Convolutional Neural Networks for Natural Image Classification

Fola Ogundero
Student ID: F035372
Course: Artificial Intelligence
Module: 20COP508 - Machine Learning
Module Lead: Dr. Baihua Li
Loughborough University

***Abstract*—In this project the capabilities of different neural network techniques including, Multi layered and single layered networks and the intricacies within their parameters and functionalities to further accuracy and prevent loss, while convergence is observed regarding standardisation via the use of confusion matrices.**

I. Introduction

Image classification is a subfield of problems within image processing concerned with categorising samples into semantically related groups. In this project, a neural network classifier is implemented from Keras – an open-source Python library. While other algorithms, such as naive Bayes, support vector machines (SVM), and decision trees, could be used for other classification tasks, image data forms a high-dimensional input vector which increases the complexity of features substantially. Neural networks possess a large number of parameters with which to extract these features, and thus are able to outperform the other algorithms.

II. Data

The dataset used was obtained from Kaggle.com, comprising of 6,899 images in JPEG format divided into eight distinct groups: airplane, car, cat, dog, flower, fruit, motorbike, and person. Though most of these images are below 500 pixels in width and length there were some images found within the flower class which significantly surpassed this with the greatest width being found in this category being 2737 pixels and the greatest height being 2655. The lowest height and width found within all the images was a width of 43 pixels and the lowest width being 50. In terms of the size of the images the smallest was found to be only 3 KB while the largest was 2,559 KB. All eight of the image categories had different origins which will be referenced below.

- Airplane http://host.robots.ox.ac.uk/pascal/VOC
- Car https://ai.stanford.edu/ jkrause/cars/car$_{dataset.html}$
- Cat https://www.kaggle.com/c/dogs-vs-cats
- Dog https://www.kaggle.com/c/dogs-vs-cats
- Flower http://www.image-net.org
- Fruit https://www.kaggle.com/moltean/fruits
- Motorbike http://host.robots.ox.ac.uk/pascal/VOC
- Person http://www.briancbecker.com/blog/research/pubfig83-lfw-dataset

III. Methods

A. Input

The input layer of the network contains neurons encoding the values of the input pixels. We initially build a dictionary with the given names of our subdirectories of our populated dataset folder reading through the directories, the training data for the network will consist of many 32 by 32 pixel greyscale images converted with the use of opencv through its cv2.IMREAD-GREYSCALE and cv2.resize functions. the input layer contains 1024=32×32 neurons in other terms each image is an 32x32 array, which when flattened out to be a 1-d tensor of size 1024. The input pixels are greyscale, with a value of 0.0 representing white, a value of 1.0 representing black, and in between values representing gradually darkening shades of grey.

B. Pre-processing

There were 5 main pre-processing tasks undertaken to ensure the data could be used correctly. Firstly, concatenating the 8 dictionaries and all the images within the associative arrays into the X dataset. Secondly, the Y dataset is created and iterates over the “airplane” dictionary to fill the Y dataset for each element in the total length of the dictionary, this is then followed up by extending each of the other dictionaries: car, cat, dog, flower, fruit, motorbike and person contents into the dataset by iteration over its argument and adding each element to the dataset and extending the dataset, where the size of the dataset increases by the number of elements in the

dictionary argument. In the next step the images are rescaled in order to obtain the range [0,1] by dividing by 255 which is the maximum value of the pixels.
The formula to get this [0,1] range, is as follows.

$$\frac{data - min(data)}{max(data) - min(data)}$$

$$\frac{data}{max(data)} = \frac{data}{255}$$

The third step, expanding on what was mentioned earlier in the input section encompasses the reshape of the data using NumPy to create a 1d greyscale 32 x 32 array where the -1 in the array represents an unknown dimension that numpy will solve by looking at the length of the array and remaining dimensions to ensure the criteria for the new array shape is compatible with the original shape.

Next, the categorical data input are converted into a dense numpy array(0 or 1's) for storing response variables in dummy columns indicating the presence of a particular label or not, through scikit-learns LabelBinarizer function. The final step in preprocessing before building the model is the action of splitting the data arrays into two subsets for training and testing the data, reducing the chances of inaccurate predictions and miscalculations by lowering the chance of under and overfitting. Though the default test size for the split if the parameters are not set is 0.25, the chosen ratio to follow abides by Paretos principle(2) in which a ratio of 80/20 or 20%.

