Animal Image Classification using Deep Learning

Thamindu Sasitha Melroy Pereira
The University of Adelaide

Abstract

Image classification is one of the most researched areas in the field of computer vision. Nowadays, classifying some objects into their particular classes is not more difficult because these problems are addressed using architectures like Convolution Neural Network (CNN). In this paper, we are trying to classify the animals into 151 categories, and we evaluate our prediction and efficiency using the accuracy and FLOPS metrics. We have obtained 97.93% accuracy in predicting the animal classes from the test data. Furthermore, for robustness, we also achieved 98.04% accuracy on the out-of-sample data extracted using scraping of the 151 animal classes.

1. Introduction

Image Classification is a fundamental task that attempts to comprehend an entire image as a whole. The goal is to classify the image by assigning it to a specific label. Image Classification typically relates to the analysis of photographs with just one object visible and require precisely labeled large-scale datasets. In image classification, given an input image, the goal is to predict the class which it belongs to. For humans, this is not a major deal, but teaching computers to sight is a challenging task that has become a broad area of research interest, and both classic computer vision and Deep Learning (DL) techniques have been developed. Classic techniques use local descriptors to try to find similarities between images, but today, advances in technology allow the use of DL techniques to automatically extract representative features and patterns from each image.

CNN (convolutional neural network) is one category of deep learning neutral networks. It is a kind of multilayer neural network with artificial neurons. The multilayer mianly consists of three basic types: input layer, hidden layer and output layer. The kernel existed in first hidden layer can gain the convolved feature from input images and then the concolved feature can be transmitted to the next layer. During the process, the feature obtained becomes more complex as layer is deeper. With different image classes, CNN can extract picture features respectively and proceed identification.

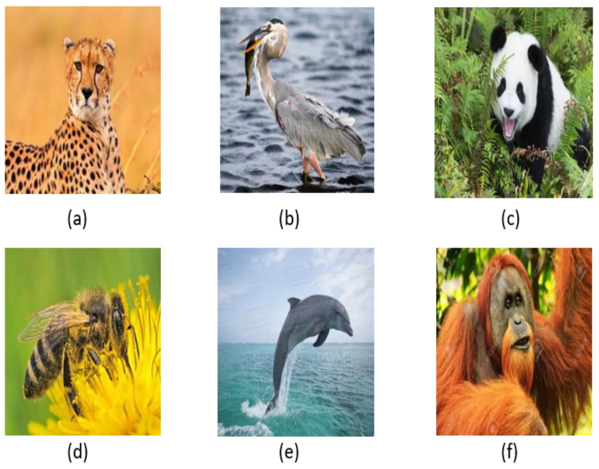

Figure 1. Image samples from class (a) acinonyx-jubatus (b) ardea-herodias (c) ailuropoda-melanoleuca (d) apis-mellifera (e) pongo-abelii (f) tursiops-truncatus of the dataset

Here, the task is to train a classification model to categorize each animal into one of the 151 categories for the Animal dataset which contains a total of 6270 images with 151 different animals. In this report, we have implemented multiple deep learning models to classify the animals into 151 category.

The primary or the base model is the simple four Convolution Neural networks followed up by Max pooling and ReLu as the activation function. The accuracy of the baseline is 37%. Therefore, we have implemented improved version of the baseline model using other techniques to get the accuracy above 95% for the two test datasets which includes given test data from the assignment and the externally scraped image to test the robustness of the model for prediction. We present multiple experiments of different image classification models to find out the best performing model for the animal dataset.

In summary, our contribution to the literature can be listed as follows: • Applying the data augmentation strategy separately for train and validation data to improve the accuracy. • Applying the Validation strategy by utilising the Validation data to minimise the mis-classification error. • Testing the robustness of the model using out of sample datasets.

The organization of this paper is as follows: Section 2 reviews the prior work related to CNN methods such as AlexNet, ResNet and DenseNet. Section 3 starts with a detailed analysis of baseline model, followed up with our approach of using the methodology of transfer learning and pre-trained weights for image classification. Section 4 gives an in-detail analysis and results of our experiments to improve the accuracy and lessen the computational cost. Section 5 contains the Conclusion of the experimental results and the limitations.

