Image Classification in Machine Learning

Y3920134

***Abstract*—In this assessment, we were aim to train a machine learning model that able to classify the image of "flowers-102 dataset" collected and prepared by researchers from the University of Oxford. The convolutional neural network(CNN) architecture was designed to categorises 8189 images of flowers into 102 categories. The main goal of this assessment is to evaluate its classification accuracy on the official flowers-102 test set. In the architecture, classification accuracy can be maximise by constructed multiple convolutional and pooling layers, then fully connected layers. This assignment is done in Python 3 with PyTorch as machine learning libraries. This model achieved 52.07 % on accuracy of test set, which compromise around 3202/6149 right answers.**

I. Introduction

In machine learning, image classification is a supervised learning technique that takes an image as input and predicts a particular class as the output. Based on my study in image classifier, most of the research has been directed towards the classification of different categories, as posed in CIFAR-10 data set. The classes in the dataset includes, airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck. Numerous recent approaches have been successful in this study. Nevertheless, our objective is to use computer vision technique to differentiate between visually similar image which pose greater challenges for the assessment.

Most previous research in Flower-102 dataset encounter the same issues which is different species that look the same both in colour and shape. Due to vast amount of images and lack of flora knowledge in me, I found the difficulty to encounter this issue. However, I found that the previous researchers that have done this image classification use different approach and method to tackle this problem.

In this assessment, I tackle the issue of image classification through the idea of convolutional neural network(CNN) model. The Flowers-102 compromises 8189 color images of flowers distributed across 102 distinct category which range around 40 to 258 per class. These images pose a range of challenges, including diverse backgrounds, different lightning conditions and different shapes,size and colour. Thus, it serves as an excellent data set for evaluating the accuracy of image classification models.

The main aim of this assessment is to create an efficient CNN architecture that can productively learn and generalise from the training data, resulting in a high level of accuracy on the test set.

II. METHOD

This evaluation focuses on developing a convolutional neural network (CNN) for the image classification from the Flower 102 dataset. The approach consist of various key stages, such as convolution, ReLU, MaxPool layers and loss functions. But before we could use the layers in networks, data preprocessing is needed.

Data preprocessing plays an important role in the machine learning pipeline. It serves to optimise model performance, eliminate outliers and preparing data for specific algorithms. It consist of resizing, normalisation and data augmentation.

For resize, train and test dataset are changed to 224x224 dimension. Then, data augmentation is needed to prevent overfitting, improve generalisation and acts as a form of regularisation. This includes flipping, rotating, cropping and colour jittering the train images. Lastly , I set mean = [0.485,0.456,0.406] and std = [0.229, 0.224, 0.225] for normalisation.

Next, I build a model architecture which plays a pivotal role in image classifier machine learning. The CNN architecture designed for Flowers-102 dataset consist of the convolutional layers, fully connected layers abd output layer. Firstly, a convolutional layer that takes an input with 3 channels, also represented as RGB colour, applies 64 filters of size 3x3 with padding 1. Then, batch normalization layer was used to stabilise and accelerate the training. Max pooling layer with 2x2 window to down sample the feature maps by a factor of 2. This process repeats for different numbers of filters for each following convolutional layer.

The output from convolutional layers is converted into a 1D tensor in fully connected layers before being passed on. Subsequently, it enters a fully connected layer with an input size of 512 * 14 * 14, followed by batch normalization and ReLU activation function. Finally, a dropout layer is utilized during training to address overfitting by randomly setting 15% of the activation to zero.

For output layer, it has a fully connected layer that outputs 102 features, which represents the number of categories in Flowers-102 dataset. Then, it undergoes batch normalisation for the final output.

Lastly, in VGG16 class, there is also a forward method. It defines the pass of the model and return input x through the sequential model of self.model.

Then, the model was trained using loss function, optimizer, batch size and epochs. Cross-Entropy was selected as loss function because the machine was designed to measure the difference between predicted and actual class of the image. Adam optimizer (Adaptive Moment Estimation) was chosen because it is effective in handling extensive datasets and their adaptability in learning rates.

