COMP4106 COMPUTER VISION REPORT: IMAGE CLASSIFICATION AND SEGMENTATION USING CNNS ON OXFORD FLOWER DATASET

Rajat Goyal

School of Computer Science, University of Nottingham, Nottingham, NG7 2RD, UK

ABSTRACT

Computer vision can be used in image classification and segmentation. A wide range of possible architectures of Convolutional Neural Networks (CNNs) can be applied. Each model created must be evaluated for this purpose. The best models produced for image classification was Model-3, and the best model for image segmentation was Unet. These were the models submitted for final assessment.

1. INTRODUCTION

Computer vision is an increasingly important field as it deals with visual data and helps the computer extract information from it. These extracted features can be used in various real-life scenarios such as security surveillance using object detection, automation in various fields like self-driving cars, quality control in manufacturing, medical imaging, and biometrics [1].

The earlier stages of computer vision showcased applications of various algorithms like thresholding (Otsu) which uses the intensity of pixels, region growing (KNN), and energy-based optimisation like active contours, graph cuts etc [2]. But since the advancement in deep learning methods like CNNs, computer vision has seen a rapid revolution which has enabled us to perform all the above-mentioned tasks with higher accuracy and efficiency. The earlier methods were primitive in various ways. But CNNs handles those well as they have automatic learnable filters, and their feature map has spatial hierarchy. This allows the model to learn low level features like edges and textures in early stages and learn high complexity features in deeper layers like objects. It also enables translation variance which means the position of the relevant object in the image is no longer relevant and the model still ends up performing well on it. In addition, it is highly generalisable and scalable as it allows higher dimension input and performs good on unseen data [3]. Hence for these various reasons, the CNN is the method of choice in computer vision applications.

In this project, CNN models were created to perform image classification and segmentation. Section 2 outlines the methodology in creating these models, including an explanation of the dataset. Section 3 goes through the CNN architectures created. Section 4 tabulates their results and Section 5 discusses them. Finally in Section 6 the final conclusions are drawn.

2. METHODS

2.1 Dataset information

The dataset provided for this project is based on the Oxford Flower Dataset [4]. It is an ensemble of images retrieved from the internet and self-taken photographs. The images have a range of scale, viewpoint, pose and light variations [5].

2.1.1 Classification Dataset

The classification dataset consists of 17 classes containing 80 images each. The images were chosen such that there are diverse and identical examples spread among the classes. This change in intra-class variability and similarity makes it challenging for model to learn features on it. For example, some classes (e.g., daffodils and windflower) can't be classified on shape alone and others (e.g., dandelion and buttercup) on colour [5]. Thereby this diverse dataset is good for training a model that generalises good and is robust for predictions.

2.1.2 Segmentation Dataset

It is a subset of the dataset mentioned in section 2.1.1 containing 70 images belonging to daffodil flower class. It also contains the groundtruth maps corresponding to each image for the model to evaluate its performance. The groundtruth maps are divided into 5 classes: flowers, background, boundaries, leaves and sky and their corresponding class ids are [1 3 0 2 4]. For this paper, only class flowers and background are taken into consideration.

2.2 Data Pre-processing

2.2.1 Classification Pre-processing

All images were stored in a single folder while the image number is their file name. Images belonging to a particular class were numerically ordered. For example, images 1-80 belonged to class 1, images 81-160 belonged to class 2 and so on. First, an *imageDatastore* object was made which created a reference of all images hence enabling efficient management

of memory and faster runtime. It also uses multithreading and parallelisation making it a better option than running a loop on directory to read images one by one. *Label source* was set to 'none' while creating this object as there are no subfolders whose name can be used as labels. Then command *repelem was used* to repeat an array of values 1-17 for 80 times consecutively on each element and converted into categorical datatype to be used as labels. The images were further resized to a shape of 256×256×3 as per the nature of the task.

This datastore was then divided into three sets: training, validation, and test set. The original *imageDatastore* was split into a ratio of 80:20 where 80 goes to *TrainVal* and 20 goes to test. Then the *TrainVal* object was further split into a ratio of 80:20 for training and validation set simultaneously.

