

Project Survival 101

November 22, 2017 nuric@



What to do?

So many possibilities

- Give up and code in Haskell
- I have a macbook, I'm good
- SSH into every lab machine
- Say I didn't have enough time or resources

Have 20GB of data,
deep neural model,
need to pre-process
some images, big
data, machine
learning data deep
big blockchain
learning machine
mining

Source Code, github + gitlab

- Use github to keep a copy for yourself as part of your profile (if you want make it private with the student package)
- Use gitlab to integrate CI locally within DoC network
- Try to integrate a **linter** (ex pylint if in Python) to keep things tidy early, it will get messy
- **Automate** repetitive work done after writing code using Gitlab Runners

 passed	#37899 by 	 master  3834336d  Made application closing unif...		 00:02:41  3 months ago
 passed	#37823 by 	 master  91460cf9  Normalised default context se...		 00:05:02  3 months ago

Gitlab Continuous Integration (CI)

Local gitlab integrates well with other infrastructure we have. Use gitlab CI to **automate repetitive tasks**, doesn't have to be just unit tests.

You use **runners** to actually execute the CI pipeline, we will use the DoC IaaS (cloudstack) to create them.

```
# .gitlab-ci.yml
```

```
test:
```

```
  script:
```

```
    - make test
```

BIG DATA

Things to consider carefully:

- **Size** of data, is it BIG? Videos are the worst.
- **Interaction**, is it read or write heavy, for example external library vs data you generate
- **Origin**, is it public, bound to university, medical data is often restricted

Mainly have 5 options:

- **Home dir** (~/), good for code, smaller data you generate, has quota limit but can be extended if all other options fail
- **Bitbucket** (/vol/bitbucket/\$USER), good for large read-only data, can be slow
- **Local** (/data/\$USER) local SSD disk, high performance, limited to only one machine
- **Postgres** (db.doc.ic.ac.uk), for structured sql data, good for filtering querying etc
- **Onedrive** (1TB) good for personal backup

BIG DATA

Housekeeping:

- **zip** old courseworks and backup on some personal cloud account.
- **~/Downloads**, always has stale stuff, like specs from first year.
- **~/cache**, is mostly filled with browser stuff, change their settings

For coding stuff:

- **git clone --depth=1** for things that you are not going to develop. Just need HEAD.
- **git gc** for garbage collecting stale objects etc. Can go **--aggressive**
- **chmod 644 or 755** on either home or bitbucket files / folders for duplicate data across projects. Then someone else can read that data from your folder share. Remember these are network shares already.

Parallel Processing

Look for multithreading opportunities! High level languages make it super easy, some ProcessPoolExecutor call is all you need.

Why don't we use all the threads? Most lab machines have 4 cores with 2 threads giving 8 executions, but **we don't want to block the machine.**

```
from time import sleep
from concurrent.futures import ProcessPoolExecutor
from multiprocessing import cpu_count

THREADS = cpu_count()-1

data = range(7)

def nasty(x):
    sleep(5)
    return 2*x-1

results = list()
with ProcessPoolExecutor(max_workers=cpu_count()-1) as executor:
    results = executor.map(nasty, data)

# Iterator will block
for r in results:
    print(r)
```

Long running jobs (this machine is so slow)

The goal is to be minimally disruptive as possible. As if you are not even there...

What **NOT** to do:

- SSH into a machine (or even worse voxel\$!) and just start running 800% cpu load
- Name script as “**donotkill_finishesin30min.py**” and then run another one every 30 minutes

But we don't want to waste idle resources either! If your thing is **urgent**, just take note:

- Beware of tmux, you can leave things running around
- Check if someone is already running something
- Do not use the entire machine resources, 1 less thread might allow a first year to read their Haskell spec while you crunch numbers in the background

HTCondor

I have 100 videos to process,
why not get 100 machines?

HTCondor is a specialized workload management system for compute-intensive jobs. Like other full-featured batch systems, HTCondor provides a **job queueing mechanism, scheduling policy, priority scheme, resource monitoring, and resource management**. Users submit their serial or parallel jobs to HTCondor, HTCondor places them into a queue, **chooses when and where to run the jobs** based upon a policy, carefully monitors their progress, and ultimately informs the user upon completion.

