News Headline Classification

Hengxin Wu

Muwen Huang

Chenyue Jin

Jianqiao Zhang

GEORGETOWN UNIVERSITY

Catalog

- **Introduction**
- **Headline Classification in English News**
  - **LSTM**
  - **CNN**
  - **BERT**
  - **HAN**
- **Headline Classification in Chinese News**
  - **LSTM**
- **Conclusion**

Introduction

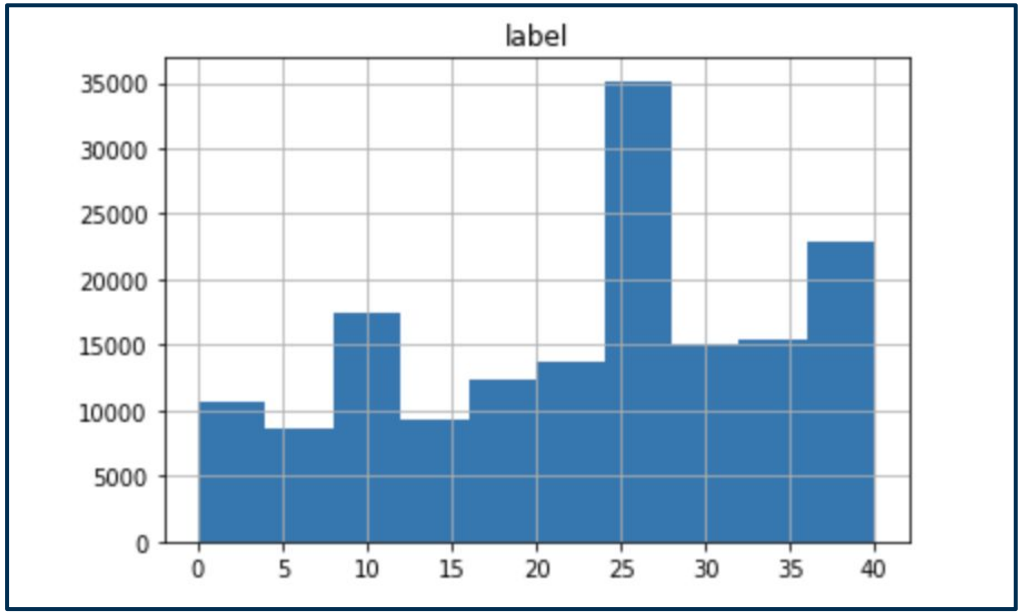
News Category Dataset v2

- **Dimension: 200853 rows and 40 unique labels**
- **Date range: from 2012-01-28 to 2018-05-26**
- **Data fields: category, headline, authors, link, short description, date**
- **Biggest drawback: imbalanced labels**

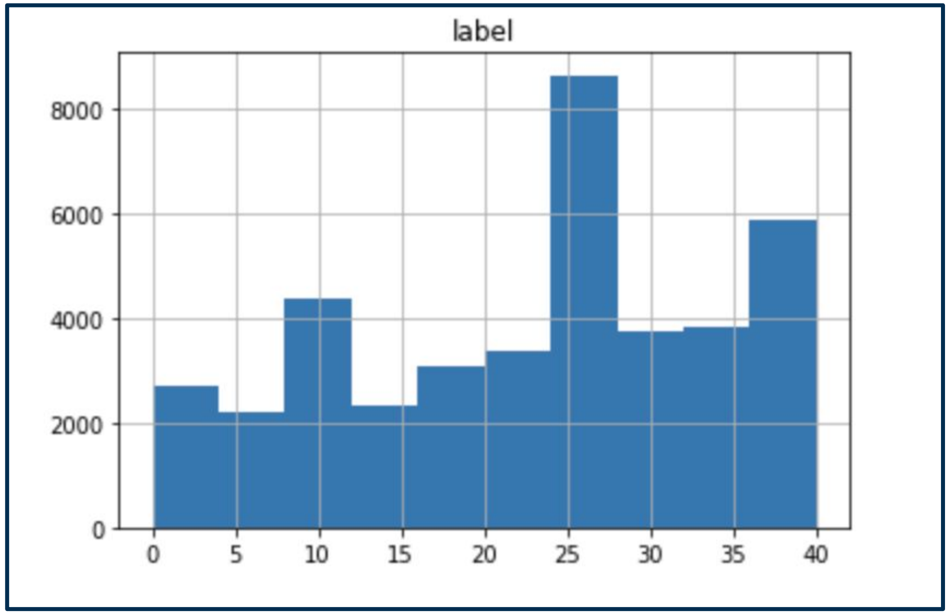
Data Source: https://www.kaggle.com/datasets/yazansalameh/news-category-dataset-v2

