Deep Convolutional Neural Networks for Image Classification

Timothy McDermott[#1]

School of Engineering and Computer Science, Victoria University of Wellington
[1]mcdermtimo@myvuw.ac.nz

***Abstract*— This document gives an overview of the configuration and performance of Convolutional Neural Networks (CNN) for multi-class image classification. It presents an overview of domain issues and standard operations and enhancement techniques for improving the quantitative output of image classification models, through both model and data improvement techniques.**

***Keywords*— CNN, Image Classification, Multi-Class**

I. Introduction

A. Purpose

Machine learning (ML) is widely recognised as an effective instrument for image classification. The primary objective of classification tasks is to label an unlabelled image; this requires the model to effectively isolate a subject from noise and predict class labels with high success.

This report investigates the implementation of a Convolutional Neural Network (CNN) for this purpose and benchmarks it against other approaches. The secondary objective is the investigation of loss and activation functions, optimisation techniques, and hyper-parameter tuning from a dataset containing images of cherries, strawberries and tomatoes - objects with a high degree of overlap. Finally, enhancement and parameter settings will be evaluated based on their effect on the algorithm's overall performance

B. Problem

Image classification tasks can be easily solved with distinct and easily separable attributes in a dataset - traits like colour can be effective indicators i.e. comparing an orange to a banana, similarly shape, edges, and other metadata can be other quick indicators. Additional complexity comes from separating the subject from the background, effectively drawing this boundary is hard and the introduction of noise can make the job substantially harder. The datasets provided exemplifies these issues, the classes are of similar shape and colour, the background is noisy and some instances are not faithful to their label. To overcome this issue the model must be able to examine the image as a whole, understanding the subject in context and surrounding information.

C. Scope

This report investigates the development process and techniques to build a highly accurate convolutional neural network for image classification with similar subjects, and improvements that can be made by transfer learning through advanced models such as Resnet [6].

II. Background

Convolutional Neural Networks are powerful tools for image classification. The ability to extract features from an image, and perform classification in one model makes it highly adaptable to varied inputs with minimal fine tuning, or through transfer learning[2].

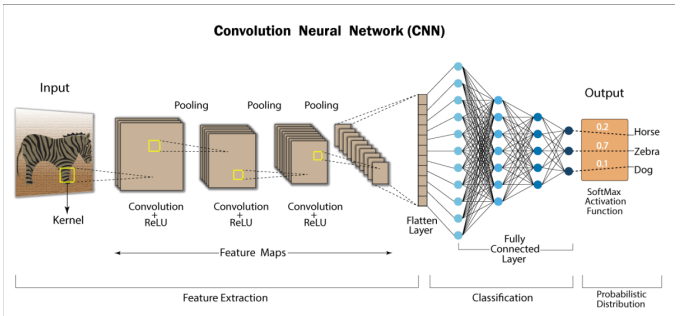


Figure 1: CNN Pipeline [7]

The pipeline for this process is reflected in Figure 1. Region(s) selected from the input with a kernel and convoluted - the input vector is manipulated by

functions which are combined to produce an output vector, or “feature map”. After the convolution operation is performed, a pooling layer is often added to the network. This layer downsamples the feature map, and aids in making the network invariant to small changes in the input. The pooling layer also helps with overfitting, by providing a form of regularization. Next, the feature map is flattened, so that it can be input into a traditional neural network - this step is known as “unrolling”. The final step of the CNN is the full connection, where the output of the previous layer is input into the next. Dropout can also be used as a regularization technique. This technique randomly drops out (sets to zero) a number of output units in a layer during training which prevents units from co-adapting too much.

III. Methodology

A. Apparatus

Hardware:

CPU: Intel i7-8700k Hexa-Core at 4.5GHz

GPU: Nvidia GTX1080 2560 CUDA Cores 8GB DDR5 VRAM at 1.7Ghz

Ram: 32 GB DDR4 at 3600 MHz

Software:

Python 3.7.6

Fast AI

Resnet18

B. Exploratory Data Analysis

Cursory analysis reveals many images that do not accurately correspond to their class, several images contain the keyword/class in a different context, such as a cartoon image of an icecream sundae on a shirt being worn by a person, or the title of a book. Fast AI recommends models be able to handle a degree of imbalanced data, however several methods are available to try to remove such outliers, our approach is outlined in the following section.

