Imperial College London

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

Machine Learning 395 CBC 2

Authors:
Arun Kothari (CID: 00642763)
Greg Collinge (CID: 00693146)
Kiyohito Kunii (CID: 00003317)
Peter Hedley (CID: 01397550)

Date: March 8, 2018

1 Layers

1.1 Linear Layer

The forward pass of a linear layer calculates the output activation y from the activation of the previous layer X according to the linear transformation

$$y = WX + b$$

where W is the set of weights connecting the linear layer and the preceding layer, and b is the bias of each neuron in the linear layer. For the backward pass the gradient with respect to W, X, and b are calculated as

$$dW = X \times d_{\text{out}} \qquad dX = W \times d_{\text{out}} \qquad db = d_{\text{out}}$$

where d_{out} is the derivative of the output of the linear function with respect to the error function placed on the network output.

1.2 ReLU Layer

The forward pass of a ReLU layer applies a non-linear activation function to the output activation of the preceding linear layer. It has an output activation y of zero for all negative inputs and leaves positive inputs unchanged.

$$y = \begin{cases} X & \text{if } X > 0; \\ 0 & \text{otherwise.} \end{cases}$$

For back-propagation the gradient is dependent on which region the function used on the forward pass. If for example the input was negative, the node did not contribute to the error and therefore no gradient can be back-propagated. The gradient d_y is therefore calculated in the backward pass as

$$d_y = \begin{cases} d_{\text{out}} & \text{if } X > 0; \\ 0 & \text{otherwise.} \end{cases}$$

where d_{out} is the derivative of the output of that ReLU function with respect to the error function placed on the network output. A node is referred to as *'dead'* if the gradient is not updated through the course of an epoch because it has a ReLU output activation of zero for all examples.

1.3 Dropout

A dropout layer adds a form of regularization to the network, and functions differently at training and test time. In the training time forward pass, neurons are deactivated with a random binomial probability p. This is implemented by multiplying the output activations by a binary mask with the same shape the the preceding layer. At test time no neurons are deactivated; the output activation is just scaled by a factor of $1 - p$. The scaling is applied to keep the overall expected output of the layer at testing time the same as at training time.
In the backward pass, the gradient is calculated by multiplying the upstream derivative d_{out} by the binary mask that was used by the same dropout layer in the forward pass. In the implementation of the the fully-connected network an *inverted* dropout layer was used, in which

the activation scaling is applied at training time instead of test time in order to minimise the computation required at test time.

1.4 Softmax

The softmax function (1) is used to normalise the each of the K outputs z_j from the final layer such that the softmax activation σ_j from each neuron is in the range 0-1 and $\Sigma\sigma_j = 1$. The extra term $log(C) = -max(z_i)$ is added to the exponent in order to improve the numerical stability of the calculation.

$$\sigma_j = \sigma(z_j) = \frac{e^{z_j + log(C)}}{\sum_k e^{z_k + log(C)}} \qquad \text{for} \qquad j,k \in \{1,\ldots,K\} \tag{1}$$

The σ_j values can be directly interpreted as the probability of each class. A higher probability value for a class indicates that the network is confident that the input features X correspond to that class. In back-propagation of the gradients, since each input has an influence on every output (due to the normalisation) the gradient of each input affects every output value. The gradient F_{jk} is calculated according to:

$$F_{jk} = \frac{\partial \sigma_j}{\partial z_k} = \begin{cases} \sigma_j(1-\sigma_j) & \text{if } j = k; \\ -\sigma(e^{z_j})\sigma(e^{z_k}) & \text{if } j \neq k. \end{cases} \tag{2}$$

2 CIFAR-10

A two layer fully connected network using ReLU activation, an optional dropout layer and a Softmax loss function was tested using the CIFAR-10 image dataset. The model was first overfitted on a small subset of 50 training images to check the behavior was as expected. A training accuracy of 100% was reached within 8 epochs and is shown in Figure 2. The overfitting test shows that the model was able to learn the test data, but gives no indication of the ability to generalise to new data. To test generalisation to new data, the model was then trained on the full dataset of 49,000 images, reaching a peak validation accuracy of 52.0% and test set accuracy of 50.3% on 1000 images. The loss and accuracy for training on the full dataset is shown in Figure 3. The parameters and architecture used for both tests are shown in Table 1. For overfitting the network, no regularisation was applied as this has the effect of counteracting overfitting. It was observed that overfitting to 100% training set accuracy within 20 epochs would occur with a wide range of parameters, whereas the test set accuracy was sensitive to changes in the learning rate and learning rate decay.

3 Hyper-parameter optimisation with FER2013

:

3.0.1 Initial architecture and parameters

A three layer network with two hidden layers of 100 neurons each was used as the starting point for optimisation. Architectures with more than three hidden layers were not considered due to the vanishing gradient problem, hence a two hidden layer network represented the middle of

the range to be searched when optimising the network. The number of neurons was selected to be comparable with the 128 that was successful on the CIFAR-10 dataset. An early stopping criterion was introduced to reduce overfitting and the time taken to search for parameters. The strategy used was to stop training at the point when the accuracy on the validation set for the most recent epoch was lower than it was ten epochs earlier. A value of ten epochs was found empirically to provide a reasonable balance between robustness to erroneous stopping due to accuracy fluctuations and the time taken to terminate after the plateau in validation accuracy had been reached. The learning rate update schedule used was to multiply the learning rate by a constant learning rate decay factor after each epoch. The initial learning rate and learning rate decay were set to $\eta = 1 \times 10^{-3}$ and $r = 0.94$ respectively, as this was successful on the CIFAR-10 dataset. The gradient update rule was SGD with momentum, with a momentum parameter of $\gamma = 0.95$.

