<u>Hierarchical Building/Floor Classification and Location Estimation using Wi-Fi Fingerprinting Based on Deep Neural Networks</u>

Introduction

There are scenarios where there is no line-of-sight signal for global positioning systems (GPSs) e.g. indoors.

But WiFi signals are usually available indoors.

A position of a user/device then can be estimated by finding the closest match between its Received Signal Strength (RSS) measurement and the fingerprints of known locations in a database.

A location fingerprint is a vector of a pair of a service set identifier (SSID) and an RSS for a Wi-Fi access point (AP) measured at a location.

Currently best solutions rely on filtering, manual data analysis, and time consuming parameter tuning to achieve reliable and accurate localization.

Task given

To understand the ideas for implementation of a prototype for WIFI localisation
To improve the accuracy of a given prototype.

Ideas of prototype implementation

1) Autoencoder

This is an artificial neural network used for unsupervised learning of efficient coding.

The aim of an autoencoder is to learn representation (encoding) for a set of data, typically for the purpose of dimensionality reduction.

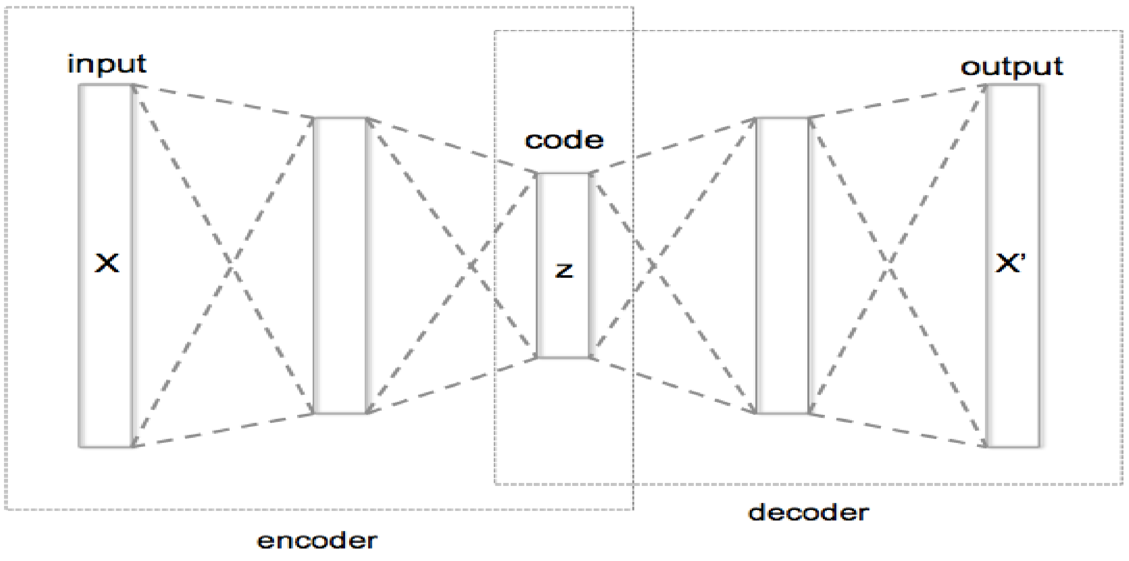

Figure 1:Autoencoder

When the unsupervised learning of weights of SAE is finished, the decoder part of the network is disconnected and typical fully-connected layers of a deep network (classifier) are connected to the output of the encoder.

