In [1]:
```python
import tensorflow as tf
from tensorflow.keras.datasets import imdb
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense, Dropout

# Load the IMDb dataset
max_features = 10000  # consider only the top 10000 words
(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=max_features)

# Pad sequences to a fixed length
maxlen = 256  # truncate or pad reviews to this length
x_train = pad_sequences(x_train, maxlen=maxlen)
x_test = pad_sequences(x_test, maxlen=maxlen)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/imdb.npz
17464789/17464789 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
```

In [2]:
```python
embedding_dim = 128  # size of the word embeddings

model = Sequential([
    Embedding(max_features, embedding_dim, input_length=maxlen),
    LSTM(64),
    Dense(32, activation='relu'),
    Dropout(0.5),  # Dropout for regularization
    Dense(1, activation='sigmoid')
])
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/embedding.py:97: UserWarning: Argument `input
_length` is deprecated. Just remove it.
  warnings.warn(
```

In [3]:
```python
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])

model.summary()  # Print a summary of the model's layers and parameters
```

Model: "sequential"

```
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
┃ Layer (type)                     ┃ Output Shape            ┃       Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
│ embedding (Embedding)            │ ?                       │   0 (unbuilt) │
├──────────────────────────────────┼─────────────────────────┼───────────────┤
│ lstm (LSTM)                      │ ?                       │   0 (unbuilt) │
├──────────────────────────────────┼─────────────────────────┼───────────────┤
│ dense (Dense)                    │ ?                       │   0 (unbuilt) │
├──────────────────────────────────┼─────────────────────────┼───────────────┤
│ dropout (Dropout)                │ ?                       │             0 │
├──────────────────────────────────┼─────────────────────────┼───────────────┤
│ dense_1 (Dense)                  │ ?                       │   0 (unbuilt) │
└──────────────────────────────────┴─────────────────────────┴───────────────┘
 Total params: 0 (0.00 B)
 Trainable params: 0 (0.00 B)
 Non-trainable params: 0 (0.00 B)
Epoch 1/10
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
/tmp/ipython-input-336065820.py in <cell line: 0>()
      9 batch_size = 64
     10 
---> 11 history = model.fit(x_train, y_train,
     12                     epochs=epochs,
     13                     batch_size=batch_size,

/usr/local/lib/python3.11/dist-packages/keras/src/utils/traceback_utils.py in error_handler(*args, **kwarg
s)
    115         filtered_tb = None
    116         try:
--> 117             return fn(*args, **kwargs)
    118         except Exception as e:
    119             filtered_tb = _process_traceback_frames(e.__traceback__)

/usr/local/lib/python3.11/dist-packages/keras/src/backend/tensorflow/trainer.py in fit(self, x, y, batch_si
ze, epochs, verbose, callbacks, validation_split, validation_data, shuffle, class_weight, sample_weight, in
itial_epoch, steps_per_epoch, validation_steps, validation_batch_size, validation_freq)
    375                 for step, iterator in epoch_iterator:
    376                     callbacks.on_train_batch_begin(step)
--> 377                     logs = self.train_function(iterator)
    378                     callbacks.on_train_batch_end(step, logs)
    379                     if self.stop_training:

/usr/local/lib/python3.11/dist-packages/keras/src/backend/tensorflow/trainer.py in function(iterator)
    218                 iterator, (tf.data.Iterator, tf.distribute.DistributedIterator)
    219             ):
--> 220                 opt_outputs = multi_step_on_iterator(iterator)
    221                 if not opt_outputs.has_value():
    222                     raise StopIteration

/usr/local/lib/python3.11/dist-packages/tensorflow/python/util/traceback_utils.py in error_handler(*args, *
*kwargs)
    148     filtered_tb = None
    149     try:
--> 150       return fn(*args, **kwargs)
    151     except Exception as e:
    152       filtered_tb = _process_traceback_frames(e.__traceback__)

/usr/local/lib/python3.11/dist-packages/tensorflow/python/eager/polymorphic_function/polymorphic_function.p
```

```
y in __call__(self, *args, **kwds)
    831
    832       with OptionalXlaContext(self._jit_compile):
--> 833         result = self._call(*args, **kwds)
    834
    835       new_tracing_count = self.experimental_get_tracing_count()

/usr/local/lib/python3.11/dist-packages/tensorflow/python/eager/polymorphic_function/polymorphic_function.p
y in _call(self, *args, **kwds)
    887       # This is the first call of __call__, so we have to initialize.