2. Related Work

Convolutional neural networks (CNNs) have supplanted other machine learning techniques for the recognition of visual objects. Since the initial discovery of neural networks, the investigation of network topologies has been a part of that research [5]. Deep convolutional neural networks [4,5] have led to a series of breakthroughs for image classification [4, 7, 11]. Deep networks naturally incorporate low, mid, and high level features [11] and classifiers in an end-to-end multilayer way, and the ”levels” of features may be enhanced by the quantity of stacked layers (depth).

2.1. AlexNet

Alex et al [4] wrote a highly-optimized GPU implementation of 2D convolution and all the other operations inherent in training convolutional neural networks. Alexnet [4] has several novel and peculiar characteristics that enhance its functionality and shorten training time. Each convolutional layer, which only makes up 1 percent of the model's parameters, was found to have a negative impact on performance.

Alex et al [4] trained a large, deep convolutional neural network on the subsets of ImageNet competitions [1] and achieved top-1 and top-5 error rates of 37.5% and 17.0%. The neural network, which has 60 million parameters and 650,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and three fully-connected layers with a final 1000-way softmax. To make training faster, Alex et al [4] used non-saturating neurons and a very efficient GPU implementation of the convolution operation. To reduce overfitting in the fully-connected layers Alex et al [4] employed a regularization method called “dropout” that proved to be very effective. Alexnet [4] experiments suggested that the network's size is limited mainly by the amount of memory available on the GPUs and by the amount of training time.

2.2. ResNet

Early studies [8, 10] demonstrated the critical relevance of network depth, and all of the top results [8, 10] on the challenging ImageNet dataset [6] used ”extremely deep” [8] models. In order to make training networks that are far deeper than those previously utilised, Kaiming et al [2] provided a residual learning approach. Kaiming et al [2] intentionally re-formulate the layers such that they learn residual functions with reference to the layer inputs rather than learning unreferenced functions.Residual nets allow these layers to suit a residual mapping rather than expecting that each few stacked layers directly match a desired underlying mapping. To create a network, they layer leftover bricks on top of one another.

On the ImageNet dataset Kaiming et al [2] evaluate residual nets with a depth of up to 152 layers while still having lower complexity. An ensemble of these residual nets achieves 3.57% error on the ImageNet test set. On the COCO object identification dataset, Kaiming et al [2] achieve a 28% relative improvement as a result of extremely deep representations.

2.3. DenseNet

Recent work [2, 9] has shown that convolutional networks can be substantially deeper, more accurate, and efficient to train if they contain shorter connections between layers close to the input and those close to the output. Gao at el [3] introduced the Dense Convolutional Network (DenseNet), which connects each layer to every other layer in a feed-forward fashion to ensure maximum information flow between layers in the network. All previous layers' feature-maps are utilised as inputs for each layer, and that layer's own feature-map is used as an input for all layers that come after it. The vanishing-gradient issue is solved, feature propagation is strengthened, feature reuse is encouraged, and the number of parameters is significantly reduced due to dense networks.

DenseNet require much fewer parameters than existing algorithms with comparable accuracy and less computation. On the ImageNet dataset Gao at el [3] evaluate 5.29% top 5 error rate.

3. Method

3.1. Baseline Model Analysis

The Baseline model comprises of four Convolution Neural networks followed up by Max pooling and ReLu as the activation function. The Baseline model is using the Padding in all the layers to preserve the original form of the image. However, we observed that the filter depth is increasing at every layer that is, during the input it started from 3 channel and at the final layer the filters have 128