Label Distribution - English News with all categories

Train

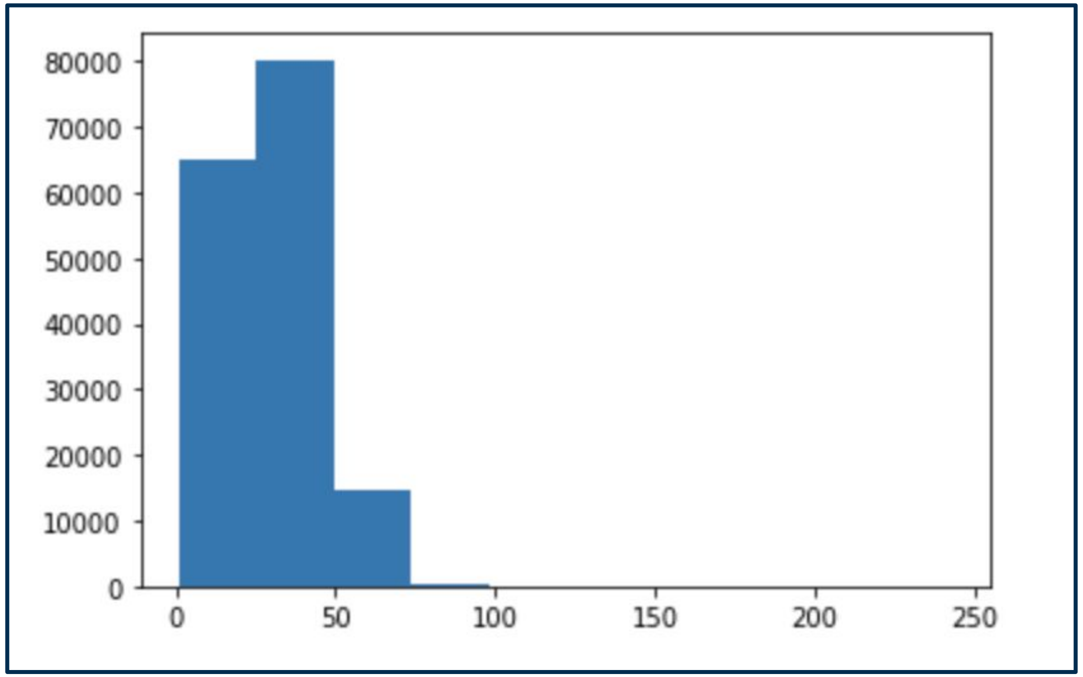


Test

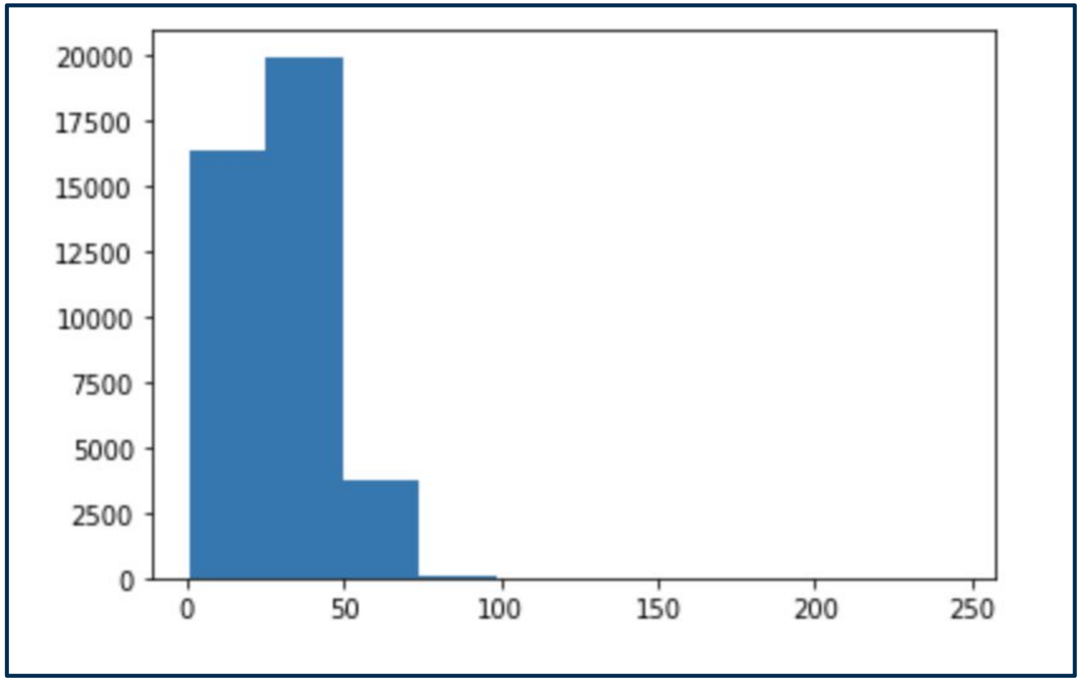


Headline Length Distribution - English News with all categories

Train

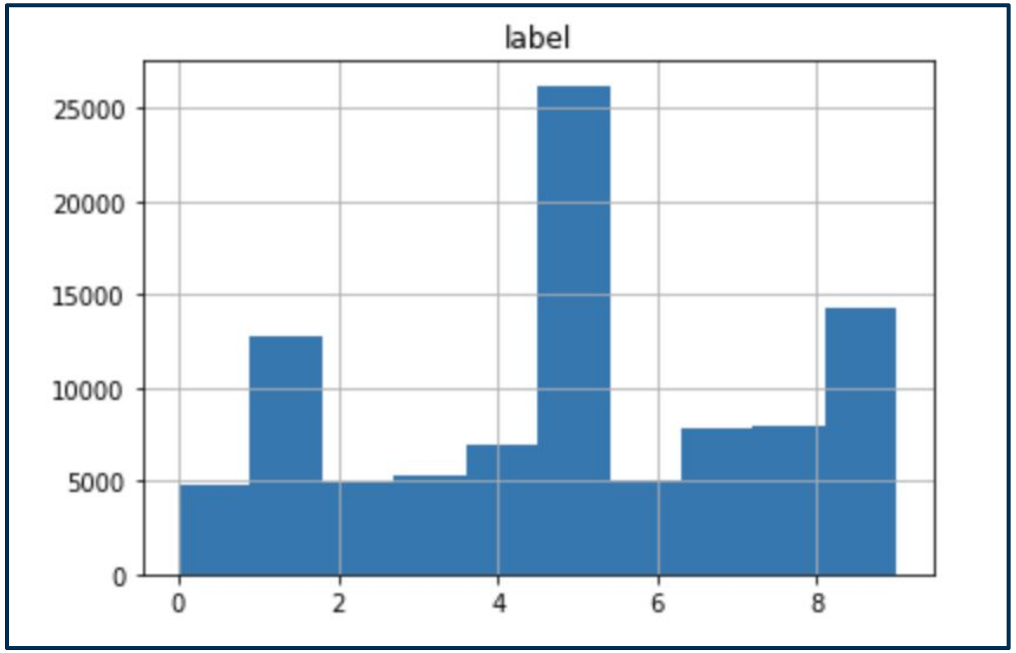


Test

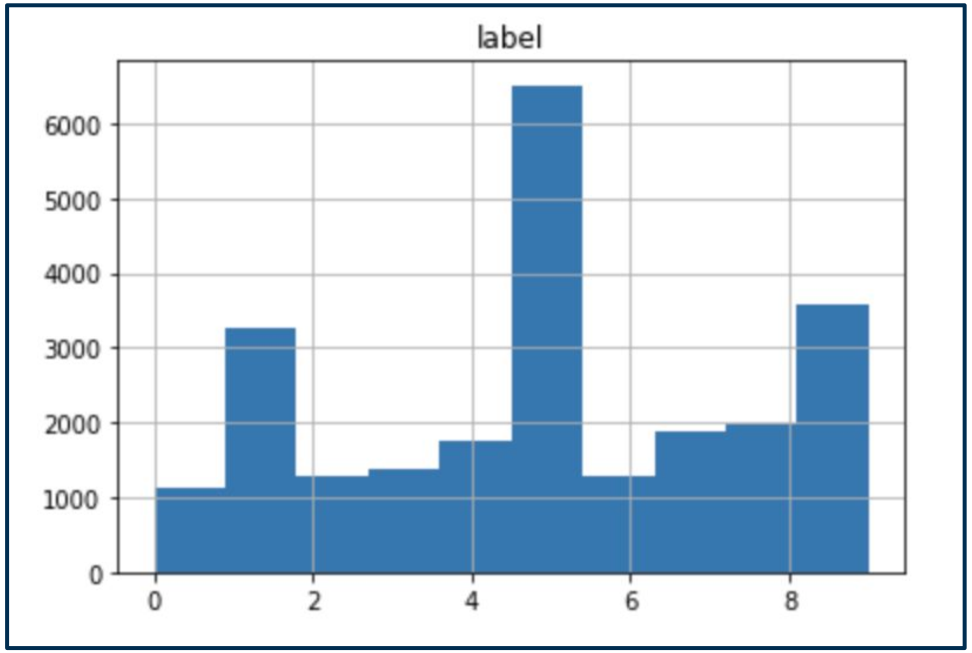


Label Distribution - English News with top 10 categories

Train

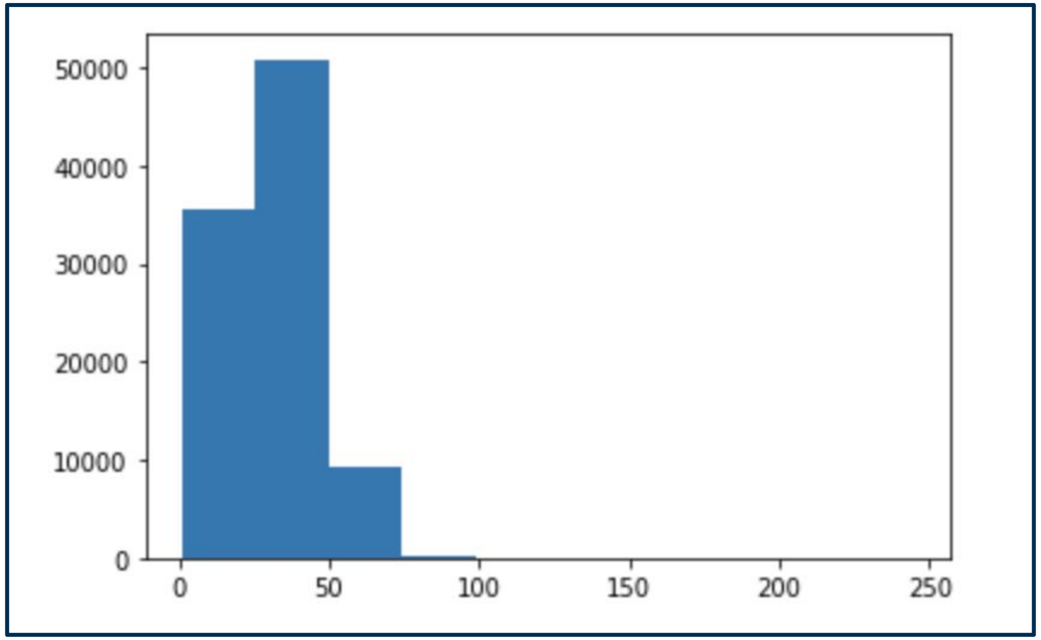


Test

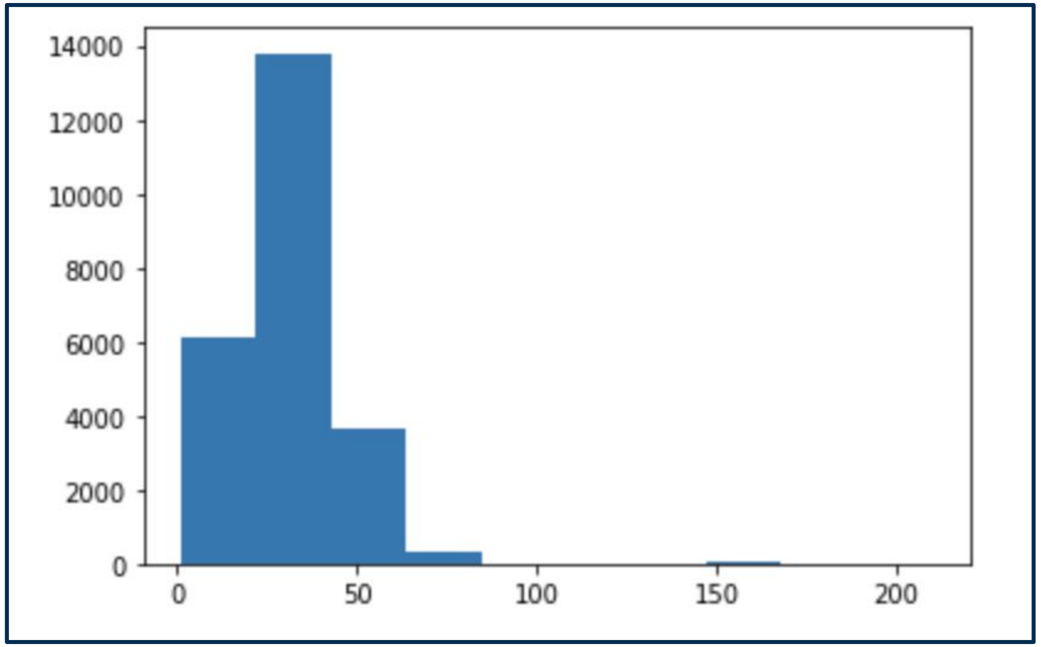


Headline Length Distribution - English News with top 10 categories

Train



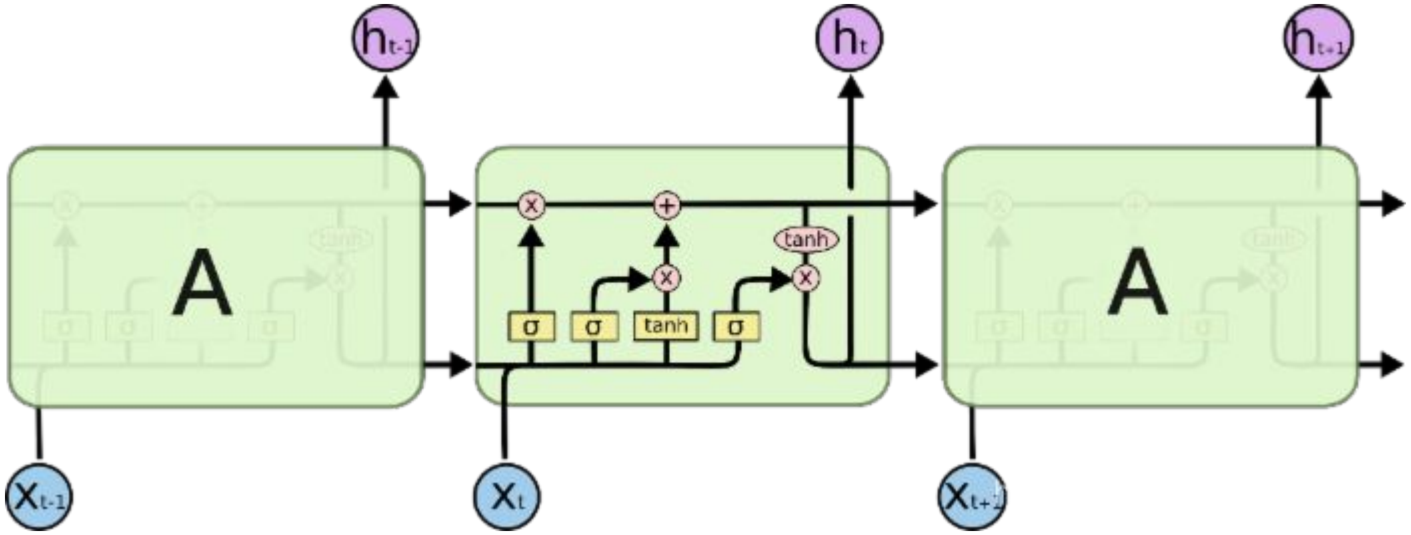
Test

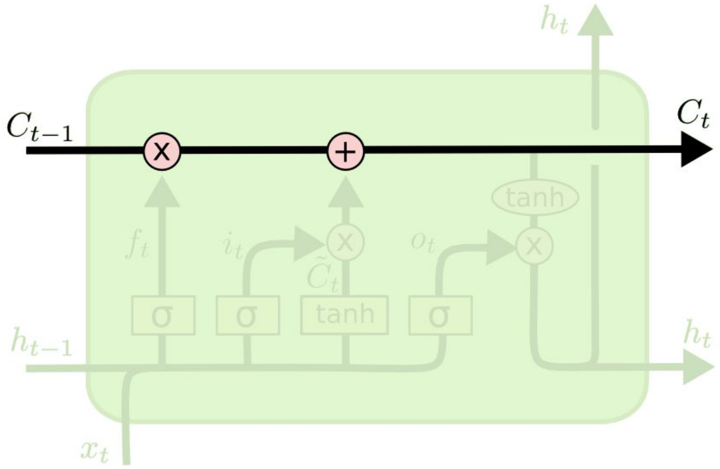


LSTM

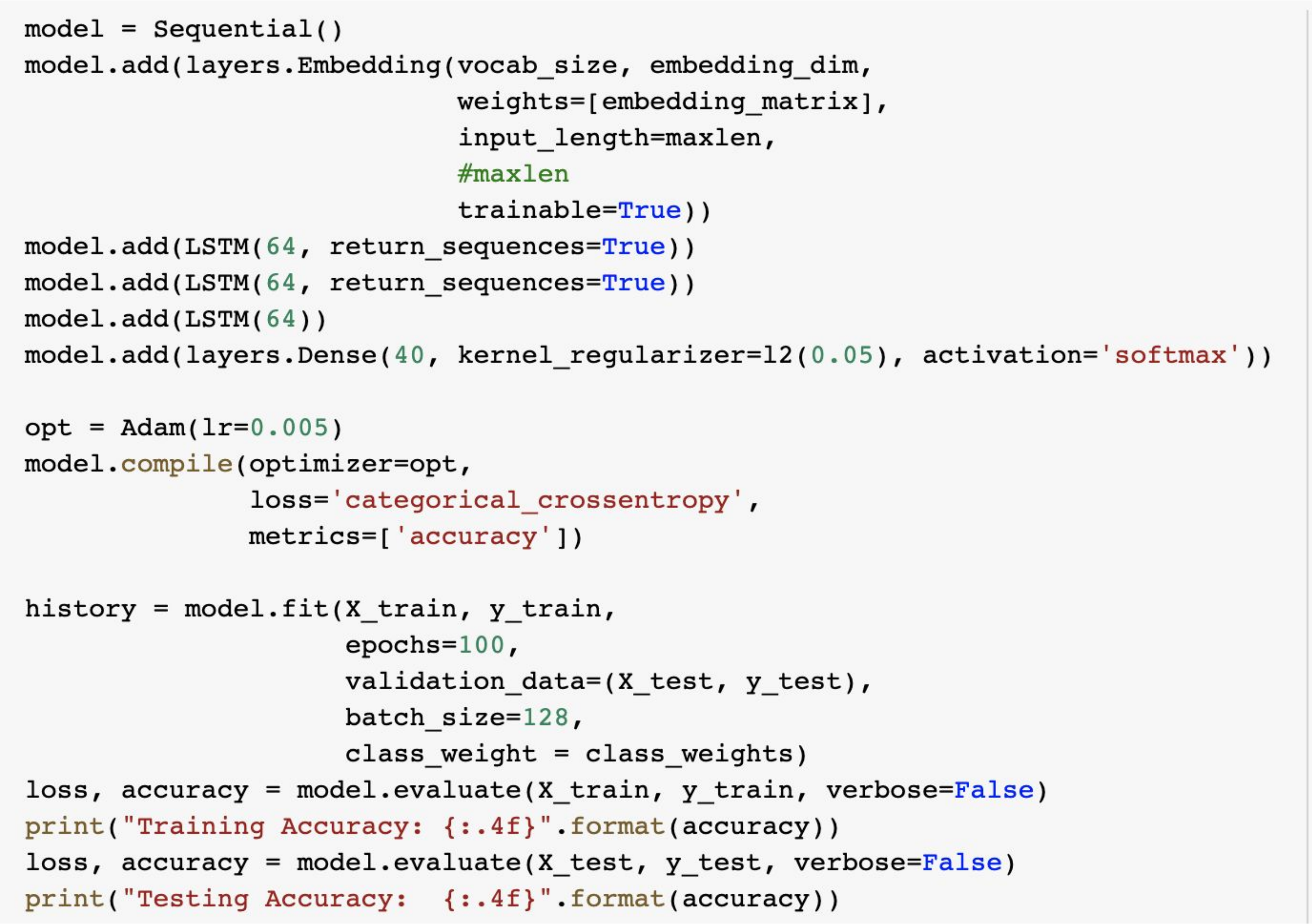
LSTM

- Long Short Term Memory networks are a special kind of RNN, capable of learning long-term dependencies.
- A chain of repeating modules of neural network.
- Have four neural network layers, interacting in a very special way.





LSTM Model



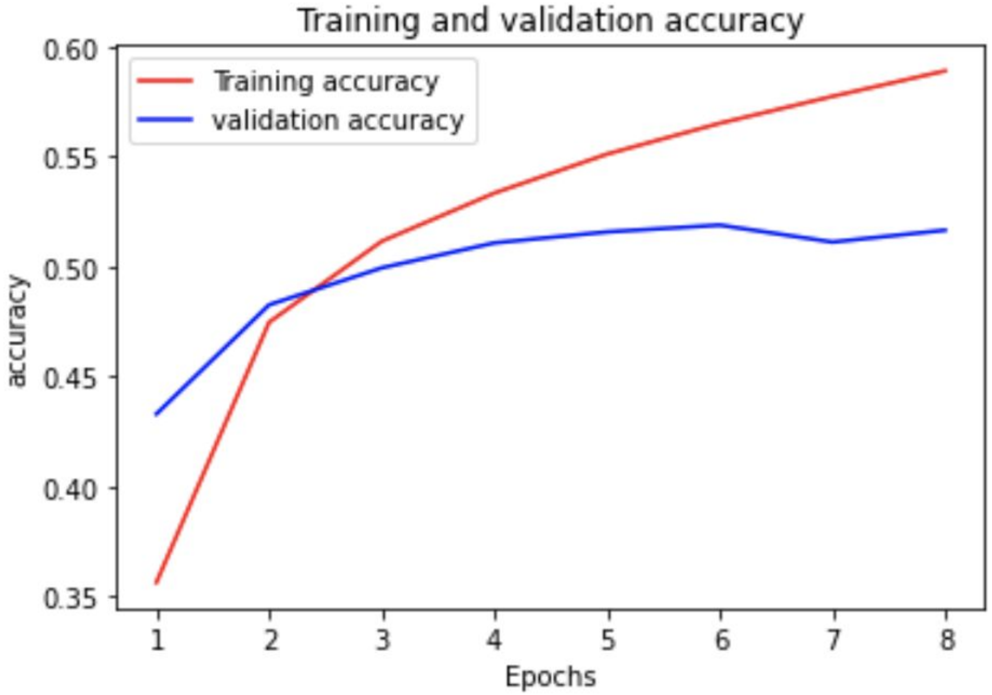
```
model = Sequential()
model.add(layers.Embedding(vocab_size, embedding_dim,
                           weights=[embedding_matrix],
                           input_length=maxlen,
                           #maxlen
                           trainable=True))
model.add(LSTM(64, return_sequences=True))
