VLG Recruitment Challenge ‘24 Report

NAME: Agrim Singh **Enrollment Number:**24321003 **WhatsApp Number:** 7599956464

---

https://github.com/th-efool/PixelPlay

Introduction

Hello, I am Agrim Singh. I am really passionate about learning about new technologies, the field of Artificial Intelligence has intrigued me for long time, this project helped me get inducted into this world.
While I am new and my knowledge might be a bit lacking for now. I am really passionate about this and want to learning everything and anything.

Approach

In this project to build a species classification model with 40 seen classes and 10 unseen classes. This project helped a ton in catapulting my progress in my AI ML learning journey.

To solve this problem, i tried going with two different approaches

1. ZSL model that utilised features and predicate matrix file provided by the competition holder, which made up for the lack of data on unseen classes
2. Building up dataset for the unseen classes

After trying and testing out lot of things and different models, i decided to go with the second one.

I builded up data for the unseen classes using Google’s “Custom Search JSON API” and when i ran out of daily limit i used “beautifulsoup” *to procure images with bing*

I would have really really liked to have uploaded that model instead of this current one but yes, i’m really grateful for this competition in helping me expand my knowledge regarding deep learning.

Model Development

1. Data Preprocessing

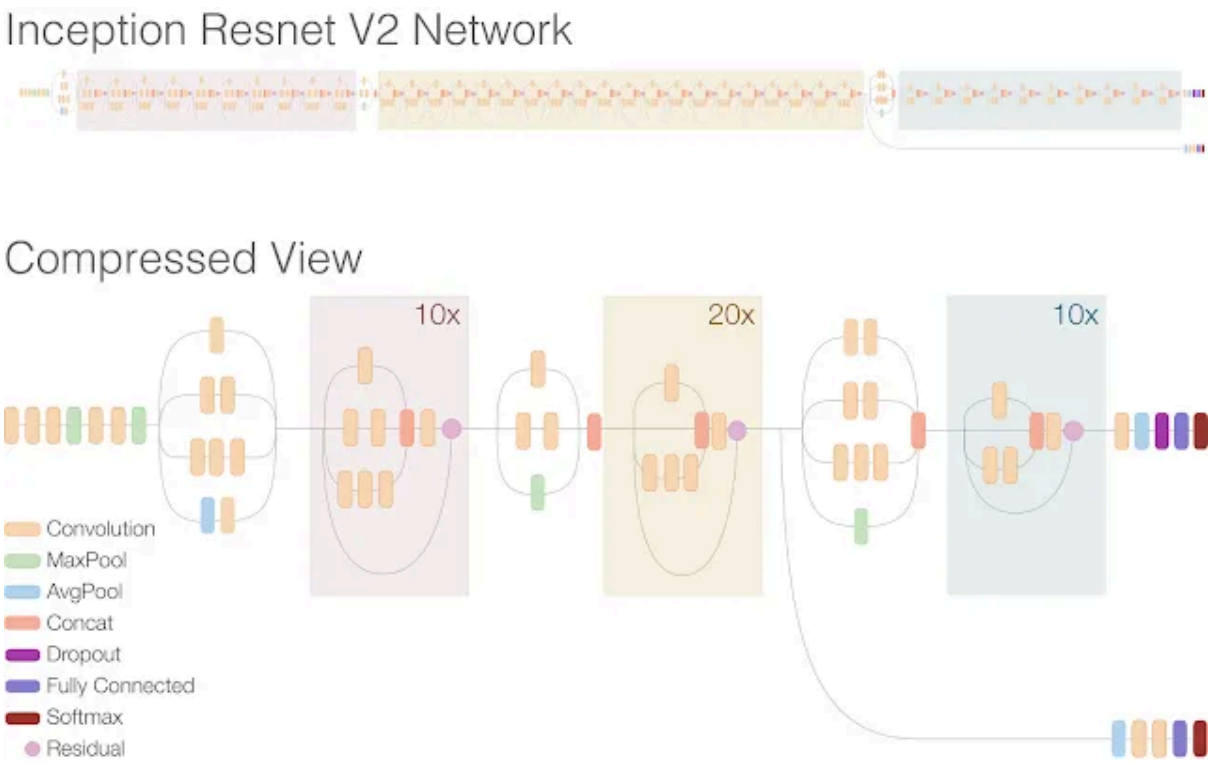
- **Images Normalization:** Scaled everything to 0-1 to keep it uniform.
- **Handling Data:** Splitted data into train(85), validate(15) {in the initial version had 70 15 15 split, but since the data was very little, i decided only having validation set and removing the test set)
- **Augmentation:** Flipping, zooming, rotating – to make up for limited data that there was

2. Model Architecture

Used "InceptionResNetV2" as it yielded the best result out of the models that i have tried.

I later delved deeper into learning about "InceptionResNetV2" to gain deeper insights into it.

It was specifically this Medium Article, about **building InceptionResnetV2 model from scratch** that gave me lot of insights into functioning and workings of the IncepitonResnetFramework
https://medium.com/the-owl/building-inception-resnet-v2-in-keras-from-scratch-a3546c4d93f0



With the output of the InceptionResnetV2 model.