<https://research.cs.wisc.edu/htcondor/>

Getting started with Condor

By default the commands are not in \$PATH, put it in bashrc or your shell config

```
export  
PATH=$PATH:/usr/local/condor/release/bin
```

<https://www.doc.ic.ac.uk/condor/>

Mainly interested in 2 commands:

- **condor_submit** [file], actually submits a job to condor using *job description file*
- **condor_q** monitors your active jobs

Note that all machines are connected to DoC so your home folder + bitbucket is available. Often, if it runs on a standard lab machine, it will run anywhere, except for GPU and other niche stuff.

Job description file

```
# uname on some machines
universe = vanilla
executable = /bin/uname
arguments = -n -m -p
output = uname.$(Process).out
log = uname.log
queue 5
```

```
# run command for every file
universe = vanilla
notification = Complete # notifies after every job!
notify_user = <your-email>
executable = /usr/bin/wc
arguments = $(item)
output = $(item).out
log = file.log
queue matching files data/*.txt
```

```
executable = foo
universe = vanilla
queue arguments from (
  15 2000
  30 2000
)
```

```
# request specific resources
request_GPUs = 1
request_memory = ...
```

You can also queue single long running jobs which Condor will handle without annoying others!

A gentle touch SIGTERM

When someone logs onto the machine you are running Condor jobs on, what happens?

Condor tries to gracefully terminate your job and move it another available machine automatically!

All you have to do is gracefully terminate your job, you may want to save some checkpoint or state to recover from.

```
"""Example raising KeyboardInterrupt"""  
import signal  
import time  
  
def f(signum, frame):  
    print("Getting interrupted")  
    raise KeyboardInterrupt  
    signal.signal(signal.SIGTERM, f)  
  
try:  
    for i in range(100): # some long function  
        print(i)  
        time.sleep(i)  
except KeyboardInterrupt:  
    print("Terminating.") # Handle interrupt somehow
```

External Resources

There are other resources outside of DoC:

- **High Performance Computing (HPC):**
Good for large simulations that can be parallelized, gives you large memory and hundreds of cores. Requires access via supervisor.
- **Research Computing Service (RCS):**
Dedicated college wide group for computing stuff. Most services oriented around HPC.

More useful considerations include student packages offered by cloud providers:

- **Microsoft Azure:** Get free access using Dreamspark (search Imperial College Dreamspark), again good for servers etc.
- **Amazon Web Services (AWS):** Students get some credit to run servers (EC2) and store data (S3).

Feedback + Questions?

www.doc.ic.ac.uk/~nuric/fb

Office hours -> Tuesday mornings (this term) -> 558C

Project Survival 102

November 30, 2017 nuric@



Deep learning

TensorFlow

Lower level “tensor” manipulation library with some high level API. Use directly if working on deep learning architectures or bulk data processing. If not, it is a little messy and non-intuitive despite Python API (it has package globals, runner sessions etc)

Runs on CPU and GPGPU for faster processing. But it would depend on your model, ex RNNs might run faster on CPU.

Keras

Actually a deep learning library built on Tensorflow or Theano (you can pick). Designed to be intuitive and hides all scaffolding code needed. If your data shapes match, it is plug and play using Numpy data similar to scikit-learn.

It still gives access to TensorFlow to build custom functions but not as flexible as using the actual library.

But what is a tensor?

```
3 # a rank 0 tensor; a scalar with shape []
```

```
[1., 2., 3.] # a rank 1 tensor; a vector with shape [3]
```

```
[[1., 2., 3.], [4., 5., 6.]] # a rank 2 tensor; a matrix with shape [2, 3]
```

```
[[[1., 2., 3.]], [[7., 8., 9.]]] # a rank 3 tensor with shape [2, 1, 3]
```

Think **n-dimensional array**, what it represents depends on the application

```
import tensorflow as tf
```

```
# Model parameters
```

```
W = tf.Variable([.3], dtype=tf.float32)
```

```
b = tf.Variable([-0.3], dtype=tf.float32)
```

```
# Model input and output
```

```
x = tf.placeholder(tf.float32)
```

```
linear_model = W*x + b
```

```
y = tf.placeholder(tf.float32)
```

```
# loss
```

```
loss = tf.reduce_sum(tf.square(linear_model - y)) # sum of the squares
```

```
# optimizer
```

```
optimizer = tf.train.GradientDescentOptimizer(0.01)
```

```
train = optimizer.minimize(loss)
```

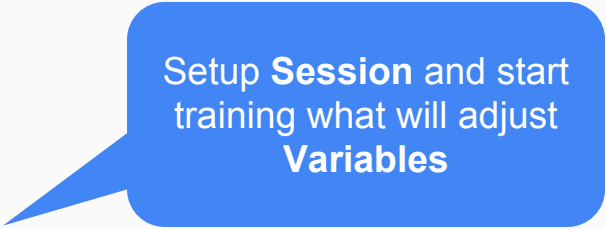
Variables are trainable through automatic differentiation.