    888       initializers = []
--> 889       self._initialize(args, kwds, add_initializers_to=initializers)
    890     finally:
    891       # At this point we know that the initialization is complete (or less

/usr/local/lib/python3.11/dist-packages/tensorflow/python/eager/polymorphic_function/polymorphic_function.p
y in _initialize(self, args, kwds, add_initializers_to)
    694     )
    695     # Force the definition of the function for these arguments
--> 696     self._concrete_variable_creation_fn = tracing_compilation.trace_function(
    697         args, kwds, self._variable_creation_config
    698     )

/usr/local/lib/python3.11/dist-packages/tensorflow/python/eager/polymorphic_function/tracing_compilation.py
in trace_function(args, kwargs, tracing_options)
    176       kwargs = {}
    177
--> 178     concrete_function = _maybe_define_function(
    179         args, kwargs, tracing_options
    180     )

/usr/local/lib/python3.11/dist-packages/tensorflow/python/eager/polymorphic_function/tracing_compilation.py
in _maybe_define_function(args, kwargs, tracing_options)
    281         else:
    282           target_func_type = lookup_func_type
--> 283         concrete_function = _create_concrete_function(
    284             target_func_type, lookup_func_context, func_graph, tracing_options
    285         )

/usr/local/lib/python3.11/dist-packages/tensorflow/python/eager/polymorphic_function/tracing_compilation.py
in _create_concrete_function(function_type, type_context, func_graph, tracing_options)
    308       attributes_lib.DISABLE_ACD, False
    309   )
--> 310   traced_func_graph = func_graph_module.func_graph_from_py_func(
    311       tracing_options.name,
    312       tracing_options.python_function,

/usr/local/lib/python3.11/dist-packages/tensorflow/python/framework/func_graph.py in func_graph_from_py_fun
c(name, python_func, args, kwargs, signature, func_graph, add_control_dependencies, arg_names, op_return_va
llue, collections, capture_by_value, create_placeholders)
   1058
   1059     _, original_func = tf_decorator.unwrap(python_func)
-> 1060     func_outputs = python_func(*func_args, **func_kwargs)
   1061
   1062     # invariant: `func_outputs` contains only Tensors, CompositeTensors,

/usr/local/lib/python3.11/dist-packages/tensorflow/python/eager/polymorphic_function/polymorphic_function.p
y in wrapped_fn(*args, **kwds)
    597         # the function a weak reference to itself to avoid a reference cycle.
    598         with OptionalXlaContext(compile_with_xla):
--> 599           out = weak_wrapped_fn().__wrapped__(*args, **kwds)
    600         return out
    601

/usr/local/lib/python3.11/dist-packages/tensorflow/python/eager/polymorphic_function/autograph_util.py in a
utograph_handler(*args, **kwargs)
     39     """Calls a converted version of original_func."""
     40     try:
---> 41       return api.converted_call(
     42           original_func,
     43           args,

/usr/local/lib/python3.11/dist-packages/tensorflow/python/autograph/impl/api.py in converted_call(f, args,
kwargs, caller_fn_scope, options)
    337   if is_autograph_artifact(f):
    338     logging.log(2, 'Permanently allowed: %s: AutoGraph artifact', f)
--> 339     return _call_unconverted(f, args, kwargs, options)
    340
    341   # If this is a partial, unwrap it and redo all the checks.

/usr/local/lib/python3.11/dist-packages/tensorflow/python/autograph/impl/api.py in _call_unconverted(f, arg
s, kwargs, options, update_cache)
    457
    458   if kwargs is not None:
```

```
--> 459     return f(*args, **kwargs)
    460   return f(*args)
    461 

/usr/local/lib/python3.11/dist-packages/tensorflow/python/autograph/impl/api.py in wrapper(*args, **kwargs)
    641   def wrapper(*args, **kwargs):
    642     with ag_ctx.ControlStatusCtx(status=ag_ctx.Status.DISABLED):
--> 643       return func(*args, **kwargs)
    644 
    645   if inspect.isfunction(func) or inspect.ismethod(func):

/usr/local/lib/python3.11/dist-packages/keras/src/backend/tensorflow/trainer.py in multi_step_on_iterator(i
terator)
    131             if self.steps_per_execution == 1:
    132                 return tf.experimental.Optional.from_value(
--> 133                     one_step_on_data(iterator.get_next())
    134                 )
    135 

/usr/local/lib/python3.11/dist-packages/tensorflow/python/util/traceback_utils.py in error_handler(*args, *
*kwargs)
    148     filtered_tb = None
    149     try:
--> 150       return fn(*args, **kwargs)
    151     except Exception as e:
    152       filtered_tb = _process_traceback_frames(e.__traceback__)

/usr/local/lib/python3.11/dist-packages/tensorflow/python/eager/polymorphic_function/polymorphic_function.p
y in __call__(self, *args, **kwds)
    831 
    832       with OptionalXlaContext(self._jit_compile):
--> 833         result = self._call(*args, **kwds)
    834 
    835       new_tracing_count = self.experimental_get_tracing_count()

/usr/local/lib/python3.11/dist-packages/tensorflow/python/eager/polymorphic_function/polymorphic_function.p
y in _call(self, *args, **kwds)
    904         # Lifting succeeded, so variables are initialized and we can run the
    905         # no_variable_creation function.
--> 906         return tracing_compilation.call_function(
    907             args, kwds, self._no_variable_creation_config
    908         )
```

```
/usr/local/lib/python3.11/dist-packages/tensorflow/python/eager/polymorphic_function/tracing_compilation.py
in call_function(args, kwargs, tracing_options)
    130   args = args if args else ()
    131   kwargs = kwargs if kwargs else {}
--> 132   function = trace_function(
    133       args=args, kwargs=kwargs, tracing_options=tracing_options
    134   )

/usr/local/lib/python3.11/dist-packages/tensorflow/python/eager/polymorphic_function/tracing_compilation.py
in trace_function(args, kwargs, tracing_options)
    176       kwargs = {}
    177 
--> 178     concrete_function = _maybe_define_function(
    179         args, kwargs, tracing_options
    180     )

/usr/local/lib/python3.11/dist-packages/tensorflow/python/eager/polymorphic_function/tracing_compilation.py
in _maybe_define_function(args, kwargs, tracing_options)
    281         else:
    282           target_func_type = lookup_func_type
--> 283         concrete_function = _create_concrete_function(
    284             target_func_type, lookup_func_context, func_graph, tracing_options
    285         )

/usr/local/lib/python3.11/dist-packages/tensorflow/python/eager/polymorphic_function/tracing_compilation.py
in _create_concrete_function(function_type, type_context, func_graph, tracing_options)
    308       attributes_lib.DISABLE_ACD, False
    309   )
--> 310   traced_func_graph = func_graph_module.func_graph_from_py_func(
    311       tracing_options.name,
    312       tracing_options.python_function,

/usr/local/lib/python3.11/dist-packages/tensorflow/python/framework/func_graph.py in func_graph_from_py_fun
c(name, python_func, args, kwargs, signature, func_graph, add_control_dependencies, arg_names, op_return_va
lue, collections, capture_by_value, create_placeholders)
   1058 
   1059     _, original_func = tf_decorator.unwrap(python_func)
-> 1060     func_outputs = python_func(*func_args, **func_kwargs)
   1061 
   1062     # invariant: `func_outputs` contains only Tensors, CompositeTensors,
/usr/local/lib/python3.11/dist-packages/tensorflow/python/eager/polymorphic_function/polymorphic_function.p
y in wrapped_fn(*args, **kwds)
    597         # the function a weak reference to itself to avoid a reference cycle.
