CSC345: Object Recognition

Jac Royston Rees   961103

November/December 2020

Contents

1 Introduction

This report and its corresponding Jupyter Notebook relate to implementing machine learning algorithms (*models*) to classify a dataset (*CIFAR10*) of 1000 test images into 10 distinct object categories. The models are be trained with 10,000 training images consisting of 1000 images for each category. The training data is stored within a 4-D array of shape 32x32x3x10000, the training labels are stored with shape 10000x1, the test images are stored with shape 32x32x3x1000 and the test labels are stored with shape 1000x1. Image dimensions correspond to the **HWCS** (Height, Width, Colour, Sample) format.

This report proposes using a *Convolutional Neural Network* (CNN) [1]; a network which takes inspiration from the visual cortex and its structure - aiding the model to recognizing patterns based upon the geometrical similarity of the data without being diverted by the position of the features within the image [2]. Using an optimisation technique known as *random search*; a stochastic algorithm which searches a large, predetermined configuration space to find the optimum solutions for maximising the accuracy of the neural network architecture [3] alongside a process known as *data augmentation.*

This report proposes that with the methods previously stated above, an extremely viable solution for the problem at hand can be achieved. The process of incrementally testing and adapting the network architecture using various techniques allow rapid growth in *test-set accuracy* and thus a better-suited model. The model is evaluated according to various criteria: precision and recall, test-set accuracy and validation accuracy. The final model results are placed within a covariance matrix for further evaluation of the incorrectly classified classes.

2 Method

2.1 Feature Processing

Initially, the dataset is loaded into several variables named as follows: $x_{train}, y_{train}, x_{test}$ and y_{test}. Following this, by examining the shape of the dataset we can gain a greater understanding of what is required to make it trainable. By evaluating the dataset we can see that to obtain an individual image, the array format $x_{train}[: , : , : , image\_index]$ is used according to the dimensionality of the data.

The datasets are then reshaped by a process known as transposition; the act of restructuring the dataset by re-arranging the dimensional structure of the data [4]. For example, in our dataset we have transposed the data from **HWCS**, into shape **SHWC**, which now enables us to visualise images according to the new array format $x_{train}[image\_index, : , : , :]$.

The data is then shuffled in-order to remove avoidable overfitting and a method known as *one-hot encoding* is used to transfer the labels of the data from a denary format (0-9) into a corresponding binary feature vector in which each value corresponds to the similarity of the data to all the known categories [5]. For example, a label of value 7 (which corresponds to a an image depicting a horse) are encoded into a feature vector [0,0,0,0,0,0,1,0,0,0].

Data normalisation is then applied; a scaling technique in which a new range is calculated from the original range [6], by making the values closer together, aiding prediction and classification methods [7]. There are a vast number of normalisation techniques, the one discussed in this paragraph is *min-max normalisation.* A simple technique in which the data is fit within a pre-defined boundary from with a pre-defined boundary (see equation 1) [8], guaranteeing all features been scaled the same amount.

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - xmin} \tag{1}$$

2.2 Model Selection

2.2.1 Selected model

The classifier used for this experiment is the *Convolutional Neural Network*, due to its ability to solve classification problems by passing data through a series of convolutional filters and simple non-linear activations [9]. They aim to use spatial information between pixels within an image to aid in the classification of the datasets [10] and are extremely effective at reducing the number of parameters in a dataset without a loss on model quality and since images are high-dimensional, they are extremely efficient at such problems and thus are an obvious choice for this experiment.

