Downloading the data

```
!curl -O
https://ai.stanford.edu/~amaas/data/sentiment/aclImdb_v1.tar.gz
!tar -xf aclImdb_v1.tar.gz
!rm -r aclImdb/train/unsup
```

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time
Current
                                 Dload  Upload   Total   Spent    Left
Speed
100 80.2M  100 80.2M    0     0  5959k      0  0:00:13  0:00:13
--:--:-- 10.7M
```

Preparing the data

```
import os, pathlib, shutil, random
from tensorflow import keras
batch_size = 32
base_dir = pathlib.Path("aclImdb")
val_dir = base_dir / "val"
train_dir = base_dir / "train"
for category in ("neg", "pos"):
    os.makedirs(val_dir / category)
    files = os.listdir(train_dir / category)
    random.Random(1337).shuffle(files)
    num_val_samples = int(0.2 * len(files))
    val_files = files[-num_val_samples:]
    for fname in val_files:
        shutil.move(train_dir / category / fname,
                    val_dir / category / fname)
```

```
import os, pathlib, shutil, random
from tensorflow import keras

batch_size = 32
max_length = 150
max_tokens = 10000
num_train_samples = 100
num_val_samples = 10000

base_dir = pathlib.Path("aclImdb")
val_dir = base_dir / "val"
train_dir = base_dir / "train"

for category in ("neg", "pos"):
    os.makedirs(val_dir / category, exist_ok=True)
    files = os.listdir(train_dir / category)
    random.Random(1337).shuffle(files)
```

```
    num_val_samples_cat = int(num_val_samples/2)
    val_files = files[-num_val_samples_cat:]
    for fname in val_files:
        shutil.move(train_dir / category / fname,
                    val_dir / category / fname)
```

Training basic sequence model

```
train_ds = keras.preprocessing.text_dataset_from_directory(
    "aclImdb/train",
    batch_size=batch_size,
    validation_split=0.2,
    subset='training',
    seed=1337)

train_ds = train_ds.take(num_train_samples)

val_ds = keras.preprocessing.text_dataset_from_directory(
    "aclImdb/train",
    batch_size=batch_size,
    validation_split=0.2,
    subset='validation',
    seed=1337)

val_ds = val_ds.take(num_val_samples)

test_ds = keras.preprocessing.text_dataset_from_directory(
    "aclImdb/test", batch_size=batch_size)

text_only_train_ds = train_ds.map(lambda x, y: x)

from tensorflow.keras import layers

text_vectorization = layers.TextVectorization(
    max_tokens=max_tokens,
    output_mode="int",
    output_sequence_length=max_length,
)

text_vectorization.adapt(text_only_train_ds)

int_train_ds = train_ds.map(lambda x, y: (text_vectorization(x), y),
                            num_parallel_calls=4)

int_val_ds = val_ds.map(lambda x, y: (text_vectorization(x), y),
                        num_parallel_calls=4)

int_test_ds = test_ds.map(lambda x, y: (text_vectorization(x), y),
                          num_parallel_calls=4)
```

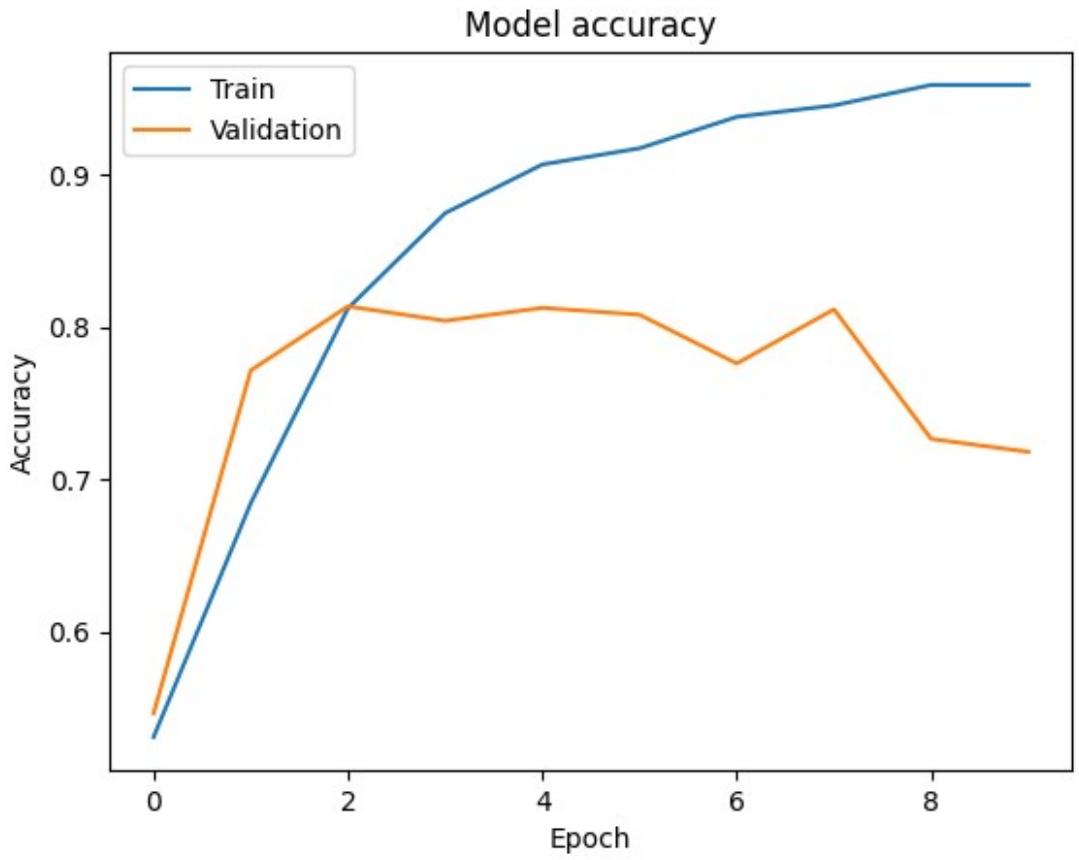
```
import tensorflow as tf

inputs = keras.Input(shape=(None,), dtype="int64")
embedded = tf.one_hot(inputs, depth=max_tokens)
x = layers.Bidirectional(layers.LSTM(32))(embedded)
x = layers.Dropout(0.5)(x)
outputs = layers.Dense(1, activation="sigmoid")(x)
model = keras.Model(inputs, outputs)

model.compile(
    optimizer="rmsprop",
    loss="binary_crossentropy",
    metrics=["accuracy"]
)

model.summary()

callbacks = [
    keras.callbacks.ModelCheckpoint("one_hot_bidir_lstm.x",
save_best_only=True)
]




history = model.fit(
    int_train_ds,
    validation_data=int_val_ds,
    epochs=10,
    callbacks=callbacks
)

model = keras.models.load_model("one_hot_bidir_lstm.x")
print(f"Test acc: {model.evaluate(int_test_ds)[1]:.3f}")
import matplotlib.pyplot as plt
```

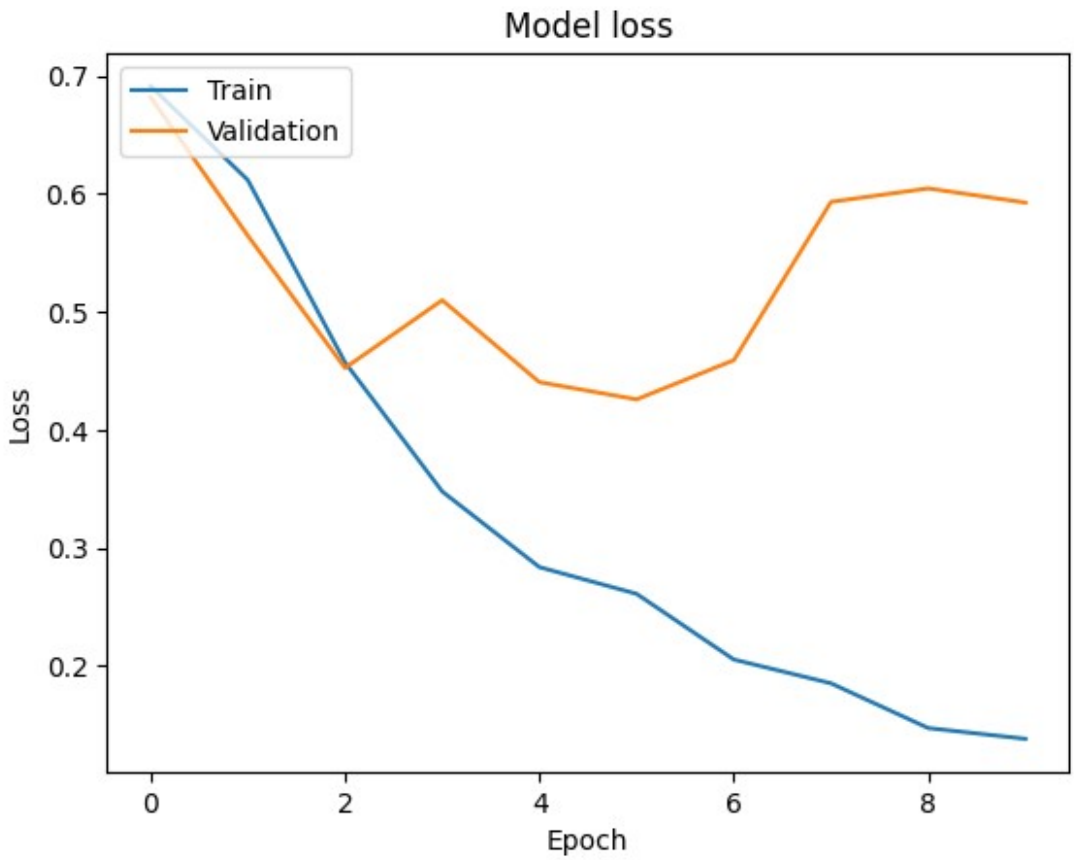
```
Found 10000 files belonging to 2 classes.
Using 8000 files for training.
Found 10000 files belonging to 2 classes.
Using 2000 files for validation.
Found 25000 files belonging to 2 classes.
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, None)]            0

