M45J: OBJECT RECOGNITION

Comparing the use of AlexNet and ResNet Convolutional Neural Networks to classify objects

Elizabeth Simpson, 951428
951428@swansea.ac.uk

Introduction

The aim of this paper is to discuss ways that machines can accurately classify images. Convolutional Neural Networks (CNN) are a machine learning technique used to create a model that can accurately predict the classification of unknown objects (Yang, Choi and Lin, 2016). However, there are many different architectures that can provide powerful frame works for classification. In this paper we will discuss which CNN architecture creates the most effective machine leaning model. We implement two CNN architectures – AlexNet and ResNet and compare the accuracy of the models using a subset of the CIFAR-100 dataset. The CIFAR-100 dataset is a large collection of objects with two sets of labels for the same dataset, "Fine" labels and "Coarse" labels. "Coarse" labels is less detailed and therefore splits the objects into fewer categories. Firstly, we implement an AlexNet CNN model, we train the model using fine and coarse labelling sets and compare its effectiveness to a more recent architecture, the ResNet CNN. Both models were scored on their learning time, accuracy of results using coarse labelling, accuracy of results using fine labelling and, on the accuracy of predicting objects from an unseen dataset. We will then evaluate the results by comparing our findings to the work completed by Sharma et al. to analyse whether our results are accurate and suggesting reasons why our models may differ.

Method

In this section we will discuss the dataset and the implementation, training, and testing of the AlexNet and ResNet Networks. We will discuss how features are extracted, normalised and how the models were trained and tested. A Convolutional Neural Network (CNN) is a deep learning algorithm that takes in images and over many iterations of mathematical functions tries to assign importance to aspects (features) of the data set in order to create a model that can accurately classify unseen images from an equivalent dataset (Sumit, 2018) . All CNN's will be written in Jupyter Notebooks using the Tensorflow and Keras framework. I chose to use CNNs for this project as it is a supervised technique and can provide high accuracy results for image recognition.

The Dataset

A large dataset is needed to build a CNN. In this instance I used a subset of the CIFAR-100 dataset. The CIFAR-100 subset contains 32x32 images of 50000 objects, ranging from plants to tractors. As CNNs are a supervised learning technique, each image is labelled with a "fine" (more specific) and "coarse" (more general) label (Krizhevsky, Nair and Hinton, 2009). Fine labels identify 100 classes while the coarse dataset identifies 20 super classes. We will analyse the effectiveness of AlexNet and ResNet models using both label variations.

Pre-processing of the data is needed to ensure all the data is within the appropriate format. Before using this data in a model, the data will be shuffled as to avoid unnecessary bias by the ordering of the images. For each model, the dataset is split into 2 sections. 80% of the dataset will be used as training data while the remaining 20% will be withheld and used to test the effectiveness of the models. All the input images must be 32x32 pixels and normalised so that there is no degeneration of gradients. The input shape for the models will be (32,32,3) holding information about height x width x RGB. One Hot Encoding will be used to assign classes a numerical value without ordering.

CNN Architecture

CNN's are made up of multiple layers. AlexNet and ResNet both use layers of Conv2D, batch normalisation, activation, pooling 2d layers to take input data and their labels and output a classification model. However, ResNet employs skip connections which significantly lowers the complexity of previous networks and can achieve human level accuracy (He *et al.*, 2016). The difference between architecture comes from the differences in layers (Sharma, Jain and Mishra, 2018).

Brief note about Layers

Conv2d – Filters (kernels) are applied to the input. (Stewart, 2019)
Batch normalisation – A standardisation layer, used to speed up and stabilise the network by using the mean and variance of the batch to reduce dependency between layers. (Ioffe and Szegedy, 2015)
Activation Function Layers – In both models, a Rectified linear Unit (ReLu) function is used in the hidden layers. The output layer of the fully connected network will use the Softmax function. (Keras, 2020)
ReLu – Rectified Linear Unit. It will take the maximum value between 0 and the input value (Stewart, 2019).
Max Pooling layer–takes the max value of the filter only, reducing dimensionality of the network (Stewart, 2019).
Flatten – Converts a pooled feature map into a 1D array that is passed to the next layer (Stewart, 2019).
Dense – Defines the number of outputs wanted from that layer (Stewart, 2019).
Dropout – During training, random neurons are ignored, used to prevent overfitting (Stewart, 2019).
ResNet Layers:

