TAMPERE UNIVERSITY OF TECHNOLOGY

SIGNAL INTERPRETATION

Lecture 6: ConvNets

February 11, 2016

Heikki Huttunen

heikki.huttunen@tut.fi

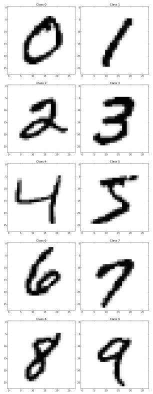
Department of Signal Processing
Tampere University of Technology

CONVNETS

Continued from previous slideset

Convolutional Network: Example

- Let's train a convnet with the famous MNIST dataset.
- MNIST consists of 60000 training and 10000 test images representing handwritten numbers from US mail.
- Each image is 28 × 28 pixels and there are 10 categories.
- Generally considered an easy problem: Logistic regression gives over 90% accuracy and convnet can reach (almost) 100%.
- However, 10 years ago, the state of the art error was still over 1%.



Convolutional Network: Example

```
# Training code (modified from mnist_cnn.py at
       Keras examples)
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers.core import Dense, Dropout,
       Activation, Flatten
from keras.layers.convolutional import
       Convolution2D, MaxPooling2D

# We use the handwritten digit database "MNIST".
# 60000 training and 10000 test images of
# size 28x28
(X_train, y_train), (X_test, y_test) = mnist.
       load_data()

num_featmaps = 32   # This many filters per layer
num_classes = 10    # Digits 0,1,...,9
num_epochs = 50     # Show all samples 50 times
w, h = 5, 5         # Conv window size
```

```
model = Sequential()

# Layer 1: needs input_shape as well.
model.add(Convolution2D(num_featmaps, w, h,
          input_shape=(1, 28, 28),
          activation = 'relu'))

# Layer 2:
model.add(Convolution2D(num_featmaps, w, h,
       activation = 'relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

# Layer 3: dense layer with 128 nodes
# Flatten() vectorizes the data:
# 32x10x10 -> 3200
# (10x10 instead of 14x14 due to border effect)
model.add(Flatten())
model.add(Dense(128, activation = 'relu'))
model.add(Dropout(0.5))

# Layer 4: Last layer producing 10 outputs.
model.add(Dense(num_classes, activation='softmax'))

# Compile and train
model.compile(loss='categorical_crossentropy',
       optimizer='adadelta')
model.fit(X_train, Y_train, nb_epoch=100)
```

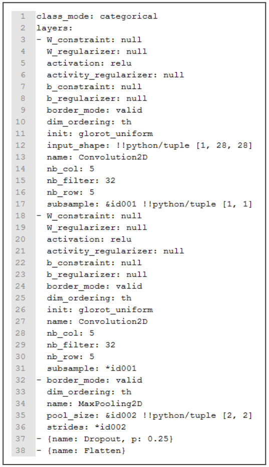
Convolutional Network: Training Log

- The code runs for about 5-10 minutes on a GPU.
- On a CPU, this would take 1-2 hours (1 epoch ≈ 500 s)

```
Using gpu device 0: Tesla K40m
Using Theano backend.
Compiling model...
Model compilation took 0.1 minutes.
Training...
Train on 60000 samples, validate on 10000 samples
Epoch 1/10
60000/60000 [================] - 31s - loss: 0.2193 - acc: 0.9322 - val_loss: 0.0519 - val_acc: 0.9835
Epoch 2/10
60000/60000 [================] - 31s - loss: 0.0807 - acc: 0.9758 - val_loss: 0.0398 - val_acc: 0.9863
Epoch 3/10
60000/60000 [================] - 31s - loss: 0.0581 - acc: 0.9825 - val_loss: 0.0322 - val_acc: 0.9898
Epoch 4/10
60000/60000 [================] - 31s - loss: 0.0500 - acc: 0.9851 - val_loss: 0.0276 - val_acc: 0.9913
Epoch 5/10
60000/60000 [================] - 31s - loss: 0.0430 - acc: 0.9872 - val_loss: 0.0287 - val_acc: 0.9906
Epoch 6/10
60000/60000 [================] - 31s - loss: 0.0387 - acc: 0.9882 - val_loss: 0.0246 - val_acc: 0.9922
Epoch 7/10
60000/60000 [================] - 31s - loss: 0.0352 - acc: 0.9897 - val_loss: 0.0270 - val_acc: 0.9913
Epoch 8/10
60000/60000 [================] - 31s - loss: 0.0324 - acc: 0.9902 - val_loss: 0.0223 - val_acc: 0.9928
Epoch 9/10
60000/60000 [================] - 31s - loss: 0.0294 - acc: 0.9907 - val_loss: 0.0221 - val_acc: 0.9926
Epoch 10/10
60000/60000 [================] - 31s - loss: 0.0252 - acc: 0.9922 - val_loss: 0.0271 - val_acc: 0.9916
Training (10 epochs) took 5.8 minutes.
```

Save and Load the Net

- The network can be saved to disk in two parts:
  - Network topology as JSON or YAML: `model.to_json()` or `model.to_yaml()`. The resulting string can be written to disk using `.write()` of a file object.
  - Coefficients are saved in HDF5 format using `model.save_weights()`. HDF5 is a serialization format similar to `.mat` or `.pkl`
- Alternatively, the net can be pickled, although this is not recommended.
- Read back to memory using `model_from_json` and `load_weights`



```
class_mode: categorical
layers:
- W_constraint: null
  W_regularizer: null
  activation: relu
  activity_regularizer: null
  b_constraint: null
  b_regularizer: null
  border_mode: valid
  dim_ordering: th
  init: glorot_uniform
  input_shape: !!python/tuple [1, 28, 28]
  name: Convolution2D
  nb_col: 5
  nb_filter: 32
  nb_row: 5
  subsample: &id001 !!python/tuple [1, 1]
- W_constraint: null
  W_regularizer: null
  activation: relu
  activity_regularizer: null
  b_constraint: null
  b_regularizer: null
  border_mode: valid
  dim_ordering: th
  init: glorot_uniform
  name: Convolution2D
  nb_col: 5
  nb_filter: 32
  nb_row: 5
  subsample: *id001
- border_mode: valid
  dim_ordering: th
  name: MaxPooling2D
  pool_size: &id002 !!python/tuple [2, 2]
  strides: *id002
- {name: Dropout, p: 0.25}
- {name: Flatten}
```

Part of network definition in YAML format.

Network Structure

- It is possible to look into the filters on the convolutional layers.

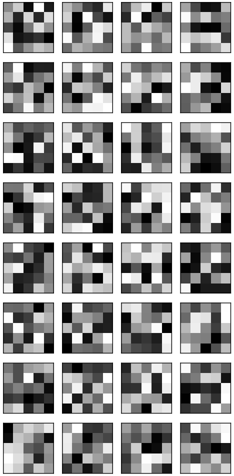
```
# First layer weights (shown on the right):
weights = model.layers[0].get_weights()[0]
```

- The second layer is difficult to visualize, because the input is 32-dimensional:

```
# Zeroth layer weights:
>>> model.layers[0].get_weights()[0].shape
(32, 1, 5, 5)
# First layer weights:
>>> model.layers[1].get_weights()[0].shape
(32, 32, 5, 5)
```

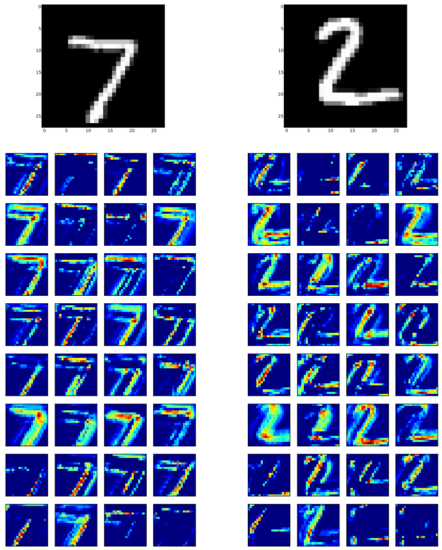
- The dense layer is the 5th (conv → conv → maxpool → dropout → flatten → dense).

```
# Fifth layer weights map 3200 inputs to 128 outputs.
# This is actually a matrix multiplication.
>>> model.layers[5].get_weights()[0].shape
(3200, 128)
```