I attached an **Global Average Pooling 2D** layer to squash info nicely without throwing any important bits away.

And connected that **GAP2D Layer** with an Dense Layer for Phase1 Training Model

For the phase2 training model, i introduced an dropout layer to introduce some randomness

3. Training

- **Optimizer:** Adam, and also, the learning rate (0.01 in Phase 1 and smaller in Phase 2) made a really big difference for me).
- **Epochs:** Started with 8 aggressive ones, then dialed it back to refine.
- **Hardware:** Ran this on “i3 12th Gen Intel(R) Core(TM) i3-12100 3.30GHz

---

Explainability

Loss Function
The loss function is basically what tells the model how wrong it is after making predictions. It’s like a scorecard during training that helps adjust the model to perform better. For classification, I used categorical cross-entropy because it compares the predicted probabilities with the actual class labels and punishes **wrong confident predictions a lot**.

In my project, the loss function played a huge role in helping the model figure out how to handle both seen and unseen classes better.

InceptionResNetV2 combines the strengths of Inception modules and residual connections, making it a powerful architecture for image classification. It progresses through a hierarchy of features, from basic patterns to class-specific details, while residual connections help avoid the vanishing gradient problem. This ensures the model can learn effectively, even with very deep layers.

Low-Level Features
The initial convolutional layers detect basic elements like edges, corners, and textures, which act as the foundation for more complex features. Pooling layers, like max pooling, reduce the size of feature maps while preserving essential information.
Example: For a horse, the model first detects the outlines of the body and the textures of the coat.

Mid-Level Features
Inception modules come into play here, allowing the model to process patterns at multiple scales simultaneously. This helps the model recognize medium-level features like shapes and patterns, while residual connections ensure smooth learning by skipping layers when necessary.
Example: At this stage, the model identifies the flow of the horse's mane or the gradient/contours of its legs.

High-Level Features
The deeper convolutional layers focus on class-specific attributes, enabling the model to distinguish between visually similar categories. The Global Average Pooling (GAP) layer compresses the spatial information into a compact representation, preventing overfitting and summarizing the image.
Example: The model can now differentiate between the mane of a horse and the stripes of a zebra.

How It All Comes Together
By combining low-level, mid-level, and high-level features, InceptionResNetV2 works systematically to classify images accurately. For example, it starts with edges and textures, identifies shapes and patterns in the middle layers, and finally uses high-level attributes to confidently label the image as a horse.

Model Development

The model was developed in two phases

Phase 1: Aggressive Training with High Learning Rate

- Learning Rate: Adam optimizer with a learning rate of 0.01
- Training Epochs: Around 8 epochs
- Layer Structure:
    - Pretrained "InceptionResNetV2" model
    - Global Average Pooling 2D layer
    - Dense output layer (activation="softmax")

Phase 2: Fine-Tuning with Lower Learning Rate and Dropout Layer

- Model Initialization: Loaded the best model from Phase 1
- Dropout: Added a `Dropout(0.15)` layer to prevent overfitting
- Dense Layers: Final Dense output layer with softmax activation.
- Other configurations with the follow did not improve performance significantly.
    - additional ReLU-activated dense layers
    - trying unfreezing top30 layers, top100, last 30

- Applying weights to attended features using "Multiply()([x, attention_weights])" before procedding to concatenating GAP2D GMP2D
- Other attempts to introduce some non-linearity

Results

Phase2
Epoch 10/25
313/313 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 523s 2s/step - accuracy: 0.8265 - loss: 1.4712 - val_accuracy: 0.9123 - val_loss: 1.0106 - learning_rate: 1.0000e-05

THE DATA RIGHT BEFORE THE INFLICTION POINT BEFORE THE MODEL STARTED OVERSHOOTING AND MEMORIZING/OVERFITTING instead of GENERALIZING

Conclusion

Challenges

1. **Handling Limited Visual Information -** on testing and checking i saw model struggle where limited image dataset was available and where only partial visual features were present
2. **Model Architecture Selection -** i spent lot of time selecting different models, making changes in code, to make sure that it bought out the models max potential and judging and testing based on that.
3. **Hyperparameter Optimization -** using batch size of 32 really heated by laptop a lot that it shut down few times, so i had to play with batch size settings, learning rate, dropout rate
4. **Incorporating Unseen Classes -** it took so much of my cognitive effort to try to find way to make some strategy for the unseen classes specifically
5. **Fine-Tuning the Model -** After training the model once, i trained it again with little tweeks for fine tuning for that, i builded and tried lot of different models that'd make the fine-tuning optimal
6. **Finding the model Overfitting** at different places and trying to come up strategy to tackle the problem

Insights Gained

1. This project, re-reminded of how important Adaptive experimentation is, instead of being adamant on just one way of doing things.
2. Learning about different model architectures, this experience would be really really so helpful for me for future classification projects as an president

I learned a really lot lot from this, i got familiar with so many terms, spent hours learning researching on different concepts, trying testing configuring.

I literally spent hours asking gemini and chatgpt questions on everything, each minute detail on what is that, how is it working, how does it relate to this, to that. Implementation cases, practicality, etc, etc.