Questions

1) Can you think of another application where automatic recognition of hand-written numbers would be useful?

An application where automatic recognition of hand-written numbers would be useful is in sorting mail/posts at central or sub-central postal facilities which allow faster sorting of packages and posts.

Another application is for audits where the documents to be audited are mostly handwritten. It would be easier to use a digit recognizer to convert them to computable digits without much human effort.

2) In the output text in your console, how long did each epoch take to run?

There are 12 epochs as defined in our model and on an average each epoch takes 2.37s to run

A close up of text on a white background

Description automatically generated

3) Plot the training history and add it to your answers :



4) Describe how the accuracy on the training and validation sets progress differently across epochs, and what this tells us about the generalisation of the model?

From the plot above we can see that model starts off with a better accuracy on the validation set than the training set. But the training set gradually catches up to the validation set by the 4th epoch and shows a gentle increase.

Validation slightly dips in the last epoch.

In the end they have a major chunk of accuracy which is similar.

This means that the model performs almost as good on training data as on out of training[validation] data. This means that the model generalizes really well.

5) What values do you get for the model’s accuracy and loss?

| loss. | acc|

| 0.2983791| 0.9188|

6) Discuss whether this accuracy is sufficient for some uses of automatic hand-written digit classification?

Though the accuracy is not high enough for sensitive applications which require high performance and accuracy, it can be safely said that it would not be a bad choice to use the same in reducing manual work such as sorting postal codes. Because in postal codes there are various levels at which this recognizer can be checked such as the postal code, street number, house number and tied to a validator system which will raise an alarm if the letter is wrongly sorted. So 1 mistake in every 10 digits[approximately] is not such a bad thing after all.

7) How does linear activation of units limit the possible computations this model can perform?

We cannot perform non-linear mappings from inputs to outputs

Also takes longer to run as compared to using non-linear activations.

If we use linear activations here, we do not get enough information to classify the outputs correctly . All we get is a weighted average at the end [another linear function], so we may end up losing specific information. Also if we need non-linear decision boundary for our classification we wont be able to achieve that with a linear activation of units.

8) Plot the training history and add it to your answers



9) How does the training history differ from the previous model, for the training and validation sets? What does this tell us about the generalisation of the model?

The updated model with relu does not generalize as well as the previous one with the linear activation. The performance on the validation set is a lot lower than the performance on the training set and the loss is higher on the validation set than the training set as seen in the graph above. This shows that there might be some overfitting involved.

10) How does the new model’s accuracy on test set classification differ from the previous model? Why do you think this is?

| loss. | acc|

| 0.0752342| 0.9796|

11) Plot the training history and add it to your answers :



12) How does the training history differ from the previous model, for the training and validation sets? What does this tell us about the generalisation of the model?

The performance on the training set and the validation set are very similar for this model. But it generalizes better than the previous model, because it performs better on training data and out of training data. But there is still a light amount of overfitting involved, though not as much as the previous one.

13) What values do you get for the model’s accuracy and loss?

| loss. | acc|

| 0.0451061| 0.9911|

14) Discuss whether this accuracy is sufficient for some uses of automatic hand-written digit classification.

The accuracy is quite sufficient for automatic hand-written digit classification in applications where 1 mistake in a 100 digits is doable and can be checked manually furtheron. For example: In postal codes there are multiple levels of check involved where there is not just a postal code, but also house number, street number etc.

15) Describe the principles of overfitting and how dropout can reduce this?

Large neural nets trained on relatively small datasets can overfit the training data.

This has the effect of the model learning the statistical noise in the training data, which results in poor performance when the model is evaluated on new data, e.g. a test dataset. Generalization error increases due to overfitting.

Dropout prevents overfitting due to a layer's "over-reliance" on a few of its inputs. Because these inputs aren't always present during training (i.e. they are dropped at random), the layer learns to use all of its inputs, improving generalization.

What you describe as "overfitting due to too many iterations" can be countered through early stopping.

16) How does the training history differ from the previous (convolutional) model, for both the training and validation sets, and for the time taken to run each model epoch?

The loss of the model with dropout History of the training and validation sets are even closer than in the previous models. Training time was about 10 seconds longer on average. The model with the dropout rate added has a slight bit underfitting as opposed to the slight bit of overfitting in the deep model.

|  |  |
| --- | --- |
| Plot Deep Model | Plot Deep Drop Model |
|  |  |

17) What does this tell us about the generalisation of the two models?

The models generalize well. The deep model with the dropout generalizes slightly better than the deep model without the dropout.

18) What code did you use to define the model described here?

