Automated Snake Classification with Deep Learning

Rishi Khare (RA2111047010027)
Student
Department of CINTEL
SRM Institute of Science and Technology
Kattankulathur, Chennai 603203

Arnav Sud (RA2111047010038)
Student
Department of CINTEL
SRM Institute of Science and Technology
Kattankulathur, Chennai 603203

Jyotirmay Jaswal (RA2111047010034)
Student
Department of CINTEL
SRM Institute of Science and Technology
Kattankulathur, Chennai 603203

Mayur Pal (RA2111047010033)
Student
Department of CINTEL
SRM Institute of Science and Technology
Kattankulathur, Chennai 603203

Aditya Vasudevan (RA2111047010006)
Student
Department of CINTEL
SRM Institute of Science and Technology
Kattankulathur, Chennai 603203

Abstract:

Automatic snake classification is the process of identifying snake species using image processing techniques. This system is helpful in reducing the death by snake bites and to suggest appropriate anti venom for the victim in a short span of time. The previous works have built systems with relatively smaller databases using ML and older deep architectures. The systems were capable of identifying only a few snake species and/or had lower accuracies. However, the Snake classification can further be improved in order to make the system more robust. The proposed system is capable of identifying about 770 classes of snake species and is built with a relatively larger dataset using newer deep learning architecture ResNeXt50-V2. An ensembled model is used to further improve the system to achieve an accuracy of 86% and F1-score of 0.65.

1. Introduction:

The occurrence of fatalities and limb amputations due to snakebites represents a significant challenge for healthcare institutions. Annually, there are approximately 2 to 3 million cases of envenomation, with 450,000 to 600,000 of these cases requiring treatment due to the potential for enduring disability and disfigurement. However, identifying the specific snake responsible for the bite is a complex task for several reasons:

i)Snake species exhibit a remarkable diversity in regions prone to snakebites, as exemplified by India, home to around 300 distinct snake species.

ii)Local communities and healthcare providers often possess limited knowledge of herpetology, making it challenging to accurately identify snakes involved in snakebite incidents.
iii)Inadequate understanding of the epidemiological significance of various snake species further compounds the issue.
Consequently, individuals bitten by snakes often face delays in receiving prompt treatment because they struggle to identify the biting snake or provide essential information about the incident. This uncertainty places medical practitioners in the difficult position of making educated guesses regarding the snake's identity, potentially leading to further complications and, regrettably, patient fatalities. Such uncertainties contribute to a decreased survival rate among snakebite patients.
Therefore, the ability to identify snakes through low-resolution images has the potential to significantly enhance patient survival rates. Additionally, this system could play a role in wildlife conservation by aiding in the identification of endangered snake species.
The automation of snake species identification, using computer vision techniques, could have a positive impact on public health. To date, only a limited number of computer vision and machine learning algorithms specifically designed for snake identification have been developed. The most successful model available as of 2022 achieved an F1-score of 0.656 using a pre-trained Vanilla ResNet50-V2 architecture. However, these models have yet to be deployed in real-world situations where lives are at stake. Thus, the creation of an automated and robust system for snake species identification represents a crucial objective, addressing issues related to biodiversity, conservation, and global health.

2. Methodology:

In this section, we will dive into the dataset and foundational methodology employed for the execution of the task.

2.1. Datasets Used

The primary source of data for this project was online biodiversity platforms, particularly iNaturalist and HerpMapper. Additional data was collected by scraping images from Flickr and gathering images from private collections and museums. The resulting dataset exhibits a pronounced long-tailed class distribution, with the most common species (Thamnophis Sirtalis) represented by a substantial 22,163 images, while the least common species (Achalinus Formosanus) had only 10 images. The final dataset was divided into two distinct sets: a training set containing 347,405 images and a validation set with 38,601 images. Both sets maintain the same class distribution, with each class having at least one image in the validation set. Additionally, a testing set was created, comprising 26,531 images encompassing all 772 classes, maintaining a similar class distribution. These images are accompanied by metadata providing information about the continent and country where each image was captured. In some cases, the

metadata simply contains "UNKNOWN" for snake depictions where location information is unavailable.