C. Model

Model: take a weighted sum of the features and add bias to get the logit. Convert the logit to probability via the logistic-sigmoid function.

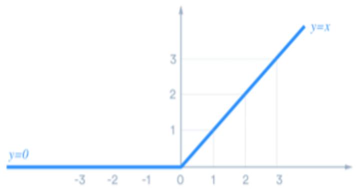
$$Logits = X * W + b$$

$$Y_{Predicted} =^{softmax(logits)}$$

Logits = X*W +b Y predicted = softmax(logits) Loss = cross_entropy(Y, Y predicted), where Y is a one-hot vector

The model is created sequentially meaning where each tensor has the same 1 input and output tensor for each layer which is ran in order based on creation. The network consisted of 2 convolutional layers which create a convolutional kernel for the input. These layers start with 32 filters followed by 64 filters and both use a 3x 3 kernel activated by Relu.

ReLU is a kind of activation function in which its results are represented as linear among all positive values while zero for all its negative values allowing it to have easy and fast training, it corresponds easily as well as its linearity implying that the trajectory would not plateau or peak as the so therefore the gradient doesn't disappear as it does in less optimal functions such as seen in a sigmoid function. Lastly it is sparsely activated in which every input becomes unlikely to activate while a negative input is supplied resulting in sparsity consequently leading to a greater predictive capacity and leading to the chance of overfitting being reduced as well as noise within the network.[5][6][7]



graph from source[4]

$$f(x) = x^{+} = \max(0, x)$$

function from source[8]

Additionally, a pooling is added to merge semantically similar features into one while reducing dimensionality, this is carried out by computing the maximum of a local patch of units into one or few feature maps defined by the pool_size.[9] Negatives of this approach being that parts of the data are lost inevitably due to the dimensions being reduced.

A dropout layer of 0.25 is also included in order to make the network less responsive to the weights of each neuron , thereby, resulting in enhanced generalisations and a lower possibility of overfitting the training data[10][11]. There are 2 dropout layers the 1st appearing after the max pooling and 2nd being later on after the flatten and dense layers which will be mentioned later on in this paper.

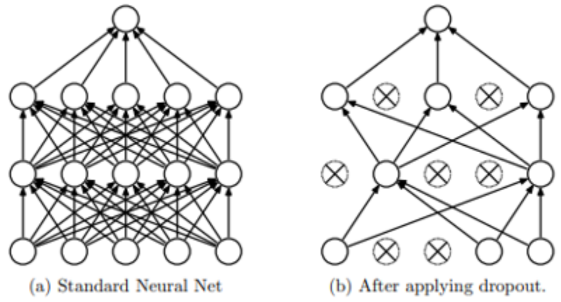


figure from source[12]

A flatten layer followed by the dense layer is included to flatten the input our array without affecting the batch size allowing for mapping between the now flattened input tensor and first hidden layer which in this case is the dense layer. Meaning that the object that was now the serialised input tensor, can be connect to each element within the hidden arrays rather than being inexplicit.[13][14]

The last decision layer within our network is a "SoftMax" function. This is a sigmoid for normalising the output from various classes into numbers within an interval of (0,1).Therefore, the model output can regarded as the likelihood that the entity being referenced too is belonging to the relevant group, this facilitates the networks interpretation classification of the input vector by the output from the network.[15][16][17]

$$\sigma_i(z) := \frac{\exp(\lambda z_i)}{\sum_{j=1}^{n} \exp(\lambda z_j)}, 1 \leq i \leq n.$$

figure from source[18]

The 8 neurons of this SoftMax being the 8 distinguished classes airplane, car, cat, dog, flower, fruit, motorbike, and person.