Average pooling layers-uses the average value of the filter, reducing dimensionality of the network (Stewart, 2019)
Lambda layers – used for simple operations, allows arbitrary Tensorflow functions to be used. (Bhadani, 2019)
Concatenation layers – takes in list of tensors and reduces it into one tensor with all inputs (Chollet, 2020)

Features

Features are extracted in a CNN by using multiple layers to create a network, the AlexNet and ResNet have different architectures, both of which have an input layer, followed by multiple convolutional and pooling layers ending with a Softmax, fully connected, output layer to extract features (Bodapati and Veeranjaneyulu, 2019). A CNN follows a hierarchical model where the most significant features are processed and funnelled toward the output layer to create an optimal model.

AlexNet

AlexNet was chosen because it is a relatively simple and early implementation of a CNN, made in 2012 (Khan *et al.*, 2020). There are 9 layers, made up of convolutional (partial) Conv2D, Batch Normalisation and Pooling layers. 5 layers are partial, meaning that each node in the layer is connected only to its close neighbours in the next layer. Then there are 3 layers of fully connected nodes – all nodes in the layer connect to all the nodes in the next layer. Finally an output layer that uses Softmax activation function to reduce the outputs into our 20 output classes (Kumar, 2020). The kernel sizes start off at 5x5 and are reduced to 3x3. Dropout was also used in the fully connected layers to reduce overfitting. Overfitting is one of the main issues faced by CNN's, caused when the model is too specialised to the training data used, meaning the model doesn't correctly learn the overarching patterns. The implementation of the AlexNet CNN was based on the original code by Kumar (Kumar, 2020).

ResNet

Resnet is short for Residual Network (He *et al.*, 2016) and is a much deeper model, meaning this model has more layers. Deeper models are more accurate but are more complex and thus take longer to train. Deeper networks can also start to overfit the data so the deeper the network, the more likely your accuracy will plateau and start to degrade (Sharma, Jain and Mishra, 2018). Resnet was chosen for this project as it is claimed to be the next biggest development after AlexNet so would make a good comparison. Resnet uses blocks and skipping layers to reduce the issue of vanishing gradients (Feng, 2017), and to reduce dependency between layers, depending on the layers output. The use of blocks are considered the main performance enhancement of ResNet compared to previous architectures (Khan *et al.*, 2020). Blocks simply refer to the process when the activation layer is carried.

While there are many variants of ResNet we will focus on ResNet50. ResNet50 is the simplest version of the improved architecture with the fewest layers, reducing complexity meaning implementation, training and evaluation is possible given the time and machine constraints. ResNet50 has 50 layers and to aid in building a reliable ResNet I will be using the ResNetModel package which claimed 98% peak performance in 62 training epochs for the CIFAR-100 dataset created by McDonnell (McDonnell, 2018). In addition to the common layers of AlexNet, McDonnell's ResNet uses, lambda layers, Concatenation layers and average pooling layers rather than maximum pooling layers. The initial hypothesis is that the ResNet will be the most accurate model.

Results and Evaluation

AlexNet and ResNet models were implemented and trained on the CIFAR-100 dataset and tested on unseen test data. Overall, the AlexNet model was successful, it was easier to implement and took fewer attempts to produce a model which could classify images of unseen objects with an accuracy of 47%. However, the ResNet model produced did not behave as expected and after comparing these results with the work by Sharma et al. indicate that the model was unsuccessful.

Firstly, the AlexNet Model. Overall, the AlexNet models performed in the expected range. After training the model for 4.5 hours, 100 epochs were run with 350 steps per iteration. When trained on the coarse dataset, the accuracy produced on unseen images was 48%, with a loss (sum of errors made per example during testing) of 2.67 (Figure 1). As expected, the model performed better on the coarse dataset as there were fewer classes and therefore more obvious features that collected the images into general categories. However, the model also performed successfully on the fine labelling set, with an accuracy of 36% on the unseen test data. The loss value is larger which indicated that predictions were overall less accurate compared to the coarse data labels. The number of epochs and steps per epochs were kept the same to provide a fair test. The time taken to train both models was consistent, with each epoch taking approximately 150 seconds to complete. It is important to note that during training on both labelling sets, the accuracy value on the training data was significantly larger than the validation accuracy, this could indicate that there was some overfitting of