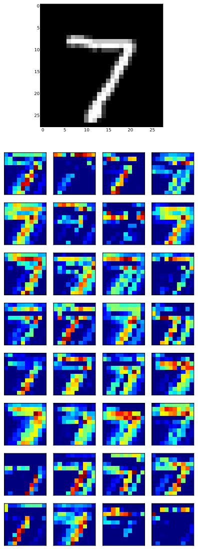
Network Activations

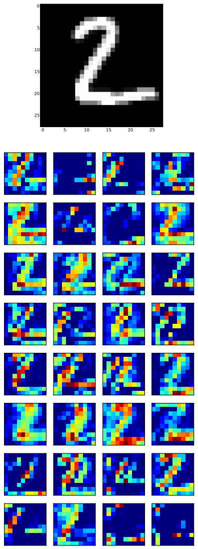
- The layer outputs are usually more interesting than the filters.
- These can be visualized as well.
- For details, see Keras FAQ.
- Note: the outputs are actually grayscale; here in color only for better visualization.



Second Layer Activations

- On the next layer, the figures are downsampled to 12x12.
- This provides *spatial invariance*: The same activation results although the input would be slightly displaced.



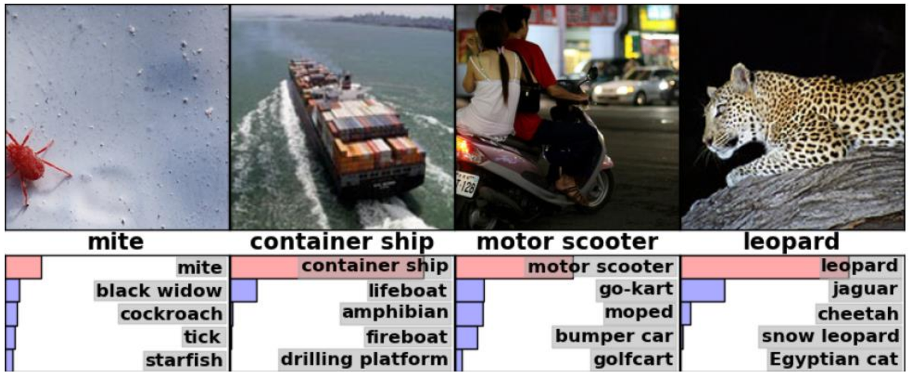


DEEP LEARNING HIGHLIGHTS OF 2015-2016

2015 was Full of Breakthroughs:
Let’s See Some of Them

Image Recognition

- Imagenet is the standard benchmark set for image recognition
- Classify 256x256 images into 1000 categories, such as ”person”, ”bike”, ”cheetah”, etc.
- Total 1.2M images
- Many error metrics, including top-5 error: error rate with 5 guesses



Picture from Alex Krizhevsky et al., ”ImageNet Classification with Deep Convolutional Neural Networks”, 2012

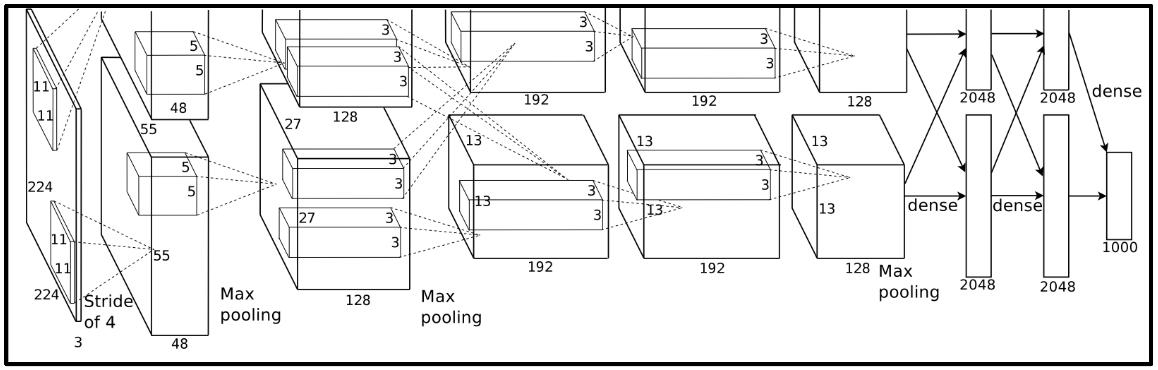
ILSVRC2012

- ILSVRC2012[1] was a game changer
- ConvNets dropped the top-5 error 26.2% → 15.3 %.
- The network is now called *AlexNet* named after the first author (see previous slide).
- Network contains 8 layers (5 convolutional followed by 3 dense); altogether 60M parameters.

[1] Imagenet Large Scale Visual Recognition Challenge

The AlexNet

- The architecture is illustrated in the figure.
- The pipeline is divided to two paths (upper & lower) to fit to 3GB of GPU memory available at the time (running on 2 GPU's)
- Introduced many tricks for *data augmentation*
- Left-right flip
- Crop many subimages (224x224)



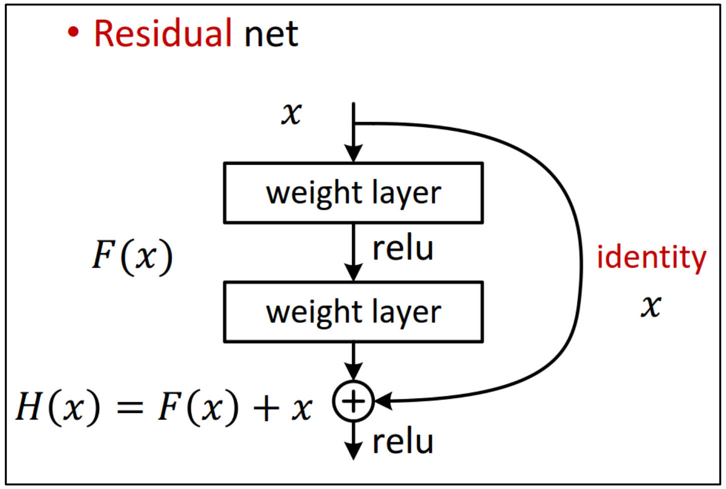
Picture from Alex Krizhevsky et al., "ImageNet Classification with Deep Convolutional Neural Networks", 2012

ILSVRC2014

- Since 2012, ConvNets have dominated
- 2014 there were 2 almost equal teams:
  - GoogLeNet Team with 6.66% Top-5 error
  - VGG Team with 7.33% Top-5 error
- In some subchallenges VGG was the winner
- GoogLeNet: 22 layers, only 7M parameters due to fully convolutional structure and clever *inception* architecture
- VGG: 19 layers, 144M parameters

ILSVRC2015

- Winner MSRA (Microsoft Research) with TOP-5 error 3.57 %
- 152 layers! 51M parameters.
- Built from residual blocks (which include the inception trick from previous year)
- Key idea is to add *identity shortcuts,* which make training easier



