# assignment6_2

July 18, 2021

[1]:

```python
from keras import layers
from keras import models
```

[2]:

```python
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(150, 150, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(1, activation='softmax'))
```

[3]:

```python
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 148, 148, 32)      896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 74, 74, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 72, 72, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 36, 36, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 34, 34, 128)       73856
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 17, 17, 128)       0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 15, 15, 128)       147584
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 7, 7, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 6272)              0
_________________________________________________________________
dense (Dense)                (None, 512)               3211776
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 513
=================================================================
Total params: 3,453,121
Trainable params: 3,453,121
Non-trainable params: 0
_________________________________________________________________
```

[5]:

```python
from keras import optimizers

model.compile(optimizer='rmsprop',
             loss='categorical_crossentropy',
             metrics=['accuracy'])
```

[11]:

```python
from keras.datasets import cifar10

(x_train, y_train), (x_test, y_test) = cifar10.load_data()
```

[14]:

```python
# x_train = x_train.reshape((60000, 28, 28, 1))
x_train = x_train.astype('float32')/255

# x_test = x_test.reshape((10000, 28, 28, 1))
x_test = x_test.astype('float32')/255
```

[18]:

```python
from keras.utils import to_categorical

x_train = to_categorical(x_train)
x_test = to_categorical(x_test)
```

[6]:

```python
x_val = x_train[:10000]
partial_x_train = x_train[10000:]
y_val = y_train[:10000]
partial_y_train = y_train[10000:]
```

