NATURAL LANGUAGE PROCESSING

Beller Stefan, Binder Lukas, Karasinski Maciej Jacek, Kasper Bianca, Wichser Ilona

Outline

- Dataset
- LSTM
- Dropout and LayerNormalization
- GPUs
- Models 1 - 9
- Final network
- Tracking of experiments

IMDB dataset for sentiment analysis

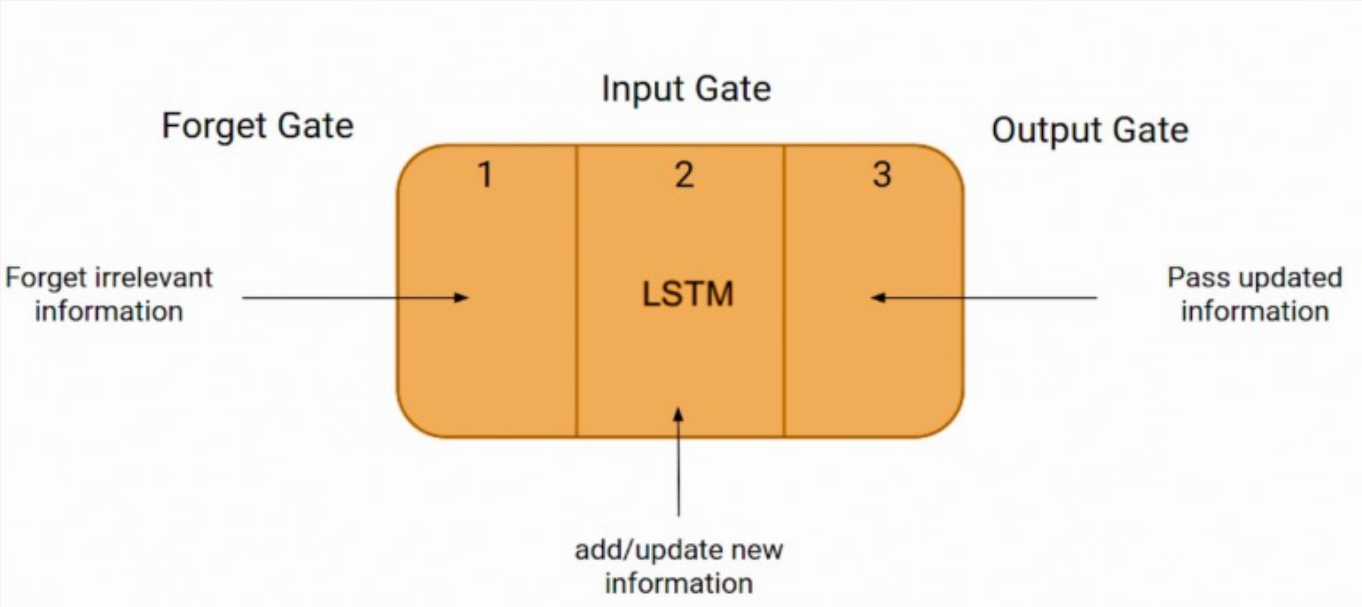
- Contains 50,000 movie reviews
- “good” and “negative” sentiment class labels

- Vocabulary size for our model: 5000 words

LSTM

- Used for all our models
- Special kind of RNN: capable of learning long-term dependencies
- Forgets, remembers and updates information

- 3 main gates



Why LSTM?

- Explicity designed to avoid long-term dependency problems
- Useful for sentiment analysis of longer/ multiple sentences

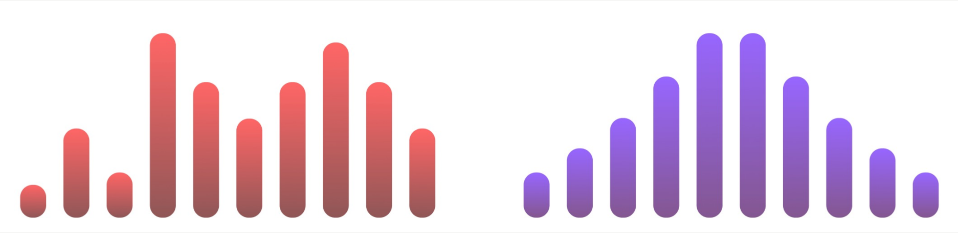
Parameters:

- units: Dimensionality of output space
- return_sequences=True
  - *Enables output of previous LSTM layer to be used as an input to next LSTM layer*