2.2. Network Architecture

While a baseline Convolutional Neural Network (CNN) can be augmented to tackle more complex problems and potentially yield improved performance, it has been observed that simply adding more layers to the network can lead to performance degradation. This decline in performance may be attributed to issues related to optimization functions, network initialization, and, more notably, the vanishing gradient problem. To address this challenge, the ResNet architecture was introduced. At the heart of ResNet is the concept of a residual network block.
This block incorporates a skip connection, which enables the network to skip certain layers in the neural network and feed the output of one layer directly to a subsequent layer, not necessarily the immediately adjacent one. This skip connection establishes an alternative pathway for the flow of gradients, effectively mitigating the vanishing gradient problem.

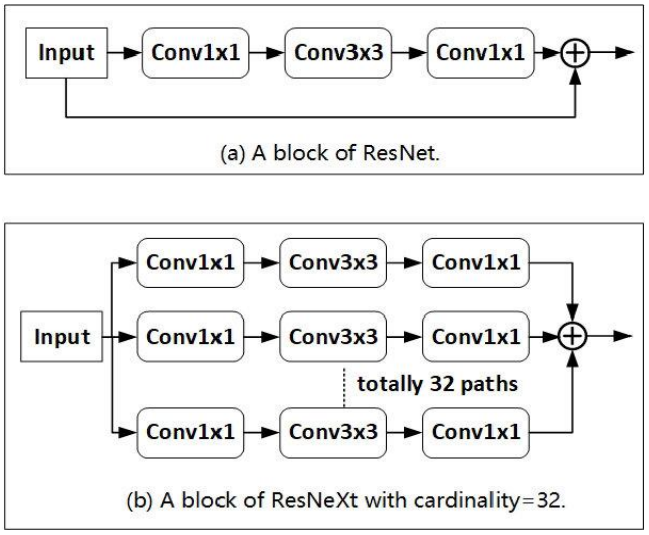


Fig.1.Network Blocks of ResNet and ResNeXt

To enhance the efficacy of the ResNet architecture without inflating the number of parameters, the ResNeXt architecture was introduced. In the ResNeXt framework, each residual network block, as depicted in Figure (a), is divided into a variable number of paths, where the number of paths within the ResNeXt block is referred to as its "cardinality." In Figure (b), the cardinality is set to 32. It's noteworthy that all these paths share the same network structure.
Both the ResNet and ResNeXt network blocks exhibit distinct widths. In ResNet, layer-1 comprises a single convolution layer with a width of 64, whereas in ResNeXt, layer-1 incorporates 32 separate convolution layers, each with a width of 4 (32*4 width). As the cardinality is increased, the validation error tends to decrease, resulting in an overall enhancement in the network's performance.

3. Implementation

3.1. Data Preprocessing

Table 1

Subset	# of images	% of data	min. # of images/class
Training	245,185	85.24%	17
Validation	14,029	4.88%	2
Testing	28,418	9.88%	1
Total	287,632	100	20

In our implementation, the training data, as indicated in Table 1, is used to train the model. During training, the validation data is employed to test the model at the end of each epoch, enabling the fine-tuning of weight parameters. The test data, on the other hand, is used to evaluate the final model after the entire training process.
The images supplied by the SnakeClef2021 dataset exhibit varying sizes. To standardize the training images, they are initially resized to dimensions of 224×224×3 using a random crop-resize operation. These images also undergo horizontal and vertical flips with a probability of 0.5. Additionally, there is a 0.5 probability of applying scaling and rotation to the images. Further, all images are normalized, with a standard deviation of 0.23 and a mean of 0.5.
Similarly, for the validation and testing datasets, their images are resized to 224×224×3 dimensions using a resizing function, and they are normalized with a standard deviation of 0.23 and a mean of 0.5, respectively. Both training and testing images undergo augmentation using the Albumentations Python library. The augmentation methods applied include RandomResizedCrop, Transpose, Resize, HorizontalFlip, VerticalFlip, ShiftScaleRotate, and Normalize, enhancing the robustness of the dataset.