CODE:

|  |
| --- |
| modelDeepDropCifar <- keras\_model\_sequential() %>%  layer\_conv\_2d(filters = 32, kernel\_size = c(3,3),activation = 'relu', input\_shape = c(32, 32, 3), padding = 'same') %>%  layer\_conv\_2d(filters = 32, kernel\_size = c(3,3),activation = 'relu') %>%  layer\_max\_pooling\_2d(pool\_size = c(2,2)) %>%  layer\_dropout(rate = 0.25) %>%  layer\_conv\_2d(filters = 32, kernel\_size = c(3,3),activation = 'relu', padding = 'same') %>%  layer\_conv\_2d(filters = 32, kernel\_size = c(3,3),activation = 'relu') %>%  layer\_max\_pooling\_2d(pool\_size = c(2,2)) %>%  layer\_dropout(rate = 0.25) %>%  layer\_flatten() %>%  layer\_dense(units = 512, activation = 'relu') %>%  layer\_dropout(rate = 0.5) %>%  layer\_dense(units = 10, activation = 'softmax')  summary(modelDeepDropCifar)  modelDeepDropCifar %>% compile(  loss = 'categorical\_crossentropy',  optimizer = optimizer\_rmsprop(lr = 0.0001, decay = 1e-6),  metrics = c('accuracy')  )  historyDeepDropCifar<- modelDeepDropCifar %>% fit(  x\_train\_cifar, y\_train\_cifar,  batch\_size =32,  epochs = 20,  verbose = 1,  validation\_data = list(x\_test\_cifar, y\_test\_cifar),  validation\_split = 0.2,  shuffle = TRUE  )  plot(historyDeepDropCifar)  scoreDeepDropCifar <- modelDeepDropCifar %>% evaluate(  x\_test\_cifar, y\_test\_cifar,  verbose = 0  ) |

MODEL SUMMARY:

|  |
| --- |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Layer (type) Output Shape Param #  =========================================================================  conv2d\_6 (Conv2D) (None, 32, 32, 32) 896  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  conv2d\_7 (Conv2D) (None, 30, 30, 32) 9248  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  max\_pooling2d\_3 (MaxPooling2D) (None, 15, 15, 32) 0  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  dropout\_3 (Dropout) (None, 15, 15, 32) 0  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  conv2d\_8 (Conv2D) (None, 15, 15, 32) 9248  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  conv2d\_9 (Conv2D) (None, 13, 13, 32) 9248  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  max\_pooling2d\_4 (MaxPooling2D) (None, 6, 6, 32) 0  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  dropout\_4 (Dropout) (None, 6, 6, 32) 0  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  flatten\_2 (Flatten) (None, 1152) 0  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  dense\_4 (Dense) (None, 512) 590336  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  dropout\_5 (Dropout) (None, 512) 0  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  dense\_5 (Dense) (None, 10) 5130  =========================================================================  Total params: 624,106  Trainable params: 624,106  Non-trainable params: 0 |

19) Execute this model fit command. After your fitting is finished, plot the training history and put it in your answers :



| loss| acc|

| 0.83876| 0.7099|

By putting shuffle to FASLE:



| loss | acc|

| 0.8395911| 0.7073|

The data in the cifar set is not ordered. So setting a shuffle=TRUE is void and has no effect on the accuracy of the said model.



Adam improved accuracy slightly [by 0.0300]

| loss| acc|

| 0.7657529| 0.7371|

20)How does the training history differ from the convolutional model for digit recognition? Why do you think this is?

The accuracy and loss function seem to take more epochs before they plateau. This is because it takes the model longer to figure out the patterns in the data. The training loss is slightly higher than the validation loss. This model seems to be having an unknown fit because the result is counter intuitive to how the model works. This is also because of the addition of dropouts which adds some noise to the data to prevent overfitting.

21) How does the time taken for each training epoch differ from the convolutional model for digit recognition? Give several factors that may contribute to this difference

Each epoch took roughly 112.45seconds on average.

The network is more convoluted [in sense], and has a deeper layering than the previous models. Hence it takes longer to run.

22) Read the research paper “Performance-optimized hierarchical models predict neural responses in higher visual cortex”, available from: <http://www.pnas.org/content/pnas/111/23/8619.full.pdf> Write a short (~500 word) summary of the experimental approach and results.

**Problem:** Varied tunings of neurons in the inferior temporal cortex are difficult to characterize.

Objective: Modelling approach to yield a quantitatively accurate model of the Inferior Temporal Cortex. The task is to find a neural network model that matches or maybe even exceeds human performance on object recognition tasks.