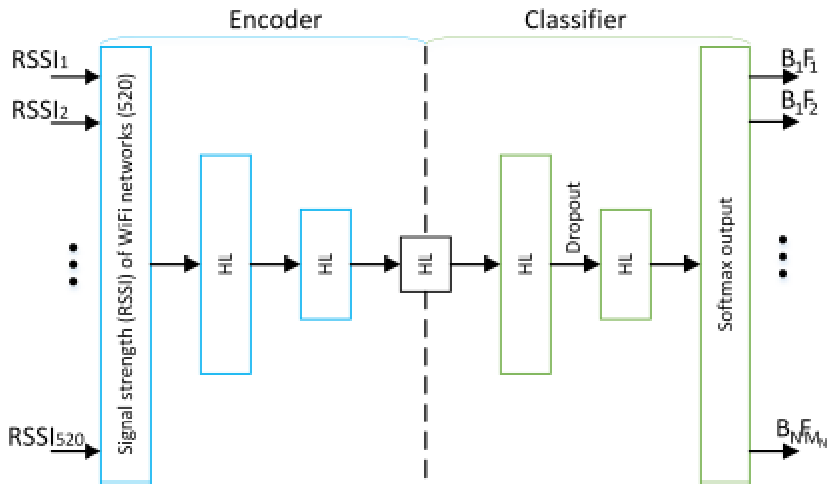

Figure 2 : shows the classifier

2) Dropout

Dropout is a technique where randomly selected neurons are ignored during training.

The effect is that the network becomes less sensitive to the specific weights of neurons. This in turn results in a network that is capable of better generalization and is less likely to overfit the training data.

Before the supervised learning is performed, the labelled data is divided into the training, validation, and testing sets.

During the training process, weights of SAE learned during unsupervised training and weights in classifier are modified to create a final system.

In the simplest case, where there is one hidden layer, the encoder stage of an autoencoder takes the input ***X*** and maps it to ***Z***.

$$\mathbf{z} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$$

z =code, *latent variables*, or *latent representation*.
σ= activation function (sigmoid /rectified linear unit)
w= weight matrix
b=bias vector

After that, the decoder stage of the autoencoder maps *z to x'* of the same shape as *x.*

$$\mathbf{x}' = \sigma'(\mathbf{W}'\mathbf{z} + \mathbf{b}')$$

Measure the deviation of ***X'*** from the input (typically using squared error). Back-propagate the error through the network and perform weight update.

$$e_j(n) = d_j(n) - y_j(n)$$

***d*=** the target value
***y*=** value produced by the neuron

The node weights are adjusted based on corrections that minimize the error in the entire output, given by

$$\mathcal{E}(n) = \frac{1}{2}\sum_j e_j^2(n).$$

Using gradient descent, the change in each weight is

$$\Delta w_{ji}(n) = -\eta \frac{\partial \mathcal{E}(n)}{\partial v_j(n)} y_i(n)$$

y=the output of the previous neuron
η= the *learning rate*

3) Epoch

An *epoch* describes the number of times the algorithm sees the *entire* data set One epoch means one pass of the full training set. Usually it may contain a few iterations.

Because usually we divide the training set into batches, each epoch go through the whole training set. Each iteration goes through one batch.

RESULTS

I will first present the results gotten with using the default parameters. The results for different cases where different parameters in the system were changed will be presented.

Results with default parameters (60% accurancy)

```
#+STARTUP: showall
* System parameters
  - Numpy random number seed: 0
  - Ratio of training data to overall data: 0.70
  - Ratio of validation data to overall data: 0.20
  - Ratio of test data to overall data: 0.10
  - Number of epochs: 20
  - Batch size: 10
  - SAE hidden layers: 256-128-64-128-256
  - SAE activation: tanh
  - SAE bias: False
  - SAE optimizer: adam
  - SAE loss: mse
  - Classifier hidden layers: 128-256-512
  - Classifier hidden layer activation: relu
  - Classifier bias: False
  - Classifier optimizer: adagrad
  - Classifier loss: categorical_crossentropy
  - Classifier dropout rate: 0.20
* Performance
  - Loss = 1.267468e+00
  - Accuracy (overall): 6.034483e-01
```

First case (accuracy=43%)	Second case (accuracy=56%)
#+STARTUP: showall	#+STARTUP: showall
* System parameters	* System parameters
- Numpy random number seed: 0	- Numpy random number seed: 0
- Ratio of training data to overall data: 0.70	- Ratio of training data to overall data: 0.70
- Ratio of validation data to overall data: 0.20	- Ratio of validation data to overall data: 0.20
- Ratio of test data to overall data: 0.10	- Ratio of test data to overall data: 0.10
- Number of epochs: 20	- Number of epochs: 50
- Batch size: 20	- Batch size: 50
- SAE hidden layers: 256-64-8-64-256	- SAE hidden layers: 64-8-64
- SAE activation: tanh	- SAE activation: tanh
- SAE bias: False	- SAE bias: False
- SAE optimizer: adam	- SAE optimizer: adam
- SAE loss: mse	- SAE loss: mse
	- Classifier hidden layers: 64-32-7

