In this lab, we will learn how to use the Keras library to build convolutional neural networks. We will also use the popular MNIST dataset and we will compare our results to using a conventional neural network.

Objective for this Notebook
1. How to use the Keras library to build convolutional neural networks. 2. Convolutional Neural Network with One Convolutional and Pooling Layers. 3. Convolutional Neural Network with Two Convolutional and Pooling Layers.

Table of Contents

Import Keras and Packages

Let's start by importing the keras libraries and the packages that we would need to build a neural network.

```
# All Libraries required for this lab are listed below. The libraries pre-installed on Skills Network Labs are commented.
# If you run this notebook on a different environment, e.g. your desktop, you may need to uncomment and install certain libraries.

!pip install numpy==1.21.4
!pip install pandas==1.3.4
!pip install keras==2.1.6
```

```
Collecting numpy==1.21.4
  Downloading numpy-1.21.4-cp37-cp37m-
manylinux_2_12_x86_64.manylinux2010_x86_64.whl (15.7 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 15.7/15.7 MB 59.9 MB/s eta
0:00:0000:0100:01
py
  Attempting uninstall: numpy
    Found existing installation: numpy 1.21.6
    Uninstalling numpy-1.21.6:
      Successfully uninstalled numpy-1.21.6
Successfully installed numpy-1.21.4
Collecting pandas==1.3.4
  Downloading pandas-1.3.4-cp37-cp37m-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl (11.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 11.3/11.3 MB 64.9 MB/s eta
0:00:0000:0100:01
ent already satisfied: python-dateutil>=2.7.3 in
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from
pandas==1.3.4) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from
```

```
pandas==1.3.4) (2022.2.1)
Requirement already satisfied: numpy>=1.17.3 in
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from
pandas==1.3.4) (1.21.4)
Requirement already satisfied: six>=1.5 in
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from
python-dateutil>=2.7.3->pandas==1.3.4) (1.16.0)
Installing collected packages: pandas
  Attempting uninstall: pandas
    Found existing installation: pandas 1.3.5
    Uninstalling pandas-1.3.5:
      Successfully uninstalled pandas-1.3.5
Successfully installed pandas-1.3.4
Requirement already satisfied: keras==2.1.6 in
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (2.1.6)
Requirement already satisfied: six>=1.9.0 in
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from
keras==2.1.6) (1.16.0)
Requirement already satisfied: pyyaml in
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from
keras==2.1.6) (6.0)
Requirement already satisfied: h5py in
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from
keras==2.1.6) (2.8.0)
Requirement already satisfied: numpy>=1.9.1 in
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from
keras==2.1.6) (1.21.4)
Requirement already satisfied: scipy>=0.14 in
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from
keras==2.1.6) (1.7.3)
```

```
import warnings
warnings.filterwarnings('ignore')
```

```
import keras
from keras.models import Sequential
from keras.layers import Dense
from keras.utils import to_categorical
```

When working with convolutional neural networks in particular, we will need additional packages.

```
from keras.layers.convolutional import Conv2D # to add convolutional
layers
from keras.layers.convolutional import MaxPooling2D # to add pooling
layers
from keras.layers import Flatten # to flatten data for fully connected
layers
```

Convolutional Layer with One set of convolutional and pooling layers

```
# import data
from keras.datasets import mnist

# load data
(X_train, y_train), (X_test, y_test) = mnist.load_data()

# reshape to be [samples][pixels][width][height]
X_train = X_train.reshape(X_train.shape[0], 28, 28,
1).astype('float32')
X_test = X_test.reshape(X_test.shape[0], 28, 28, 1).astype('float32')
```

```
Downloading data from https://s3.amazonaws.com/img-datasets/mnist.npz
11493376/11490434 [==============================] - 0s 0us/step
```

Let's normalize the pixel values to be between 0 and 1

```
X_train = X_train / 255 # normalize training data
X_test = X_test / 255 # normalize test data
```

Next, let's convert the target variable into binary categories

```
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)

num_classes = y_test.shape[1] # number of categories
```

Next, let's define a function that creates our model. Let's start with one set of convolutional and pooling layers.

```
def convolutional_model():

    # create model
    model = Sequential()
    model.add(Conv2D(16, (5, 5), strides=(1, 1), activation='relu',
input_shape=(28, 28, 1)))
    model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))

    model.add(Flatten())
    model.add(Dense(100, activation='relu'))
    model.add(Dense(num_classes, activation='softmax'))

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

Finally, let's call the function to create the model, and then let's train it and evaluate it.