TAMPEREEN TEKNILLINEN YLIOPISTO

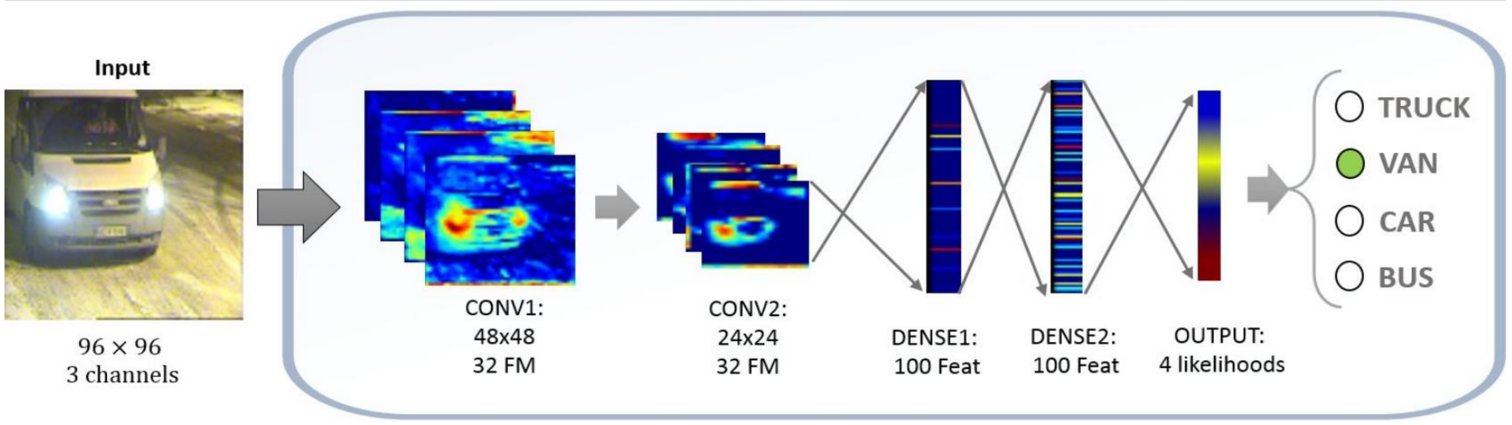
Pictures from MSRA ICCV2015 slides

ILSVRC2016?

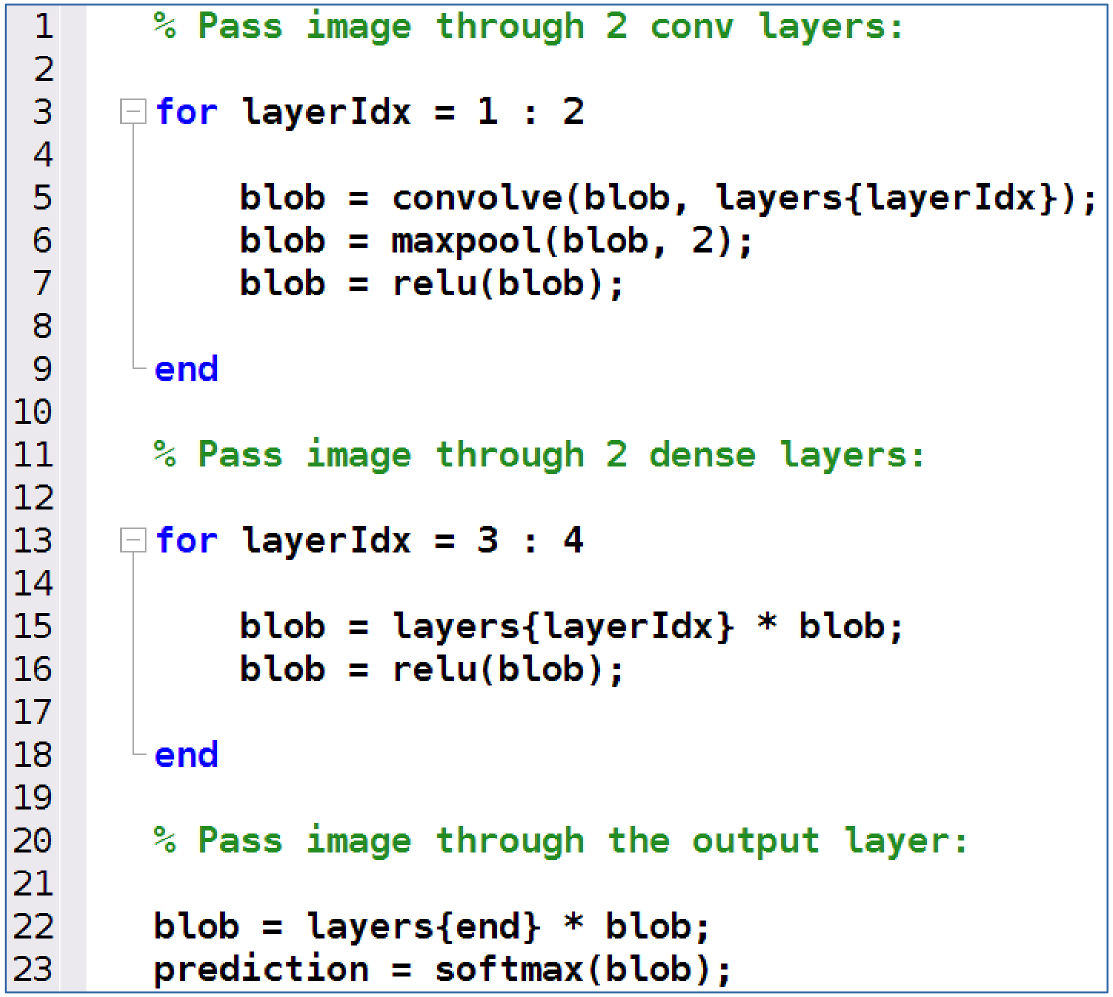
- ILSVRC2016 will be again full of surprises
- Most likely extremely deep nets will dominate
- MSRA experimented already with a 1202 layer net in ILSVRC2016
- Still, too slow, too overfitting, slightly worse than 152 layer model
- These obstacles may be circumvented with new regularizers, multi-GPU and new tools: https://github.com/Microsoft/CNTK

Back to Earth

- TUT has studied shallow convolutional architectures for fast/real time detection tasks
- For example, Automatic Car Type Detection from Picture: **Van, Bus, Truck** or **Normal Vehicle**
- The network recognizes the car type (4 classes) with 98% accuracy (13 000 images).



Components of the Network



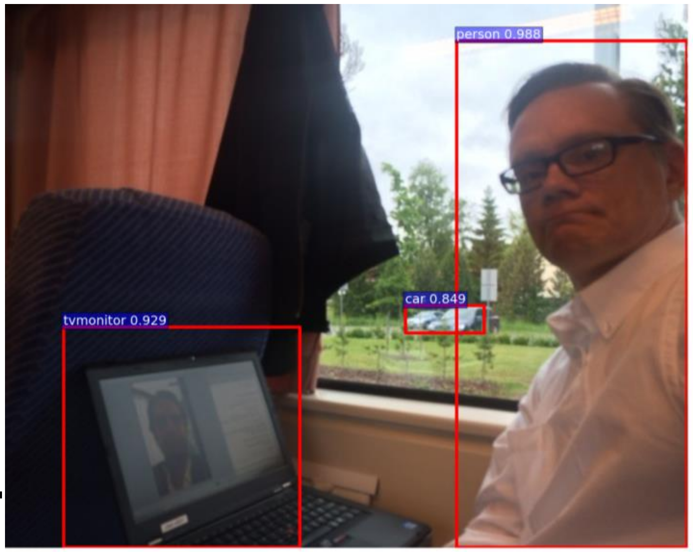
```
% Pass image through 2 conv layers:

for layerIdx = 1 : 2

    blob = convolve(blob, layers{layerIdx});
    blob = maxpool(blob, 2);
    blob = relu(blob);

end

% Pass image through 2 dense layers:

for layerIdx = 3 : 4

    blob = layers{layerIdx} * blob;
    blob = relu(blob);

end

% Pass image through the output layer:

blob = layers{end} * blob;
prediction = softmax(blob);
```

