Google Universal Image Embedding Challenge

Sumant Kulkarni
Arizona State University
Tempe, AZ
sskulk30@asu.edu
1222412217

Pratyush Pandey
Arizona State University
Tempe, AZ
ppande28@asu.edu
1225718442

Dhanraj Bhosale
Arizona State University
Tempe, AZ
dbhosal1@asu.edu
1225506620

Shayal Shamsu
Arizona State University
Tempe, AZ
sshamsu@asu.edu
1222629096

***Abstract*—A key component of computer vision applications are image representations. Traditionally, per-domain models have been the main focus of research on image embedding learning. Instead of creating generic embedding models that might be applied to all domains simultaneously, studies often propose generic embedding learning strategies that are applied to various domains independently. The generated models in this challenge, which is being hosted by Kaggle in partnership with Google research and Google lens, are required to return a pertinent database that contains images from multiple domains to a given query image. The task is to determine not only the generic category of an object (e.g., an arch), but also the specific instance of the object (”Arc de Triomphe de l’Étoile, Paris, France”). In this project, we are using a pre-trained CLIP model to extract features from the image and an ArcFace loss function for training our model. The model trained and submitted (late submission) by us for the Kaggle competition scored 0.681, which is at par with the top 8 score amongst all competitors. We have trained the model on 17691 different instance categories.**

***Index Terms*—Instance level recognition, CLIP, ArcFace**

I. Introduction

Image representations are a critical building block of computer vision applications. Traditionally, research on image embedding learning has been conducted with a focus on per-domain models. Generally, papers propose generic embedding learning techniques which are applied to different domains separately rather than developing generic embedding models which could be applied to all domains combined. Instance Level Recognition (ILR) is tackled by training a deep learning model with a large set of images. Capturing features of all object domains in a single dataset and training a model that can distinguish between them is challenging. The work presented here is structured in a representation learning format to create a model that extracts feature embedding for the images. This generated model is submitted via Kaggle Notebooks to be verified on a held-out test set, performs a k-nearest-neighbors lookup, and scores the resulting embedding quality. Google research believes this multi-domain ILR is the key to real-world visual search applications, such as augmenting cultural exhibits in a museum, organizing photo collections, visual commerce, and more.

II. METHODS

A. Datasets

Since the task is to classify objects at the instance level, the corresponding datasets should select instance-level data. The competition provided categories of data used for evaluation but not the data, so building a suitable dataset was important. According to the challenge, the categories are - Products, Landmarks, Artwork, Memes, Illustrations, apparel, and Accessories. After collecting information from major open websites, the selected datasets are as follows.

- Imagenet - 1K classes [1]
- Products10K - 10K classes [2]
- Google Landmark Recognition 2021 - Top 7k class images [3]

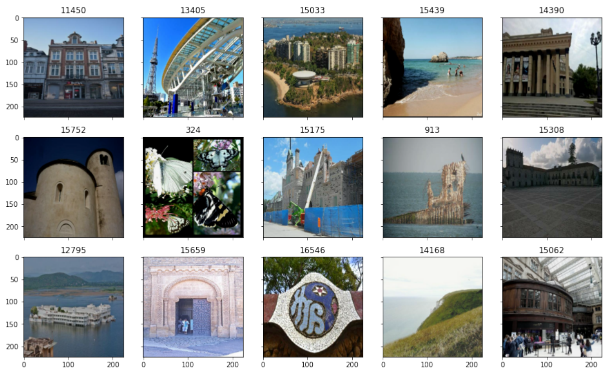


Fig. 1. A sample of images in Dataset

Fig 1 shows a sample of images present in the dataset. Datasets were in TFRecord format, a simple format for storing a sequence of binary records. We defined a deserializing function for obtaining the images and labels from the records.

A total of 17K classes of objects were used for training. To reduce the size of the dataset, only 50 images per class were taken into the dataset. The data was split such that 90 percent is used for training, while 10 percent is used for validation. So a total of 478185 images for training and 62828 images for validation.

B. Model Architecture

Fig 2 is the overview of our model architecture. The model is divided into two parts where one part will be used for training (Training model), and the other will be submitted as part of the embedding model requirement.