Placeholders are your inputs and outputs, the values you set

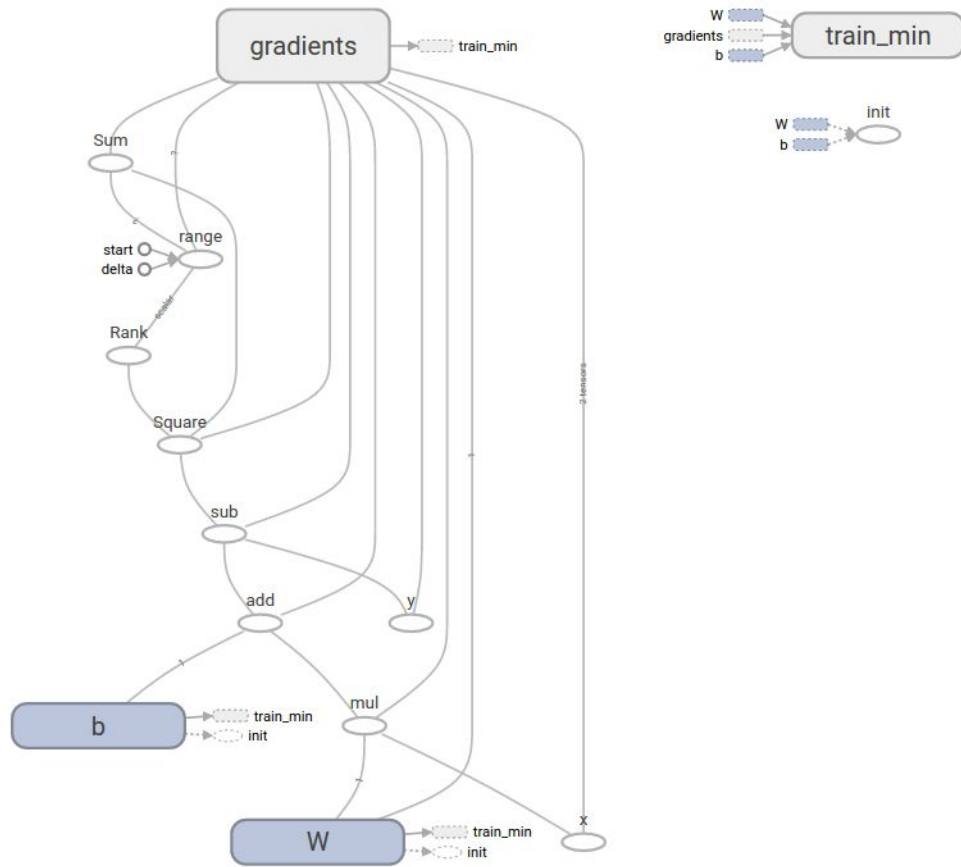
```
# training data
x_train = [1, 2, 3, 4]
y_train = [0, -1, -2, -3]

# training loop
init = tf.global_variables_initializer()
sess = tf.Session()
sess.run(init) # reset values to wrong
for i in range(1000):
    sess.run(train, {x: x_train, y: y_train})

# evaluate training accuracy
curr_W, curr_b, curr_loss = sess.run([W, b, loss], {x: x_train, y: y_train})
print("W: %s b: %s loss: %s"%(curr_W, curr_b, curr_loss))
```



Setup **Session** and start training what will adjust **Variables**



Training $y = w \cdot x + b$ with TensorFlow, the execution graph

Keras

Building complex networks

Well that's enough of TensorFlow.
Using the good bits we are interested in larger, more complex neural networks. Keras allows us to create **modular** networks with a **cleaner API**.

```
import numpy as np
from keras.models import Sequential
from keras.layers import Dense
```