3.0.2 Optimising the learning rate

A validation set of 1000 images from the training set was used so that the validation accuracy could be computed for each epoch. The validation accuracy was chosen as the quantity to maximise throughout the optimisation process as it provides a direct measure of the expected performance on unseen data from the same distribution. The learning rate was optimised on the initial architecture using a 2D grid search on learning rate and learning rate decay, as shown in Figure 4. The optimal values were found to be $\eta = 1.4 \times 10^{-2}$ and $r = 0.8$ respectively.

3.0.3 Effect of dropout regularisation

The effect of varying the dropout parameter on the initial architecture with the optimised learning rate can be seen in Figure 5. It was found that no value of dropout led to an improvement in the validation accuracy, and that values of $p > 0.1$ caused a significant decrease in performance. Generally dropout would be expected to improve the generalisation to new data and hence and cause an improvement in validation accuracy, especially as the network was observed overfitting the training data (in Figure 8 the training accuracy approaches 100%). This may be due to the small nature of the network, as most neurons are needed to form a prediction. Dropout also increases the time taken for the network to converge, so it is possible that a benefit would have been obtained with a different learning rate schedule.

3.0.4 Effect of L2 regularisation

The effect of varying the L2 regularisation parameter on the initial architecture with the optimised learning rate can be seen in Figure 6. As with dropout, it was found that none of the L2 values sampled led to an improvement in validation accuracy. It would be expected that an L2 regularisation term would improve generalisation to new data by adding a penalty for non-zero weights, and hence reducing overfitting to the training data. Since both L2 and dropout did not improve the validation accuracy, the optimised model has no regularisation mechanism and is therefore prone to overfitting.

3.0.5 Optimising the topology of the network

Finally, The topology of the network was optimised by considering architectures with between one and three hidden layers. Hidden layers were optimised in series using a 1D recursive grid search method as displayed in Figure 1, where successively smaller regions were searched with

higher granularity. If there was more than one layer, subsequent layers were optimised using the optimised number of neurons in the preceding layers rather than those in the initial architecture. The optimal architecture was found to be a two hidden layer network in a 544×801 configuration. The optimisation process results are shown in Figure 7.

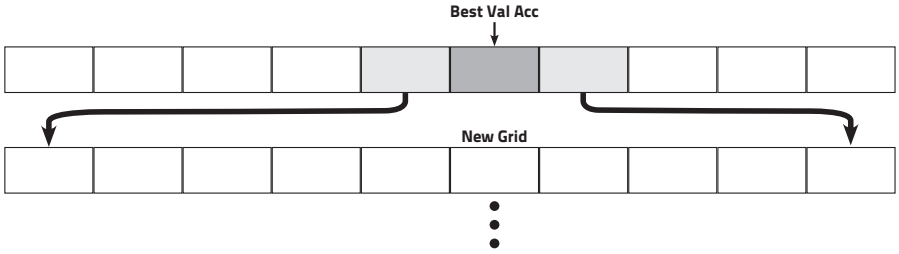


Figure 1: Demonstration of Recursive Grid search approach to hyper-parameter tuning

3.0.6 Results

The parameters of the initial and optimised models are shown in Table 2. The training of the optimised model is shown in Figure 8, and the state at the 10^{th} epoch was selected for testing as this had the peak validation set accuracy of 46.5% and should suffer less from overfitting than later epochs. The performance of the optimised model was evaluated on the entire test set and achieved a classification rate of 47.2%. The normalised confusion matrix is shown in Figure 9, and the recall, precision and F_1 scores by class are shown in Table 3.

4 Comparison of CNN with feed-forward network

A CNN based on the VGG architecture was implemented in Keras and trained on the FER2013 dataset. An accuracy of 60.8% was achieved the test set. The normalised confusion matrix is shown in Figure 9, and the recall, precision and F_1 scores by class are shown in Table 3. The overall classification rate was significantly higher than 47.2% obtained with the optimised feed-forward network. From the confusion matrix it is notable that the CNN makes more specific confusions between the emotions than the feed-forward network. For example it has strong confusion between Anger and Disgust, and Sadness and Fear, which may be evidence for higher level feature extraction than the feed-forward network which confuses classes more uniformly. The F_1 scores of the CNN are generally higher than for the feed-forward; the most significant exception is for Disgust, which is significantly underrepresented in the data set. The out-performance of the feed-forward network by the CNN is expected, as the CNN architecture is specifically designed for image recognition and makes explicit use of the spatial relationship between neurons. Between convolutional layers each neuron is only connected to a small region in the preceding layer, drastically reducing the number of parameters per layer and allowing scalability to larger images and deeper networks. This allows the extraction of high level features in the CNN, whereas the fully connected network makes no assumptions about the connections between neighboring neurons. For this model the training data was also preprocessed dynamically during training, effectively expanding the dataset and improving generalisation to new data. The best performing model in the 2013 Kaggle competition was 71.1%, and an accuracy of 60.8% would have corresponded to 12th position on the leaderboard. The relatively low accuracy on FER2013 compared to other

image recognition tasks reflects the fact that emotion recognition is characterised by abstract high level features, making it even more difficult for the simple feed-forward network to make accurate classifications.

