**VLG Recruitment Challenge ‘24 Report**

* **Mehul Dhiman (24112073) (9817889909)**

**Introduction**

This project aimed to develop a Neural Network to classify animal images into their respective classes. The task posed unique challenges, particularly the requirement to handle classes effectively, using only the provided training data. The task involved training a CNN on a dataset of labeled animal images. Total there were 50 classes of animals but only 40 classes had training data which provided a unique problem. The focus was on extracting meaningful features from images to enable robust generalization.

### **Model Development**

#### **1. Data Preprocessing:** I took various procedures to process the data given.

* **Image Preprocessing techniques:** First the image is converted to 224 by 224 size as this size is most efficient for the DenseNet base that will be used by us later. As the cv2 module converts an image into bgr instead of rgb. So, we must first convert bgr into rgb as the DenseNet is trained on rgb array.
* **Normalization techniques:** We use the inbuilt preprocessing function by DenseNet so that it performs with the most efficiency. It scales the dataset which helps in better training of the model.
* **Augmentation techniques:** Using data augmentation caused a decrease in accuracy as observed by me, this may be explained by the fact that the test dataset is more closely related to the training dataset instead of any augmentation of it. So instead of data augmentation I used the extend function of python lists to copy paste the list and then used the slicing function to reduce it to 250 images. 250 was chosen as it was not too big so that RAM overflows and not too small that model overfits.

#### **2. Model Architecture:**

* The first layer of model includes DenseNet convolution layer with trainable equal to false. This will help in extracting the important features out of the image.
* Global avg pooling layer reduces the dimensions by avg out the image. It also reduces computation power and helps to reduce the noise. It also flattens out the array for the dense network to use.
* Next layer contains with a dense network with 1024 neurons and activation relu. Activation relu is used to reduce computation and L2 regularization is used to reduce overfitting.
* Dropout layer is here to simply reduce the overfitting that may arise from using 1024 neurons in the previous layer.
* Next layer contains the last layer which results in array of all classes. Softmax helps in resulting array being probabilities instead of continues values.

#### **3. Training:**

* **Loss Function:** As the output is not a hot encoded array but instead an index, that is why Sparse Categorical Cross-Entropy function is used.
* **Optimizer:** Adam optimizer is used with a customized learning rate of 1e-4. A low learning rate is used to reduce instability.
* **Epochs and Batch Size:** Full batch size and epochs of 20 is used. Full batch size is used to result in better model and 20 is fair enough as the convolution layer is set to be not trained
* **Hardware:** Kaggle cloud environment is used with GPU P1000 to speed up the model training.

### **Explainability:**

* DenseNet provides us with suitable weights with many convolution layers which helps in extracting features out of the image.
* First layer helps in extracting edges and curves out of image that is then passed to next layer
* Second layer helps in bringing more out of these basic features like resulting in ears and eye
* After many layers of DenseNet, the model can figure out features and their positions in the image which is then passed to pooling layer to easy out computation.
* Then the dense network helps in recognizing the image by using features and their positions in the image.
* The dropout layer and L2 regularization is used to prevent overfitting the model.
* At last, the last layer outputs array with probabilities of each class as softmax function is used

### **Results**

* Model generalizes well with each input and new class because, as explained before, convolution layers of DenseNet provide optimum feature extraction which later can be used to train model. This enrich feature extraction is one of the important reasons why model generalizes.
* Later dense network layers help in identifying the image.
* Accuracy of the model is taken care by various methods such as using optimum epochs , regularization , low learning rate so that model does not overfits.

### **Conclusion**

### **Challenges**

* First challenge faced by me was using kaggle but eventually I learnt it step by step
* Then I got to know that I have to learn pretrained models to get better accuracy and so I learnt them and it took a lot of time but it was worth it.
* Then I was having a challenge that my model was underfitting when using the pretrained model , we caused me a lot of anger issues but after a lot of googling and chatgpt I was able to figure it out that cv2 was converting image into bgr format instead of rgb and that I have to use the inbuilt preprocessor of the pretrained model to get the best efficiency out of that model.
* Later I faced a problem that I realized later on that the training data has less classes than test dataset, so I then thought of the solution that is importing data of those 10 classes. It gave me a little problem in importing as I was new to kaggle but after consistent persistence I was able to import them.
* Then I realized that dataset had a imbalance which was causing problem in training instead of going the traditional way of data augmentation I went the route of simply copying and pasting the data over and over again as I observed that data augmentation was harming the model instead of helping by decreasing the accuracy.
* Another problem I faced was a lot of errors and debugging. There were a lot of instances I wanted to just punch the laptop but worked with patience.
* Lastly, the main challenge faced by me was getting out of my comfort zone. Even though never tried kaggle I still participated in the challenge, and as I observed that the growth was exponentially huge. Now I will be able to participate in competitions with more confidence.

### **Learning outcomes**

* Yes, I learnt a lot of things out of this challenge. From a knowledge perspective I learnt a lot from fixing bugs to pretrained models, from kaggle to how to handle an error. It helped me open my knowledge barriers which was helpful.
* From a mental perspective, I also grew. I became more patient with bugs and errors, how to push even though mind says to give up, to give it all, to get out of comfort zone, and lastly to grow and never stop learning.