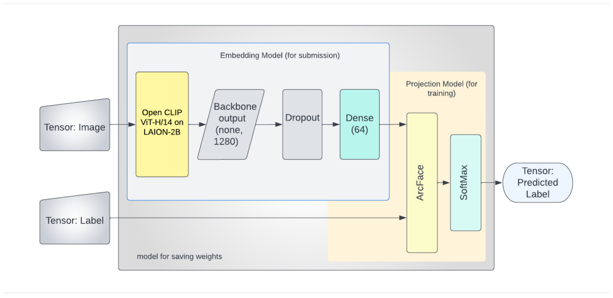

Embedding Model (for submission)
Projection Model (for training)
Tensor: Image
Open CLIP ViT-H/14 on LAION-2B
Backbone output (none, 1280)
Dropout
Dense (64)
ArcFace
SoftMax
Tensor: Predicted Label
Tensor: Label
model for saving weights

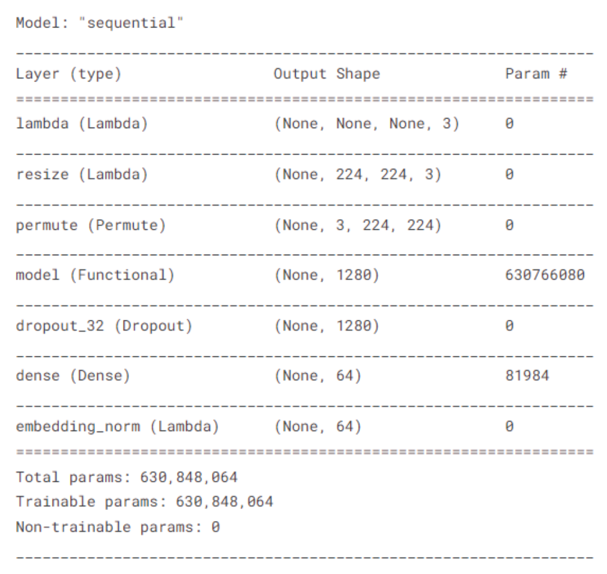
Fig. 2. Model Architecture

Training Model: Backbone(CLIP) + Dropout + Dense(64-D) + ArcFace + Softmax(17691 classes)

We used OpenCLIP-ViT-H-14-Visual [9] as the backbone for extracting features from the images to obtain a strong baseline [7]. This openCLIP implementation of the CLIP model was pre-trained on a two-billion-scale image-text pairs dataset LAION-2B. A dropout layer followed this with 0.2 dropout rate and a fully connected dense layer with 64 dimensions output as the neck. The ArcFace [6] and Softmax layer were stacked as the head where ArcFace improvised the Softmax functionality.

Embedding Model : Backbone(CLIP) + Dropout + Dense(64-D) + L2 Norm

This is the submission model as part of the Kaggle competition. Since the competition did not include classification as a requirement, but instead requires an image embedding model, we remove the ArcFace and Softmax layer and add an L2 Norm layer for normalization of the embedding vector after the dense layer. Normalization in the training is happening in the ArcFace layer.