3.2. Accumulation

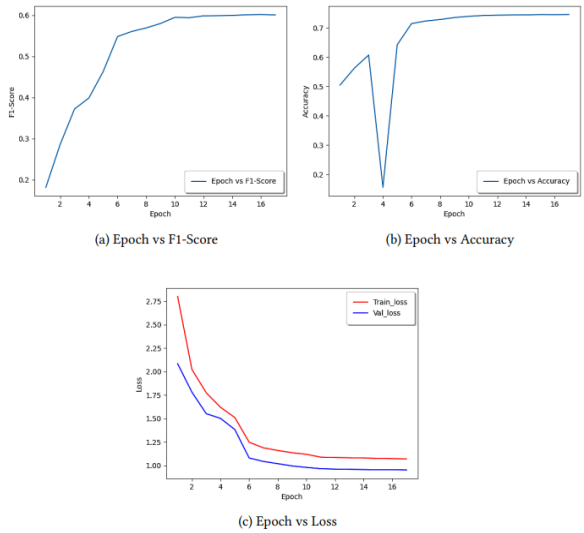
Table 2

ResNeXt50-V2 Architecture

Conv1	L1 - [224, 7 x 7, 64]		
Conv2	**3 Stacked Residual Blocks**		
	L1- [64, 1 × 1, 128] L2- [128, 3 × 3, 128] L3- [128, 1 × 1, 256]	L1- [256, 1 × 1, 128] L2- [128, 3 × 3, 128] L3- [128, 1 × 1, 256]	L1- [256, 1 × 1, 128] L2- [128, 3 × 3, 128] L3- [128, 1 × 1, 256]
Conv3	**4 Stacked Residual Blocks**		
	L1- [256, 1 × 1, 256] L2- [256, 3 × 3, 256] L3- [256, 1 × 1, 512] Downsampling L0 -[256, 1 × 1, 512]	L1- [512, 1 × 1, 256] L2- [256, 3 × 3, 256] L3- [256, 1 × 1, 512]	
	L1- [512, 1 × 1, 256] L2- [256, 3 × 3, 256] L3- [256, 1 × 1, 512]	L1- [512, 1 × 1, 256] L2- [256, 3 × 3, 256] L3- [256, 1 × 1, 512]	
Conv4	**6 Stacked Residual Blocks**		
	L1- [512, 1 × 1, 512] L2- [512, 3 × 3, 512] L3- [512, 1 × 1, 1024] Downsampling L0 -[512, 1 × 1, 1024]	L1- [1024, 1 × 1, 512] L2- [512, 3 × 3, 512] L3- [512, 1 × 1, 1024]	L1- [1024, 1 × 1, 512] L2- [512, 3 × 3, 512] L3- [512, 1 × 1, 1024]
	L1- [1024, 1 × 1, 512] L2- [512, 3 × 3, 512] L3- [512, 1 × 1, 1024]	L1- [1024, 1 × 1, 512] L2- [512, 3 × 3, 512] L3- [512, 1 × 1, 1024]	L1- [1024, 1 × 1, 512] L2- [512, 3 × 3, 512] L3- [512, 1 × 1, 1024]
Conv5	**3 Stacked Residual Blocks**		
	L1- [1024, 1 × 1, 1024] L2- [1024, 3 × 3, 1024] L3- [1024, 1 × 1, 1024] Downsampling L0 -[1024, 1 × 1, 2048]	L1- [2048, 1 × 1, 1024] L2- [1024, 3 × 3, 1024] L3- [1024, 1 × 1, 2048]	L1- [2048, 1 × 1, 1024] L2- [1024, 3 × 3, 1024] L3- [1024, 1 × 1, 2048]
Softmax	L1 – [2048, - , 772]		