D. Outputs

The output layer of the network contains 8 neurons. If the first neuron fires, i.e., has an output 1 , then that will indicate that the network thinks the image as an airplane. If the second neuron fires then that will indicate that the network thinks the image is a car. And so on. A little more precisely, by numbering the output neurons from 0 through 8, and figure out which neuron has the highest activation value. If that neuron is, say, neuron number 6, then our network will guess that the input image was a fruit, and that pattern continues

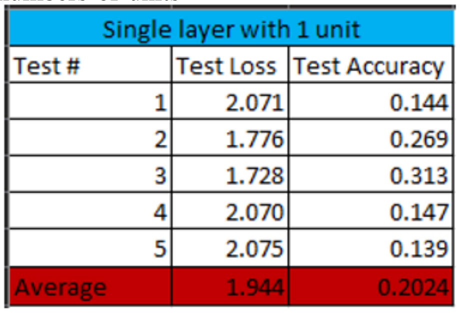
IV. Experiments

A. Multilayered vs Single Layered

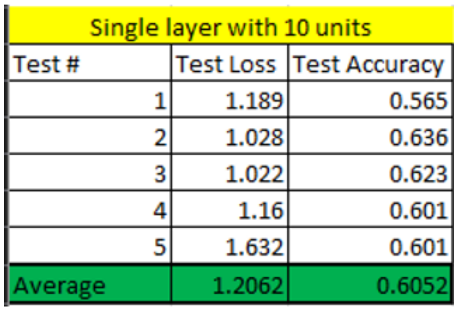
The first experiment carried out was comparing the results of converting our multi-layered CNN into a simple single layered network of just 1 Unit, whereby the weighted inputs are directly sent from the inputs to the outputs rather than going through one or more hidden layer. The number of units being the number of neurons in that layer.[19][20][21]

One important aspect of using a single layer network being that the flatten function was needed to be called in order to collapse the dimensions of the networks shape to a dimension of 1 which is the requirement of a dense layer.[22]

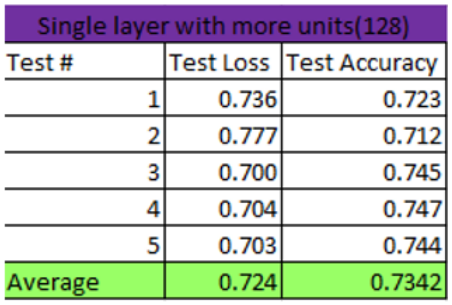
Below are the results for single layered networks with different numbers of units



Single layer with 1 unit		
Test #	Test Loss	Test Accuracy
1	2.071	0.144
2	1.776	0.269
3	1.728	0.313
4	2.070	0.147
5	2.075	0.139
Average	1.944	0.2024



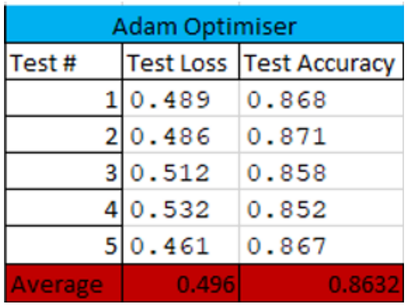
Single layer with 10 units		
Test #	Test Loss	Test Accuracy
1	1.189	0.565
2	1.028	0.636
3	1.022	0.623
4	1.16	0.601
5	1.632	0.601
Average	1.2062	0.6052



Single layer with more units(128)		
Test #	Test Loss	Test Accuracy
1	0.736	0.723
2	0.777	0.712
3	0.700	0.745
4	0.704	0.747
5	0.703	0.744
Average	0.724	0.7342

B. Optimiser

The next set of experiments involved were testing the network against different optimisers, The one found to have the best set of results as expected was the Adaptive Moment Estimation (Adam) optimiser a backpropagation method in which gradient descent optimisation algorithm which minimises our error function when comparing the labels to the output of the network by calculating dynamic learning rates on each parameter on top of, caching the decaying averages of past gradients and squared gradients .[23][24]

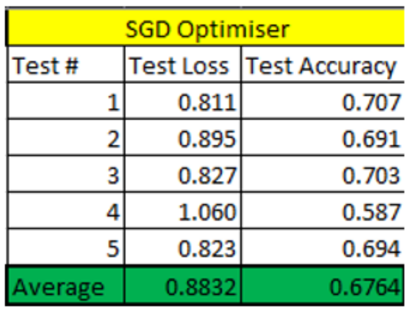


Adam Optimiser		
Test #	Test Loss	Test Accuracy
1	0.489	0.868
2	0.486	0.871
3	0.512	0.858
4	0.532	0.852
5	0.461	0.867
Average	0.496	0.8632

