08_svhn_object_detection

September 29, 2021

1 Object Detection with Street View House Numbers

This notebook illustrates how to build a deep CNN using Keras' functional API to generate multiple outputs: one to predict how many digits are present, and five for the value of each in the order they appear.

1.1 Imports & Settings

[1]:
```
import pandas as pd
import numpy as np
from keras.utils import to_categorical
from sklearn.metrics import confusion_matrix
from keras.models import Sequential, Model
from keras.callbacks import ModelCheckpoint
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Input, Dropout,␣
↪BatchNormalization, Activation
```

```
Using TensorFlow backend.
```

1.2 Best Architecture

Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks, Goodfellow, et al, 2014

- eight convolutional hidden layers,
- one locally connected hidden layer
- two densely connected hidden layers.
- the first hidden layer contains maxout units with three filters per unit
- the others contain rectifier units
- the number of units is [48, 64, 128, 160] for the first four layers
- 192 for all other locally connected layers
- the fully connected layers contain 3,072 units each.
- Each convolutional layer includes max pooling and subtractive normalization
- The max pooling window size is 2×2.
- The stride alternates between 2 and 1 at each layer, so that half of the layers don't reduce the spatial size of the representation
- All convolutions use zero padding on the input to preserve representation size.
- The subtractive normalization operates on 3x3 windows and preserves representation size.
- All convolution kernels were of size 5×5.

- We trained with dropout applied to all hidden layers but not the input.

The best-performing architecture on the original dataset has eight convolutional layers and two final fully-connected layers. The convolutional layers are similar so that we can define a function to simplify their creation:

[2]:
```
def svhn_layer(model, filters, strides, n, input_shape=None):
    if input_shape is not None:
        model.add(Conv2D(filters, kernel_size=5, padding='same', name='CONV{}'.
 ↪format(n), input_shape=input_shape))
    else:
        model.add(Conv2D(filters, kernel_size=5, padding='same',␣
 ↪activation='relu', name='CONV{}'.format(n)))
    model.add(BatchNormalization(name='NORM{}'.format(n)))
    model.add(MaxPooling2D(pool_size=2, strides=strides, name='POOL{}'.
 ↪format(n)))
    model.add(Dropout(0.2, name='DROP{}'.format(n)))
    return model
```

The entire model combines the Sequential and functional API as follows:

[3]:
```
model = Sequential()

svhn_layer(model, 48, 1, n=1, input_shape=(32,32,1))

for i, kernel in enumerate([48, 64, 128, 160] + 3 * [192], 2):
    svhn_layer(model, kernel, strides=2 if i % 2 == 0 else 1, n=i)

model.add(Flatten())
model.add(Dense(3072, name='FC1'))
model.add(Dense(3072, name='FC2'))
y = model.output

n_digits = (Dense(units=6, activation='softmax'))(y)
digit1 = (Dense(units=10, activation='softmax'))(y)
digit2 = (Dense(units=11, activation='softmax'))(y)
digit3 = (Dense(units=11, activation='softmax'))(y)
digit4 = (Dense(units=11, activation='softmax'))(y)
digit5 = (Dense(units=11, activation='softmax'))(y)

svhn_model = Model(inputs=model.input, outputs=[n_digits, digit1, digit2,␣
 ↪digit3, digit4, digit5])
```

As a result, the model produces six distinct outputs that we can evaluate.