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lambda (Lambda)              (None, None, None, 3)     0
_________________________________________________________________
resize (Lambda)              (None, 224, 224, 3)       0
_________________________________________________________________
permute (Permute)            (None, 3, 224, 224)       0
_________________________________________________________________
model (Functional)           (None, 1280)              630766080
_________________________________________________________________
dropout_32 (Dropout)         (None, 1280)              0
_________________________________________________________________
dense (Dense)                (None, 64)                81984
_________________________________________________________________
embedding_norm (Lambda)      (None, 64)                0
=================================================================
Total params: 630,848,064
Trainable params: 630,848,064
Non-trainable params: 0
_________________________________________________________________
```

Fig. 3. Embedded Model Summary

C. *CLIP Model - Backbone*

Computer vision models, in particular, perform well in specific tasks but often fail to generalize to tasks for which they have not been trained. OpenAI's CLIP (Contrastive Language-Image Pre-Training) closes this gap by reframing the problem and using contrastive pre-training. Instead of predicting label text, CLIP is trained to predict how likely the image corresponds to the text.

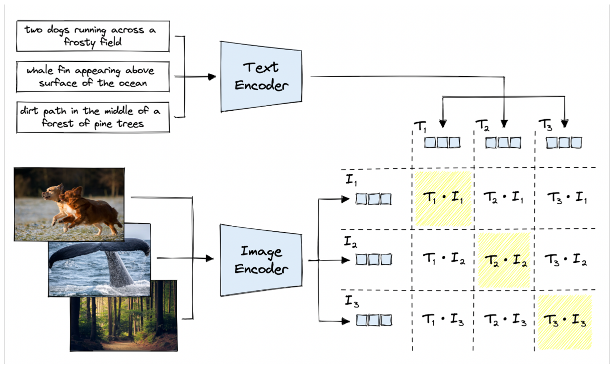

two dogs running across a frosty field
whale fin appearing above surface of the ocean
dirt path in the middle of a forest of pine trees
Text Encoder
Image Encoder
T1
T2
T3
I1
I2
I3
T1·I1
T2·I1
T3·I1
T1·I2
T2·I2
T3·I2
T1·I3
T2·I3
T3·I3

Fig. 4. Contrastive pretraining with CLIP. [10]

In the pre-trained model, which is in contrast to typical image models that simultaneously train an image feature extractor and a linear classifier to predict a label, CLIP jointly trains an image encoder and a text encoder to predict the correct pairings among a batch of (image,text) training data. We split an image into patches and provide the sequence of linear embeddings of these patches as input to the model. To do this, CLIP learns a multi-modal embedding space by jointly training an image encoder and text encoder to maximize the cosine similarity of the image and text embeddings of the N real pairs in the batch while minimizing the cosine similarity of the embeddings of the $N^2 - N$ incorrect pairings and optimize a symmetric cross-entropy loss over these similarity scores.

Zero-Shot Learning is a machine learning paradigm in which test data from classes that were not used during training are evaluated using a pre-trained model. In other words, a model must be able to include new categories without any prior semantic knowledge.Due to 'Zero-Shot' capabilities, CLIP models can be applied to nearly arbitrary visual classification tasks. For a strong baseline, we use the transformer based OpenClip-ViT-H-14.