After this, further experiments were carried out using a different optimiser this time stochastic gradient descent (SGD), which carries out an update for each training example x(i) and labely(i)[25][26]

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta; x^{(i)}; y^{(i)}).$$

SGD Function[27]

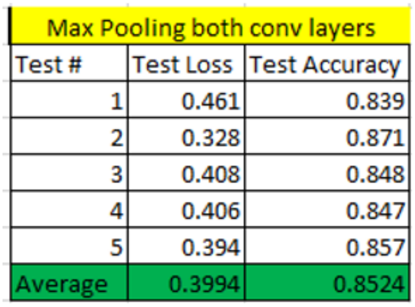


SGD Optimiser		
Test #	Test Loss	Test Accuracy
1	0.811	0.707
2	0.895	0.691
3	0.827	0.703
4	1.060	0.587
5	0.823	0.694
Average	0.8832	0.6764

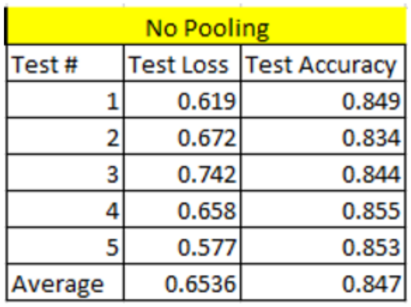
The Adam optimiser was highly shown to be advantageous where not only the algorithms speed and efficiency increase but the training time was significantly shorter

C. Pooling

Pooling layers to combine semantically similar features into one and reduce dimensionality were tested. Several studies were looked at when carrying out the literature review by which it was found to be useful[28] other alternatives being the implementation of sided convolutions which would allow the automated spatial down sampling.[29]



Max Pooling both conv layers		
Test #	Test Loss	Test Accuracy
1	0.461	0.839
2	0.328	0.871
3	0.408	0.848
4	0.406	0.847
5	0.394	0.857
Average	0.3994	0.8524

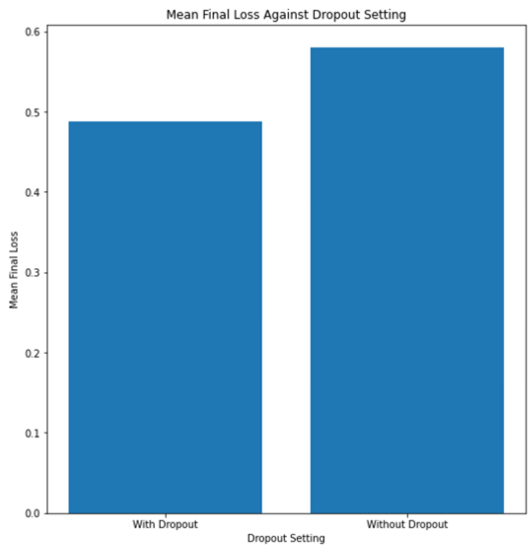


No Pooling		
Test #	Test Loss	Test Accuracy
1	0.619	0.849
2	0.672	0.834
3	0.742	0.844
4	0.658	0.855
5	0.577	0.853
Average	0.6536	0.847

What can be seen from the results is that the effect of implementing pooling was positive resulting in almost twice as low loss while also having a high average test accuracy.

D. Wilcox Ranks Testing

The last test that was carried out was a Wilcox and Ranks test, this is important in finding out the likeliness that the result from the first and second run are coming from the same probability distribution.



As shown from the Figure, through the Wilcox and ranks test, the final loss by an average of 0.09 was reduced in the network. The significance of the test was shown across 20 trials both with and without dropout showing that no more than 0.1% probability that the results from the trials share distribution. From this what can be understood is that there is >99% chance that the resulting better results were due to dropout.

V. Conclusion

This study's results provide further evidence of the efficacy of convolutional neural networks the classification of common-place, real-world objects in images. The architecture may be further optimised through the fine-tuning of hyperparameters such as network topology, optimiser function, or the application of dropout to minimise overfitting. Further work may be done to explore methods from existing literature, which may be applied to this project in the further improvement of everyday performance as well as, skip connections to avoid problems such as the vanishing gradient problem.