2.3 Training And Testing

2.3.1 Training The Classifier

CNNs are trained by passing input into the initial *convolution layer* l_0 with the shape of the image (32x32x3) and producing a feature map as output through applying a set of filters whose parameters are to be learned (height and width of the filter is smaller than the input volume) and sliding this filter along the height and width of the input volume and computing the dot products between the filter and input, where the *stride* (how much we slide) parameter is equal to (3,3); passing the result through an activation function such as the Re-Lu (see equation 2) or the soft-max(see equation 3) functions [11]. Convolution layers also appear deeper into the network architecture, and when the layer is of format l_i where $i > 0$ it conforms to the following structure: an input of $m_1^{(l-1)}$ feature maps are extracted from the previous layer with each feature map having size $(m_2^{(l-1)} \cdot m_3^{(l-1)})$ The output of the layer l consists of $m_1^{(l)}$ feature maps with size $(m_2^l \cdot m_3^l)$ [12]. There are various regularisation techniques implemented throughout the network including *batch normalisation* which re-centers and re-scales the input layers

$$ReLu = max(0, x) \tag{2}$$

$$Softmax = \frac{e^8 k}{\sum_j e^{s_j}} \tag{3}$$

The output of the convolution layers are then passed into a *pooling layer*, replacing the output of the previous layer $l_i - 1$ with a max-summary (only passes the maximum values within the group of activations) from a small window within the feature map to reduce the size of the data (sub-sample) whilst concurrently reducing processing time and down-sampling the output [13](see equation 4). This is achieved by applying a window function $u(n \cdot n)$ to the small window patch of size=(2, 2) and outputted a feature map with decreased resolution.

$$a_j = \max_{N \cdot N}(a_i^{n \cdot n} u(n, n)) \tag{4}$$

The final layer, which is used to classify the data is the *dense layer*, which connects each layer to every other layer in a feed-forward fashion, in our case of size 10 - one for each category, and is outputted in the format of one-hot encoding [14].

But the question remains: How do CNNs learn? CNNs learn in a similar way to many other models, however, instead of solely learning one filter and weight, it learns multiple within each layer to obtain the optimal result. The purpose of training the CNN is to minimize the error between what the network outputs, and the desired output on the data's corresponding label. The error function is defined below and has the adaptability to function for weights and bias [15].

$$J(\theta) = -\frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{K} [y_k^{(i)} log \hat{y}_k^{(i)} + (1 - y_k^{(i)}) log(1 - \hat{y}_k^{(i)})] \tag{5}$$

The error gradient is calculated through partial-derivatives of the error function with respect to the weighted sum of the input to a neuron, known as *backpropogation* by going back along the network to derive the gradient. Once the error gradient has been derived, various optimisation techniques can be applied to train the network, which are described in the following paragraphs.

The first optimiser which used is Stochastic Gradient Descent (SGD) an optimisation technique which, at each iteration, takes a small *step* in the direction of the negative gradient to minimize the loss value, the parameter $\alpha > 0$ is known as the learning rate, which controls the size of each step at each iteration. The algorithm assigns error proportionally according to each weight within the network [11].

The second optimiser is the *Adam*(adaptive moment estimation) optimiser; an optimisation technique which computes individual learning rates which adapt for different parameters from estimations based upon first and second moments of each gradient [16].

The third model is compiled with the best performing optimisation algorithm from the initial two variants of the model and the previously stated random search algorithm to gain a model architecture which is finely tuned for the dataset, I will not be going into too much detail regarding the tuner in this report - if it is of interest there is detail regarding the algorithm in the corresponding notebook.

Finally, the model which after evaluation is deemed the most optimal, is re-trained with a regularisation technique known as *Data Augmentation*, a regularisation technique which generates new training instances from existing data and thus synthetically increases the size of the data set and thus has the advantage of reducing over-fitting by giving an altered version of the original data at each epoch [17].

2.3.2 Testing The Network

The network is tested through applying the model to a data-set of previously unseen images, known as the *test set*. The reason for this is to provide an unbiased evaluation of the model which was fit on the training data and observe how the model generalises to unseen data.

This is only one aspect of testing - during the models ”training phase”, the training set itself is split in a 80%:20% format, with 20% being held back as a *validation set*, which functions in the same way as the test set, but is evaluated at each epoch [18]. Validation sets and test sets differ by the fact that test sets are completely new and unseen data from a separate data set, and validation sets are selected from within the training data.