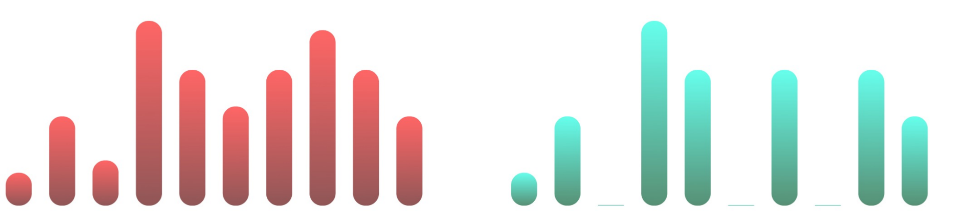
Dropout vs. LayerNormalization

- Both to avoid overfitting

- **LayerNormalization:** Normalize the activations of the previous layer for each given example in a batch independently

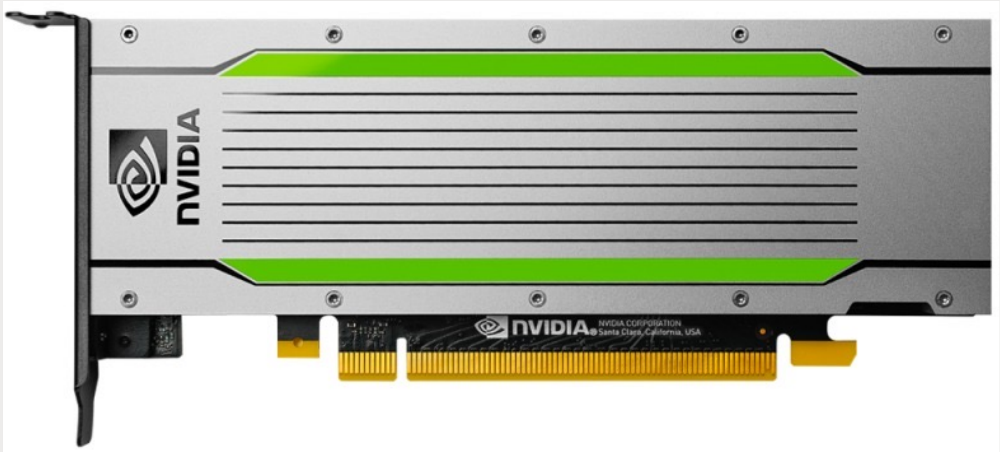


- **Dropout:** Randomly sets input units to 0

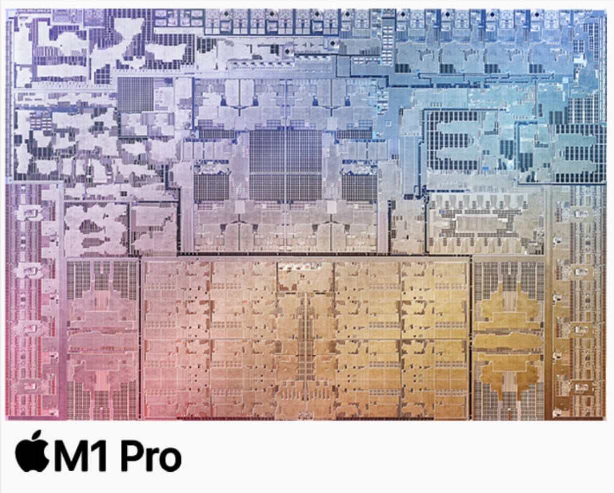


GPUs

- **Google Colab:** Tesla T4



- **Mac:** M1 Pro



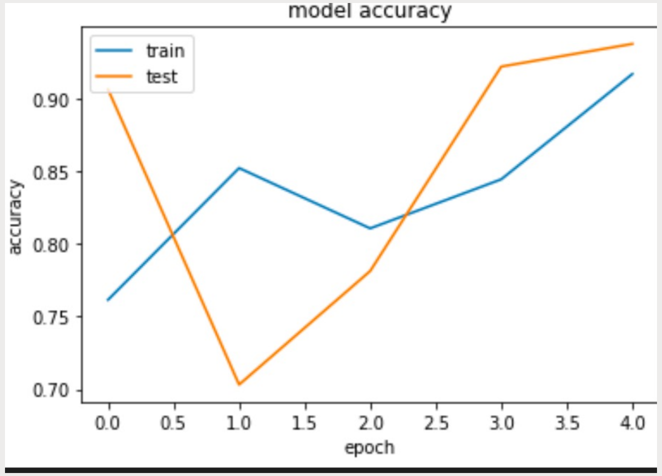
First networks

- Original: 1x model.add(SimpleRNN(100))

