School of Computer Science and Applied Mathematics
UNIVERSITY OF THE WITWATERSRAND, JOHANNESBURG

COMS4036A Computer Vision Project
Snake Species Identification - Report

Zubair Ahmed Bulbulia, 1249593,
Neil Fabiao, 2216748,
Computer Science Honours

20 November 2019

1 Introduction

The field of computer vision largely involves extracting useful information from images and video. The field involves multiple disciplines, and approaches have been used to solve many challenging problems such as facial and gesture recognition, object detection, activity classification and many other vision-based problems. Classification problems are one such key area of computer vision problems. In this project, we make use of vision algorithms to attempt to classify various species of snakes, which can potentially be used to identify a species of snake from a given image. This could be useful in potentially determining whether a snake is harmless or venomous.

2 Technical Problem Details

As previously stated, the goal of this project is to identify different types of snake species, some of which are harmless and others venomous. The problem of accurate snake species classification is an important because snakebite is a neglected disease that affects large amounts of people the world over, especially in poor/rural communities. Accurately classifying snakes is of utmost importance so that in the event of snakebite, the correct anti-venoms can be administered or treatment can be planned and lives can be saved.

Manually performing identification can prove challenging due to high species diversity, as well as other factors such as incomplete or misleading information provided by snakebite victims, in addition to the lack of knowledge or resources in herpetology. The use of computer vision approaches to accurately solve this identification problem could potentially save many lives.

The provided dataset has 82,601 RGB images of snakes, consisting of 45 snake species to be identified. Some challenges relating to the dataset and the nature of the problem itself that were encountered were as follows :

- Class imbalance - some classes of snake species contain a large number of images and others have relatively few, as shown in Fig. 1 below.
- Visual discrepancy in images - images are of various resolution,
  images may show the snake clearly or may have grass or water etc. which conceal the snake, size of the snake in the image and where it is located, etc.
- Some species have patterns that vary depending on their age and location
- Two species might look very similar, with one being venomous and the other not

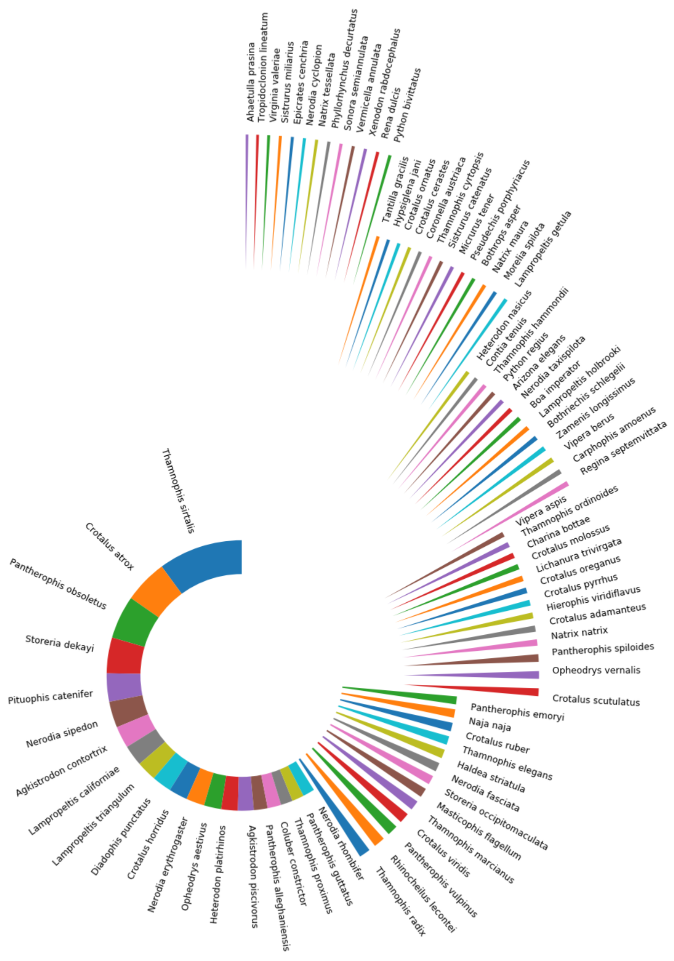

FIGURE 1: Image depicting class imabalance

After the identification of the various problems found relating to the given data, the proposed solution to solve described problem above were developed and follows in the next section.