    598         with OptionalXlaContext(compile_with_xla):
--> 599           out = weak_wrapped_fn().__wrapped__(*args, **kwds)
    600         return out
    601 

/usr/local/lib/python3.11/dist-packages/tensorflow/python/eager/polymorphic_function/autograph_util.py in a
utograph_handler(*args, **kwargs)
     39     """Calls a converted version of original_func."""
     40     try:
---> 41       return api.converted_call(
     42           original_func,
     43           args,

/usr/local/lib/python3.11/dist-packages/tensorflow/python/autograph/impl/api.py in converted_call(f, args,
kwargs, caller_fn_scope, options)
    329   if conversion.is_in_allowlist_cache(f, options):
    330     logging.log(2, 'Allowlisted %s: from cache', f)
--> 331     return _call_unconverted(f, args, kwargs, options, False)
    332 
    333   if ag_ctx.control_status_ctx().status == ag_ctx.Status.DISABLED:

/usr/local/lib/python3.11/dist-packages/tensorflow/python/autograph/impl/api.py in _call_unconverted(f, arg
s, kwargs, options, update_cache)
    457 
    458   if kwargs is not None:
--> 459     return f(*args, **kwargs)
    460   return f(*args)
    461 

/usr/local/lib/python3.11/dist-packages/tensorflow/python/autograph/impl/api.py in wrapper(*args, **kwargs)
    641   def wrapper(*args, **kwargs):
    642     with ag_ctx.ControlStatusCtx(status=ag_ctx.Status.DISABLED):
--> 643       return func(*args, **kwargs)
    644 
    645   if inspect.isfunction(func) or inspect.ismethod(func):

/usr/local/lib/python3.11/dist-packages/keras/src/backend/tensorflow/trainer.py in one_step_on_data(data)
    112         def one_step_on_data(data):
    113             """Runs a single training step on a batch of data."""
```

```
--> 114             outputs = self.distribute_strategy.run(step_function, args=(data,))
    115             outputs = reduce_per_replica(
    116                 outputs,

/usr/local/lib/python3.11/dist-packages/tensorflow/python/distribute/distribute_lib.py in run(***failed res
olving arguments***)
   1671       fn = autograph.tf_convert(
   1672           fn, autograph_ctx.control_status_ctx(), convert_by_default=False)
-> 1673       return self._extended.call_for_each_replica(fn, args=args, kwargs=kwargs)
   1674 
   1675   def reduce(self, reduce_op, value, axis):

/usr/local/lib/python3.11/dist-packages/tensorflow/python/distribute/distribute_lib.py in call_for_each_rep
lica(self, fn, args, kwargs)
   3261       kwargs = {}
   3262     with self._container_strategy().scope():
-> 3263       return self._call_for_each_replica(fn, args, kwargs)
   3264 
   3265   def _call_for_each_replica(self, fn, args, kwargs):

/usr/local/lib/python3.11/dist-packages/tensorflow/python/distribute/distribute_lib.py in _call_for_each_re
plica(self, fn, args, kwargs)
   4059   def _call_for_each_replica(self, fn, args, kwargs):
   4060     with ReplicaContext(self._container_strategy(), replica_id_in_sync_group=0):
-> 4061       return fn(*args, **kwargs)
   4062 
   4063   def _reduce_to(self, reduce_op, value, destinations, options):

/usr/local/lib/python3.11/dist-packages/tensorflow/python/autograph/impl/api.py in wrapper(*args, **kwargs)
    641   def wrapper(*args, **kwargs):
    642     with ag_ctx.ControlStatusCtx(status=ag_ctx.Status.DISABLED):
--> 643       return func(*args, **kwargs)
    644 
    645   if inspect.isfunction(func) or inspect.ismethod(func):

/usr/local/lib/python3.11/dist-packages/keras/src/backend/tensorflow/trainer.py in train_step(self, data)
     79 
     80             # Update weights
---> 81             self.optimizer.apply_gradients(zip(gradients, trainable_weights))
     82         else:
     83             warnings.warn("The model does not have any trainable weights.")
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/optimizers/base_optimizer.py in apply_gradients(self, gra
ds_and_vars)
    461     def apply_gradients(self, grads_and_vars):
    462         grads, trainable_variables = zip(*grads_and_vars)
--> 463         self.apply(grads, trainable_variables)
    464         # Return iterations for compat with tf.keras.
    465         return self._iterations

/usr/local/lib/python3.11/dist-packages/keras/src/optimizers/base_optimizer.py in apply(self, grads, traina
ble_variables)
    525 
    526                 # Apply gradient updates.
--> 527                 self._backend_apply_gradients(grads, trainable_variables)
    528                 # Apply variable constraints after applying gradients.
    529                 for variable in trainable_variables:

/usr/local/lib/python3.11/dist-packages/keras/src/optimizers/base_optimizer.py in _backend_apply_gradients
(self, grads, trainable_variables)
    591 
    592             # Run update step.
