Chapter 1: Neural Networks Foundations
### Model 1

<table>
<thead>
<tr>
<th>Layer Type</th>
<th>Output Shape</th>
<th>Param # Connected to</th>
</tr>
</thead>
<tbody>
<tr>
<td>dense_1</td>
<td>(None, 16)</td>
<td>dense_input_1</td>
</tr>
<tr>
<td>dense_1</td>
<td>(None, 16)</td>
<td>0</td>
</tr>
<tr>
<td>activation_1 (Activation)</td>
<td>(None, 16)</td>
<td>dense_1</td>
</tr>
</tbody>
</table>

Total params: 16

Train on 10000 samples, validate on 10000 samples

Epoch 1

Train - loss: 8.2768 - acc: 0.8707 - val_loss: 8.2762 - val_acc: 0.8701

### Model 2

<table>
<thead>
<tr>
<th>Layer Type</th>
<th>Output Shape</th>
<th>Param # Connected to</th>
</tr>
</thead>
<tbody>
<tr>
<td>dense_1</td>
<td>(None, 16)</td>
<td>dense_input_1</td>
</tr>
<tr>
<td>dense_1</td>
<td>(None, 16)</td>
<td>0</td>
</tr>
<tr>
<td>activation_1 (Activation)</td>
<td>(None, 16)</td>
<td>dense_1</td>
</tr>
</tbody>
</table>

Total params: 16

Train on 10000 samples, validate on 10000 samples

Epoch 1

Train - loss: 8.2768 - acc: 0.8707 - val_loss: 8.2762 - val_acc: 0.8701
<table>
<thead>
<tr>
<th>Layer Type</th>
<th>Output Shape</th>
<th>Param #</th>
<th>Connected to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dense</td>
<td>None</td>
<td>108,096</td>
<td>dense_input</td>
</tr>
<tr>
<td>Activation</td>
<td>None</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dense</td>
<td>None</td>
<td>1500</td>
<td>activation</td>
</tr>
<tr>
<td>Activation</td>
<td>None</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dense</td>
<td>None</td>
<td>1500</td>
<td>activation</td>
</tr>
<tr>
<td>Activation</td>
<td>None</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dense</td>
<td>None</td>
<td>1500</td>
<td>activation</td>
</tr>
<tr>
<td>Activation</td>
<td>None</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dense</td>
<td>None</td>
<td>1296</td>
<td>activation</td>
</tr>
<tr>
<td>Activation</td>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

![Training and testing accuracy](image1.png)

![Model accuracy](image2.png)
Chapter 2: Keras Installation and API
Chapter 3: Deep Learning with ConvNets
model loss

- train
- test

epoch

Loss:

```
Layer (Type)       Output Shape  Param #  Connected to
Convolution2D_1    (None, 32, 32, 32)  896  convolution2D_input_18[0][0]
Activation_1       (None, 32, 32, 32)  0  convolution2D_1[0][0]
Convolution2D_2    (None, 32, 32, 32)  3264  activation_1[0][0]
Activation_2       (None, 32, 32, 32)  0  convolution2D_2[0][0]
MaxPooling2D_1     (None, 32, 16, 16)  0  activation_2[0][0]
Dropout_1          (None, 32, 16, 16)  0  maxpooling2D_1[0][0]
Convolution2D_3    (None, 32, 16, 16)  10240  dropout_1[0][0]
Activation_3       (None, 32, 16, 16)  0  convolution2D_3[0][0]
Convolution2D_4    (None, 32, 16, 16)  10240  activation_3[0][0]
Activation_4       (None, 32, 16, 16)  0  convolution2D_4[0][0]
MaxPooling2D_2     (None, 32, 8, 8)  0  activation_4[0][0]
Dropout_2          (None, 32, 8, 8)  0  maxpooling2D_2[0][0]
Flatten_1          (None, 2048)  0  dropout_2[0][0]
Dense_1            (None, 512)  104928  flatten_1[0][0]
Activation_5       (None, 512)  0  dense_1[0][0]
Dropout_3          (None, 512)  0  activation_5[0][0]
Dense_2            (None, 32)  16656  dropout_3[0][0]
Activation_6       (None, 32)  0  dense_2[0][0]
```

Total params: 47,586

Train on 40000 samples, validate on 10000 samples

Epoch 1/5

```
Train - [49295/49295] - loss: 0.5279 - acc: 0.8446 - val_loss: 0.5028 - val_acc: 0.8685
```

Epoch 2/5

```
Train - [49295/49295] - loss: 0.5267 - acc: 0.8446 - val_loss: 0.5009 - val_acc: 0.8685
```

Epoch 3/5

```
Train - [49295/49295] - loss: 0.5261 - acc: 0.8446 - val_loss: 0.4998 - val_acc: 0.8685
```

Epoch 4/5

```
Train - [49295/49295] - loss: 0.5258 - acc: 0.8446 - val_loss: 0.4993 - val_acc: 0.8685
```

Epoch 5/5

```
Train - [49295/49295] - loss: 0.5257 - acc: 0.8446 - val_loss: 0.4990 - val_acc: 0.8685
```

```
Test accuracy: 0.8685000000000001
```
Chapter 4: Generative Adversarial Networks and WaveNet
This flower is pink, white, and yellow in color, and has petals that are striped.
Figure 1: DCGAN generator used for LSUN some modeling. A 100 dimensional uniform distribution $Z$ is projected to a small spatial extent convolutional representation with many feature maps. A series of four fractionally-strided convolutions (in some recent papers, these are wrongly called deconvolutions) then convert this high level representation into a $64 \times 64$ pixel image. Notably, no fully connected or pooling layers are used.
Chapter 5: Word Embeddings
Chapter 6: Recurrent Neural Network — RNN

![Graph showing training and validation accuracy and loss over epochs.](image)
Train on 98112 samples, validate on 42648 samples
Epoch 1/1: [37s - loss: 0.0056 - mean_squared_error: 0.0056 - val_loss: 0.0038 - val_mean_squared_error: 0.0038]
Epoch 4/5: [37s - loss: 0.0056 - mean_squared_error: 0.0056 - val_loss: 0.0038 - val_mean_squared_error: 0.0038]
Epoch 8/5: [37s - loss: 0.0056 - mean_squared_error: 0.0056 - val_loss: 0.0038 - val_mean_squared_error: 0.0038]
Epoch 1/1: [37s - loss: 0.0043 - mean_squared_error: 0.0043 - val_loss: 0.0038 - val_mean_squared_error: 0.0038]
Epoch 4/5: [37s - loss: 0.0043 - mean_squared_error: 0.0043 - val_loss: 0.0038 - val_mean_squared_error: 0.0038]
Epoch 8/5: [37s - loss: 0.0043 - mean_squared_error: 0.0043 - val_loss: 0.0038 - val_mean_squared_error: 0.0038]

Loss: 0.0033, RMSE: 0.0036
Chapter 7: Additional Deep Learning Models
Chapter 8: AI Game Playing
Appendix: Conclusion

Unique users of the Keras documentation.