In our project, we are only using the image encoder part of the CLIP model, and it is used to get an embedding vector for the image. We used a dropout layer to eliminate the possibility of overfitting and a 64-D dense layer to get a vector of 64-D for each image in the dataset.

D. *ArcFace Layer*

The end of a typical classification network is often where SoftMax and Categorical Cross-Entropy loss are applied. Numbers are converted into probabilities using SoftMax.

SoftMax's disadvantage is that it doesn't create a safety margin, which causes the borders to be a little blurry. When

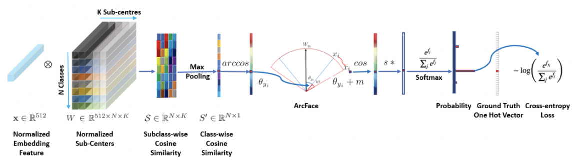

K Sub-centres
N Classes
Max Pooling
$arccos$
θ_{y_i}
W_{y_i}
x_i
$\theta_{y_i}+m$
ArcFace
cos
$s*$
$\frac{e^{f_j}}{\sum_j e^{f_j}}$
Softmax
$-\log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right)$
$\mathbf{x} \in \mathbb{R}^{512}$
Normalized Embedding Feature
$W \in \mathbb{R}^{512\times N\times K}$
Normalized Sub-Centers
$\mathcal{S} \in \mathbb{R}^{N\times K}$
Subclass-wise Cosine Similarity
$S' \in \mathbb{R}^{N\times 1}$
Class-wise Cosine Similarity
Probability
Ground Truth One Hot Vector
Cross-entropy Loss

Fig. 5. ArcFace Overview [6]

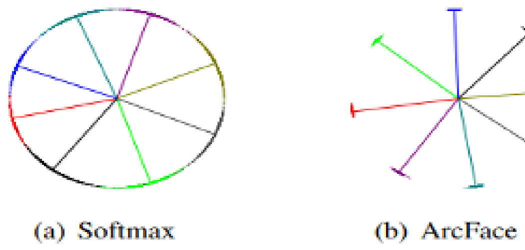


Fig. 6. A toy example using 8 classes and 2D features under Softmax and ArcFaceloss. Samples are indicated by dots, while the center direction of each class is shown by lines. Based on feature normalization, all facefeatures are moved to an arc space with a fixed radius. As the additive angular margin penalty is taken into account, the geodesic distance margin between the nearest classes becomes clear [6]

two images belong to the same class, we want their vectors to be as similar as possible, and when they belong to two separate classes, we want their vectors to be as diverse as possible. This means that, like SVM, we wish to generate a margin.

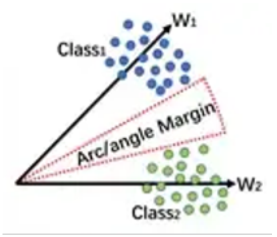


Fig. 7. Blue and green points represent embedding features from two different classes. ArcFace can directly impose angular (arc) margin between classes

ArcFace [6] estimates the geodesic distance on the hypersphere rather than using the euclidean distance.

The 64 D embedding vector which we obtained from the backbone model is normalized along with the weight vector of the layer ,$W \epsilon R^{dxn}$,where d is the dimension of the feature vector and n is total number of classes in the training set and calculated the angle between them. So we obtain a vector of, where each column is the angle between weight vector ,which represents a particular class, and the embedding vector the image.As we said earlier ,angular margin m is added to these angles and the cosine of the resulting angles is computed.The radius of the hypersphere s,is multiplied and the resultant $1Xn$ vector is passed to the softmax layer where probability of the image belonging to each class is calculated and that class which has the highest probability is predicted as the class of the image. Equations of the SoftMax and ArcFace is given here.

$$-\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{W_{y_i}^{T}x_i+b_{y_i}}}{\sum_{j=1}^{n}e^{W_j^{T}x_i+b_j}}$$

x_i denotes the deep feature of the i-th sample, belonging to the y_i-th class.