 tf.one_hot (TFOpLambda)     (None, None, 10000)       0

 bidirectional (Bidirection  (None, 64)                2568448
```

```
 al)

 dropout (Dropout)           (None, 64)                0

 dense (Dense)               (None, 1)                 65

=================================================================
Total params: 2568513 (9.80 MB)
Trainable params: 2568513 (9.80 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
Epoch 1/10
100/100 [==============================] - 26s 158ms/step - loss:
0.6912 - accuracy: 0.5309 - val_loss: 0.6817 - val_accuracy: 0.5465
Epoch 2/10
100/100 [==============================] - 15s 150ms/step - loss:
0.6115 - accuracy: 0.6844 - val_loss: 0.5644 - val_accuracy: 0.7715
Epoch 3/10
100/100 [==============================] - 14s 144ms/step - loss:
0.4571 - accuracy: 0.8122 - val_loss: 0.4523 - val_accuracy: 0.8135
Epoch 4/10
100/100 [==============================] - 5s 48ms/step - loss: 0.3477
- accuracy: 0.8747 - val_loss: 0.5097 - val_accuracy: 0.8040
Epoch 5/10
100/100 [==============================] - 15s 148ms/step - loss:
0.2835 - accuracy: 0.9066 - val_loss: 0.4403 - val_accuracy: 0.8125
Epoch 6/10
100/100 [==============================] - 15s 150ms/step - loss:
0.2609 - accuracy: 0.9172 - val_loss: 0.4256 - val_accuracy: 0.8080
Epoch 7/10
100/100 [==============================] - 5s 49ms/step - loss: 0.2052
- accuracy: 0.9378 - val_loss: 0.4589 - val_accuracy: 0.7760
Epoch 8/10
100/100 [==============================] - 5s 50ms/step - loss: 0.1851
- accuracy: 0.9453 - val_loss: 0.5930 - val_accuracy: 0.8115
Epoch 9/10
100/100 [==============================] - 5s 48ms/step - loss: 0.1472
- accuracy: 0.9588 - val_loss: 0.6044 - val_accuracy: 0.7265
Epoch 10/10
100/100 [==============================] - 6s 64ms/step - loss: 0.1380
- accuracy: 0.9588 - val_loss: 0.5924 - val_accuracy: 0.7180
782/782 [==============================] - 17s 21ms/step - loss:
0.4374 - accuracy: 0.8109
Test acc: 0.811
```

```
# Plot training & validation accuracy values
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
```

```
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()

# Plot training & validation loss values
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()
```





Instantiating an Embedding layer

```
import matplotlib.pyplot as plt
```

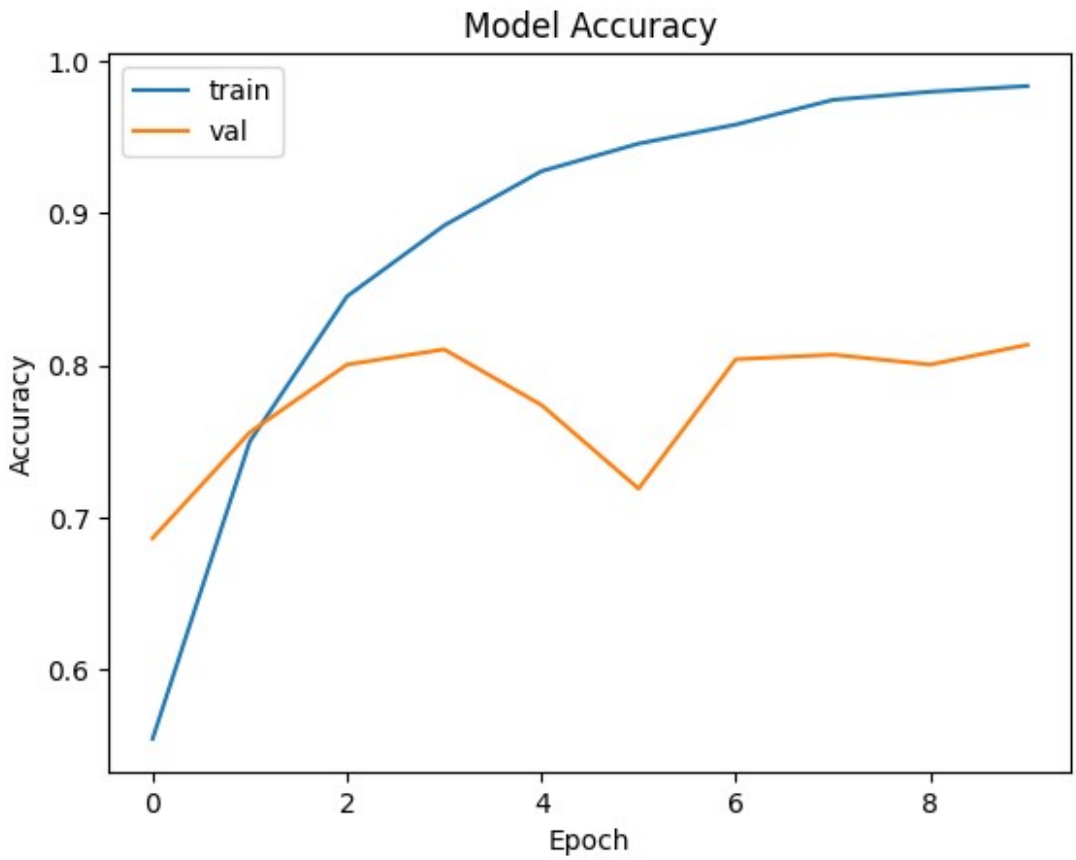
Model that uses an Embedding layer trained from scratch

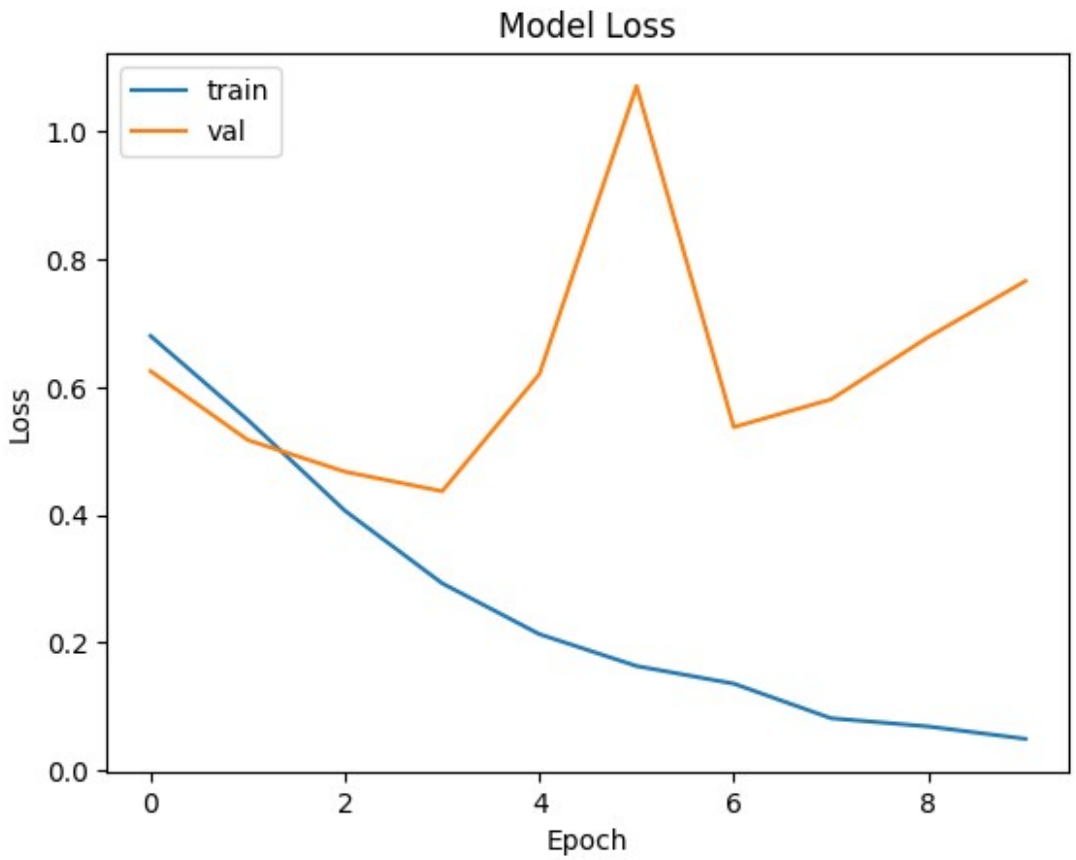
```
inputs = keras.Input(shape=(None,), dtype="int64")
embedded = layers.Embedding(input_dim=max_tokens, output_dim=256)
(inputs)
x = layers.Bidirectional(layers.LSTM(32))(embedded)
x = layers.Dropout(0.5)(x)
outputs = layers.Dense(1, activation="sigmoid")(x)
model = keras.Model(inputs, outputs)
model.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
model.summary()

callbacks = [
    keras.callbacks.ModelCheckpoint("embeddings_bidir_gru.x",
                                    save_best_only=True)
]
```

