IMAGE CLASSIFICATION WITH A CONVOLUTIONAL NEURAL NETWORK

Daniel Petterson

Victoria University of Wellington

ABSTRACT

Index Terms— Convolutional Neural Network, Image Classification

1. INTRODUCTION

Image classification has previously been a labour intensive task that required a human eye. In the past decade the use of Convolutional Neural Networks (CNNs) has become widespread in computer vision tasks due to their ability to accurately detect patterns for object identification and image classification [1]. The benefits of using a CNN over a fully connected multilayer perceptron (MLP) include mitigating the risk of overfitting in the event of insufficient data and being more memory efficient.

2. PROBLEM INVESTIGATION

We aim to compare the classification performance of a conventional MLP with that of a suitably configured convolutional neural network.

2.1. Data

The initial training dataset consists of a total of 4,500 RGB images across 3 classes, with 1,500 images per class. The 3 classes are tomato, cherry, and strawberry and the classes are mutually exclusive. As the classes are equally distributed across the dataset we don't need to consider any techniques to deal with an imbalance of images from each class.

2.2. Preprocessing

The images were drawn from photo hosting site Flickr from a number of different contributors. They are of differing resolutions so all images were scaled to 256x256 while maintaining the aspect ratio of the original images. A 230x230 crop was taken from the resized image and there was a 50% of an image either being vertically or horizontally flipped and the brightness, hue and saturation fo the image were changed slightly. The purpose of these preprocessing techniques was to improve the generalisability of the model and improve its performance on unseen data.

3. METHODOLOGY

3.1. Model Structure

The CNN is constructed as follows;

Table 1. CNN Model Structure

Layer	Activation	Regularisation
2D Convolutional	ReLU	BatchNorm
2D Convolutional	ReLU	BatchNorm
2D Convolutional	ReLU	BatchNorm
Fully Connected	ReLU	Dropout
Fully Connected	ReLU	Dropout
Fully Connected	ReLU	
Fully Connected		

3.1.1. Convolutional Layers

A conventional convolution operation involves sliding a kernel, a matrix of numbers , over the input and generating a feature map whose values are determined by the following formula;

$$G[m,n] = (f * h)[m,n] = \sum_j \sum_k h[j,k] f[m-j, n-k] \tag{1}$$

The dimensions of the resulting feature map are governed by the size of the kernel, the stride (how many points it moves horizontally/vertically after each convolution), and the padding. The depth of the output tensor is equivalent to the number of filters.

3.1.2. Fully Connected Layers

These layers are the same as those in a traditional MLP where each element in one layer is connected to each element in the following layer.

3.2. K Fold Cross Validation

By splitting the entire training dataset into k segments we are able to use k-1 segments as the training set while retaining one segment for evaluation. This process is repeated k times so that each time a different segment will serve as the holdout or validation set. By doing so we are able to get a more accurate estimate of how the model will perform on unseen data. For this experiment we set k to 5 so that for each fold we will be training on 80% of the total number of images and be using 20% for testing.

3.3. Loss Function

The loss function is crucial to the training of a model and its output represents the suitability of the current model parameters. For this experiment we use Cross Entropy as our loss function which we aim to minimise in order to optimise our model. Cross-entropy is equal to the negative log likelihood so minimising it is equivalent to maximising the likelihood i.e. performing Maximum Likelihood Estimation.

$$H(g, f) = -\sum_{x} g(x) log f(x) \tag{2}$$

$$NLL = -\sum_{j=1}^{M} y_i log \hat{y}_j \tag{3}$$

where y and $g(x)$ represent the ground truth or real distribution and $\hat{y}_j$ and $f(x)$ are the predicted probabilities or generated data so by optimising for either we estimate distribution parameters that more closely resemble the true distribution.

3.4. Optimisation Method

The final model uses the Adam optimisation algorithm as it works well with the sparse gradients generated by the ReLU function and, by way of its step-size annealing, tends to converge faster than other algorithms such as Stochastic Gradient Descent or AdaGrad [2].

3.5. Regularisation

3.5.1. Batch Normalisation

Batch Normalisation normalises the data using the mean and standard deviation of the input batch which helps to stabilise the network and increase speed in the same way that normalising is employed in data preprocessing. It is important as the distribution of inputs can shift over time and this is amplified in deeper networks which can lead to slower model training [3].

3.5.2. Dropout

Dropout reduces overfitting by deactivating a unit with probability p which in turn reduces the reliance of each unit in a layer to the units in the other hidden layers[4]. Each unit in a fully connected layer has a 50% chance of being deactivated.

3.6. Activation Function

We will be using the Rectified Linear Unit(ReLU) as the activation function for all convolutional and fully connected layers except the final layer. ReLU is a popular choice due to its resistance to the vanishing gradient effect in the backpropogation algorithm [5]. It creates sparse matrices through transforming any negative value to zero. The equation is as follows,

$$f(x) = max(0, x) \tag{4}$$

The benefits of this are two-fold, firstly it can significantly reduce the amount of memory required and secondly it can lead to faster training of the network.

3.7. Hyperparameter Settings

To correctly train the CNN to be able to classify images, many hyperparameters need to be adjusted; these hyperparameters are pivotal in determining the performance of the network and the time required for convergence. For this experiment we trained the network over 75 epochs use minibatches of 128 images and a learning rate of $1e^{-3}$.

3.7.1. Batch Size

It was found that a reduction in batch size is associated with more rapidly reducing loss but this reduction in batch size comes at significantly higher computational expense. A lower batch size coupled with a smaller learning rate can allow the network to train better especially if over a sufficient large number of epochs [6].

3.7.2. Learning Rate

There is a range at which each optimiser performs optimally; too low a learning rate never progresses, too high a learning rate causes instability and never converges. As such our learning rate was partially informed by the selected optimiser, Adam, and by the batch size.

3.7.3. Epochs

An epoch is equivalent to one complete cycle through the dataset and generally a greater number of epochs is associated with higher model performance but it can also lead to overfitting and may come at great computational cost.

Table 2. MLP Model Structure

Layer	Activation	Regularisation
Fully Connected	ReLU	Dropout
Fully Connected	ReLU	Dropout
Fully Connected	ReLU	
Fully Connected		

4. RESULTS

The baseline MLP that we compare against is structured as follows; In an MLP the first layer has the same number of neurons as there are pixels in the input image (230x230). Due to the MLP effectively being the latter segment of our CNN model it is unsurprising that it trains faster as it requires significantly less multiplications each epoch. The classification accuracy of the CNN is considerably higher than the baseline due to the convolutional layers creating feature maps that can then be used to learn features of the different classes.

5. CONCLUSIONS AND FUTURE WORK

In conclusion, Convolutional Neural Networks exhibit higher performance in image classification tasks than multilayer perceptrons in part due to their ability to automatically learn patterns that aid in accurate classification. This performance, however, is computationally expensive and it would be beneficial to look into methods to reduce the computational cost while retaining a similar level of accuracy.

6. REFERENCES

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[2] Diederik P. Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," 2017.

[3] Sergey Ioffe and Christian Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*. PMLR, 2015, pp. 448–456.

[4] Pierre Baldi and Peter J Sadowski, "Understanding dropout," *Advances in neural information processing systems*, vol. 26, pp. 2814–2822, 2013.

[5] Xavier Glorot, Antoine Bordes, and Yoshua Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2011, pp. 315–323.

[6] Ibrahem Kandel and Mauro Castelli, "The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset," *ICT express*, vol. 6, no. 4, pp. 312–315, 2020.