W_j^T denotes the j-th column of the weight W and b_j is the bias term.
The batch size and the class number are N and n, respectively.

Fig. 8. SoftMax Loss Function

$$-\frac{1}{N}\sum_{i=1}^{N}log\frac{e^{s*(\cos(\theta_{y_i}+m))}}{e^{s*(\cos(\theta_{y_i}+m))}+\sum_{j=1,j\neq y_i}^{n}e^{s*cos\theta_j}}$$

where θ_j is the angle between the weight W_j and the feature x_i
s - feature scale, the hypersphere radius
m - angular margin penalty

Fig. 9. ArcFace Loss Function

As we can see that the only difference between the two loss functions is the logit, i.e., the power of the SoftMax, and the margin is added to ground truth. Normalizing features and weights of the layer is essential. It forces the prediction to be dependent only on the angle; therefore, the embeddings are distributed on the hypersphere within a radius of ’s.’

III. Implementation

A. Configuration

We used Kaggle Notebook for training the model. All required computing resources were made available by Kaggle in the notebook. We used TPU V3-8, having 8 cores and 16 GB ram, for training our model. Access to datasets was available in the Kaggle environment. We built the model using Tensorflow, and it took about 3 hours to train the model in the environment for 10 epochs. Only The embedded model with its weights was submitted to Kaggle for scoring. We also simulated the performance of our embedding model by creating a dataset of 150 random images from various classes and performing a k-nearest-neighbors lookup.

B. Hyperparameters Tuning

After predicting the class, categorical cross-entropy is calculated. We used Adam as the optimizer for the model. The Batch size was set at 1600, with 200 samples for each TPU core. We did 10 epochs for training our model.

We used a learning rate scheduler for each epoch: a Linear Warmup With Linear Decay. It is a learning rate schedule in which we increase the learning rate linearly for updates for the initial few epochs and then linearly decay afterward.

This approach to learning rate reduces volatility in the early stages of training. The additive margin,m, and radius of the hypersphere ’s’ is set at 0.3 and 30, respectively. At the end of 10 epochs, the model with the weights having the highest absolute accuracy of the validation set is saved.

C. Simulation

To simulate the image retrieval that the competition will perform on the privately held data, we developed a dataset of

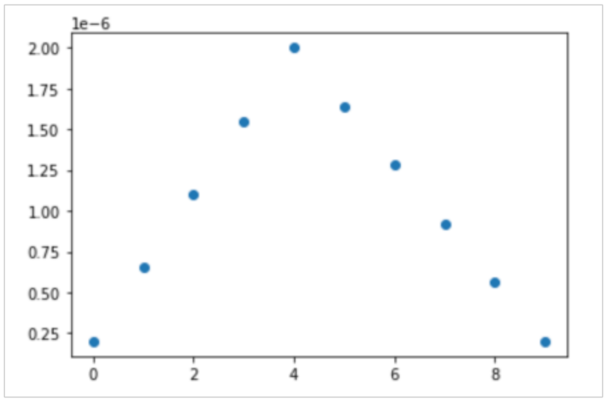


Fig. 10. ArcFace Loss Function

150 random images with following categories for simulation purpose.

TABLE I
IMAGE CATEGORIES FOR SIMULATION DATASET

Bulldog	Truck	Pizza	Tower	Shirt
German shepherd	Car	Cake	Statue	Dress
Labrador	Plane	Burger	Arch	Pant

We obtained embeddings for all these database images and used a separate image for each category as test image. We used Euclidean distance metrics, on the embeddings vector, to determine the similarity of the dataset images to the given test image.

IV. RESULTS

A. Loss and Accuracy