5 Additional Questions

5.0.1 Comparing Learning Algorithms

Even if one machine learning algorithm has a statistically better performance than another on a particular dataset, that is not sufficient to say that it is a learning algorithm in general for several reasons. First, the auditability of the predictions made by the model is of particular concern in many practical applications. The way a decision tree makes a prediction can be clearly and intuitively understood by following the route through the tree, whereas the neural network is treated as a black-box method that makes predictions without a clear explanation.
Furthermore, features of the specific dataset such as the size, dimensionality, distribution of classes, and amount of noise are important in determining which algorithm will perform better. For example decision trees can be successfully trained on relatively small datasets, whereas neural networks generally require large volumes of data. The amount of hyper-parameter tuning that is required can vary significantly between machine learning algorithms. Decision trees can be trained effectively with no tuning required, whereas for each dataset that a neural network is applied to a complex and computationally expensive search for the optimal hyper-parameters is required to get the best results. In some applications the time taken to train and make predictions is an important factor, such as in real time image classification. Finally the overall classification rate does not give information about the performance on specific classes. It is possible to have a high overall classification rate while also having unacceptably poor recall or precision on specific classes. A model that has a lower classification rate overall that achieves more consistent results for each class may be preferable for many applications. This is particularly likely to be an issue when certain classes are underrepresented in the data, as it is possible for a model to achieve a high accuracy while never making any predictions for such classes.

5.0.2 Adding new Classes to Decision Trees and Neural Networks

For the decision trees, the learning algorithm would be effectively unchanged by the addition of new classes. Additional binary trees would be created and trained from the data corresponding to the new classes, and results from each tree would be combined using the same combination scheme as before. Training decision trees is not computationally expensive so training a new model from scratch would be more practical than developing a method of re-training an existing model. For the neural network the number of neurons in the output layer would have to be increased to match the total number of classes, but could otherwise be be left unchanged. The network could either be re-trained from scratch (randomly initialised weights), or training could be continued on the new dataset, adjusting the previously learned weights. The justification for continued training is that certain features may have already been learned that are common between the emotions. In general some adjustment to the hyper-parameters would be required when any change is made to the structure of the dataset or the architecture of the network.

Appendix A Figures and Tables

Parameter	Overfit	Train
Hidden layer neurons	128	128
Learning rate	1e-3	1e-3
Learning rate decay	1	0.85
Batch size	25	65
Weight scale	1e-2	1e-2
Regularisation	0	1e-4
Dropout	0	0
Data type	np.float64	np.float64
Update rule	sgd	sgd

Table 1: Parameters for training a two-layer fully connected network.