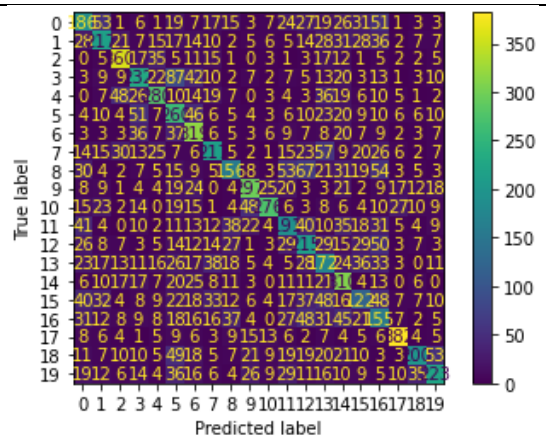
the data which should be investigated in the future, with more fine tuning of the hyperparameters, and possibly more dropout layers are needed to reduce the risk of training an overfitted model.

Table of Results (Figure 1)			
		AlexNet	**ResNet50**
Coarse Labels	**Accuracy**	94.0%	35.0%
	Validation accuracy	48.0%	5.0%
	Time to Train	4.5 hours	5.3 hours
	Test Loss	2.37	4.4
	Validation loss	2.7	65.4
Fine Labels	**Accuracy**	90.0%	28.0%
	Validation accuracy	36.0%	1.0%
	Time to Train	4.6 hours	5.3 hours
	Test Loss	3.83	
	epochs	100	50
	Steps per epoch	350	50

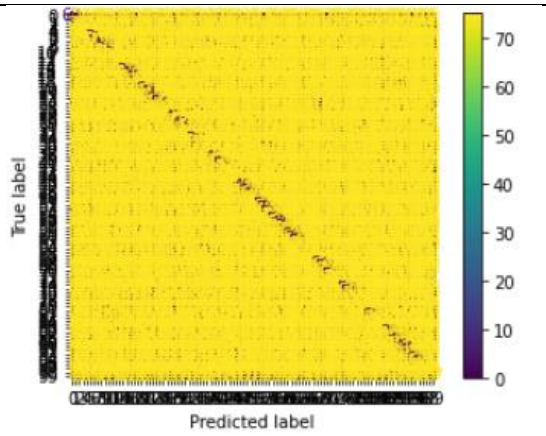
The confusion matrix in Figure 2(a) shows a clear correlation between the true and predicted labels, as expected. The matrix shows that the AlexNet model is more effective at classifying the images in label 3 (Flowers) and 18 (Trees) as these are shown in yellow, indicating a higher number of correctly identified images. The confusion matrix 2(b) indicating the fine dataset is difficult to compare to 2a due to more class information being displayed and the shorter colour scale (with yellow indicating a higher probability of the label) reduces the effectiveness of the matrix to display information comparable to Figure 2a. As the overall accuracy of the Fine labelled categorisation is lower, we would expect to see fewer defining trends in the confusion matrix as indicated by the reduced range in colours.

Confusion matrix (Figure 2)