model.add(LSTM(64, return_sequences=True))
model.add(LSTM(64))
model.add(layers.Dense(40, kernel_regularizer=l2(0.05), activation='softmax'))

opt = Adam(lr=0.005)
model.compile(optimizer=opt,
              loss='categorical_crossentropy',
              metrics=['accuracy'])

history = model.fit(X_train, y_train,
                    epochs=100,
                    validation_data=(X_test, y_test),
                    batch_size=128,
                    class_weight = class_weights)
loss, accuracy = model.evaluate(X_train, y_train, verbose=False)
print("Training Accuracy: {:.4f}".format(accuracy))
loss, accuracy = model.evaluate(X_test, y_test, verbose=False)
print("Testing Accuracy:  {:.4f}".format(accuracy))
```

- Keras
- Sequential Layer
- Embedding Layer
- LSTM Layer
- Dense Layer

Train Accuracy VS Test Accuracy

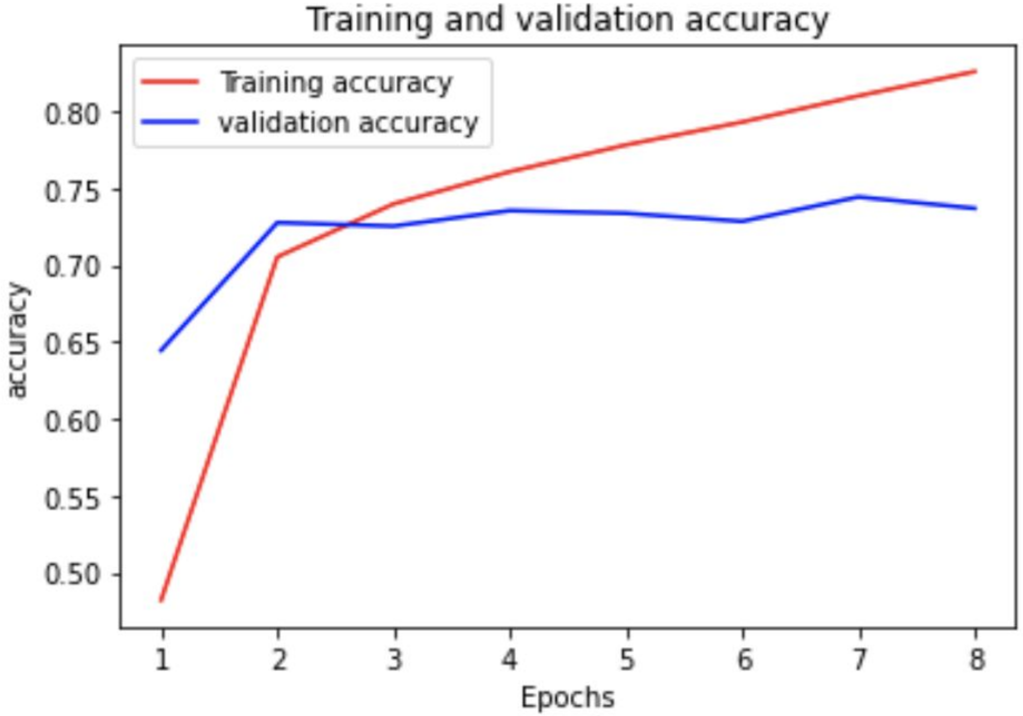
Full

- Accuracy for cross validation is around 0.5



Partial (Top 10 labels)

- Accuracy for cross validation is around 0.7



1D-CNN

1D-CNN

- 2D CNNs use 2D convolutional kernels to predict the segmentation map for a single slice.

- 1D Convolutional Neural Networks are similar to 2D CNN.

- 1D Convolutional Neural Networks are used mainly used on text and 1D signals.