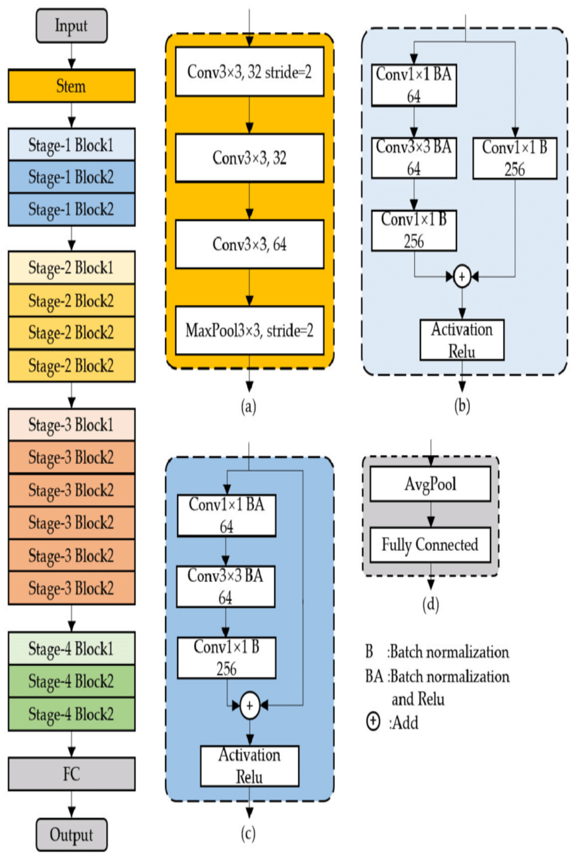

Figure 2. ResNet-50 Architecture

channel and this will lead to higher feature extraction from the images.

In the Baseline Model it uses the Adam optimization, Cross Entropy Loss as a loss function. It has one fully connected layer of 151 neurons which uses the Log Softmax at the output layer for classifying the image classes. The Baseline Model with 10 epochs has obtained 36.7% accuracy on the test data whereas, FLOPS are 0.69G.

3.2. Our Approach

Our approach for solving the animal classification problem was mainly a trial and error of using various pre-trained architectures with good data augmentation and validation strategies. The best selected model from the pool of experimental models using the given test data was later tested with out of sample data. Here, out of sample data, are the scraped images from the Google image downloader. In our experimentation, we tested the dataset on Alexnet [4], Resnet50 [2], and Densenet121 [3] pre-trained models with weights and without weights from the PyTorch. The best selected model, Resnet50 [2], had a 98% accuracy on the given test data, whereas it had a 98.04% accuracy on out-of-sample test data.

We conducted several analyses on the images, and one of the analyses was to identify the class balance on all the data splits, that is, training, validation, and test data. We found that the training and testing data had all the image classes, i.e., 151, whereas the validation data had only 134 unique image classes. Since validation data is important for determining the best parameter for test data, having all the classes of images in the validation data is crucial. Because of having missing classes in the validation data, we had less accuracy in the test data. Since we had a restriction on changing the data split, we needed a good validation strategy.

Due to restrictions on changing the data split and its images, we came up with a solution to minimise the validation misclassification error. Here, we used the tracker to track the validation misclassification for each class during training epochs. The misclassified classes of validation data were later extracted a few samples from the training data and the data augmented them in various ways so that the model learns better for these misclassified classes to select the best parameters. Using this approach, we were able to get a higher score from 96% to 97% using ResNet50 [2].Our validation strategy acts as our failure case study because here we are trying to learn the failure cases of validation data in the train data by augmenting the datasets multiple times so that model learns the animals appearance in various orientation.

We found that data augmentation was a crucial step in the training and validation data because, while we used the same data augmentation on all the data splits, we achieved a 79.74% accuracy using AlexNet [4] on given test data. However, we obtained better accuracy of 86.76% using the AlexNet [4] when the data augmentation was different for training and validation data. We found that the training data had many augmentation steps and the output image after transformation was a little different in brightness, angle, cut outs etc. Hence, the original physical structure of animals was missing due to augmentation. We kept the validation data with minimal augmentation using the Resize method only. Hence, having the original image structure helped us to score higher in the AlexNet architecture [4].