- Classifier hidden layers: 128-256-512 - Classifier hidden layer activation: relu - Classifier bias: False - Classifier optimizer: adagrad - Classifier loss: categorical_crossentropy - Classifier dropout rate: 0.10 * Performance - Loss = 1.810004e+00 - Accuracy (overall): 4.366126e-01	- Classifier hidden layer activation: relu - Classifier bias: False - Classifier optimizer: adagrad - Classifier loss: categorical_crossentropy - Classifier dropout rate: 0.20 * Performance - Loss = 4.773298e+00 - Accuracy (overall): 5.578093e-02

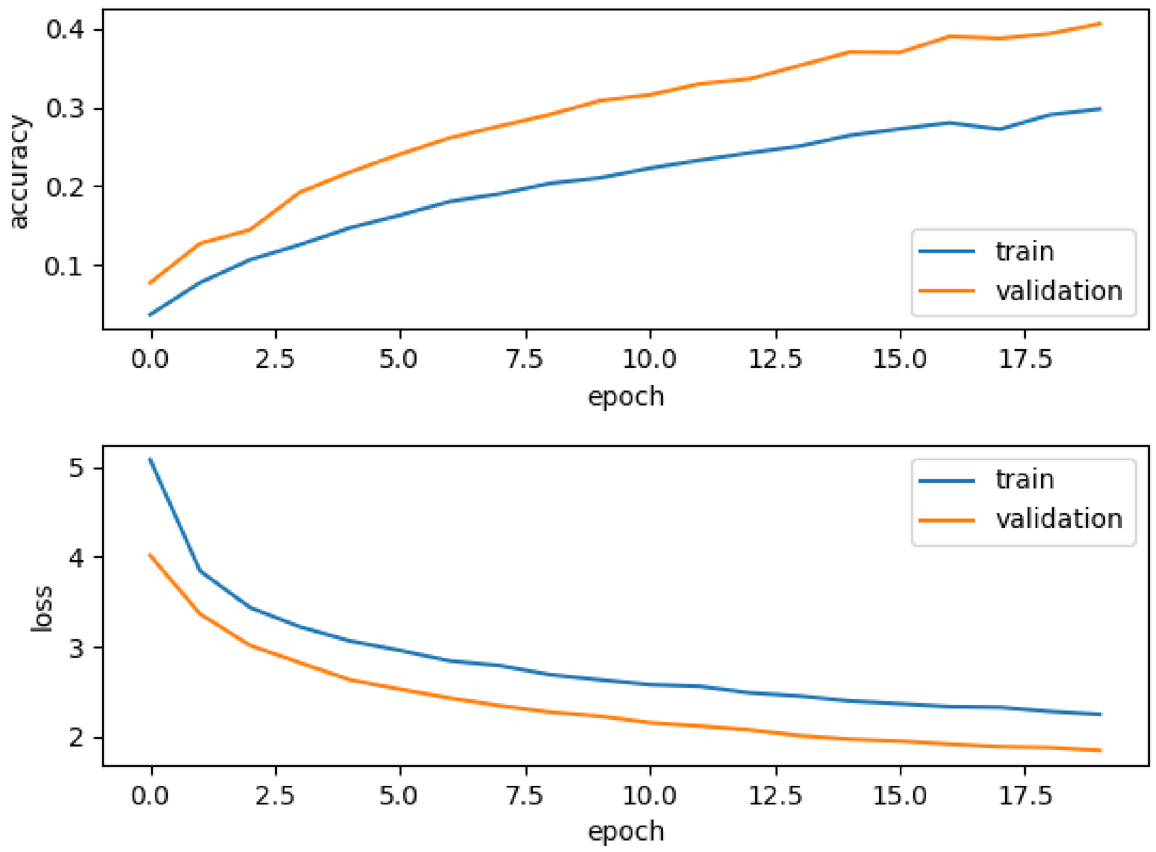

Figure 3: Graph for first case

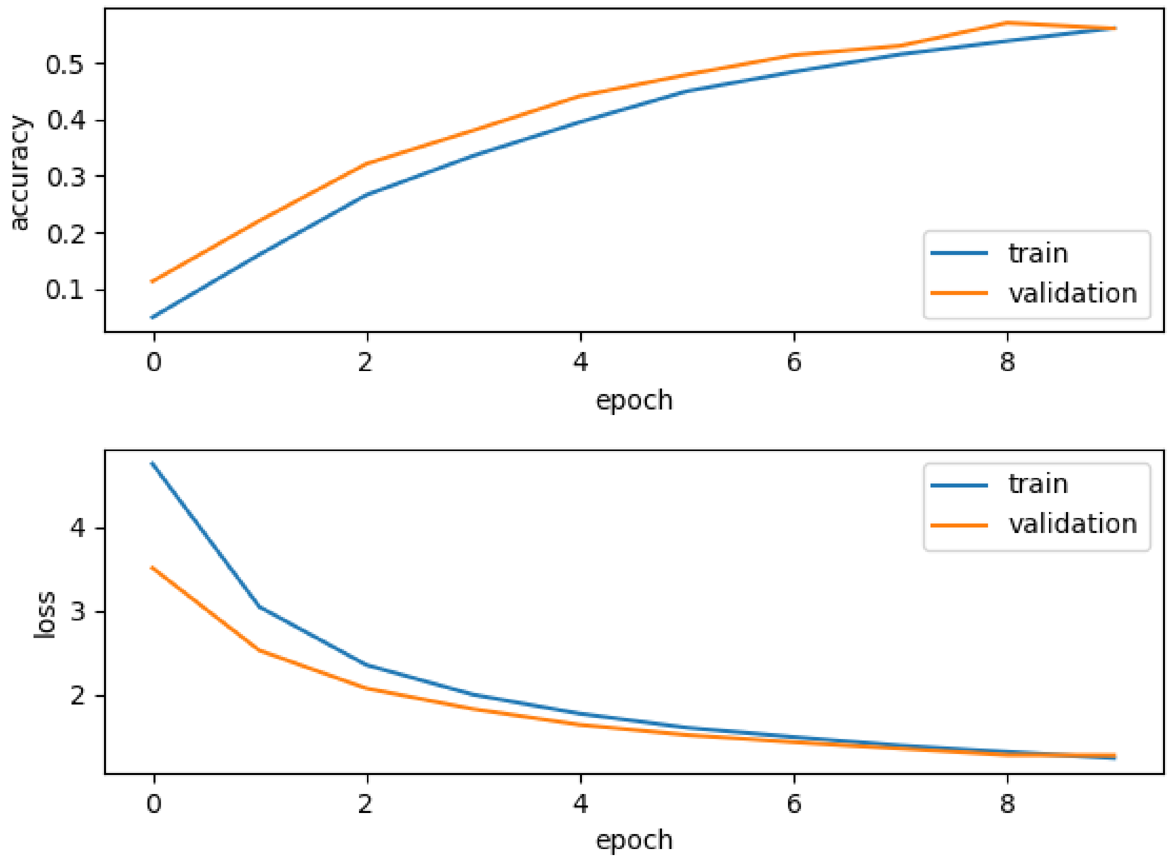

Figure 4: Graph for the second case

Third case (accuracy = 64%)	Fourth case (accuracy=70%)
#+STARTUP: showall * System parameters - Numpy random number seed: 0 - Ratio of training data to overall data: 0.70 - Ratio of validation data to overall data: 0.20 - Ratio of test data to overall data: 0.10 - Number of epochs: 20 - Batch size: 10 - SAE hidden layers: 512-256-128-256-512 - SAE activation: relu - SAE bias: False - SAE optimizer: adam - SAE loss: mse	#+STARTUP: showall * System parameters - Numpy random number seed: 0 - Ratio of training data to overall data: 0.70 - Ratio of validation data to overall data: 0.20 - Ratio of test data to overall data: 0.10 - Number of epochs: 100 - Batch size: 100 - SAE hidden layers: 128-64-128 - SAE activation: tanh - SAE bias: False - SAE optimizer: adam - SAE loss: mse