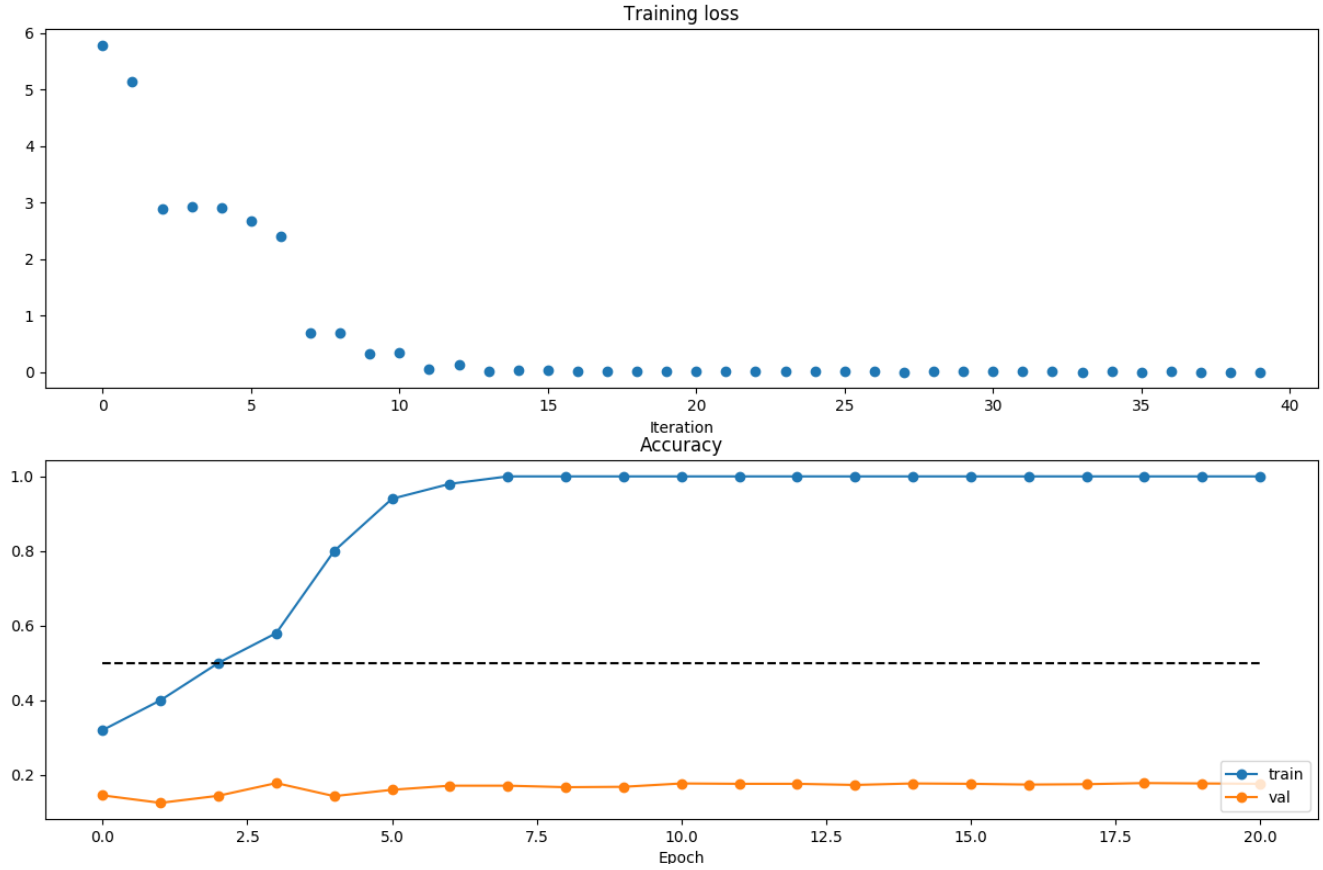


Figure 2: (Overfitting on 50 CIFAR-10 images) the two layer fully connected network with 128 neurons in the hidden layer.

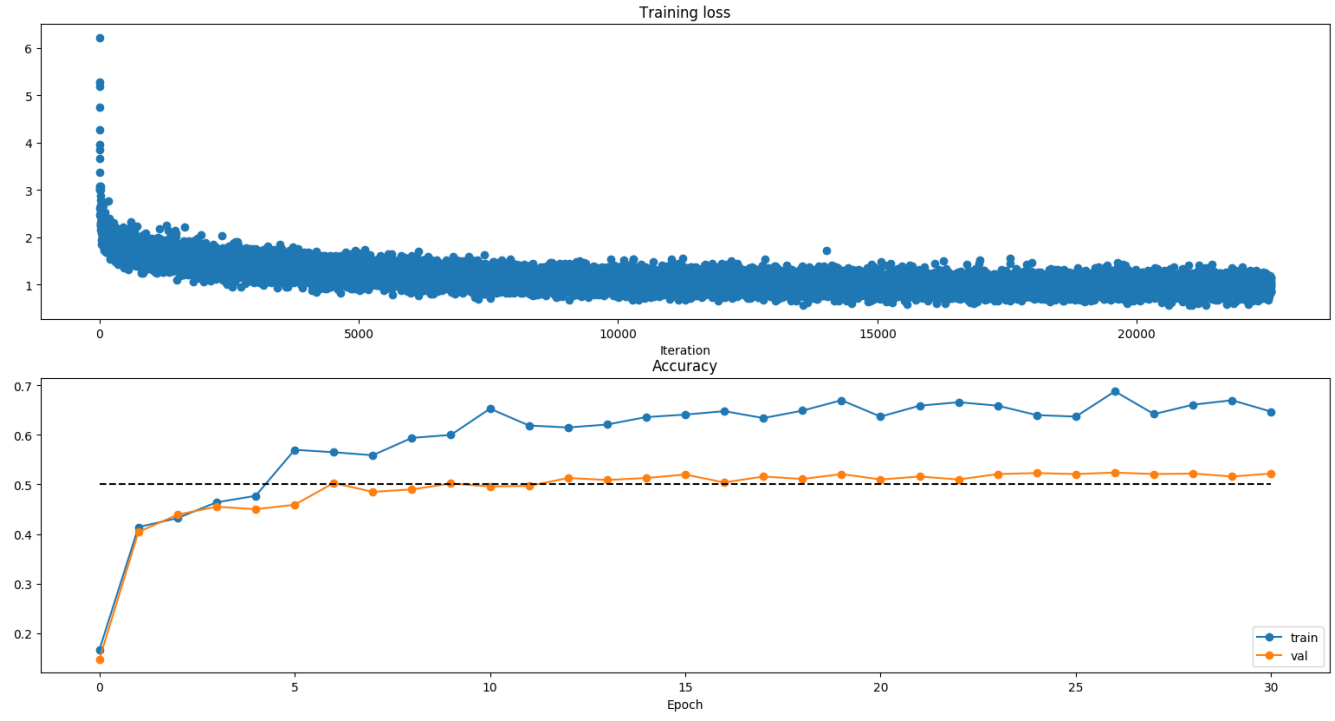


Figure 3: (Training on the full CIFAR-10 dataset) the two layer fully connected network with 128 neurons in the hidden layer.