```
history = model.fit(int_train_ds, validation_data=int_val_ds,
epochs=10, callbacks=callbacks)

model = keras.models.load_model("embeddings_bidir_gru.x")
print(f"Test acc: {model.evaluate(int_test_ds)[1]:.3f}")

# Plot the training and validation accuracy
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()

# Plot the training and validation loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()
```

```
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, None)]            0

 embedding (Embedding)       (None, None, 256)         2560000

 bidirectional_1 (Bidirecti  (None, 64)                73984
 onal)

 dropout_1 (Dropout)         (None, 64)                0

 dense_1 (Dense)             (None, 1)                 65

=================================================================
Total params: 2634049 (10.05 MB)
Trainable params: 2634049 (10.05 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
Epoch 1/10
100/100 [==============================] - 22s 179ms/step - loss:
0.6799 - accuracy: 0.5547 - val_loss: 0.6248 - val_accuracy: 0.6865
Epoch 2/10
100/100 [==============================] - 16s 164ms/step - loss:
0.5486 - accuracy: 0.7500 - val_loss: 0.5172 - val_accuracy: 0.7560
```

```
Epoch 3/10
100/100 [==============================] - 14s 142ms/step - loss:
0.4065 - accuracy: 0.8453 - val_loss: 0.4675 - val_accuracy: 0.8005
Epoch 4/10
100/100 [==============================] - 14s 143ms/step - loss:
0.2931 - accuracy: 0.8919 - val_loss: 0.4371 - val_accuracy: 0.8105
Epoch 5/10
100/100 [==============================] - 3s 32ms/step - loss: 0.2135
- accuracy: 0.9275 - val_loss: 0.6202 - val_accuracy: 0.7740
Epoch 6/10
100/100 [==============================] - 3s 27ms/step - loss: 0.1637
- accuracy: 0.9456 - val_loss: 1.0707 - val_accuracy: 0.7190
Epoch 7/10
100/100 [==============================] - 3s 27ms/step - loss: 0.1361
- accuracy: 0.9581 - val_loss: 0.5373 - val_accuracy: 0.8040
Epoch 8/10
100/100 [==============================] - 3s 25ms/step - loss: 0.0820
- accuracy: 0.9744 - val_loss: 0.5807 - val_accuracy: 0.8070
Epoch 9/10
100/100 [==============================] - 4s 44ms/step - loss: 0.0692
- accuracy: 0.9797 - val_loss: 0.6781 - val_accuracy: 0.8005
Epoch 10/10
100/100 [==============================] - 2s 21ms/step - loss: 0.0498
- accuracy: 0.9834 - val_loss: 0.7657 - val_accuracy: 0.8135
782/782 [==============================] - 9s 10ms/step - loss: 0.4582
- accuracy: 0.7906
Test acc: 0.791
```


Model Accuracy
train
val
Accuracy
Epoch
1.0
0.9
0.8
0.7
0.6
0
2
4
6
8



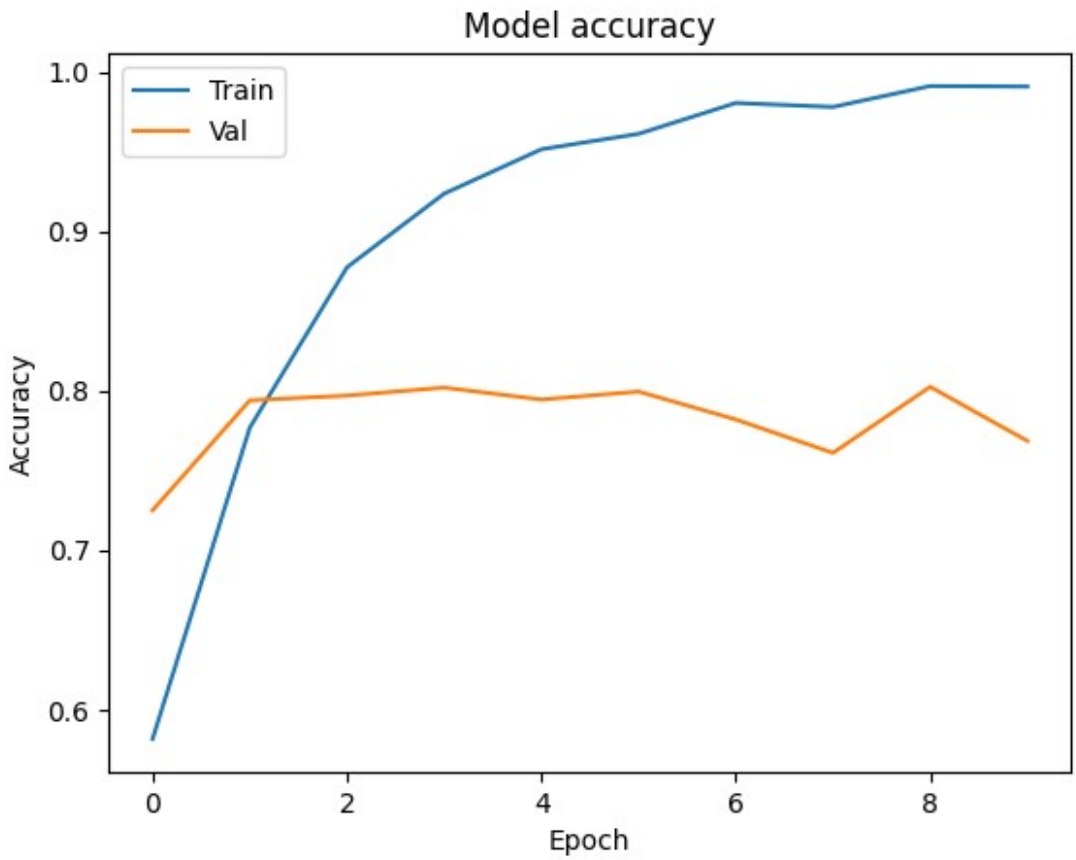
Model 3 - An Embedding Layer with Masking enabled

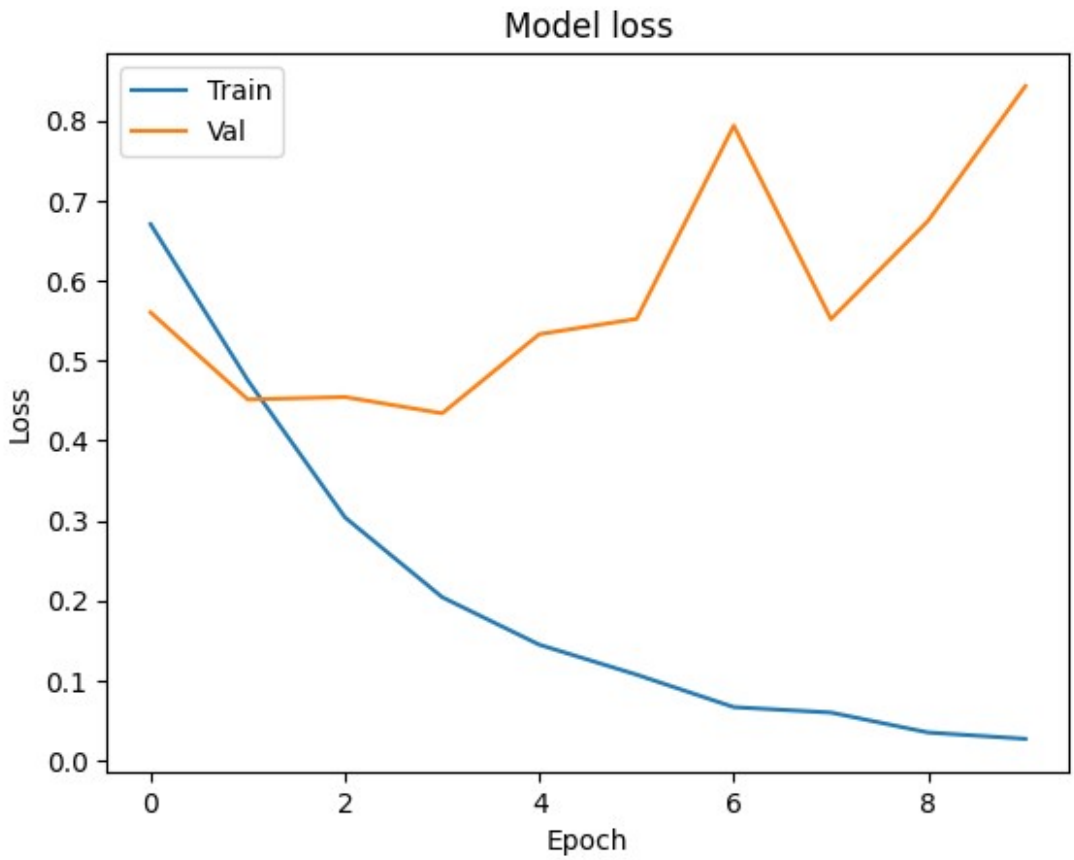
```
import matplotlib.pyplot as plt

# Define the model architecture
inputs = keras.Input(shape=(None,), dtype="int64")
embedded = layers.Embedding(
    input_dim=max_tokens, output_dim=256, mask_zero=True)(inputs)
x = layers.Bidirectional(layers.LSTM(64))(embedded)
x = layers.Dropout(0.3)(x)
outputs = layers.Dense(1, activation="sigmoid")(x)
model = keras.Model(inputs, outputs)