2.2.2 Segmentation Pre-processing

First the images and their maps were split into various folders to be used as train, validation, and test set with the ratio 80:10:10. Notice that the images and their corresponding groundtruth maps were stored in separate folders. An *imageDatastore* object with no labels and a corresponding *pixelLabelDatastore* was created for each set. Further *pixelLabelDatastore objects* were used to store pixel-level annotations. These two objects were combined to form the final *pixelLabelImageDatastore for each set* which were fed to convolutional neural network.

2.2.3 Normalization

The dataset was normalised in three ways. Zero-center, given by the equation $X_{norm} = X - \mu$ where each pixel is subtracted from the mean changing it to zero. Zscore, given by $X_{norm} = \frac{X-\mu}{\sigma}$ where each pixel is subtracted by their mean and divided by standard deviation changing the final mean to 0 and standard deviation to 1. Rescale-zero-one, given by $X_{norm} = \frac{X-\min(X)}{\max(X)-\min(X)}$, transforms the pixel in a range of 0-1 [6].

2.3 CNN

A neural network type called a convolutional neural network, or CNN or ConvNet, is particularly adept at processing input with a grid-like architecture, like an image. Lines, curves, and other basic patterns are detected initially by the layers, followed by more intricate patterns like faces and objects [7].

2.3.1 Convolutional Layer

Convolution between two functions create a third function that expresses how the form of one function is changed by another. Convolution kernels are used by CNN which is a tiny 2D matrix convolved over an image. By performing matrix multiplication and addition on the input picture, a kernel maps on the result, which is of reduced dimensions and hence easier to deal with [7].

2.3.2 Pooling Layer

Similar to how convolutions operate on each local region of an image, pooling is a vector to scalar transformation. However, it computes the average of the pixels in that region (Average Pooling) or just chooses the pixel with the highest intensity and ignores the others (Max Pooling) [8].

2.3.3 Activation Layer

In a neural network, activation functions are used to compute the weighted total of inputs and biases, which is then used to determine whether or not a neuron may be activated [9]. Non-linear activation functions are used in all layers except the final layer as the model would become linear with no depth in layers without it. The final layer could have linear or non-linear activation functions according to the nature of the task: linear for regression and non-linear for classification.

2.3.4 Batch Normalization Layer

CNNs experience internal covariate shift problem at the time of training sometimes. It occurs due to the changes in data distribution as information moves through the network. Batch normalization layer is used to shift the batch's mean back to zero and standard deviation to one [10].

2.3.5 Global Average Pooling Layer

This layer "flattens" the output from the preceding levels into a single vector that may be used as an input in the following fully connected layers.

2.3.6 Fully Connected Layers

A neural network with fully connected layers is one in which each neuron uses a weights matrix to apply a linear transformation to the input vector [11]. As a result, all layer-to-layer relationships are present. In order to produce final predictions, these layers, which are typically placed at the end of the network, learn weights during training.

2.3.7 Dropout Layers

Dropout layers randomly terminates certain activations by setting them to 0 hence turning off those neurons. This helps the model to generalise better and hence tackle overfitting [8].

2.3.8 Classification Layers

A classification layer generates the class label by figuring out the probabilities associated with each class and selecting the one with the highest probability.

2.3.9 Transposed Convolutional Layer

A transposed convolutional layer is generally applied for upsampling. It creates an output feature map with a larger spatial dimension than the input feature map [12].

2.3.10 Loss Functions

Penalty for a poor prediction is called loss. Various loss functions used in CNNs:

- Cross-entropy:

$$L\ (classification)\ =\ -\sum_{k}^{K} y^{(k)} \log \hat{y}^{k}$$

- Dice-coefficient: [13]

$$L\ (segmentation) = 1 - \frac{2\sum_{pixels} y\hat{y}}{\sum_{pixels} y^2 + \sum_{pixels} \hat{y}^2}$$

2.4 Evaluation Metrics

The metrics used to evaluate the image classification models is accuracy and weighted IOU (intersection over union) is used for evaluating image segmentation.