Parameter	Initial	Optimised
Hidden layer 1 neurons	100	544
Hidden layer 2 neurons	100	801
Learning rate	1e-3	1.4e-2
Learning rate decay	0.95	0.80
Batch size	70	70
Weight scale	1e-2	1e-2
Regularisation	0	0
Dropout	0	0
Data type	np.float64	np.float64
Update rule	sgd-momentum	sgd-momentum
Momentum	0.95	0.95
Validation Accuracy	0.38.0	46.5

Table 2: Parameters of the initial and optimised fully connected network.

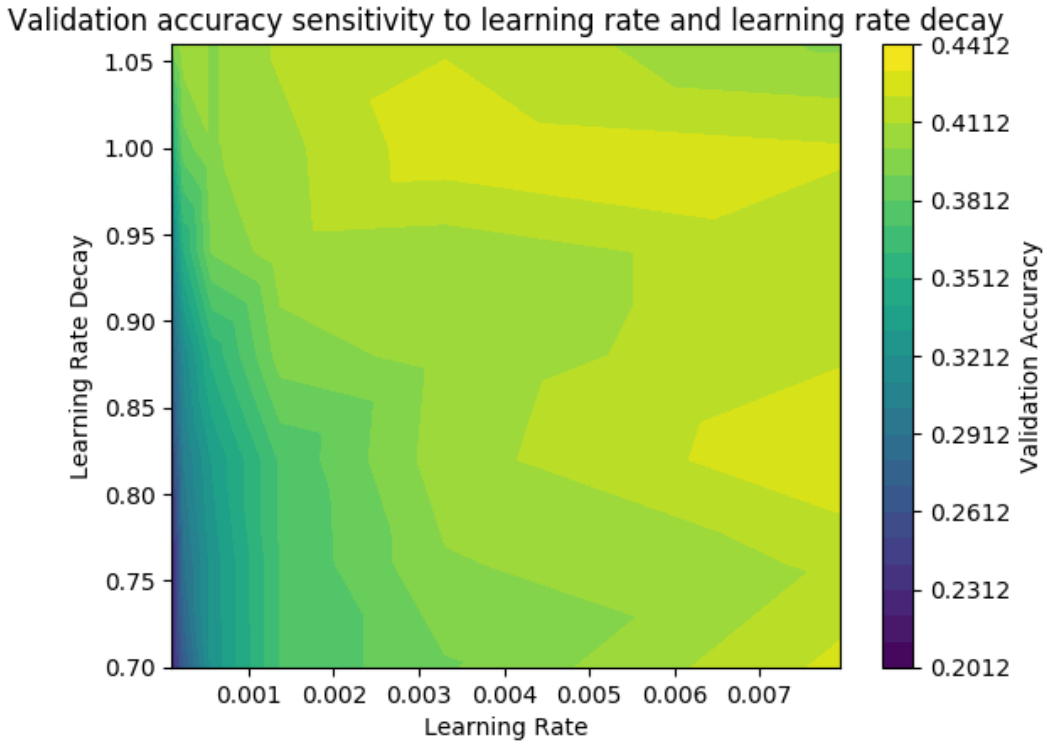
Emotion	Precision	Recall	F_1
1	0.36	0.30	0.32
2	0.75	0.38	0.50
3	0.33	0.34	0.33
4	0.59	0.66	0.62
5	0.42	0.37	0.39
6	0.70	0.63	0.66
7	0.38	0.43	0.40

(a) Feed-forward network.

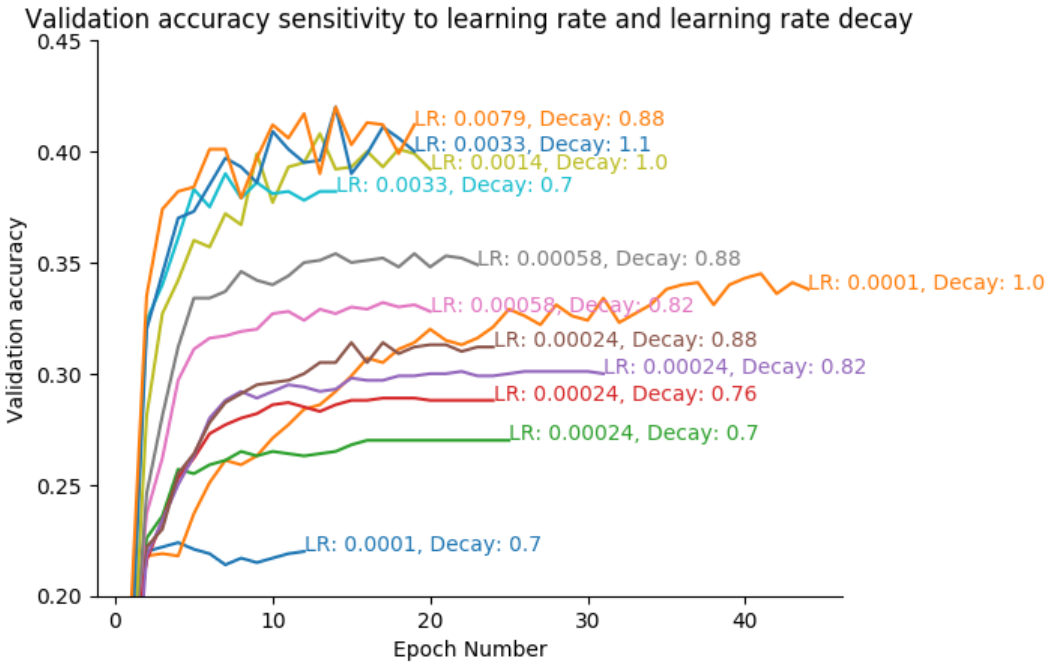
Emotion	Precision	Recall	F_1
1	0.53	0.54	0.53
2	0.67	0.14	0.24
3	0.43	0.21	0.28
4	0.80	0.85	0.83
5	0.50	0.52	0.51
6	0.69	0.81	0.75
7	0.51	0.62	0.56

