In [ ]:
```
# Install TensorFlow
# !pip install -q tensorflow-gpu==2.0.0-beta1

try:
  %tensorflow_version 2.x  # Colab only.
except Exception:
  pass

import tensorflow as tf
print(tf.__version__)
```

```
`%tensorflow_version` only switches the major version: `1.x` or `2.x`.
You set: `2.x  # Colab only.`. This will be interpreted as: `2.x`.


TensorFlow 2.x selected.
2.0.0-beta1
```

In [ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split

from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.layers import Dense, Input, GlobalMaxPooling1D
from tensorflow.keras.layers import Conv1D, MaxPooling1D, Embedding
from tensorflow.keras.models import Model
```

In [ ]:
```
# Unfortunately this URL doesn't work directly with pd.read_csv
!wget -nc https://lazyprogrammer.me/course_files/spam.csv
```

```
File ‘spam.csv’ already there; not retrieving.
```

In [ ]:
```
df = pd.read_csv('spam.csv', encoding='ISO-8859-1')
```

In [ ]:
```
df.head()
```

Out[ ]:

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN
1	ham	Ok lar... Joking wif u oni...	NaN	NaN	NaN
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	NaN	NaN	NaN
3	ham	U dun say so early hor... U c already then say...	NaN	NaN	NaN
4	ham	Nah I don't think he goes to usf, he lives aro...	NaN	NaN	NaN

In [ ]:
```
# drop unnecessary columns
df = df.drop(["Unnamed: 2", "Unnamed: 3", "Unnamed: 4"], axis=1)
```

In [ ]:
```
df.head()
```

Out[ ]:

	v1	v2
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

In [ ]:
```
# rename columns to something better
df.columns = ['labels', 'data']
```

In [ ]:
```
df.head()
```

Out[ ]:

	labels	data
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

In [ ]:
```
# create binary labels
df['b_labels'] = df['labels'].map({'ham': 0, 'spam': 1})
Y = df['b_labels'].values
```

In [ ]:
```
# split up the data
df_train, df_test, Ytrain, Ytest = train_test_split(df['data'], Y, test_size=0.33)
```

In [ ]:
```
# Convert sentences to sequences
MAX_VOCAB_SIZE = 20000
tokenizer = Tokenizer(num_words=MAX_VOCAB_SIZE)
tokenizer.fit_on_texts(df_train)
sequences_train = tokenizer.texts_to_sequences(df_train)
sequences_test = tokenizer.texts_to_sequences(df_test)
```

In [ ]:
```
# get word -> integer mapping
word2idx = tokenizer.word_index
V = len(word2idx)
print('Found %s unique tokens.' % V)
```

```
Found 7257 unique tokens.
```

In [ ]:
```
# pad sequences so that we get a N x T matrix
data_train = pad_sequences(sequences_train)
print('Shape of data train tensor:', data_train.shape)

# get sequence length
T = data_train.shape[1]
```

```
Shape of data train tensor: (3733, 121)
```

In [ ]:
```
data_test = pad_sequences(sequences_test, maxlen=T)
print('Shape of data test tensor:', data_test.shape)
```

```
Shape of data test tensor: (1839, 121)
```

In [ ]:
```
# Create the model

# We get to choose embedding dimensionality
D = 20

# Note: we actually want to the size of the embedding to (V + 1) x D,
# because the first index starts from 1 and not 0.
# Thus, if the final index of the embedding matrix is V,
# then it actually must have size V + 1.

i = Input(shape=(T,))
x = Embedding(V + 1, D)(i)
x = Conv1D(32, 3, activation='relu')(x)
x = MaxPooling1D(3)(x)
x = Conv1D(64, 3, activation='relu')(x)
x = MaxPooling1D(3)(x)
x = Conv1D(128, 3, activation='relu')(x)
x = GlobalMaxPooling1D()(x)
x = Dense(1, activation='sigmoid')(x)

model = Model(i, x)
```

In [ ]:
```
# Compile and fit
model.compile(
  loss='binary_crossentropy',
  optimizer='adam',
  metrics=['accuracy']
)