C. Pre Processing

Once identifying the presence of bad instances, they could be eliminated. Effectively working out which are bad instances could be done from the images own statistics, manually, or using a classifier. I chose to use the image classifier I was building to also perform the processing, where the top losses could be tracked to identify problem instances for removal. Through both the confusion matrix and top losses methods we can identify a range of misclassified, good, and poor images - all of which the classifier struggled on. Misclassified images are the biggest source of top losses, and can be further explored with the confusion matrix [3]. Artificial images, that is those that do not contain the subject in a natural case are also not ideal as they are not the kind of material the model will be used to classify and are then dead weight. Removal of the outliers can be done if performance is unsatisfactory [3]. In this case there were very few misclassifications, and performance does not warrant implementing an automated elimination method. I have manually deleted some of the worst cases from the training data - this would not scale.

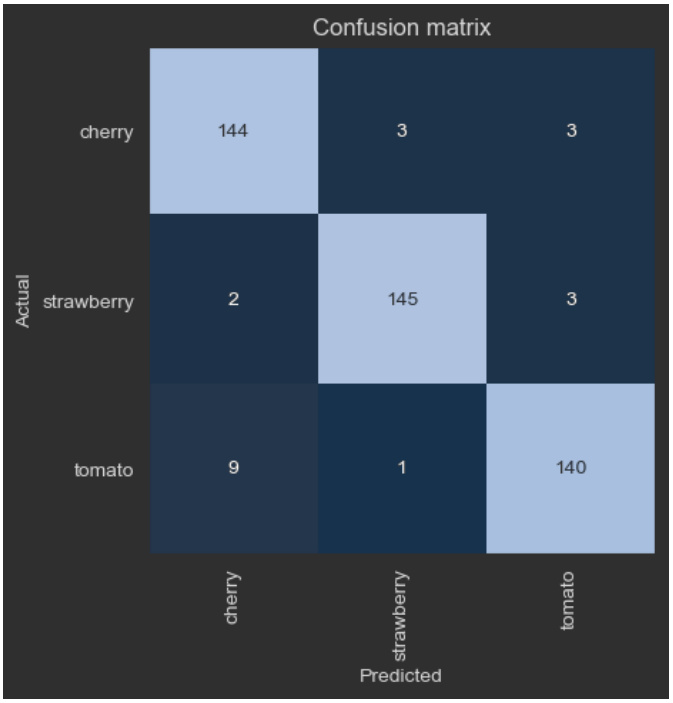


Figure 2: Classification Confusion Matrix

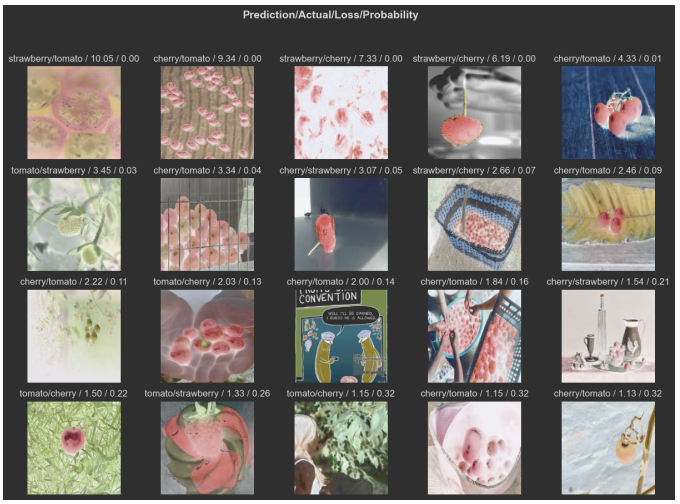


Figure 3: Top Losses

Several image processing techniques were tested to see if they were able to amplify any inter-class

variance, or eliminate background noise. Thresholding, morphological, and channel processing were tested on the images, while these were able to effectively reduce background noise, the resultant data also suffered losses, preventing background elimination without data-quality suffering. This is especially prevalent in the noisy samples, while the clearer images have better results - this leads to diminished returns as the "easier" instances are made better at the cost of the harder instances.