# Compile the model
model.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
model.summary()

# Define the callbacks
callbacks = [

keras.callbacks.ModelCheckpoint("embeddings_bidir_gru_with_masking.x",
```

```
                                    save_best_only=True),
    keras.callbacks.History()
]

# Train the model
history = model.fit(int_train_ds, validation_data=int_val_ds,
epochs=10, callbacks=callbacks)

# Load the best model
model = keras.models.load_model("embeddings_bidir_gru_with_masking.x")
print(f"Test acc: {model.evaluate(int_test_ds)[1]:.3f}")

# Plot the training and validation accuracy and loss
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='upper left')
plt.show()

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='upper left')
plt.show()
```

```
Model: "model_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_3 (InputLayer)        [(None, None)]            0

 embedding_1 (Embedding)     (None, None, 256)         2560000

 bidirectional_2 (Bidirecti  (None, 128)               164352
 onal)

 dropout_2 (Dropout)         (None, 128)               0

 dense_2 (Dense)             (None, 1)                 129

=================================================================
Total params: 2724481 (10.39 MB)
Trainable params: 2724481 (10.39 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
Epoch 1/10
```

```
100/100 [==============================] - 37s 295ms/step - loss:
0.6705 - accuracy: 0.5816 - val_loss: 0.5601 - val_accuracy: 0.7250
Epoch 2/10
100/100 [==============================] - 23s 236ms/step - loss:
0.4753 - accuracy: 0.7769 - val_loss: 0.4514 - val_accuracy: 0.7940
Epoch 3/10
100/100 [==============================] - 6s 59ms/step - loss: 0.3040
- accuracy: 0.8775 - val_loss: 0.4544 - val_accuracy: 0.7970
Epoch 4/10
100/100 [==============================] - 21s 213ms/step - loss:
0.2043 - accuracy: 0.9237 - val_loss: 0.4341 - val_accuracy: 0.8020
Epoch 5/10
100/100 [==============================] - 4s 43ms/step - loss: 0.1449
- accuracy: 0.9516 - val_loss: 0.5330 - val_accuracy: 0.7945
Epoch 6/10
100/100 [==============================] - 3s 30ms/step - loss: 0.1075
- accuracy: 0.9613 - val_loss: 0.5520 - val_accuracy: 0.7995
Epoch 7/10
100/100 [==============================] - 4s 41ms/step - loss: 0.0669
- accuracy: 0.9806 - val_loss: 0.7936 - val_accuracy: 0.7820
Epoch 8/10
100/100 [==============================] - 3s 30ms/step - loss: 0.0601
- accuracy: 0.9781 - val_loss: 0.5518 - val_accuracy: 0.7610
Epoch 9/10
100/100 [==============================] - 3s 30ms/step - loss: 0.0352
- accuracy: 0.9912 - val_loss: 0.6747 - val_accuracy: 0.8025
Epoch 10/10
100/100 [==============================] - 5s 45ms/step - loss: 0.0273
- accuracy: 0.9909 - val_loss: 0.8431 - val_accuracy: 0.7685
782/782 [==============================] - 12s 11ms/step - loss:
0.4604 - accuracy: 0.7908
Test acc: 0.791
```


Model accuracy
Train
Val
1.0
0.9
0.8
0.7
0.6
Accuracy
0
2
4
6
8
Epoch



Model 4 - Using Pretrained word embedding

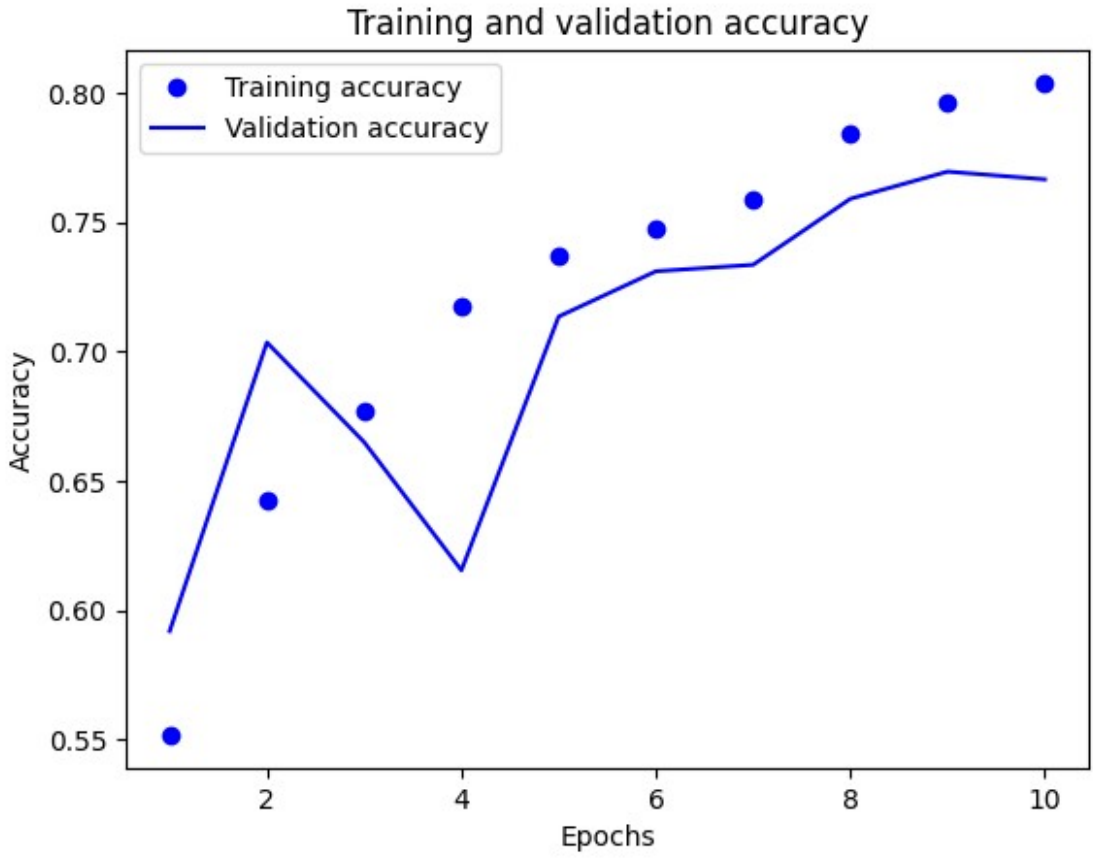
```
#!wget http://nlp.stanford.edu/data/glove.6B.zip
#!unzip -q glove.6B.zip

import numpy as np
path_to_glove_file = "/content/glove.6B.100d.txt"

embeddings_index = {}
with open(path_to_glove_file) as f:
    for line in f:
        word, coefs = line.split(maxsplit=1)
        coefs = np.fromstring(coefs, "f", sep=" ")
        embeddings_index[word] = coefs

print(f"Found {len(embeddings_index)} word vectors.")

embedding_dim = 100
max_tokens = 10000
max_len = 150
num_samples = 100
validation_samples = 10000
```

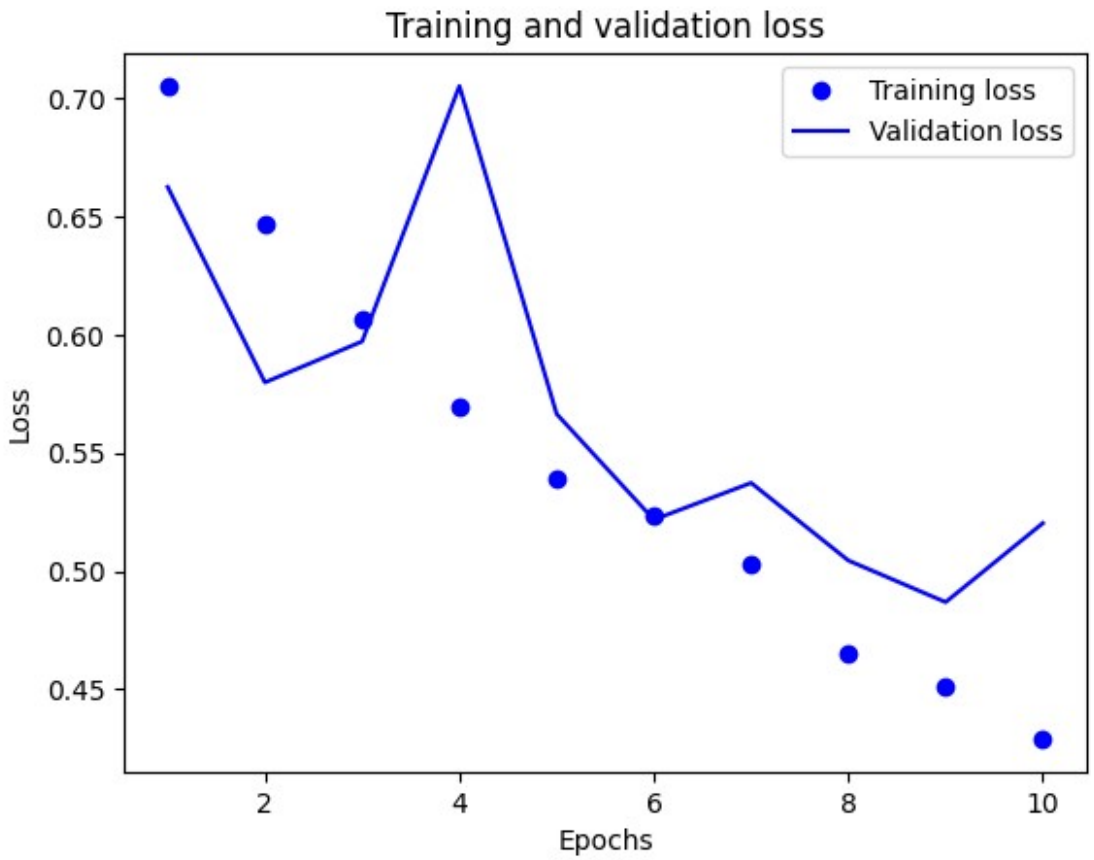
```
vocabulary = text_vectorization.get_vocabulary()
word_index = dict(zip(vocabulary, range(len(vocabulary))))
word_index = {k: v for k, v in word_index.items() if v < max_tokens}