[4]:
```
svhn_model.summary()
```

```
_________________________________________________________________________________
_________________
```

```
Layer (type)                 Output Shape              Param #     Connected to
==================================================================================
==================
CONV1_input (InputLayer)     (None, 32, 32, 1)         0
----------------------------------------------------------------------------------
------------------
CONV1 (Conv2D)               (None, 32, 32, 48)        1248
CONV1_input[0][0]
----------------------------------------------------------------------------------
------------------
NORM1 (BatchNormalization)   (None, 32, 32, 48)        192         CONV1[0][0]
----------------------------------------------------------------------------------
------------------
POOL1 (MaxPooling2D)         (None, 31, 31, 48)        0           NORM1[0][0]
----------------------------------------------------------------------------------
------------------
DROP1 (Dropout)              (None, 31, 31, 48)        0           POOL1[0][0]
----------------------------------------------------------------------------------
------------------
CONV2 (Conv2D)               (None, 31, 31, 48)        57648       DROP1[0][0]
----------------------------------------------------------------------------------
------------------
NORM2 (BatchNormalization)   (None, 31, 31, 48)        192         CONV2[0][0]
----------------------------------------------------------------------------------
------------------
POOL2 (MaxPooling2D)         (None, 15, 15, 48)        0           NORM2[0][0]
----------------------------------------------------------------------------------
------------------
DROP2 (Dropout)              (None, 15, 15, 48)        0           POOL2[0][0]
----------------------------------------------------------------------------------
------------------
CONV3 (Conv2D)               (None, 15, 15, 64)        76864       DROP2[0][0]
----------------------------------------------------------------------------------
------------------
NORM3 (BatchNormalization)   (None, 15, 15, 64)        256         CONV3[0][0]
----------------------------------------------------------------------------------
------------------
POOL3 (MaxPooling2D)         (None, 14, 14, 64)        0           NORM3[0][0]
----------------------------------------------------------------------------------
------------------
DROP3 (Dropout)              (None, 14, 14, 64)        0           POOL3[0][0]
----------------------------------------------------------------------------------
------------------
CONV4 (Conv2D)               (None, 14, 14, 128)       204928      DROP3[0][0]
----------------------------------------------------------------------------------
------------------
NORM4 (BatchNormalization)   (None, 14, 14, 128)       512         CONV4[0][0]
----------------------------------------------------------------------------------
```

```
------------------
POOL4 (MaxPooling2D)            (None, 7, 7, 128)    0           NORM4[0][0]
--------------------------------------------------------------------------------
------------------
DROP4 (Dropout)                 (None, 7, 7, 128)    0           POOL4[0][0]
--------------------------------------------------------------------------------
------------------
CONV5 (Conv2D)                  (None, 7, 7, 160)    512160      DROP4[0][0]
--------------------------------------------------------------------------------
------------------
NORM5 (BatchNormalization)      (None, 7, 7, 160)    640         CONV5[0][0]
--------------------------------------------------------------------------------
------------------
POOL5 (MaxPooling2D)            (None, 6, 6, 160)    0           NORM5[0][0]
--------------------------------------------------------------------------------
------------------
DROP5 (Dropout)                 (None, 6, 6, 160)    0           POOL5[0][0]
--------------------------------------------------------------------------------
------------------
CONV6 (Conv2D)                  (None, 6, 6, 192)    768192      DROP5[0][0]
--------------------------------------------------------------------------------
------------------
NORM6 (BatchNormalization)      (None, 6, 6, 192)    768         CONV6[0][0]
--------------------------------------------------------------------------------
------------------
POOL6 (MaxPooling2D)            (None, 3, 3, 192)    0           NORM6[0][0]
--------------------------------------------------------------------------------
------------------
DROP6 (Dropout)                 (None, 3, 3, 192)    0           POOL6[0][0]
--------------------------------------------------------------------------------
------------------
CONV7 (Conv2D)                  (None, 3, 3, 192)    921792      DROP6[0][0]
--------------------------------------------------------------------------------
------------------
NORM7 (BatchNormalization)      (None, 3, 3, 192)    768         CONV7[0][0]
--------------------------------------------------------------------------------
------------------
POOL7 (MaxPooling2D)            (None, 2, 2, 192)    0           NORM7[0][0]
--------------------------------------------------------------------------------
------------------
DROP7 (Dropout)                 (None, 2, 2, 192)    0           POOL7[0][0]
--------------------------------------------------------------------------------
------------------
CONV8 (Conv2D)                  (None, 2, 2, 192)    921792      DROP7[0][0]
--------------------------------------------------------------------------------
------------------
NORM8 (BatchNormalization)      (None, 2, 2, 192)    768         CONV8[0][0]
--------------------------------------------------------------------------------
```

```
__________________
POOL8 (MaxPooling2D)            (None, 1, 1, 192)    0           NORM8[0][0]
____________________________________________________________________________
__________________
DROP8 (Dropout)                 (None, 1, 1, 192)    0           POOL8[0][0]
____________________________________________________________________________
__________________
flatten_1 (Flatten)             (None, 192)          0           DROP8[0][0]
____________________________________________________________________________
__________________
FC1 (Dense)                     (None, 3072)         592896      flatten_1[0][0]
____________________________________________________________________________
__________________
FC2 (Dense)                     (None, 3072)         9440256     FC1[0][0]
____________________________________________________________________________
__________________
dense_1 (Dense)                 (None, 6)            18438       FC2[0][0]
____________________________________________________________________________
__________________
dense_2 (Dense)                 (None, 10)           30730       FC2[0][0]
____________________________________________________________________________
__________________
dense_3 (Dense)                 (None, 11)           33803       FC2[0][0]
____________________________________________________________________________
__________________
dense_4 (Dense)                 (None, 11)           33803       FC2[0][0]
____________________________________________________________________________
__________________
dense_5 (Dense)                 (None, 11)           33803       FC2[0][0]
____________________________________________________________________________
__________________
dense_6 (Dense)                 (None, 11)           33803       FC2[0][0]
============================================================================
==================
Total params: 13,686,252
Trainable params: 13,684,204
Non-trainable params: 2,048
____________________________________________________________________________
__________________
```

1.2.1 Get Data

[5]:
```
svhn_model.compile(optimizer='adam',
                   loss='categorical_crossentropy',
                   metrics=["accuracy"])
```

[6]:
```
with pd.HDFStore('images/svhn/data.h5') as store:
    X_train = store['train/data'].values.reshape(-1, 32, 32, 1)
    y_train = store['train/labels']
    X_test = store['test/data'].values.reshape(-1, 32, 32, 1)
    y_test = store['test/labels']
```

[7]:
```
train_digits = [to_categorical(d) for d in y_train.values.T]
test_digits = [to_categorical(d) for d in y_test.values.T]
```

[8]:
```
svhn_path = 'models/svhn.cnn.weights.best.hdf5'
```

[9]:
```
checkpointer = ModelCheckpoint(filepath=svhn_path,
                               verbose=1,
                               save_best_only=True)
```