- Classifier hidden layers: 128-256-512 - Classifier hidden layer activation: relu - Classifier bias: False - Classifier optimizer: adagrad - Classifier loss: categorical_crossentropy - Classifier dropout rate: 0.08 * Performance - Loss = 1.134682e+00 - Accuracy (overall): 6.399594e-01	- Classifier hidden layers: 64-128-256 - Classifier hidden layer activation: elu - Classifier bias: False - Classifier optimizer: adagrad - Classifier loss: categorical_crossentropy - Classifier dropout rate: 0.20 * Performance - Loss = 1.059734e+00 - Accuracy (overall): 6.977688e-01

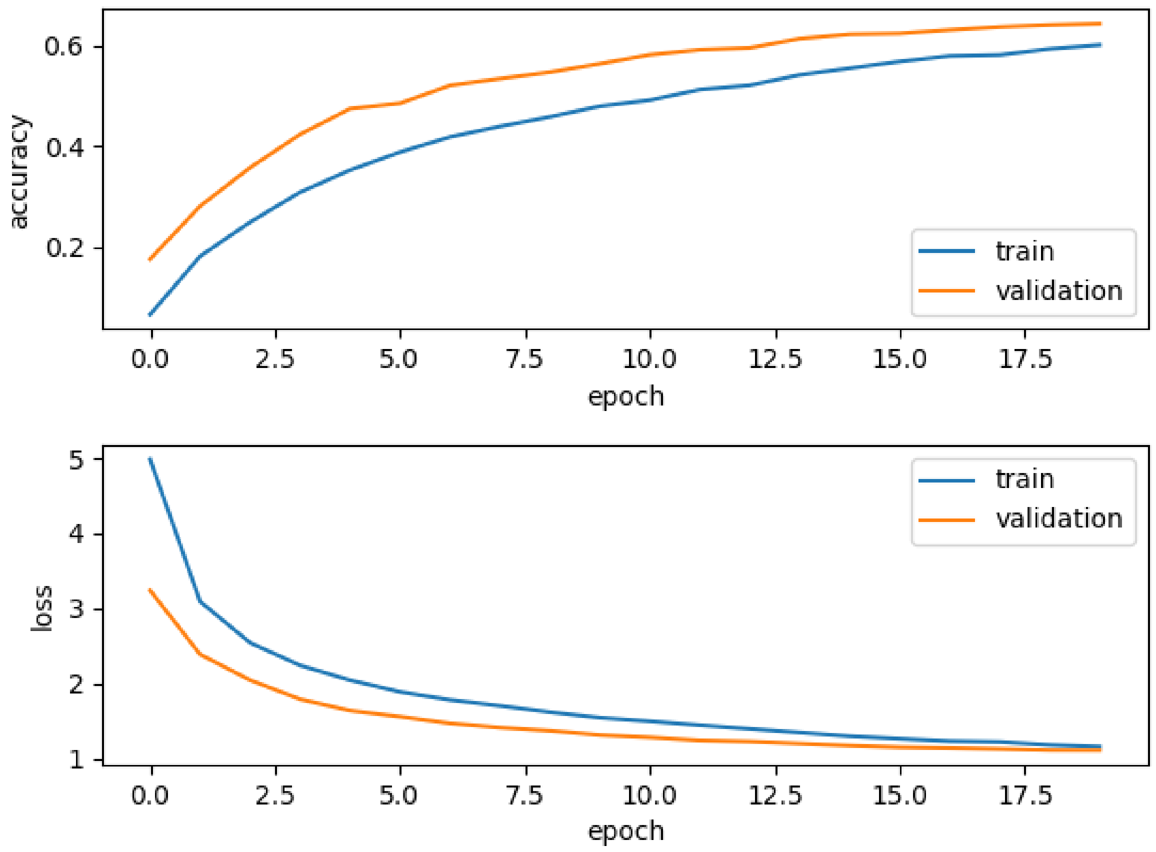

Figure 5: Graph for third case

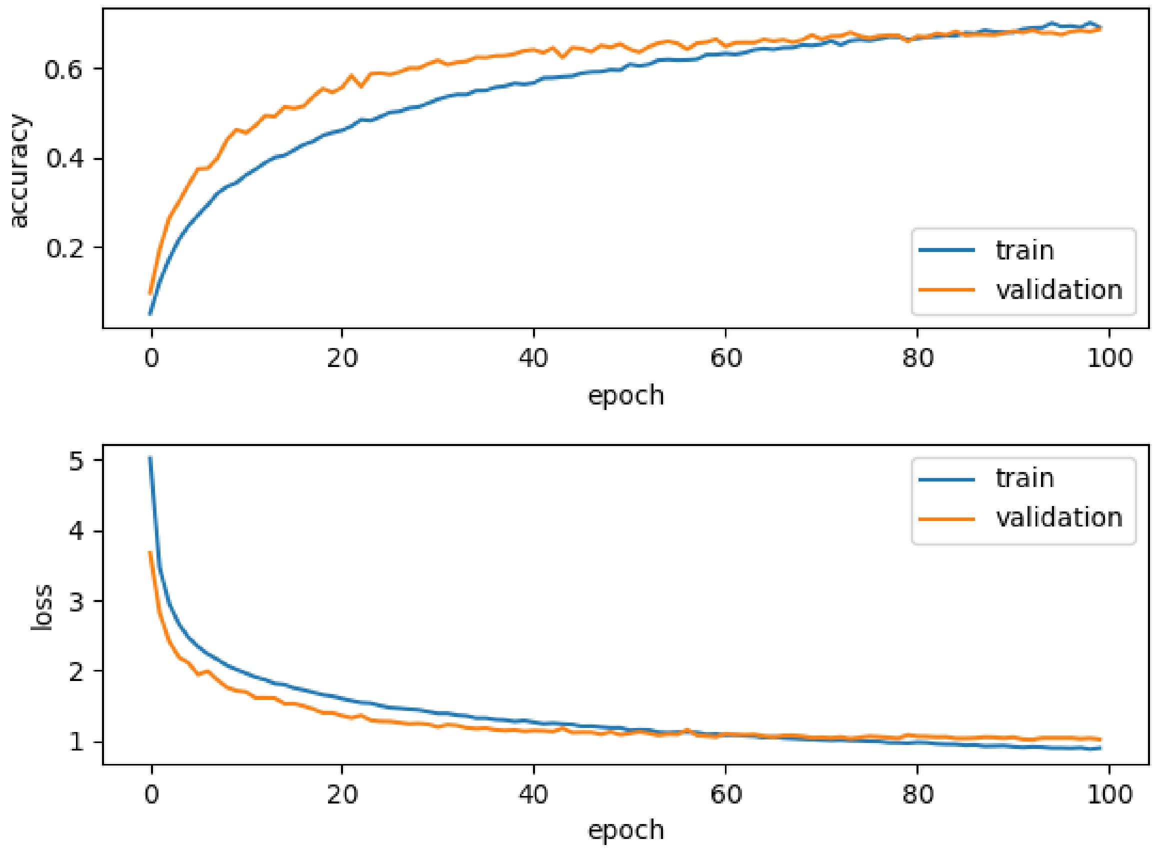

Figure 6: Graph for fourth case

Fifth case (accuracy=72%)	Best case (accuracy=73%)
#+STARTUP: showall * System parameters - Numpy random number seed: 0 - Ratio of training data to overall data: 0.70 - Ratio of validation data to overall data: 0.20 - Ratio of test data to overall data: 0.10 - Number of epochs: 100 - Batch size: 100 - SAE hidden layers: 128-64-128 - SAE activation: tanh - SAE bias: False - SAE optimizer: adam - SAE loss: mse	#+STARTUP: showall * System parameters - Numpy random number seed: 0 - Ratio of training data to overall data: 0.70 - Ratio of validation data to overall data: 0.20 - Ratio of test data to overall data: 0.10 - Number of epochs: 200 - Batch size: 200 - SAE hidden layers: 256-128-256 - SAE activation: relu - SAE bias: False - SAE optimizer: adam - SAE loss: mse

