**VLG Recruitment Challenge ‘24 Report**

* **AMEYA GUPTA (24112016) (7895646931)**

**Introduction**

This project was to design a neural network for classifying images of different animals into classes totaling 50(40 seen classes(given in train dataset) and 10 unseen classes), by leveraging the combination of CNNs and model local feature extraction and global dependency. The emphasis for this task had to be laid on the robust learning and generalization of features from the training dataset provided.

### **Model Development**

#### **1. Data Preprocessing:**

* All images were resized to a size of 240x240
* For better generalization of the model Random Zoom and Random Horizontal flip was applied
* Random Rotation by +- 72 degrees was also used for the same reason
* Random Contrast and Brightness too

#### **2. Model Architecture:** EfficientNetV2L(last 80 layers unfreezed) a powerful CNN for image classification and 2 dense (ReLU) layers and a dropout layer..

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

* **Loss Function:SparseCategoricalCrossentropy(from\_logits=True, for softmax outputs)**
* **Optimizer:SGD Optimizer was used**
* **Epochs and Batch Size:20 epochs and batch size was 32**
* **Hardware: GEFORCE RTX 4050**

### **Explainability:**

1. **Data Augmentation**:  
   * The data augmentation layer helps artificially expand the training dataset by applying random transformations like flipping, rotation, zoom, brightness, and contrast. This improves the model’s ability to generalize, reducing overfitting by providing more variations of the same data.
2. **Pretrained Model (EfficientNetV2L)**:  
   * I’m using a pre-trained EfficientNetV2L model, which has been trained on a large dataset (ImageNet). This provides a robust starting point since the pretrained model already has learned useful features for visual patterns, this improves accuracy by leveraging learned features from ImageNet, which are useful for many general tasks.
   * Also this model has one of the highest accuracy available on keras applications….
   * I also liked the working method of the model as it modifies the Conv. layers in it too by rescaling the images….changing the dimensions to filters….modifying feature maps , modifying depth and many more….  
     Therefore I went with it…
   * I also un freezed the last 80 layers so that the model can use pretrained weights but also alter the weights according to the database we have provided..giving a better flattened input to aur dense layer up next…
   * Though using this model and also unfreezing few layers was computationally expensive i still went with this…as i had enough resources for this large model to work
3. **Dense Layer**:  
   * I have used a dense layer with 256 units and ReLU activation function for further process and also used regularization for reducing the impact of large weights…overall providing less overfitting….
4. **Dropout**:  
   * I’ve also used a dropout layer with a rate of 0.5 randomly disables 50% of the neurons during training, which also helps prevent overfitting. By reducing reliance on specific neurons, dropout improves generalization.
5. **Final Layer**:  
   * The final Dense(50) layer outputs a vector of 50 values, corresponding to your 50 classes. I’ve used the softmax activation in the loss function (from\_logits=True) to reduce computational errors if I would have used softmax activation directly in the final layer to output probabilities for each class.

Together, this method efficiently combines data augmentation, a pretrained model for feature extraction, and techniques like regularization and dropout to prevent overfitting while fine-tuning the model on your specific dataset.

### **Results**

### Train dataset = blue

Validation dataset = orange

### 

* **At the 20th epoch  
  Train\_loss = 0.9307  
  Train\_accuracy = 0.9221  
  Validation\_error = 0.7837  
  Validation\_accuracy = 0.9434**
* **And got .93385 prediction accuracy in kaggle(60% of the predictions)**

### **Conclusion**

### **Challenges**

* I was a newbie in this area..so i know basics only…so this was my first image classification project…so first i viewed a lot of tutorials and github repos on this with trying to learn every new concept…

1. **Installing Tensorflow on my laptop**As I wanted to use my laptop's resources, I had to install tensorflow-gpu…which took me a day as I was new to this library and all the dependencies it needs.
2. **Using Self made architecture**I experimented with a lot of different architectures…to reduce overfitting ... .but the val\_accuracy stuck at .27…later I came to know about transfer learning.
3. **Selecting best pre trained model**I played with multiple models available on keras applications ... .and tried to understand their working. I tried ResNet50(val\_accuracy = .75)....ResNet101(val\_accuracy = .80)....ResNet101V2(was overfitting a lot)....  
    So after a lot of researching i settle for EfficientNetV2L(val\_accuracy = .93)
4. **Selecting unfreezed layers and next dense layer**It was a game of trial and error as I knew what to do for overfitting. I played along with the number of units,regularizer value, number of layers to unfreeze,dropout layer value……
5. **Selecting Optimizer**I experimented with Adam and SGD. I tried to understand how they work…and after seeing the results I went with SGD as it was faster…
6. **Getting data for unseen classes**Actually I saw at very end that there were 50 classes and the train data was given for 40 classes only…so then i started to collect dataset images for unseen classes and it took me a day to collect images as i tried to learn web scraping but at the end i had to use a chrome extension only…  
   This gave my prediction accuracy a huge boost
7. So I experimented with multiple more things like resolution, batch size, validation percentage and also tried augmenting data which indeed gave me better results.

* All this took a lot of time and experimentation…..i believe it was the right decision to use dedicated gpu for training to get all the control in my hand and train the model as much as i want…as i learned from experimentation only..

### **Learning outcomes**

* I learned a hell lot of new things from this challenge….  
  I learned about the CNN technology…preprocessing data…..data augmentation……how different factors affect the accuracy…how tensorflow works..also a bit of web scraping..  
  Transfer learning….I would say I learned a lot in these last few days….
* I regret that I wasn't able to use the predicate matrices given…I would like to learn more.

**Analysis of Special Package**

* The model performed really well on this data…..  
  **ACCURACY = 99%**
* Only 1 image was predicted wrong (063.png)  
  predicted = hippopotamus  
  Actual = beaver
* I would say this dataset was hard as many animals were shown in hard surrounding and with objects blocking them too
* I think…. use of data augmentation and also using pretrained model on large dataset helped a lot
* If we use a dataset like this.. The model will train out to be more generalised and robust as the animal in many cases is shown with other objects ... .different orientations and surroundings…which will give us a better model to perform well on new data.