(b) Convolutional network.

Table 3: Performance metrics by class on the FER2013 public test data.



(a) Peak validation accuracy as a function of learning rate and learning rate decay.



(b) Validation accuracy training curves for different grid positions.

Figure 4: Validation accuracy sensitivity for the initial architecture. The maximum classification rate on the validation set was found to be at learning rate $\eta = 1.4 \times 10^{-2}$ and decay factor $r = 0.8$.

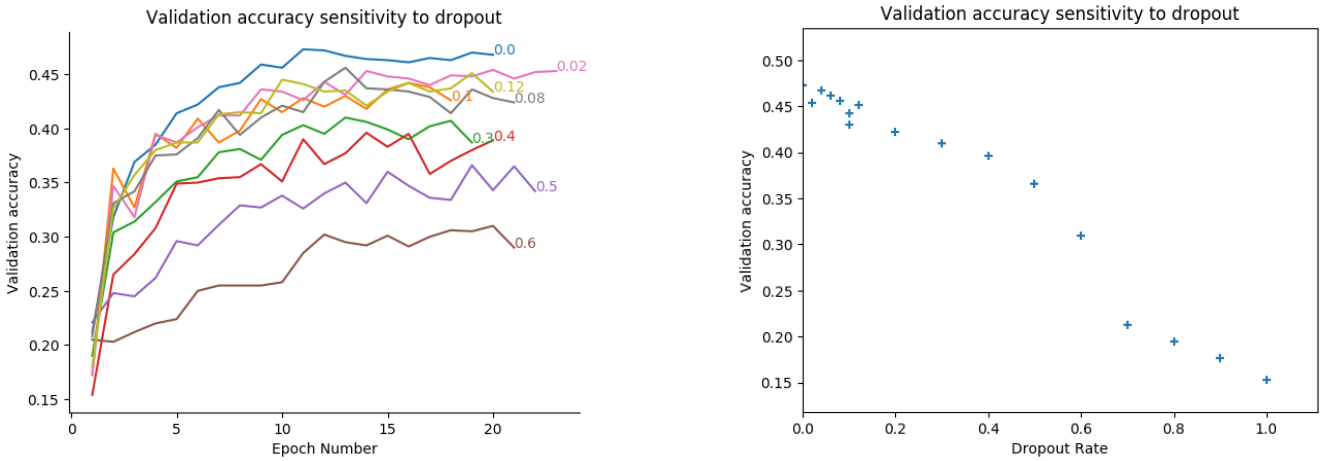


Figure 5: Effect of dropout regularisation on the initial architecture with optimised learning rate. The validation accuracy curves over training *(left)* and peak validation accuracies *(right)* are shown for a range of p values. No value of $p > 0$ was found to improve the validation accuracy.

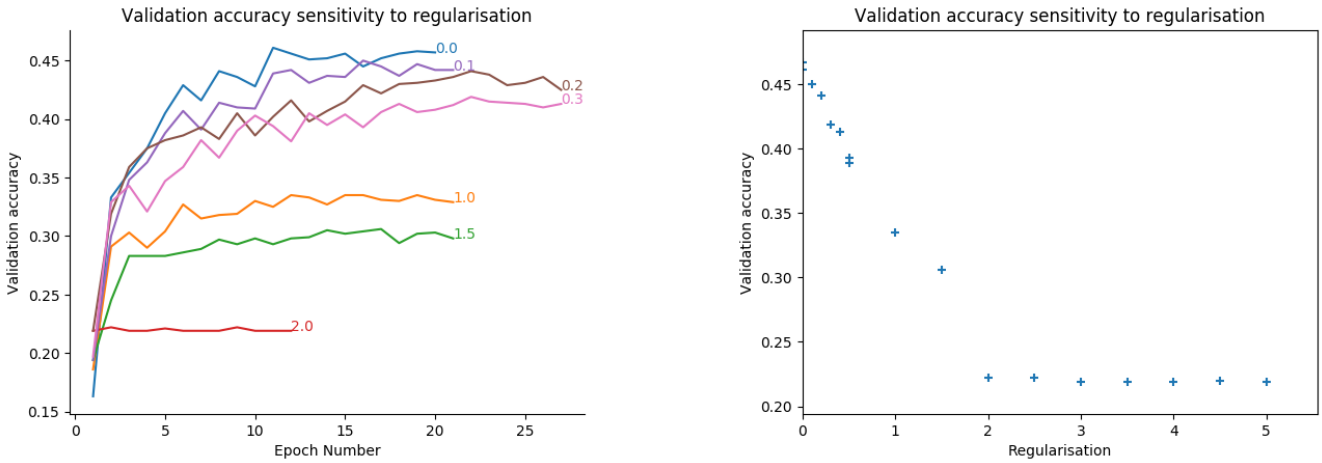


Figure 6: Effect L2 regularisation on the initial architecture with optimised learning rate. The validation accuracy curves over training *(left)* and peak validation accuracies *(right)* are shown for a range of p values. No value of reg > 0 was found to improve the validation accuracy.