Accuracy	$\frac{TP + TN}{TP + TN + FP + FN}$
Weighted Intersection over union	$\frac{\sum_k Weight_k \frac{Area\ of\ overlap_k}{Area\ of\ union_k}}{Total\ weight}$

Table 1. Evaluation metrics equations

3. ARCHITECTURES

3.1 Image classification

Various architectures were researched and compared. In the self-made architectures, there were minor changes in convolutional layers trying different number of layers and kernels. For the transfer learning models, there architectures were imported using the deep learning toolbox [14]. The hyperparameters used for all models were: adam as the optimiser, max epochs of 500, mini batch size of 128, learning rate of 0.0001, validation frequency differs according to the size of training set, and validation patience of 20.

3.1.1 Classification Architecture

Model	Number of parameters	Number of layers
Model-1	4 mil	34
Model-2	12.5 mil	58
Model-3 (final)	27.9 mil	43
Alexnet	60.9 mil	25
Vgg19	143.6 mil	47
Resnet101	44.6 mil	347
InceptionV3	23.8 mil	315
InceptionResnetV2	55.8 mil	824

Table 2. Various architectures used for classification

An architecture consisting of 43 layers was deployed. It took an image of shape 256×256×3 as an input and downsampled it while learning various feature maps which goes from low-level features to high-level features as the depth of the network increased. Final classification layer was consisting of 17 neurons which outputs a probability corresponding to each class and the class with highest value was selected as the predicted class.

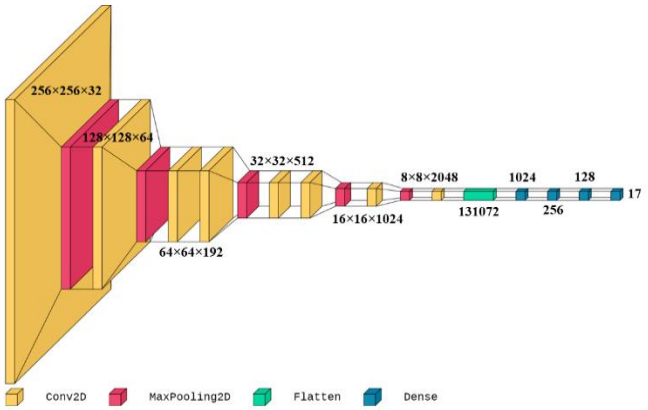

Figure 1. Model-3 architecture for image classification

3.1.2 Segmentation Architecture

Model	Number of parameters	Number of layers
Model-1	985.9K	44
Model-2	312.9K	36
Model-3	99.2K	28
Unet	31 mil	58

Table 3. Various architectures used for segmentation

An architecture consisting of 58 layers called Unet was deployed. It can be divided into two components: downsampling and upsampling. The downsampling was done for the purpose of feature learning and upsampling was done as the desirable output is supposed to be the same resolution as the input image while classifying its pixels hence of the depth 2 as there were 2 classes (foreground and background). The Unet is named after the U-shape formed when the downsampling and upsampling layers are connected to each other [15]. The last layer should have balanced class weights as generally in segmentation task the classes are highly imbalanced between foreground and background which results in wrong prediction and the model might get biased towards the class with more pixels. Hence, the classes were balanced with their weights before classifying the pixels in final layer.

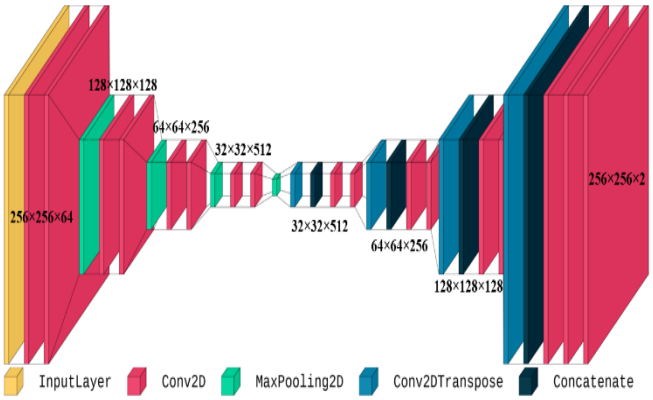

Figure 2. Unet architecture for image segmentation

4. Results

Various models and architectures were researched and compared for both classification and segmentation tasks which included self-made and pre-trained imported architectures.

