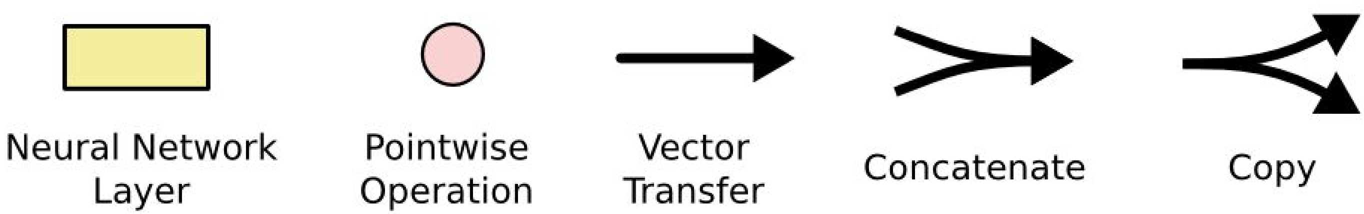
LSTM: Long Short-Term Memory

from [colah's blog](http://colah.github.io/posts/2015-08-Understanding-LSTMs/) http://colah.github.io/posts/2015-08-Understanding-LSTMs/



$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$



LSTM: Sampled Wikipedia articles

from Andrej Karpathy blog http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Naturalism and decision for the majority of Arab countries' capitalide was grounded by the Irish language by [[John Clair]], [[An Imperial Japanese Revolt]], associated with Guangzham's sovereignty. His generals were the powerful ruler of the Portugal in the [[Protestant Immineners]], which could be said to be directly in Cantonese Communication, which followed a ceremony and set inspired prison, training. The emperor travelled back to [[Antioch, Perth, October 25|21]] to note, the Kingdom of Costa Rica, unsuccessful fashioned the [[Thrales]], [[Cynth's Dajoard]], known in western [[Scotland]], near Italy to the conquest of India with the conflict.

LSTM: Sampled Linux source code

from [Andrej Karpathy blog](http://karpathy.github.io/2015/05/21/rnn-effectiveness/) http://karpathy.github.io/2015/05/21/rnn-effectiveness/

```c
/*
 * Increment the size file of the new incorrect UI_FILTER group information
 * of the size generatively.
 */
static int indicate_policy(void)
{
  int error;
  if (fd == MARN_EPT) {
    /*
     * The kernel blank will coeld it to userspace.
     */
    if (ss->segment < mem_total)
      unblock_graph_and_set_blocked();
    else
      ret = 1;
    goto bail;
  }
  segaddr = in_SB(in.addr);
  selector = seg / 16;
  setup_works = true;
  for (i = 0; i < blocks; i++) {
    seq = buf[i++];
    bpf = bd->bd.next + i * search;
    if (fd) {
      current = blocked;
    }
  }
  rw->name = "Getjbbregs";
  bprm_self_clearl(&iv->version);
  regs->new = blocks[(BPF_STATS << info->historidac)] | PFMR_CLOBATHINC_SECONDS << 12;
  return segtable;
}
```


```c
segaddr = in_SB(in.addr);
selector = seg / 16;
setup_works = true;
for (i = 0; i < blocks; i++) {
  seq = buf[i++];
  bpf = bd->bd.next + i * search;
  if (fd) {
    current = blocked;
  }
}
```

Unidad 6: Redes Recurrentes

Curso: Redes Neuronales Profundas

Ejemplo en Keras

- Recurrent Neural Network (RNN)
- Long Short Term Memory (LSTM)
- Sentiment analysis on movie reviews
- Large Movie Review Dataset (IMDB dataset)
- Task: given a movie review, predict whether it is positive or negative.

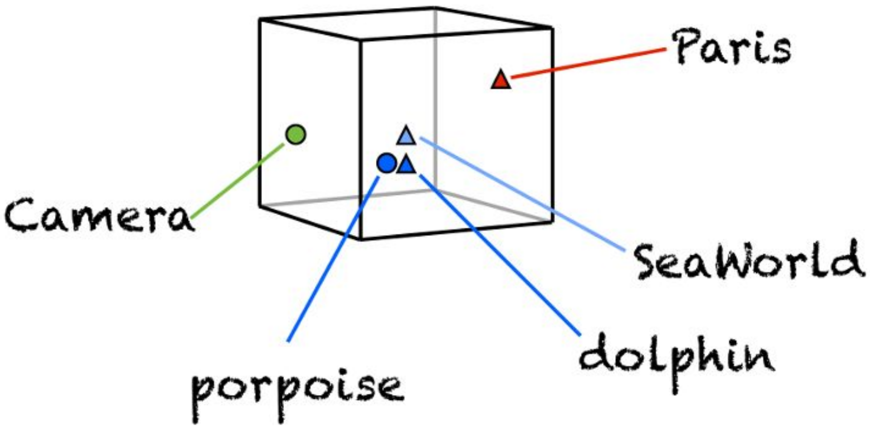
Ejemplo en Keras

```
import numpy as np
from keras.preprocessing import sequence
from keras.models import Sequential
from keras.layers import Embedding, Dense, LSTM
from keras.datasets import imdb
```

Ejemplo en Keras

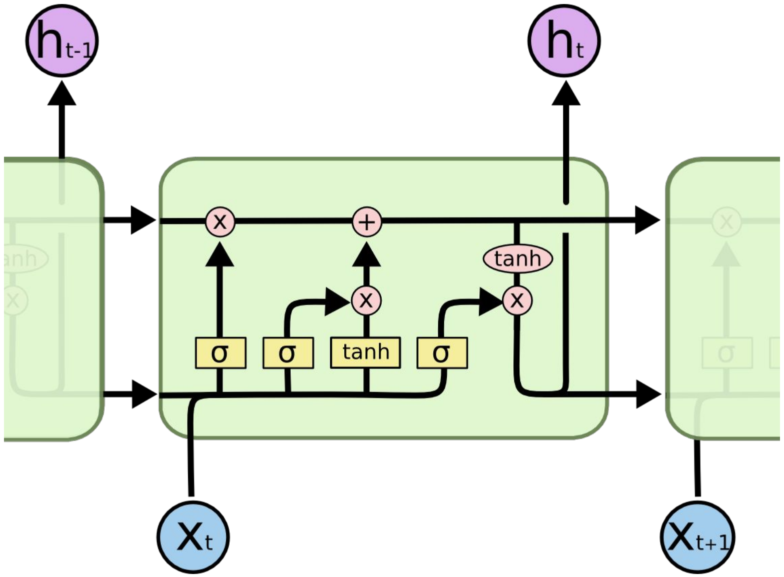
```
max_features = 20000
max_length = 80



print('Loading data...')
(X_train, y_train), (X_test, y_test) =
           imdb.load_data(nb_words=max_features)
X_train = sequence.pad_sequences(X_train, max_length)
X_test = sequence.pad_sequences(X_test, max_length)
```

Ejemplo en Keras




```
max_features = 20000
max_length = 80
embedding_dim = 256

model = Sequential()
model.add(Embedding(max_features, embedding_dim,
                    input_length=max_length,
                    dropout=0.2))
```