embedding_matrix = np.zeros((max_tokens, embedding_dim))
for word, i in word_index.items():
    if i < max_tokens:
        embedding_vector = embeddings_index.get(word)
    if embedding_vector is not None:
        embedding_matrix[i] = embedding_vector

embedding_layer = layers.Embedding(
    max_tokens,
    embedding_dim,

embeddings_initializer=keras.initializers.Constant(embedding_matrix),
    trainable=False,
    mask_zero=True,
)

inputs = keras.Input(shape=(None,), dtype="int64")
embedded = embedding_layer(inputs)
x = layers.Bidirectional(layers.LSTM(128))(embedded)
x = layers.Dropout(0.8)(x)
outputs = layers.Dense(1, activation="sigmoid")(x)
model = keras.Model(inputs, outputs)
model.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
model.summary()

callbacks = [

keras.callbacks.ModelCheckpoint("glove_embeddings_sequence_model.x",
                                    save_best_only=True)
]


history = model.fit(int_train_ds.take(num_samples).cache(),
          validation_data=int_val_ds.take(validation_samples).cache(),
          epochs=10, callbacks=callbacks)

model = keras.models.load_model("glove_embeddings_sequence_model.x")
_, test_acc = model.evaluate(int_test_ds.take(validation_samples))
print(f"Test acc: {test_acc:.3f}")

# Plot training and validation accuracy
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
epochs = range(1, len(acc) + 1)
```

```
plt.plot(epochs, acc, 'bo', label='Training accuracy')
plt.plot(epochs, val_acc, 'b', label='Validation accuracy')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()

# Plot training and validation loss
loss = history.history['loss']
val_loss = history.history['val_loss']
plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

```
Found 400000 word vectors.
Model: "model_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_4 (InputLayer)        [(None, None)]            0

 embedding_2 (Embedding)     (None, None, 100)         1000000

 bidirectional_3 (Bidirecti  (None, 256)               234496
 onal)

 dropout_3 (Dropout)         (None, 256)               0

 dense_3 (Dense)             (None, 1)                 257

=================================================================
Total params: 1234753 (4.71 MB)
Trainable params: 234753 (917.00 KB)
Non-trainable params: 1000000 (3.81 MB)
_________________________________________________________________
Epoch 1/10
100/100 [==============================] - 36s 281ms/step - loss:
0.7051 - accuracy: 0.5519 - val_loss: 0.6626 - val_accuracy: 0.5920
Epoch 2/10
100/100 [==============================] - 26s 262ms/step - loss:
0.6469 - accuracy: 0.6428 - val_loss: 0.5799 - val_accuracy: 0.7035
Epoch 3/10
100/100 [==============================] - 2s 25ms/step - loss: 0.6062
- accuracy: 0.6772 - val_loss: 0.5972 - val_accuracy: 0.6650
Epoch 4/10
```

```
100/100 [==============================] - 3s 34ms/step - loss: 0.5697
- accuracy: 0.7172 - val_loss: 0.7053 - val_accuracy: 0.6155
Epoch 5/10
100/100 [==============================] - 25s 250ms/step - loss:
0.5395 - accuracy: 0.7372 - val_loss: 0.5665 - val_accuracy: 0.7135
Epoch 6/10
100/100 [==============================] - 26s 262ms/step - loss:
0.5238 - accuracy: 0.7475 - val_loss: 0.5217 - val_accuracy: 0.7310
Epoch 7/10
100/100 [==============================] - 2s 22ms/step - loss: 0.5031
- accuracy: 0.7584 - val_loss: 0.5373 - val_accuracy: 0.7335
Epoch 8/10
100/100 [==============================] - 26s 260ms/step - loss:
0.4652 - accuracy: 0.7844 - val_loss: 0.5045 - val_accuracy: 0.7590
Epoch 9/10
100/100 [==============================] - 27s 272ms/step - loss:
0.4513 - accuracy: 0.7962 - val_loss: 0.4868 - val_accuracy: 0.7695
Epoch 10/10
100/100 [==============================] - 2s 23ms/step - loss: 0.4290
- accuracy: 0.8037 - val_loss: 0.5203 - val_accuracy: 0.7665
782/782 [==============================] - 11s 12ms/step - loss:
0.4966 - accuracy: 0.7564
Test acc: 0.756
```





Using different training samples for the embedding layer

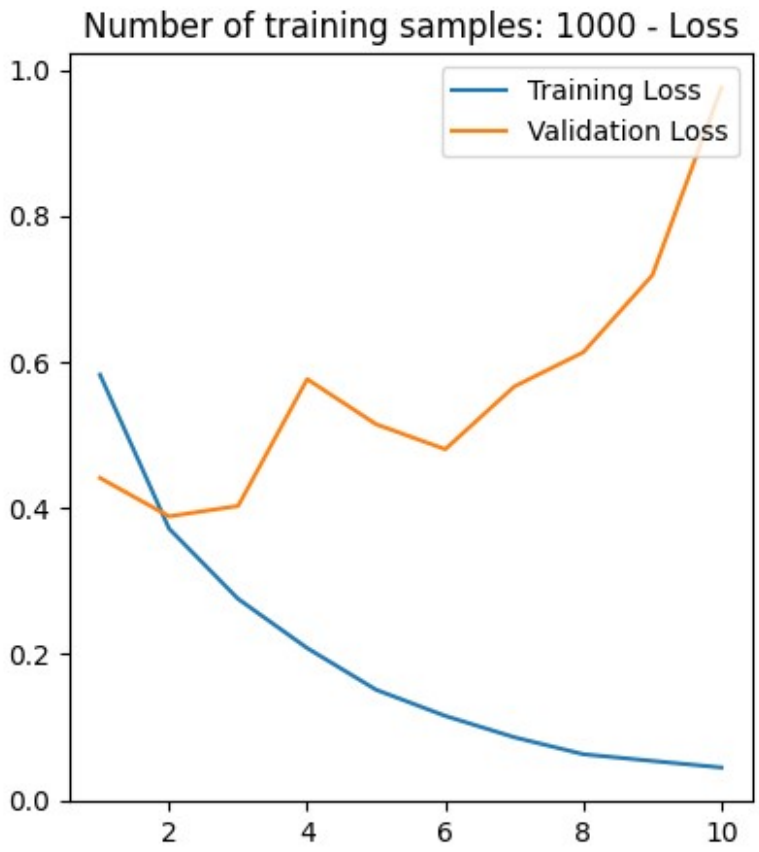
```
import os, pathlib, shutil, random
import matplotlib.pyplot as plt
from tensorflow import keras
from tensorflow.keras import layers

batch_size = 32
max_length = 150
max_tokens = 10000
num_val_samples = 10000

base_dir = pathlib.Path("/content/aclImdb")
val_dir = base_dir / "val"
train_dir = base_dir / "train"

for category in ("neg", "pos"):
    os.makedirs(val_dir / category, exist_ok=True)
    files = os.listdir(train_dir / category)
    random.Random(1337).shuffle(files)
    num_val_samples_cat = int(num_val_samples/2)
    val_files = files[-num_val_samples_cat:]
    for fname in val_files:
```

```
        shutil.move(train_dir / category / fname,
                    val_dir / category / fname)

test_ds = keras.preprocessing.text_dataset_from_directory(
    "aclImdb/test", batch_size=batch_size)

text_only_train_ds = keras.preprocessing.text_dataset_from_directory(
    "aclImdb/train",
    batch_size=batch_size,
    validation_split=0.2,
    subset='training',
    seed=1337).map(lambda x, y: x)

text_vectorization = layers.TextVectorization(
    max_tokens=max_tokens,
    output_mode="int",
    output_sequence_length=max_length,
)

text_vectorization.adapt(text_only_train_ds)

for num_train_samples in [1000, 5000, 10000, 15000, 20000, 25000]:
    train_ds = keras.preprocessing.text_dataset_from_directory(
        "aclImdb/train",
        batch_size=batch_size,
        validation_split=0.2,
        subset='training',
        seed=1337
    ).take(num_train_samples)

    val_ds = keras.preprocessing.text_dataset_from_directory(
        "aclImdb/train",
        batch_size=batch_size,
        validation_split=0.2,
        subset='validation',
        seed=1337
    ).take(num_val_samples)

    int_train_ds = train_ds.map(lambda x, y: (text_vectorization(x),
y), num_parallel_calls=4)
    int_val_ds = val_ds.map(lambda x, y: (text_vectorization(x), y),
num_parallel_calls=4)
    int_test_ds = test_ds.map(lambda x, y: (text_vectorization(x), y),
num_parallel_calls=4)

    inputs = keras.Input(shape=(None,), dtype="int64")
    embedded = layers.Embedding(
        input_dim=max_tokens, output_dim=256, mask_zero=True)(inputs)
    x = layers.Bidirectional(layers.LSTM(32))(embedded)
    x = layers.Dropout(0.5)(x)
```

```
    outputs = layers.Dense(1, activation="sigmoid")(x)
    model = keras.Model(inputs, outputs)
    model.compile(optimizer="rmsprop",
                  loss="binary_crossentropy",
                  metrics=["accuracy"])

    callbacks = [

keras.callbacks.ModelCheckpoint(f"embeddings_bidir_gru_with_masking_{n
um_train_samples}.x",
                                        save_best_only=True)
    ]
    history = model.fit(int_train_ds, validation_data=int_val_ds,
epochs=10, callbacks=callbacks)

    model =
keras.models.load_model(f"embeddings_bidir_gru_with_masking_{num_train
_samples}.x")
    test_acc = model.evaluate(int_test_ds)[1]
    print(f"Number of training samples: {num_train_samples} - Test
accuracy: {test_acc:.3f}")



    # Plot the training and validation loss
    loss = history.history['loss']
    val_loss = history.history['val_loss']
    epochs_range = range(1, len(acc) + 1)

    plt.figure(figsize=(10, 5))
    plt.subplot(1, 2, 1)
    plt.plot(epochs_range, loss, label='Training Loss')
    plt.plot(epochs_range, val_loss, label='Validation Loss')
    plt.legend(loc='upper right')
    plt.title(f'Number of training samples: {num_train_samples} -
Loss')

    plt.show()

#Plot the training and validation accuracies
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
epochs_range = range(1, len(acc) + 1)
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 2)
plt.plot(epochs_range, acc, label='Training accuracy')
plt.plot(epochs_range, val_acc, label='Validation accuracy')
plt.legend(loc='lower right')
plt.title(f'Number of training samples: {num_train_samples} -
```