4.1 Classification

Models	Validation Accuracy	Test Accuracy
Model-1	76.47%	75.37%
Model-2	76.92%	77.21%
Model-3	84.16%	83.09%
Alexnet	43.44%	46.32%
Vgg19	47.06%	50.74%
Resnet101	45.70%	39.71%
InceptionV3	46.15%	49.63%
InceptionResnetV2	41.18%	41.18%

Table 3. Accuracies returned by classification architectures

As **Model-3** returned the best evaluation metrics performing good on both validation and test sets meaning the model works just fine even on the unseen data, it was selected as the final model for the classification task. As the classes were balanced (each class containing 80 images) accuracy is a good measure to compare the performance.

4.2 Segmentation

Models	Validation Accuracy	Mean Accuracy (Test)	Weighted IOU
Model-1	98.55%	97.19	92.11
Model-2	99.79%	98.08%	94.68
Model-3	99.46%	97.19%	92.11
Unet	99.56%	98.45%	97.40

Table 4. Accuracies returned by segmentation architectures

Unet returned the best evaluation measures with the highest weighted IOU score on the test set and hence was selected as final model for segmentation task.

5. DISCUSSION

The *rescale-zero-one* normalization ended up working best on the classification task whereas the default *zerocenter* normalization performed best on segmentation task. This conclusion was made by comparing the evaluation metrics returned on test set for each normalization.

Data augmentation was experimented with, and it was found out that the model is performing better without data augmentation. Model-3 ended up returning 71.49% validation accuracy and 73.17% test accuracy on augmented image datastore. This was a surprising finding as data augmentation is expected to increase the model generalisation and make it more robust by introducing various scenarios in the training dataset, so it doesn't overfit. But in this case, the images were carefully selected for the dataset according to the scientific criterion like focus, field of view, composition, etc. Hence the augmented dataset might have worked better on a test set with noise or poor quality of images. But the dataset used for this project had good quality images with flowers usually being in the centre of the frame hence training the model on images that are shifted or zoomed might make the performance worse on the chosen test set.

It was also found that transfer learning models performed poorly contradicting to the common belief that they perform good in almost every scenario. But this shows that if pretrained weights are used it will not necessarily fit in a new dataset especially if the dataset size is small resulting in underfitting. This statement is backed by the ~40-50% accuracy achieved by almost all transfer learning models tested.

The class frequencies turned out to be 32:68 for flower and background simultaneously. Hence the decision to use weighted classes in the final model was made. But since Unet uses unweighted classes by default, the model was broken down into layers which were stored as a graph. Then the final layer was replaced with pixel classification layer having weighted classes passed as a parameter. But removal and replacement of layers resulted in decoder detaching from the previous layers. To solve this problem, the layers were manually connected in the graph by their names passed as parameters with the help of *analyzeNetwork.* Unet returned the best weighted IOU score of 97.40 i.e., the class-wise weighted some of the pixels that overlapped in the segmented image and groundtruth maps divided by the total pixels.

Even though Unet is the heaviest model comparatively, the dataset it was trained on is small (train set contained 57 images) hence it trained quick. But in real scenarios there could be a need to deal with large dataset which will take longer time to train. It can be noticed that Model-2 was quite close to Unet in terms of performance (weighted IOU of 94.68) while decreasing the numbers of parameters by almost 90%. Hence Model-2 can also be selected in real-life cases due to faster training time given it is capable of replicating the same performance and display generalisation on the new dataset as well.