The training dataset is organized into 3860 batches, with each batch containing 5 folds or mini-batches. To ensure efficient training and mitigate the introduction of noisy gradients, it is imperative that each mini-batch encompasses representatives from all classes. For this purpose, the images in each mini-batch have been thoughtfully selected to achieve this inclusivity.
To maintain a balanced class distribution in each data split and align it with the distribution in the complete training dataset, a stratified k-fold cross-validation strategy with k = 5 has been employed.
To dynamically control the learning rate throughout training, a cosine annealing with warm restarts scheduler has been

implemented. The initial learning rate is set at 1e-4, and the weight matrices of the architecture are fine-tuned using the Adam optimizer. This combination of techniques optimizes the training process and aids in the model's convergence.



(a) Epoch vs F1-Score (b) Epoch vs Accuracy

(c) Epoch vs Loss

Fig.2. Performance Measures

3.3. Constructing the Model

The ResNext model's architecture comprises a Conv1 layer with 64 filters, each of size 7 × 7. The 64 feature maps produced by Conv1 serve as the input for three stacked Conv2 blocks. The output of Conv2 consists of 256 feature maps, which in turn serve as input for four stacked Conv3 blocks. The output of Conv3 generates 512 feature maps, which are used as input for six stacked Conv4 blocks. The output of Conv4 produces 1024 feature maps, serving as input for three stacked Conv5 blocks. The final output of Conv5 results in 2048 feature maps, which are then passed through a softmax layer, producing 772 classes.

The model summary for the implemented ResNeXt50-V2 is presented in Table 2. During training, the following hyperparameters were utilized:

Input image size: 224 × 224 × 3
Image augmentations applied during training: RandomResizedCrop, Transpose, Resize, HorizontalFlip, VerticalFlip, ShiftScaleRotate, and Normalize methods from the Albumentations Python library.
Training duration: 17 epochs
Loss function: CrossEntropyLoss with default parameters
Learning rate: 1e-4
Optimizer: Adam optimizer

4. Results and Analysis

This section delves into the examination of both the training and testing procedures, evaluating them through a range of performance metrics.

4.1. Training Analysis

The training process was subject to a thorough analysis to assess the model's stability and its susceptibility to overfitting.
Throughout the training process, F1-Score and training accuracy were monitored for every epoch and visualized in Figure 2a and 2b, respectively. It's evident from Figure 2a that the F1-Score steadily increases from the very first epoch and stabilizes at approximately 0.6 by the 14th epoch. In Figure 2b, the training accuracy reaches a stable state around the 11th epoch.

Simultaneously, training and validation losses were computed for each epoch and depicted in Figure 2c. The graph clearly illustrates that the validation loss for every epoch is consistently lower compared to the training loss. This observation suggests that the model generated is resilient against overfitting, as it generalizes well to unseen data.

4.2. Testing Analysis

Table 3

Performance Measures

ResNeXt50-V2 Models	F1-Country Score	F1-Score	Accuracy
Model-13	0.42	0.38	0.66
Model-14	0.51	0.40	0.69
Model-15	0.44	0.40	0.68
Model-16	0.40	0.38	0.68
Model-17	0.37	0.37	0.68
Ensembled Model	0.67	0.68	0.86

The training process was subject to a thorough analysis to assess the model's stability and its susceptibility to overfitting.

During the testing phase, rather than relying solely on the most recent model for predictions, we leveraged an ensemble model. To assemble this ensemble, we selected the models corresponding to the last 5 epochs (models 13, 14, 15, 16, and 17). These models were chosen as the top performers because their accuracy and F1-Score had stabilized during their respective epochs.

The ensemble process involved generating predictions from each individual model and then averaging these predictions. Experiments were conducted to generate predictions from both individual and ensemble models, as illustrated in Table 3. It's important to note that models 13, 14, 15, 16, and 17 produce different predictions for each image, reflecting the differences in their weight matrices. Each model may have specialized in making predictions for distinct sets of images.