We found that ResNet50 [2] and DenseNet121 [3] had similar accuracy, but DenseNet121 [3] was slower compared to ResNet50 [2], hence we chose ResNet50 [2] as our final model. However, we wanted our model to be robust, so we tested the model using various out-of-sample data by scraping the images from Google image downloader, a third-party package. We observed that ResNet50 [2] had a 98.04% accuracy on the external images.

Therefore, using all the above approaches and strategies our model was able to learn and predict better.

4. Experiments

4.1. Dataset Description

In the given animal classification dataset, we have 6270 images and we used random split for dividing the datasets into training, validation, and testing data by using a random seed of 10. There are 5,330 images of training data, 313 images of validation data, and 627 images of testing data. These images are of 224 * 224 size, and these datasets are used for finding the best model with the best model parameters. whereas we also used out-of-sample animal classification data scraped from Google Image Downloader, a third-party package, to test the model's predictability. These external images are 224 * 224 in size and in jpg format.

4.2. Experimental Analysis

In our experiment we have implemented Multiple models and they are as follows:
1. AlexNet
2. DenseNet121
3. ResNet50

Above models are tested with pre trained weight and without pre trained model to analyze the performance of the model. Models are trained and tested using several hyper parameters to get the best model. We implement our experiments with PyTorch, and due to limitation in computational power we used Google Colab and Lambda Labs Cloud GPU 1 RTX 6000. Our experiment on each model has gone into various strategies to identify the best model and below is the table that shows various strategies we employed during the selection of the model. Following are approaches and strategies used for the models

Models	Same Data Augmentation	Different Data Augmentation	Validation Trick	Accuracy (%)
Baseline	Y			36.7
AlexNet	Y			79.74
AlexNet		Y		86.76
DenseNet121		Y	Y	97
ResNet50		Y	Y	98

Figure 3. Approaches and strategies used for the models

From the above table we can observe that we were able to predict with 98% accuracy using strategies like different data augmentation on train and validation data with validation strategy. For every model we tested with it's corresponding pre trained weight and without pre trained weights to understand the prediction accuracy. Below is the table that shows various model with it's corresponding weights and accuracy. Following are the performance metrics obtained from the various models in our experiment.

Models	Pre-trained weights	Without pre-trained weights	Accuracy (%)	Loss	FLOPS
Baseline		Y	36.7	5.24	0.69G
AlexNet	Y		86.44	1.625	1.39G
AlexNet		Y	2.71	5.016	1.39G
DenseNet121	Y		97	0.48	5.69G
DenseNet121		Y	33.97	3.5604	5.69G
ResNet50	Y		98	0.4	8.17G
ResNet50		Y	36.52	3.64	8.17G

Figure 4. Performance Metrices of Various Plots

From the above analysis we can see that model with pre trained weight has a higher accuracy than the model without pre trained weight. To understand more about the model behavior we will analyze it using their corresponding train and validation data loss.

4.2.1 Comparing AlexNet using pre-trained weight and without pre-trained weight

AlexNet model without pre trained weight

AlexNet with pre-trained model, we were able to achieve 86.44% accuracy and the loss was 1.625 on the given test data. The efficiency of the model is determined by the FLOPS metric, and for the AlexNet with pre-trained weight, we obtained 1.39G. The AlexNet model without pre-trained weight has an accuracy of 2.71%, which is less than the model with pre-trained weight. Furthermore, we can analyze these models more carefully by looking into their train and validation loss curves.