To evaluate the model at each epoch, cross entropy loss is used. Since the model deals with multi class classification, Sparse Categorical Accuracy and Sparse top K categorical accuracy was used as a metric during model training and the plots for 10 epochs can be found in this paper below. As we can see from these plots, model loss is decreasing and the accuracy of the model is increasing after each epoch during the training. As we can see from Fig 11, we got a "double-descent" performance curve instead of the U-shaped bias–variance trade-off curve.

B. Kaggle Competition Results

The evaluation was based on a retrieval task on a dataset of 5,000 test query images and 200,000 index images, from which similar images are retrieved. In contrast to ImageNet, which includes categorical labels, the images in this dataset are labeled at the instance level [4].

Submissions are evaluated according to the mean Precision@5 metric, introducing a small modification to avoid penalizing queries with fewer than 5 expected index images. In detail, the metric is computed as follows:

$$mP@5 = \frac{1}{Q}\sum_{q=1}^{Q}\frac{1}{min(n_q,5)}\sum_{j=1}^{min(n_q,5)} rel_q j$$

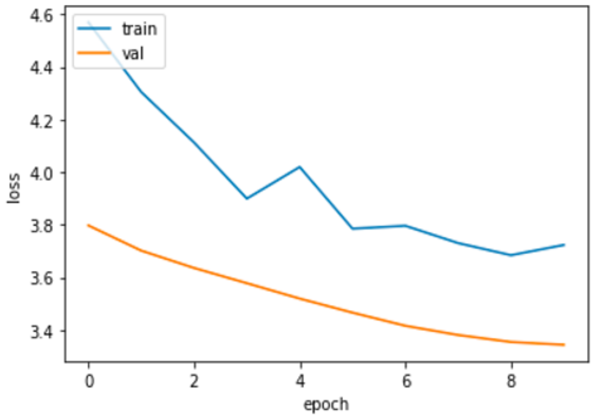


Fig. 11. Cross Entropy Loss

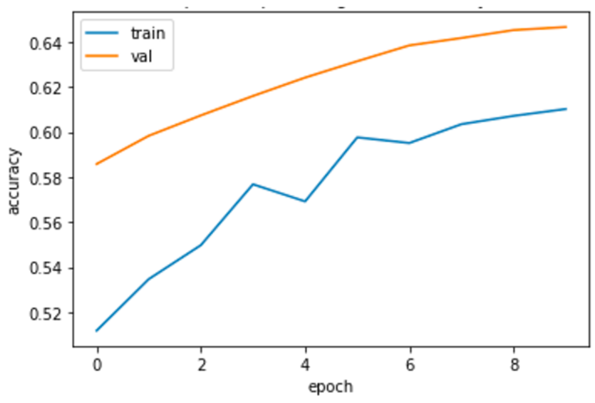


Fig. 12. Sparse Top K Categorical Accuracy

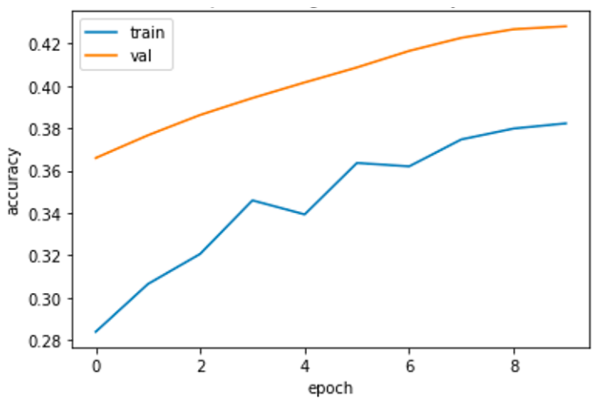
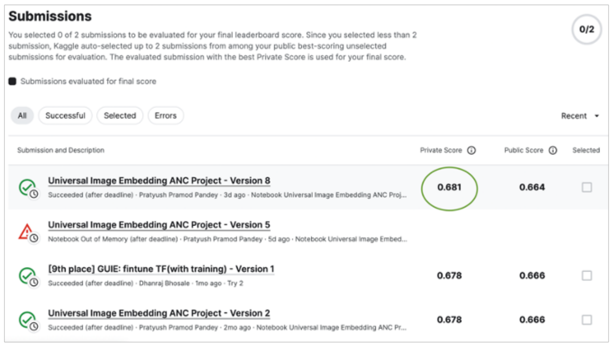


Fig. 13. Sparse Categorical Accuracy

where, Q is the number of query images, n_q is the number of index images containing an object in common with the query image, and that $n_q > 0$.

$rel_q(j)$ denotes the relevance of prediction j for the q-th query: it's 1 if the j-th prediction is correct, and 0 otherwise.

The model trained and submitted by us for the competition scored 0.681 which is at par with the top 10 score amongst all competitors.



Submissions

You selected 0 of 2 submissions to be evaluated for your final leaderboard score. Since you selected less than 2 submission, Kaggle auto-selected up to 2 submissions from among your public best-scoring unselected submissions for evaluation. The evaluated submission with the best Private Score is used for your final score.

0/2

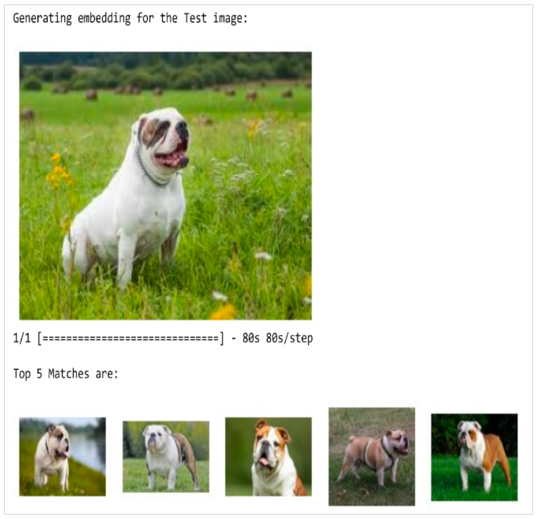
■ Submissions evaluated for final score

Submission and Description	Private Score	Public Score	Selected
Universal Image Embedding ANC Project - Version 8 Succeeded (after deadline) · Pratyush Pramod Pandey · 3d ago · Notebook Universal Image Embedding ANC Proj...	0.681	0.664	☐
Universal Image Embedding ANC Project - Version 5 Notebook Out of Memory (after deadline) · Pratyush Pramod Pandey · 5d ago · Notebook Universal Image Embed...			
[9th place] GUIE: fintune TF(with training) - Version 1 Succeeded (after deadline) · Dhanraj Bhosale · 1mo ago · Try 2	0.678	0.666	☐
Universal Image Embedding ANC Project - Version 2 Succeeded (after deadline) · Pratyush Pramod Pandey · 2mo ago · Notebook Universal Image Embedding ANC Pr...	0.678	0.666	☐

Fig. 14. Kaggle Submission Score

C. Simulation Result

For the simulation, we used a dog image from the dataset we created as the query image. As we can see (Fig 15 and 16), we were able to retrieve images of a dog and the top 5 images of the dog of the same breed.

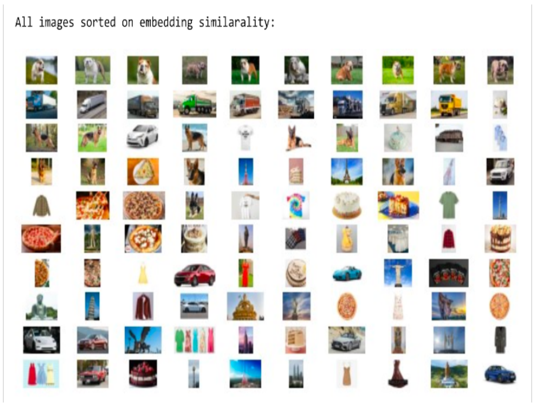