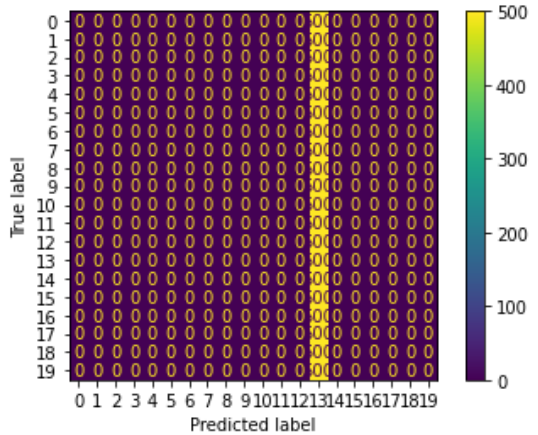
(a) AlexNet with Coarse labels



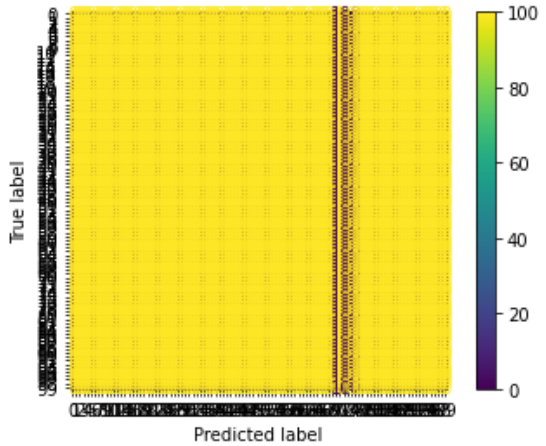
(b) AlexNet with Fine labels



(c) ResNet with Coarse labels



(d) ResNet with Fine labels

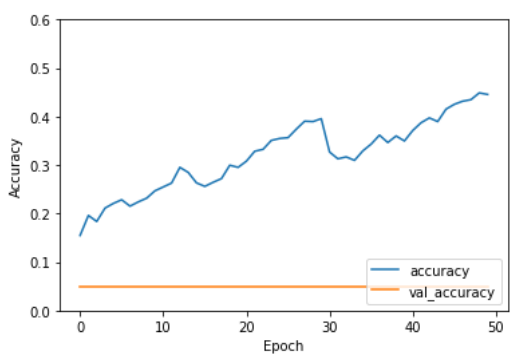


To compare my overall results against Sharma et al, while their evaluation techniques included multiple methods, their AlexNet achieved an average accuracy of 44.10% over the CIFAR-100 dataset using the finer labelling set, I would argue that the AlexNet model used in this report is comparable to their results and therefore can be considered a reliable representation of the abilities of the AlexNet architecture to classify images (Sharma, Jain and Mishra, 2018).

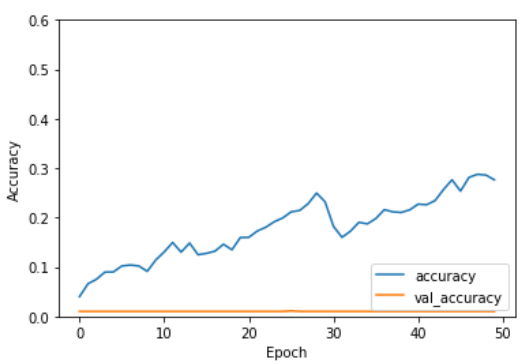
Unfortunately, the ResNet failed to train on either of the coarse labelling categories or the fine labels. The ResNets training accuracy using the coarse labels vs using the fine labels were consistent with the findings from AlexNet – the coarse labels, with fewer classes, allowed for a more accurate model to predict unseen images into the 20 classes. However, while the accuracy on the training data improved slowly, the validation accuracy never increased from its starting values on either model. From the studied conducted by Sharma et al. a ResNet approach was evaluated at being 59.82% effective at categorising labels, the implemented ResNet model only achieved 5% on Coarse labels and 1% on the fine labels. The failure of this attempt could be contributed to the reduced number of epochs, the reduced number of batch sizes and the reduced number of steps per epochs which were reduced to 50 each due to time constraints. As Batch size, epochs and epoch steps all contribute to increased accuracy, they also increase the time taken to train the model. Even with the reduction of these parameters the ResNet took longer to train than the AlexNet. Further, we can see that the number of epochs was not adequate to converge the model, from Figures 3a and 3b we can see that neither of the models hit a convergence state, meaning that the models were still attempting to find optimal parameters for classification even at the end of training. However, as the validation accuracy did not change even slightly after 50 epochs, I believe the issues was more fundamental than the need to tweak the hyper parameters.

Convergence of ResNet Models during training (Figure 3)

(a) ResNet with Coarse Labels Learning



(b) ResNet with Fine Labels



This issue may be in the layers of the ResNet itself, possibly by unfreezing layers to allow the ResNet to learn the underlying features of the dataset and reducing the number of blocks skipped could improve accuracy, however McDonnell uses the same model and number of layers effectively in his study. The model may be too big, as ResNets have many layers resulting in overfitting which would also explain why the training accuracy was so high while the validation accuracy remained low. Perhaps the learning rate may be too small/big however, the learning rate was changed throughout the ResNet training cycles with no impact. So perhaps the input data was incorrectly pre-processed as ResNets require a very specific pre-processing technique, this could have caused incorrect labels to be input leading to the confusion matrices showing the same incorrect patterns as seen in figures 2c and 2d.

In all of the above cases, hindsight would indicate that the use of number of epochs, batch size, input generators, pre-processing, learning rates should all be included in all models and kept as consistent as possible to allow for a more meaningful evaluation of the models architecture and a meaningful investigation into the failure of the ResNet.

