**Pixel Play'25 Challenge**

-Aditya Chauhan (24125002) <8955396001>

**Introduction**

Zero-shot image classification aims to classify images into categories that that the model has not seen during training. Here this problem was approached by leveraging the semantic relationships between the embedding vectors of the animals of seen and unseen classes. Various iterations were made in the model to predict the unseen classes with higher accuracy.

**Model**

The models were implemented using TensorFlow and Keras. Transfer learning and fine tuning were employed because of the low number of images provided for training.

* *Preprocessing and data augmentation:*

All images were resized to 224\*224 and rescaled by dividing pixel values by 255 for faster training.

Keras Data augmentation layers RandomCrop, RandomFlip, RandomRotation and RandomContrast were used for the most part.

* *Model architecture:*

MobileNetV2 was used for feature extraction because it is fast and quickly converges to a solution. It also achieves competitive accuracy compared to more computationally expensive models.

The architecture of MobileNetV2 consists of a series of convolutional layers, followed by depth-wise separable convolutions, bottleneck design and squeeze-and-excitation (SE) blocks. These components reduce the number of parameters and computations required while maintaining the model’s ability to capture complex features.

The top layer(s) of the model are explained under the Explainability heading.

* *Training:*

1. Loss: Categorical cross entropy was used in Model-1 whereas MSE was used in Model-2
2. Optimizer: Adam was used as it works just fine for most applications
3. Regularization: Dropout regularization was used.
4. Hardware: Kaggle GPU P100

**Explainability**

The provided predicate matrix represents all classes as 85-dimensional vectors in the embedding space, here each direction in the semantic space has a certain meaning (for example colour, size etc.). Due to this property simple mathematical equations can be used to derive new meaningful vectors. For example:

***E***(man) – ***E***(woman) = ***E***(uncle) – ***E***(aunt) {here ***E***(x)represents the embedding vector of ‘x’}

Here the difference in the two vectors on either sides correspond to only the gender feature, therefore both sides of the equation, which is another vector, come out to be almost equal.

Theoretically, this logic should be applicable to all the features in our 85-dimensional embedding space.

Also, similar to the intuition of dot product in 3-D space, the value of dot product of two 85-D vectors divided by their L2 norms represents how similar they are.

Note that here we are assuming that the provided semantic relationships in form of vectors are directly related to visual features of the animals.

The following are two of the approaches that I came up with:

**Model-1:** Say we have an unseen class Z and two seen classes X and Y, then if our model predicts (in the final softmax layer) probability of image being of class X to be 0.7 and of class Y as 0.3 then the embedding vector of Z should be close to the linear combination of embedding vectors of X and Y, that is

***E***(Z) = 0.7\****E***(X) + 0.3\****E***(Y)

The final prediction vector was computed using this concept and finally cosine similarity was used to predict the class.

**Model-2:** This model was trained to directly predict the vector embeddings from given image (without the softmax activation layer), that is, the output layer contained 85 units and MSE loss was used for training.

**[[1]](#footnote-1)**

**Results**

The overall accuracy on the test set was 58.45% using Model-2 but both the models perform much better in the cases where the test set contained of only one type of images either seen or unseen.

On the Special Package greater than 92% accuracy was achieved because it contained only the images of only one type, that is the seen classes. Performance on the special package can be increased by simply training the model without using any vector embeddings concept (as it converts to a basic supervised learning task).

Similarly, if we consider that the test set contains images from unseen classes only then almost 42% accuracy\* was achieved on the unseen classes! That is pretty high considering the model had never seen even a single image from those classes during training.

**Challenges**

* Generalized zero-shot learning puts forward a tough challenge because it is difficult for the model to estimate if a given sample is from a seen or unseen class, therefore the performance in the case where only samples of one type are included is much better than in the case of generalized samples
* As stated above maintaining accuracy on both seen and unseen classes involves a major trade-off
* Finding an appropriate method to use the provided vector embeddings

**Learning outcomes**

* Explored the concept of meta learning for zero and one-shot classification but was unable implement it, would love to learn more about it and implement it with the VLG team in the future
* Concepts of transfer learning and fine tuning
* Explored TensorFlow’s functional and sub-classing APIs while also strengthening my command on the sequential API
* Working with pandas and scikit-learn

1. \* 18.9% accuracy on the Kaggle test set(when only predicting out of the 10 unseen classes) in which approximately 45% images were from the unseen classes, hence 18.9% divided by 0.45, which gives 42% [↑](#footnote-ref-1)