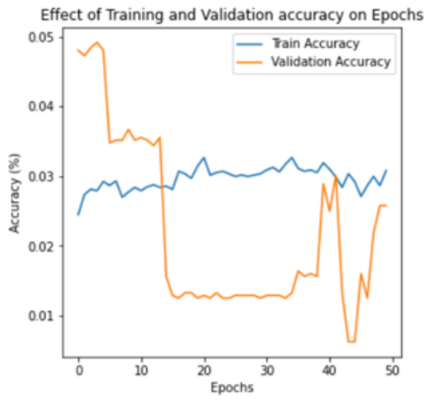

Figure 5. Accuracy plot

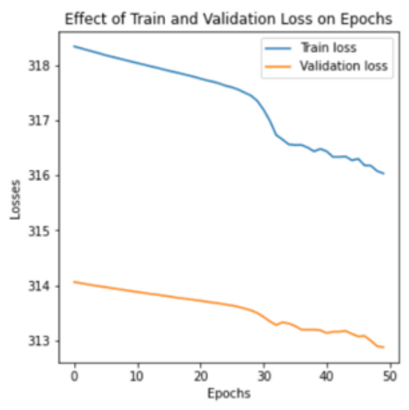

Figure 6. Loss plot

From the above plot, we can observe that the loss curve for the train and validation data using the model without pre-trained weight has a high loss. From the accuracy plot, we can see that the accuracy of the validation data is lower compared to the training data. Therefore we can say that using AlexNet without pre trained weight gives lower accuracy for the given animal classification task.

AlexNet model with pre trained weight
From the below plot of loss curve and accuracy plot, which is from the AlexNet model with pre-trained weight, we can observe that the curve is very smooth. However, we can see that after a few epochs of validation data, the loss stopped decreasing. The accuracy metric shows that the model has achieved accuracy of 86.44%, but we can see that the accuracy is flat after 10–20 epochs. Overall, we can say that for the AlexNet model on the animal classification task, it is better to use the pre-trained weight and model using different data augmentation for different splits.

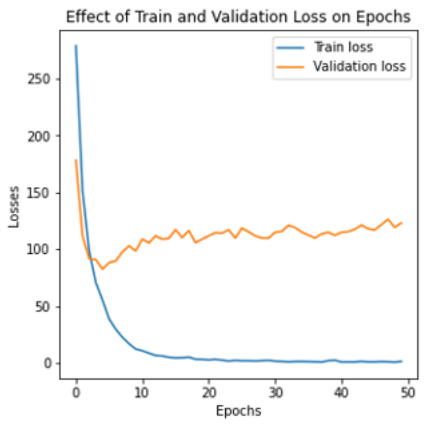

Figure 7. Accuracy plot

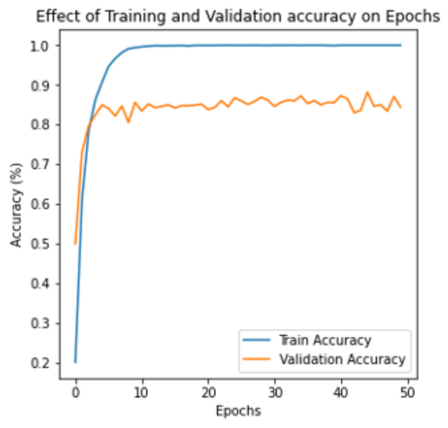

Figure 8. Loss plot

4.2.2 Comparing DenseNet121 using Pre trained weight and without pre trained weight

In the DenseNet121 model with and without pre-trained model, we used different data augmentation methods, and the model with pre-trained weight has an accuracy of 97%, while the model without pre-trained weight has 33.97%. Here we can clearly see that there is a large margin between the accuracy. However, the efficiency of the model is the same, that is, 5.69G FLOPS.

DenseNet121 model without pre trained weight
Following are the loss and accuracy plot for the DenseNet121 without pre trained weights:

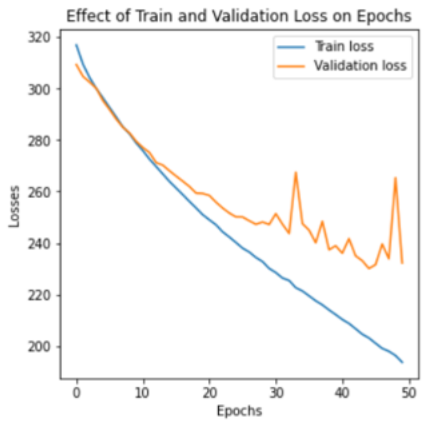

Figure 9. Accuracy plot

From the above plot, we can see that the loss curve and accuracy curve are not smooth, We also observed that DenseNet121 is slower compared to AlexNet. However, DenseNet121 with pre-trained weight is better in accuracy