[10]:
```
epochs = 25
result = svhn_model.fit(x=X_train,
                        y=train_digits,
                        batch_size=32,
                        epochs=epochs,
                        verbose=1,
                        validation_data=(X_test, test_digits),
                        callbacks=[checkpointer])
```

```
/home/stefan/.pyenv/versions/miniconda3-latest/envs/ml4t/lib/python3.6/site-
packages/ipykernel_launcher.py:8: UserWarning: The `nb_epoch` argument in `fit`
has been renamed `epochs`.


Train on 33401 samples, validate on 13068 samples
Epoch 1/25
33401/33401 [==============================] - 314s 9ms/step - loss: 9.7087 -
dense_1_loss: 1.5593 - dense_2_loss: 3.0150 - dense_3_loss: 3.1225 -
dense_4_loss: 1.5303 - dense_5_loss: 0.4141 - dense_6_loss: 0.0675 -
dense_1_acc: 0.6416 - dense_2_acc: 0.2507 - dense_3_acc: 0.1959 - dense_4_acc:
0.6649 - dense_5_acc: 0.9424 - dense_6_acc: 0.9931 - val_loss: 7.5946 -
val_dense_1_loss: 1.0219 - val_dense_2_loss: 3.0613 - val_dense_3_loss: 2.4888 -
val_dense_4_loss: 0.8573 - val_dense_5_loss: 0.1596 - val_dense_6_loss: 0.0057 -
val_dense_1_acc: 0.7795 - val_dense_2_acc: 0.1369 - val_dense_3_acc: 0.2273 -
val_dense_4_acc: 0.7792 - val_dense_5_acc: 0.9887 - val_dense_6_acc: 0.9998

Epoch 00001: val_loss improved from inf to 7.59463, saving model to
models/svhn.cnn.weights.best.hdf5
Epoch 2/25
33401/33401 [==============================] - 347s 10ms/step - loss: 10.0200 -
dense_1_loss: 1.5287 - dense_2_loss: 2.9456 - dense_3_loss: 3.1548 -
dense_4_loss: 1.8049 - dense_5_loss: 0.5168 - dense_6_loss: 0.0692 -
dense_1_acc: 0.6140 - dense_2_acc: 0.2336 - dense_3_acc: 0.1955 - dense_4_acc:
```

```
0.6667 - dense_5_acc: 0.9380 - dense_6_acc: 0.9928 - val_loss: 6.2765 -
val_dense_1_loss: 0.9886 - val_dense_2_loss: 2.0373 - val_dense_3_loss: 2.3389 -
val_dense_4_loss: 0.8177 - val_dense_5_loss: 0.0916 - val_dense_6_loss: 0.0025 -
val_dense_1_acc: 0.5766 - val_dense_2_acc: 0.2546 - val_dense_3_acc: 0.1850 -
val_dense_4_acc: 0.8272 - val_dense_5_acc: 0.9887 - val_dense_6_acc: 0.9998

Epoch 00002: val_loss improved from 7.59463 to 6.27648, saving model to
models/svhn.cnn.weights.best.hdf5
Epoch 3/25
33401/33401 [==============================] - 300s 9ms/step - loss: 12.0488 -
dense_1_loss: 1.9204 - dense_2_loss: 3.4604 - dense_3_loss: 3.4291 -
dense_4_loss: 2.2792 - dense_5_loss: 0.7772 - dense_6_loss: 0.1824 -
dense_1_acc: 0.5915 - dense_2_acc: 0.2173 - dense_3_acc: 0.1894 - dense_4_acc:
0.6448 - dense_5_acc: 0.9261 - dense_6_acc: 0.9866 - val_loss: 7.0373 -
val_dense_1_loss: 1.1656 - val_dense_2_loss: 2.3608 - val_dense_3_loss: 2.4869 -
val_dense_4_loss: 0.9296 - val_dense_5_loss: 0.0921 - val_dense_6_loss: 0.0023 -
val_dense_1_acc: 0.7186 - val_dense_2_acc: 0.2559 - val_dense_3_acc: 0.2228 -
val_dense_4_acc: 0.8205 - val_dense_5_acc: 0.9886 - val_dense_6_acc: 0.9998

Epoch 00003: val_loss did not improve from 6.27648
Epoch 4/25
33401/33401 [==============================] - 300s 9ms/step - loss: 10.5717 -
dense_1_loss: 1.6174 - dense_2_loss: 2.9550 - dense_3_loss: 3.1532 -
dense_4_loss: 2.0699 - dense_5_loss: 0.6426 - dense_6_loss: 0.1336 -
dense_1_acc: 0.5945 - dense_2_acc: 0.2221 - dense_3_acc: 0.1899 - dense_4_acc:
0.6535 - dense_5_acc: 0.9386 - dense_6_acc: 0.9892 - val_loss: 6.7318 -
val_dense_1_loss: 0.8219 - val_dense_2_loss: 2.3542 - val_dense_3_loss: 2.4985 -
val_dense_4_loss: 0.9514 - val_dense_5_loss: 0.1035 - val_dense_6_loss: 0.0023 -
val_dense_1_acc: 0.7293 - val_dense_2_acc: 0.1330 - val_dense_3_acc: 0.2228 -
val_dense_4_acc: 0.8186 - val_dense_5_acc: 0.9868 - val_dense_6_acc: 0.9998

Epoch 00004: val_loss did not improve from 6.27648
Epoch 5/25
33401/33401 [==============================] - 301s 9ms/step - loss: 10.4707 -
dense_1_loss: 1.8261 - dense_2_loss: 2.9297 - dense_3_loss: 3.0503 -
dense_4_loss: 2.0377 - dense_5_loss: 0.5622 - dense_6_loss: 0.0648 -
dense_1_acc: 0.5889 - dense_2_acc: 0.2226 - dense_3_acc: 0.2073 - dense_4_acc:
0.6547 - dense_5_acc: 0.9358 - dense_6_acc: 0.9946 - val_loss: 5.7507 -
val_dense_1_loss: 0.6633 - val_dense_2_loss: 2.0542 - val_dense_3_loss: 2.1970 -
val_dense_4_loss: 0.7546 - val_dense_5_loss: 0.0791 - val_dense_6_loss: 0.0025 -
val_dense_1_acc: 0.7681 - val_dense_2_acc: 0.2820 - val_dense_3_acc: 0.2464 -
val_dense_4_acc: 0.8292 - val_dense_5_acc: 0.9887 - val_dense_6_acc: 0.9998

Epoch 00005: val_loss improved from 6.27648 to 5.75069, saving model to
models/svhn.cnn.weights.best.hdf5
Epoch 6/25
33401/33401 [==============================] - 300s 9ms/step - loss: 9.0226 -
dense_1_loss: 1.4348 - dense_2_loss: 2.6869 - dense_3_loss: 3.0158 -
```

```
dense_4_loss: 1.5244 - dense_5_loss: 0.3444 - dense_6_loss: 0.0162 -
dense_1_acc: 0.5971 - dense_2_acc: 0.2392 - dense_3_acc: 0.1948 - dense_4_acc:
0.6762 - dense_5_acc: 0.9505 - dense_6_acc: 0.9982 - val_loss: 7.5383 -
val_dense_1_loss: 1.0520 - val_dense_2_loss: 2.1542 - val_dense_3_loss: 2.9518 -
val_dense_4_loss: 1.2616 - val_dense_5_loss: 0.1162 - val_dense_6_loss: 0.0025 -