--> 593             self._backend_update_step(
    594                 grads, trainable_variables, self.learning_rate
    595             )

/usr/local/lib/python3.11/dist-packages/keras/src/backend/tensorflow/optimizer.py in _backend_update_step(s
elf, grads, trainable_variables, learning_rate)
    118         grads_and_vars = list(zip(grads, trainable_variables))
    119         grads_and_vars = self._all_reduce_sum_gradients(grads_and_vars)
--> 120         tf.__internal__.distribute.interim.maybe_merge_call(
    121             self._distributed_tf_update_step,
    122             self._distribution_strategy,

/usr/local/lib/python3.11/dist-packages/tensorflow/python/distribute/merge_call_interim.py in maybe_merge_c
all(fn, strategy, *args, **kwargs)
     49   """
     50   if strategy_supports_no_merge_call():
---> 51     return fn(strategy, *args, **kwargs)
     52   else:
     53     return distribute_lib.get_replica_context().merge_call(

/usr/local/lib/python3.11/dist-packages/keras/src/backend/tensorflow/optimizer.py in _distributed_tf_update
_step(self, distribution, grads_and_vars, learning_rate)
```

```
    132 
    133         for grad, var in grads_and_vars:
--> 134             distribution.extended.update(
    135                 var,
    136                 apply_grad_to_update_var,

/usr/local/lib/python3.11/dist-packages/tensorflow/python/distribute/distribute_lib.py in update(self, var,
fn, args, kwargs, group)
   3005         return self._update(var, fn, args, kwargs, group)
   3006     else:
-> 3007       return self._replica_ctx_update(
   3008           var, fn, args=args, kwargs=kwargs, group=group)
   3009 

/usr/local/lib/python3.11/dist-packages/tensorflow/python/distribute/distribute_lib.py in _replica_ctx_upda
te(self, var, fn, args, kwargs, group)
   2884       return self.update(var, fn, merged_args, merged_kwargs, group=group)
   2885 
-> 2886     return replica_context.merge_call(merge_fn, args=args, kwargs=kwargs)
   2887 
   2888   def _gather_to(self, value, destinations, axis, options=None):

/usr/local/lib/python3.11/dist-packages/tensorflow/python/distribute/distribute_lib.py in merge_call(self,
merge_fn, args, kwargs)
   3476     merge_fn = autograph.tf_convert(
   3477         merge_fn, autograph_ctx.control_status_ctx(), convert_by_default=False)
-> 3478     return self._merge_call(merge_fn, args, kwargs)
   3479 
   3480   def _merge_call(self, merge_fn, args, kwargs):

/usr/local/lib/python3.11/dist-packages/tensorflow/python/distribute/distribute_lib.py in _merge_call(self,
merge_fn, args, kwargs)
   3483         _CrossReplicaThreadMode(self._strategy))  # pylint: disable=protected-access
   3484     try:
-> 3485       return merge_fn(self._strategy, *args, **kwargs)
   3486     finally:
   3487       _pop_per_thread_mode()

/usr/local/lib/python3.11/dist-packages/tensorflow/python/autograph/impl/api.py in wrapper(*args, **kwargs)
    641   def wrapper(*args, **kwargs):
    642     with ag_ctx.ControlStatusCtx(status=ag_ctx.Status.DISABLED):
--> 643       return func(*args, **kwargs)
```

```
    644 
    645   if inspect.isfunction(func) or inspect.ismethod(func):

/usr/local/lib/python3.11/dist-packages/tensorflow/python/distribute/distribute_lib.py in merge_fn(_, *merg
ed_args, **merged_kwargs)
   2882 
   2883     def merge_fn(_, *merged_args, **merged_kwargs):
-> 2884       return self.update(var, fn, merged_args, merged_kwargs, group=group)
   2885 
   2886     return replica_context.merge_call(merge_fn, args=args, kwargs=kwargs)

/usr/local/lib/python3.11/dist-packages/tensorflow/python/distribute/distribute_lib.py in update(self, var,
fn, args, kwargs, group)
   3003           fn, autograph_ctx.control_status_ctx(), convert_by_default=False)
   3004       with self._container_strategy().scope():
-> 3005         return self._update(var, fn, args, kwargs, group)
   3006     else:
   3007       return self._replica_ctx_update(

/usr/local/lib/python3.11/dist-packages/tensorflow/python/distribute/distribute_lib.py in _update(self, va
r, fn, args, kwargs, group)
   4073     # The implementations of _update() and _update_non_slot() are identical
   4074     # except _update() passes `var` as the first argument to `fn()`.
-> 4075     return self._update_non_slot(var, fn, (var,) + tuple(args), kwargs, group)
   4076 
   4077   def _update_non_slot(self, colocate_with, fn, args, kwargs, should_group):

/usr/local/lib/python3.11/dist-packages/tensorflow/python/distribute/distribute_lib.py in _update_non_slot
(self, colocate_with, fn, args, kwargs, should_group)
   4079     # once that value is used for something.
   4080     with UpdateContext(colocate_with):
-> 4081       result = fn(*args, **kwargs)
   4082       if should_group:
   4083         return result

/usr/local/lib/python3.11/dist-packages/tensorflow/python/autograph/impl/api.py in wrapper(*args, **kwargs)
    641   def wrapper(*args, **kwargs):
    642     with ag_ctx.ControlStatusCtx(status=ag_ctx.Status.DISABLED):
--> 643       return func(*args, **kwargs)
    644 
    645   if inspect.isfunction(func) or inspect.ismethod(func):
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/backend/tensorflow/optimizer.py in apply_grad_to_update_v
ar(var, grad, learning_rate)
    129     ):
    130         def apply_grad_to_update_var(var, grad, learning_rate):
--> 131             return self.update_step(grad, var, learning_rate)
    132 
    133         for grad, var in grads_and_vars:

/usr/local/lib/python3.11/dist-packages/keras/src/optimizers/adam.py in update_step(self, gradient, variabl
e, learning_rate)
    102     def update_step(self, gradient, variable, learning_rate):
    103         """Update step given gradient and the associated model variable."""