3 Methodology

Traditional computer vision methods are used to extract features from images and can be used to classify images based on these features. An advantage of this approach is that hand-crafted features can be specified as well as shape and contour models, which can fit specific problem requirements well. However, manually extracting features is an involved process and may require a lot of fine-tuning/tweaking.
Deep learning approaches, on the other hand, have shown to be very successful at classification tasks. They have outperformed traditional vision-based methods given enough labelled data for various problems through the use of convolutional neural networks(CNNs) and have high accuracy. An advantage of this approach is that input data pre-processing and feature extraction is not necessary, as deep networks learn the features they need for classification on their own. For this reason, CNN's were chosen as the approach to be used for this project.

There are many CNN architectures available that have different things to offer. These include the number of layers (how deep the network is) as well as the type of layers used, amongst others. The network architectures used and some detail on each of the models are detailed as follows:

- ResNet-50 - The ResNet-50 is a 50 layer residual network and the model consists of 5 stages each with a convolution and Identity block. Where each of the convolution block has 3 convolution layers and each identity block also has 3 convolution layers. It has a total of 25.6 million trainable parameters and it skips connections to allow for faster convergence.
- VGG11 with BN - VGG is a deep network that has 8 convolutional layers and 3 connected layers. It uses 3x3 convolutions, but has a large number of filters, resulting in a total of 123.6 million parameters. The model uses Rectified Linear Units (ReLUs) as activation functions, with batch normalisation. Batch normalisation allows a layer to learn a little bit more independently of other layers, which allows for better generalisation.
- ResNeXt-50 - is a deep neural network with 50 layers. The model is based on resnet and borrows from the inception architecture. It is a parallelised implementation in which a resnet block is repeated and aggregated. The version used has approximately 27.56 million parameters.

- DenseNet-121 - A densely connected network with 121 layers. It is state-of-the-art CNN where each layer obtains additional inputs from all previous layer and passes on its own feature-maps to all of the following layers. Concatenation is used such that each layer receives shared information from preceding layers. It has 7.97 million parameters. In addition to this, dropout was used to try and prevent over-fitting [3].
- ShuffleNetv2 - is a very efficient neural network that employs a “channel shuffle” or group convolution, that obtains the input data from different groups of layers. The input and output channels are fully related and improve the information communication between different groups of channels as well as it’s accuracy. With a total of 4.2 million trainable parameters, it has 1x1 convolution layer and is a light-weight network [4].

Fully connected layers (dense layers) have neurons that might become dependent on each other. This can lead to overfitting when training a model as explained by [1]. To help prevent this as well as to reduce redundancy in classification we make use of drop out layers.
Dropout forces a neural network to learn the features that are most useful in conjunction with many different random subsets of the other neurons.
Dropout can also change the number of iterations required for convergence and reduce training time for each epoch. Dropout layers were used for DenseNet.

The following section describes the experimental setup used to train the various models used.

3.1 Experiment setup

For all of the aforementioned models that were used, the training data was split using an 80/20 split for training and validation. This allows us to see if the model is learning correctly or is overfitting to the training data. A batch size of 16 was chosen for training, as after some experimentation it was observed to provide better accuracy in general than a batch size of 32. The input size was kept at a size of 224x224 for all the models used.

Initially, training was done using the regular RGB images with both the DenseNet and ResNet models. After observing the results, it was decided to try and improve them using more suitable models and trying other models, and augmenting the input data. This involves normalizing the data as well as implementing augmentations such as performing resizing/cropping, rotating the image by a random amount between 0 and 90 degrees, and flipping the image on its axis. These input augmentations can be seen as in Figure 2 below.

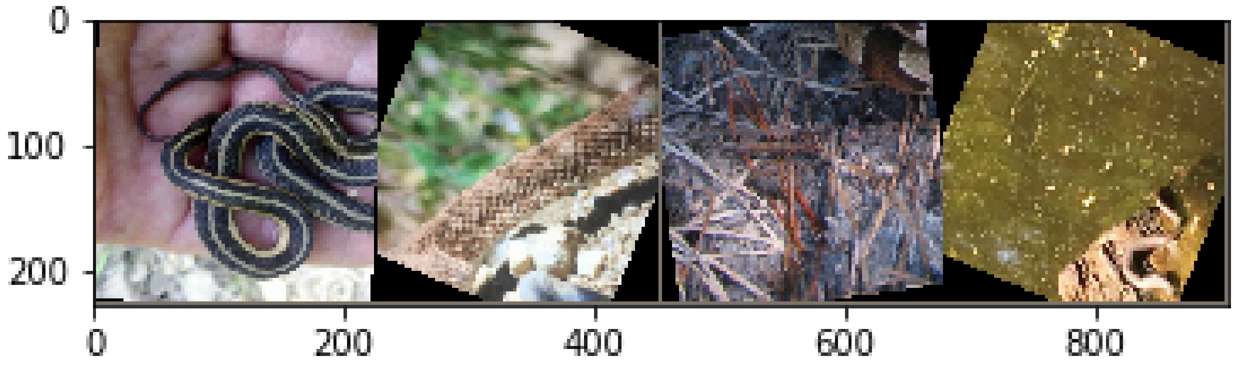

FIGURE 2: 4 of the inputs after augmentation

DenseNet was then re-trained with augmented input data. VGG-11 and ResNeXt-50 were then trained as well. ResNeXt made use of augmented data while VGG did not. These took a long time to train and while they were training, a ShuffleNet was trained as well using augmented input images.