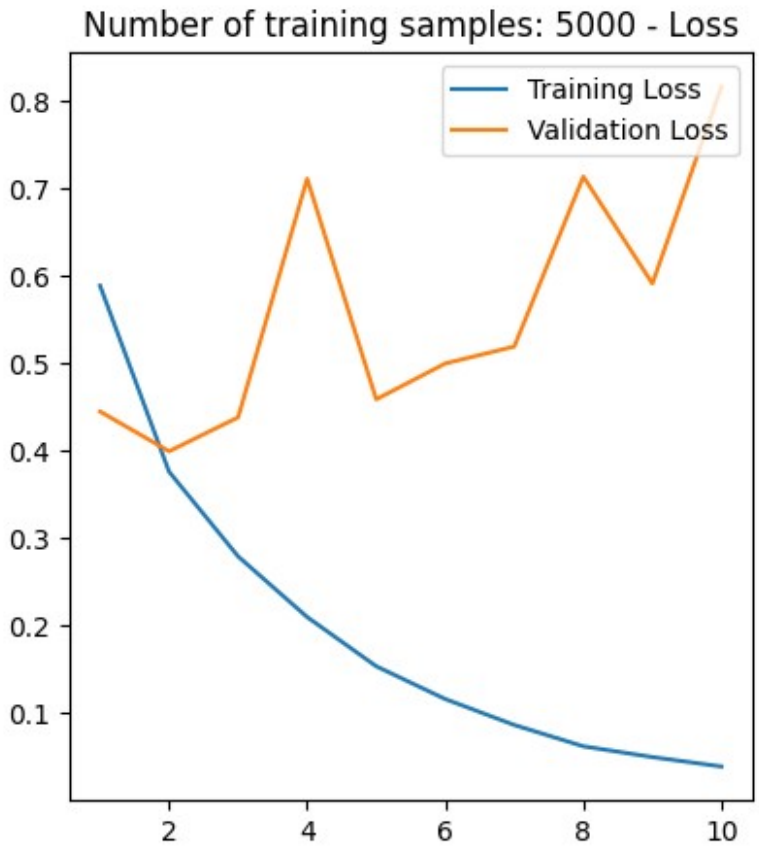
```
Accuracy')
plt.show()
```

```
Found 25000 files belonging to 2 classes.
Found 10000 files belonging to 2 classes.
Using 8000 files for training.
Found 10000 files belonging to 2 classes.
Using 8000 files for training.
Found 10000 files belonging to 2 classes.
Using 2000 files for validation.
Epoch 1/10
250/250 [==============================] - 44s 143ms/step - loss:
0.5825 - accuracy: 0.6833 - val_loss: 0.4412 - val_accuracy: 0.8115
Epoch 2/10
250/250 [==============================] - 28s 111ms/step - loss:
0.3718 - accuracy: 0.8407 - val_loss: 0.3886 - val_accuracy: 0.8295
Epoch 3/10
250/250 [==============================] - 7s 28ms/step - loss: 0.2756
- accuracy: 0.8891 - val_loss: 0.4028 - val_accuracy: 0.8385
Epoch 4/10
250/250 [==============================] - 8s 31ms/step - loss: 0.2082
- accuracy: 0.9231 - val_loss: 0.5765 - val_accuracy: 0.7870
Epoch 5/10
250/250 [==============================] - 7s 29ms/step - loss: 0.1511
- accuracy: 0.9449 - val_loss: 0.5147 - val_accuracy: 0.8140
Epoch 6/10
250/250 [==============================] - 6s 24ms/step - loss: 0.1153
- accuracy: 0.9615 - val_loss: 0.4805 - val_accuracy: 0.8280
Epoch 7/10
250/250 [==============================] - 8s 31ms/step - loss: 0.0861
- accuracy: 0.9704 - val_loss: 0.5665 - val_accuracy: 0.8155
Epoch 8/10
250/250 [==============================] - 6s 22ms/step - loss: 0.0628
- accuracy: 0.9790 - val_loss: 0.6134 - val_accuracy: 0.8185
Epoch 9/10
250/250 [==============================] - 8s 32ms/step - loss: 0.0537
- accuracy: 0.9820 - val_loss: 0.7189 - val_accuracy: 0.8270
Epoch 10/10
250/250 [==============================] - 6s 23ms/step - loss: 0.0447
- accuracy: 0.9865 - val_loss: 0.9757 - val_accuracy: 0.7970
782/782 [==============================] - 12s 12ms/step - loss:
0.3985 - accuracy: 0.8212
Number of training samples: 1000 - Test accuracy: 0.821
```



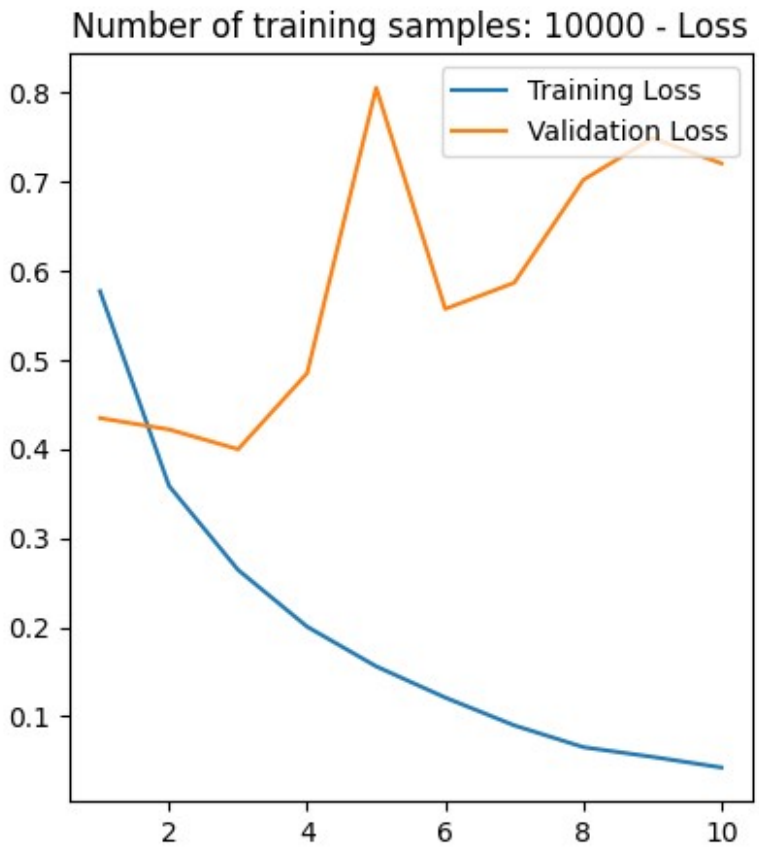

```
Found 10000 files belonging to 2 classes.
Using 8000 files for training.
Found 10000 files belonging to 2 classes.
Using 2000 files for validation.
Epoch 1/10
250/250 [==============================] - 45s 147ms/step - loss:
0.5887 - accuracy: 0.6783 - val_loss: 0.4450 - val_accuracy: 0.7950
Epoch 2/10
250/250 [==============================] - 28s 113ms/step - loss:
0.3761 - accuracy: 0.8407 - val_loss: 0.3994 - val_accuracy: 0.8200
Epoch 3/10
250/250 [==============================] - 8s 31ms/step - loss: 0.2793
- accuracy: 0.8911 - val_loss: 0.4382 - val_accuracy: 0.8155
Epoch 4/10
250/250 [==============================] - 8s 31ms/step - loss: 0.2100
- accuracy: 0.9235 - val_loss: 0.7110 - val_accuracy: 0.7695
Epoch 5/10
250/250 [==============================] - 6s 25ms/step - loss: 0.1538
- accuracy: 0.9481 - val_loss: 0.4588 - val_accuracy: 0.8260
Epoch 6/10
250/250 [==============================] - 8s 33ms/step - loss: 0.1163
- accuracy: 0.9616 - val_loss: 0.4998 - val_accuracy: 0.8220
Epoch 7/10
```

```
250/250 [==============================] - 6s 24ms/step - loss: 0.0866
- accuracy: 0.9711 - val_loss: 0.5192 - val_accuracy: 0.8235
Epoch 8/10
250/250 [==============================] - 6s 23ms/step - loss: 0.0623
- accuracy: 0.9796 - val_loss: 0.7134 - val_accuracy: 0.8135
Epoch 9/10
250/250 [==============================] - 7s 29ms/step - loss: 0.0500
- accuracy: 0.9844 - val_loss: 0.5911 - val_accuracy: 0.8270
Epoch 10/10
250/250 [==============================] - 8s 30ms/step - loss: 0.0392
- accuracy: 0.9886 - val_loss: 0.8162 - val_accuracy: 0.8240
782/782 [==============================] - 12s 11ms/step - loss:
0.4014 - accuracy: 0.8194
Number of training samples: 5000 - Test accuracy: 0.819
```