Model 1:

- 1x model.add(LSTM(100, return_sequences=True))

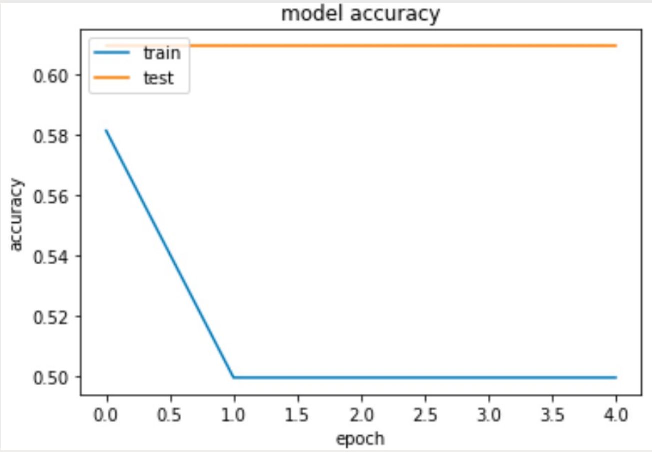
  → ~90% val_accuracy, ~50% test_accuracy



Model 2:

- 2x model.add(LSTM(100, return_sequences=True))

  → ~62% val_accuracy, 50% train_accuracy



Model 3

6m/epoch

embeddingSize = 64

- model.add(LSTM(64, return_sequences=True))
- model.add(Dropout(0.3))
- model.add(LSTM(32, return_sequences=True))
- model.add(Dropout(0.3))
- model.add(LSTM(16, return_sequences=True))
- model.add(Dropout(0.3))
- model.add(LSTM(8, return_sequences=False))
- model.add(LayerNormalization())

- Test accuracy: 0.5009

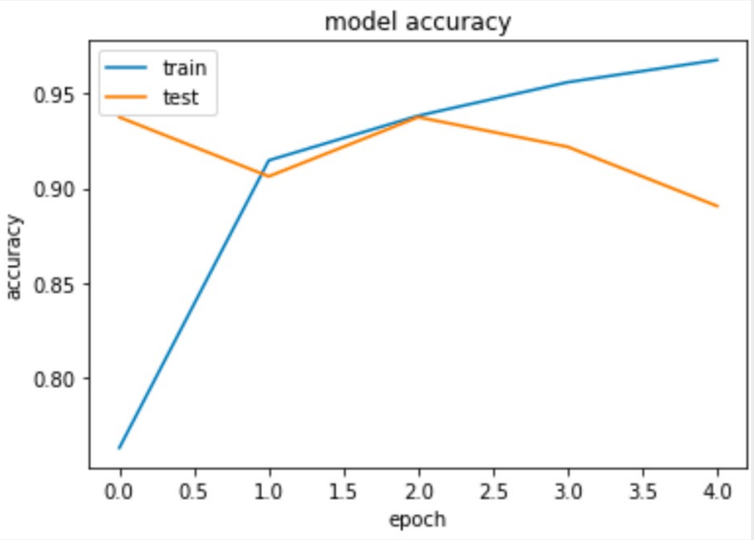
Model 4

5-10m/epoch

embeddingSize = 128

- model.add(LSTM(64, return_sequences=True, dropout=0.1))
- model.add(LayerNormalization())
- model.add(LSTM(32, return_sequences=True, dropout=0.1))
- model.add(LayerNormalization())
- model.add(LSTM(64, return_sequences=False))
- model.add(LayerNormalization())