--> 104         lr = ops.cast(learning_rate, variable.dtype)
    105         gradient = ops.cast(gradient, variable.dtype)
    106         local_step = ops.cast(self.iterations + 1, variable.dtype)

/usr/local/lib/python3.11/dist-packages/keras/src/ops/core.py in cast(x, dtype)
    801     if any_symbolic_tensors((x,)):
    802         return Cast(dtype=dtype)(x)
--> 803     return backend.core.cast(x, dtype)
    804 
    805 

/usr/local/lib/python3.11/dist-packages/keras/src/backend/tensorflow/core.py in cast(x, dtype)
    215         return x
    216     else:
--> 217         return tf.cast(x, dtype=dtype)
    218 
    219 

/usr/local/lib/python3.11/dist-packages/tensorflow/python/util/traceback_utils.py in error_handler(*args, *
*kwargs)
    148     filtered_tb = None
    149     try:
--> 150       return fn(*args, **kwargs)
    151     except Exception as e:
    152       filtered_tb = _process_traceback_frames(e.__traceback__)

/usr/local/lib/python3.11/dist-packages/tensorflow/python/util/dispatch.py in op_dispatch_handler(*args, **
kwargs)
   1258       # Fallback dispatch system (dispatch v1):
   1259       try:
```

```
-> 1260         return dispatch_target(*args, **kwargs)
   1261       except (TypeError, ValueError):
   1262         # Note: convert_to_eager_tensor currently raises a ValueError, not a

/usr/local/lib/python3.11/dist-packages/tensorflow/python/ops/math_ops.py in cast(x, dtype, name)
   1010       # allows some conversions that cast() can't do, e.g. casting numbers to
   1011       # strings.
-> 1012       x = ops.convert_to_tensor(x, name="x")
   1013       if x.dtype.is_complex and base_type.is_floating:
   1014         logging.warn(

/usr/local/lib/python3.11/dist-packages/tensorflow/python/profiler/trace.py in wrapped(*args, **kwargs)
    181         with Trace(trace_name, **trace_kwargs):
    182           return func(*args, **kwargs)
--> 183       return func(*args, **kwargs)
    184 
    185     return wrapped

/usr/local/lib/python3.11/dist-packages/tensorflow/python/framework/ops.py in convert_to_tensor(value, dtyp
e, name, as_ref, preferred_dtype, dtype_hint, ctx, accepted_result_types)
    734   # TODO(b/142518781): Fix all call-sites and remove redundant arg
    735   preferred_dtype = preferred_dtype or dtype_hint
--> 736   return tensor_conversion_registry.convert(
    737       value, dtype, name, as_ref, preferred_dtype, accepted_result_types
    738   )

/usr/local/lib/python3.11/dist-packages/tensorflow/python/framework/tensor_conversion_registry.py in conver
t(value, dtype, name, as_ref, preferred_dtype, accepted_result_types)
    207   overload = getattr(value, "__tf_tensor__", None)
    208   if overload is not None:
--> 209     return overload(dtype, name)  #  pylint: disable=not-callable
    210 
    211   for base_type, conversion_func in get(type(value)):

/usr/local/lib/python3.11/dist-packages/keras/src/backend/tensorflow/core.py in __tf_tensor__(self, dtype,
name)
     82     # Overload native accessor.
     83     def __tf_tensor__(self, dtype=None, name=None):
---> 84         return tf.convert_to_tensor(self.value, dtype=dtype, name=name)
     85 
     86     # Methods below are for SavedModel support
```

```
/usr/local/lib/python3.11/dist-packages/tensorflow/python/util/traceback_utils.py in error_handler(*args, *
*kwargs)
    148     filtered_tb = None
    149     try:
--> 150       return fn(*args, **kwargs)
    151     except Exception as e:
    152       filtered_tb = _process_traceback_frames(e.__traceback__)

/usr/local/lib/python3.11/dist-packages/tensorflow/python/util/dispatch.py in op_dispatch_handler(*args, **
kwargs)
   1258       # Fallback dispatch system (dispatch v1):
   1259       try:
-> 1260         return dispatch_target(*args, **kwargs)
   1261       except (TypeError, ValueError):
   1262         # Note: convert_to_eager_tensor currently raises a ValueError, not a

/usr/local/lib/python3.11/dist-packages/tensorflow/python/framework/tensor_conversion.py in convert_to_tens
or_v2_with_dispatch(value, dtype, dtype_hint, name)
    159     ValueError: If the `value` is a tensor not of given `dtype` in graph mode.
    160   """
--> 161   return convert_to_tensor_v2(
    162       value, dtype=dtype, dtype_hint=dtype_hint, name=name
    163   )

/usr/local/lib/python3.11/dist-packages/tensorflow/python/framework/tensor_conversion.py in convert_to_tens
or_v2(value, dtype, dtype_hint, name)
    169   """Converts the given `value` to a `Tensor`."""
