Character Recognition Neural Network: A Commentary

Dublin City University
CA686 Foundations of Artificial Intelligence
Patrick Travers 21267100
patrick.travers3@mail.dcu.ie
5/12/2021

Problem definition

Problem outline

The problem is to build a neural network to classify 'doodled' digits (DD). This is a comparable undertaking to building an MNIST classifier. The difference between MNIST data and doodles is the former depicts digits hand-written on paper whereas the latter are hand-written on a computer screen.

Problematisation of MNIST classifier as doodle recogniser

Throughout this project, MNIST classifiers are applied in the context of doodle recognition. A number of weaknesses with this approach in general should be made explicit. There are systematic differences in hand-written digits as compared to doodled digits which may affect a model's generalisability across these tasks. The edges of MNIST digits exhibit the nuances of authentic pen stroke. To mimic this characteristic a blurring effect is added to doodles throughout this project. However, it is not guaranteed that this measure will be enough to dupe the model into viewing both input classes equally. Also, hand-written digits are more likely to intimate the right- or left-handedness of the writer than doodles—at least this seems to be the case on casual observance of MNIST images. These generalised characteristics may underlie systematic differences between MNIST images and doodles, affecting an MNIST model's doodle recognition capabilities.

Pre-existing solution (Model I)

Model definition

The Coding Train's 'Character recognition neural network' which we will term Model I is a feed-forward neural network with 64 hidden nodes. Since it was designed as an MNIST classifier, it accepts 784 input nodes (MNIST images are 28x28=784). It has 10 output nodes, each giving the probability that the input's ground-truth represents the ten digits 0 to 9.

Notes on model appraisal

Although Model I was designed as an MNIST classifier, we will appraise it here as a DD classifier; DDs are the main focus of this project. Evaluation of Model I as a DD classifier was done in two steps. First, the model was trained until testing accuracy on unseen MNIST digits exceeded the arbitrary threshold of 93%. Then, many DDs were manually inputted within Ancient Brain's UI to observe and record the model's performance.

In carrying out initial checks on Model I's performance, it was observed that there are certain 'quirks' in the model that can be exploited in order to force the model into giving a specific response, without faithfully inputting a legitimate digit. Having gained this knowledge at an early stage in the project, it was deemed appropriate to evaluate the model in two separate ways: firstly, using normal or 'naïve' DDs (NDD), where a faithful representation of a DD was inputted. The second mode of appraisal was 'gamed' DDs (GDD) where knowledge of the model's quirks was exploited to achieve a given output, with little regard for inputting a faithful representation of a ground-truth digit.

A description of each GDD is given in Table 1 below (Appendix 1 gives visual examples of both NDDs and GDD used in model appraisal). Note throughout these GDD descriptions, 'centre' means the effective centre of the input space. On the Ancient Brain UI, this effective centre was learned through trial-and-error, as it is actually further to the left than a naïve user may anticipate.

Appraising Model I's performance

On NDDs, the model correctly identified digits on 33% of trials. This figure was found by inputting 100 NDDs (10 of each digit) on the Ancient Brain UI and recording output correctness. There were significant differences on the model's performance depending on digit. The model achieved 7/10 or higher on digits 0, 2, and 4. 6/10 was achieved on digit 6. However, between all digits 1, 3, 5, 7, 8, and 9 there was a grand total of five correct outputs. Evidently performance on these digits needs to be improved.

On GDDs, the model was correct on 74% of trials. 7/10 or higher was achieved on all digits except for 0. Zero's GDD had previously been found very effectively in eliciting the desired response, but the approach is not failsafe as demonstrated here. On ten trials, no GDD zeros

Desired Output	GDD Description
0	A 'normal' zero, made to straddle full height of input area
1	Straight line down input-space centre
2	Straight line from lower centre towards bottom-right corner
3	A shallow three amounting to a wiggled line down the centre
4	Horizontal line straddling input width, slightly below mid-height
5	Straight line from upper centre towards top-right corner
6	A six with a small loop in lower input and exaggerated looped arm
7	Horizontal line at very top of input and diagonal downward line
8	A narrow theta in mid-input space
9	A small circle in mid-input space

Table 1. A description of GDDs used in model appraisal. For visual examples please see Appendix 1.

were correctly identified. Even so, the improved doodle recognition performance with GDDs compared to NDDs is clear.

In the process of gathering these figures and the prior process of identifying GDDs, a qualitative observation was made about the model's performance—it is very sensitive to the doodle's positioning in the input space. This was especially noticeable with the digit 1. Unless the doodle is drawn in exactly the correct location, the model fails to recognise the digit (please refer to the subtle difference between digit 1's GDD and NDD in Appendix 1). There are two contributary factors worthy of discussion. Firstly, the MNIST dataset on which the model is trained is pre-processed so that digits are always positioned centrally. Secondly, the fact that Model I is a feed-forward network means that it is quite sensitive to which input nodes in particular are active. In contrast, a convolutional neural network would take greater account of the digit's overall two-dimensional shape and would be less dependent on the exact location of the digit in the input space.

Improvement areas identified

An adjusted model which we will term Model II should improve on Model I's NDD recognition score, especially on digits 1, 3, 5, 7, 8, and 9. As a secondary improvement, Model II should make it more difficult to achieve elevated scores through GDDs. To reliably elicit a given response from the model, a faithful representation of that digit should be necessary.

Model II

Changes made

Some changes between the improved Model II and the previously discussed Model I are discussed presently. Numbers in parentheses refer to relevant lines of code.

Two variables *vshift* and *hshift* are introduced (55-58) as parameters defining how much vertical and horizontal translation will be allowed at random.