Ejemplo en Keras




```
max_features = 20000
max_length = 80
embedding_dim = 256

model = Sequential()
model.add(Embedding(max_features, embedding_dim,
                    input_length=max_length,
                    dropout=0.2))
model.add(LSTM(output_dim=embedding_dim,
               dropout_W=0.2,
               dropout_U=0.2,
               consume_less='gpu', unroll=False))
```

Ejemplo en Keras

```
model = Sequential()
model.add(Embedding(max_features, embedding_dim,
                    input_length=max_length,
                    dropout=0.2))
model.add(LSTM(output_dim=embedding_dim, dropout_W=0.2,
               dropout_U=0.2, consume_less=mode))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

Ejemplo en Keras

```
start_time = time.time()

mfit = model.fit(X_train, y_train,
                 batch_size=batch_size,
                 nb_epoch=epochs,
                 validation_data=(X_test, y_test))

average_time_per_epoch = (time.time()-start_time)/epochs
```

Testing mode = 'cpu'

Train on 25000 samples, validate on 25000 samples

```
Epoch 1/10 - 35s - loss: 0.5445 - acc: 0.7162 - val_loss: 0.3777 - val_acc: 0.8338
Epoch 2/10 - 35s - loss: 0.3812 - acc: 0.8404 - val_loss: 0.3804 - val_acc: 0.8330
Epoch 3/10 - 35s - loss: 0.3176 - acc: 0.8677 - val_loss: 0.3713 - val_acc: 0.8410
Epoch 4/10 - 35s - loss: 0.2674 - acc: 0.8898 - val_loss: 0.4061 - val_acc: 0.8317
Epoch 5/10 - 35s - loss: 0.2286 - acc: 0.9102 - val_loss: 0.4070 - val_acc: 0.8305
Epoch 6/10 - 35s - loss: 0.1903 - acc: 0.9277 - val_loss: 0.4435 - val_acc: 0.8266
Epoch 7/10 - 35s - loss: 0.1622 - acc: 0.9375 - val_loss: 0.5855 - val_acc: 0.8156
Epoch 8/10 - 35s - loss: 0.1412 - acc: 0.9468 - val_loss: 0.5214 - val_acc: 0.8233
Epoch 9/10 - 35s - loss: 0.1300 - acc: 0.9502 - val_loss: 0.5882 - val_acc: 0.7984
Epoch 10/10- 35s - loss: 0.1170 - acc: 0.9549 - val_loss: 0.5502 - val_acc: 0.8191
```

mode = 'cpu' preprocesses input to the LSTM which typically results in faster computations at the expense of increased peak memory usage as the preprocessed input must be kept in memory.

Testing mode = 'mem'

Train on 25000 samples, validate on 25000 samples

```
Epoch 1/10 - 34s - loss: 0.5531 - acc: 0.7056 - val_loss: 0.4007 - val_acc: 0.8204
Epoch 2/10 - 34s - loss: 0.3868 - acc: 0.8360 - val_loss: 0.3890 - val_acc: 0.8268
Epoch 3/10 - 34s - loss: 0.3237 - acc: 0.8625 - val_loss: 0.4675 - val_acc: 0.8200
Epoch 4/10 - 34s - loss: 0.2717 - acc: 0.8910 - val_loss: 0.3785 - val_acc: 0.8323
Epoch 5/10 - 34s - loss: 0.2288 - acc: 0.9105 - val_loss: 0.4086 - val_acc: 0.8367
Epoch 6/10 - 34s - loss: 0.1956 - acc: 0.9249 - val_loss: 0.4562 - val_acc: 0.8263
Epoch 7/10 - 34s - loss: 0.1615 - acc: 0.9382 - val_loss: 0.4849 - val_acc: 0.8222
Epoch 8/10 - 34s - loss: 0.1377 - acc: 0.9480 - val_loss: 0.4977 - val_acc: 0.8213
Epoch 9/10 - 34s - loss: 0.1241 - acc: 0.9537 - val_loss: 0.5055 - val_acc: 0.8166
Epoch 10/10- 34s - loss: 0.1118 - acc: 0.9574 - val_loss: 0.5692 - val_acc: 0.8201
```

mode = 'mem' does away with the preprocessing, meaning that it might take a little longer, but should require less peak memory.

Testing mode = 'gpu'

Train on 25000 samples, validate on 25000 samples

