3d-image-recognition-using-cnn-1

March 3, 2024

[124]:
```
import pandas as pd
import numpy as np
import tensorflow as tf
from tensorflow.keras import datasets,layers,models
import matplotlib.pyplot as plt
```

[125]:
```
(X_train,y_train),(X_test,y_test)= datasets.cifar10.load_data()
X_train.shape
```

[125]:
```
(50000, 32, 32, 3)
```

[126]:
```
X_test.shape
```

[126]:
```
(10000, 32, 32, 3)
```

[127]:
```
y_train.shape
```

[127]:
```
(50000, 1)
```

[128]:
```
y_train[:10]
```

[128]:
```
array([[6],
       [9],
       [9],
       [4],
       [1],
       [1],
       [2],
       [7],
       [8],
       [3]], dtype=uint8)
```

[129]:
```
y_train = y_train.reshape(-1,)
y_train[:10]
```

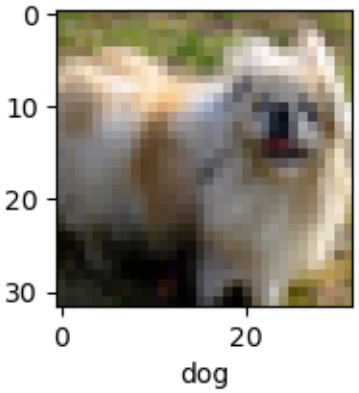
[129]:
```
array([6, 9, 9, 4, 1, 1, 2, 7, 8, 3], dtype=uint8)
```

[130]:
```
y_test=y_test.reshape(-1)
```

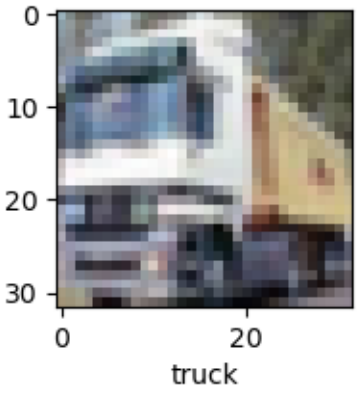
[131]:
```
classes =␣
↪["airplane","automobile","bird","cat","deer","dog","frog","horse","ship","truck"]
```

[132]:
```
def plot_sample(X, y, index):
    plt.figure(figsize = (15,2))
    plt.imshow(X[index])
    plt.xlabel(classes[y[index]])
```