    170   # preferred_dtype = preferred_dtype or dtype_hint
--> 171   return tensor_conversion_registry.convert(
    172       value, dtype, name, preferred_dtype=dtype_hint
    173   )

/usr/local/lib/python3.11/dist-packages/tensorflow/python/framework/tensor_conversion_registry.py in conver
t(value, dtype, name, as_ref, preferred_dtype, accepted_result_types)
    232
    233     if ret is None:
--> 234       ret = conversion_func(value, dtype=dtype, name=name, as_ref=as_ref)
    235
    236     if ret is NotImplemented:

/usr/local/lib/python3.11/dist-packages/tensorflow/python/ops/resource_variable_ops.py in _dense_var_to_ten
sor(var, dtype, name, as_ref)
```

```
   2376
   2377 def _dense_var_to_tensor(var, dtype=None, name=None, as_ref=False):
-> 2378   return var._dense_var_to_tensor(dtype=dtype, name=name, as_ref=as_ref)  # pylint: disable=protect
ed-access
   2379
   2380

/usr/local/lib/python3.11/dist-packages/tensorflow/python/ops/resource_variable_ops.py in _dense_var_to_ten
sor(***failed resolving arguments***)
   1622       return self.read_value().op.inputs[0]
   1623     else:
-> 1624       return self.value()
   1625
   1626   def __iadd__(self, unused_other):

/usr/local/lib/python3.11/dist-packages/tensorflow/python/ops/resource_variable_ops.py in value(self)
    656       return self._cached_value
    657     with ops.colocate_with(None, ignore_existing=True):
--> 658       return self._read_variable_op()
    659
    660   def _as_graph_element(self):

/usr/local/lib/python3.11/dist-packages/tensorflow/python/ops/resource_variable_ops.py in _read_variable_op
(self, no_copy)
    841           result = read_and_set_handle(no_copy)
    842     else:
--> 843       result = read_and_set_handle(no_copy)
    844
    845     if not context.executing_eagerly():

/usr/local/lib/python3.11/dist-packages/tensorflow/python/ops/resource_variable_ops.py in read_and_set_hand
le(no_copy)
    831       if no_copy and forward_compat.forward_compatible(2022, 5, 3):
    832         gen_resource_variable_ops.disable_copy_on_read(self.handle)
--> 833       result = gen_resource_variable_ops.read_variable_op(
    834           self.handle, self._dtype)
    835       _maybe_set_handle_data(self._dtype, self.handle, result)

/usr/local/lib/python3.11/dist-packages/tensorflow/python/ops/gen_resource_variable_ops.py in read_variable
_op(resource, dtype, name)
    546   # Add nodes to the TensorFlow graph.
    547   dtype = _execute.make_type(dtype, "dtype")
```

```
--> 548   _, _, _op, _outputs = _op_def_library._apply_op_helper(
    549         "ReadVariableOp", resource=resource, dtype=dtype, name=name)
    550   _result = _outputs[:]

/usr/local/lib/python3.11/dist-packages/tensorflow/python/framework/op_def_library.py in _apply_op_helper(o
p_type_name, name, **keywords)
    774   # Requires that op_def has passed validation (using the C++
    775   # ValidateOpDef() from ../framework/op_def_util.h).
--> 776   with g.as_default(), ops.name_scope(name) as scope:
    777     if fallback:
    778       _ExtractInputsAndAttrs(op_type_name, op_def, allowed_list_attr_map,

/usr/lib/python3.11/contextlib.py in __enter__(self)
    135         del self.args, self.kwds, self.func
    136         try:
--> 137             return next(self.gen)
    138         except StopIteration:
    139             raise RuntimeError("generator didn't yield") from None

/usr/local/lib/python3.11/dist-packages/tensorflow/python/framework/func_graph.py in _func_graph_as_default
_inner_cm(func_graph, outer_cm)
    890       (device_stack_has_callable(graph._device_function_stack) or
    891        (func_graph._distribution_strategy_stack and
--> 892         not ops.executing_eagerly_outside_functions()))):
    893     # Hard-code devices from device functions in the function body
    894     func_graph._device_function_stack = graph._device_function_stack.copy()

/usr/local/lib/python3.11/dist-packages/tensorflow/python/framework/ops.py in executing_eagerly_outside_fun
ctions()
   4869   else:
   4870     outer_context, _ = _get_outer_context_and_inner_device_stack()
-> 4871     with outer_context():
   4872       return context.executing_eagerly()
   4873 

/usr/lib/python3.11/contextlib.py in __enter__(self)
    135         del self.args, self.kwds, self.func
    136         try:
--> 137             return next(self.gen)
    138         except StopIteration:
    139             raise RuntimeError("generator didn't yield") from None
```

```
/usr/local/lib/python3.11/dist-packages/tensorflow/python/eager/context.py in _mode(self, mode)
   1160       # record a context switch; graph-based context switches are only logged
   1161       # when a graph is registered as the default graph.