References

[1] https://www.codecogs.com/latex/eqneditor.php
[2] Dunford, R., Su, Q., Tamang, E. (2014). The pareto principle.
[3] Brownlee,Jason.(2019). A Gentle Introduction to the Rectified Linear Unit (ReLU). https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/
[4] Jang, E., Gu, S., Poole, B. (2016). Categorical reparameterization with gumbelsoftmax. arXiv preprint arXiv:1611.01144.
[5] Liu, D. (2017). A Practical Guide to ReLU . https://medium.com/@danqing/a-practical-guide-to-relu-b83ca804f1f7
[6] Goodfellow, I., Bengio, Y., Courville, A. (2016). Deep learning. MIT press.
[7] Li, Q., Cai, W., Wang, X., Zhou, Y., Feng, D. D., Chen, M. (2014, December). Medical image classification with convolutional neural network. In 2014 13th international conference on control automation robotics vision (ICARCV) (pp. 844-848). IEEE.
[8] Sharma,S.(2017).Activation functions in Neural Networks. https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6
[9] LeCun, Y., Bengio, Y., Hinton, G. (2015). Deep learning. nature, 521(7553), 436-444. doi:10.1038/nature14539

[10] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research, 15(1), 1929-1958.

[11] Wu, H., Gu, X. (2015). Towards dropout training for convolutional neural networks. Neural Networks, 71, 1-10.

[12] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research, 15(1), 1929-1958.

[13] Ketkar, N. (2017). Introduction to keras. In Deep learning with Python (pp. 97-111). Apress, Berkeley, CA.

[14] 14. Liu, X., Yang, D., El Gamal, A. (2017, October). Deep neural network architectures for modulation classification. In 2017 51st Asilomar Conference on Signals, Systems, and Computers (pp. 915-919). IEEE.

[15] Su, H., Xu, H. (2015). Multi-softmax deep neural network for semi-supervised training. In Sixteenth Annual Conference of the International Speech Communication Association.

[16] M. Zhang, W. Li and Q. Du, ”Diverse Region-Based CNN for Hyperspectral Image Classification,” in IEEE Transactions on Image Processing, vol. 27, no. 6, pp. 2623-2634, June 2018.

[17] Jang, E., Gu, S., Poole, B. (2016). Categorical reparameterization with gumbel-softmax. arXiv preprint arXiv:1611.01144.

[18] Gao, B., Pavel, L. (2017). On the properties of the softmax function with application in game theory and reinforcement learning. arXiv preprint arXiv:1704.00805.

[19] Hayashi, Y., Sakata, M., Gallant, S. I. (1990). Multi-layer versus single-layer neural networks and an application to reading hand-stamped characters. In International Neural Network Conference (pp. 781-784). Springer, Dordrecht.

[20] Agrawal, P., Girshick, R., Malik, J. (2014, September). Analyzing the performance of multilayer neural networks for object recognition. In European conference on computer vision (pp. 329-344). Springer, Cham.

[21] Antipov, G., Berrani, S., Dugelay, J. (2016). Minimalistic CNN-based ensemble model for gender prediction from face images. Pattern Recognit. Lett., 70, 59-65.

[22] Ketkar, N. (2017). Introduction to keras. In Deep learning with Python (pp. 97-111). Apress, Berkeley, CA

[23] Bock, S., Goppold, J., Weiß, M. (2018). An improvement of the convergence proof of the ADAM-Optimizer. arXiv preprint arXiv:1804.10587.

[24] Zhang, Z. (2018, June). Improved adam optimizer for deep neural networks. In 2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS) (pp. 1-2). IEEE.

[25] Keskar, N. S., Socher, R. (2017). Improving generalization performance by switching from adam to sgd. arXiv preprint arXiv:1712.07628.

[26] Bottou, L. (2012). Stochastic gradient descent tricks. In Neural networks: Tricks of the trade (pp. 421-436). Springer, Berlin, Heidelberg.

[27] Ruder,S.(2016). An overview of gradient descent optimization algorithms

[28] Shifat-E-Rabbi, M., Yin, X., Fitzgerald, C.E. and Rohde, G.K. (2020), Cell Image Classification: A Comparative Overview. Cytometry, 97: 347-362. doi:10.1002/cyto.a.23984

[29] Radford, A., Metz, L., Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434.