[133]:
```
plot_sample(X_train, y_train, 40)
```



[134]:
```
plot_sample(X_train, y_train, 1)
```



[123]:
```
X_train = X_train / 255
X_test = X_test / 255
```

[147]:
```
ann = models.Sequential([
        layers.Flatten(input_shape=(32,32,3)),
```

```
        layers.Dense(3000, activation='relu'),
        layers.Dense(1000, activation='relu'),
        layers.Dense(10, activation='sigmoid')
    ])

ann.compile(optimizer='SGD',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

ann.fit(X_train, y_train, epochs=5)
```

```
Epoch 1/5
 888/1563 [================>…] - ETA: 25s - loss: nan - accuracy:
0.0991
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
Cell In[147], line 12
      1 ann = models.Sequential([
      2         layers.Flatten(input_shape=(32,32,3)),
      3         layers.Dense(3000, activation='relu'),
      4         layers.Dense(1000, activation='relu'),
      5         layers.Dense(10, activation='sigmoid')
      6     ])
      8 ann.compile(optimizer='SGD',
      9               loss='sparse_categorical_crossentropy',
     10               metrics=['accuracy'])
---> 12 ann.fit(X_train, y_train, epochs=5)

File␣
 ↪~\AppData\Roaming\Python\Python311\site-packages\keras\src\utils\traceback_ut ls.
 ↪py:65, in filter_traceback.<locals>.error_handler(*args, **kwargs)
     63 filtered_tb = None
     64 try:
---> 65     return fn(*args, **kwargs)
     66 except Exception as e:
     67     filtered_tb = _process_traceback_frames(e.__traceback__)

File ~\AppData\Roaming\Python\Python311\site-packages\keras\src\engine\training
 ↪py:1807, in Model.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)
   1799 with tf.profiler.experimental.Trace(
   1800     "train",
   1801     epoch_num=epoch,
   (…)
   1804     _r=1,
```

```
   1805 ):
   1806     callbacks.on_train_batch_begin(step)
-> 1807     tmp_logs = self.train_function(iterator)
   1808     if data_handler.should_sync:
   1809         context.async_wait()

File␣
 ↪~\AppData\Roaming\Python\Python311\site-packages\tensorflow\python\util\trace ack_utils.
 ↪py:150, in filter_traceback.<locals>.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__)

File␣
 ↪~\AppData\Roaming\Python\Python311\site-packages\tensorflow\python\eager\poly morphic_functi
 ↪py:832, in Function.__call__(self, *args, **kwds)
    829 compiler = "xla" if self._jit_compile else "nonXla"
    831 with OptionalXlaContext(self._jit_compile):
--> 832   result = self._call(*args, **kwds)
    834 new_tracing_count = self.experimental_get_tracing_count()
    835 without_tracing = (tracing_count == new_tracing_count)

File␣
 ↪~\AppData\Roaming\Python\Python311\site-packages\tensorflow\python\eager\poly morphic_functi
 ↪py:868, in Function._call(self, *args, **kwds)
    865   self._lock.release()
    866   # In this case we have created variables on the first call, so we run␣
 ↪the
    867   # defunned version which is guaranteed to never create variables.
--> 868   return tracing_compilation.call_function(
    869       args, kwds, self._no_variable_creation_config
    870   )
    871 elif self._variable_creation_config is not None:
    872   # Release the lock early so that multiple threads can perform the cal
    873   # in parallel.
    874   self._lock.release()

File␣
 ↪~\AppData\Roaming\Python\Python311\site-packages\tensorflow\python\eager\poly morphic_functi
 ↪py:139, in call_function(args, kwargs, tracing_options)
    137 bound_args = function.function_type.bind(*args, **kwargs)
    138 flat_inputs = function.function_type.unpack_inputs(bound_args)
--> 139 return function._call_flat(  # pylint: disable=protected-access
    140     flat_inputs, captured_inputs=function.captured_inputs
    141 )
```

```
File␣
 ↪~\AppData\Roaming\Python\Python311\site-packages\tensorflow\python\eager\poly morphic_functi
 ↪py:1323, in ConcreteFunction._call_flat(self, tensor_inputs, captured_inputs)
  1319 possible_gradient_type = gradients_util.PossibleTapeGradientTypes(args)
  1320 if (possible_gradient_type == gradients_util.POSSIBLE_GRADIENT_TYPES_NO E
  1321     and executing_eagerly):
  1322   # No tape is watching; skip to running the function.
-> 1323   return self._inference_function.call_preflattened(args)
  1324 forward_backward = self._select_forward_and_backward_functions(
  1325     args,
  1326     possible_gradient_type,
  1327     executing_eagerly)
  1328 forward_function, args_with_tangents = forward_backward.forward()

File␣
 ↪~\AppData\Roaming\Python\Python311\site-packages\tensorflow\python\eager\poly morphic_functi
 ↪py:216, in AtomicFunction.call_preflattened(self, args)
    214 def call_preflattened(self, args: Sequence[core.Tensor]) -> Any:
    215   """Calls with flattened tensor inputs and returns the structured␣
 ↪output."""
--> 216   flat_outputs = self.call_flat(*args)
    217   return self.function_type.pack_output(flat_outputs)

File␣
 ↪~\AppData\Roaming\Python\Python311\site-packages\tensorflow\python\eager\poly morphic_functi
 ↪py:251, in AtomicFunction.call_flat(self, *args)
    249 with record.stop_recording():
    250   if self._bound_context.executing_eagerly():
--> 251     outputs = self._bound_context.call_function(
    252         self.name,
    253         list(args),
    254         len(self.function_type.flat_outputs),
    255     )
    256   else:
    257     outputs = make_call_op_in_graph(
    258         self,
    259         list(args),
    260         self._bound_context.function_call_options.as_attrs(),
    261     )

File␣
 ↪~\AppData\Roaming\Python\Python311\site-packages\tensorflow\python\eager\cont xt.
 ↪py:1486, in Context.call_function(self, name, tensor_inputs, num_outputs)
  1484 cancellation_context = cancellation.context()
  1485 if cancellation_context is None:
-> 1486   outputs = execute.execute(
  1487       name.decode("utf-8"),
  1488       num_outputs=num_outputs,
  1489       inputs=tensor_inputs,
```

```
   1490       attrs=attrs,
   1491       ctx=self,
   1492   )
   1493 else:
   1494   outputs = execute.execute_with_cancellation(
   1495       name.decode("utf-8"),
   1496       num_outputs=num_outputs,
   (…)
   1500       cancellation_manager=cancellation_context,
   1501   )

File␣
 ↪~\AppData\Roaming\Python\Python311\site-packages\tensorflow\python\eager\execute.
 ↪py:53, in quick_execute(op_name, num_outputs, inputs, attrs, ctx, name)
     51 try:
     52   ctx.ensure_initialized()
---> 53   tensors = pywrap_tfe.TFE_Py_Execute(ctx._handle, device_name, op_name
     54                                       inputs, attrs, num_outputs)
     55 except core._NotOkStatusException as e:
     56   if name is not None:

KeyboardInterrupt:
```