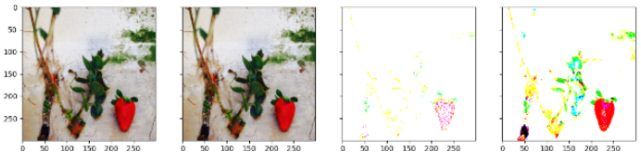


Figure 2: Channel Processing

This observation is repeated across approaches.

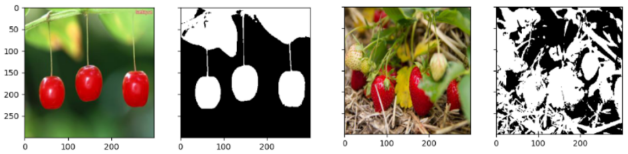


Figure 3: Threshold Processing

Ultimately these transforms were not implemented due to the loss of input data, and the additional effort required to implement and tune.

Instead, an alternative approach was taken to augment the dataset through the creation of new training data. Transformation functions were created with operators set to rotate, zoom, flip, warp, and alter lighting. The effect of these transforms is to supplement the dataset and mimic a larger training sample. These transforms make the model generalize better by eliminating dependence on coincidental attributes such as orientation - a cherry is still a cherry if it is sideways.

Figure 4: Augmentation Transforms

D. Base Model

A baseline MLP model was used as a reference point for the base CNN before optimization and switching to the transfer learning model.

MLP Model

```
model = Sequential()
model.add(Input(shape=(300, 300, 3)))
model.add(Flatten())
model.add(Dense(350, activation='relu'))
model.add(Dense(50, activation='relu'))
model.add(Dense(3, activation='softmax'))
model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
res = model.fit(train, epochs=50, verbose=1, validation_data=valid)
```

Figure 5: MLP Baseline Architecture

CNN Model

```
model = Sequential()
model.add(Input(shape=(300, 300, 3)))
model.add(Conv2D(32, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(3, activation='softmax'))
model.compile(loss='categorical_crossentropy',
              optimizer='sgd',
              metrics=['accuracy'])
```

Figure 6: Unoptimized CNN Architecture

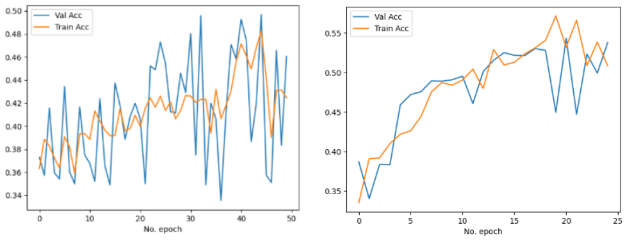


Figure 7: Average Base MLP vs Unoptimized CNN Accuracy

These models and their statistics were used as a reference point to mark the areas of improvement in the algorithm through various techniques and are sub-optimal implementations.

E. Enhancements

Activation function: RELU

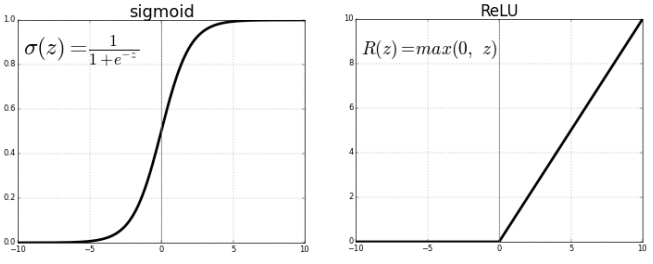


Figure 8: Sigmoid vs RELU [5]

RELU and Sigmoid are the most common activation functions. Sigmoid has some beneficial attributes, its form is more complex than RELU can be beneficial for models where the goal is predicting the probability of an output, since probabilities exist between 0 and 1. RELU is an alternative approach that has a monotonic derivative and function [1]. Computationally it is far simpler and can be performed with an if statement, while sigmoid requires an exponent calculation. This leads to faster convergence and better performance relative to the training time, leading to its selection.