-> 1162       self.context_switches.push(False, eager_mode, None)
   1163     try:
   1164       yield

/usr/local/lib/python3.11/dist-packages/tensorflow/python/eager/context.py in push(self, is_building_functi
on, enter_context_fn, device_stack)
    380
    381     self.stack.append(
--> 382         ContextSwitch(is_building_function, enter_context_fn, device_stack)
    383     )
    384

<string> in <lambda>(_cls, is_building_function, enter_context_fn, device_stack)

KeyboardInterrupt:
```

In [4]:

```python
# Train the model
epochs = 10
batch_size = 64

history = model.fit(x_train, y_train,
                    epochs=epochs,
                    batch_size=batch_size,
                    validation_split=0.2)
```

```
Epoch 1/10
313/313 ━━━━━━━━━━━━━━━━━━━━ 153s 480ms/step - accuracy: 0.6765 - loss: 0.5851 - val_accuracy: 0.7956 - val
_loss: 0.4339
Epoch 2/10
313/313 ━━━━━━━━━━━━━━━━━━━━ 204s 488ms/step - accuracy: 0.8818 - loss: 0.3070 - val_accuracy: 0.8678 - val
_loss: 0.3125
Epoch 3/10
313/313 ━━━━━━━━━━━━━━━━━━━━ 200s 485ms/step - accuracy: 0.9309 - loss: 0.1998 - val_accuracy: 0.8674 - val
_loss: 0.3799
Epoch 4/10
313/313 ━━━━━━━━━━━━━━━━━━━━ 212s 516ms/step - accuracy: 0.9468 - loss: 0.1580 - val_accuracy: 0.8040 - val
_loss: 0.4523
Epoch 5/10
313/313 ━━━━━━━━━━━━━━━━━━━━ 153s 490ms/step - accuracy: 0.9459 - loss: 0.1469 - val_accuracy: 0.8716 - val
_loss: 0.3894
Epoch 6/10
313/313 ━━━━━━━━━━━━━━━━━━━━ 152s 485ms/step - accuracy: 0.9634 - loss: 0.1056 - val_accuracy: 0.8586 - val
_loss: 0.4866
Epoch 7/10
313/313 ━━━━━━━━━━━━━━━━━━━━ 201s 481ms/step - accuracy: 0.9628 - loss: 0.1046 - val_accuracy: 0.8708 - val
_loss: 0.5045
Epoch 8/10
313/313 ━━━━━━━━━━━━━━━━━━━━ 160s 509ms/step - accuracy: 0.9871 - loss: 0.0429 - val_accuracy: 0.8582 - val
_loss: 0.5656
Epoch 9/10
313/313 ━━━━━━━━━━━━━━━━━━━━ 161s 516ms/step - accuracy: 0.9913 - loss: 0.0324 - val_accuracy: 0.8534 - val
_loss: 0.6193
Epoch 10/10
313/313 ━━━━━━━━━━━━━━━━━━━━ 155s 495ms/step - accuracy: 0.9936 - loss: 0.0280 - val_accuracy: 0.8500 - val
_loss: 0.6821
```

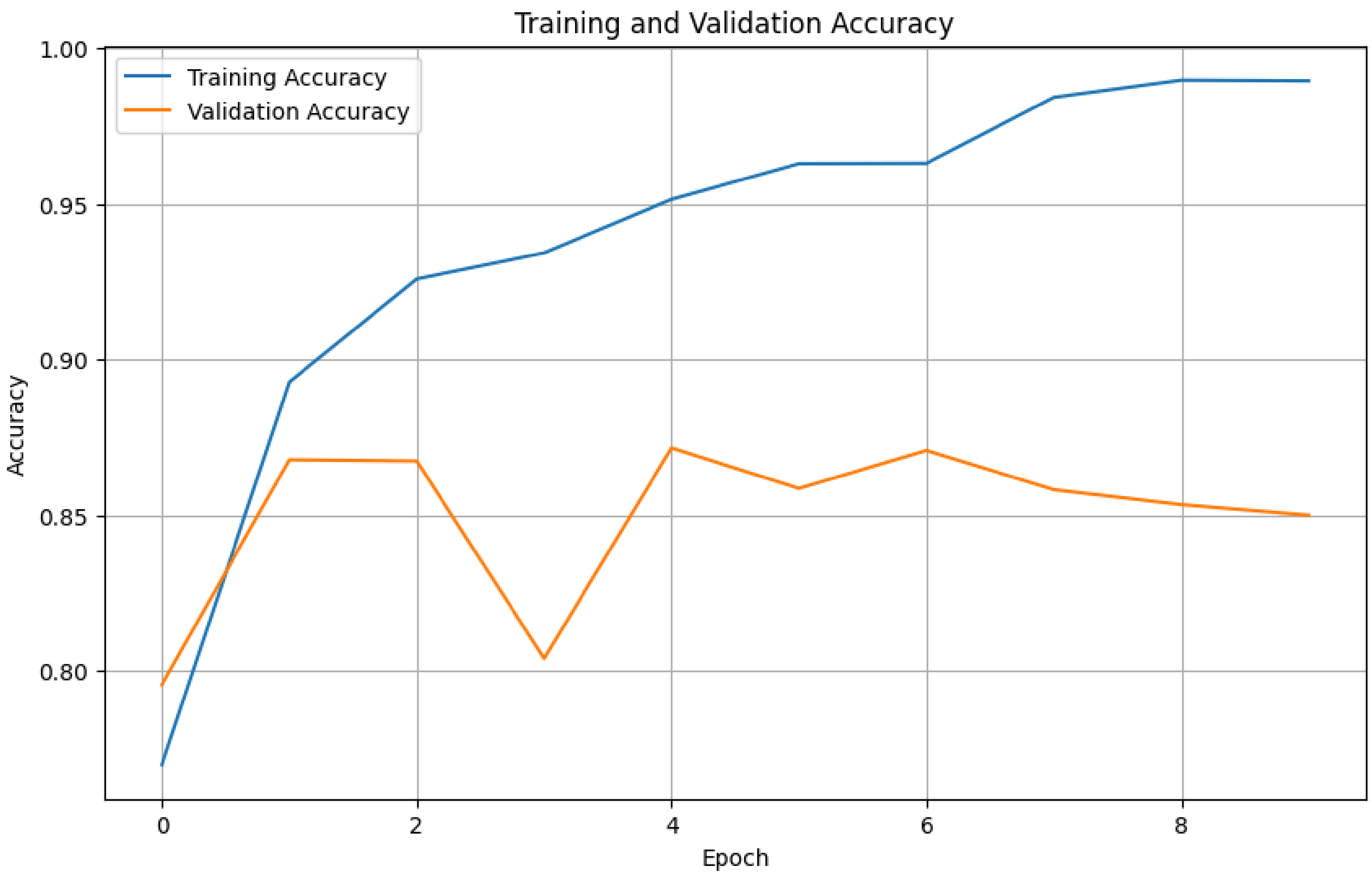
In [5]:

```python
loss, accuracy = model.evaluate(x_test, y_test)
print(f"Test Accuracy: {accuracy*100:.2f}%")