print('Training model...')
r = model.fit(
  data_train,
  Ytrain,
  epochs=5,
  validation_data=(data_test, Ytest)
)
```

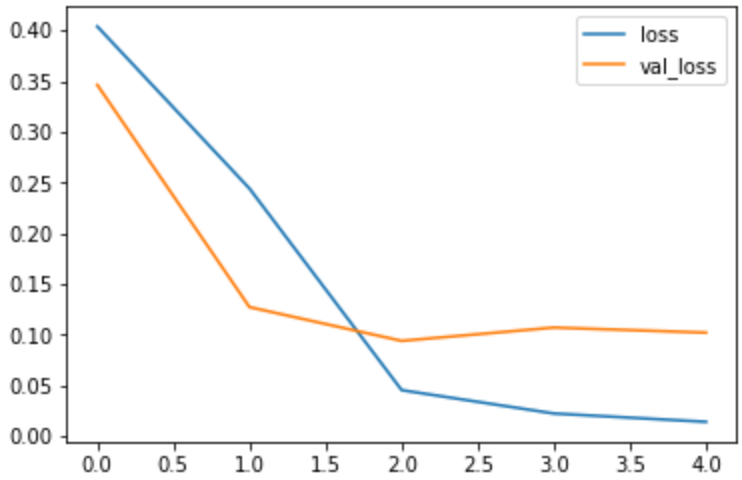
```
WARNING: Logging before flag parsing goes to stderr.
W0817 17:10:21.218855 140024925730688 deprecation.py:323] From /tensorflow-2.0.0b1/pytho
n3.6/tensorflow/python/ops/math_grad.py:1250: add_dispatch_support.<locals>.wrapper (fro
m tensorflow.python.ops.array_ops) is deprecated and will be removed in a future versio
```

```
n.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
Training model...
Train on 3733 samples, validate on 1839 samples
Epoch 1/5
3733/3733 [==============================] - 5s 1ms/sample - loss: 0.4038 - accuracy: 0.
8647 - val_loss: 0.3462 - val_accuracy: 0.8684
Epoch 2/5
3733/3733 [==============================] - 1s 334us/sample - loss: 0.2440 - accuracy:
0.8947 - val_loss: 0.1272 - val_accuracy: 0.9706
Epoch 3/5
3733/3733 [==============================] - 1s 330us/sample - loss: 0.0454 - accuracy:
0.9874 - val_loss: 0.0939 - val_accuracy: 0.9777
Epoch 4/5
3733/3733 [==============================] - 1s 331us/sample - loss: 0.0223 - accuracy:
0.9941 - val_loss: 0.1070 - val_accuracy: 0.9810
Epoch 5/5
3733/3733 [==============================] - 1s 331us/sample - loss: 0.0141 - accuracy:
0.9960 - val_loss: 0.1020 - val_accuracy: 0.9793
```

In [ ]:

```
# Plot loss per iteration
import matplotlib.pyplot as plt
plt.plot(r.history['loss'], label='loss')
plt.plot(r.history['val_loss'], label='val_loss')
plt.legend()
```

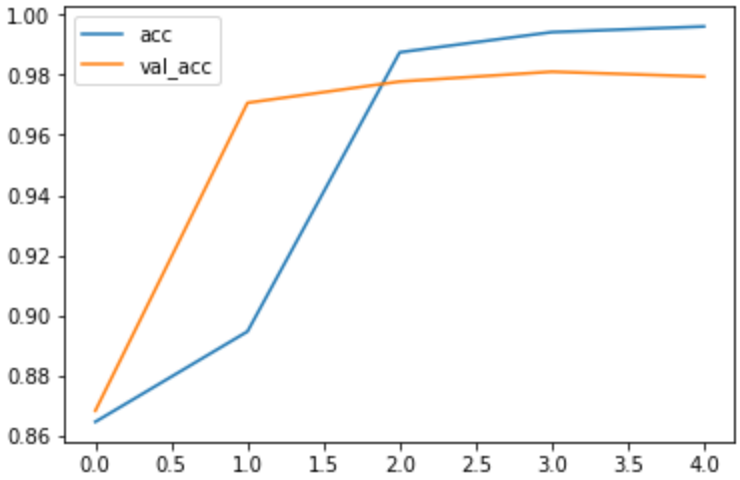
Out[ ]: <matplotlib.legend.Legend at 0x7f595ac7a9b0>



In [ ]:

```
# Plot accuracy per iteration
plt.plot(r.history['accuracy'], label='acc')
plt.plot(r.history['val_accuracy'], label='val_acc')
plt.legend()
```

Out[ ]: <matplotlib.legend.Legend at 0x7f5958430898>


acc
val_acc
1.00
0.98
0.96
0.94
0.92
0.90
0.88
0.86
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0