- OVERFITTING
- Test accuracy: 0.7838

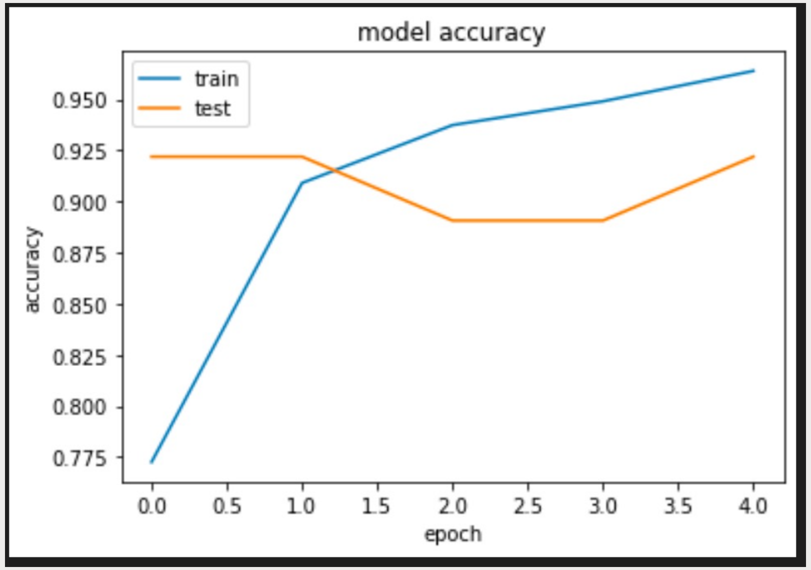


Model 5 (lstm_3)

5-10m/epoch

embeddingSize = 128

- model.add(LSTM(64, return_sequences=True, dropout=0.2))
- model.add(LayerNormalization())
- model.add(LSTM(32, return_sequences=True, dropout=0.2))
- model.add(LayerNormalization())
- model.add(LSTM(64, return_sequences=False))
- model.add(LayerNormalization())



- OVERFITTING
- Test accuracy: 0.8174

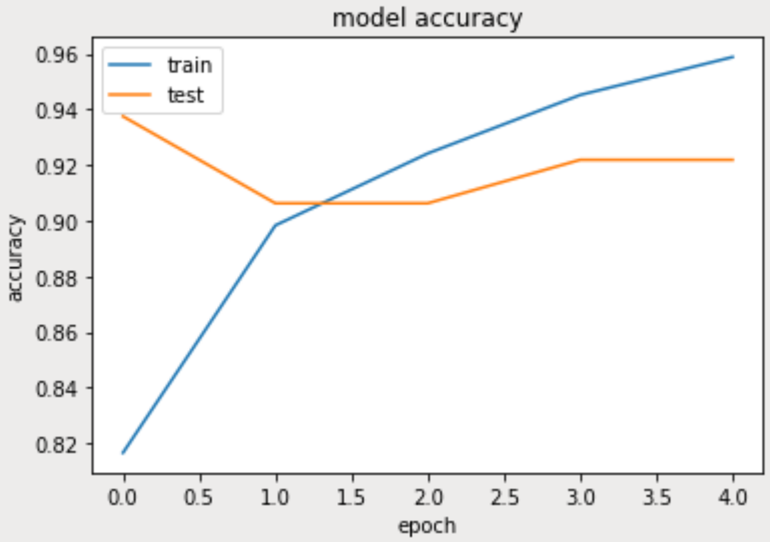
Model 6 (lstm_2)
27-33s/epoch

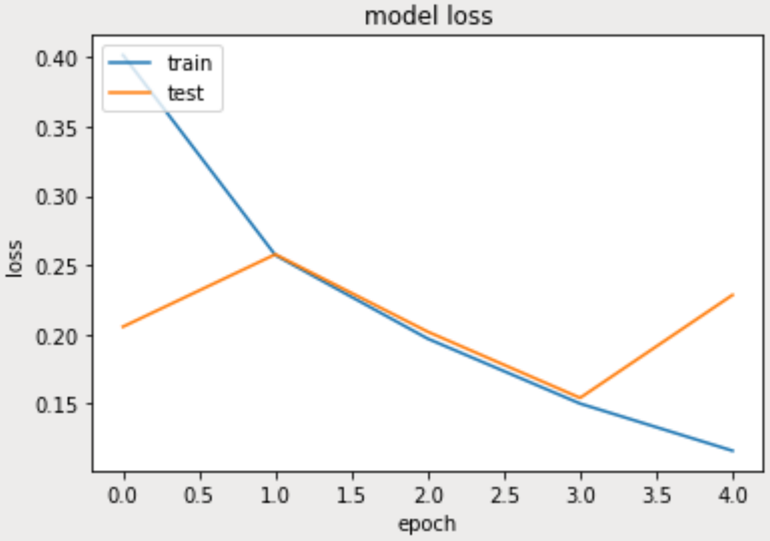
embeddingSize = 128

- model.add(LSTM(64, return_sequences=True))
- model.add(Dropout(0.2))

- model.add(LSTM(64, return_sequences=False))
- model.add(LayerNormalization())

- Test accuracy: 0.8336
- Train accuracy: 0.9588
- Validation accuracy: 0.9219





Model 7 (lstm_do)