Dense layer implements
 $Wx + b$

```
model = Sequential()
model.add(Dense(1, activation='linear', input_dim=1))
```

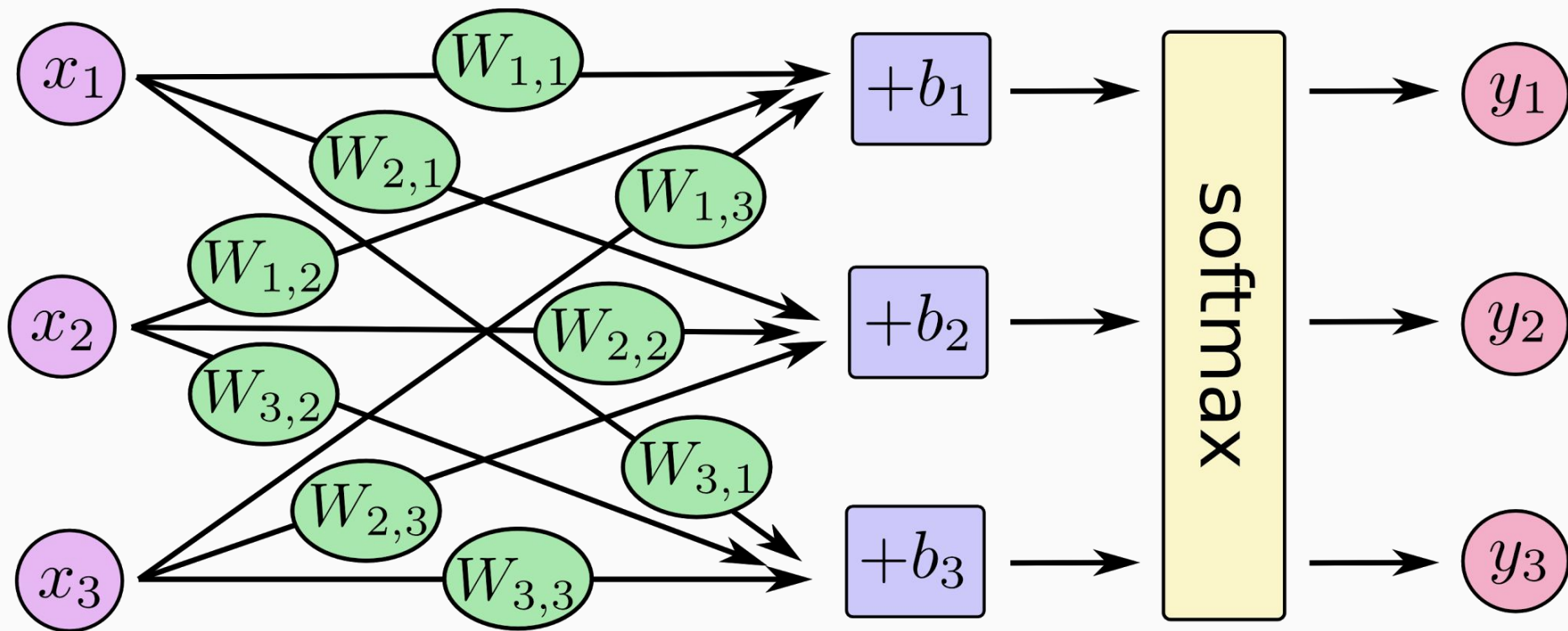
```
model.compile(loss='mse',
              optimizer='sgd',
              metrics=['accuracy'])
```

Compile just sets what
loss and training we want

```
x_train = np.array([1, 2, 3, 4])
y_train = np.array([0, -1, -2, -3])
```