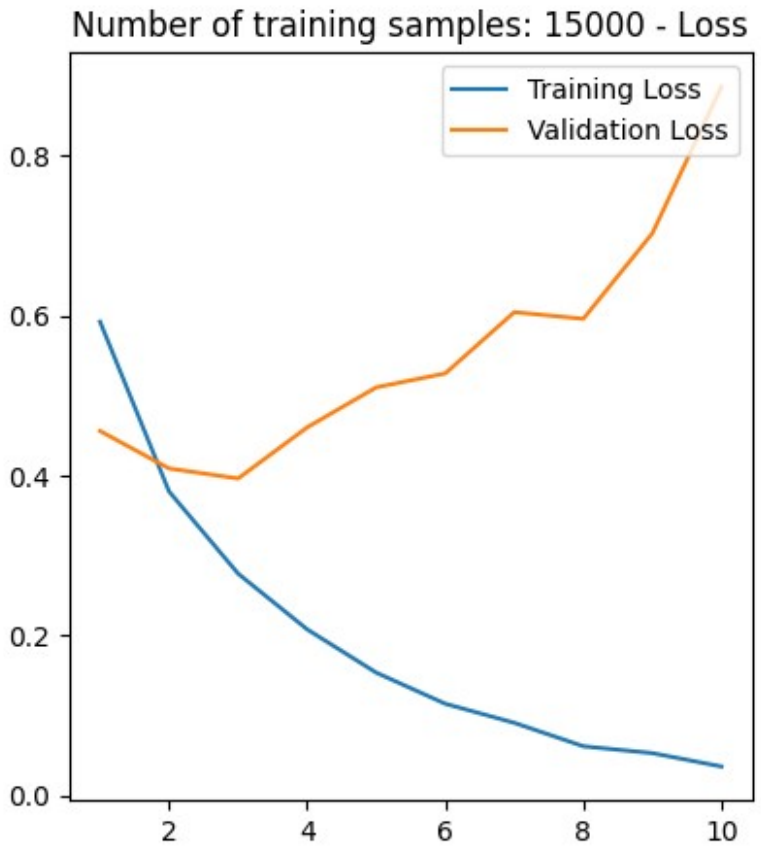

```
Found 10000 files belonging to 2 classes.
Using 8000 files for training.
Found 10000 files belonging to 2 classes.
Using 2000 files for validation.
Epoch 1/10
250/250 [==============================] - 45s 144ms/step - loss:
0.5773 - accuracy: 0.6895 - val_loss: 0.4348 - val_accuracy: 0.8070
```

```
Epoch 2/10
250/250 [==============================] - 28s 111ms/step - loss:
0.3586 - accuracy: 0.8506 - val_loss: 0.4220 - val_accuracy: 0.8070
Epoch 3/10
250/250 [==============================] - 25s 102ms/step - loss:
0.2643 - accuracy: 0.8950 - val_loss: 0.3999 - val_accuracy: 0.8330
Epoch 4/10
250/250 [==============================] - 6s 25ms/step - loss: 0.2006
- accuracy: 0.9290 - val_loss: 0.4853 - val_accuracy: 0.7950
Epoch 5/10
250/250 [==============================] - 8s 32ms/step - loss: 0.1561
- accuracy: 0.9455 - val_loss: 0.8058 - val_accuracy: 0.7710
Epoch 6/10
250/250 [==============================] - 6s 24ms/step - loss: 0.1210
- accuracy: 0.9574 - val_loss: 0.5573 - val_accuracy: 0.8225
Epoch 7/10
250/250 [==============================] - 8s 30ms/step - loss: 0.0898
- accuracy: 0.9700 - val_loss: 0.5868 - val_accuracy: 0.8170
Epoch 8/10
250/250 [==============================] - 6s 23ms/step - loss: 0.0653
- accuracy: 0.9786 - val_loss: 0.7023 - val_accuracy: 0.8080
Epoch 9/10
250/250 [==============================] - 6s 23ms/step - loss: 0.0544
- accuracy: 0.9830 - val_loss: 0.7492 - val_accuracy: 0.8290
Epoch 10/10
250/250 [==============================] - 7s 28ms/step - loss: 0.0425
- accuracy: 0.9884 - val_loss: 0.7208 - val_accuracy: 0.8280
782/782 [==============================] - 11s 12ms/step - loss:
0.4183 - accuracy: 0.8237
Number of training samples: 10000 - Test accuracy: 0.824
```



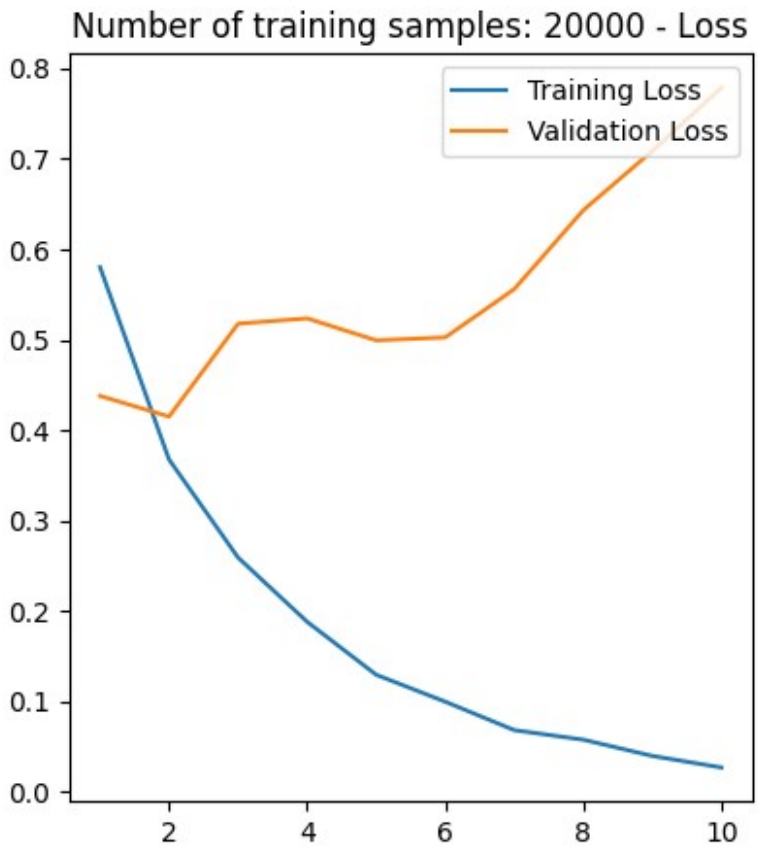

```
Found 10000 files belonging to 2 classes.
Using 8000 files for training.
Found 10000 files belonging to 2 classes.
Using 2000 files for validation.
Epoch 1/10
250/250 [==============================] - 45s 149ms/step - loss:
0.5927 - accuracy: 0.6661 - val_loss: 0.4560 - val_accuracy: 0.8055
Epoch 2/10
250/250 [==============================] - 28s 114ms/step - loss:
0.3804 - accuracy: 0.8418 - val_loss: 0.4092 - val_accuracy: 0.8235
Epoch 3/10
250/250 [==============================] - 27s 107ms/step - loss:
0.2777 - accuracy: 0.8926 - val_loss: 0.3966 - val_accuracy: 0.8260
Epoch 4/10
250/250 [==============================] - 8s 33ms/step - loss: 0.2080
- accuracy: 0.9250 - val_loss: 0.4605 - val_accuracy: 0.8060
Epoch 5/10
250/250 [==============================] - 6s 25ms/step - loss: 0.1539
- accuracy: 0.9444 - val_loss: 0.5105 - val_accuracy: 0.8145
Epoch 6/10
250/250 [==============================] - 8s 31ms/step - loss: 0.1150
- accuracy: 0.9592 - val_loss: 0.5280 - val_accuracy: 0.8165
Epoch 7/10
```

```
250/250 [==============================] - 6s 24ms/step - loss: 0.0912
- accuracy: 0.9684 - val_loss: 0.6046 - val_accuracy: 0.8155
Epoch 8/10
250/250 [==============================] - 8s 31ms/step - loss: 0.0619
- accuracy: 0.9779 - val_loss: 0.5962 - val_accuracy: 0.8190
Epoch 9/10
250/250 [==============================] - 6s 23ms/step - loss: 0.0533
- accuracy: 0.9834 - val_loss: 0.7032 - val_accuracy: 0.8290
Epoch 10/10
250/250 [==============================] - 6s 23ms/step - loss: 0.0366
- accuracy: 0.9891 - val_loss: 0.8864 - val_accuracy: 0.8050
782/782 [==============================] - 13s 12ms/step - loss:
0.4076 - accuracy: 0.8211
Number of training samples: 15000 - Test accuracy: 0.821
```