val_dense_1_acc: 0.6340 - val_dense_2_acc: 0.2808 - val_dense_3_acc: 0.1517 -
val_dense_4_acc: 0.8238 - val_dense_5_acc: 0.9878 - val_dense_6_acc: 0.9998

Epoch 00006: val_loss did not improve from 5.75069
Epoch 7/25
33401/33401 [==============================] - 300s 9ms/step - loss: 9.2151 -
dense_1_loss: 1.3050 - dense_2_loss: 2.6196 - dense_3_loss: 3.0590 -
dense_4_loss: 1.6750 - dense_5_loss: 0.4976 - dense_6_loss: 0.0590 -
dense_1_acc: 0.6211 - dense_2_acc: 0.2426 - dense_3_acc: 0.1979 - dense_4_acc:
0.6724 - dense_5_acc: 0.9399 - dense_6_acc: 0.9945 - val_loss: 8.7602 -
val_dense_1_loss: 1.4513 - val_dense_2_loss: 2.5046 - val_dense_3_loss: 3.3595 -
val_dense_4_loss: 1.2598 - val_dense_5_loss: 0.1825 - val_dense_6_loss: 0.0025 -
val_dense_1_acc: 0.6394 - val_dense_2_acc: 0.2049 - val_dense_3_acc: 0.2309 -
val_dense_4_acc: 0.8176 - val_dense_5_acc: 0.9887 - val_dense_6_acc: 0.9998

Epoch 00007: val_loss did not improve from 5.75069
Epoch 8/25
33401/33401 [==============================] - 300s 9ms/step - loss: 9.2256 -
dense_1_loss: 1.3348 - dense_2_loss: 2.7656 - dense_3_loss: 2.9522 -
dense_4_loss: 1.6488 - dense_5_loss: 0.4458 - dense_6_loss: 0.0785 -
dense_1_acc: 0.6076 - dense_2_acc: 0.2368 - dense_3_acc: 0.2079 - dense_4_acc:
0.6690 - dense_5_acc: 0.9459 - dense_6_acc: 0.9946 - val_loss: 5.7633 -
val_dense_1_loss: 0.6405 - val_dense_2_loss: 2.0585 - val_dense_3_loss: 2.2410 -
val_dense_4_loss: 0.7348 - val_dense_5_loss: 0.0862 - val_dense_6_loss: 0.0023 -
val_dense_1_acc: 0.7633 - val_dense_2_acc: 0.2815 - val_dense_3_acc: 0.2723 -
val_dense_4_acc: 0.8295 - val_dense_5_acc: 0.9887 - val_dense_6_acc: 0.9998

Epoch 00008: val_loss did not improve from 5.75069
Epoch 9/25
33401/33401 [==============================] - 300s 9ms/step - loss: 8.3413 -
dense_1_loss: 1.1721 - dense_2_loss: 2.5734 - dense_3_loss: 2.8296 -
dense_4_loss: 1.4378 - dense_5_loss: 0.3183 - dense_6_loss: 0.0101 -
dense_1_acc: 0.6214 - dense_2_acc: 0.2454 - dense_3_acc: 0.2163 - dense_4_acc:
0.6824 - dense_5_acc: 0.9518 - dense_6_acc: 0.9987 - val_loss: 6.1459 -
val_dense_1_loss: 0.6629 - val_dense_2_loss: 2.2397 - val_dense_3_loss: 2.3905 -
val_dense_4_loss: 0.7725 - val_dense_5_loss: 0.0778 - val_dense_6_loss: 0.0025 -
val_dense_1_acc: 0.7604 - val_dense_2_acc: 0.2866 - val_dense_3_acc: 0.2414 -
val_dense_4_acc: 0.8294 - val_dense_5_acc: 0.9887 - val_dense_6_acc: 0.9998

Epoch 00009: val_loss did not improve from 5.75069
Epoch 10/25
33401/33401 [==============================] - 301s 9ms/step - loss: 7.9722 -
dense_1_loss: 0.9641 - dense_2_loss: 2.5649 - dense_3_loss: 2.7847 -
```

```
dense_4_loss: 1.3546 - dense_5_loss: 0.2976 - dense_6_loss: 0.0062 -
dense_1_acc: 0.6604 - dense_2_acc: 0.2463 - dense_3_acc: 0.2215 - dense_4_acc:
0.6827 - dense_5_acc: 0.9528 - dense_6_acc: 0.9993 - val_loss: 5.9636 -
val_dense_1_loss: 0.5994 - val_dense_2_loss: 2.2489 - val_dense_3_loss: 2.2672 -
val_dense_4_loss: 0.7618 - val_dense_5_loss: 0.0838 - val_dense_6_loss: 0.0024 -
val_dense_1_acc: 0.7866 - val_dense_2_acc: 0.2130 - val_dense_3_acc: 0.2549 -
val_dense_4_acc: 0.8293 - val_dense_5_acc: 0.9887 - val_dense_6_acc: 0.9998

Epoch 00010: val_loss did not improve from 5.75069
Epoch 11/25
33401/33401 [==============================] - 301s 9ms/step - loss: 7.9460 -
dense_1_loss: 1.0460 - dense_2_loss: 2.5665 - dense_3_loss: 2.7105 -
dense_4_loss: 1.2992 - dense_5_loss: 0.3177 - dense_6_loss: 0.0061 -
dense_1_acc: 0.6582 - dense_2_acc: 0.2477 - dense_3_acc: 0.2201 - dense_4_acc:
0.6821 - dense_5_acc: 0.9508 - dense_6_acc: 0.9996 - val_loss: 5.9762 -
val_dense_1_loss: 0.6113 - val_dense_2_loss: 2.2477 - val_dense_3_loss: 2.3819 -
val_dense_4_loss: 0.6613 - val_dense_5_loss: 0.0718 - val_dense_6_loss: 0.0022 -
val_dense_1_acc: 0.7749 - val_dense_2_acc: 0.2903 - val_dense_3_acc: 0.2882 -
val_dense_4_acc: 0.8295 - val_dense_5_acc: 0.9887 - val_dense_6_acc: 0.9998

Epoch 00011: val_loss did not improve from 5.75069
Epoch 12/25
33401/33401 [==============================] - 299s 9ms/step - loss: 8.7124 -
dense_1_loss: 1.3090 - dense_2_loss: 2.6156 - dense_3_loss: 2.8611 -
dense_4_loss: 1.5322 - dense_5_loss: 0.3541 - dense_6_loss: 0.0404 -
dense_1_acc: 0.6048 - dense_2_acc: 0.2486 - dense_3_acc: 0.1979 - dense_4_acc:
0.6781 - dense_5_acc: 0.9508 - dense_6_acc: 0.9959 - val_loss: 5.8453 -
val_dense_1_loss: 0.6830 - val_dense_2_loss: 2.1058 - val_dense_3_loss: 2.2155 -
val_dense_4_loss: 0.7460 - val_dense_5_loss: 0.0926 - val_dense_6_loss: 0.0025 -
val_dense_1_acc: 0.7609 - val_dense_2_acc: 0.2356 - val_dense_3_acc: 0.2804 -
val_dense_4_acc: 0.8299 - val_dense_5_acc: 0.9886 - val_dense_6_acc: 0.9998

Epoch 00012: val_loss did not improve from 5.75069
Epoch 13/25
33401/33401 [==============================] - 300s 9ms/step - loss: 7.7992 -
dense_1_loss: 1.1011 - dense_2_loss: 2.4061 - dense_3_loss: 2.6669 -
dense_4_loss: 1.3306 - dense_5_loss: 0.2881 - dense_6_loss: 0.0064 -
dense_1_acc: 0.6675 - dense_2_acc: 0.2619 - dense_3_acc: 0.2235 - dense_4_acc:
0.6909 - dense_5_acc: 0.9555 - dense_6_acc: 0.9993 - val_loss: 5.5272 -
val_dense_1_loss: 0.5294 - val_dense_2_loss: 2.1043 - val_dense_3_loss: 2.1249 -
val_dense_4_loss: 0.6635 - val_dense_5_loss: 0.0920 - val_dense_6_loss: 0.0131 -
val_dense_1_acc: 0.8047 - val_dense_2_acc: 0.2677 - val_dense_3_acc: 0.2661 -
val_dense_4_acc: 0.8315 - val_dense_5_acc: 0.9886 - val_dense_6_acc: 0.9989

Epoch 00013: val_loss improved from 5.75069 to 5.52721, saving model to
models/svhn.cnn.weights.best.hdf5
Epoch 14/25
33401/33401 [==============================] - 300s 9ms/step - loss: 8.3126 -
```

```
dense_1_loss: 1.1739 - dense_2_loss: 2.5540 - dense_3_loss: 2.8229 -
dense_4_loss: 1.4067 - dense_5_loss: 0.3355 - dense_6_loss: 0.0196 -
dense_1_acc: 0.6641 - dense_2_acc: 0.2491 - dense_3_acc: 0.2169 - dense_4_acc:
0.6898 - dense_5_acc: 0.9499 - dense_6_acc: 0.9984 - val_loss: 6.0660 -
val_dense_1_loss: 0.7099 - val_dense_2_loss: 2.1479 - val_dense_3_loss: 2.3164 -
val_dense_4_loss: 0.8092 - val_dense_5_loss: 0.0800 - val_dense_6_loss: 0.0025 -
val_dense_1_acc: 0.7799 - val_dense_2_acc: 0.2821 - val_dense_3_acc: 0.2812 -
val_dense_4_acc: 0.8214 - val_dense_5_acc: 0.9887 - val_dense_6_acc: 0.9998

Epoch 00014: val_loss did not improve from 5.52721
Epoch 15/25
33401/33401 [==============================] - 300s 9ms/step - loss: 7.5857 -
dense_1_loss: 0.9664 - dense_2_loss: 2.4766 - dense_3_loss: 2.6251 -
dense_4_loss: 1.2467 - dense_5_loss: 0.2664 - dense_6_loss: 0.0046 -
dense_1_acc: 0.7038 - dense_2_acc: 0.2580 - dense_3_acc: 0.2236 - dense_4_acc:
0.6914 - dense_5_acc: 0.9562 - dense_6_acc: 0.9995 - val_loss: 5.7650 -
val_dense_1_loss: 0.5534 - val_dense_2_loss: 2.2184 - val_dense_3_loss: 2.2215 -
val_dense_4_loss: 0.6872 - val_dense_5_loss: 0.0821 - val_dense_6_loss: 0.0024 -
val_dense_1_acc: 0.8177 - val_dense_2_acc: 0.2360 - val_dense_3_acc: 0.2803 -
val_dense_4_acc: 0.8326 - val_dense_5_acc: 0.9887 - val_dense_6_acc: 0.9998

Epoch 00015: val_loss did not improve from 5.52721
Epoch 16/25
33401/33401 [==============================] - 299s 9ms/step - loss: 7.5178 -
dense_1_loss: 0.9746 - dense_2_loss: 2.3657 - dense_3_loss: 2.5889 -
dense_4_loss: 1.2901 - dense_5_loss: 0.2931 - dense_6_loss: 0.0054 -
dense_1_acc: 0.6835 - dense_2_acc: 0.2644 - dense_3_acc: 0.2268 - dense_4_acc:
0.6926 - dense_5_acc: 0.9549 - dense_6_acc: 0.9996 - val_loss: 5.6495 -
val_dense_1_loss: 0.5974 - val_dense_2_loss: 2.0846 - val_dense_3_loss: 2.1424 -
val_dense_4_loss: 0.7194 - val_dense_5_loss: 0.1038 - val_dense_6_loss: 0.0018 -
val_dense_1_acc: 0.8003 - val_dense_2_acc: 0.2565 - val_dense_3_acc: 0.2804 -
val_dense_4_acc: 0.8326 - val_dense_5_acc: 0.9887 - val_dense_6_acc: 0.9998

Epoch 00016: val_loss did not improve from 5.52721
Epoch 17/25
33401/33401 [==============================] - 299s 9ms/step - loss: 6.8725 -
dense_1_loss: 0.8248 - dense_2_loss: 2.2207 - dense_3_loss: 2.3957 -
dense_4_loss: 1.1621 - dense_5_loss: 0.2571 - dense_6_loss: 0.0121 -
dense_1_acc: 0.7181 - dense_2_acc: 0.2645 - dense_3_acc: 0.2383 - dense_4_acc:
0.6961 - dense_5_acc: 0.9560 - dense_6_acc: 0.9991 - val_loss: 5.3294 -
val_dense_1_loss: 0.5088 - val_dense_2_loss: 1.9996 - val_dense_3_loss: 2.0526 -
val_dense_4_loss: 0.6806 - val_dense_5_loss: 0.0860 - val_dense_6_loss: 0.0017 -
val_dense_1_acc: 0.8134 - val_dense_2_acc: 0.3084 - val_dense_3_acc: 0.2937 -
val_dense_4_acc: 0.8285 - val_dense_5_acc: 0.9887 - val_dense_6_acc: 0.9998

Epoch 00017: val_loss improved from 5.52721 to 5.32939, saving model to
models/svhn.cnn.weights.best.hdf5
Epoch 18/25
```

```
33401/33401 [==============================] - 300s 9ms/step - loss: 5.9644 -
dense_1_loss: 0.6070 - dense_2_loss: 1.9914 - dense_3_loss: 2.0994 -
dense_4_loss: 1.0266 - dense_5_loss: 0.2358 - dense_6_loss: 0.0041 -
dense_1_acc: 0.7590 - dense_2_acc: 0.2706 - dense_3_acc: 0.2524 - dense_4_acc:
0.6994 - dense_5_acc: 0.9568 - dense_6_acc: 0.9996 - val_loss: 5.2905 -
val_dense_1_loss: 0.4665 - val_dense_2_loss: 2.0324 - val_dense_3_loss: 2.0407 -
val_dense_4_loss: 0.6526 - val_dense_5_loss: 0.0965 - val_dense_6_loss: 0.0017 -
val_dense_1_acc: 0.8421 - val_dense_2_acc: 0.2877 - val_dense_3_acc: 0.3045 -
val_dense_4_acc: 0.8336 - val_dense_5_acc: 0.9887 - val_dense_6_acc: 0.9998

Epoch 00018: val_loss improved from 5.32939 to 5.29050, saving model to
models/svhn.cnn.weights.best.hdf5
Epoch 19/25
33401/33401 [==============================] - 301s 9ms/step - loss: 5.8362 -
dense_1_loss: 0.5555 - dense_2_loss: 1.9841 - dense_3_loss: 2.0715 -
dense_4_loss: 0.9914 - dense_5_loss: 0.2303 - dense_6_loss: 0.0034 -
dense_1_acc: 0.7807 - dense_2_acc: 0.2703 - dense_3_acc: 0.2594 - dense_4_acc:
0.7007 - dense_5_acc: 0.9566 - dense_6_acc: 0.9997 - val_loss: 5.1838 -
val_dense_1_loss: 0.4323 - val_dense_2_loss: 2.0267 - val_dense_3_loss: 2.0451 -
val_dense_4_loss: 0.6057 - val_dense_5_loss: 0.0721 - val_dense_6_loss: 0.0019 -
val_dense_1_acc: 0.8537 - val_dense_2_acc: 0.2837 - val_dense_3_acc: 0.2984 -