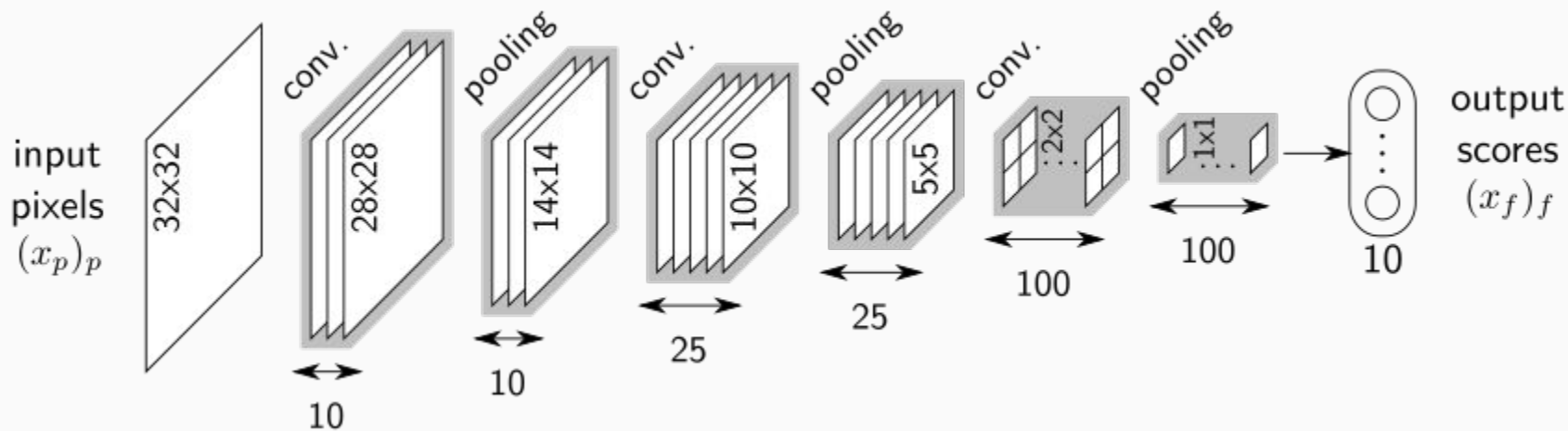
```
model.fit(x_train, y_train, epochs=5, batch_size=1)
```

```
print(model.predict(np.array([5, 6])))
```

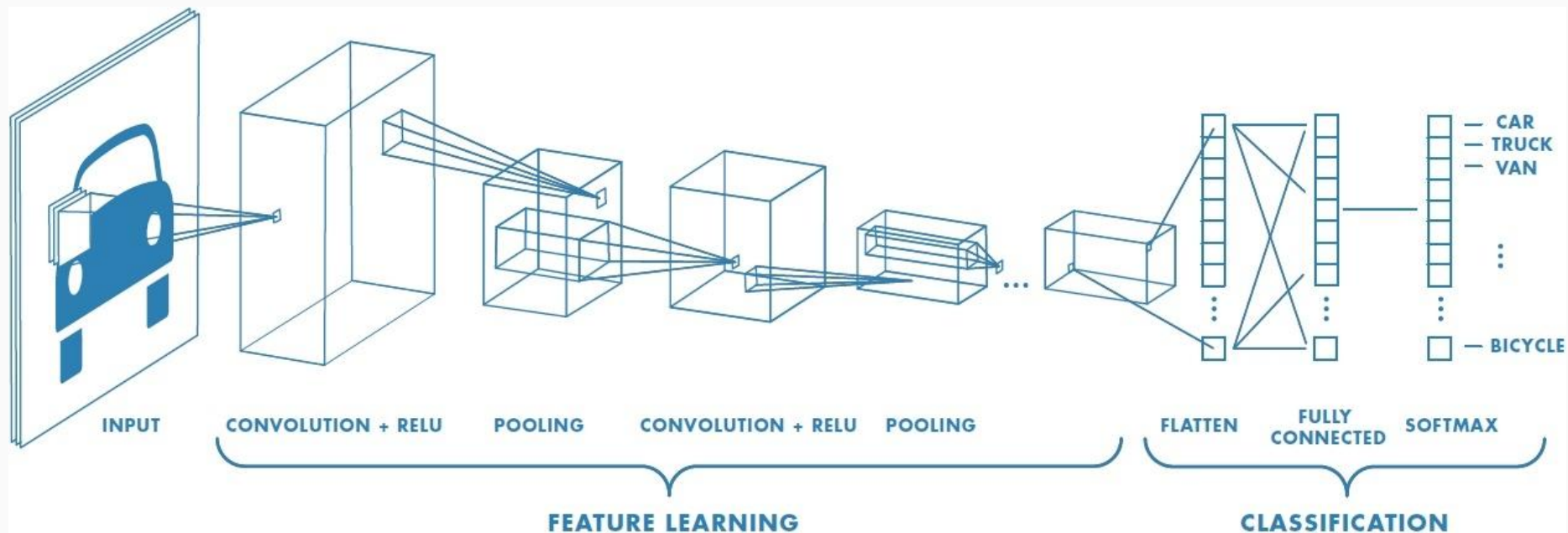


Representation of a Dense layer.