- **Convolution**: 5x5 window
- **Maxpooling**: 2x2 downsampling with the maximum
- **Relu**: max(x, 0)
- **Matrix multiplication**
- **Softmax**:

$$[\sigma(\mathbf{x})]_k = \frac{\exp(x_k)}{\sum \exp(x_k)}$$

Object Localization

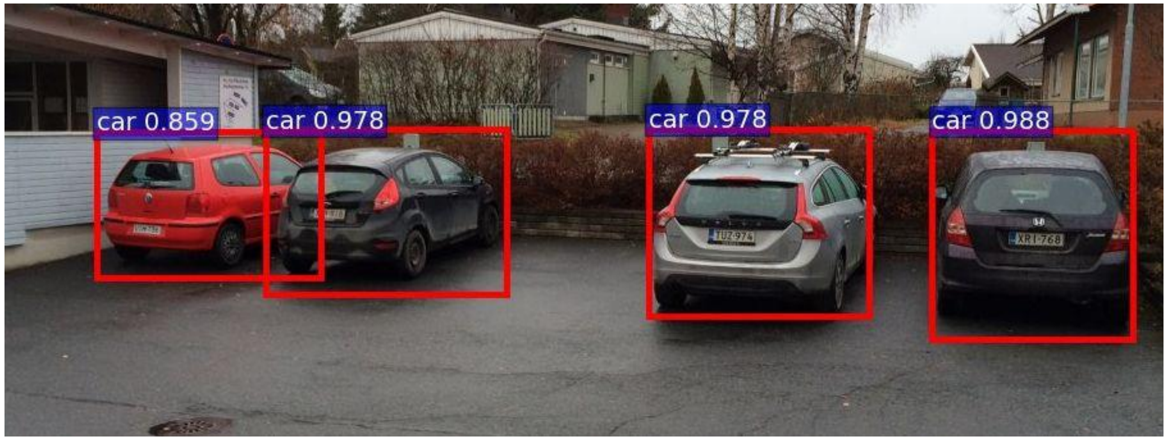
- **Object localization** attempts to find objects from the image together *with their location.*
- The most successful approaches are again deep net based.
- The key question is the speed: it is not possible to use the net at all positions at all scales.
- Instead, the framework relies on **object proposals**: A few hundred bounding boxes that may contain an object.
- Additionally, the process speeds up by processing all conv layers at once.



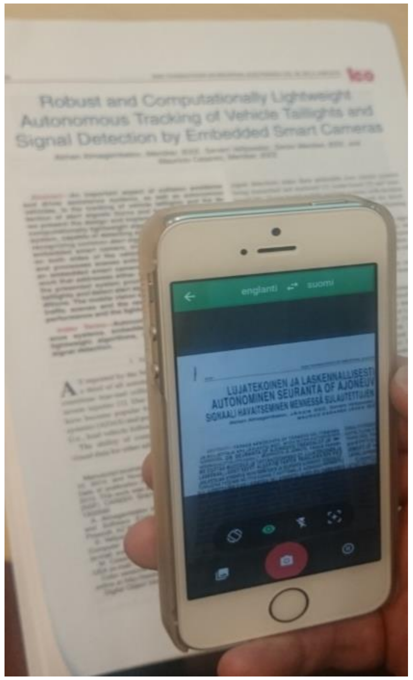
R-CNN

- The tool for ConvNet based object localization is R-CNN (*Regions with CNN*).
- History: CVPR2014: **R-CNN;** ICCV2015: **Fast R-CNN;** NIPS2015: **Faster R-CNN**
- Python implementation of the latter here:

    https://github.com/rbgirshick/py-faster-rcnn



Real-Time Applications



Pedestrian Detection

(Google)

Machine Translation

(Google)

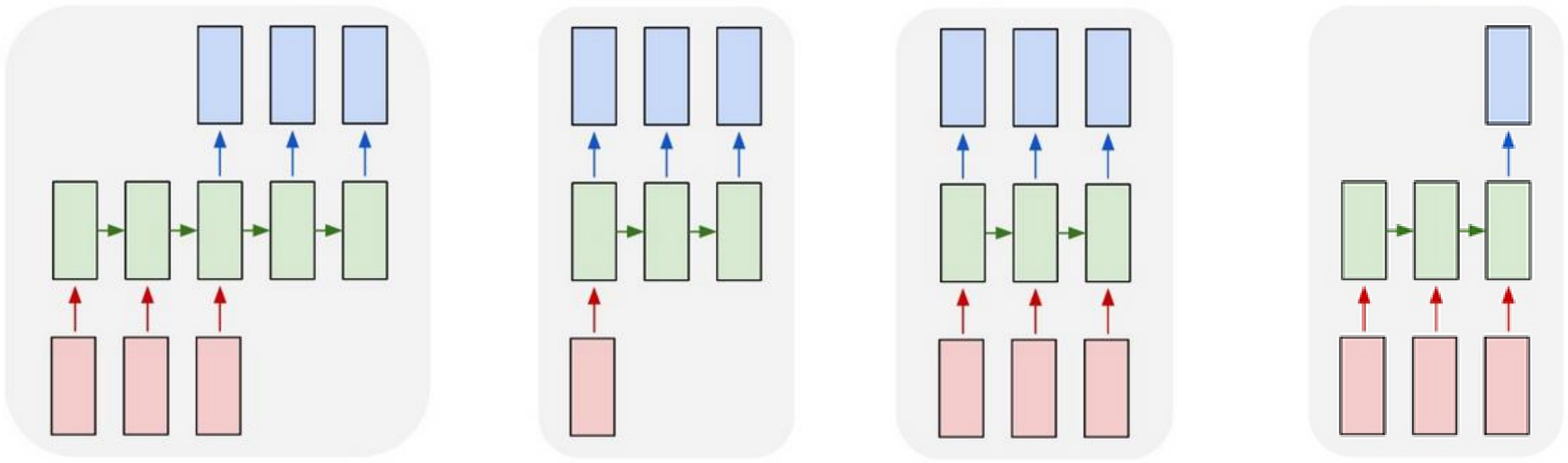
CamFind App

(CloudSight)

TAMPEREEN TEKNILLINEN YLIOPISTO

Recurrent Networks

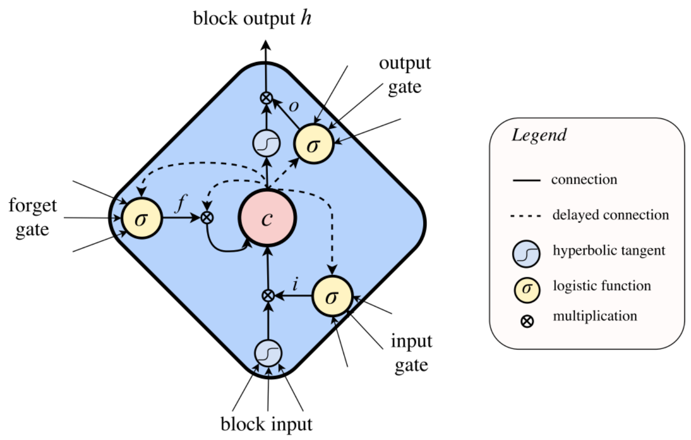
- Recurrent networks process sequences of arbitrary length; *e.g.,*
  - Sequence → sequence
  - Image → sequence
  - Sequence → class ID



Picture from http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Recurrent Networks

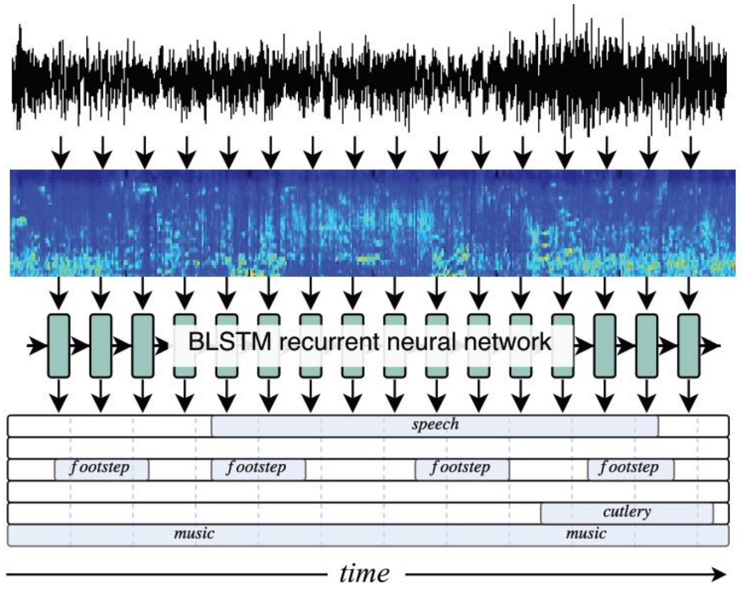
- Recurrent net consist of special nodes that remember past states.
- Each node receives 2 inputs: the data and the previous state.
- Most popular recurrent node type is *Long Short Term Memory* (LSTM) node.
- LSTM includes also *gates*, which can turn on/off the history and a few additional inputs.