```
# build the model
model = convolutional_model()
```

```
# fit the model
model.fit(X_train, y_train, validation_data=(X_test, y_test),
epochs=10, batch_size=200, verbose=2)

# evaluate the model
scores = model.evaluate(X_test, y_test, verbose=0)
print("Accuracy: {} \n Error: {}".format(scores[1], 100-
scores[1]*100))
```

```
WARNING:tensorflow:From
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages/keras/
backend/tensorflow_backend.py:68: The name tf.get_default_graph is
deprecated. Please use tf.compat.v1.get_default_graph instead.

WARNING:tensorflow:From
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages/keras/
backend/tensorflow_backend.py:508: The name tf.placeholder is
deprecated. Please use tf.compat.v1.placeholder instead.

WARNING:tensorflow:From
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages/keras/
backend/tensorflow_backend.py:3837: The name tf.random_uniform is
deprecated. Please use tf.random.uniform instead.

WARNING:tensorflow:From
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages/keras/
backend/tensorflow_backend.py:3661: The name tf.nn.max_pool is
deprecated. Please use tf.nn.max_pool2d instead.

WARNING:tensorflow:From
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages/keras/
optimizers.py:757: The name tf.train.Optimizer is deprecated. Please
use tf.compat.v1.train.Optimizer instead.

WARNING:tensorflow:From
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages/keras/
backend/tensorflow_backend.py:3014: The name tf.log is deprecated.
Please use tf.math.log instead.

WARNING:tensorflow:From
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages/tensorf
low/python/ops/math_grad.py:1250:
add_dispatch_support.<locals>.wrapper (from
tensorflow.python.ops.array_ops) is deprecated and will be removed in
a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
WARNING:tensorflow:From
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages/keras/
```

```
backend/tensorflow_backend.py:977: The name tf.assign_add is
deprecated. Please use tf.compat.v1.assign_add instead.

Train on 60000 samples, validate on 10000 samples
Epoch 1/10

2022-10-19 16:51:53.074642: I
tensorflow/core/platform/cpu_feature_guard.cc:142] Your CPU supports
instructions that this TensorFlow binary was not compiled to use:
SSE4.1 SSE4.2 AVX AVX2 AVX512F FMA
2022-10-19 16:51:53.084213: I
tensorflow/core/platform/profile_utils/cpu_utils.cc:94] CPU Frequency:
2593905000 Hz
2022-10-19 16:51:53.085137: I
tensorflow/compiler/xla/service/service.cc:168] XLA service
0x562d0c37d330 executing computations on platform Host. Devices:
2022-10-19 16:51:53.085210: I
tensorflow/compiler/xla/service/service.cc:175]   StreamExecutor
device (0): <undefined>, <undefined>
2022-10-19 16:51:53.195225: W
tensorflow/compiler/jit/mark_for_compilation_pass.cc:1412] (One-time
warning): Not using XLA:CPU for cluster because envvar TF_XLA_FLAGS=--
tf_xla_cpu_global_jit was not set.  If you want XLA:CPU, either set
that envvar, or use experimental_jit_scope to enable XLA:CPU.  To
confirm that XLA is active, pass --vmodule=xla_compilation_cache=1 (as
a proper command-line flag, not via TF_XLA_FLAGS) or set the envvar
XLA_FLAGS=--xla_hlo_profile.