Convolutional Neural Networks (CNNs)



Convolutional Neural Networks (CNNs)

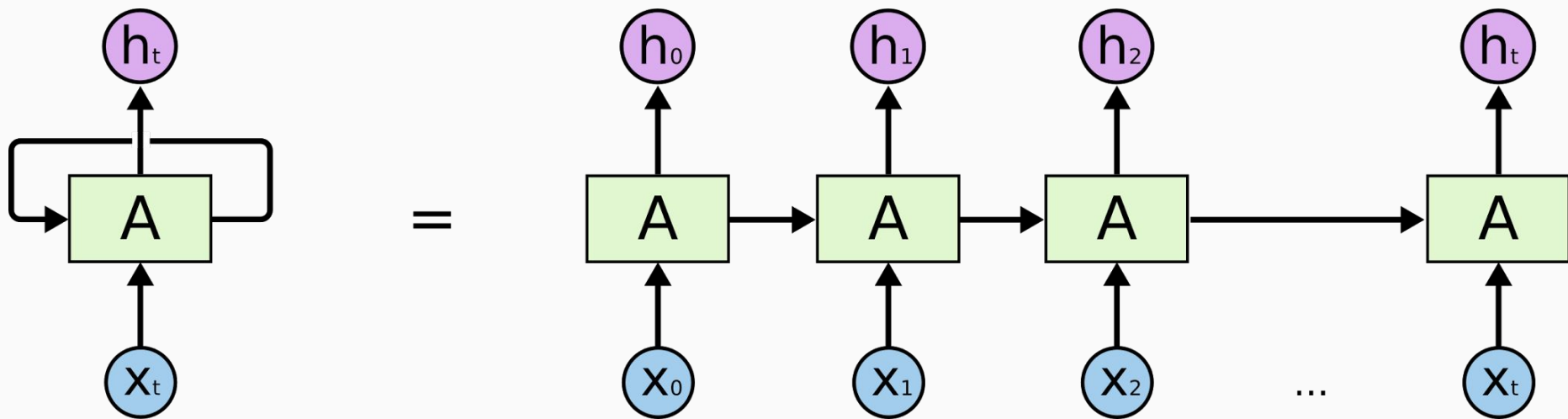


```
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),
                activation='relu',
                input_shape=input_shape))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))
```

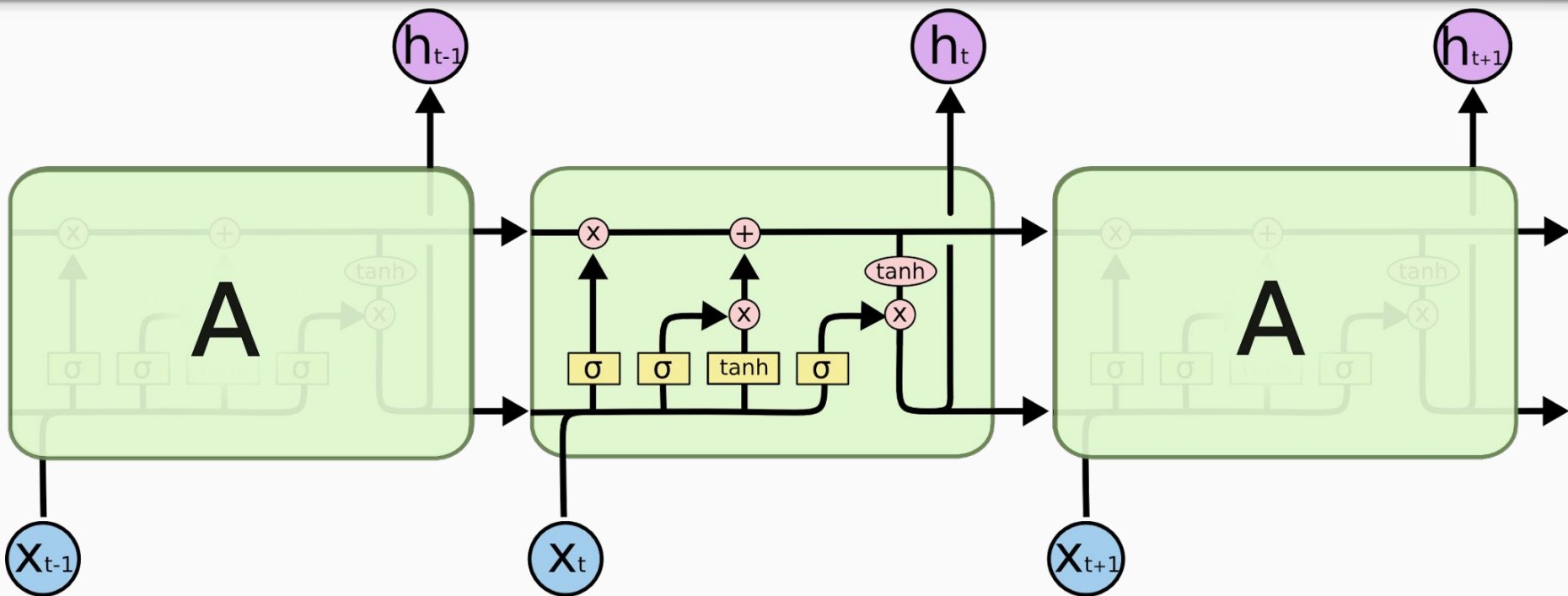
Convolution layers + Max
Pooling

Final dense layer for
 $P(X=x)$ output using
sigmoid function

Recursive Neural Networks (RNNs)



Long Short-Term Memory (LSTM)



```
model = Sequential()  
model.add(LSTM(128, input_shape=(maxlen, len(chars))))  
model.add(Dense(len(chars)))  
model.add(Activation('softmax'))
```

LSTM layer applies the same LSTM unit over a time series (timesteps)

Surely it's not that simple? Well there is always preparing the data into the format the neural network expects...

End-To-End Memory Network for bAbI Tasks

Story

daniel went to the office
john moved to the garden
john went back to the kitchen
daniel moved to the garden
mary went to the kitchen
daniel went to the bedroom
john went back to the hallway
sandra travelled to the garden
sandra travelled to the bedroom
daniel moved to the kitchen