Picture from G. Parascandolo M.Sc. Thesis, 2015.
http://urn.fi/URN:NBN:fi:tty-201511241773

Recurrent Networks

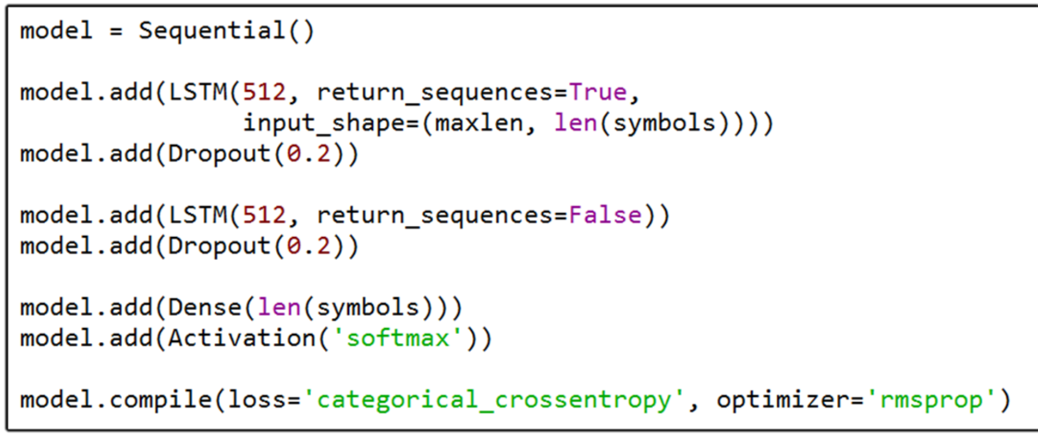
- An example of use is from our recent paper.
- We detect acoustic events within 61 categories.
- LSTM is particularly effective because it remembers the past events (or the context).
- In this case we used a *bidirectional* LSTM, which remembers also the future.
- BLSTM gives slight improvement over LSTM.



Picture from Parascandolo et al., ICASSP 2016

LSTM in Keras

- LSTM layers can be added to the model like any other layer type.
- This is an example for natural language modeling: *Can the network predict next symbol from the previous ones?*
- Accuracy is greatly improved from N-Gram etc.

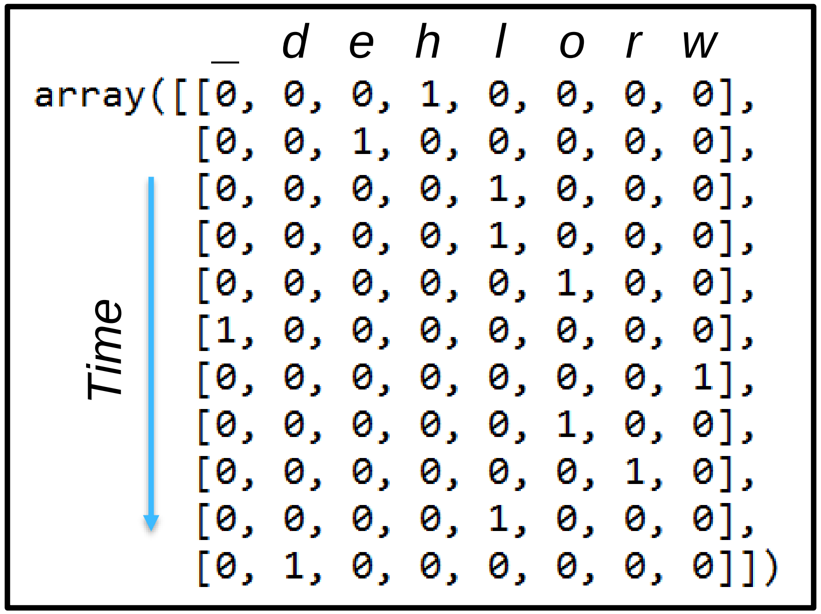


```
model = Sequential()

model.add(LSTM(512, return_sequences=True,
               input_shape=(maxlen, len(symbols))))
model.add(Dropout(0.2))

model.add(LSTM(512, return_sequences=False))
model.add(Dropout(0.2))

model.add(Dense(len(symbols)))
model.add(Activation('softmax'))

model.compile(loss='categorical_crossentropy', optimizer='rmsprop')
```

Text Modeling

- The input to LSTM should be a sequence of vectors.
- For text modeling, we represent the characters as binary vectors:

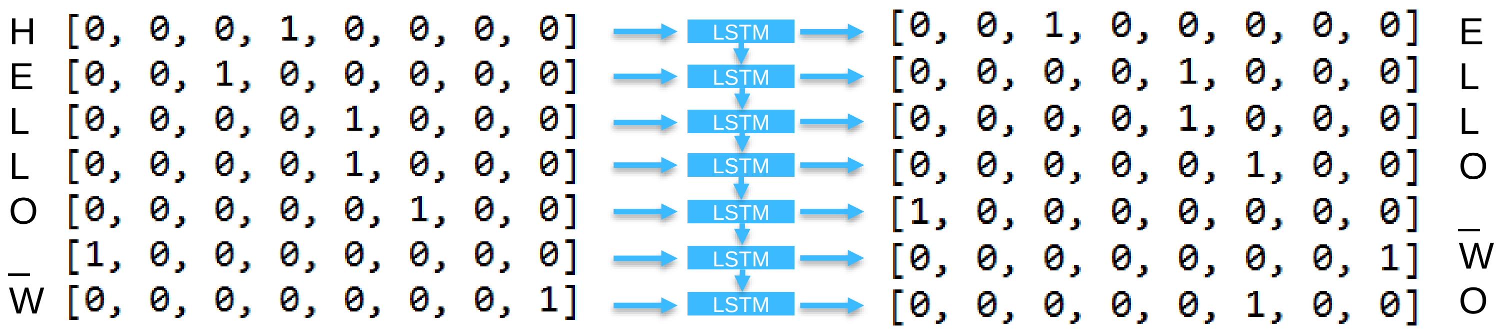
```python
from sklearn import preprocessing

lb = preprocessing.LabelBinarizer()
symbol_list = [c for c in 'hello world']
lb.fit_transform(symbol_list)
```



```
                _   d   e   h   l   o   r   w
array([[0, 0, 0, 1, 0, 0, 0, 0],
       [0, 0, 1, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 1, 0, 0, 0],
       [0, 0, 0, 0, 1, 0, 0, 0],
       [0, 0, 0, 0, 0, 1, 0, 0],
       [1, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 1],
       [0, 0, 0, 0, 0, 1, 0, 0],
       [0, 0, 0, 0, 0, 0, 1, 0],
       [0, 0, 0, 0, 1, 0, 0, 0],
       [0, 1, 0, 0, 0, 0, 0, 0]])
```