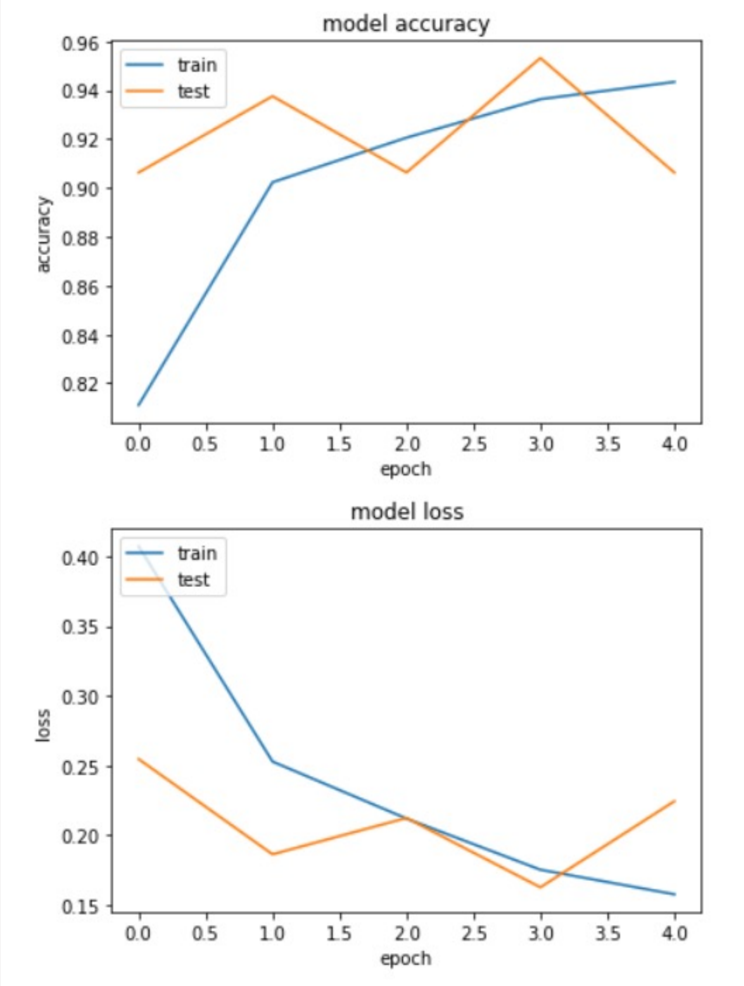
27-31s/epoch

embeddingSize = 128

- model.add(LSTM(64, return_sequences=True))
- model.add(Dropout(0.2))

- model.add(LSTM(64, return_sequences=False))
- model.add(Dropout(0.2))

- Test accuracy: 0.8722
- Train accuracy: 0.9433
- Validation accuracy: 0.9062



Model 8 (lstm_norm)