A utility function getRandomInt() is introduced (100-103 for later use within the augment() function

The augment() function is Model II's key feature (227-273). It takes an MNIST input array and returns that same input with some random vertical and/or horizontal translation. The function is called on lines 291 and 324.

Difficulties and failed experiments

Initially the project had aimed to draw on data augmentation tools from Tensorflow.js. However, random cropping and padding is not supported in Javascript as it is in Python. Tensorflow's offerings were an encumbrance and were eventually abandoned for the currented implementation of augmented().

Gaining familiarity with the intricacies of JavaScript's Uint8Array datatype was a hurdle in itself.

Experiments were carried out with differing numbers of hidden nodes, but no significant improvements were observed that would have warranted extra computation.

Appraising Model II

On NDDs, the model scored 81%. This is a significant improvement on Model I's NDDs and even surpasses Model I's GDD score. Again, these figures were achieved by inputting 100 NDDs (10 of each digit). As before, there are distinctions to be made on performance across digits. Apart from 7 and 9, Model II identified all digits with at least 8/10 accuracy. Scores for 7 and 9 were 2/10 and 3/10 respectively. It is worth noting that Model I had also failed on digits 7 and 9. Model II made significant improvements on digits 1, 3, 5, 8, and to a lesser extent 6.

On GDD's, Model II also correctly identified 81% of inputs. All digits were identified with at least 8/10 accuracy except for 7 which was never correctly identified, and 4 where the score was 5/10. GDD performance was marginally better than Model I's GDDs. Perhaps the most pertinent observation is that Model II cannot be 'gamed' to the same extent as Model I; no significant performance improvement is achieved by using GDDs versus NDDs with Model II.

There is a qualification to these results which needs mentioning. Model I evaluation was carried out once a test score of 93% was observed on unseen MNIST digits. Model II was evaluated once a *stable* score of 93% was observed. In effect, Model II was trained for more iterations which may have influenced the differing scores.

Why the improved performance

As noted above in Model I's 'Improvement areas identified', the uniformity of MNIST data was posing difficulties for doodle recognition. MNIST images are pre-processed so that digits are centred. This leads to the poor performance when presented with messy user-inputted data. Model II added data augmentation measures which made the model more generalisable and prevented over-dependence on input space positioning.

While it was hoped that Model II's alterations would prevent GDDs from performing well, this was not the case. GDDs perform just as well as NDDs on Model II. A potential antidote to this persistent issue is discussed presently.

A weakness of feed-forward neural networks in general when it comes to digit classification is that they take little account of two-dimensional visual features. Rather, they focus on detecting patterns, arrangements and combinations of specific input nodes. This is in contrast to convolutional neural networks which are capable of considering two-dimensional features, focusing more on shape and other distinctive visual cues. It is little wonder that the state-of-the art on MNIST recognition is dominated by convolutional networks. Perhaps such a network would be better equipped to overcome Model I and Model II's susceptibility to being deceived by GDDs.

Further scope for improvements

The least satisfactory outcome of this entire project was that recognition of digits 7 and 9 was only achieved using GDDs. Neither model I nor Model II succeeded in recognising genuine attempts to doodle these digits. This may be caused by these digits' input nodes forming a subset of other digits' inputs. For example, inputs representing digit 9 may, in general, be subsets of a typical input for digit 8. Further investigation will be needed to establish if this is the case, and whether this could be affecting the models' poor performance on these digits.

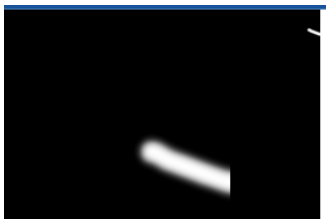
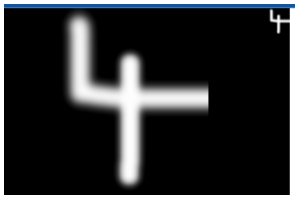
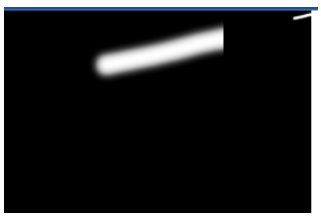
Another matter that hasn't been understood in the present analysis is why Model I is capable of identifying GDDs of digit 7, whereas this capability was lost in Model II.

As a general criticism of this analysis, perhaps its focus on GDDs is undue. The discussion of GDDs is intended to illustrate a general weakness of feed-forward networks for digit recognition, in that they can be manipulated into reliably giving a desired response using inputs that nowhere near resemble that digit's ground-truth. As shown in Appendix 1, of the GDDs used in this analysis, digits 2, 4, 5, and 9 are especially wayward from a genuine depiction of the ground-truth. However, who is to say which output(s) the model *should* tend towards, given these inputs? Perhaps it is clear that 9's GDD ought to be classified as a zero, but in all other cases perhaps it is unfair to criticise the model for producing the results discussed above.

Potential future works

The above analysis has clearly stated how and why a convolutional network may offer improved performance on doodle recognition. An obvious next step would be to test such a network on the task. An MNIST model by Andrei Karpathy offers a ready template for this endeavour. For reasons outlined above, it is anticipated that such a model may overcome key remaining weaknesses in Model II—it's failure on doodles representing 7 and 9, and its susceptibility to deception by GDDs.

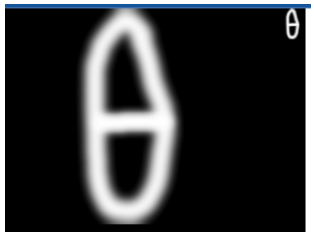
Appendix 1: typical examples of NDDs and GDDs used in the analysis

Ground truth	Normal doodled digits (NDDs)	Gamed doodled digits (GDDs)
0		
1		
2		
3		
4		
5		

6

7

8



9