[148]:
```
from sklearn.metrics import confusion_matrix , classification_report
import numpy as np
y_pred = ann.predict(X_test)
y_pred_classes = [np.argmax(element) for element in y_pred]

print("Classification Report: \n", classification_report(y_test,␣
 ↪y_pred_classes))
```

```
313/313 [==============================] - 2s 7ms/step
Classification Report:
              precision    recall  f1-score   support

           0       0.10      1.00      0.18      1000
           1       0.00      0.00      0.00      1000
           2       0.00      0.00      0.00      1000
           3       0.00      0.00      0.00      1000
           4       0.00      0.00      0.00      1000
           5       0.00      0.00      0.00      1000
           6       0.00      0.00      0.00      1000
           7       0.00      0.00      0.00      1000
           8       0.00      0.00      0.00      1000
           9       0.00      0.00      0.00      1000

    accuracy                           0.10     10000
```

```
   macro avg       0.01      0.10      0.02     10000
weighted avg       0.01      0.10      0.02     10000
```

```
D:\anaconda\Lib\site-packages\sklearn\metrics\_classification.py:1469:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
D:\anaconda\Lib\site-packages\sklearn\metrics\_classification.py:1469:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
D:\anaconda\Lib\site-packages\sklearn\metrics\_classification.py:1469:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

[149]:
```
cnn = models.Sequential([
    layers.Conv2D(filters=32, kernel_size=(3, 3), activation='relu',␣
 ↪input_shape=(32, 32, 3)),
    layers.MaxPooling2D((2, 2)),

    layers.Conv2D(filters=64, kernel_size=(3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),

    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax')
])
```

```
WARNING:tensorflow:From C:\Users\Pratik
Nagare\AppData\Roaming\Python\Python311\site-
packages\keras\src\layers\pooling\max_pooling2d.py:161: The name tf.nn.max_pool
is deprecated. Please use tf.nn.max_pool2d instead.
```

[150]:
```
cnn.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