 - 41s - loss: 0.2678 - acc: 0.9293 - val_loss: 0.0892 - val_acc:
0.9716
Epoch 2/10
 - 39s - loss: 0.0765 - acc: 0.9782 - val_loss: 0.0546 - val_acc:
0.9831
Epoch 3/10
 - 39s - loss: 0.0537 - acc: 0.9843 - val_loss: 0.0463 - val_acc:
0.9857
Epoch 4/10
 - 38s - loss: 0.0422 - acc: 0.9876 - val_loss: 0.0435 - val_acc:
0.9852
Epoch 5/10
 - 39s - loss: 0.0337 - acc: 0.9903 - val_loss: 0.0448 - val_acc:
0.9854
Epoch 6/10
 - 38s - loss: 0.0278 - acc: 0.9923 - val_loss: 0.0394 - val_acc:
0.9863
Epoch 7/10
 - 39s - loss: 0.0228 - acc: 0.9933 - val_loss: 0.0434 - val_acc:
0.9865
Epoch 8/10
 - 38s - loss: 0.0194 - acc: 0.9940 - val_loss: 0.0378 - val_acc:
```

```
0.9885
Epoch 9/10
 - 38s - loss: 0.0162 - acc: 0.9953 - val_loss: 0.0458 - val_acc:
0.9872
Epoch 10/10
 - 39s - loss: 0.0122 - acc: 0.9965 - val_loss: 0.0360 - val_acc:
0.9878
Accuracy: 0.9878
 Error: 1.2199999999999989
```

Convolutional Layer with two sets of convolutional and pooling layers

Let's redefine our convolutional model so that it has two convolutional and pooling layers instead of just one layer of each.

```
def convolutional_model():

    # create model
    model = Sequential()
    model.add(Conv2D(16, (5, 5), activation='relu', input_shape=(28,
28, 1)))
    model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))

    model.add(Conv2D(8, (2, 2), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))

    model.add(Flatten())
    model.add(Dense(100, activation='relu'))
    model.add(Dense(num_classes, activation='softmax'))

    # Compile model
    model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])
    return model
```

Now, let's call the function to create our new convolutional neural network, and then let's train it and evaluate it.

```
# build the model
model = convolutional_model()

# fit the model
model.fit(X_train, y_train, validation_data=(X_test, y_test),
epochs=10, batch_size=200, verbose=2)

# evaluate the model
scores = model.evaluate(X_test, y_test, verbose=0)
```

```
print("Accuracy: {} \n Error: {}".format(scores[1], 100-
scores[1]*100))
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/10
 - 41s - loss: 0.4767 - acc: 0.8632 - val_loss: 0.1346 - val_acc:
0.9630
Epoch 2/10
 - 41s - loss: 0.1216 - acc: 0.9636 - val_loss: 0.0820 - val_acc:
0.9756
Epoch 3/10
 - 41s - loss: 0.0841 - acc: 0.9742 - val_loss: 0.0612 - val_acc:
0.9818
Epoch 4/10
 - 42s - loss: 0.0675 - acc: 0.9793 - val_loss: 0.0568 - val_acc:
0.9812
Epoch 5/10
 - 41s - loss: 0.0571 - acc: 0.9827 - val_loss: 0.0526 - val_acc:
0.9834
Epoch 6/10
 - 41s - loss: 0.0515 - acc: 0.9838 - val_loss: 0.0434 - val_acc:
0.9859
Epoch 7/10
 - 41s - loss: 0.0448 - acc: 0.9862 - val_loss: 0.0370 - val_acc:
0.9893
Epoch 8/10
 - 41s - loss: 0.0397 - acc: 0.9875 - val_loss: 0.0382 - val_acc:
0.9881
Epoch 9/10
 - 41s - loss: 0.0372 - acc: 0.9885 - val_loss: 0.0363 - val_acc:
0.9877
Epoch 10/10
```

Thank you for completing this lab!

This notebook was created by Alex Aklson. I hope you found this lab interesting and educational. Feel free to contact me if you have any questions!

Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2020-09-21	2.0	Srishti	Migrated Lab to Markdown and added to course repo in GitLab