```
Epoch 1/10 - 20s - loss: 0.4951 - acc: 0.7595 - val_loss: 0.4459 - val_acc: 0.8022
Epoch 2/10 - 20s - loss: 0.3549 - acc: 0.8503 - val_loss: 0.3978 - val_acc: 0.8324
Epoch 3/10 - 20s - loss: 0.2939 - acc: 0.8785 - val_loss: 0.3868 - val_acc: 0.8299
Epoch 4/10 - 20s - loss: 0.2479 - acc: 0.9032 - val_loss: 0.3935 - val_acc: 0.8301
Epoch 5/10 - 20s - loss: 0.2161 - acc: 0.9142 - val_loss: 0.4458 - val_acc: 0.8222
Epoch 6/10 - 20s - loss: 0.1896 - acc: 0.9252 - val_loss: 0.4551 - val_acc: 0.8246
Epoch 7/10 - 20s - loss: 0.1699 - acc: 0.9371 - val_loss: 0.4537 - val_acc: 0.8138
Epoch 8/10 - 20s - loss: 0.1446 - acc: 0.9453 - val_loss: 0.5767 - val_acc: 0.8053
Epoch 9/10 - 20s - loss: 0.1327 - acc: 0.9498 - val_loss: 0.4977 - val_acc: 0.8118
Epoch 10/10- 20s - loss: 0.1164 - acc: 0.9566 - val_loss: 0.5607 - val_acc: 0.8128
```

mode = 'gpu' concatenates the input, output and forget gate's weights into one, large matrix, resulting in faster computation time as the GPU can utilize more cores, at the expense of reduced regularization because the same dropout is shared across the gates.

Image Captioning

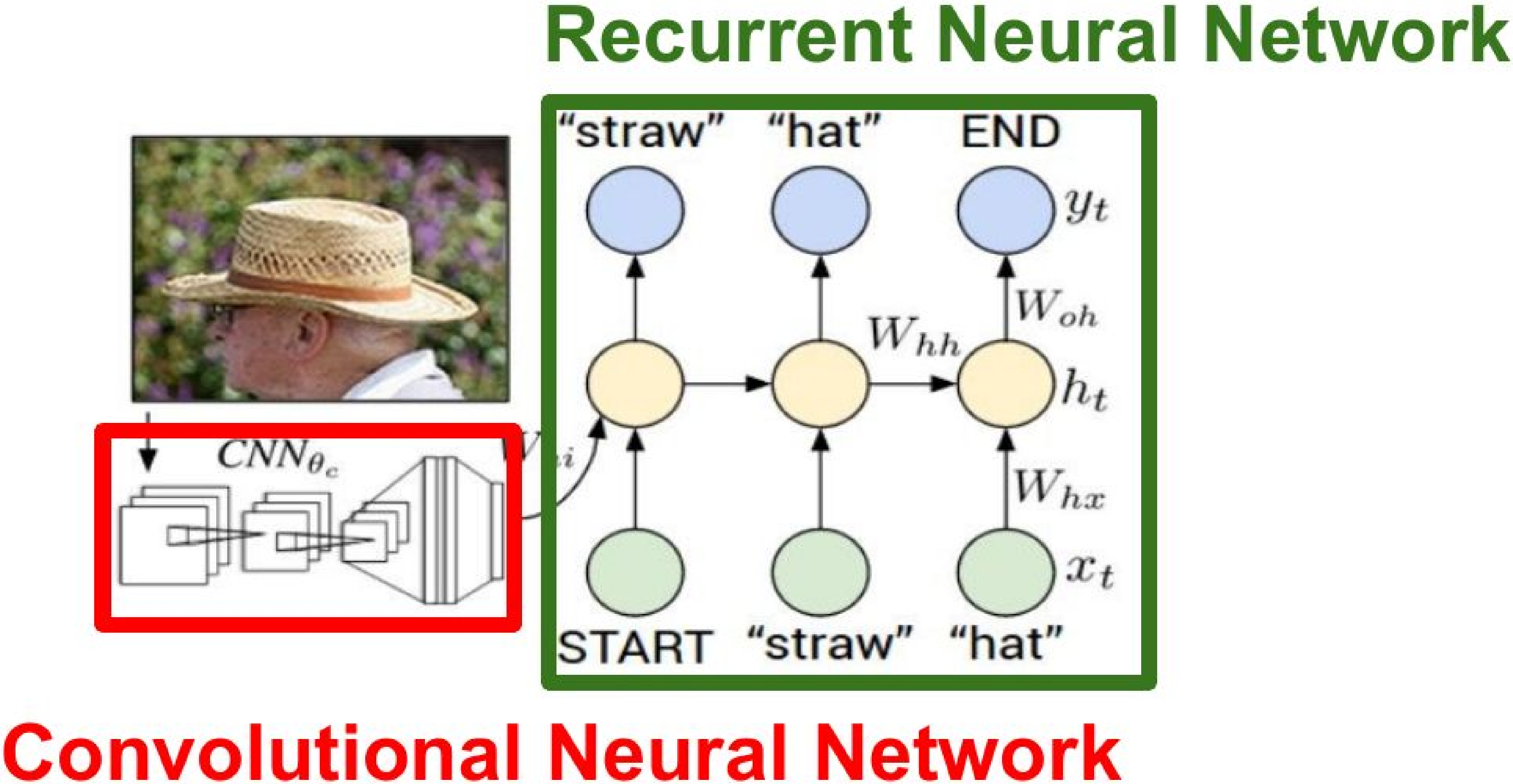
Curso: Redes Neuronales Profundas

a man riding a bike on a dirt path through a forest.
bicyclist raises his fist as he rides on desert dirt trail.
this dirt bike rider is smiling and raising his fist in triumph.
a man riding a bicycle while pumping his fist in the air.
a mountain biker pumps his fist in celebration.

a man riding a bike on a dirt path through a forest.
bicyclist raises his fist as he rides on desert dirt trail.
this dirt bike rider is smiling and raising his fist in triumph.
a man riding a bicycle while pumping his fist in the air.
a mountain biker pumps his fist in celebration.

MS COCO - mscoco.org
Alrededor de 300K imágenes

Estructura general



del curso de Fei-Fei, Karpathy y Johnson http://cs231n.stanford.edu

Cómo funciona en test

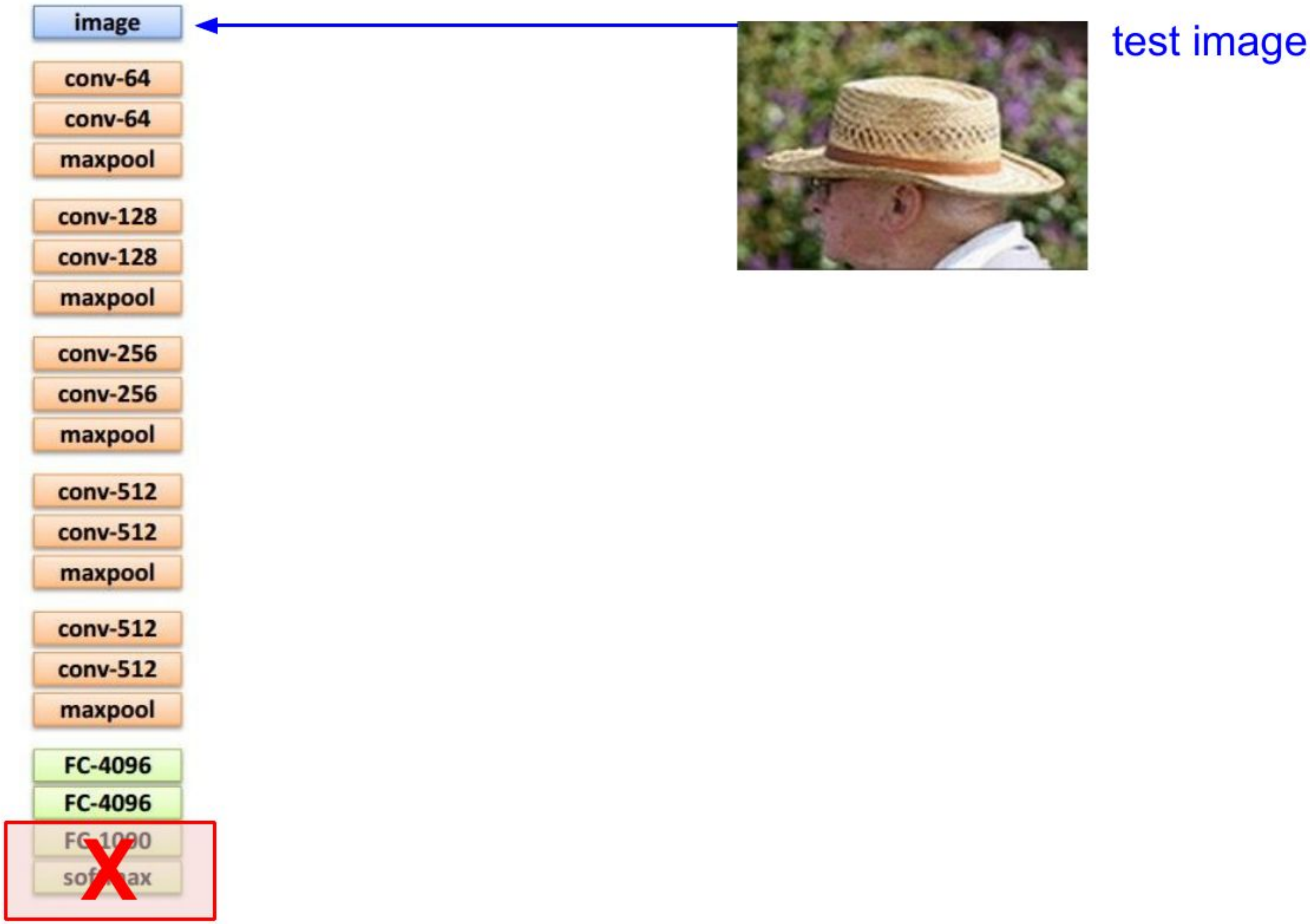
test image

del curso de Fei-Fei, Karpathy y Johnson http://cs231n.stanford.edu

Cómo funciona en test

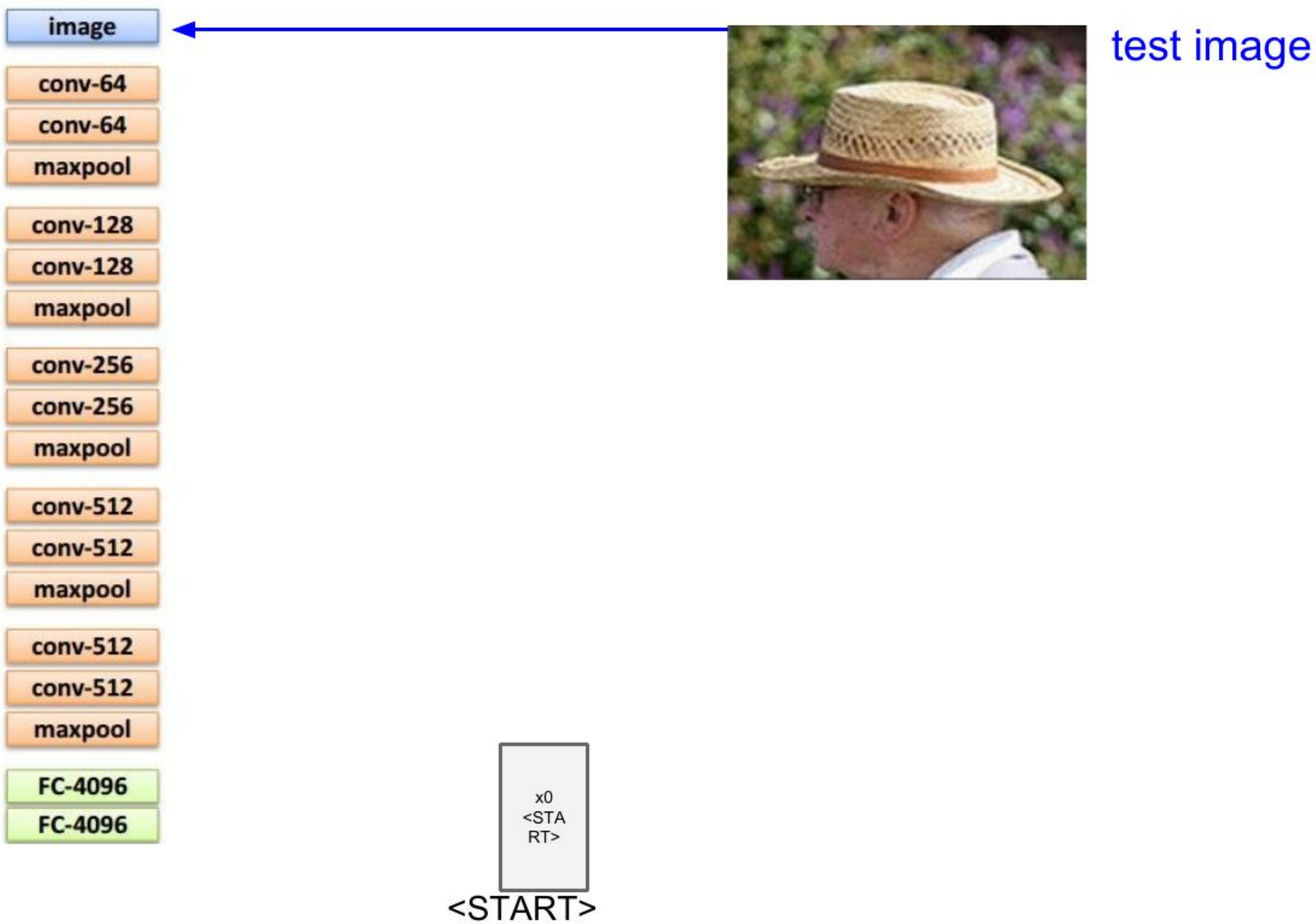
del curso de Fei-Fei, Karpathy y Johnson http://cs231n.stanford.edu

Cómo funciona en test



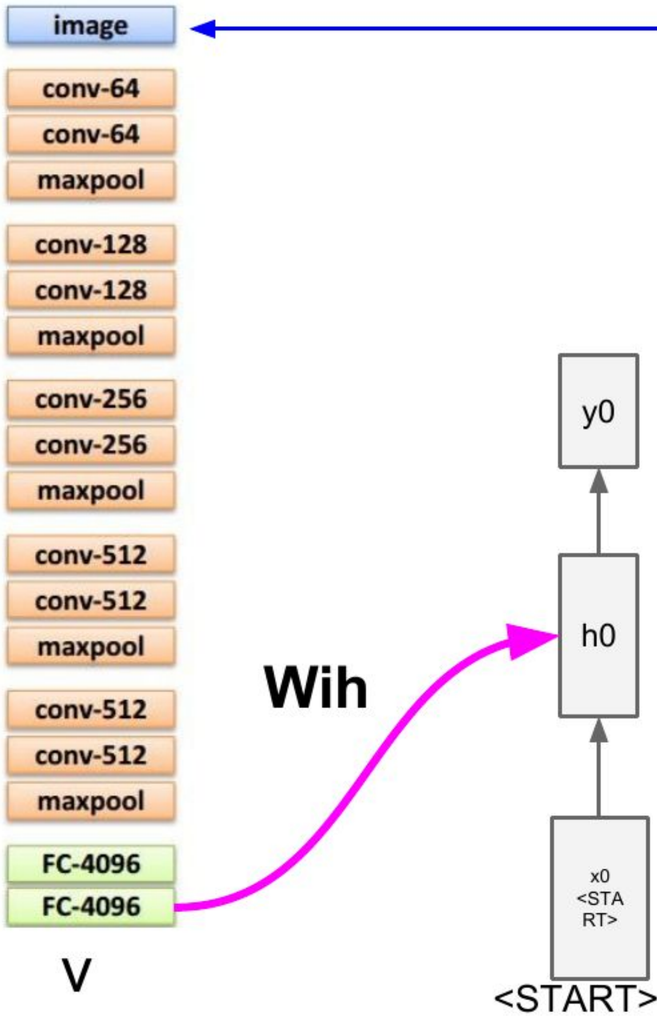
del curso de Fei-Fei, Karpathy y Johnson http://cs231n.stanford.edu

Cómo funciona en test



del curso de Fei-Fei, Karpathy y Johnson http://cs231n.stanford.edu

Cómo funciona en test



test image

before:

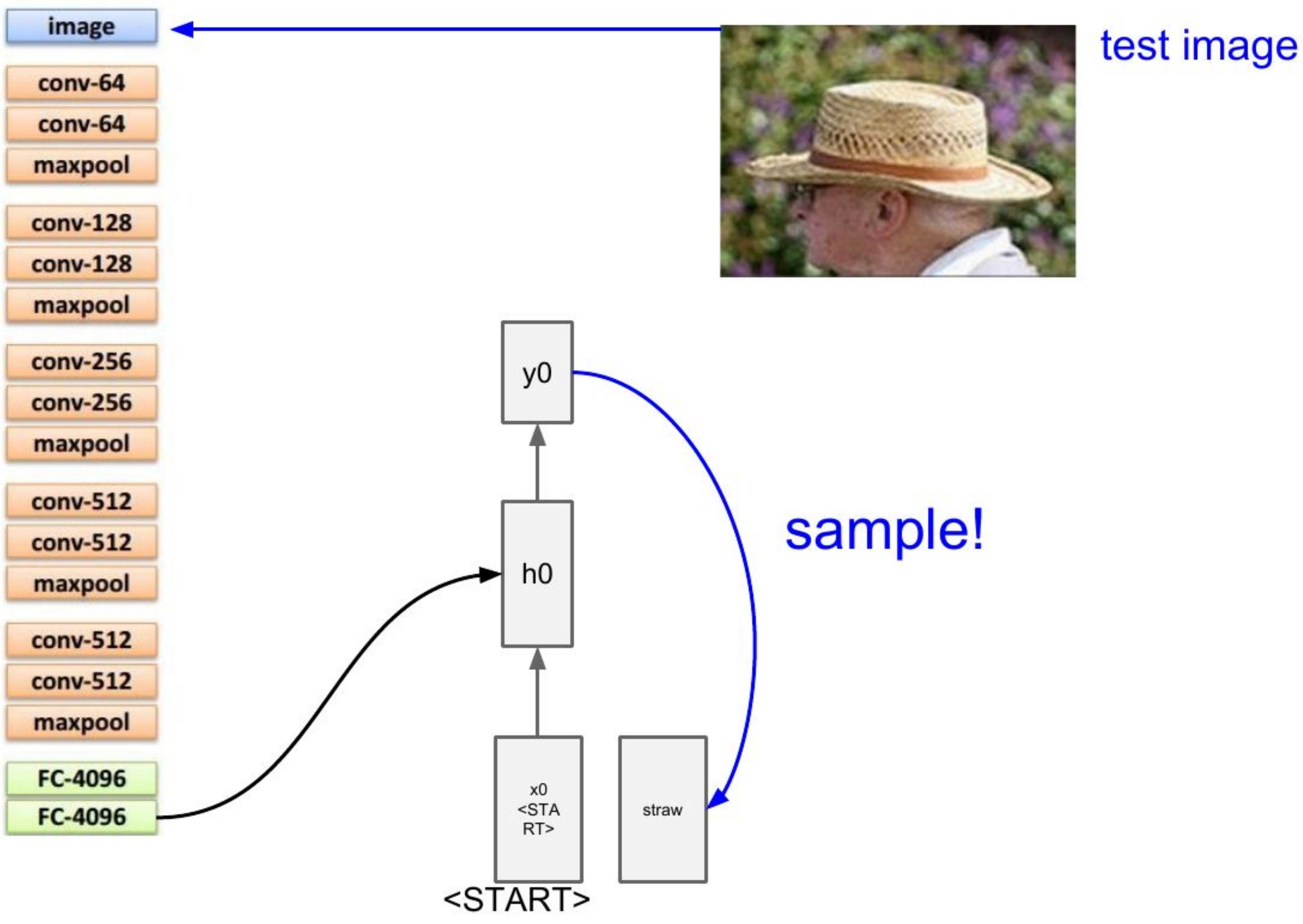
h = tanh(Wxh * x + Whh * h)

now:

h = tanh(Wxh * x + Whh * h **+ Wih * v**)

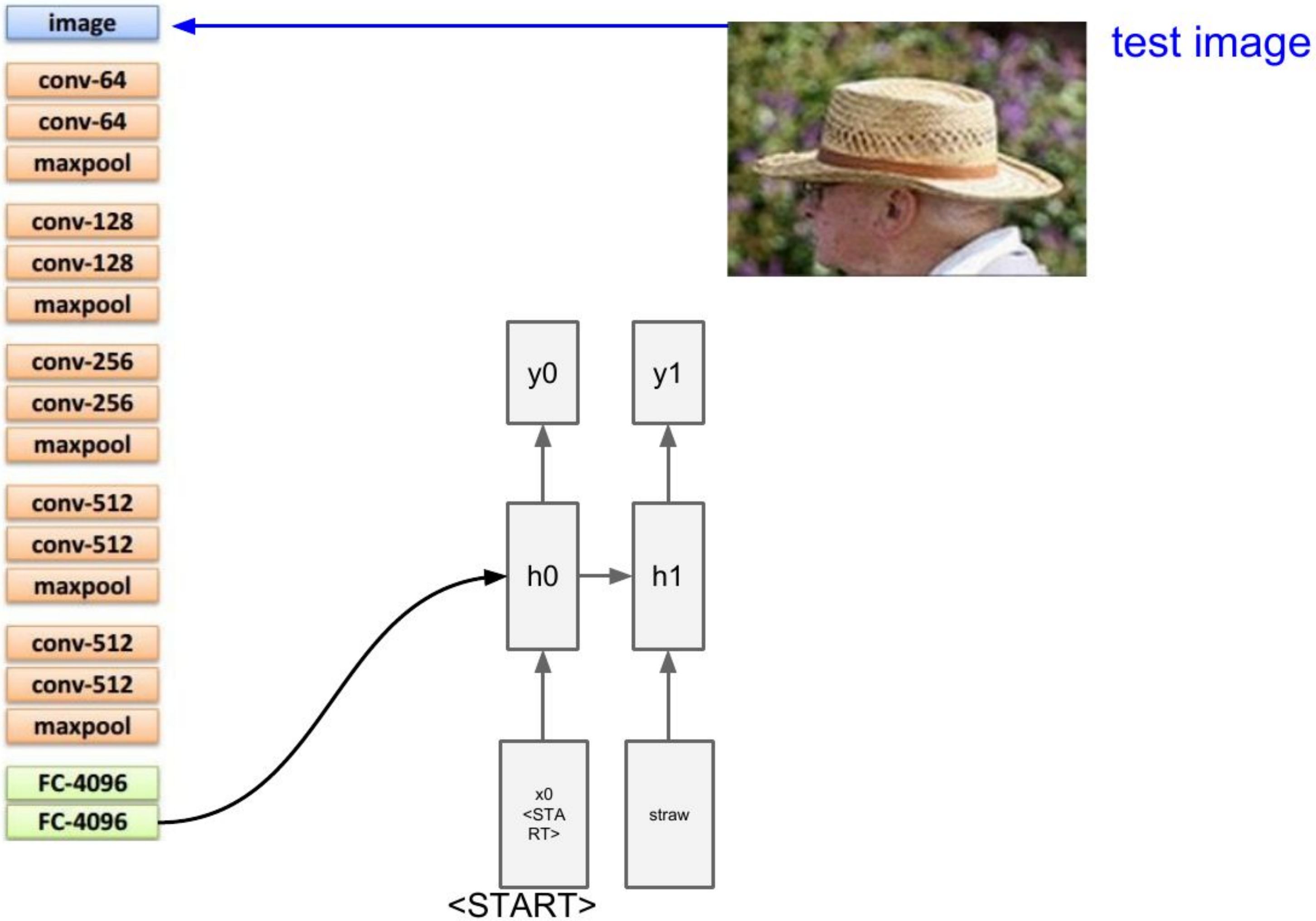
del curso de Fei-Fei, Karpathy y Johnson http://cs231n.stanford.edu

Cómo funciona en test



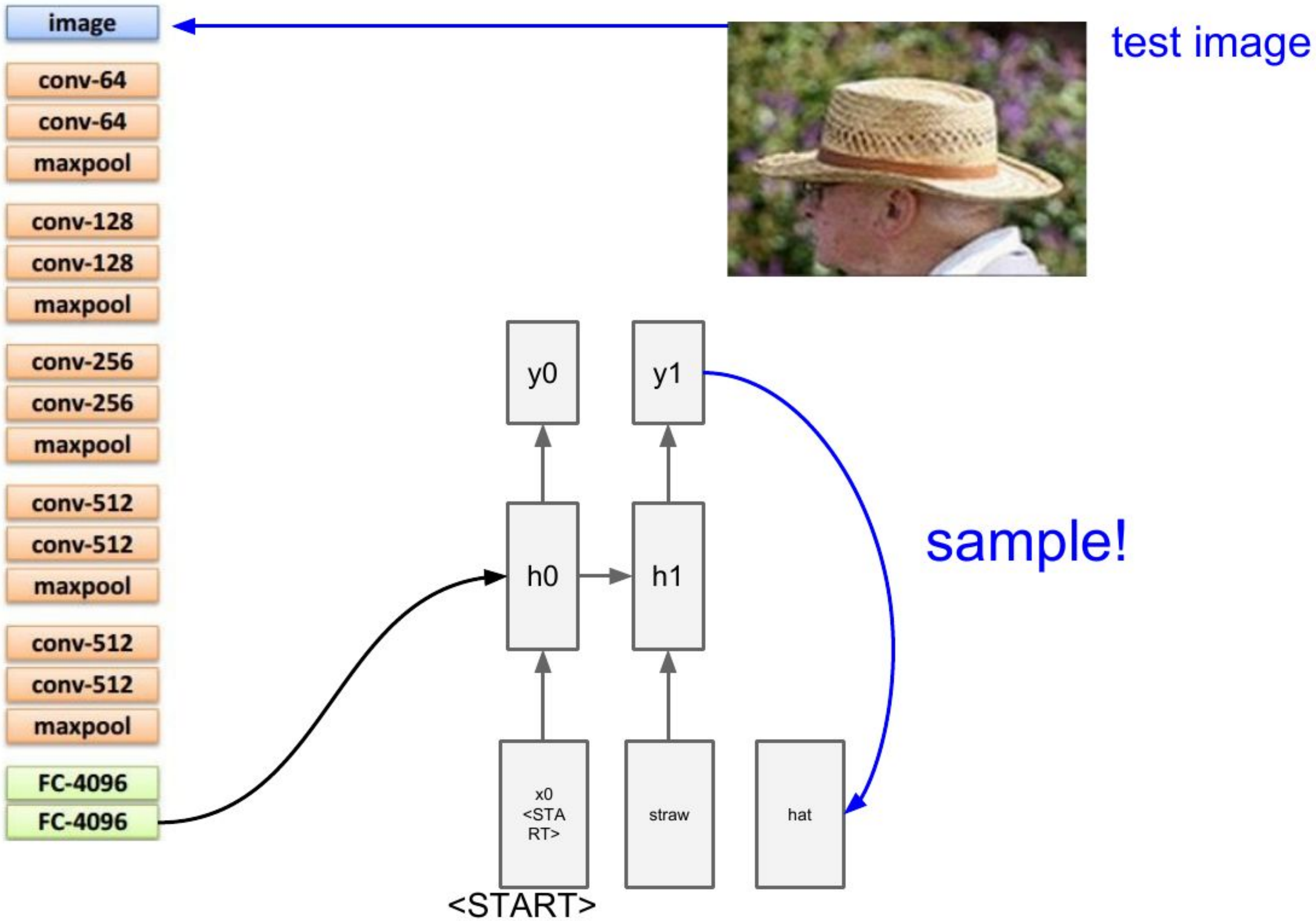
del curso de Fei-Fei, Karpathy y Johnson http://cs231n.stanford.edu

Cómo funciona en test



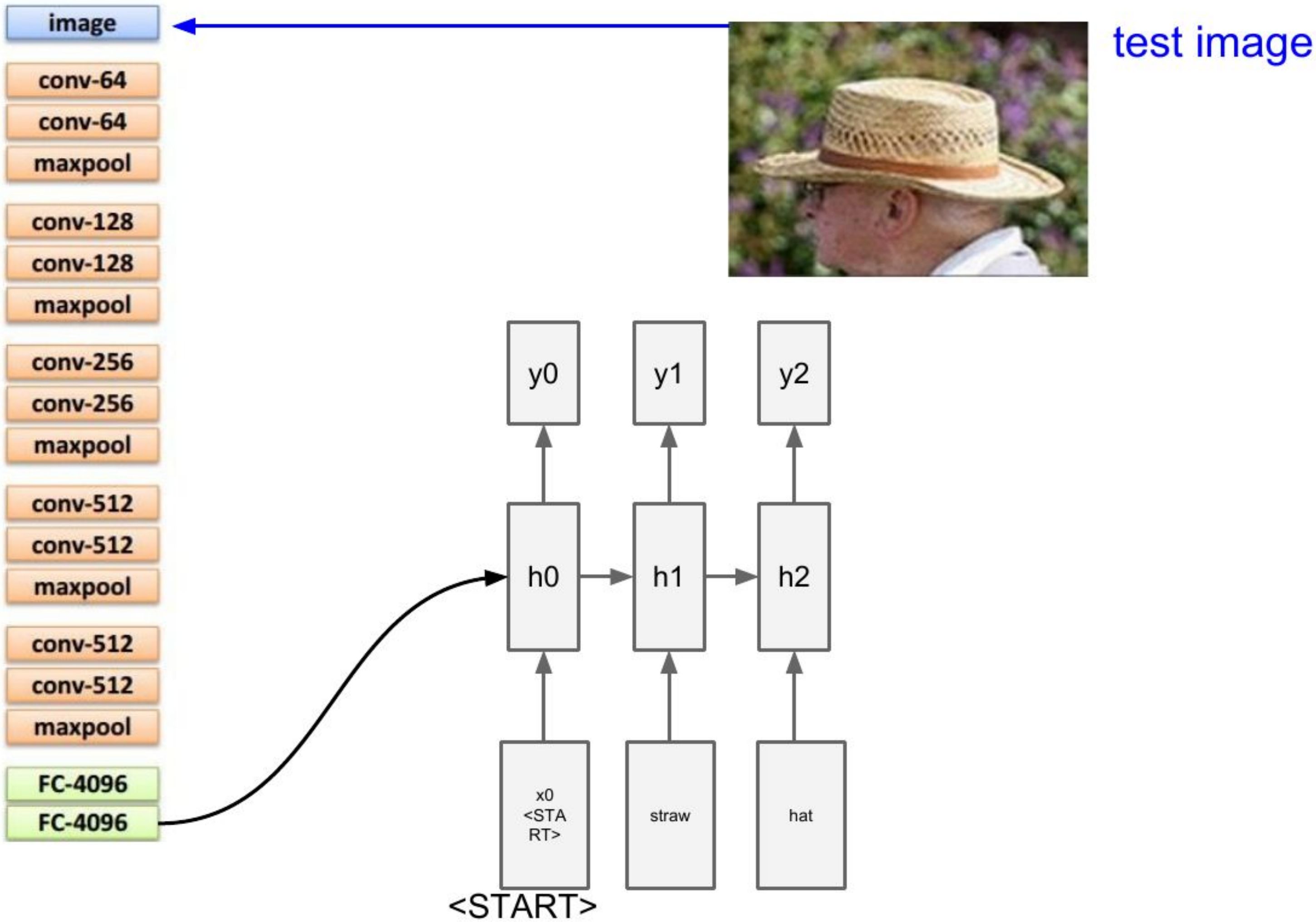
del curso de Fei-Fei, Karpathy y Johnson http://cs231n.stanford.edu

Cómo funciona en test



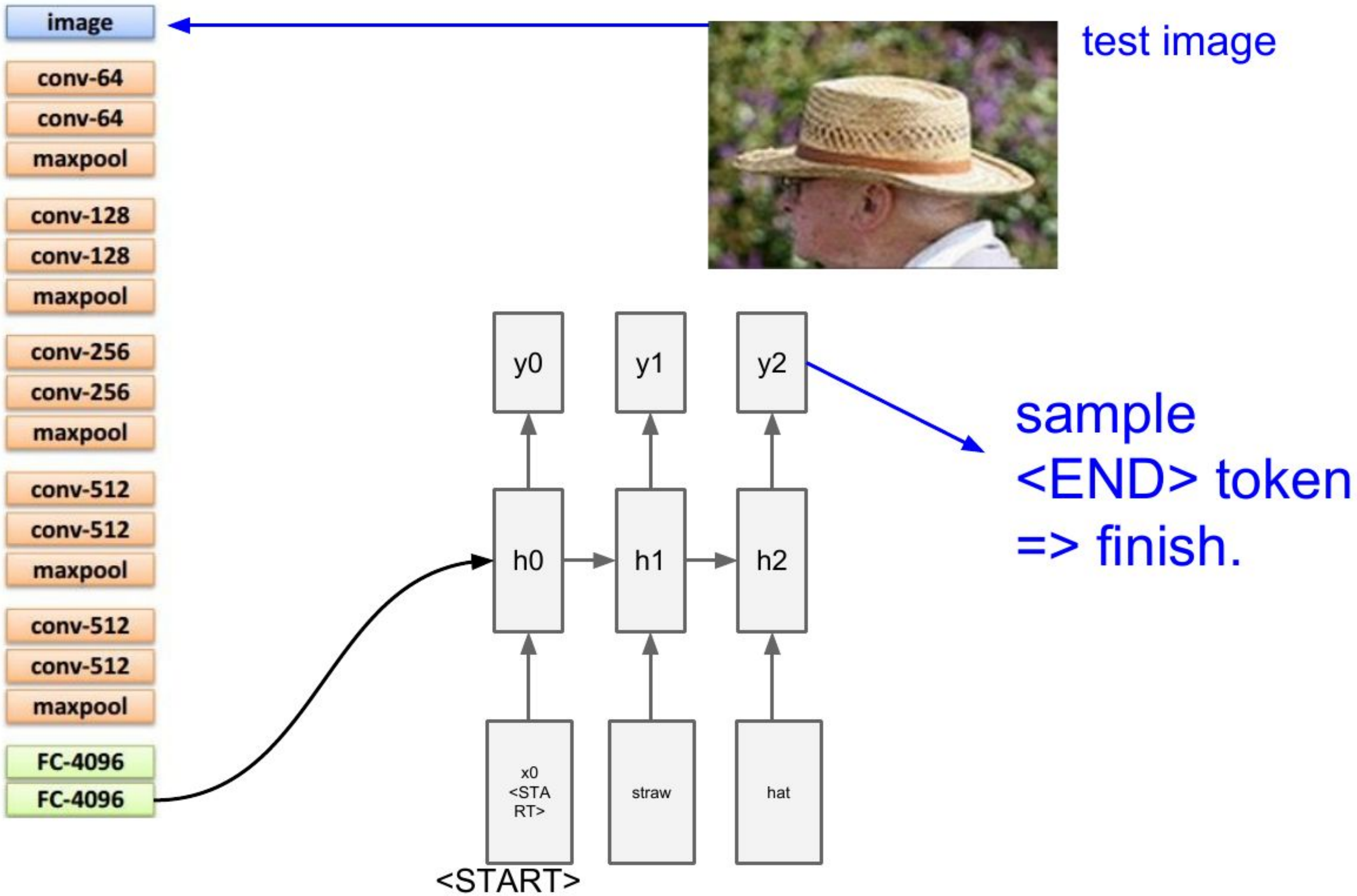
del curso de Fei-Fei, Karpathy y Johnson http://cs231n.stanford.edu

Cómo funciona en test



del curso de Fei-Fei, Karpathy y Johnson http://cs231n.stanford.edu

Cómo funciona en test

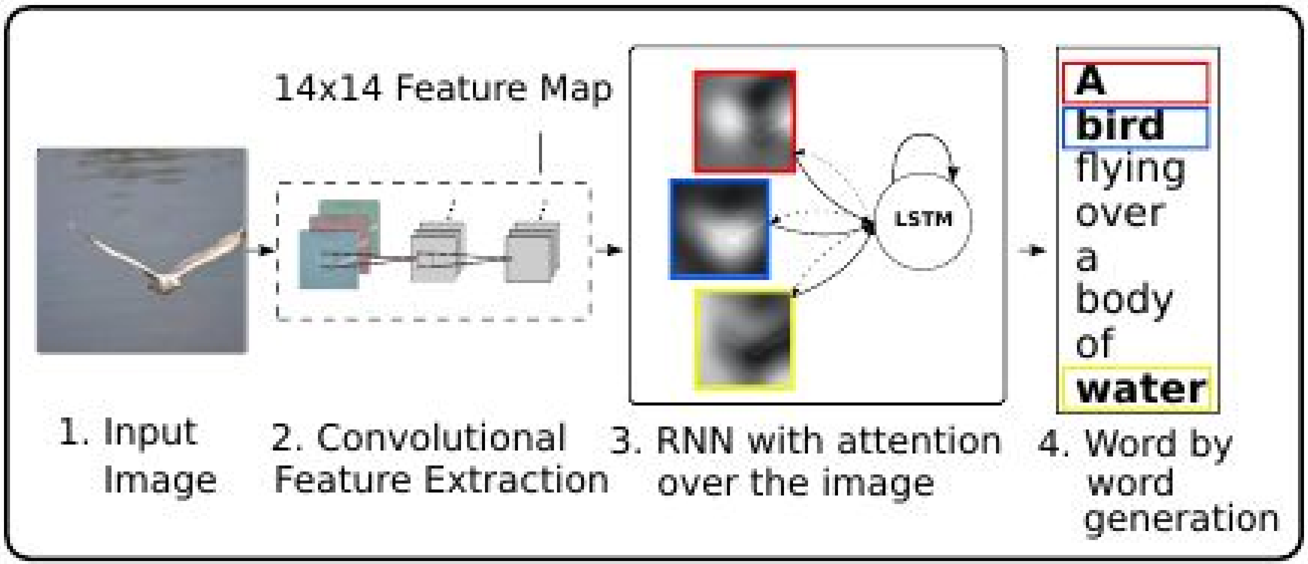


del curso de Fei-Fei, Karpathy y Johnson http://cs231n.stanford.edu

Algunos Resultados

Vinyals, O., Toshev, A., Bengio, S., & Erhan, D. (2015). Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 3156-3164)

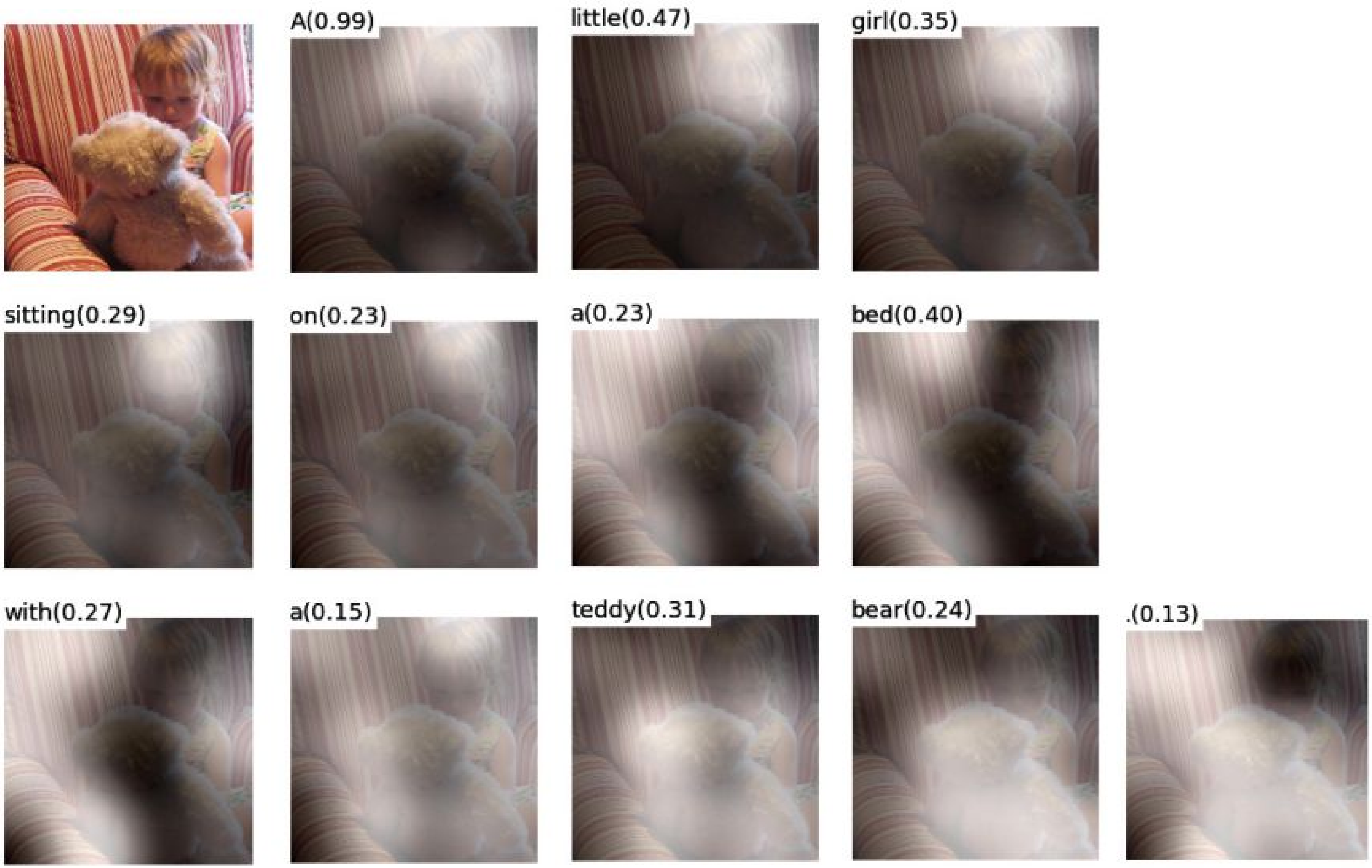