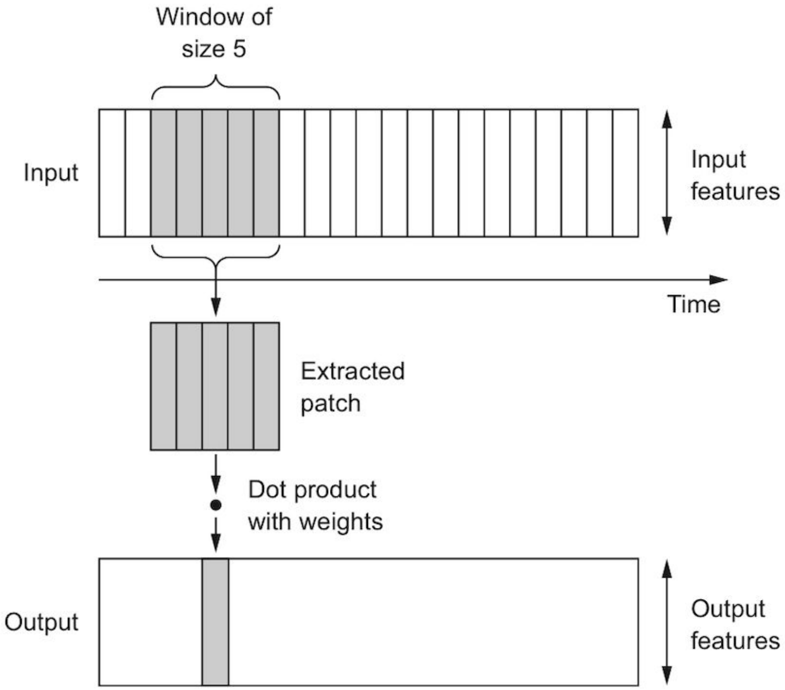
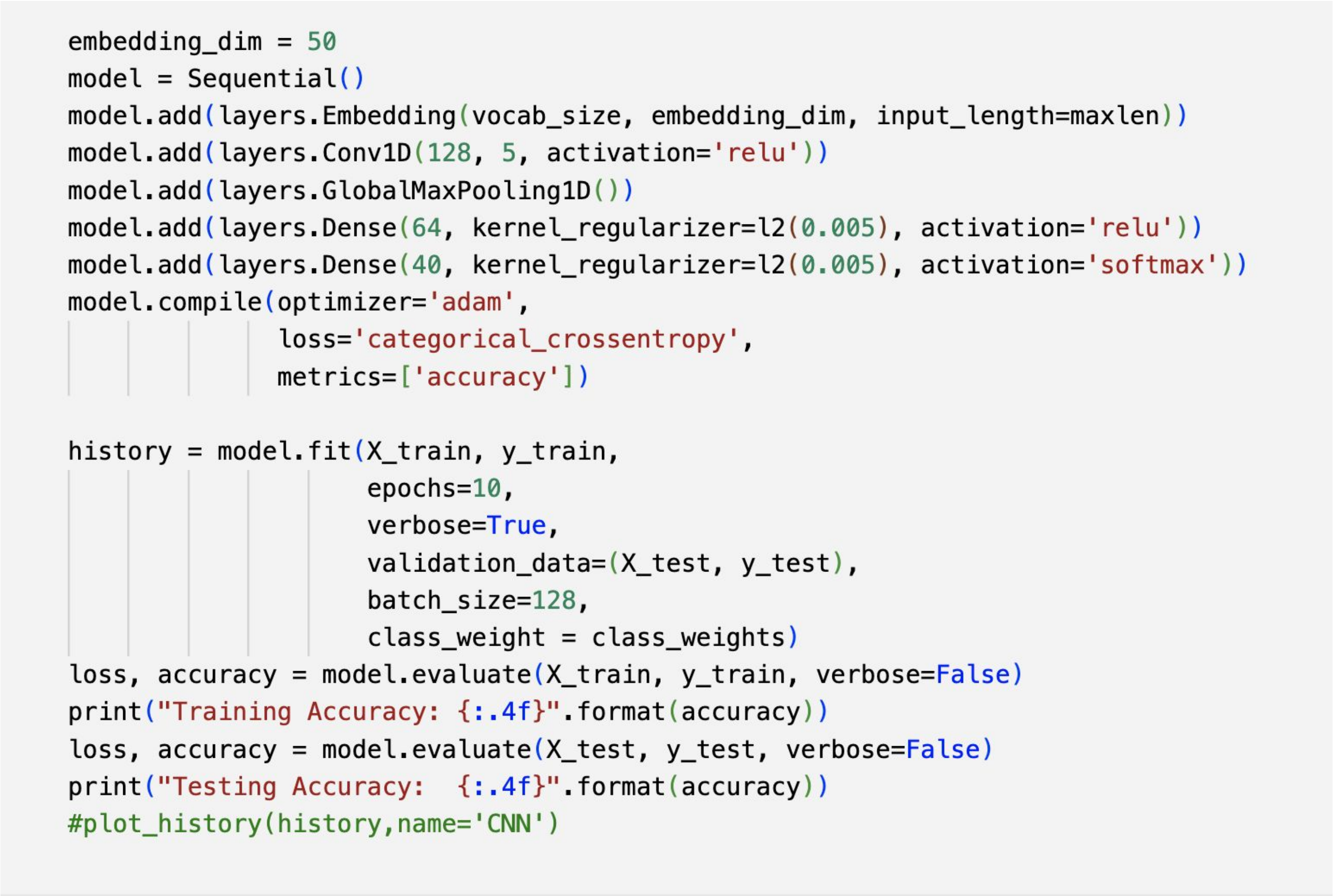


Figure 6.26 **How 1D convolution works: each output timestep is obtained from a temporal patch in the input sequence.**

1D-CNN Model

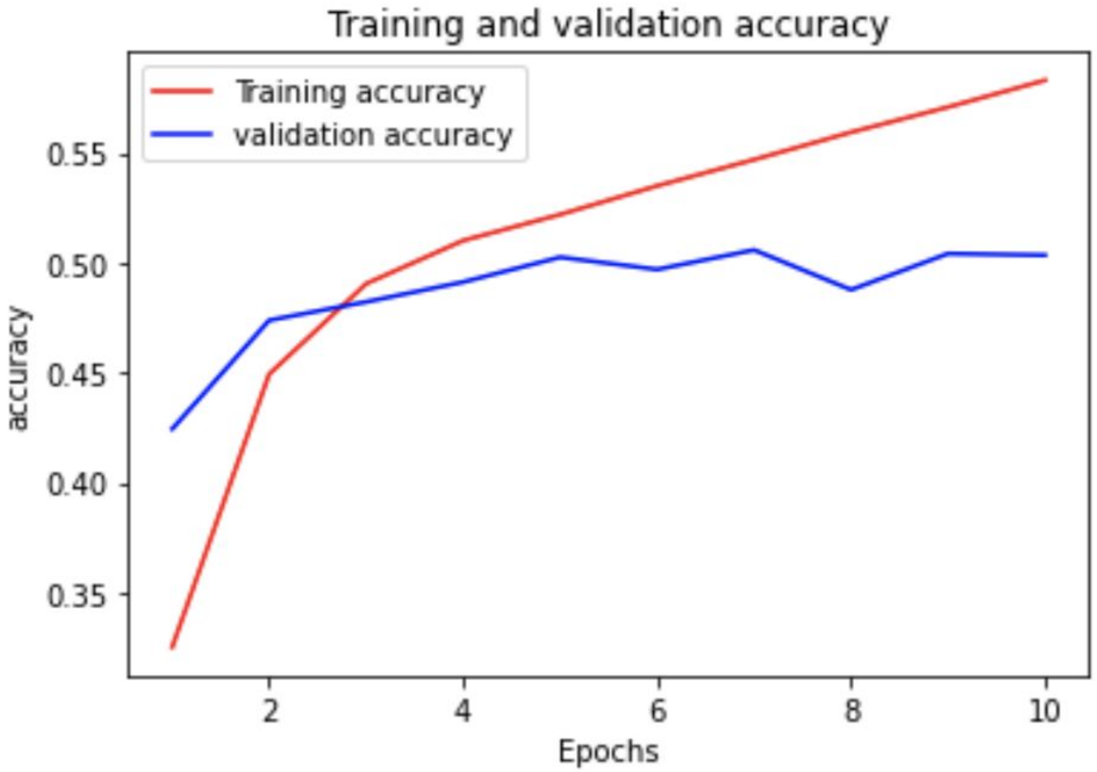


```
embedding_dim = 50
model = Sequential()
model.add(layers.Embedding(vocab_size, embedding_dim, input_length=maxlen))
model.add(layers.Conv1D(128, 5, activation='relu'))
model.add(layers.GlobalMaxPooling1D())
model.add(layers.Dense(64, kernel_regularizer=l2(0.005), activation='relu'))
model.add(layers.Dense(40, kernel_regularizer=l2(0.005), activation='softmax'))
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

history = model.fit(X_train, y_train,
                    epochs=10,
                    verbose=True,
                    validation_data=(X_test, y_test),
                    batch_size=128,
                    class_weight = class_weights)
loss, accuracy = model.evaluate(X_train, y_train, verbose=False)
print("Training Accuracy: {:.4f}".format(accuracy))
loss, accuracy = model.evaluate(X_test, y_test, verbose=False)
print("Testing Accuracy:  {:.4f}".format(accuracy))
#plot_history(history,name='CNN')
```

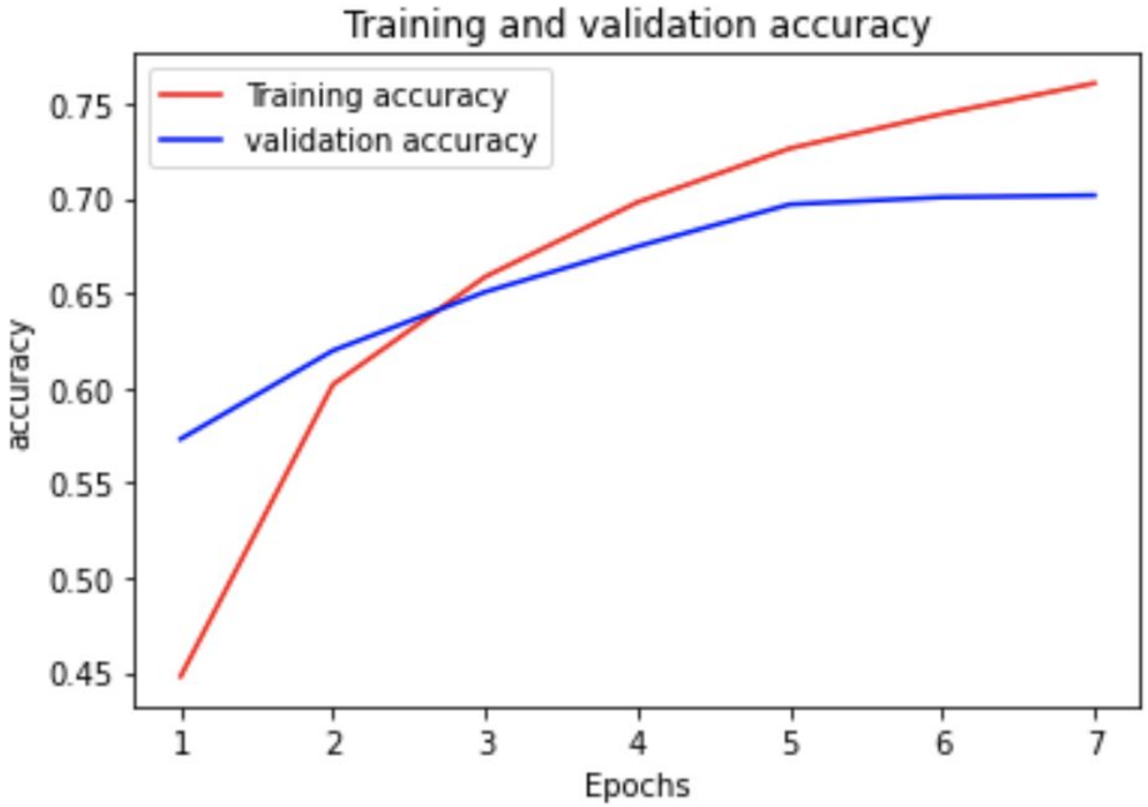
- Embedding Layer
- Conv1D Layer
- GlobalMaxPooling1D Layer
- Dense Layer

Train Accuracy VS Test Accuracy

Full



Partial



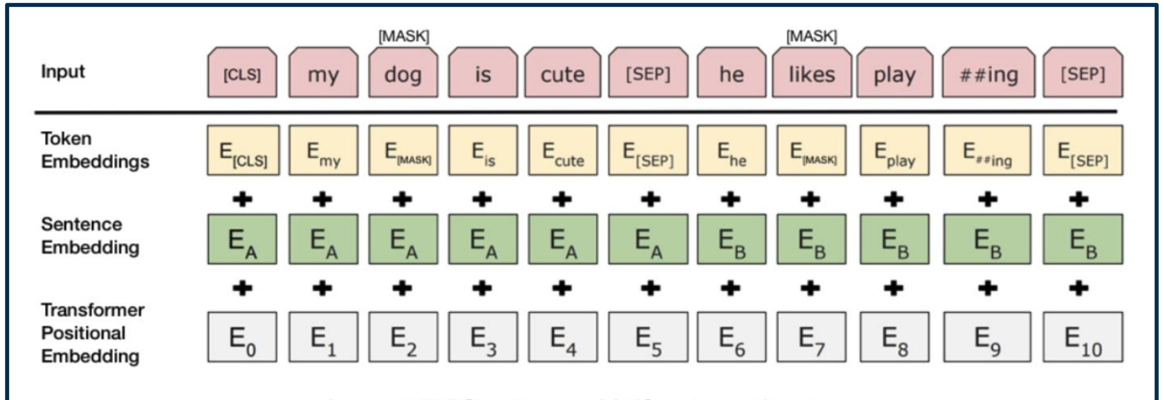
BERT

BERT

- A transformer-based model that uses self-attention as a primary means to express the relationship between variables in a sequence.