6. CONCLUSION

The best model for image segmentation was Model-3, with a final classification accuracy of **83.09%,** and the best model for image segmentation was Unet, which had weighted IOU of **97.40**. These models are submitted as the final submission according to the task as they returned the best evaluation metrics out of all models researched. Both models ended up performing good, but their reliability is yet to be tested in future on larger dataset as the dataset used for the research was quite small and cleaned up for the task. Another idea is implementation of other types of convolutions. For example, mobilenet which uses depthwise and pointwise hence significantly decreasing the number of parameters by kernel trick and still be able to perform well.

7. REFERENCES

[1] Li, Y. and Zhang, Y. (2020). *Application Research of Computer Vision Technology in Automation*. [online] IEEE Xplore. doi:https://doi.org/10.1109/CIBDA50819.2020.00090.

[2] Andrew French, COMP4106 Computer Vision Lecture Slides, University of Nottingham, 2023

[3] O' Mahony, N. et al. (2019). *Deep Learning vs. Traditional Computer Vision*. [online] Available at: https://arxiv.org/ftp/arxiv/papers/1910/1910.13796.pdf [Accessed 7 Apr. 2023].

[4] Nilsback, M. and Zisserman, A. (*no date*). *Visual Geometry Group - University of Oxford*. [online] www.robots.ox.ac.uk. Available at: https://www.robots.ox.ac.uk/~vgg/data/flowers/17/index.html [Accessed 3 Apr. 2023].

[5] Nilsback, M.-E. and Zisserman, A. (2006). *A Visual Vocabulary for Flower Classification*. [online] Available at: https://www.robots.ox.ac.uk/~vgg/publications/2006/Nilsback06/nilsback06.pdf [Accessed 5 Apr. 2023].

[6] MathWorks (*no date*). *Image input layer - MATLAB - MathWorks United Kingdom*. [online] uk.mathworks.com. Available at: https://uk.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.imageinputlayer.html#d123e66056 [Accessed 15 Apr. 2023].

[7] Lecun, Y. et al. (1998). Gradient-Based Learning Applied to Document Recognition. *PROC. OF THE IEEE*, [online] 86(11). Available at: http://vision.stanford.edu/cs598_spring07/papers/Lecun98.pdf [Accessed 21 Apr. 2023].

[8] Olafenwa, J. (2018). *Components of convolutional neural networks*. [online] Medium. Available at: https://towardsdatascience.com/components-of-convolutional-neural-networks-6ff66296b456 [Accessed 25 Apr. 2023].

[9] Panneerselvam, L. (2021). *Activation Functions | What are Activation Functions*. [online] Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2021/04/activation-functions-and-their-derivatives-a-quick-complete-guide/ [Accessed 26 Apr. 2023].

[10] Ioffe, S. (2015). *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. [online] Available at: https://arxiv.org/pdf/1502.03167.pdf [Accessed 25 Apr. 2023].

[11] Unzueta, D. (2022). *Fully Connected Layer vs Convolutional Layer: Explained | Built In*. [online] builtin.com. Available at: https://builtin.com/machine-learning/fully-connected-layer [Accessed 26 Apr. 2023].

[12] Anwar, A. (2021). *What is Transposed Convolutional Layer?* [online] Medium. Available at: https://towardsdatascience.com/what-is-transposed-convolutional-layer-40e5e6e31c11#:~:text=The%20transposed%20Convolutional%20Layer%20is [Accessed 29 Apr. 2023].

[13] Xin Chen, COMP3009 Machine Learning Lecture Slides, University of Nottingham, 2022

[14] MathWorks (*no date*). *Deep Learning Toolbox Documentation - MathWorks United Kingdom*. [online] uk.mathworks.com. Available at: https://uk.mathworks.com/help/deeplearning/ [Accessed 28 Apr. 2023].

[15] Ronneberger, O., Fischer, P. and Brox, T. (2015). *U-Net: Convolutional Networks for Biomedical Image Segmentation*. [online] Available at: https://arxiv.org/pdf/1505.04597.pdf [Accessed 11 May 2023].