Modelos más complejos



Agregado de un mecanismo de “atención”: una máscara que multiplica al resultado de la CNN, que se actualiza en cada paso

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhutdinov, R., ... & Bengio, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention. *arXiv preprint arXiv:1502.03044*, *2*(3), 5.

A little girl sitting on a bed with a teddy bear.

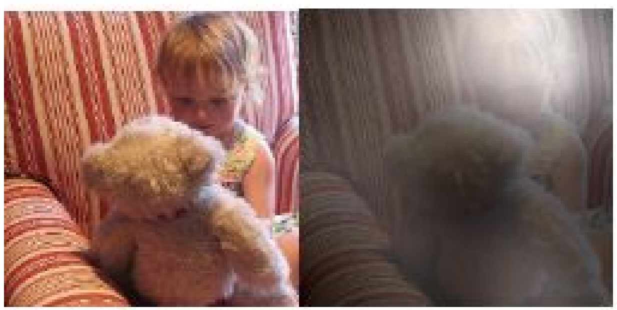


Modelos más complejos

A woman is throwing a frisbee in a park.

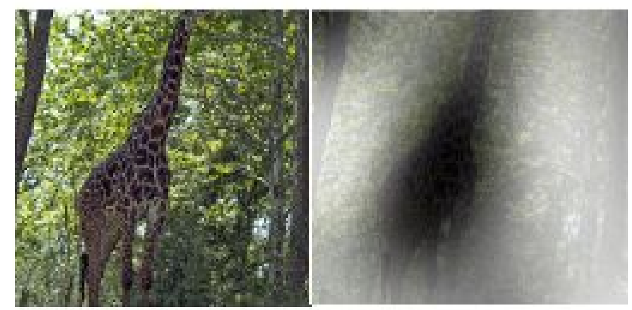
A dog is standing on a hardwood floor.

A stop sign is on a road with a mountain in the background.



A little girl sitting on a bed with a teddy bear.

A group of people sitting on a boat in the water.



A giraffe standing in a forest with trees in the background.

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhutdinov, R., ... & Bengio, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention. *arXiv preprint arXiv:1502.03044*, *2*(3), 5.

¿Cómo medir performance?

Candidate: `the the the the the the the.`

Reference 1: `The cat is on the mat.`

Reference 2: `There is a cat on the mat.`

Candidate 1: `I always invariably perpetu- ally do.`

Candidate 2: `I always do.`

Reference 1: `I always do.`

Reference 2: `I invariably do.`

Reference 3: `I perpetually do.`

¿Cómo medir performance?

Métricas: BLEU-1, …, BLEU-4

Papineni, Kishore, et al. "BLEU: a method for automatic evaluation of machine translation." *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, 2002.

METEOR

Denkowski, Michael, and Alon Lavie. "Meteor universal: Language specific translation evaluation for any target language." In Proceedings of the Ninth Workshop on Statistical Machine Translation. 2014.