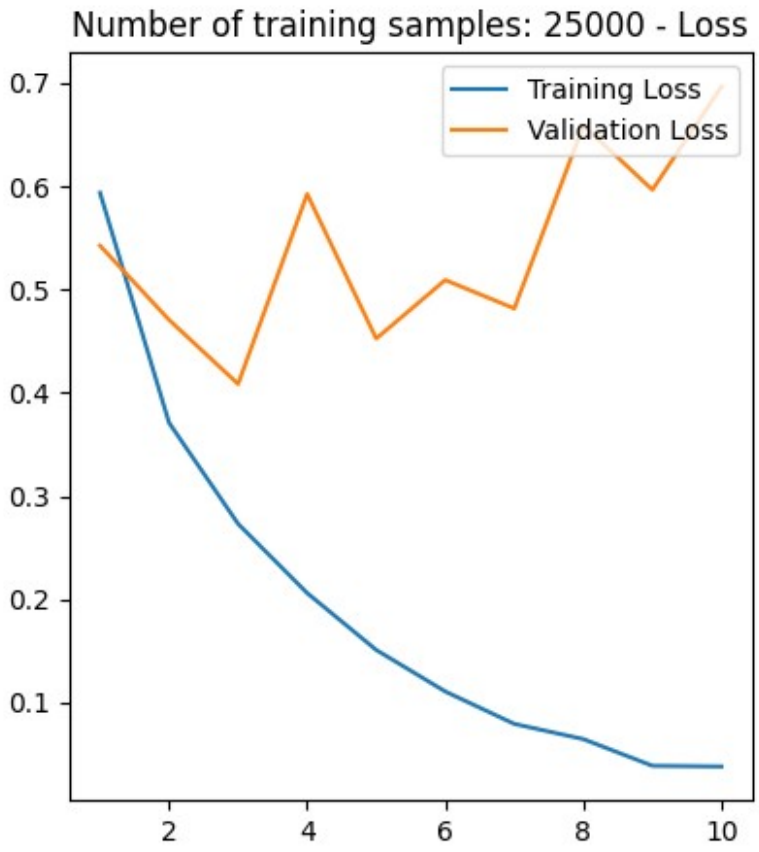

```
Found 10000 files belonging to 2 classes.
Using 8000 files for training.
Found 10000 files belonging to 2 classes.
Using 2000 files for validation.
Epoch 1/10
250/250 [==============================] - 44s 144ms/step - loss:
0.5803 - accuracy: 0.6844 - val_loss: 0.4380 - val_accuracy: 0.8050
```

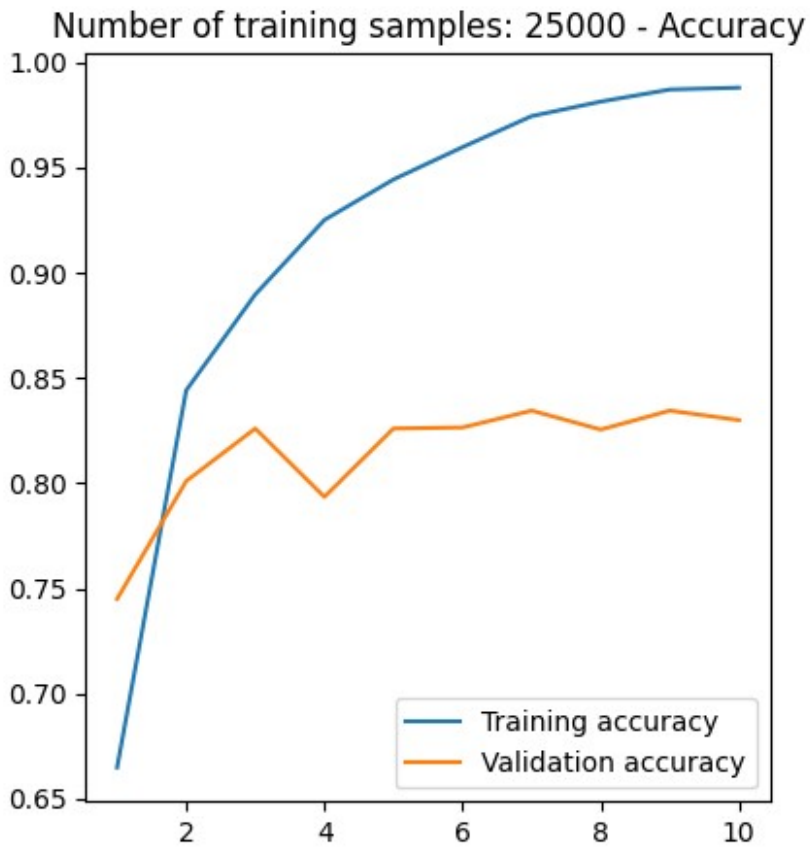
```
Epoch 2/10
250/250 [==============================] - 29s 115ms/step - loss:
0.3676 - accuracy: 0.8480 - val_loss: 0.4152 - val_accuracy: 0.8145
Epoch 3/10
250/250 [==============================] - 8s 33ms/step - loss: 0.2592
- accuracy: 0.9003 - val_loss: 0.5178 - val_accuracy: 0.8245
Epoch 4/10
250/250 [==============================] - 7s 28ms/step - loss: 0.1883
- accuracy: 0.9325 - val_loss: 0.5237 - val_accuracy: 0.8000
Epoch 5/10
250/250 [==============================] - 6s 25ms/step - loss: 0.1297
- accuracy: 0.9532 - val_loss: 0.4994 - val_accuracy: 0.8065
Epoch 6/10
250/250 [==============================] - 6s 24ms/step - loss: 0.0999
- accuracy: 0.9668 - val_loss: 0.5027 - val_accuracy: 0.8215
Epoch 7/10
250/250 [==============================] - 8s 31ms/step - loss: 0.0685
- accuracy: 0.9758 - val_loss: 0.5561 - val_accuracy: 0.8125
Epoch 8/10
250/250 [==============================] - 6s 23ms/step - loss: 0.0579
- accuracy: 0.9827 - val_loss: 0.6433 - val_accuracy: 0.8320
Epoch 9/10
250/250 [==============================] - 7s 26ms/step - loss: 0.0400
- accuracy: 0.9879 - val_loss: 0.7088 - val_accuracy: 0.8250
Epoch 10/10
250/250 [==============================] - 6s 24ms/step - loss: 0.0272
- accuracy: 0.9919 - val_loss: 0.7787 - val_accuracy: 0.8090
782/782 [==============================] - 13s 12ms/step - loss:
0.4184 - accuracy: 0.8064
Number of training samples: 20000 - Test accuracy: 0.806
```




```
Found 10000 files belonging to 2 classes.
Using 8000 files for training.
Found 10000 files belonging to 2 classes.
Using 2000 files for validation.
Epoch 1/10
250/250 [==============================] - 45s 146ms/step - loss:
0.5936 - accuracy: 0.6649 - val_loss: 0.5425 - val_accuracy: 0.7450
Epoch 2/10
250/250 [==============================] - 28s 114ms/step - loss:
0.3710 - accuracy: 0.8441 - val_loss: 0.4706 - val_accuracy: 0.8010
Epoch 3/10
250/250 [==============================] - 26s 106ms/step - loss:
0.2731 - accuracy: 0.8898 - val_loss: 0.4085 - val_accuracy: 0.8260
Epoch 4/10
250/250 [==============================] - 8s 32ms/step - loss: 0.2061
- accuracy: 0.9251 - val_loss: 0.5924 - val_accuracy: 0.7935
Epoch 5/10
250/250 [==============================] - 6s 25ms/step - loss: 0.1511
- accuracy: 0.9444 - val_loss: 0.4524 - val_accuracy: 0.8260
Epoch 6/10
250/250 [==============================] - 7s 29ms/step - loss: 0.1108
- accuracy: 0.9597 - val_loss: 0.5092 - val_accuracy: 0.8265
Epoch 7/10
```

```
250/250 [==============================] - 8s 31ms/step - loss: 0.0795
- accuracy: 0.9745 - val_loss: 0.4816 - val_accuracy: 0.8345
Epoch 8/10
250/250 [==============================] - 6s 23ms/step - loss: 0.0647
- accuracy: 0.9814 - val_loss: 0.6588 - val_accuracy: 0.8255
Epoch 9/10
250/250 [==============================] - 8s 30ms/step - loss: 0.0389
- accuracy: 0.9871 - val_loss: 0.5965 - val_accuracy: 0.8345
Epoch 10/10
250/250 [==============================] - 7s 28ms/step - loss: 0.0382
- accuracy: 0.9880 - val_loss: 0.6963 - val_accuracy: 0.8300
782/782 [==============================] - 12s 12ms/step - loss:
0.4233 - accuracy: 0.8230
Number of training samples: 25000 - Test accuracy: 0.823
```




Number of training samples: 25000 - Accuracy
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
2
4
6
8
10
Training accuracy
Validation accuracy