Loss Function:

Fast AI Provides a range of options for the Loss function, the final model implemented the loss function as the Cross Entropy Loss Flat. A loss function is used to guide the training algorithm in the right direction, by evaluating the model's output and calculating the difference betweeN the predicted output and the true output. A loss function is required to be differentiable, the Cross Entropy Loss Flat meets this requirement while also providing a robust metric of performance. The aim is to reduce this metric by changing the weights in the neurons.

Cross entropy is a commonly used function, which is used to measure the disagreement between two probability distributions. The cross entropy loss function can be used with a SoftMax activation function, to provide a probability that the model will pick the correct class. For example, if the probability of picking the correct class is 0.5, and the probability of selecting the wrong class is 0.5, then the cross entropy is -ln(0.5). The negative sign is used to encourage a reduction in the cross entropy.

The cross entropy has a linear relationship with the number of wrong answers. However, this relationship can be improved by the use of a Log function. This takes the cross entropy formula and multiplies it by -1. The results are smaller, and the linear relationship is still maintained. The Log function is not used because it can cause numerical instability. Flattening of the output means the tensors are flattened before trying to take the losses as it is more convenient.

Minibatch Size:

The mini batch size is the number of training examples used for each iteration of training. This impacts the training time, since the gradient is calculated for the batch of samples. A larger batch size requires more memory, but can improve generalisation by increasing the number of samples used to calculate the gradient

Optimization: CUDA Acceleration

Torch support for GPU Acceleration was leveraged to make use of the CUDA cores available to the system. The GPU allows for the faster training of models by calculating the gradient in parallel across the cores. This can provide a large improvement in training time, and is one of the benefits of using a GPU over a CPU. This allows for faster runs than a CPU with the same configuration, or setting higher parameters to better explore and exploit the solution space in the same time frame.

Dataset Augmentation: Transforms

As described in preprocessing.
The number of additional images generated is not fixed but instead governed by probabilities, it is possible to calculate *exactly* how many new images are created, but they are certainly more than doubled.

Transfer Learning: ResNet18

Transfer learning is a common technique in machine learning, and is used to take advantage of the weights and building blocks provided by pre-trained models. These models are trained on large datasets, and provide a similar and effective solution to building models from scratch. This solution is effective in this scenario due to the limited amount of training data available.

ResNet18 was selected as the model to use because of its high performance and relatively light weight compared to the other larger versions, such as 34, 50, 101 - which it has even been able to outperform on smaller problems such as this one [6]. The solution was implemented by loading the pre-trained model, and modifying the final layer to be a 3-layer linear layer. This is due to the number of classes in the dataset, which is 3.

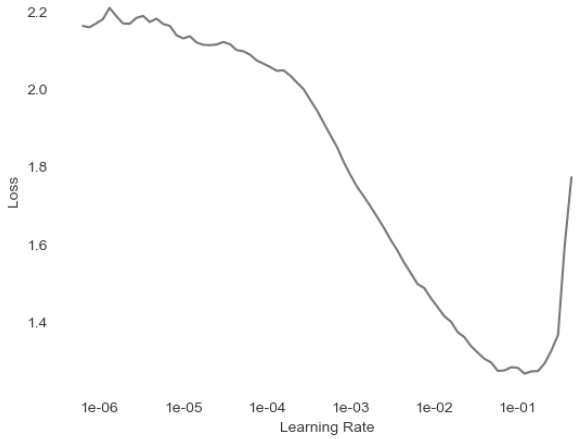
Normalization Strategy: ImagenetStats

The images were normalized using the ImageNet statistics. This is a common normalization

technique, as ImageNet is a large dataset with a high degree of variance. The mean and standard deviation are used to normalize the images. This ensures that all of the images are standardized, which is important since the neural network can only learn from the data that it is given. This technique is used with transfer learning, and is applied to the training data. This is the same dataset that was used to train Resnet, maintaining continuity between the two is not critical but may be beneficial. For the same reason I resize the images from 300x300 to 224x224.

Learning Rate Optimization.

I used the learning rate (lr) finder to identify the relationship between loss and lr, this allows you to quickly set an ideal range for learning rate from a region on the graph, capturing a varied sample of potential learning rates to test across a cycle.