[151]:
```
cnn.fit(X_train, y_train, epochs=10)
```

```
Epoch 1/10
1563/1563 [==============================] - 17s 9ms/step - loss: 2.7309 -
accuracy: 0.1037
```

```
Epoch 2/10
1563/1563 [==============================] - 15s 10ms/step - loss: 2.3028 -
accuracy: 0.0984
Epoch 3/10
1563/1563 [==============================] - 15s 10ms/step - loss: 2.3028 -
accuracy: 0.0990
Epoch 4/10
1563/1563 [==============================] - 16s 10ms/step - loss: 2.3050 -
accuracy: 0.0996
Epoch 5/10
1563/1563 [==============================] - 16s 10ms/step - loss: 2.3028 -
accuracy: 0.0994
Epoch 6/10
1563/1563 [==============================] - 17s 11ms/step - loss: 2.3028 -
accuracy: 0.0988
Epoch 7/10
1563/1563 [==============================] - 16s 11ms/step - loss: 2.3028 -
accuracy: 0.0983
Epoch 8/10
1563/1563 [==============================] - 15s 9ms/step - loss: 2.3028 -
accuracy: 0.0978
Epoch 9/10
1563/1563 [==============================] - 15s 10ms/step - loss: 2.3028 -
accuracy: 0.0957
Epoch 10/10
1563/1563 [==============================] - 16s 10ms/step - loss: 2.3028 -
accuracy: 0.0990
```

[151]:
```
<keras.src.callbacks.History at 0x1ea900503d0>
```

[152]:
```
cnn.evaluate(X_test,y_test)
```

```
313/313 [==============================] - 1s 4ms/step - loss: 2.3026 -
accuracy: 0.1000
```

[152]:
```
[2.3026442527770996, 0.10000000149011612]
```

[153]:
```
y_pred = cnn.predict(X_test)
y_pred[:5]
```

```
313/313 [==============================] - 1s 4ms/step
```

[153]:
```
array([[0.09971637, 0.10177267, 0.09745668, 0.09848896, 0.10034841,
        0.10028911, 0.10042525, 0.10147531, 0.09965564, 0.10037153],
       [0.09971637, 0.10177267, 0.09745668, 0.09848896, 0.10034841,
        0.10028911, 0.10042525, 0.10147531, 0.09965564, 0.10037153],
       [0.09971637, 0.10177267, 0.09745668, 0.09848896, 0.10034841,
        0.10028911, 0.10042525, 0.10147531, 0.09965564, 0.10037153],
```

```
       [0.09971637, 0.10177267, 0.09745668, 0.09848896, 0.10034841,
        0.10028911, 0.10042525, 0.10147531, 0.09965564, 0.10037153],
       [0.09971637, 0.10177267, 0.09745668, 0.09848896, 0.10034841,
        0.10028911, 0.10042525, 0.10147531, 0.09965564, 0.10037153]],
      dtype=float32)
```

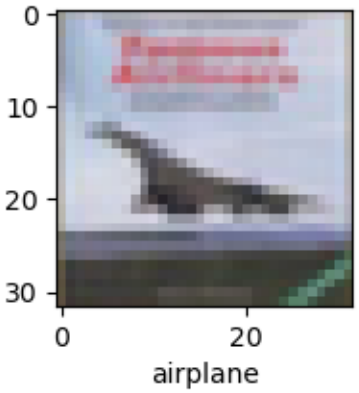
[154]:
```
y_classes = [np.argmax(element) for element in y_pred]
y_classes[:5]
```

[154]:
```
[1, 1, 1, 1, 1]
```

[156]:
```
y_test[:5]
```

[156]:
```
array([3, 8, 8, 0, 6], dtype=uint8)
```

[166]:
```
plot_sample(X_test, y_test,3)
```



[165]:
```
classes[y_classes[3]]
```

[165]:
```
'automobile'
```

[164]:
```
classes[y_classes[3]]
```

[164]:
```
'automobile'
```

[ ]:

[ ]:

[ ]: