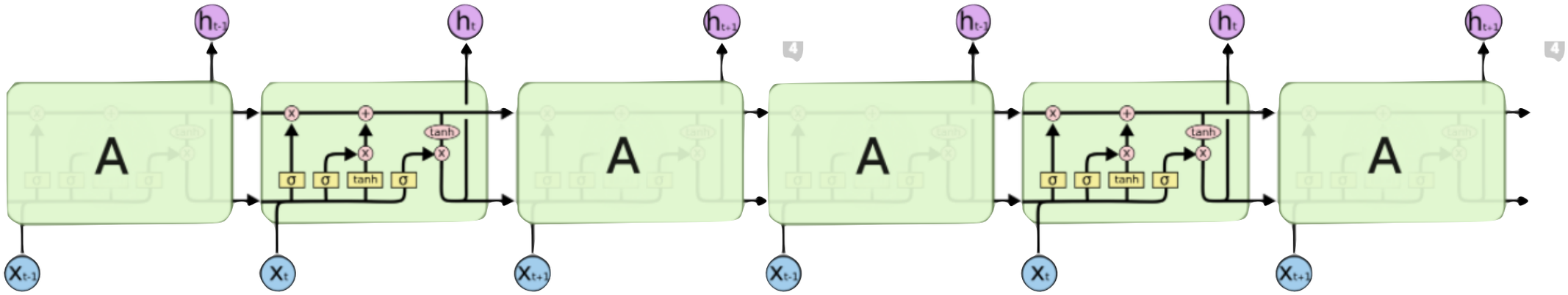
Text Sentiment Analysis with rNN on the IMDB Dataset

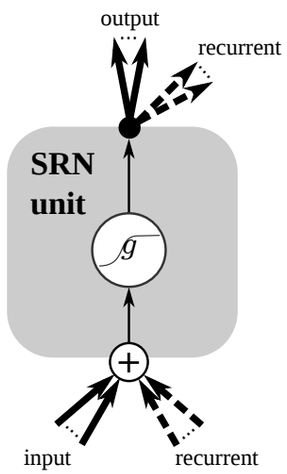
PyTorch and TensorFlow Comparative Evaluation

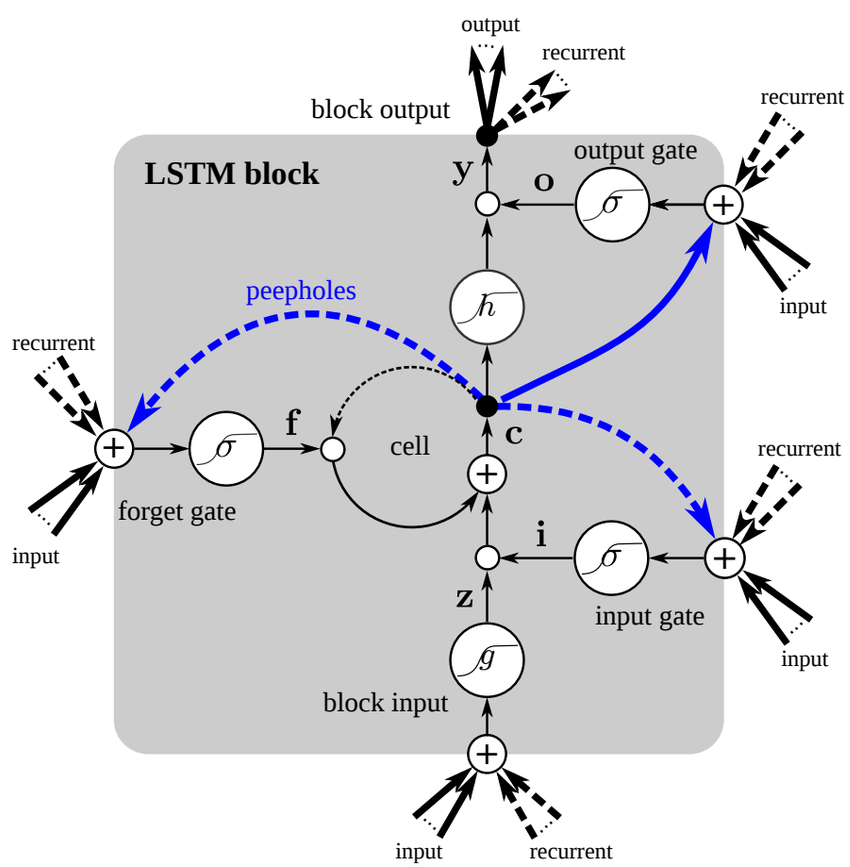
Stefanos Laskaridis
sl829@cam.ac.uk

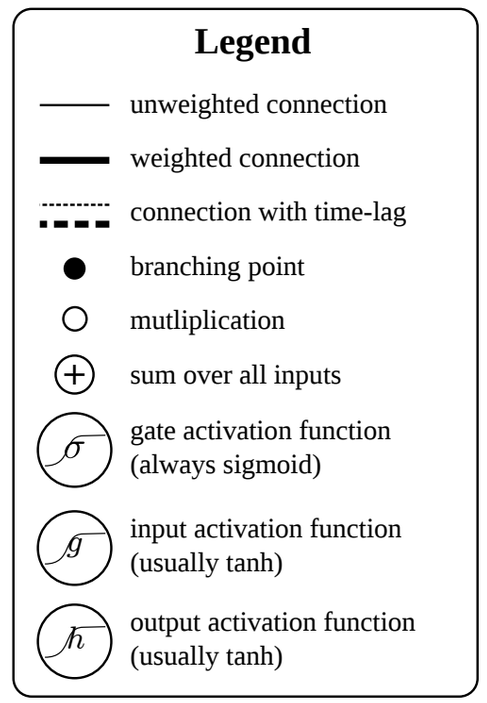
R244: Large-Scale Data Processing and Optimisation

Recurrent Neural Networks





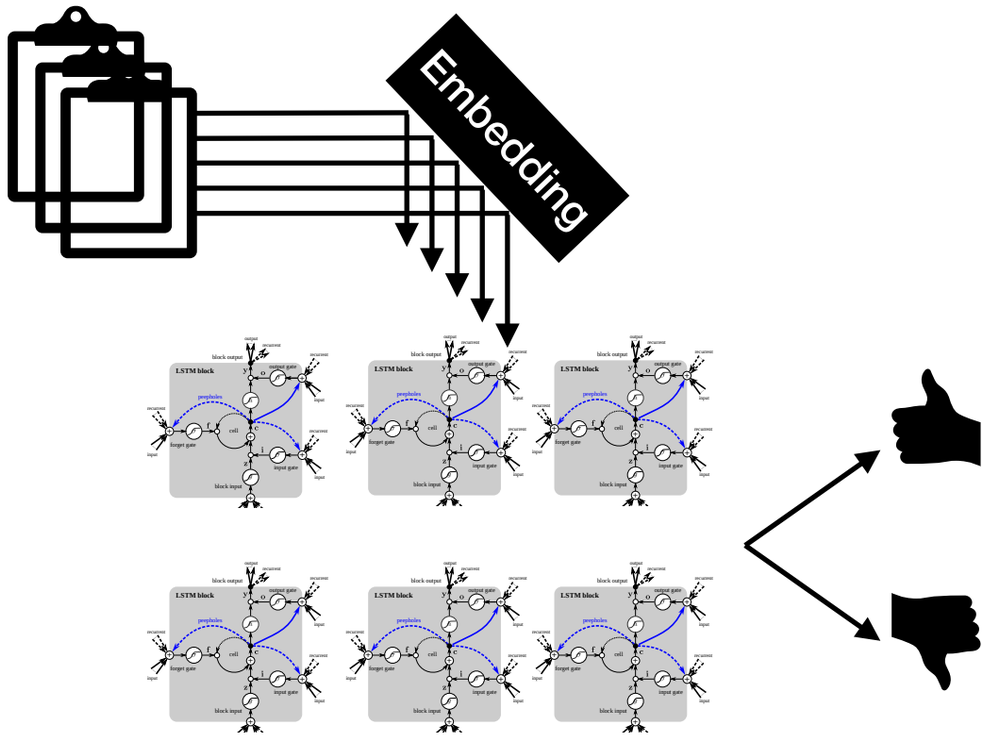




Images taken from [5] and [6]

Sentiment Analysis

- Make out the general sentiment of a sentence.
- Based word vectors, ngrams, word embeddings, etc.
- Binary or Multiclass classification in different sentiments.



The IMDB dataset

```
from keras.datasets import imdb

(X_train, y_train), (X_test, y_test) = imdb.load_data()

X = np.concatenate((X_train, X_test), axis=0)
y = np.concatenate((y_train, y_test), axis=0)

print('X shape:', X.shape)
print('y shape:', y.shape)

# Number of classes for classification
print('Dictionary size:', len(np.unique(y)))

# Number of unique words in dataset (dictionary)
print(len(np.unique(np.hstack(X))))

# Avg review length
lengths = [len(r) for r in X]
avg = np.mean(lengths)
stdev = np.std(lengths)
print("length mean:", avg)
print("length stdev", stdev)
```

```
X shape: (50000,)
y shape: (50000,)
Dictionary size: 2
88585
length mean: 234.75892
length stdev 172.911494587
```

PyTorch vs. TensorFlow

- Python implementation of Torch (Lua)
- Imperative programming model
- Great integration with Python
- Beta version
- CPU, GPU
- Facebook, CMU, Stanford, NYU, ParisTech, ENS, …

- Multiple frontends: C++, Python, Java, Go
- Declarative API
- Imperative API through Eager [3]
- Data Flow graphs, with partial graph execution
- CPU, GPU, TPU, Mobile (TensorFlow Lite)
- Google, AirBnb, Uber, SAP, ebay, Intel

Images taken from [1] and [2]

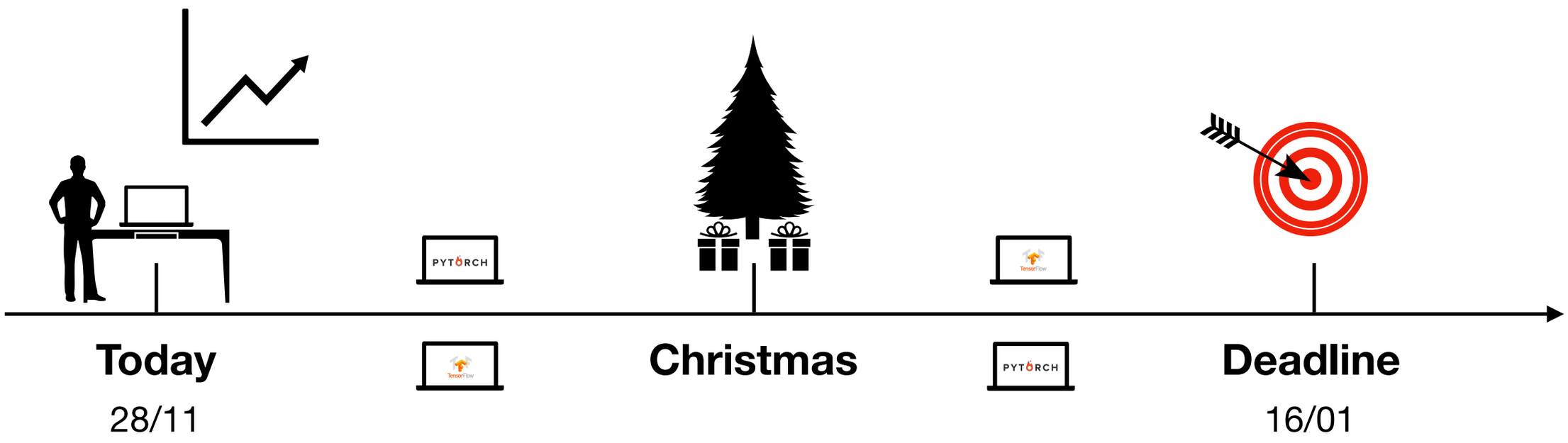
Significance

- Research:
  - Prominent Question: Have you tried using DNN?
- Programming paradigm is shifting
  - Induction, data-centric approach
- TensorFlow and PyTorch are among the primary tools used by the industry and academia
- RNNs are great for data with temporal relations (e.g. text, speech)

Exploration 

- Evaluate how the top 2 Deep Learning Frameworks perform in CPU-only computations
  - Lack of available Nvidia GPU :(
  - and a Google TPU :(
  - Maybe test on Public Cloud - Amazon Spot Instances?
- Can you do “Deep” learning on CPUs?
  - Explore the limits on a “commodity” laptop
  - How far can “fast-prototyping” go?

Working Plan



- So far:
  - Tools installations and playground setup
  - Dataset exploration
  - API familiarisation

- Deliverables:
  - Different RNN depths and architectures comparative benchmarks
  - Accuracy benchmarks
  - Computability benchmarks
  - Results interpretation

Thank you

Q&A

Stefanos Laskaridis
sl829@cam.ac.uk

References

1. PyTorch
   http://pytorch.org/

2. TensorFlow
   https://www.tensorflow.org

3. TensorFlow Eager
   https://research.googleblog.com/2017/10/eager-execution-imperative-define-by.html

4. IMDB Dataset
   http://ai.stanford.edu/~amaas/papers/wvSent_acl2011.pdf
   http://ai.stanford.edu/~amaas/data/sentiment/
   https://s3.amazonaws.com/text-datasets/imdb.npz

5. http://colah.github.io/posts/2015-08-Understanding-LSTMs/

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