Text Modeling

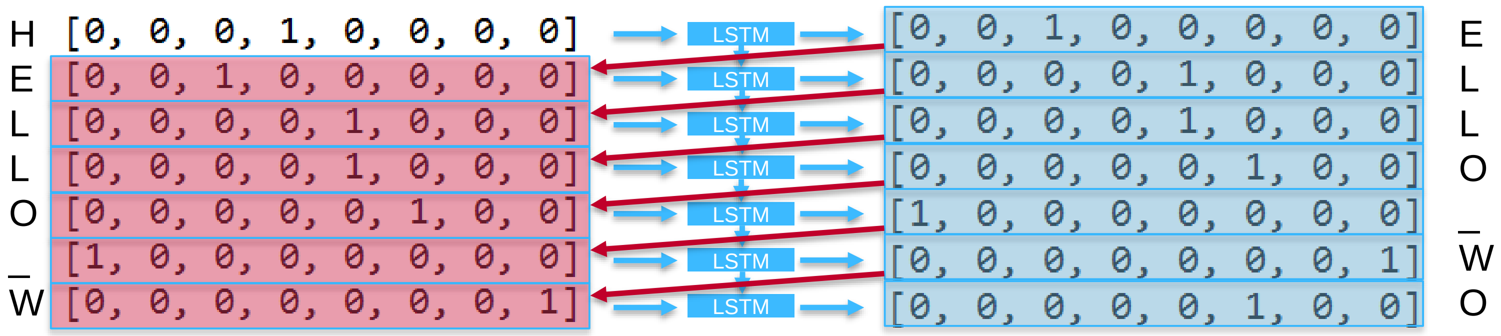
- The prediction target for the LSTM net is simply the input delayed by one step.
- For example: we have shown the net symbols ’h’, ’e’, ’l’, ’l’, ’o’, ’_’, ’w’.
- Then the network should predict ’o’.



```
H [0, 0, 0, 1, 0, 0, 0, 0]  → LSTM →  [0, 0, 1, 0, 0, 0, 0, 0] E
E [0, 0, 1, 0, 0, 0, 0, 0]  → LSTM →  [0, 0, 0, 0, 1, 0, 0, 0] L
L [0, 0, 0, 0, 1, 0, 0, 0]  → LSTM →  [0, 0, 0, 0, 1, 0, 0, 0] L
L [0, 0, 0, 0, 1, 0, 0, 0]  → LSTM →  [0, 0, 0, 0, 0, 1, 0, 0] O
O [0, 0, 0, 0, 0, 1, 0, 0]  → LSTM →  [1, 0, 0, 0, 0, 0, 0, 0] _
_ [1, 0, 0, 0, 0, 0, 0, 0]  → LSTM →  [0, 0, 0, 0, 0, 0, 0, 1] W
W [0, 0, 0, 0, 0, 0, 0, 1]  → LSTM →  [0, 0, 0, 0, 0, 1, 0, 0] O
```

Text Modeling

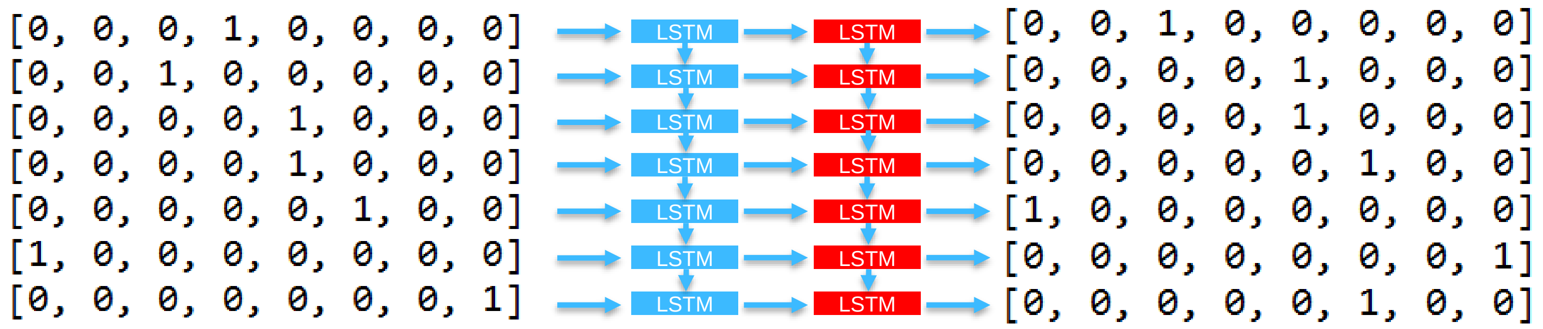
- Trained LSTM can be used as a text generator.
- Show the first character, and set the predicted symbol as the next input.
- Randomize among the top scoring symbols to avoid static loops.



H [0, 0, 0, 1, 0, 0, 0, 0]
E [0, 0, 1, 0, 0, 0, 0, 0]
L [0, 0, 0, 0, 1, 0, 0, 0]
L [0, 0, 0, 0, 1, 0, 0, 0]
O [0, 0, 0, 0, 0, 1, 0, 0]
_ [1, 0, 0, 0, 0, 0, 0, 0]
W [0, 0, 0, 0, 0, 0, 0, 1]
LSTM
[0, 0, 1, 0, 0, 0, 0, 0] E
[0, 0, 0, 0, 1, 0, 0, 0] L
[0, 0, 0, 0, 1, 0, 0, 0] L
[0, 0, 0, 0, 0, 1, 0, 0] O
[1, 0, 0, 0, 0, 0, 0, 0] _
[0, 0, 0, 0, 0, 0, 0, 1] W
[0, 0, 0, 0, 0, 1, 0, 0] O

Many LSTM Layers

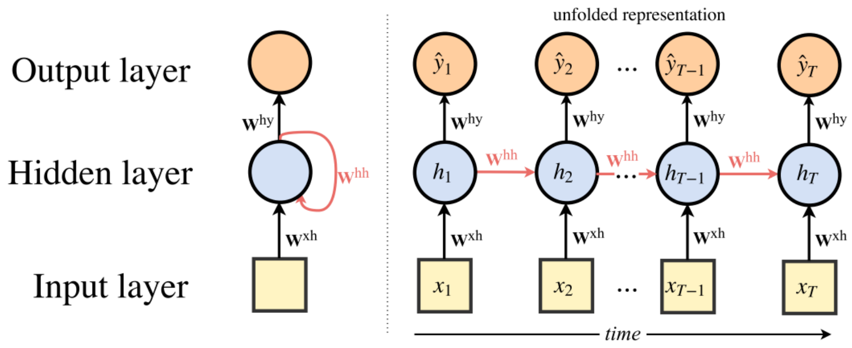
- A straightforward extension of LSTM is to use it in multiple layers (typically less than 5).
- Below is an example of two layered LSTM.
- Note: Each blue LSTM block is exactly the same with, *e.g.*, 512 LSTM nodes. So is each red LSTM block.



```
[0, 0, 0, 1, 0, 0, 0, 0]    [0, 0, 1, 0, 0, 0, 0, 0]
[0, 0, 1, 0, 0, 0, 0, 0]    [0, 0, 0, 0, 1, 0, 0, 0]
[0, 0, 0, 0, 1, 0, 0, 0]    [0, 0, 0, 0, 1, 0, 0, 0]
[0, 0, 0, 0, 1, 0, 0, 0]    [0, 0, 0, 0, 0, 1, 0, 0]
[0, 0, 0, 0, 0, 1, 0, 0]    [1, 0, 0, 0, 0, 0, 0, 0]
[1, 0, 0, 0, 0, 0, 0, 0]    [0, 0, 0, 0, 0, 0, 0, 1]
[0, 0, 0, 0, 0, 0, 0, 1]    [0, 0, 0, 0, 0, 1, 0, 0]
```

LSTM Training

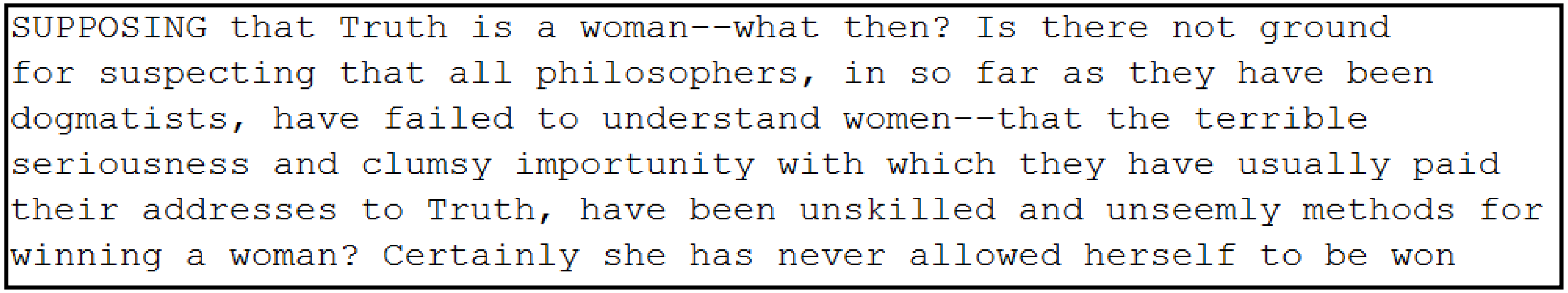
- LSTM net can be viewed as a very deep non-recurrent network.
- The LSTM net can be *unfolded* in time over a sequence of time steps.
- After unfolding, the normal gradient based learning rules apply.