- Since Bert is originally designed for generating language models, it utilizes the [CLS] token and the [SEP] token to represent the beginning and the end of each sentence.

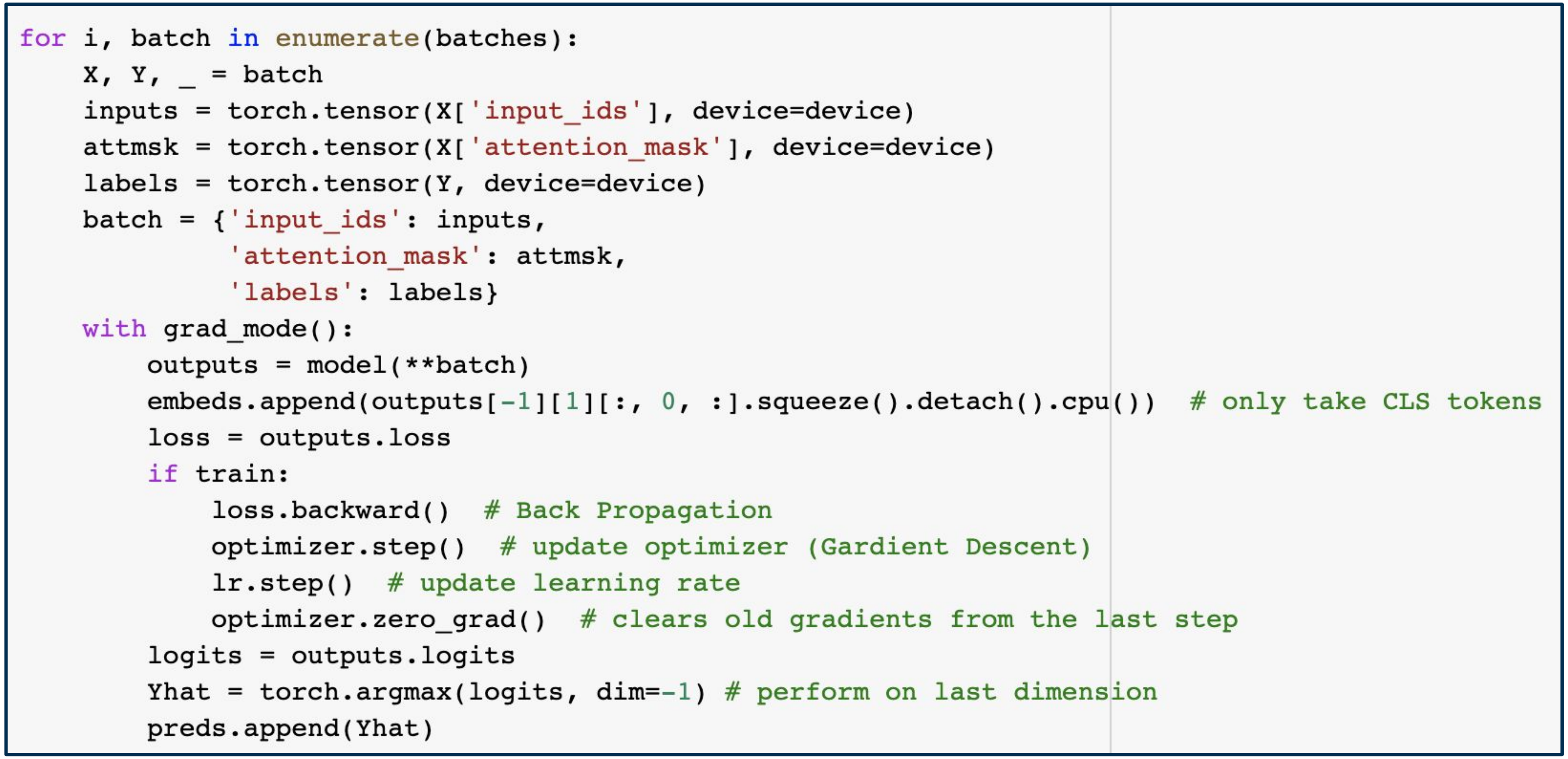
- For classification tasks, a classification layer can be added on top of the output for the [CLS] token.



Input
[MASK]
[MASK]
[CLS] my dog is cute [SEP] he likes play ##ing [SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} $E_{[MASK]}$ E_{is} E_{cute} $E_{[SEP]}$ E_{he} $E_{[MASK]}$ E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Sentence Embedding
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Transformer Positional Embedding
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}

Source: BERT [Devlin et al., 2018], with modifications

Bert Model



```
for i, batch in enumerate(batches):
    X, Y, _ = batch
    inputs = torch.tensor(X['input_ids'], device=device)
    attmsk = torch.tensor(X['attention_mask'], device=device)
    labels = torch.tensor(Y, device=device)
    batch = {'input_ids': inputs,
             'attention_mask': attmsk,
             'labels': labels}
    with grad_mode():
        outputs = model(**batch)
        embeds.append(outputs[-1][1][:, 0, :].squeeze().detach().cpu())  # only take CLS tokens
        loss = outputs.loss
        if train:
            loss.backward()  # Back Propagation
            optimizer.step()  # update optimizer (Gardient Descent)
            lr.step()  # update learning rate
            optimizer.zero_grad()  # clears old gradients from the last step
        logits = outputs.logits
        Yhat = torch.argmax(logits, dim=-1) # perform on last dimension
        preds.append(Yhat)
```

Training loop from scratch

- Retrieve input ids, attention mask, and labels from batch
- Apply pretrained model
- Save CLS embeddings
- Perform gradient descent for training batches
- Get the predictions

Using all categories

Training set

	loss	f1-score	accuracy
epoch 1	1.550	0.412	0.600
epoch 2	1.201	0.514	0.677

Test set

	loss	f1-score	accuracy
epoch 1	1.221	0.506	0.666

Using top 10 categories

Training set

	loss	f1-score	accuracy
epoch 1	0.618	0.765	0.812
epoch 2	0.430	0.834	0.869

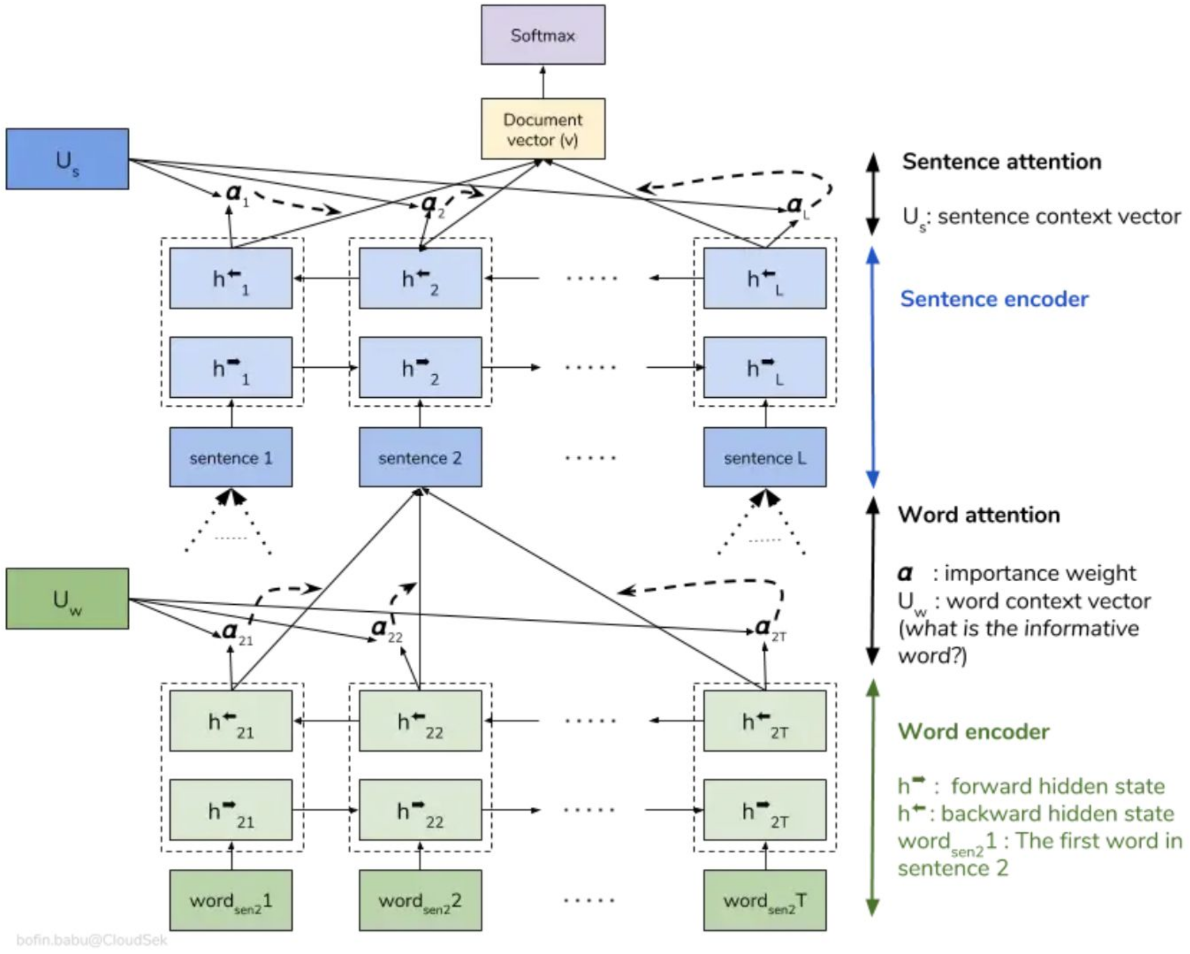
Test set

	loss	f1-score	accuracy
epoch 1	0.463	0.817	0.854

Hierarchical Attention Networks

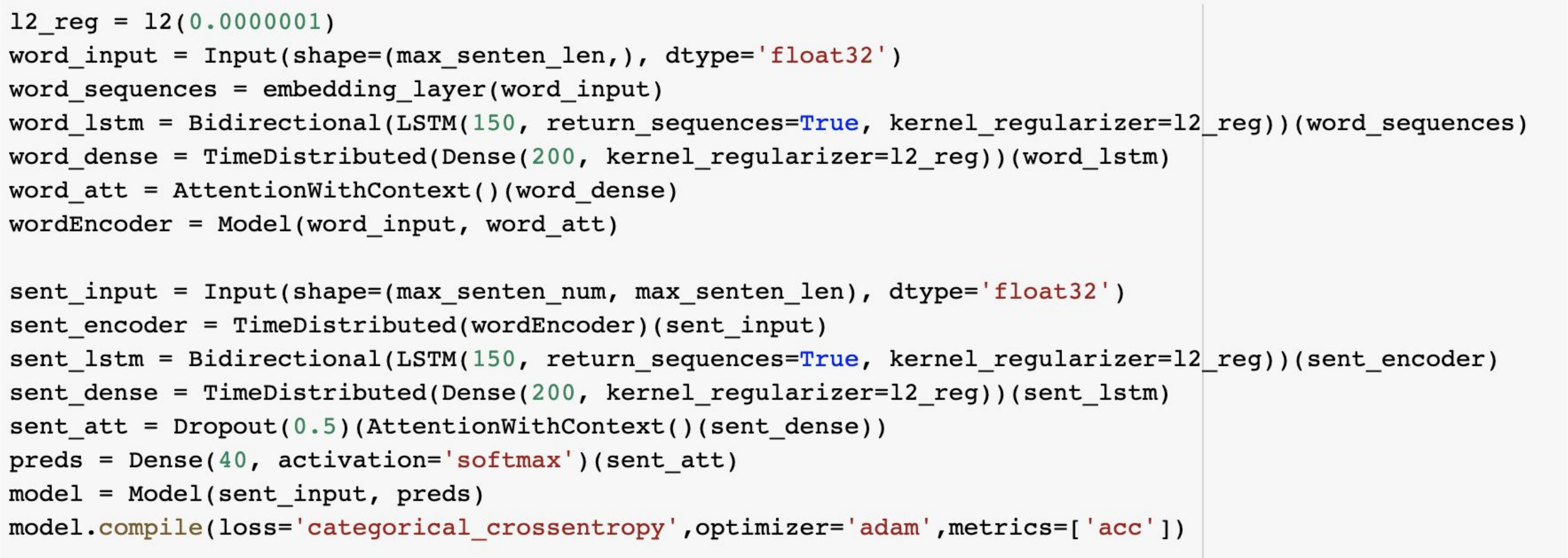
HAN

It uses stacked recurrent neural networks on word level followed by attention model.