# Example of a prediction
# To predict on a new review, you'd need to tokenize and pad it first.
# Here, we'll just predict on a test sample.
sample_review = x_test[0:1]
prediction = model.predict(sample_review)
print(f"Prediction: {prediction[0][0]:.4f}")
print(f"Actual Label: {y_test[0]}")
```

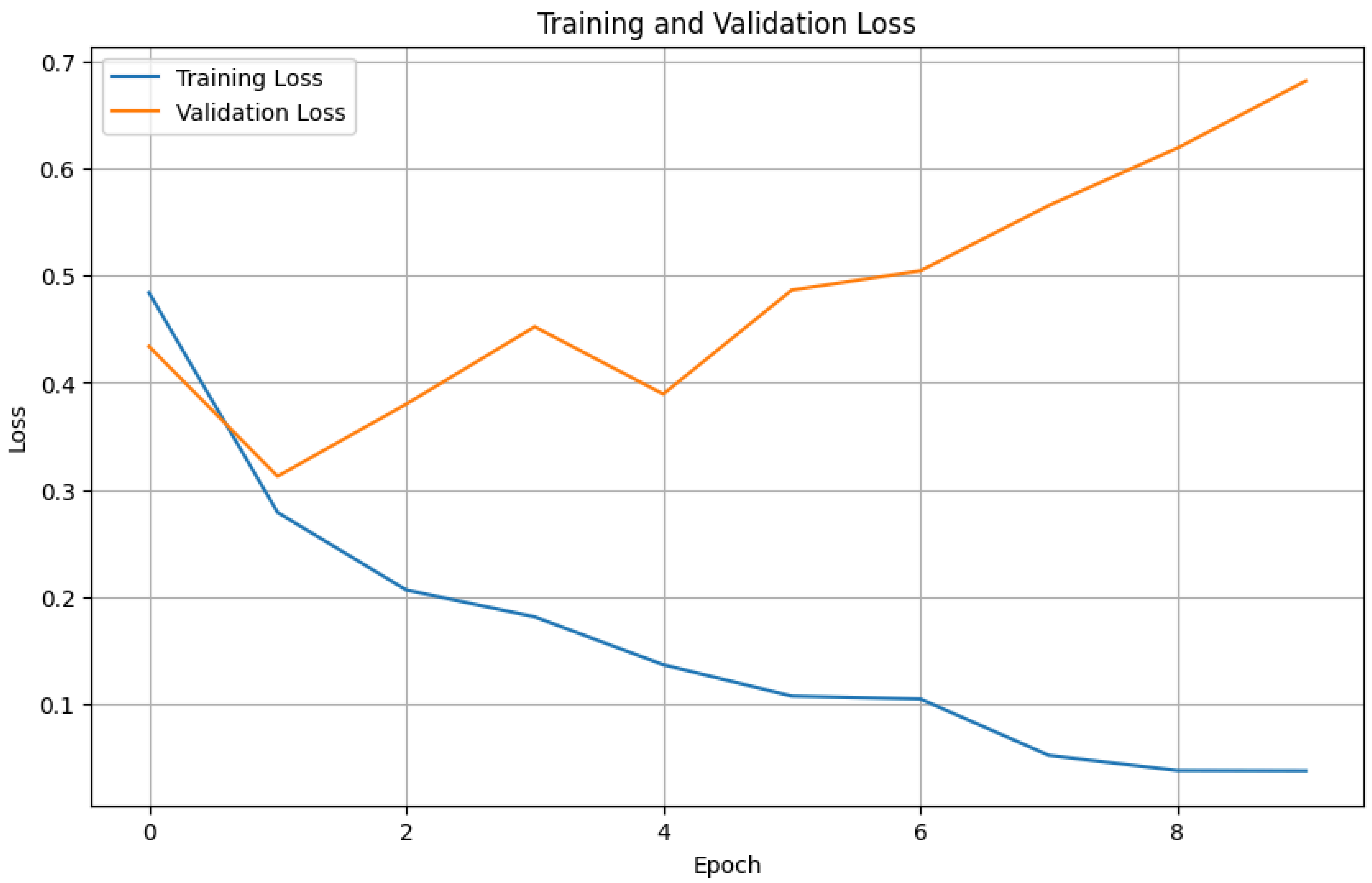
```
782/782 ━━━━━━━━━━━━━━━━━━━━ 46s 59ms/step - accuracy: 0.8485 - loss: 0.7358
Test Accuracy: 85.11%
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 334ms/step
Prediction: 0.1390
Actual Label: 0
```

In [7]:

```
import matplotlib.pyplot as plt
# Plot training and validation accuracy
plt.figure(figsize=(10, 6))
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.grid(True)
plt.show()

# Plot training and validation loss
plt.figure(figsize=(10, 6))
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.grid(True)
plt.show()
```



Training and Validation Accuracy
Training Accuracy
Validation Accuracy
Accuracy
Epoch
1.00
0.95
0.90
0.85
0.80
0
2
4
6
8


Training and Validation Loss
Training Loss
Validation Loss
Loss
Epoch
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
2
4
6
8

```
In [ ]:
```