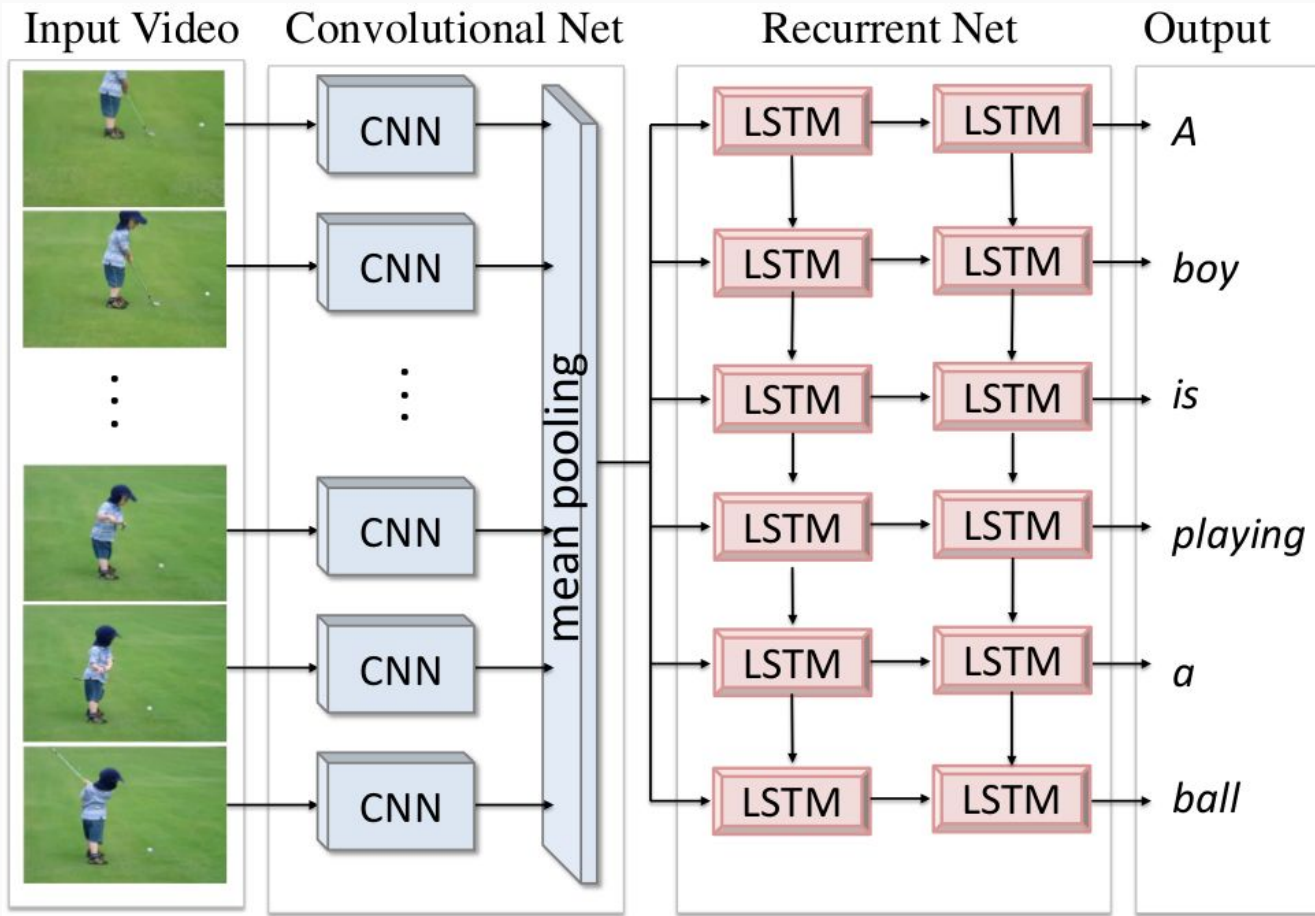
Question ⓘ

is sandra in the bedroom?

Answer

Predict answer Get new story

Text Mem 1 Mem 2 Mem 3



Not so fast

- You need good **data** to train these models. Having 2 sets of MRI images will not give a magic results on cancer diagnosis.
- Getting the data **ready** is often more work than building the network. Ex, vectorising inputs, loading images etc.
- Easy to build large network but much **harder** to train. Don't go crazy with extra layers to create a deeper network.
- Expect failure and frustration more often than not...

Tips

- Run your network with random weights, **do you get random outputs?** Or is it all 0s, 1s, check that.
- **Try to overfit a single point.** Does your network actually learn? If it cannot on a single data point something is wrong.
- Use **fit_generator** for large data that is streamed into the network like videos.
- A **GPU might be slower** for your network architecture test before crying for GPUs.

I need root to install packages -> no

Check if package already exists, if not install in **bitbucket**, recommend using *virtualenv* to isolate packages:

```
export  
PYTHONUSERBASE=/vol/bitbucket/$USER/pypi/
```

```
export  
PATH=$PATH:/vol/bitbucket/$USER/pypi/bin
```

```
pip3 install --user tensorflow[-gpu] keras  
...
```

- You can be clever and share installations by sharing bitbucket or home folders if you have similar requirements.
- Most cases just require you to install somewhere you have access and then tell the program where to find it.
- If you are uncertain, double check with CSG before giving up.

Monitoring

Take a tell don't ask approach

- Code got stuck? Your loss function is NaN? Why isn't this working after X epochs?
- Don't be a victim, look after your jobs, or let your jobs alert you when it is important
- Check with commands like `top`, `who`, to see what is going on before you leave it overnight

Easy monitoring

Here is a one liner if you are using a script to send an email:

```
blah.sh && mail -s "Jobs done"  
$USER@imperial.ac.uk <<< "Foo  
finished running, wake up!"
```

Make sure it doesn't turn into a spam machine though, instead there are better alternatives.

Use a webhook into a service like Slack or Discord. It is often as simple as a post request.

```
keras.callbacks.RemoteMonitor(root= 'discord_url',  
path= '', field= 'data')
```



Mini Kevin BOT

```
{"acc": 0.15353750004339964, "loss": 4.144216682434082, "epoch": 50}
```

Feedback + Questions?

www.doc.ic.ac.uk/~nuric/fb

Office hours -> Tuesday mornings (this term) -> 558C