Each model was trained for a various number of epochs. Instead of training from scratch, each model also used pre-trained weights. These weights were trained on the ImageNet dataset and are provided by PyTorch[7]. The results obtained for each model is presented in the subsequent section.

3.2 Implementation/Results

The first models we used were DenseNet-121(DN) and ResNet-50(RN). The former is a fairly new, state-of-the-art CNN and has shown to provide comparable or better results than RN on the ImageNet dataset. DN has dropout layers to provide for better generalisation and also has 17.6 million parameters less than RN, but RN has a lower Top-1 error(%) than DN [6].
After training it was found that ResNet-50 does indeed provides better results than DN across all metrics, with a higher accuracy, F-Score and Cohen's Kappa as well as a lower loss for both training and validation. Based on [6](which shows that RN has a lower Top-1 error than DN), this is an expected result.

ResNet-50 shows it's best results at 6 epochs, and at 10 epochs all the training metrics had improved, but validation metrics had worsened. This is likely due to overfitting occurring after the 6th epoch.

ShuffleNetv2 produces the worst results. This is also an expected case as it is a mobile neural network and has the least trainable parameters. It may have better performance if trained for a longer duration. Although some of the results are poor, when considering that a random choice has an accuracy of 2.22%, all of these models' results are promising baselines for measuring performance.
VGG-11 and ResNeXt-50 both give fairly decent results, as can be seen in Figure 3 and Figure 4 below.

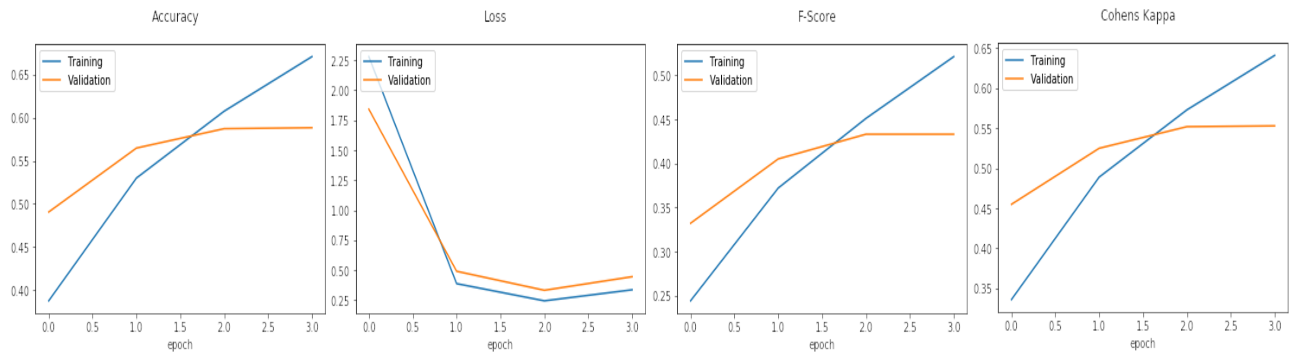

FIGURE 3: Training & Validation Accuracy, Loss, F-Score and Cohen's Kappa for VGG-11