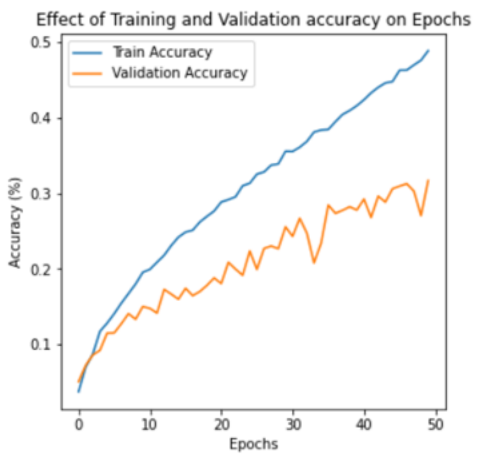

Figure 10. Loss plot

and predictions.

DenseNet121 model with pre trained weight
Following are the loss and accuracy plot for the DenseNet121 with pre trained weight

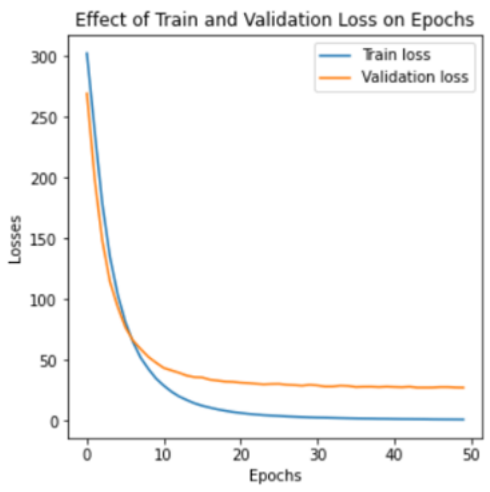

Figure 11. Accuracy plot

From the above plot, we can clearly see that training and validation losses are decreasing, whereas the accuracy of the model is very high. From the loss and accuracy plots, we can say that loss and accuracy plots indicate that Densenet121 is fitting the data clearly and it is making the predictions correctly. Hence, when comparing the model without pre-trained weight, DenseNet121 with a pre-trained model is better.

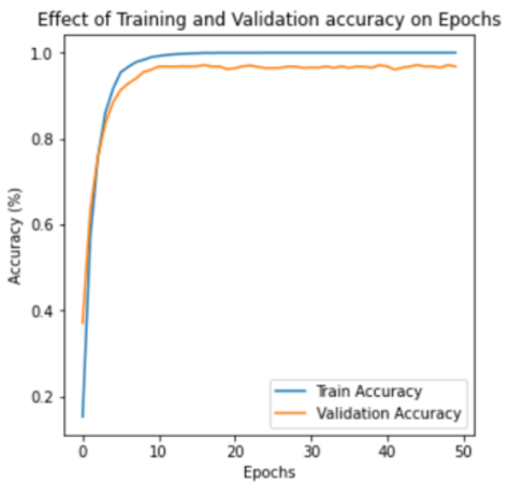

Figure 12. Loss plot

4.2.3 Comparing ResNet50 using Pre trained weight and without pre trained weight

In the ResNet50 model with and without pre-trained model, we used different data augmentation methods on data splits, and the model with pre-trained weight has an accuracy of 98%, while the model without pre-trained weight has 36.52%. Here we can clearly see that there is a large margin between the accuracy. However, the efficiency of the model is the same, that is, 8.17G FLOPS.

ResNet50 model without pre trained weight

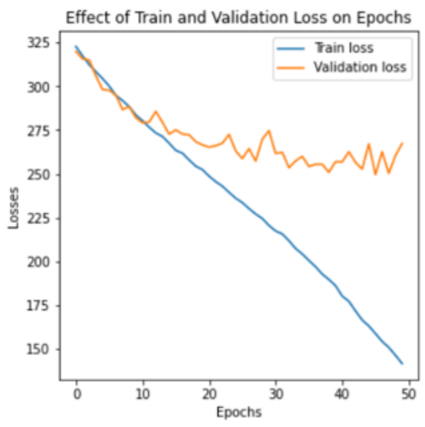

Figure 13. Accuracy plot

From the above loss curve and accuracy plot for model without pre trained weight has the similar plot as the DenseNet121. The curve plot for both loss accuracy are not smooth Therefore we can observe that these models are not suitable for prediction because of high loss after prediction.