**Approach:**

By means of array electrophysiology responses of 168 ITC (inferior temporal cortex) neurons to images from a set of 5760 photorealistic 3d objects in cluttered backgrounds. The backgrounds were randomly selected natural scenes, to ensure that the background-content is uncorrelated with the object.

Then high throughput computation was used to evaluate other neural network models on the same image set measuring categorization performance as well as ITC neural predictivity.

Categorization performance was measured on Support Vector Machines Linear Classifier and cross validation testing was done on them. To assess the neural predictivity, a linear regression for each target ITC neuron site was used. This was mapped to identifying a synthetic neuron built on linear weighting of the model outputs that would resemble or match that space on fixed sample images. This was then tested for response predictions against actual neural site outputs on novel images.

Models were built from large parameter space of CNN’s. This was used to approximate the general retinotopic structure of the ventral system through spatial complexity in any region of vision identical to other places. The CNN layers were stacked hierarchically to create deep neural networks.

**Results**: The steps followed show that there is optimization involved to directly guide neural mechanisms. A model with perfect neural predictivity in Inferior Temporal Cortex will exhibit high performance because the ITC itself does. The converse being true is demonstrated within a biologically plausible model class which is made by combining high throughput computational and electrophysiology techniques to explore biologically plausible hierarchical neural network models and then measure them against V4 and ITC. This is also used to show that there is a strong correlation between a model’s performance on high variation object recognition and translating it to predict individual ITC neuron firings. It is also proved that top down performance thresholds directly shape the intermediary visual representations.

Category ideal observer models were significantly less predictive of IT responses than the high performing model constructed via hierarchical modular optimization. This suggests that high performance on category selectivity does not suffice to explain IT responses. The results suggest that further structural constraints on the shaping of the visual cortex are imposed by the hierarchical network architecture.

23) Play around with these settings and see how they affect your ability to learn classification of different data sets. Write down what you found and how you interpret the effects of these settings. Depending on your inclination and how long the other questions took you, this may be 10 minutes work or an hour

Findings:

We need to classify a bunch of points based on their location in a 2d image. The given image is our training data set[ considering] with a bunch of points classified as blue and the other half classified as orange. Our objective here is to create a neural network, given no prior knowledge, can figure out if a given point should be blue or orange and predicts successfully which classification it should be. We know ahead of time the correct classification for each of the points using which we will train our neural network.

Starting with a dataset that we want to play with. The inputs are the x and y coordinates of each data point. So, for classification our neural network has to only work with these two values and they start off as equally weighted. So each of the inputs are connected to neurons in the hidden layer by the factor of a weight, which can be adjusted/manipulated to create the learning that we want. These in turn are fed into more hidden layers or the output neurons, which will ultimately decide which classification will be predicted. Keeping in mind, this is a binary classification problem, ie. Either blue or orange. Thus we only need a single signal in actuality which comes into the output.

The thickness of the connections signify their weights.

Activation function: We should try not using a step function. A popular activation function alternative could be ReLU/ Sigmoid.

Learning Rate: The step size in the gradient descent that we are doing.

We can either add more neurons to a layer or add more layers. Activation function has a significant effect on the Test Loss and if used well, can reduce overfitting to a large extent.

Regularization:

Regularization Rate:

Additionally adding more output layers and hidden layer neurons, we get a very nice learning on the model with the slightest bit of overfitting which can of course be tweaked by adding more of the above.

24) What is the minimum you need in the network to classify the spiral shape with a test set loss of below 0.1?

The minimum needed is:

Inputs: X1 , X2, Sin(X1), Sin(X2)

Hidden Layers : 1 with 3 x neurons

Learning Rate: 0.01

Activation : Sigmoid [changed from tanh to Sigmoid]

Regularization: L2

Regularization Rate : 0[default]

Fun Fact: without regularization collinear variable’s coefficient estimates will explode into infinite solution space, but with regularization, even with the regularization parameter tending to zero, we will have stable values. One of the uses of regularization is to tackle multicollinearity.

<https://stats.stackexchange.com/questions/162011/is-setting-lambda-equal-to-zero-the-same-thing-as-not-applying-regularization-at>

Problem Type: Classification[default]

Ratio of training to test data: 90:10

Noise : 0

Batch Size: 1 [Mini batch -> gives more improved accuracy but is computationally more expensive]

Most of the parameters have been left as default as the question asks for the “minimum” needed in the network to classify the data. Before the 230th epoch, we have already achieved Test loss of below 0.1 as asked in the question.

A screenshot of a computer

Description automatically generated