Batch normalizing a familiar model

Remember the **digits dataset** you trained in the first exercise of this chapter?

A multi-class classification problem that you solved using `softmax` and 10 neurons in your output layer.

You will now build a new deeper model consisting of 3 hidden layers of 50 neurons each, using batch normalization in between layers. The `kernel_initializer` parameter is used to initialize weights in a similar way.

- Import `BatchNormalization` from keras layers.
- Build your deep network model, use **50 neurons for each hidden layer** adding batch normalization in between layers.
- Compile your model with stochastic gradient descent, `sgd`, as an optimizer.

```
# Import batch normalization from keras layers

from keras.layers import BatchNormalization


# Build your deep network

batchnorm_model = Sequential()

batchnorm_model.add(Dense(50, input_shape=(64,), activation='relu', kernel_initializer='normal'))

batchnorm_model.add(BatchNormalization())

batchnorm_model.add(Dense(50, activation='relu', kernel_initializer='normal'))

batchnorm_model.add(BatchNormalization())

batchnorm_model.add(Dense(50, activation='relu', kernel_initializer='normal'))

batchnorm_model.add(BatchNormalization())

batchnorm_model.add(Dense(10, activation='softmax', kernel_initializer='normal'))


# Compile your model with sgd

batchnorm_model.compile(optimizer='sgd', loss='categorical_crossentropy', metrics=['accuracy'])
```