HAN Structure

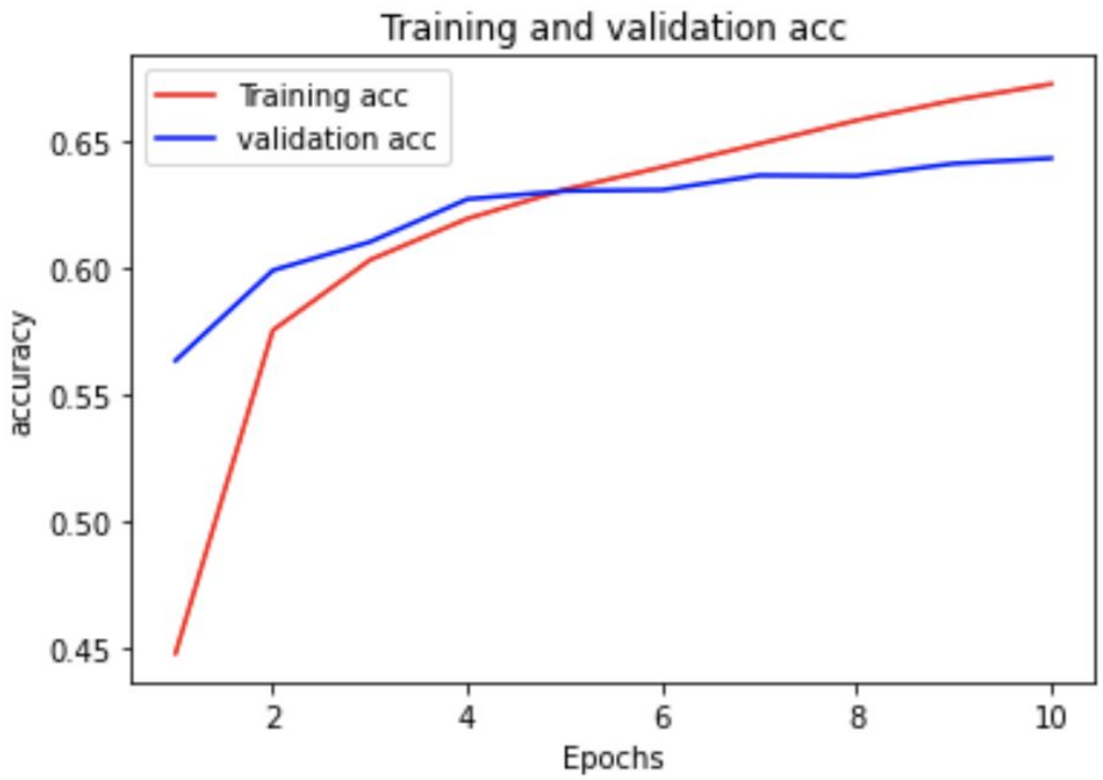
HAN Model



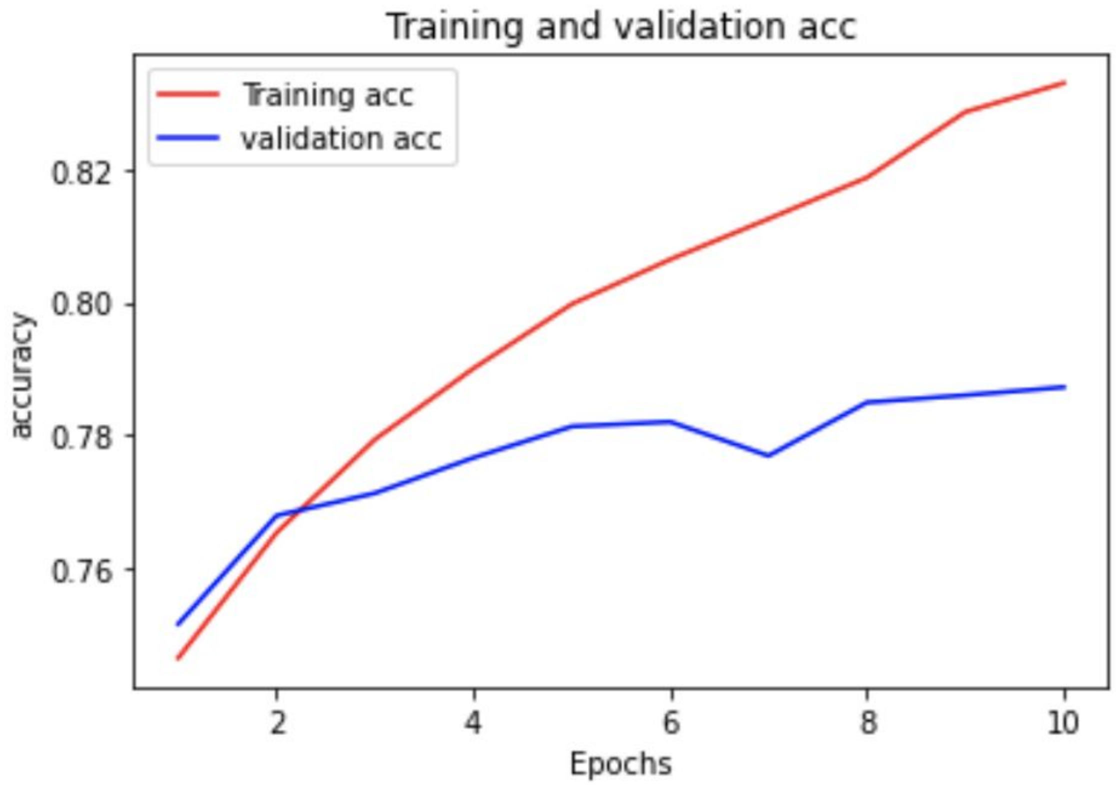
```
l2_reg = l2(0.0000001)
word_input = Input(shape=(max_senten_len,), dtype='float32')
word_sequences = embedding_layer(word_input)
word_lstm = Bidirectional(LSTM(150, return_sequences=True, kernel_regularizer=l2_reg))(word_sequences)
word_dense = TimeDistributed(Dense(200, kernel_regularizer=l2_reg))(word_lstm)
word_att = AttentionWithContext()(word_dense)
wordEncoder = Model(word_input, word_att)

sent_input = Input(shape=(max_senten_num, max_senten_len), dtype='float32')
sent_encoder = TimeDistributed(wordEncoder)(sent_input)
sent_lstm = Bidirectional(LSTM(150, return_sequences=True, kernel_regularizer=l2_reg))(sent_encoder)
sent_dense = TimeDistributed(Dense(200, kernel_regularizer=l2_reg))(sent_lstm)
sent_att = Dropout(0.5)(AttentionWithContext()(sent_dense))
preds = Dense(40, activation='softmax')(sent_att)
model = Model(sent_input, preds)
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['acc'])
```

Training Accuracy VS Testing Accuracy

Full



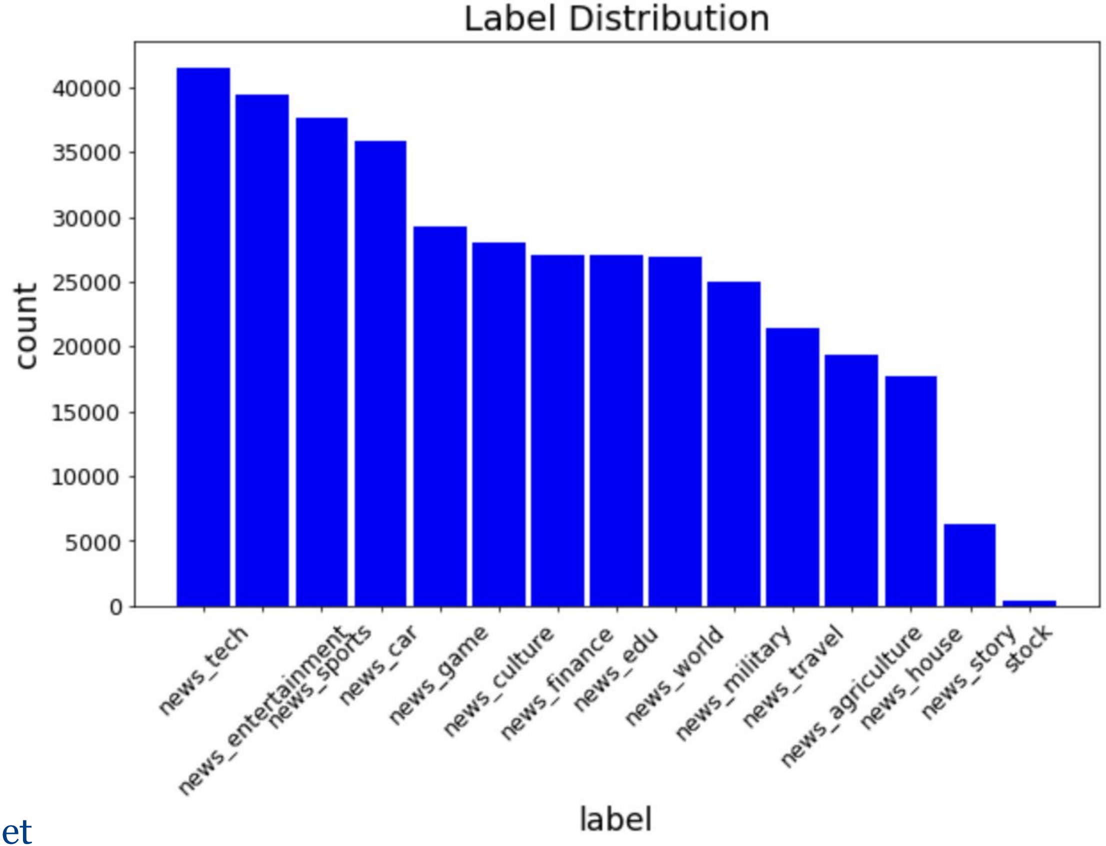
Partial



Headline Classification in Chinese News (LSTM)

Chinese News Headline Classification

- 38000+ news headlines, 15 categories (labels) from a Chinese news application



[TouTiao.com](https://www.toutiao.com)

