Basic NN for predicting Heart problems in babies

Abstract:

The idea of this project is to get the state of a baby heart and use a Data set that has been classified by expert obstetricians. With this we can see 3 different classifications:

Fetal state class code (N=normal; S=suspect; P=pathologic),I give them values of 1,2 and 3.

The data:

The data used for this project was obtained in UCI, it uses fetal heart rate (FHR) and uterine contraction (UC)**.**

Link: https://archive.ics.uci.edu/ml/datasets/Cardiotocography

The data has 2126 instances, i use half of them for training, and the other half for testing.

Model Building:

```
network = models.Sequential()

network.add(layers.Dense(128,activation='tanh', input_shape=(36,)))

network.add(layers.Dense(64,activation='tanh'))

network.add(layers.Dense(32,activation='tanh'))

network.add(layers.Dense(16,activation='tanh'))

network.add(layers.Dense(4,activation='softmax'))
```

Model Architecture:

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 128)               4736
_________________________________________________________________
dense_2 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_3 (Dense)              (None, 32)                2080
_________________________________________________________________
dense_4 (Dense)              (None, 16)                528
_________________________________________________________________
dense_5 (Dense)              (None, 4)                 68
=================================================================
Total params: 15,668
Trainable params: 15,668
Non-trainable params: 0
_________________________________________________________________
```

I use all danse layers, a tanh activation and a softmax for classifying and getting the probability of each class.

Model training

```
Epoch 16/25
1063/1063 [==============================] - 0s 79us/step - loss: 0.1541 - accuracy: 0.9595 - val_loss: 0.4054 - val_accuracy: 0.8109
Epoch 17/25
1063/1063 [==============================] - 0s 76us/step - loss: 0.1366 - accuracy: 0.9746 - val_loss: 0.3846 - val_accuracy: 0.8608
Epoch 18/25
1063/1063 [==============================] - 0s 82us/step - loss: 0.1224 - accuracy: 0.9831 - val_loss: 0.3683 - val_accuracy: 0.9153
Epoch 19/25
1063/1063 [==============================] - 0s 74us/step - loss: 0.1108 - accuracy: 0.9897 - val_loss: 0.3535 - val_accuracy: 0.9294
Epoch 20/25
1063/1063 [==============================] - 0s 90us/step - loss: 0.1006 - accuracy: 0.9934 - val_loss: 0.3310 - val_accuracy: 0.9464
Epoch 21/25
1063/1063 [==============================] - 0s 75us/step - loss: 0.0925 - accuracy: 0.9953 - val_loss: 0.3203 - val_accuracy: 0.9445
Epoch 22/25
1063/1063 [==============================] - 0s 85us/step - loss: 0.0854 - accuracy: 0.9953 - val_loss: 0.3045 - val_accuracy: 0.9445
Epoch 23/25
1063/1063 [==============================] - 0s 78us/step - loss: 0.0795 - accuracy: 0.9962 - val_loss: 0.2958 - val_accuracy: 0.9445
Epoch 24/25
1063/1063 [==============================] - 0s 74us/step - loss: 0.0742 - accuracy: 0.9962 - val_loss: 0.2864 - val_accuracy: 0.9370
Epoch 25/25
1063/1063 [==============================] - 0s 73us/step - loss: 0.0701 - accuracy: 0.9962 - val_loss: 0.2769 - val_accuracy: 0.9276
1063/1063 [==============================] - 0s 23us/step
test_acc: 0.927563488483429  test_loss: 0.27686026058775687
```

This is the model while training and we can also see the test accuracy and the test loss at the end.