3 Results

3.1 Evaluation Metrics

Evaluation of the model can be done in various ways. The way this report evaluates the model is according to test-set accuracy, whereby the prediction labels outputted by the model are compared to the actual labels (y_{test}) and a percentage is formed by taking the number of correct predictions divided by the number of predictions made.

however, this is not enough to evaluate a model due to factors such as classes on the border of classification, has it only been classified as that particular class since its the most similar? Due to this, other measurements must be taken in the form of *precision and recall.* Precision is defined as the fraction of relevant examples (true positives) among all of the examples which were predicted to the class(see figure 6). Recall is defined as the fraction of examples which were predicted to belong to a class in comparison to all of the examples that truly belong in the class(see figure 7).

$$precision = \frac{truepositives}{truepositives + falsepositives} \quad (6) \qquad recall = \frac{truepositives}{truepositives + falsenegatives} \quad (7)$$

When these methods are combined, we can achieve a evaluation metric named the *F-Score*, defined as the harmonic mean of the model’s precision and recall (see figure 8) [19].

$$F_{score} = \frac{truepositives}{truepositives + \frac{1}{2}(falsepositive + falsenegatives)} \quad (8)$$

3.2 Quantitative Results

Test-Set Accuracy And F_1 Score			
Model	Test Loss	Test Accuracy	F_1 Score
SGD	1.5564	44.43%	44.43%
ADAM	1.0927	61.80%	61.80%
Tuned	1.5921	69.30%	69.30%
Tuned W/ Data Augmentation	1.2297	75.20%	75.20%

A table depicting the evaluation metric results from the experiments.

Each model of the experiment had an initial run of 30 epochs each. Whereby the SGD and ADAM models were then compared to one another through quantitative data analysis and the metrics previously stated. It is clear that the Adam optimisation model outperforms the SGD model at every test, achieving an F_1 Score which is 17.37% higher than that of the SGD model. From this analysis, the Adam model was the clear option to select for model tuning.

The same process was once again carried out, however, this time, with the tuner, 100 different and unique models were compiled and ran with 30 epochs each. At each convolution layer, the default neuron number was equal to that of the previous models, however, the tuner had the opportunity to select a random number of units between 32-256, each with a step size of 32. The regularisation parameters of each of the models were all identical (the recommended average of 0.5 for dropout regularisation).

The outcome of tuning the original best-performing model was a model which, once again, was able to achieve an F_1 Score 7.50% higher than that of the original Adam Optimiser by simply changing the number of units at each convolution layer. Whilst this is a good result, the tuned model still performs with a much larger test-loss than the original ADAM model and thus, by re-training the tuned-model (which already has a higher-test set accuracy) with the process of Data Augmentation the test-loss has decreased by a value of 0.3624 and the F_1 score has increased in accuracy by another 5.90%. Due to the training data consistently changing during augmentation, and since the tuned-model was already the most efficient model; the tuned-model was trained with 120 epochs.

$$\begin{bmatrix} \mathbf{72} & 5 & 4 & 1 & 2 & 1 & 0 & 2 & 5 & 8 \\ 1 & \mathbf{92} & 0 & 1 & 0 & 0 & 0 & 0 & 3 & 3 \\ 7 & 2 & \mathbf{58} & 4 & 7 & 8 & 8 & 1 & 0 & 5 \\ 3 & 1 & 5 & \mathbf{56} & 1 & 17 & 8 & 0 & 3 & 6 \\ 1 & 1 & 2 & 1 & \mathbf{73} & 0 & 14 & 3 & 3 & 2 \\ 0 & 0 & 3 & 15 & 4 & \mathbf{58} & 7 & 4 & 4 & 5 \\ 0 & 2 & 2 & 3 & 4 & 0 & \mathbf{88} & 0 & 0 & 1 \\ 2 & 0 & 4 & 8 & 3 & 5 & 0 & \mathbf{71} & 1 & 6 \\ 4 & 2 & 1 & 0 & 0 & 0 & 1 & 0 & \mathbf{89} & 3 \\ 5 & 10 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \mathbf{84} \end{bmatrix}$$