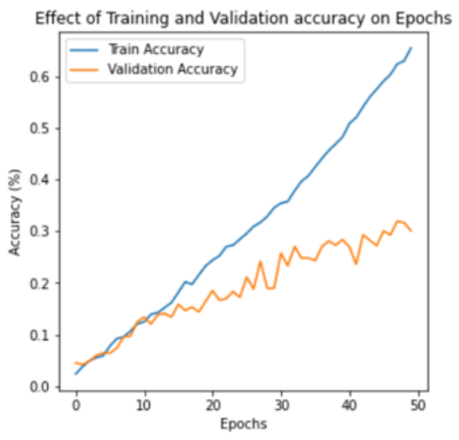

Figure 14. Loss plot

ResNet50 model with pre trained weight

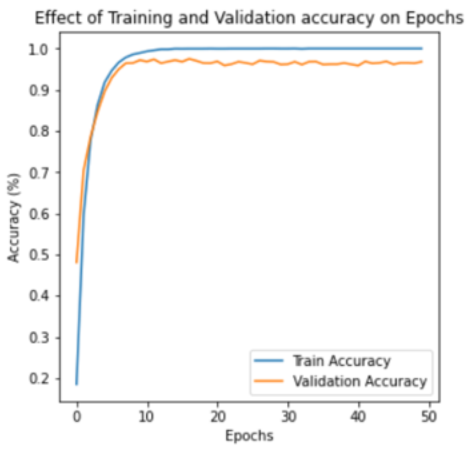

Figure 15. Accuracy plot

From the above analysis, we can see that using ResNet50 with pre trained weight model, has a good accuracy curve and loss curve. The loss in ResNet50 is gradually decreasing where as accuracy is increasing.

Hence from the above analysis we can say that model with pre trained weight are good for the given animal classification task. Therefore for best model we will consider using the pre trained weigh model.

4.2.4 Best Model

We have conducted multiple experiments for analyzing the images to develop the strategy to obtain the best accuracy for the animal classification task.

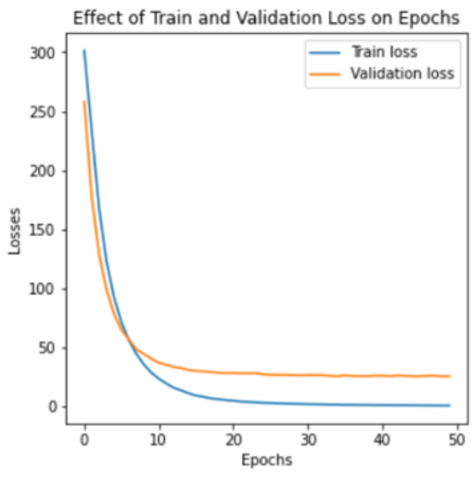

Figure 16. Loss plot

To obtain the best model we used following experimented strategies

We started our experiment from the data analysis of class balance where we found that validation data had missing classes and this observation led us to explore more on validation data and it's strategy because having the right validation strategy is important in any models because the validation data is used for selecting the best parameter. However, we had restriction that we cannot change the data split so we came up with idea of minimizing validation error. Here we extract the missing class from the validation data and use those classes to learn more in training data so that it predicts correctly in the test data. Using this approach it helped us to Jump over score from 96% to 97%.

We also used the data augmentation method to improve our score, by default we had the data augmentation for the entire data, and our accuracy was very less in AlexNet that is 79% . We understood that train data has many transformation and the output of the train transformed data was actually degrading the quality of physical appearance of the animal. So we though to apply transformation separately for validation data to preserve the physical appearance of the animal therefore we used simple transformation as resizing. This approach helped us to jump the score from 79% to 86% score.