val_dense_4_acc: 0.8339 - val_dense_5_acc: 0.9887 - val_dense_6_acc: 0.9998

Epoch 00019: val_loss improved from 5.29050 to 5.18384, saving model to
models/svhn.cnn.weights.best.hdf5
Epoch 20/25
33401/33401 [==============================] - 300s 9ms/step - loss: 5.7492 -
dense_1_loss: 0.5183 - dense_2_loss: 1.9807 - dense_3_loss: 2.0561 -
dense_4_loss: 0.9647 - dense_5_loss: 0.2260 - dense_6_loss: 0.0035 -
dense_1_acc: 0.8006 - dense_2_acc: 0.2722 - dense_3_acc: 0.2640 - dense_4_acc:
0.7038 - dense_5_acc: 0.9567 - dense_6_acc: 0.9997 - val_loss: 5.0958 -
val_dense_1_loss: 0.4138 - val_dense_2_loss: 1.9990 - val_dense_3_loss: 2.0242 -
val_dense_4_loss: 0.5869 - val_dense_5_loss: 0.0702 - val_dense_6_loss: 0.0016 -
val_dense_1_acc: 0.8528 - val_dense_2_acc: 0.2821 - val_dense_3_acc: 0.2938 -
val_dense_4_acc: 0.8320 - val_dense_5_acc: 0.9886 - val_dense_6_acc: 0.9998

Epoch 00020: val_loss improved from 5.18384 to 5.09576, saving model to
models/svhn.cnn.weights.best.hdf5
Epoch 21/25
33401/33401 [==============================] - 300s 9ms/step - loss: 5.6786 -
dense_1_loss: 0.4902 - dense_2_loss: 1.9743 - dense_3_loss: 2.0372 -
dense_4_loss: 0.9448 - dense_5_loss: 0.2281 - dense_6_loss: 0.0039 -
dense_1_acc: 0.8117 - dense_2_acc: 0.2783 - dense_3_acc: 0.2739 - dense_4_acc:
0.7070 - dense_5_acc: 0.9564 - dense_6_acc: 0.9997 - val_loss: 5.0250 -
val_dense_1_loss: 0.3896 - val_dense_2_loss: 1.9856 - val_dense_3_loss: 1.9781 -
val_dense_4_loss: 0.5987 - val_dense_5_loss: 0.0715 - val_dense_6_loss: 0.0014 -
val_dense_1_acc: 0.8681 - val_dense_2_acc: 0.3003 - val_dense_3_acc: 0.3285 -
val_dense_4_acc: 0.8332 - val_dense_5_acc: 0.9887 - val_dense_6_acc: 0.9998
```

```
Epoch 00021: val_loss improved from 5.09576 to 5.02496, saving model to
models/svhn.cnn.weights.best.hdf5
Epoch 22/25
33401/33401 [==============================] - 300s 9ms/step - loss: 5.4920 -
dense_1_loss: 0.4311 - dense_2_loss: 1.9516 - dense_3_loss: 1.9804 -
dense_4_loss: 0.9063 - dense_5_loss: 0.2195 - dense_6_loss: 0.0031 -
dense_1_acc: 0.8393 - dense_2_acc: 0.2869 - dense_3_acc: 0.2946 - dense_4_acc:
0.7124 - dense_5_acc: 0.9567 - dense_6_acc: 0.9997 - val_loss: 5.0159 -
val_dense_1_loss: 0.4236 - val_dense_2_loss: 1.9530 - val_dense_3_loss: 1.8985 -
val_dense_4_loss: 0.6544 - val_dense_5_loss: 0.0848 - val_dense_6_loss: 0.0016 -
val_dense_1_acc: 0.8393 - val_dense_2_acc: 0.2959 - val_dense_3_acc: 0.3415 -
val_dense_4_acc: 0.8248 - val_dense_5_acc: 0.9880 - val_dense_6_acc: 0.9998

Epoch 00022: val_loss improved from 5.02496 to 5.01586, saving model to
models/svhn.cnn.weights.best.hdf5
Epoch 23/25
33401/33401 [==============================] - 300s 9ms/step - loss: 5.3284 -
dense_1_loss: 0.3936 - dense_2_loss: 1.9325 - dense_3_loss: 1.9058 -
dense_4_loss: 0.8792 - dense_5_loss: 0.2143 - dense_6_loss: 0.0030 -
dense_1_acc: 0.8545 - dense_2_acc: 0.3042 - dense_3_acc: 0.3127 - dense_4_acc:
0.7168 - dense_5_acc: 0.9565 - dense_6_acc: 0.9997 - val_loss: 4.4285 -
val_dense_1_loss: 0.2693 - val_dense_2_loss: 1.8365 - val_dense_3_loss: 1.7403 -
val_dense_4_loss: 0.5202 - val_dense_5_loss: 0.0603 - val_dense_6_loss: 0.0019 -
val_dense_1_acc: 0.9082 - val_dense_2_acc: 0.3590 - val_dense_3_acc: 0.3763 -
val_dense_4_acc: 0.8363 - val_dense_5_acc: 0.9887 - val_dense_6_acc: 0.9998

Epoch 00023: val_loss improved from 5.01586 to 4.42847, saving model to
models/svhn.cnn.weights.best.hdf5
Epoch 24/25
33401/33401 [==============================] - 301s 9ms/step - loss: 4.9983 -
dense_1_loss: 0.3741 - dense_2_loss: 1.7137 - dense_3_loss: 1.8308 -
dense_4_loss: 0.8674 - dense_5_loss: 0.2082 - dense_6_loss: 0.0041 -
dense_1_acc: 0.8620 - dense_2_acc: 0.3933 - dense_3_acc: 0.3331 - dense_4_acc:
0.7158 - dense_5_acc: 0.9563 - dense_6_acc: 0.9996 - val_loss: 4.0349 -
val_dense_1_loss: 0.2573 - val_dense_2_loss: 1.5541 - val_dense_3_loss: 1.6254 -
val_dense_4_loss: 0.5037 - val_dense_5_loss: 0.0916 - val_dense_6_loss: 0.0029 -
val_dense_1_acc: 0.9212 - val_dense_2_acc: 0.4495 - val_dense_3_acc: 0.4117 -
val_dense_4_acc: 0.8397 - val_dense_5_acc: 0.9868 - val_dense_6_acc: 0.9998

Epoch 00024: val_loss improved from 4.42847 to 4.03493, saving model to
models/svhn.cnn.weights.best.hdf5
Epoch 25/25
33401/33401 [==============================] - 300s 9ms/step - loss: 4.5203 -
dense_1_loss: 0.3145 - dense_2_loss: 1.5287 - dense_3_loss: 1.6466 -
dense_4_loss: 0.8343 - dense_5_loss: 0.1934 - dense_6_loss: 0.0028 -
dense_1_acc: 0.8868 - dense_2_acc: 0.4535 - dense_3_acc: 0.3923 - dense_4_acc:
0.7207 - dense_5_acc: 0.9566 - dense_6_acc: 0.9997 - val_loss: 3.5610 -
```

```
val_dense_1_loss: 0.2380 - val_dense_2_loss: 1.3407 - val_dense_3_loss: 1.4230 -
val_dense_4_loss: 0.5082 - val_dense_5_loss: 0.0494 - val_dense_6_loss: 0.0018 -
val_dense_1_acc: 0.9206 - val_dense_2_acc: 0.5243 - val_dense_3_acc: 0.4994 -
val_dense_4_acc: 0.8391 - val_dense_5_acc: 0.9888 - val_dense_6_acc: 0.9998

Epoch 00025: val_loss improved from 4.03493 to 3.56102, saving model to
models/svhn.cnn.weights.best.hdf5
```