In the ensemble model, the average of the predictions generated by all considered models is taken. This approach significantly enhances the accuracy of the model, as indicated in the table. The ensemble model achieved an accuracy of 85.54% and an F1-Score of 0.738, demonstrating its superior performance.

5. Conclusion

In summary, snake species identification is a complex endeavor, mainly due to the vast diversity of snake species and the presence of a dataset with a skewed class distribution. We have developed an automatic classification system utilizing the ResNeXt50-V2 architecture, which has yielded an F1-score of 0.738.

Looking ahead, there are avenues for further improvement in the system's accuracy. This includes enhancing image preprocessing techniques to refine the input data. Additionally, exploring and implementing alternative deep learning architectures may yield even better accuracy results, thereby advancing the performance of the model in snake species identification.

References

[1] A. Patel, L. Cheung, N. Khatod, I. Matijosaitiene, A. Arteaga, J. W. Gilkey, Revealing the unknown: real-time recognition of Galápagos snake species using deep learning, Animals 10 (2020) 806.

[2] L. Picek, R. Ruiz De Castaneda, A. M. Durso, P. Sharada, Overview of the SnakeClef 2020: Automatic snake species identification challenge in: Working Notes of CLEF 2020 - Conference and Labs of the Evaluation Forum, 2020.

[3] L. Picek, A. M. Durso, R. Ruiz De Castañeda, I. Bolon, Overview of SnakeClef 2021: Automatic snake species identification with country-level focus in: Working Notes of CLEF 2021 -Conference and Labs of the Evaluation Forum, 2021.

[4] A. Gulli, S. Pal, Deep learning with Keras, Packt Publishing Ltd, 2017.

[5] D. Kingma, J. Ba, Adam: A method for stochastic optimization in: International Conference on Learning Representations, 2014.

[6] A. James, Snake classification from images, PeerJ Preprints 5 (2017).

[7] M. G. Krishnan, Impact of pretrained networks for snake species classification in: Working Notes of CLEF 2020 - Conference and Labs of the Evaluation Forum, 2020.

[8] A. P. James, B. Mathews, S. Sugathan, D. K. Raveendran, Discriminative histogram taxonomy features for snake species identification, Human-Centric Computing and Information Sciences 4 (2014).

[9] C. Szegedy, S. Ioffe, V. Vanhoucke, A. Alemi, Inception-v4, inception-resnet and the impact of residual connections on learning in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 31, 2017.

[10] A. Buslaev, V. I. Iglovikov, E. Khvedchenya, A. Parinov, M. Druzhinin, A. A. Kalinin, Albumentations: fast and flexible image augmentations, Information 11 (2020) 125.

[11] L. Bloch, A. Boketta, C. Keibel, E. Mense, A. Michailutschenko, O. Pelka, J. Rückert, L. Willemeit, C. Friedrich, Combination of image and location information for snake species identification using object detection and efficient nets in: CLEF working notes, 2020.

[12] A. M. Durso, G. K. Moorthy, S. P. Mohanty, I. Bolon, M. Salathé, R. Ruiz De Castañeda, Supervised learning computer vision benchmark for snake species identification from photographs: Implications for herpetology and global health, Frontiers in Artificial Intelligence 4 (2021) 17.

[13] A. Joly, H. Goëau, S. Kahl, L. Picek, T. Lorieul, E. Cole, B. Deneu, M. Servajean, R. Ruiz De Castañeda, I. Bolon, H. Glotin, R. Planqué, W.-P. Vellinga, A. Dorso, H. Klinck, T. Denton, I. Eggel, P. Bonnet, H. Müller, Overview of lifeclef 2021: a system-oriented evaluation of automated species identification and species distribution prediction in: Proceedings of the Twelfth International Conference of the CLEF Association (CLEF 2021)

[14] S. Xie, R. Girshick, P. Dollár, Z. Tu, K. He, Aggregated residual transformations for deep neural networks in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017.