We used those above strategies in our final ResNet50 model with following parameters:
Optimizer: SGD
Learning rate: 1e-3
Momentum: 0.9
Loss function: Cross Entropy Loss
Epoch: 50

While training and validation data we obtained the following graphs for loss curve and accuracy:

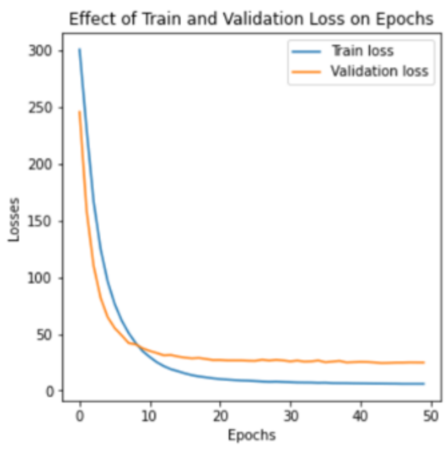

Figure 17. Accuracy plot

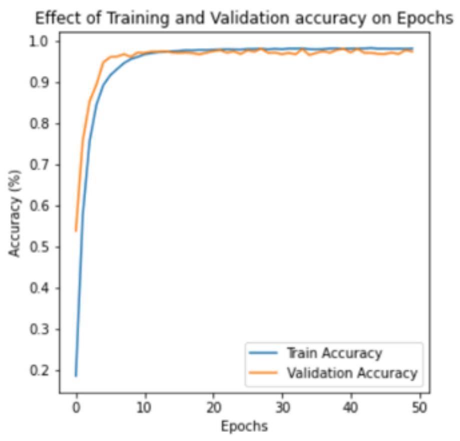

Figure 18. Loss plot

From the above plot we can see that, loss is gradually decreasing and accuracy is increasing and it is flat after 95% which indicates that the model is able to predict the validation data with 95%+ accuracy.

Using the above parameters and strategy on test data we were able to achieve 97.93% accuracy on the test data with 0.35 loss. The efficiency of the model is 8.17G. However to understand the robustness of the model we used extra test image of 1367 by scraping the internet for animal class as per the 151 classes and we found that we were still able to predict with 98.04% on external images.

From the above analysis we can find that these are best accuracy from each model. However, we will consider ResNet50 as our final model despite the DenseNet121 has 97% at 5.69G because of heavy computation required by the DenseNet121.

Table 1. Final Accuracy of the Models

Model	Accuracy(%)	FLOPS
Baseline	36.7	0.69G
AlexNet	86.44	1.39G
DenseNet	97	5.69G
ResNet50	98	8.17G

5. Conclusion

In this paper, we used image classification methods to classify the images from the large animal classes. In order to solve this problem, we approached multiple methods and strategies to identify the best model from the pool of many models. We applied multiple strategies, such as applying the data augmentation separately for training and validation data, which helped us jump the score from 79% to 86% using AlexNet. Furthermore, we also found that to minimise the validation misclassification error, we can use the missing classes of validation data to train more similar images in a more augmented form. Our validation strategy acts as our failure case study because here we are trying to learn the failure cases of validation data in the train data by augmenting the datasets multiple times so that model learns the animals appearance in various orientation. Therefore, after clear experimentation, we chose the ResNet50 as our best model as it was predicting with 97.96% accuracy for the given test data, whereas for out of sample data, it had predicted with 98% accuracy. We found that pre-trained models with weights are suitable for this animal classification task rather than using models without any pre-trained weights.

References

[1] A Berg and J Deng. and l fei-fei. large scale visual recognition challenge(ilsvrc), 2010. 2

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 2, 3

[3] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017. 2, 3

[4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012. 2, 3

[5] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989. 2

[6] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015. 2

[7] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013. 2

[8] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 2

[9] Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. Training very deep networks. *Advances in neural information processing systems*, 28, 2015. 2

[10] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015. 2

[11] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014. 2