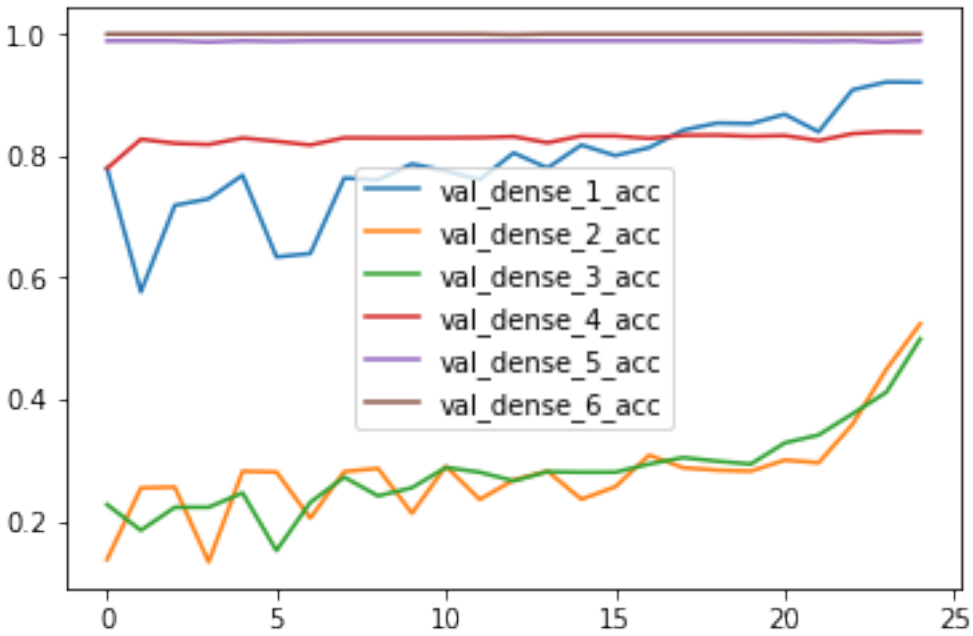
[11]:
```
n_digits, digit1, digit2, digit3, digit4, digit5 = svhn_model.predict(X_test,␣
↪verbose=1)
```

```
13068/13068 [==============================] - 33s 3ms/step
```