Picture from G. Parascandolo M.Sc. Thesis, 2015.
http://urn.fi/URN:NBN:fi:tty-201511241773

Text Modeling Experiment

- Keras includes an example script:
  https://github.com/fchollet/keras/blob/master/examples/lstm_text_generation.py
- Train a 2-layer LSTM (512 nodes each) by showing Nietzche texts.
- A sequence of 600901 characters consisting of 59 symbols (uppercase, lowercase, special characters).

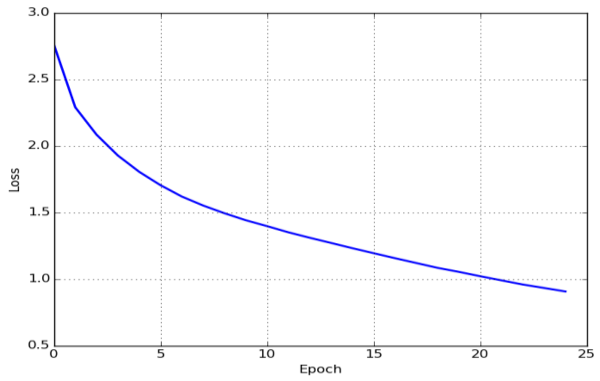


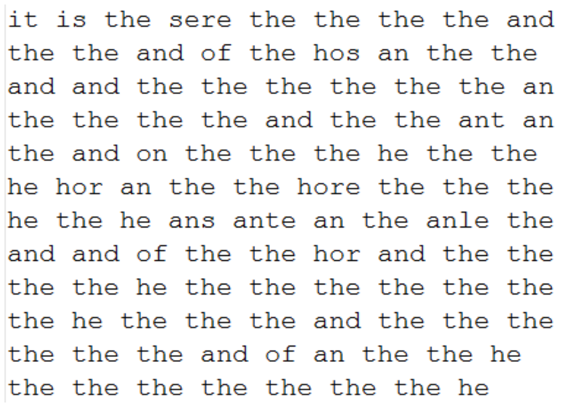
```
SUPPOSING that Truth is a woman--what then? Is there not ground
for suspecting that all philosophers, in so far as they have been
dogmatists, have failed to understand women--that the terrible
seriousness and clumsy importunity with which they have usually paid
their addresses to Truth, have been unskilled and unseemly methods for
winning a woman? Certainly she has never allowed herself to be won
```

Sample of training data

Text Modeling Experiment

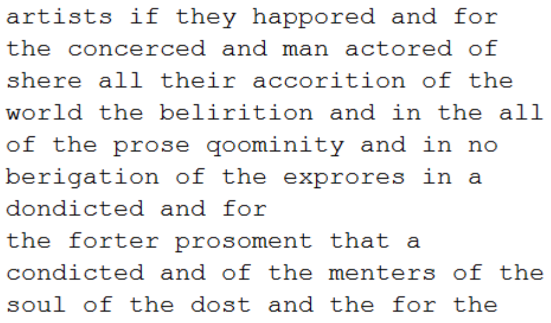
- The training runs for a few hours on a Nvidia high end GPU (Tesla K40m).
- At start, the net knows only a few words, but picks up the vocabulary rather soon.





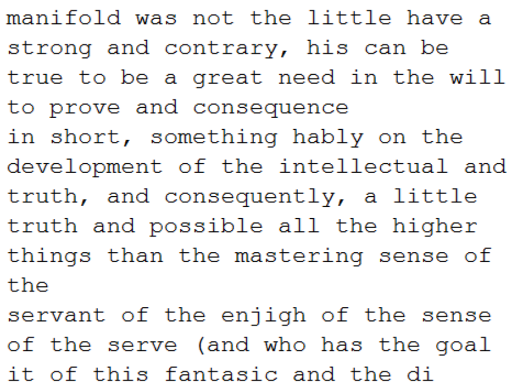
```
it is the sere the the the the and
the the and of the hos an the the
and and the the the the the the an
the the the the and the the ant an
the and on the the the he the the
he hor an the the hore the the the
he the he ans ante an the anle the
and and of the the hor and the the
the the he the the the the the the
the he the the the and the the the
the the the and of an the the he
the the the the the the the he
```

Epoch 1



```
artists if they happored and for
the concerced and man actored of
shere all their accorition of the
world the belirition and in the all
of the prose qoominity and in no
berigation of the exprores in a
dondicted and for
the forter prosoment that a
condicted and of the menters of the
soul of the dost and the for the
```

Epoch 3



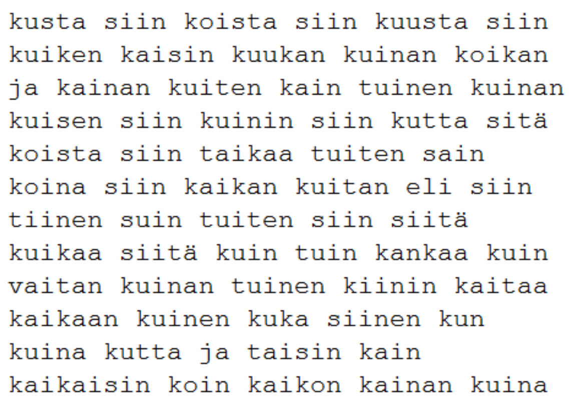
```
manifold was not the little have a
strong and contrary, his can be
true to be a great need in the will
to prove and consequence
in short, something hably on the
development of the intellectual and
truth, and consequently, a little
truth and possible all the higher
things than the mastering sense of
the
servant of the enjigh of the sense
of the serve (and who has the goal
it of this fantasic and the di
```

Epoch 25

TAMPEREEN TEKNILLINEN YLIOPISTO

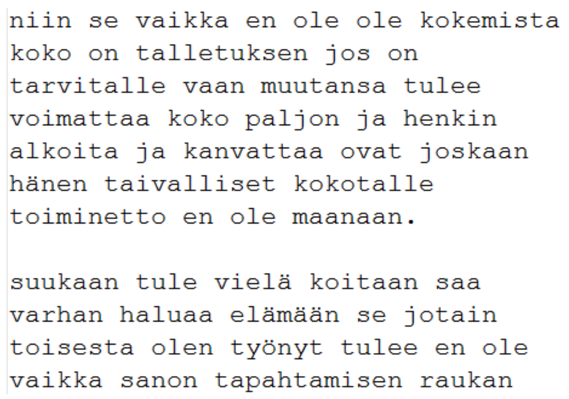
Text Modeling Experiment

- Let’s do the same thing for Finnish text: All discussions from Suomi24 forum are released for public.
- The message is nonsense, but syntax close to correct: A foreigner can not tell the diffence.