III. RESULTS AND EVALUATION

Previously, I worked on my architecture in Jupyter Notebook and run the machine using my computer's CPU. Normally, it took 4 to 5 hours to train my network with only 50 epochs. After I finished my set up in Viking, I use Visual Studio Code and run the machine using Viking's GPU. It took me 2 hours, to run 600 epoch with batch size of 32. We use Pytorch libraries in this assesment and any usage of pretrained model is not allowed. Throughout the training process, performance metrics such as training and validation loss, along with accuracy, were continuously monitored. To achieve the desired accuracy and results, many adjustments were made, including changing the architecture, modifying data augmentation techniques, and tuning hyperparameters.

At first, convolutional neural network (CNN) was selected as the architecture to train the dataset for the assessment. However, after further research about the Flowers-102 dataset, I realised that VGG-16 was proposed by the Visual Geometry Group (VGG) at the University of Oxford, which more suitable for this assessment. I am fully acknowledge that VGG-16 is actually a CNN architecture but the architecture of VGG-16 is much deeper CNN architecture compared to my initial architecture.

In my intital CNN architecture, the model accuracy only achieve from the range 0.69 % until 1.71 %. My limitation on previous architecture was number of epoch because I am using my computer's CPU. However, even after using the hyperparameter in initial architecture to the VGG16, it produces a higher accuracy.

To balance out the memory usage and training speed, a batch size of 32 is selected because it is optimal based on the the speed of processing. Meanwhile, 600 epochs is enough to get the desirable accuracy of 52.07 %. To allow a better controls in the weights of our network with respect to the loss gradient descent, learning rate at value 0.001 is chosen.

For training loop, the code will print the current epoch and step that are currently being execute so that I am constantly updated with the progress. Then, at the end of the training loop, it will print current epoch and step, along with training loss and train accuracy. This is to ensure that the training accuracy gets progressively increase and training loss gets progressively low.

Meanwhile, in validation loop, the validation loss, validation accuracy and current epoch are printed at the end of the loop. To improve the efficiency of the code, looping for the number of epoch are simultaneously done for both validation and training loop. So, the code have 2 loop in the nested loop.

The code uses Adam optimizer because it combines the advantage of two other extensions to stochastic gradient descent; AdaGrad and RMSProp. It maintains per-parameter learning rates, which are adjusted using estimates of the first and second moments of the gradients. It also includes bias correction to counteract the initial moving averages that are biased towards zero. I adjusted the "step size" = 20 to make the learning rate halved every 20 epochs as "gamma" = 0.5 . Optimizer also adds a penalty to the loss function to prevent overfitting by the size of 0.0001.

IV. CONCLUSION

In conclusion, I have developed and assessed a convolutional neural network (CNN) based on the VGG16 architecture for this image classification assessment on Flowers-102 dataset by University of Oxford.

Overall, I have learned how to make a machine learning for image classification. Despite that, I also learned that model architecture plays a big role to get better test accuracy. The idea of choosing VGG16 as architecture is absolutely a good choice for this assessment. By adding more layers to the architecture, it can improves the accuracy in testing set.

Then, we need to be careful with data augmentation because I used to believe that more data augmentation meaning better accuracy. However, it is totally wrong because too much augmentation makes an unrealistic samples while less augmentation means not enough variability.

Lastly, I realised that saving and loading models in Pytorch is important to continuing the training deploying models and reproducibility of results. I achieved low accuracy in previous model because I don't use save and load after training and validation.

V. FURTHER WORK

Although the Flower-102 dataset has demonstrated promising results, with VGG16 as a architecture model, there are plenty opportunities to explore for additional improvement and research. Since I only discover simple CNN architecture and VGG16, I am open to investigate and research more about advanced neural network. For example AlexNet, Vision Transformers and DenseNet. Thus, it can further my hypothesis that accuracy of the model is also dependent on the architecture model.

REFERENCES