[12]:
```
(y_test[0] == np.argmax(n_digits, axis=1)).sum()/len(n_digits)
```

[12]:
```
0.9206458524640343
```

[22]:
```
pd.DataFrame(result.history)[['val_dense_{}_acc'.format(i) for i in range(1,␣
↪7)]].plot();
```



[23]:
```
confusion_matrix(y_true=y_test[0], y_pred=np.argmax(n_digits, axis=1))
```

[23]:
```
array([[2363,  104,   12,    4,    0],
       [ 183, 7605,  568,    0,    0],
       [  20,   55, 2000,    6,    0],
       [   0,    0,   83,   63,    0],
```

```
       [   0,    0,    1,    1,    0]])
```

[24]:
```
confusion_matrix(y_true=y_test[1], y_pred=np.argmax(digit1, axis=1))
```

[24]:
```
array([[   0,   14,    1,    1,    0,    0,    3,    0,    0,    0],
       [   0, 3542,  100,    3,   23,    1,   13,   12,    0,    0],
       [   0,  134, 2137,  222,   15,   20,   30,   97,    2,    0],
       [   0,  112, 1029,  250,   33,  121,   39,   36,    5,    0],
       [   0,  851,   91,    3,  239,    1,   20,   31,    0,    0],
       [   0,   45,  481,  276,    2,  145,   65,   13,   25,    0],
       [   0,  226,   89,   43,   70,   26,  354,   34,   11,    4],
       [   0,  202,  335,   17,   21,    3,   18,  150,    1,    0],
       [   0,   98,  126,   78,   21,   55,  189,   20,   34,    4],
       [   0,   70,  216,   73,   12,   75,   64,   24,   22,    0]])
```

[23]:
```
pd.Series(np.argmax(digit1, axis=1)).value_counts()
```

[23]:
```
1    12704
7      302
2       55
3        7
dtype: int64
```

[18]:
```
y_test[0].value_counts(normalize=True)
```

[18]:
```
2    0.639425
1    0.190006
3    0.159244
4    0.011172
5    0.000153
Name: 0, dtype: float64
```

[ ]:
```

```