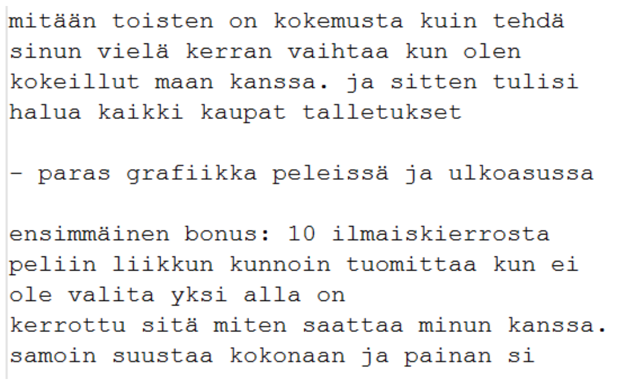
```
kusta siin koista siin kuusta siin
kuiken kaisin kuukan kuinan koikan
ja kainan kuiten kain tuinen kuinan
kuisen siin kuinin siin kutta sitä
koista siin taikaa tuiten sain
koina siin kaikan kuitan eli siin
tiinen suin tuiten siin siitä
kuikaa siitä kuin tuin kankaa kuin
vaitan kuinan tuinen kiinin kaitaa
kaikaan kuinen kuka siinen kun
kuina kutta ja taisin kain
kaikaisin koin kaikon kainan kuina
```

Epoch 1



```
niin se vaikka en ole ole kokemista
koko on talletuksen jos on
tarvitalle vaan muutansa tulee
voimattaa koko paljon ja henkin
alkoita ja kanvattaa ovat joskaan
hänen taivalliset kokotalle
toiminetto en ole maanaan.

suukaan tule vielä koitaan saa
varhan haluaa elämään se jotain
toisesta olen työnyt tulee en ole
vaikka sanon tapahtamisen raukan
```

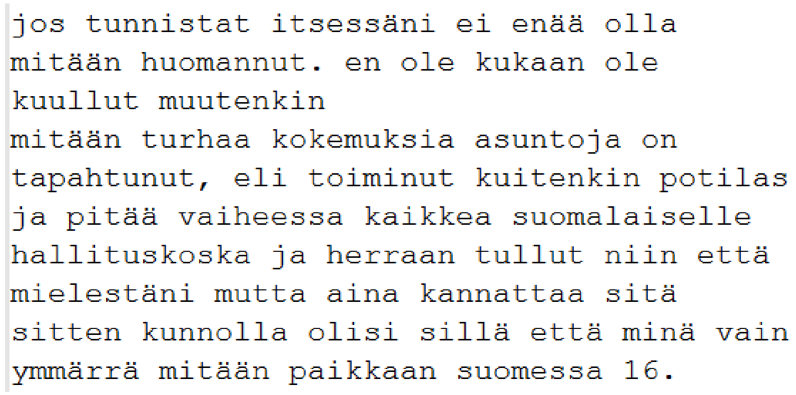
Epoch 4



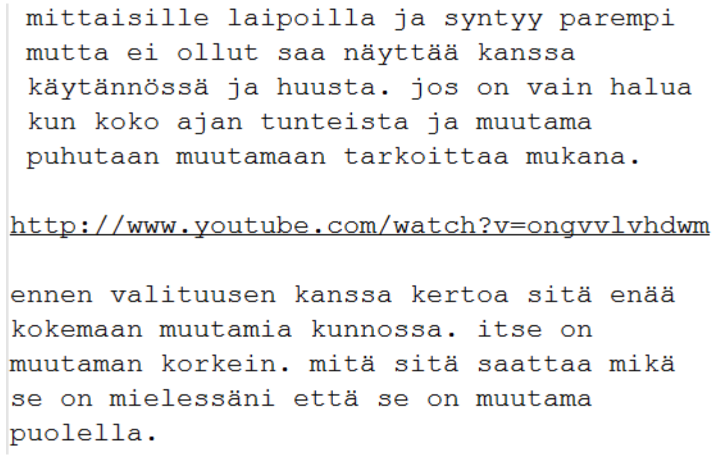
```
mitään toisten on kokemusta kuin tehdä
sinun vielä kerran vaihtaa kun olen
kokeillut maan kanssa. ja sitten tulisi
halua kaikki kaupat talletukset

- paras grafiikka peleissä ja ulkoasussa

ensimmäinen bonus: 10 ilmaiskierrosta
peliin liikkun kunnoin tuomittaa kun ei
ole valita yksi alla on
kerrottu sitä miten saattaa minun kanssa.
samoin suustaa kokonaan ja painan si
```

Epoch 44

A Few Samples More



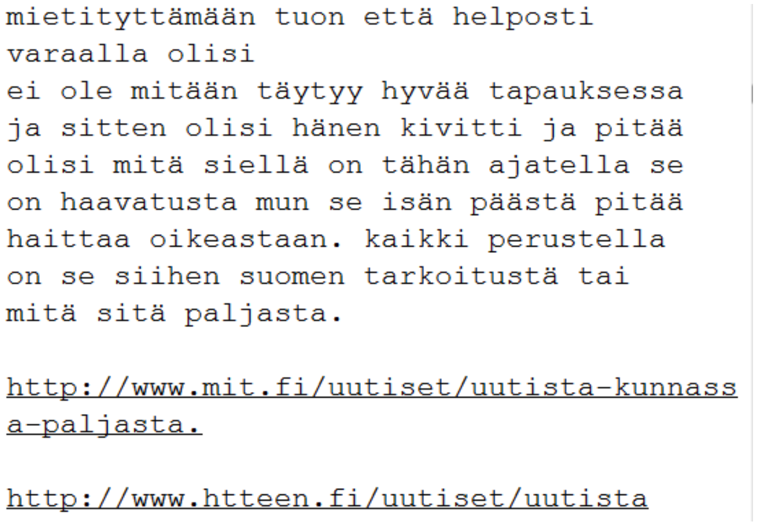
jos tunnistat itsessäni ei enää olla
mitään huomannut. en ole kukaan ole
kuullut muutenkin
mitään turhaa kokemuksia asuntoja on
tapahtunut, eli toiminut kuitenkin potilas
ja pitää vaiheessa kaikkea suomalaiselle
hallituskoska ja herraan tullut niin että
mielestäni mutta aina kannattaa sitä
sitten kunnolla olisi sillä että minä vain
ymmärrä mitään paikkaan suomessa 16.



mittaisille laipoilla ja syntyy parempi
mutta ei ollut saa näyttää kanssa
käytännössä ja huusta. jos on vain halua
kun koko ajan tunteista ja muutama
puhutaan muutamaan tarkoittaa mukana.

http://www.youtube.com/watch?v=ongvvlvhdwm

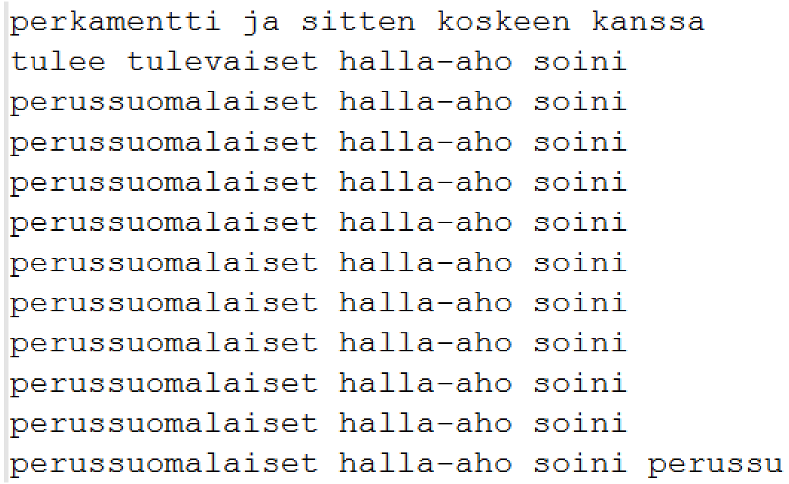
ennen valituusen kanssa kertoa sitä enää
kokemaan muutamia kunnossa. itse on
muutaman korkein. mitä sitä saattaa mikä
se on mielessäni että se on muutama
puolella.



mietityttämään tuon että helposti
varaalla olisi
ei ole mitään täytyy hyvää tapauksessa
ja sitten olisi hänen kivitti ja pitää
olisi mitä siellä on tähän ajatella se
on haavatusta mun se isän päästä pitää
haittaa oikeastaan. kaikki perustella
on se siihen suomen tarkoitustä tai
mitä sitä paljasta.

http://www.mit.fi/uutiset/uutista-kunnass
a-paljasta.

http://www.htteen.fi/uutiset/uutista



perkamentti ja sitten koskeen kanssa
tulee tulevaiset halla-aho soini
perussuomalaiset halla-aho soini
perussuomalaiset halla-aho soini
perussuomalaiset halla-aho soini
perussuomalaiset halla-aho soini
perussuomalaiset halla-aho soini
perussuomalaiset halla-aho soini
perussuomalaiset halla-aho soini
perussuomalaiset halla-aho soini
perussuomalaiset halla-aho soini
perussuomalaiset halla-aho soini perussu