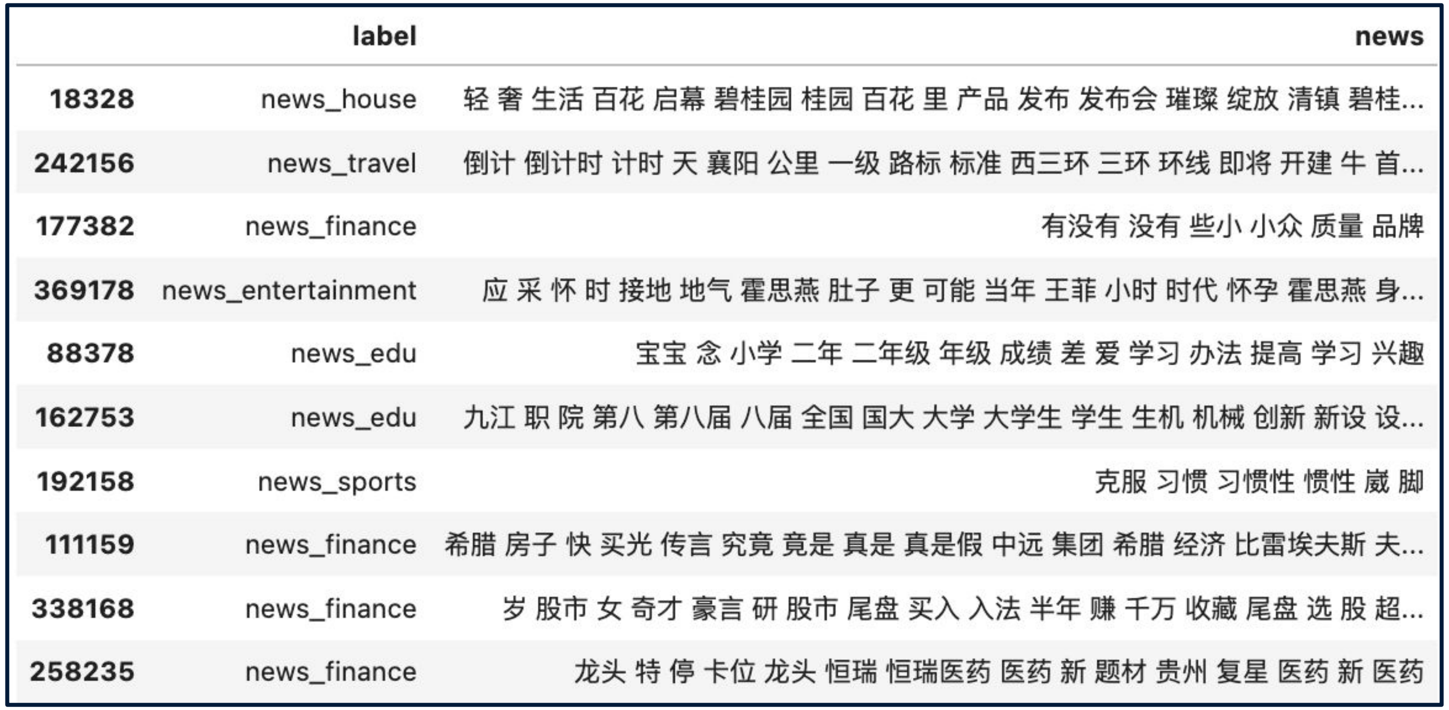
Owner: ByteDance

Data Source:

https://github.com/aceimnorstuvwxz/toutiao-text-classfication-dataset

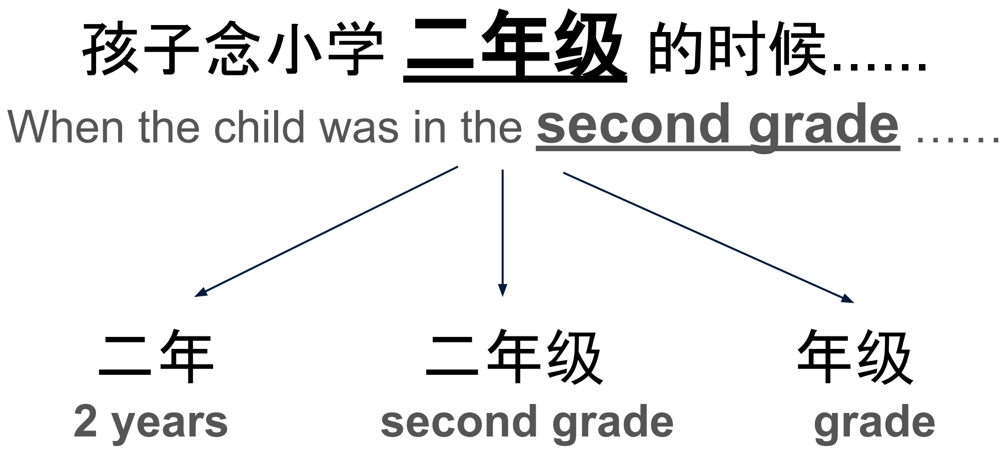
Chinese News Headline Classification

- Jieba package for Chinese text segmentation
- Full Mode



	label	news
18328	news_house	轻 奢 生活 百花 启幕 碧桂园 桂园 百花 里 产品 发布 发布会 璀璨 绽放 清镇 碧桂...
242156	news_travel	倒计 倒计时 计时 天 襄阳 公里 一级 路标 标准 西三环 三环 环线 即将 开建 牛 首...
177382	news_finance	有没有 没有 些小 小众 质量 品牌
369178	news_entertainment	应 采 怀 时 接地 地气 霍思燕 肚子 更 可能 当年 王菲 小时 时代 怀孕 霍思燕 身...
88378	news_edu	宝宝 念 小学 二年 二年级 年级 成绩 差 爱 学习 办法 提高 学习 兴趣
162753	news_edu	九江 职 院 第八 第八届 八届 全国 国大 大学 大学生 学生 生机 机械 创新 新设 设...
192158	news_sports	克服 习惯 习惯性 惯性 崴 脚
111159	news_finance	希腊 房子 快 买光 传言 究竟 竟是 真是 真是假 中远 集团 希腊 经济 比雷埃夫斯 夫...
338168	news_finance	岁 股市 女 奇才 豪言 研 股市 尾盘 买入 入法 半年 赚 千万 收藏 尾盘 选 股 超...
258235	news_finance	龙头 特 停 卡位 龙头 恒瑞 恒瑞医药 医药 新 题材 贵州 复星 医药 新 医药

Example：



孩子念小学 **二年级** 的时候……

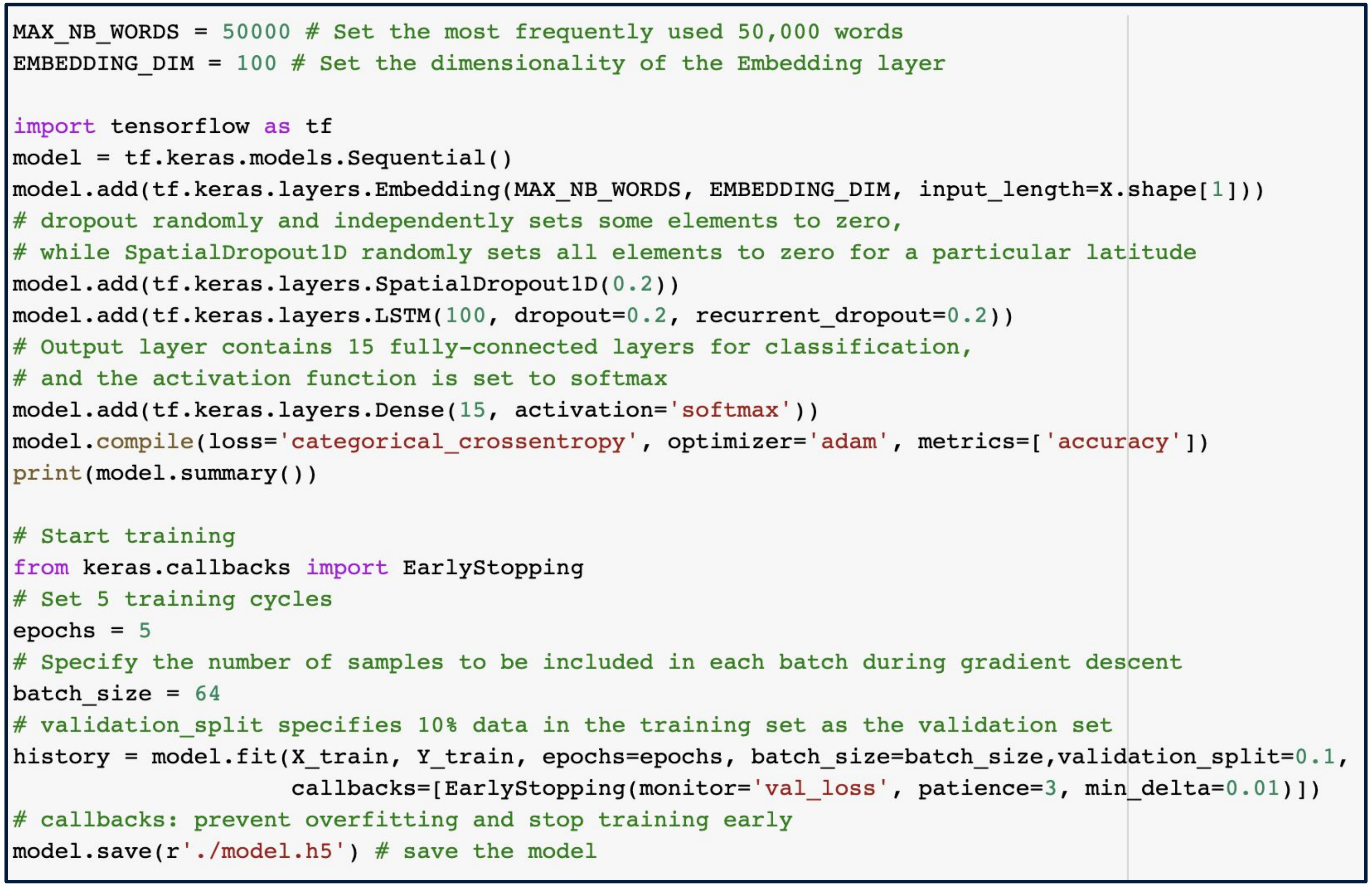
When the child was in the **second grade** ……