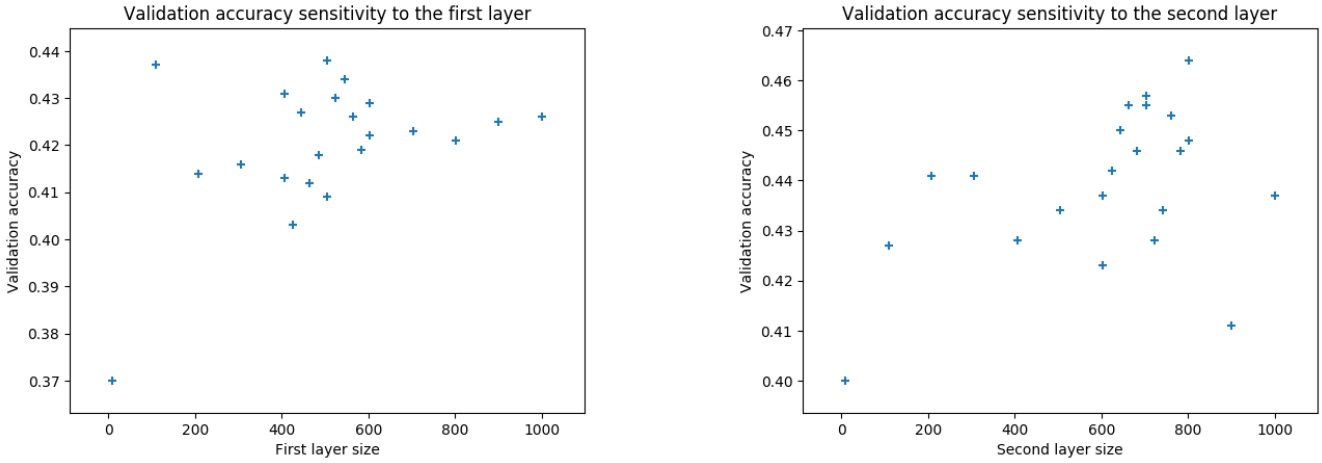


Figure 7: Optimisation of network topology. The validation accuracy as a function of the number of neurons per node is shown for the first *(left)* and second *(right)* hidden layers. A two hidden layer network with a 544×801 configuration was found to maximise validation accuracy.