- Classifier hidden layers: 64-128-256 - Classifier hidden layer activation: elu - Classifier bias: False - Classifier optimizer: adagrad - Classifier loss: categorical_crossentropy - Classifier dropout rate: 0.10 * Performance - Loss = 1.106789e+00 - Accuracy (overall): 7.185598e-01	- Classifier hidden layers: 128-256-512 - Classifier hidden layer activation: relu - Classifier bias: False - Classifier optimizer: adagrad - Classifier loss: categorical_crossentropy - Classifier dropout rate: 0.20 * Performance - Loss = 9.538585e-01 - Accuracy (overall): 7.342799e-01

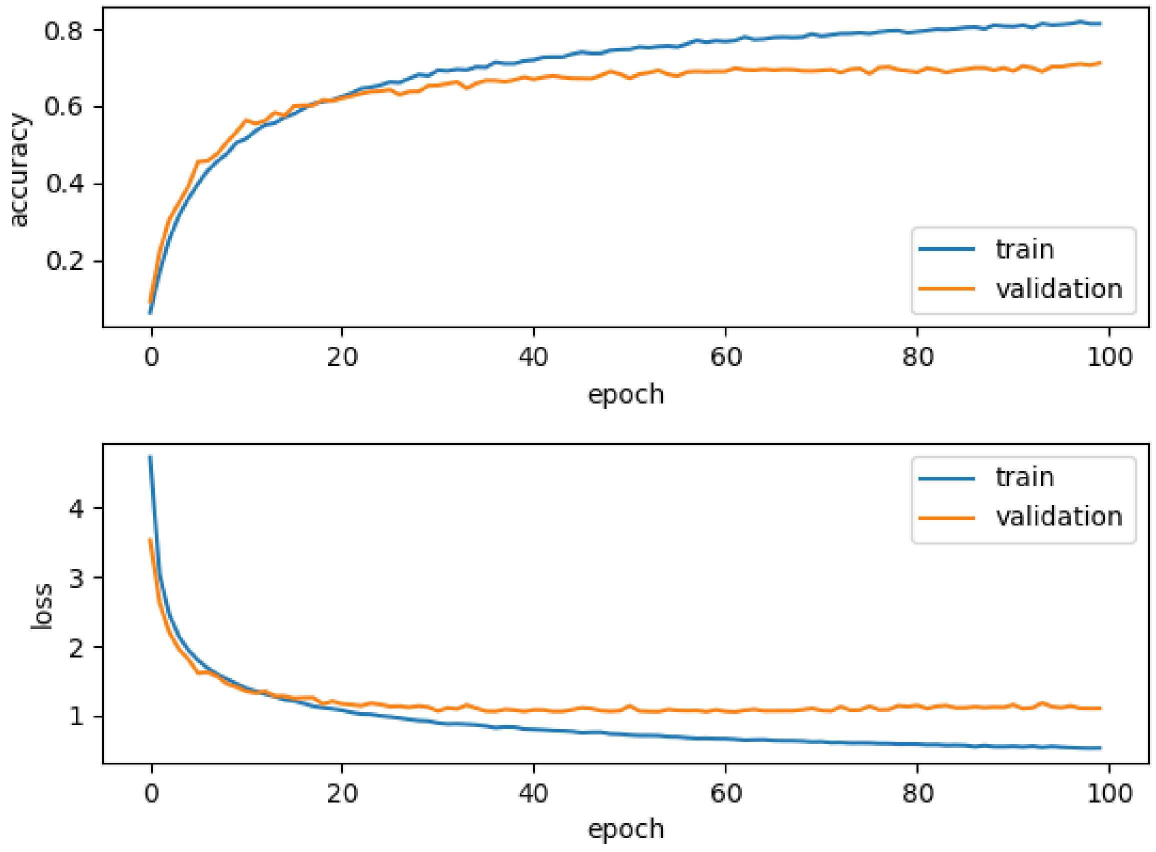

Figure 7: Graph for fifth case

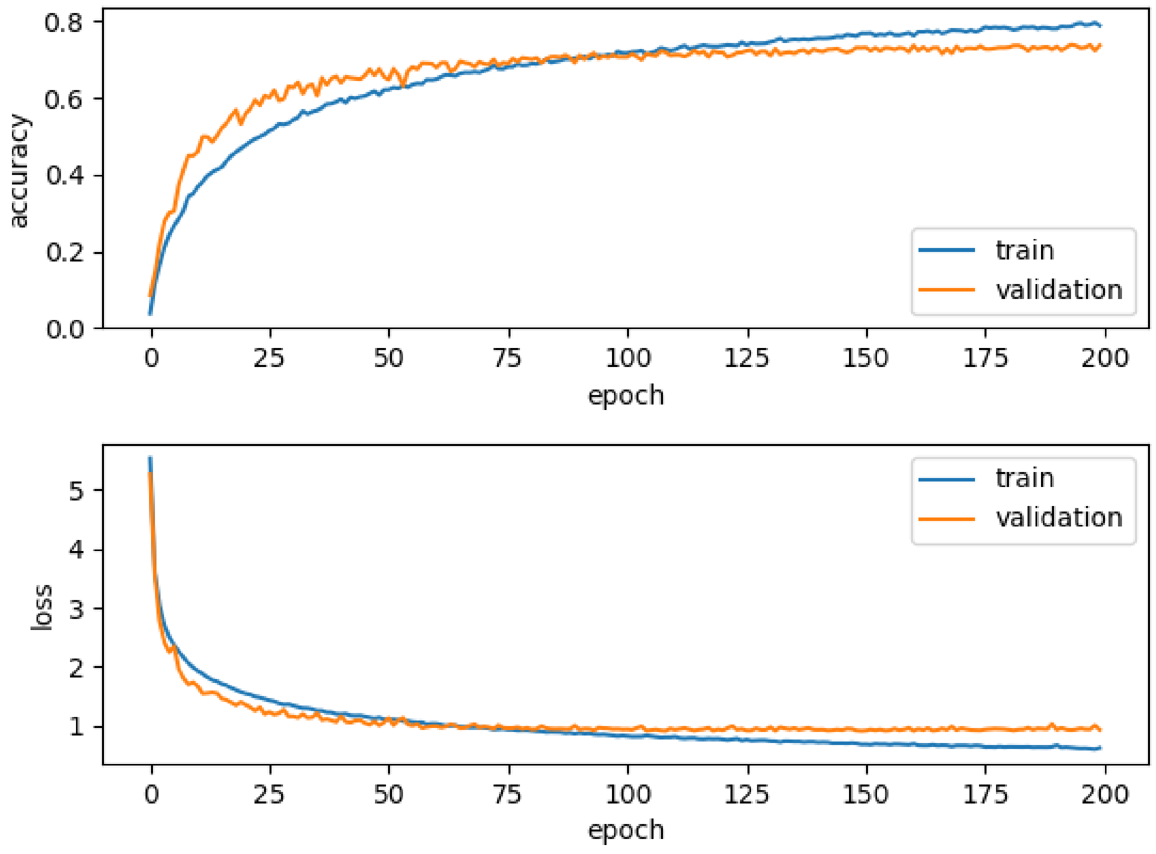

Figure 8: Graph for best case

Conclusion

Changing different parameters changes the accuracy of the whole prototype implementation.

Therefore there is need to carry out further study and change the parameters so that the accuracy is almost 100%.

References

1) K. S. Kim, R. Wang, Z. Zhong, Z. Tan, H. Song, J. Cha, and S. Lee, “Large-scale location-aware services in access: Hierarchical building/floor classification and location estimation using wi-fi fingerprinting based on deep neural networks,” in Proc. FOAN 2017, Munich, Germany, Nov. 2017, pp. 1–5.
2) M. Nowicki and J. Wietrzykowski, “Low-effort place recognition with wifi fingerprints using deep learning,” ArXiv e-prints, Apr. 2017.