```
Generating embedding for the Test image:

1/1 [==============================] - 80s 80s/step

Top 5 Matches are:
```

Fig. 15. Simulation Result



```
All images sorted on embedding similarality:
```

Fig. 16. Images in our Simulation Dataset

V. Discussions and Conclusion

A. Learnings

We obtained good results when we did the simulation of the embedding model. The top 5 images extracted from the dataset always matched the test image category. Due to the CLIP model's zero-shot capabilities, we obtained similar images of a test image whose class was not included in the training or validation dataset.

This project helps us to understand how a transformer model can be used in a computer vision application. We could apply the theory studied in the course for example fine-tuning the model's hyperparameters by using the validation accuracy and how variable learning rates can increase the speed of training the model.

During our literature review, we had considerable exposure to several academic articles on Transformers, self-attention, ArcFace, and CLIP that helped us grasp the model we obtained from Kaggle. The team members were new to the concept of the dataset that was saved in binary format prior to this.

We could only comprehend how to extract the images from this binary format in the project's early stages. We fully understood how the model works and how the ArcFace layer improves the tuning of the weights after conducting the literature review. Additionally, we learned how TPUs, as opposed to GPUs, might increase training speed.

B. Future Enhancements

This model can be further improved to take care of tricky cases like identifying the difference between an actual Eiffel tower and a souvenir or replica of the Eiffel tower. Increasing the classification categories beyond the current 17K classes might make the embeddings even more universal. This improvement requires training with additional datasets on a more significant number of instances. New approaches for model training can be used. E.g. Continuous training may be deployed in applications where images are continuously processed, like Google lens.

VI. Statement of Contributions

A. Sumant Kulkarni [illegible] [illegible] [illegible]

- Proposed the idea of participating in the Kaggle competition
- Participated in code walkthrough and develop understanding of the reference implementation
- Built the simulation dataset
- Coded the simulation program
- Contributed to development of status check reports
- Contributed to development of the final presentation
- Contributed to writing the group project report

B. Pratyush Pandey [illegible] [illegible] [illegible]

- Understand the problem statement and research on different implementation to obtain an optimized model
- Rework on the implemented code to duplicate the work in our Kaggle environment
- Tweak the epochs, hyperparameters, and learning rate scheduler to improve the model performance
- Research on different implementation of the CLIP model and obtain a strong baseline model with the highest performance
- Understand the use of CLIP model and Vision Transformer in our work
- Contribute to the development of project presentation and report

C. Dhanraj Bhosale [illegible] [illegible] [illegible]

- Worked on understanding and duplicating original code at the initial stage and obtained the model
- Successfully submitted the first working code on the Kaggle and obtained the score
- Participated in code walkthrough and develop an understanding of the reference implementation
- Contributed to the development of status check reports
- Contributed to the development of the final presentation
- Contributed to writing the group project report

D. Shayal Shamsu [illegible] [illegible] [illegible]

- Participated in code walkthrough and develop understanding of the reference implementation
- Reviewed research papers regarding Arcface and usage of Transformers in computer vision application
- Contributed to development of status check reports
- Contributed to development of the final presentation-slides regarding ArcFace,model configuration and implementation
- Contributed to writing the group project report-sections regarding ArcFace , the implementation of the model and Discussions

References

[1] Imagenet Dataset - https://www.image-net.org/index.php

[2] Products10K Dataset - https://products-10k.github.io

[3] Google Landmark Recognition 2021 Dataset - https://www.kaggle.com/competitions/landmarkrecognition-2021/data

[4] Google AI Blog - Introducing the Google Universal Image Embedding Challenge, August 4, 2022, Posted by Bingyi Cao and Mário Lipovský, Software Engineer, Google Lens, https://ai.googleblog.com/2022/08/introducing-google-universal-image.html

[5] R. Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev., in press.

[6] J. Deng, J. Guo, N. Xue and S. Zafeiriou, "ArcFace: Additive Angular Margin Loss for Deep Face Recognition," 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 4685-4694, doi: 10.1109/CVPR.2019.00482.

[7] Baseline model implementation for the Kaggle universal image embedding - https://github.com/google-research/googleresearch/tree/master/universal_embedding_challenge.

[8] Training data-efficient image transformers distillation through attention - https://arxiv.org/pdf/2012.12877.pdf

[9] https://huggingface.co/laion/CLIP-ViT-H-14-laion2B-s32B-b79K

[10] Multi-modal ML with OpenAI's CLIP https://www.pinecone.io/learn/clip/

[11] Refernce Code notebook for implementation - https://www.kaggle.com/code/akihirok/9th-place-guie-fintune-tf-clip-with-training