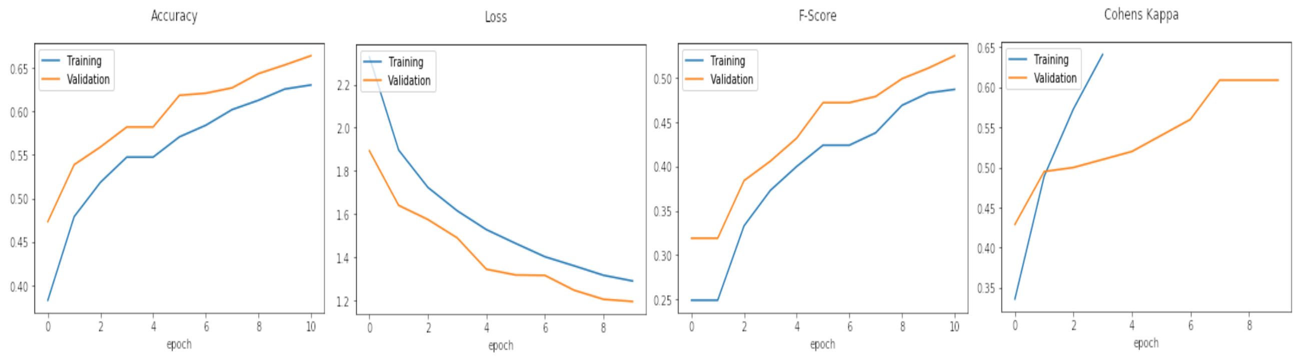

FIGURE 4: Training & Validation Accuracy, Loss, F-Score and Cohen's Kappa for ResNeXt-50

ResNeXt has the best validation metrics - as it has a Cohens Kappa of 0.609 and an accuracy of 64% - followed by VGG. This means that the model generalises fairly well. VGG has better training metrics than ResNeXt, however, the regular ResNet model(trained without augmented data) still has the best training metrics [6].

Table 1 below shows some of the key results that were obtained.

Model	Ep #	Tra Acc	Tra Loss	Tra F-Scr	Tra Cohen's K	Val. Acc	Val. Loss	Val. F-Scr	Val Cohen's K
ResNet 50	6	68.84	1.0136	0.542	0.66	56.40	1.5902	0.411	0.527
	10	83.96	0.499	0.741	0.824	55.34	1.956	0.401	0.516
VGG 11	4	67.08	1.1083	0.521	0.641	58.83	1.4767	0.433	0.553
ResNeXt 50	8	61.26	1.3607	0.469	0.578	64.32	1.2477	0.499	0.609
DenseNet 121	11	45.84	1.7563	0.319	0.420	49.45	1.5666	0.347	0.454
ShuffleNetv2	13	42.19	1.8679	0.288	0.382	44.62	1.7803	0.307	0.404

TABLE 1: Table of key training and validation results

ResNet50 followed by VGG has the best training F-Scores and ResNeXt50 has the best validation F-Score. This metric is a weighted average of the precision and recall. This is more important than the regular Top-1 accuracy because of the class imbalance present in the data.
The aforementioned models also get Cohen's Kappa values ranging from 0.5 to greater than 0.6. This is important as it is a score for inter-rater agreement. In practice, it describes how well the models perform above a baseline of just using the most commonly occurring classes for labels/predictions.

The following section details possible improvements that could be made to further improve on the obtained results.

3.3 Considerations/future work

Based on the models used and the results obtained, ResNet, ResNeXt and VGG seem promising. Better results could probably be achieved by training deeper models with these architectures, and training for a longer amount of time (more epochs or until convergence if possible). Models that are deeper have more parameters to train but should have a lower loss and extract more accurate features.
Also, exploring alternate means of augmenting and preprocessing the data may prove beneficial, such as performing scaling or shearing and forward cropping. Additionally, incorporating hand-crafted features or using multiple models to deal with intra-class variance could be a good option.
Lastly, exploring more state-of-the-art networks such as EfficientNet and DenseNet may provide better results.

4 Conclusion

To summarise, various models were created and trained for the task of performing snake species classification. The results showed that out of the various models trained, the ResNeXt50, VGG-11 and ResNet50 models all performed fairly well, having been trained for a limited number of epochs only. Performance metrics were discussed, and considerations for improvement were given. Better performance could be achieved with deeper networks or more state-of-the-art models could be explored.

References

[1] Srivastava, Nitish, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. “Dropout: A Simple Way to Prevent Neural Networks from Overfitting.” J. Mach. Learn. Res. 15 (1). JMLR.org: 1929–58. http://dl.acm.org/citation.cfm?id=2627435.2670313.

[2] Xie, Saining et al. “Aggregated Residual Transformations for Deep Neural Networks.” 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016): 5987-5995.

[3] Huang, Gao et al. “Densely Connected Convolutional Networks.” 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016): 2261-2269.

[4] Ma, Ningning et al. “ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design.” ArXiv abs/1807.11164 (2018)

[5] Budhiraja,Amar. (2016, Dec 15).*Dropout in (Deep) Machine learning* Retrieved from : https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5

[6] Tsang,Sik-Ho (2018, Nov 25).Review: DenseNet — Dense Convolutional Network (Image Classification) Retrieved from: https://towardsdatascience.com/review-densenet-image-classification-b6631a8ef803

[7] Pytorch (2019) Retrieved from: https://pytorch.org/docs/stable/torchvision/models.html