[7]:

```python
batch_size = 128
epochs=50
```

[19]:

```python
history = model.fit(x_train,
                   y_train,
                   batch_size,
                   epochs,
                   validation_data = (x_test, y_test))
```

```
Epoch 1/50
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-19-d23c0c550696> in <module>
----> 1 history = model.fit(x_train,

      2                     y_train,
      3                     batch_size,
      4                     epochs,
      5                     validation_data = (x_test, y_test))

/opt/conda/lib/python3.8/site-packages/tensorflow/python/keras/engine/training.
 ↪py in fit(self, x, y, batch_size, epochs, verbose, callbacks,␣
 ↪validation_split, validation_data, shuffle, class_weight, sample_weight,␣
 ↪initial_epoch, steps_per_epoch, validation_steps, validation_batch_size,␣
 ↪validation_freq, max_queue_size, workers, use_multiprocessing)
   1098                 _r=1):
   1099               callbacks.on_train_batch_begin(step)
-> 1100               tmp_logs = self.train_function(iterator)
   1101               if data_handler.should_sync:
   1102                 context.async_wait()

/opt/conda/lib/python3.8/site-packages/tensorflow/python/eager/def_function.py␣
 ↪in __call__(self, *args, **kwds)
    826     tracing_count = self.experimental_get_tracing_count()
    827     with trace.Trace(self._name) as tm:
--> 828       result = self._call(*args, **kwds)
    829       compiler = "xla" if self._experimental_compile else "nonXla"
    830       new_tracing_count = self.experimental_get_tracing_count()

/opt/conda/lib/python3.8/site-packages/tensorflow/python/eager/def_function.py␣
 ↪in _call(self, *args, **kwds)
    860       # In this case we have not created variables on the first call. S␣
 ↪we can
    861       # run the first trace but we should fail if variables are created
--> 862       results = self._stateful_fn(*args, **kwds)
    863       if self._created_variables:
    864         raise ValueError("Creating variables on a non-first call to a␣
 ↪function"

/opt/conda/lib/python3.8/site-packages/tensorflow/python/eager/function.py in␣
 ↪__call__(self, *args, **kwargs)
   2939     with self._lock:
   2940       (graph_function,
-> 2941        filtered_flat_args) = self._maybe_define_function(args, kwargs)

   2942     return graph_function._call_flat(
   2943         filtered_flat_args, captured_inputs=graph_function.
 ↪captured_inputs)  # pylint: disable=protected-access
```

```
/opt/conda/lib/python3.8/site-packages/tensorflow/python/eager/function.py in␣
 ↪_maybe_define_function(self, args, kwargs)
   3355                   self.input_signature is None and
   3356                   call_context_key in self._function_cache.missed):
-> 3357                 return self._define_function_with_shape_relaxation(
   3358                     args, kwargs, flat_args, filtered_flat_args,␣
 ↪cache_key_context)
   3359

/opt/conda/lib/python3.8/site-packages/tensorflow/python/eager/function.py in␣
 ↪_define_function_with_shape_relaxation(self, args, kwargs, flat_args,␣
 ↪filtered_flat_args, cache_key_context)
   3277               expand_composites=True)
   3278
-> 3279         graph_function = self._create_graph_function(
   3280             args, kwargs, override_flat_arg_shapes=relaxed_arg_shapes)
   3281         self._function_cache.arg_relaxed[rank_only_cache_key] =␣
 ↪graph_function

/opt/conda/lib/python3.8/site-packages/tensorflow/python/eager/function.py in␣
 ↪_create_graph_function(self, args, kwargs, override_flat_arg_shapes)
   3194         arg_names = base_arg_names + missing_arg_names
   3195         graph_function = ConcreteFunction(
-> 3196             func_graph_module.func_graph_from_py_func(
   3197                 self._name,
   3198                 self._python_function,

/opt/conda/lib/python3.8/site-packages/tensorflow/python/framework/func_graph.py␣
 ↪in func_graph_from_py_func(name, python_func, args, kwargs, signature,␣
 ↪func_graph, autograph, autograph_options, add_control_dependencies, arg_names,␣
 ↪op_return_value, collections, capture_by_value, override_flat_arg_shapes)
    988         _, original_func = tf_decorator.unwrap(python_func)
    989
--> 990       func_outputs = python_func(*func_args, **func_kwargs)
    991
    992       # invariant: `func_outputs` contains only Tensors,␣
 ↪CompositeTensors,

/opt/conda/lib/python3.8/site-packages/tensorflow/python/eager/def_function.py␣
 ↪in wrapped_fn(*args, **kwds)
    632             xla_context.Exit()
    633         else:
--> 634           out = weak_wrapped_fn().__wrapped__(*args, **kwds)
    635         return out
    636
```

```
/opt/conda/lib/python3.8/site-packages/tensorflow/python/framework/func_graph.p␣
↪in wrapper(*args, **kwargs)
    975           except Exception as e:  # pylint:disable=broad-except
    976             if hasattr(e, "ag_error_metadata"):
--> 977               raise e.ag_error_metadata.to_exception(e)
    978             else:
    979               raise

ValueError: in user code:

    /opt/conda/lib/python3.8/site-packages/tensorflow/python/keras/engine/
↪training.py:805 train_function  *
        return step_function(self, iterator)
    /opt/conda/lib/python3.8/site-packages/tensorflow/python/keras/engine/
↪training.py:795 step_function  **
        outputs = model.distribute_strategy.run(run_step, args=(data,))
    /opt/conda/lib/python3.8/site-packages/tensorflow/python/distribute/
↪distribute_lib.py:1259 run
        return self._extended.call_for_each_replica(fn, args=args, kwargs=kwarg
    /opt/conda/lib/python3.8/site-packages/tensorflow/python/distribute/
↪distribute_lib.py:2730 call_for_each_replica
        return self._call_for_each_replica(fn, args, kwargs)
    /opt/conda/lib/python3.8/site-packages/tensorflow/python/distribute/
↪distribute_lib.py:3417 _call_for_each_replica
        return fn(*args, **kwargs)
    /opt/conda/lib/python3.8/site-packages/tensorflow/python/keras/engine/
↪training.py:788 run_step  **
        outputs = model.train_step(data)
    /opt/conda/lib/python3.8/site-packages/tensorflow/python/keras/engine/
↪training.py:754 train_step
        y_pred = self(x, training=True)
    /opt/conda/lib/python3.8/site-packages/tensorflow/python/keras/engine/
↪base_layer.py:998 __call__
        input_spec.assert_input_compatibility(self.input_spec, inputs, self.nam
    /opt/conda/lib/python3.8/site-packages/tensorflow/python/keras/engine/
↪input_spec.py:255 assert_input_compatibility
        raise ValueError(

    ValueError: Input 0 of layer sequential is incompatible with the layer:␣
↪expected axis -1 of input shape to have value 3 but received input with shape
↪(None, 32, 32, 3, 2)
```

```
[ ]:
```