IV. Findings

A. Data:

Classification Report on Test Instances

	Precision	Recall	F1-score	Support
Cherry	0.97	0.96	0.97	150
Strawberry	0.97	0.97	0.97	150
Tomato	0.94	0.95	0.94	150
Accuracy			0.96	450
Macro avg	0.96	0.96	0.96	450
Weighted avg	0.96	0.96	0.96	450

B. Interpretation

As seen in the classification report, the final metrics are far better than those of the unoptimized CNN/MLP approaches. This is of no surprise as they were designed without care to the objective to act as a bench-mark before tuning - and the addition of transfer learning provides knowledge from a far larger dataset.

Resnet18 is a powerful model, and its performance can be seen here. The accuracy is high, and there is a lower variance in the predictions. Transfer learning creates additional hidden layers and an architecture much more complicated than that of a base CNN or MLP with the additional structure improving the ability of the model to isolate the target from the background. The additional layers therefore benefit the model significantly, especially in high noise cases - though there is a certain point where the addition of layers does not necessarily help with a given problem, and as documented in [6], can actually decrease the models performance.

The overall accuracy is very consistent at 96% when tested across various splits even with the additional noise in the dataset from bad inputs. I found an additional fine tuning could improve these results for a single cycle but led to very small decreases on different splits, which could be attributed to overfitting but in overall the model generalizes well.

V. Conclusions

A. Assessment

Based on the findings in this report it is clear that Convolution Neural Networks are ideal for multi-class image classification problems. CNNs can effectively and accurately adapt to variations in dataset and can even apply learnt knowledge between domains using transfer learning. The flexibility and completeness of CNN models are huge benefits for solving generic image classification problems quickly, and the wide range of configurable operators and hyperparameters allow for higher quality outcomes and adaptability to unbalanced and low quality data.

B. Recommendation

Implementation of the advanced outlier elimination schemes discussed in [3] could lead to better performance, though it would only be worth

it at scale since the schemes are quite complex. Additional experimentation with regularization techniques such as dropout could also potentially be tuned to assist in eliminating the outlier influence if there is a lesser number of them.

Testing additional datasets with the model and running parallel instances on a grid computing network with varied seeds could also be interesting to assess the general performance and quickly iterate on variations of transforms and elimination schemes.

Acknowledgment

Causal Productions for supplying the template form, and Michael Shell and other contributors for developing and maintaining the IEEE LaTeX style files.

References

[1] A. Thakur, “Relu vs. sigmoid function in deep neural networks,” *W&B*, 19-Aug-2020. [Online]. Available: https://wandb.ai/ayush-thakur/dl-question-bank/reports/ReLU-vs-Sigmoid-Function-in-Deep-Neural-Networks--VmlldzoyMDk0MzI. [Accessed: 25-Oct-2022].

[2] H. K. Kaiming , Z. Xiangyu, R. Shaoqing , and S. Jian, “Deep residual learning for image recognition - arxiv.” [Online]. Available: https://arxiv.org/pdf/1512.03385.pdf. [Accessed: 26-Oct-2022]

[3] K. Nakanishi, “Targeting and removing bad training data.” [Online]. Available: https://towardsdatascience.com/targeting-and-removing-bad-training-data-8ccdac5e7cc3. [Accessed: 25-Oct-2022]

[4] S. Gugger, “How do you find a good learning rate,” *Another data science student's blog*, 20-Mar-2018. [Online]. Available: https://sgugger.github.io/how-do-you-find-a-good-learning-rate.html. [Accessed: 23-Oct-2022]

[5] S. Sharma, “Activation functions in neural networks - towards Data Science.” [Online]. Available: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6. [Accessed: 25-Oct-2022].

[6] V. Feng, “An overview of ResNet and its variants - towards data science.” [Online]. Available: https://towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035. [Accessed: 25-Oct-2022].

[7] Swapna., “Convolutional Neural Network: Deep learning,” *Developers Breach*, 25-Jan-2022. [Online]. Available: https://developersbreach.com/convolution-neural-network-deep-learning/. [Accessed: 26-Oct-2022].