二年
2 years

二年级
second grade

年级
grade

LSTM Model

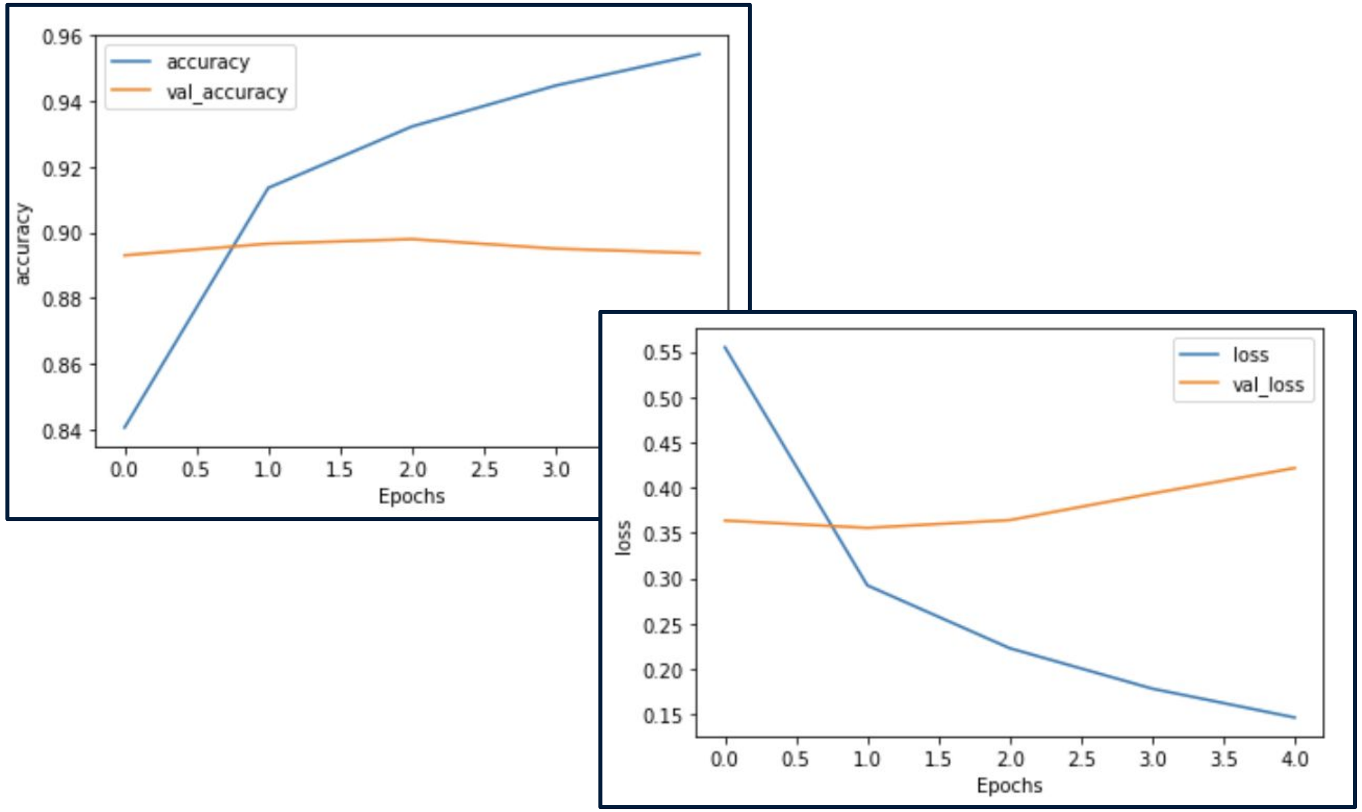


```
MAX_NB_WORDS = 50000 # Set the most frequently used 50,000 words
EMBEDDING_DIM = 100 # Set the dimensionality of the Embedding layer

import tensorflow as tf
model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Embedding(MAX_NB_WORDS, EMBEDDING_DIM, input_length=X.shape[1]))
# dropout randomly and independently sets some elements to zero,
# while SpatialDropout1D randomly sets all elements to zero for a particular latitude
model.add(tf.keras.layers.SpatialDropout1D(0.2))
model.add(tf.keras.layers.LSTM(100, dropout=0.2, recurrent_dropout=0.2))
# Output layer contains 15 fully-connected layers for classification,
# and the activation function is set to softmax
model.add(tf.keras.layers.Dense(15, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
print(model.summary())

# Start training
from keras.callbacks import EarlyStopping
# Set 5 training cycles
epochs = 5
# Specify the number of samples to be included in each batch during gradient descent
batch_size = 64
# validation_split specifies 10% data in the training set as the validation set
history = model.fit(X_train, Y_train, epochs=epochs, batch_size=batch_size,validation_split=0.1,
                    callbacks=[EarlyStopping(monitor='val_loss', patience=3, min_delta=0.01)])
# callbacks: prevent overfitting and stop training early
model.save(r'./model.h5') # save the model
```

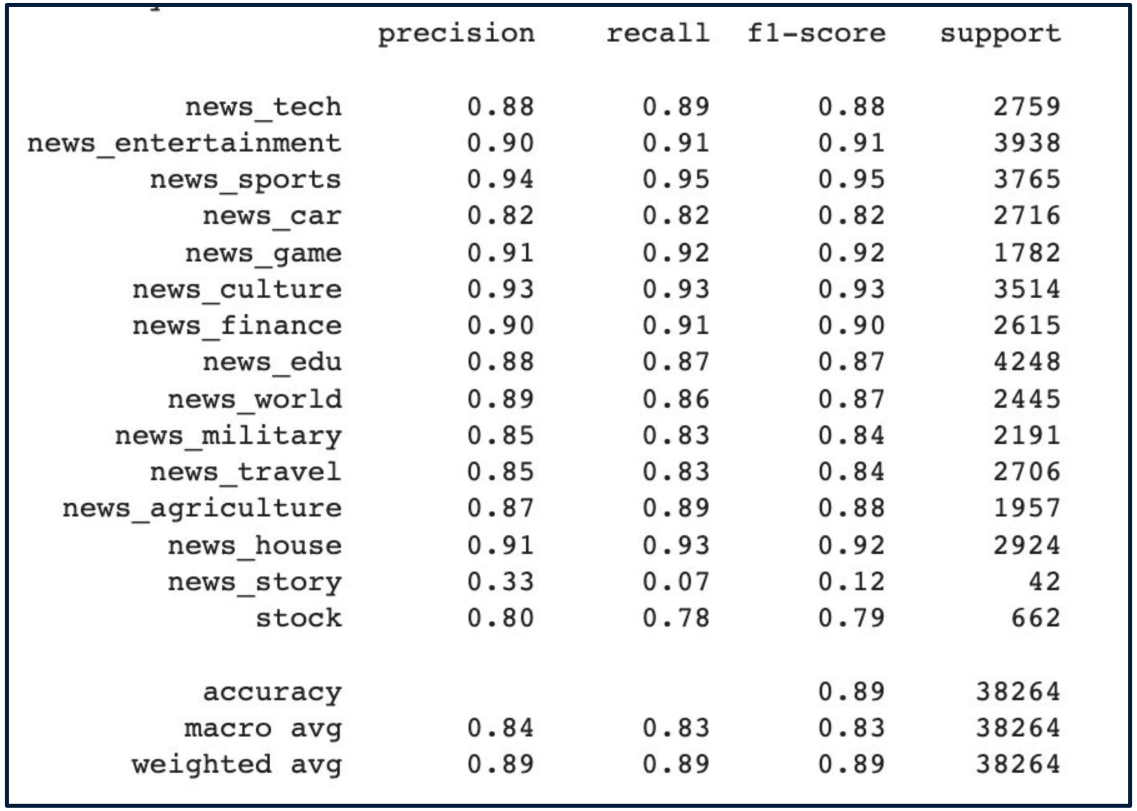
- Keras
- Sequential Layer
- Embedding Layer
- Dropout Layer
- LSTM Layer
- Dense Layer

Train Accuracy VS Test Accuracy

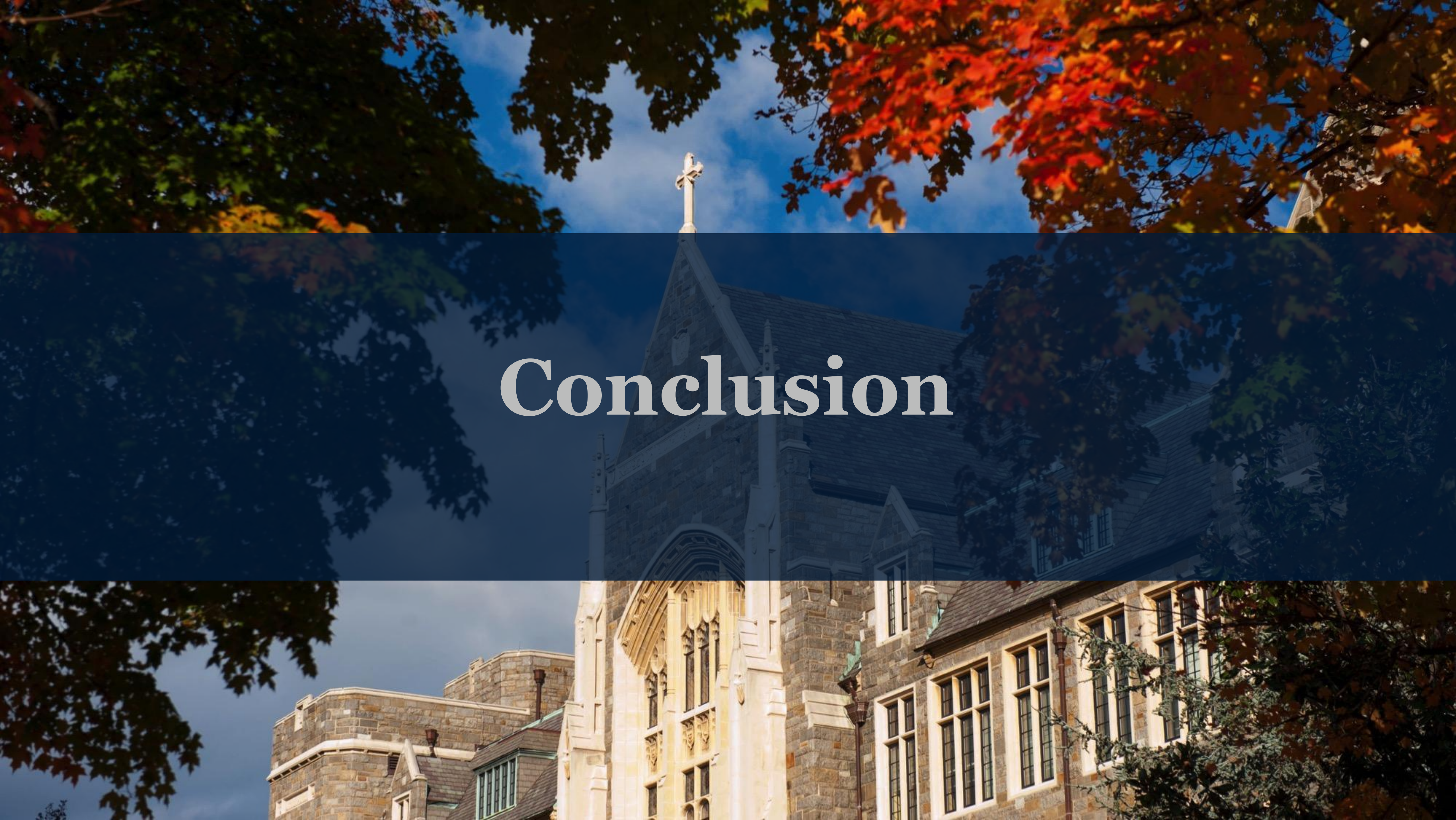
- Accuracy for cross validation is around 0.89 and the loss is around 0.37



- Accuracy for test set: 88.8%



```
                    precision    recall  f1-score   support

           news_tech       0.88      0.89      0.88      2759
  news_entertainment       0.90      0.91      0.91      3938
         news_sports       0.94      0.95      0.95      3765
            news_car       0.82      0.82      0.82      2716
           news_game       0.91      0.92      0.92      1782
        news_culture       0.93      0.93      0.93      3514
        news_finance       0.90      0.91      0.90      2615
            news_edu       0.88      0.87      0.87      4248
          news_world       0.89      0.86      0.87      2445
       news_military       0.85      0.83      0.84      2191
         news_travel       0.85      0.83      0.84      2706
    news_agriculture       0.87      0.89      0.88      1957
          news_house       0.91      0.93      0.92      2924
          news_story       0.33      0.07      0.12        42
               stock       0.80      0.78      0.79       662

            accuracy                           0.89     38264
           macro avg       0.84      0.83      0.83     38264
        weighted avg       0.89      0.89      0.89     38264
```



Conclusion

CONCLUSION

- Bert has significantly higher test accuracies among all the models for both all categories and the top 10 categories.
- HAN also has improved accuracies than LSTM and 1D-CNN.
- Note that BERT and HAN can take a lot more time to train!
- Why is the accuracy rate in Chinese higher?
  - Less categories/labels (40 compared to 15).
  - Chinese news headlines have more tokens due to Chinese text segmentation.

Thanks for watching!