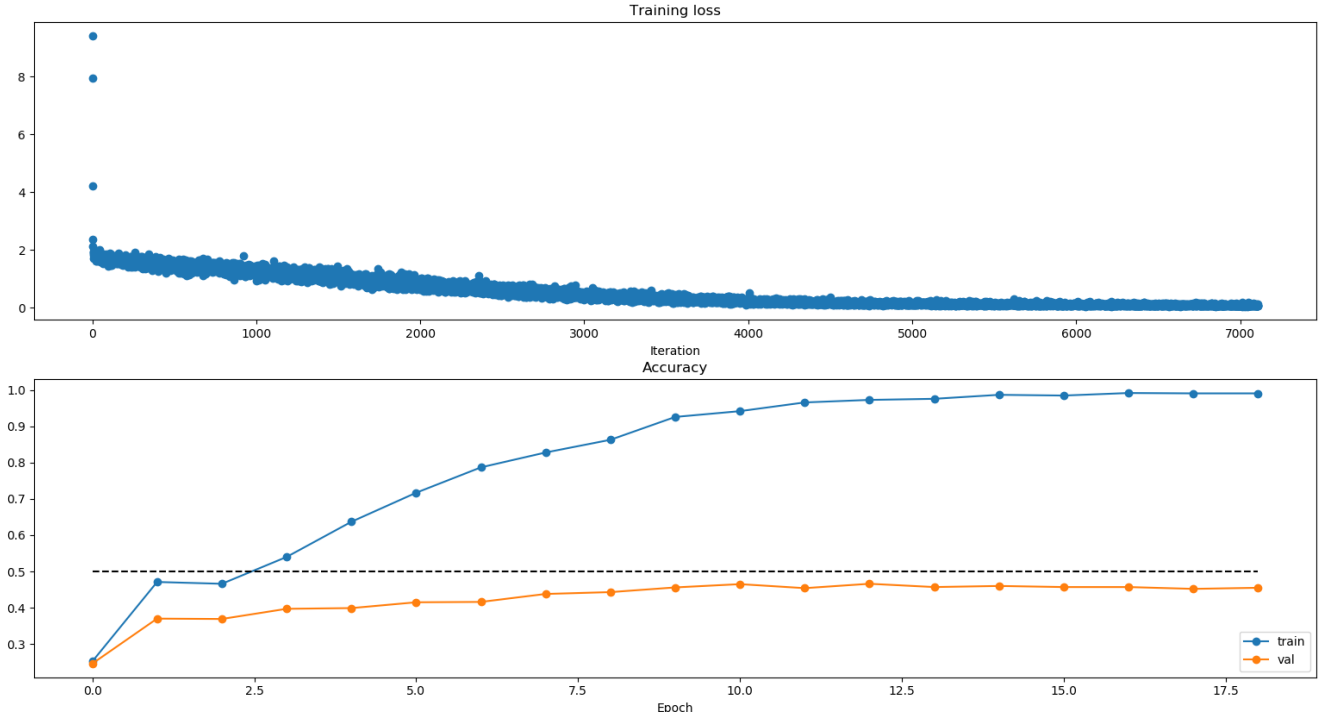
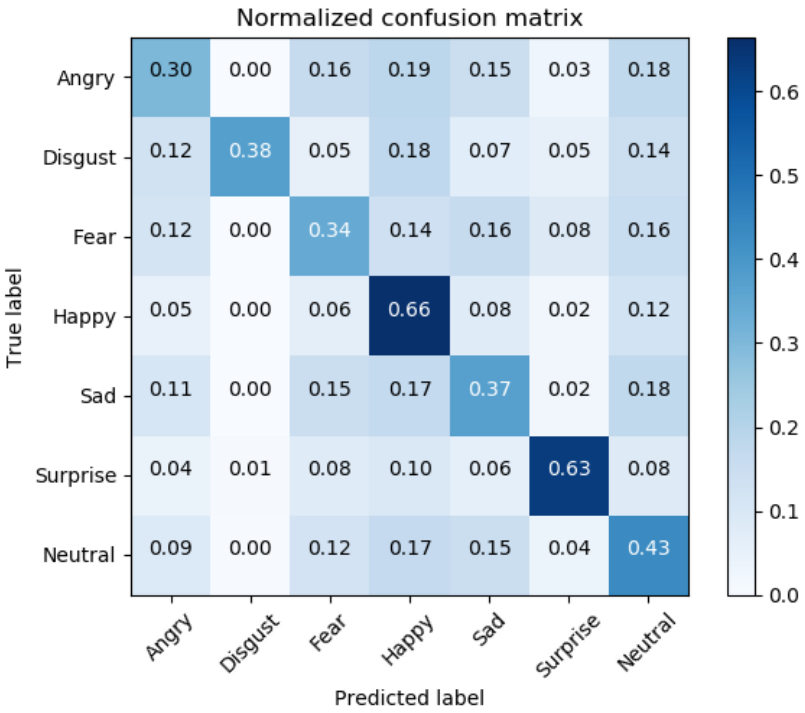
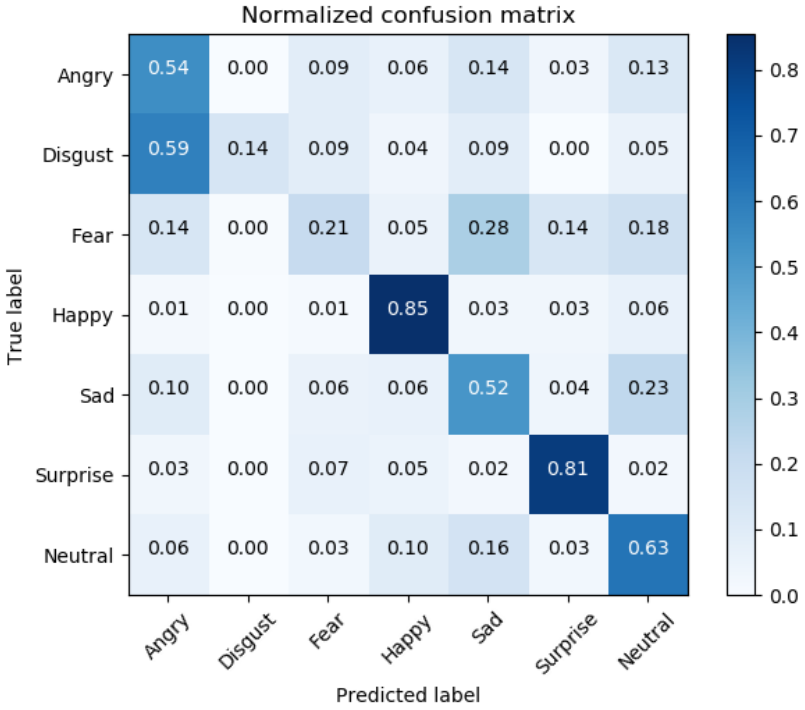


Figure 8: (Training of optimised model) Training of the three layer optimised fully connected network on the FER2013 dataset. The training set and validation set have 27709 and 1000 examples respectively, split out from the full training set. The hyper-parameters that were used for training are shown in Table 4.



(a) Feed-forward network.



(b) Convolutional network.

Figure 9: Normalised confusion matrices on the FER2013 public test data.