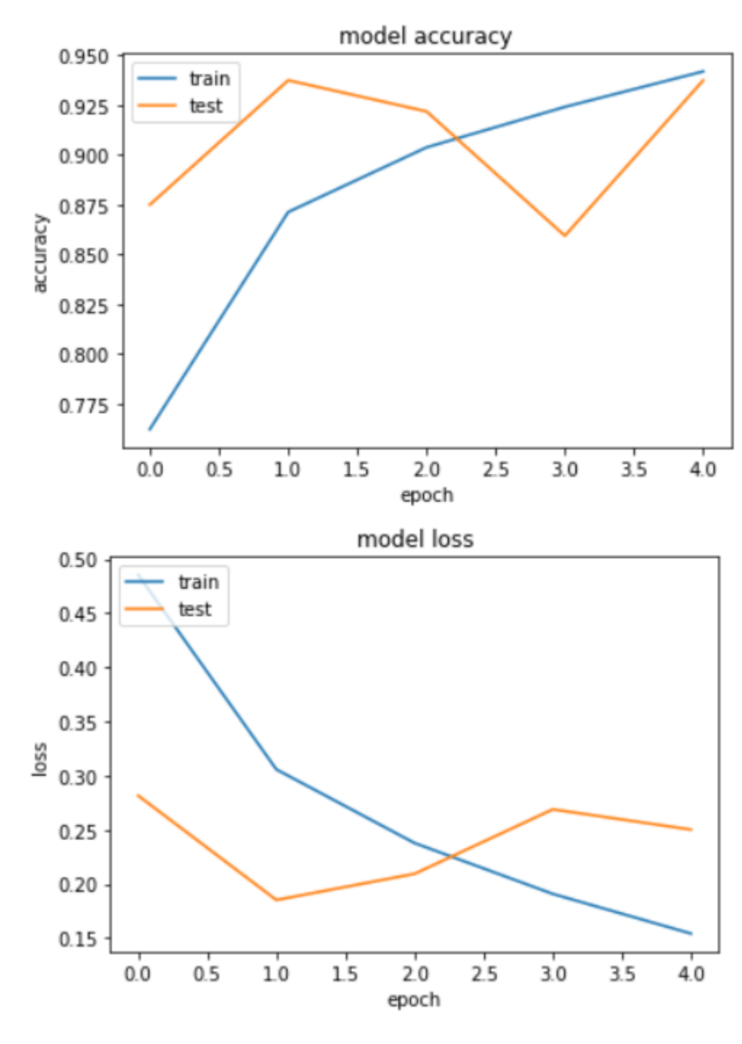
29-74s/epoch

embeddingSize = 128

- model.add(LSTM(64, return_sequences=True))
- model.add(LayerNormalization())

- model.add(LSTM(64, return_sequences=False))
- model.add(LayerNormalization())

- Test accuracy: 0.8687
- Train accuracy: 0.9420
- Validation accuracy: 0.9375



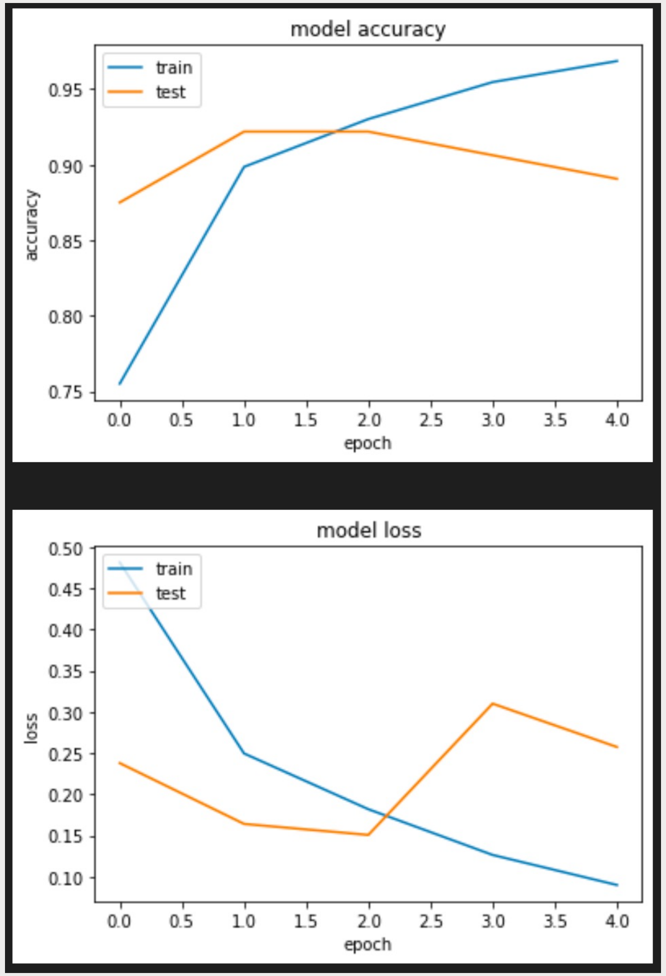
Model 9 (lstm_norm_256)
30-37s/epoch

embeddingSize = 256

- model.add(LSTM(64, return_sequences=True))
- model.add(LayerNormalization())

- model.add(LSTM(64, return_sequences=False))
- model.add(LayerNormalization())

- Test accuracy: 0.8330
- Train accuracy: 0.9661
- Validation accuracy: 0.9075



Final network: Model 7

- 2x LSTM layers, 2x Dropout
- Total parameters: 722,497

- Why selected?
  - *Highest test accuracy*
  - *Less overfitting than most other models*

- Embedding size = 128, because of community and papers suggestions
- Dropout, because better results

Deployment and tracking of experiments

- How has an experiment performed?
- We solved this by
  - Creating a branch for each experiment
  - Tracking the outcome of the branch
  - If an outcome is better than the main branch it is the new main branch

→ This was our simple way of tracking experiments

THANK YOU FOR YOUR ATTENTION.

QUESTIONS?