The matrix to the left of this text depicts the *covariance matrix* of the tuned and augmented model. It is clear that the model has accurately classified the majority of the classes, however, it is apparent that some classes can become confused with others. Most notably, class 3(cat) is misclassified into class 5(dog) a total of 17 times throughout the 100 evaluations. This shows that although the model is of a high standard, there is no guarantee that the model will consistently give accurate results. Further evaluation shows that the worst-performing class with only 56% of predictions being accurate was the cat, this is due to the consistent incorrect classification between cats and dogs, something which even a human not paying attention could get wrong! The most accurate class was the Automobile classification, with the CNN achieving an accuracy of 92%. From these results, it is clear that more unique objects within a group of objects are much easier to classify than similar ones.

4 Conclusion

To conclude, Convolutional Neural Networks are an extremely efficient method for image classification problems due to their ability to perform multiple operations on data in a high-dimensional state and inspiration from the human visual cortex.

It is evident that implementing a Convolutional Neural Network is a process of trial and error in trying to avoid local optima in the search area and it is vital to understand the requirements of the system before the implementation, e.g. What test accuracy is deemed adequate? Is computation speed an important factor? What are you actually trying to achieve through this model?

By implementing various models and incorporating external optimisation techniques (such as Random Search and Simulated Annealing) it is possible to achieve a model capable of performing image classifications to a high-standard if you are prepared to follow false-trails along the way and spend time tuning each individual parameter.

Whilst CNNs are extremely efficient models (as shown in this report), it is clear that if the data-set at hand isn't to a suitable standard or size (many aren't due to the cost increase of obtaining new data) using various regularisation methods such as data augmentation to synthetically create new pieces of data can at times have adverse effects on the functionality of the model. This can be seen from the covariance matrix provided, by translating the position of the image, certain neurons may not be activated to classify such an image and during training may cause a decrease in accuracy.

When conducting future work relating to image classification, CNNs are a prime candidate even though it has become clear that to produce an extremely efficient and accurate model there is no substitute for data-set size and as such I would set out to obtain more data for each class and will take this knowledge with me for my next project - mask-identification image classifier with a CNN.

References

[1] Yann LeCun and Yoshua Bengio. *Convolutional Networks for Images, Speech, and Time Series*, page 255–258. MIT Press, Cambridge, MA, USA, 1998.

[2] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36:193–202, 1980.

[3] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.*, 13:281–305, February 2012.

[4] Lingqun Liu. Data transposition with proc report, 2006.

[5] Patricio Cerda, Gaël Varoquaux, and Balázs Kégl. Similarity encoding for learning with dirty categorical variables. *CoRR*, abs/1806.00979, 2018.

[6] Luai Al Shalabi, Zyad Shaaban, and Basel Kasasbeh. © 2005 science publications data mining: A preprocessing engine.

[7] S. K. Patro, P. P. Sahoo, I. Panda, and K. K. Sahu. Technical analysis on financial forecasting. *ArXiv*, abs/1503.03011, 2015.

[8] S. Gopal Krishna Patro and Kishore Kumar Sahu. Normalization: A preprocessing stage, 2015.

[9] Jayanth Koushik. Understanding convolutional neural networks, 2016.

[10] Stanford University. Introduction to convolutional neural networks, 2018.

[11] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[12] David Stutz. Understanding convolutional neural networks, 2014.

[13] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, January 2014.

[14] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269, 2017.

[15] Patel Dhruv and Subham Naskar. Image classification using convolutional neural network (cnn) and recurrent neural network (rnn): A review. In Debabala Swain, Prasant Kumar Pattnaik, and Pradeep K. Gupta, editors, *Machine Learning and Information Processing*, pages 367–381, Singapore, 2020. Springer Singapore.

[16] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014. cite arxiv:1412.6980 Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

[17] Aurlien Gron. *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems.* O'Reilly Media, Inc., 1st edition, 2017.

[18] Jason Brownlee. What is the difference between test and validation datasets, 2017.

[19] Yutaka Sasaki. The truth of the f-measure. *Teach Tutor Mater*, 01 2007.