Conclusion

From this study we can conclude that CNNs are an effective way to create models that can classify images. Studies show that a ResNet architecture has potential to outperform an AlexNet on accuracy when properly modelled, due to their depth and use of blocks to reduce overfitting. However, AlexNets can produce effective models in shorter time frames. During this study, an AlexNet CNN was implemented with an accuracy of 48% on the CIFAR-100 dataset, ResNet architecture were also attempted, however, more care needs to be taken when constructing more complex networks, with consideration given to the hyper parameters and need for precise data pre-processing.

For future work, more time is needed to fine tune a ResNet model to achieve higher accuracies. Further, with the growth of popularity in VR and AR technologies, future work should include the evaluation of 3D volumetric images using a 3D AlexNet and 3D ResNet models to progress image classification into a 3D virtual world concept.

References.

Bhadani, R. (2019) *Lambda Layers in tf.keras. Lambda layers are useful when you need… | by Rahul Bhadani | Analytics Vidhya | Medium*, *Analytics Vidhya*. Available at: https://medium.com/analytics-vidhya/lambda-layer-in-tf-keras-c4e8b94c87e (Accessed: 18 April 2021).

Bodapati, J. D. and Veeranjaneyulu, N. (2019) 'Feature extraction and classification using Deep convolutional Neural Networks', *Journal of Cyber Security and Mobility*, 8(2), pp. 261–276. doi: 10.13052/jcsm2245-1439.825.

Chollet, F. (2020) *Concatenate layer*. Available at: https://keras.io/api/layers/merging_layers/concatenate/ (Accessed: 18 April 2021).

Feng, V. (2017) *An Overview of ResNet and its Variants | by Vincent Fung | Towards Data Science*, *Towards data science*. Available at: https://towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035 (Accessed: 17 April 2021).

He, K. *et al.* (2016) 'Deep residual learning for image recognition', in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 770–778. doi: 10.1109/CVPR.2016.90.

Ioffe, S. and Szegedy, C. (2015) 'Batch normalization: Accelerating deep network training by reducing internal covariate shift', in *32nd International Conference on Machine Learning, ICML 2015*. International Machine Learning Society (IMLS), pp. 448–456.

Keras (2020) *Layer activation functions*. Available at: https://keras.io/api/layers/activations/ (Accessed: 17 April 2021).

Khan, A. *et al.* (2020) 'A survey of the recent architectures of deep convolutional neural networks', *Artificial Intelligence Review*, 53(8), pp. 5455–5516. doi: 10.1007/s10462-020-09825-6.

Krizhevsky, A., Nair, V. and Hinton, G. (2009) *CIFAR-10 and CIFAR-100 datasets*, *https://www.cs.toronto.edu/~kriz/cifar.html*. Available at: https://www.cs.toronto.edu/~kriz/cifar.html (Accessed: 13 April 2021).

Kumar, V. (2020) *Hands-on Guide To Implementing AlexNet With Keras For Multi-Class Image Classification*, *Analytics India Magazine*. Available at: https://analyticsindiamag.com/hands-on-guide-to-implementing-alexnet-with-keras-for-multi-class-image-classification/ (Accessed: 13 April 2021).

McDonnell, M. D. (2018) 'TRAINING WIDE RESIDUAL NETWORKS FOR DEPLOYMENT USING A SINGLE BIT FOR EACH WEIGHT', *arXiv*. Available at: https://github.com/McDonnell-Lab/1-bit-per-weight (Accessed: 17 April 2021).

Sharma, N., Jain, V. and Mishra, A. (2018) 'An Analysis of Convolutional Neural Networks for Image Classification', *Procedia Computer Science*, 132(Iccids), pp. 377–384. doi: 10.1016/j.procs.2018.05.198.

Stewart, M. (2019) *Simple Introduction to Convolutional Neural Networks | by Matthew Stewart, PhD Researcher | Towards Data Science*, *Towards Data Science*. Available at: https://towardsdatascience.com/simple-introduction-to-convolutional-neural-networks-cdf8d3077bac (Accessed: 17 April 2021).

Sumit, S. (2018) *A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way | by Sumit Saha | Towards Data Science*, *Towards Data Science*. Available at: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 (Accessed: 18 April 2021).

Yang, F., Choi, W. and Lin, Y. (2016) 'Exploit All the Layers: Fast and Accurate CNN Object Detector with Scale Dependent Pooling and Cascaded Rejection Classifiers', in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 2129–2137. doi: 10.1109/CVPR.2016.234.