Analysis of the Google Landmark Competition 2021

Comp 4730: Machine Learning

December 12, 2022

University of Windsor

Ali Naqvi, Noah Adams, Sergio Oliveira

Abstract

In this paper, we explore the Google Landmark Recognition 2021 Competition and its various solutions. The analysis of the top submissions and the implementation of our own, resulted in the demonstration of the superiority of transformer-based architecture over Convolutional Neural Networks in very large datasets. Additionally, the difference in the structure of the winning pipeline and the runners-up are seen. This happens where conclusions on why the competitors decided to choose the models they did and the structure leading up to the neck can be made. Furthermore, the importance of ArcFace margin loss and the difference in its variant, sub-center ArcFace is emphasized when analyzing the top submissions. Finally, our approach to using a Shifted Window Transformer model resulted in an accuracy of 0.3325 for our most optimal implementation. This implementation is consistent with many of the top performers and even though it was limited, the transformer-based model still performs reasonably well in testing compared to other architectures.

Introduction

As machine learning algorithms have speedily improved, new and more challenging problems have been needed to test them. One of which is presented by the Google Landmark Dataset V2 (GLDv2). In the Google Landmark Recognition 2021 competition [4], the contestants are given the task of image recognition where a model must be built that recognizes the correct landmarks in GLDv2. This dataset is the biggest landmark dataset where it contains two-hundred thousand distinct location classes used to sort five million distinct images sourced from Wikimedia Commons. The vast number of categories makes the dataset tricky to sort. Furthermore, each class contains a wide variety of images, many of which are exclusive to testing, making the GLD even more difficult [1]. In this paper, we will explore the top submissions made for this competition and what distinguishes the winning submission from the rest. We will also look at the same top submissions' similarities in implementation. Finally, we will create a model so we can compare and evaluate the features necessary in this competition with the rest of the submissions.

Relevant Literature Review

When looking at the top 3 solutions submitted for the Google Landmark Recognition 2021 Competition [4], we can see similarities between these submissions that differentiated them from the rest.

To begin recognizing the correct landmarks in a dataset, the pipeline must have a validation strategy. All top 3 solutions used the same local validation as the previous year's winner's solution [11]. This means leveraging the landmark samples from the 2019 test set as the validation set. However, the top 3 solutions varied in their implementation strategy. The 3rd place submission [7] did not change the strategy of the previous year's winner's and kept the same implementation. Whereas the 2nd place submission [6] changed the index image set from being only a part of the GLDv2x (all images from GLDV2 belong to 81313 landmarks) to the full GLDv2x. Furthermore, the winning submission [5] only considered index images that were matches of any query image to reduce the computation time for evaluation. Because of this, their submission achieved good correlation between the local validation and leadership.

The competitors then each began with the extraction of the embeddings of input images from various backbone models. When comparing the backbones of each of the top submissions used for their models, a significant difference can be seen between the winning submission and the others. The 3rd place used convolutional neural network (CNN)-based: EfficientNet B5, B6, B7, and V2. They also used transformer-based: Swin-L-384, and a hybrid-based CvT-W24-384. The 2nd place submission [6], decided to use SWIN, CSWIN, and EfficientNet B7 to add diversity. Both 2nd and 3rd placers reached the conclusion that transformer-based architectures yield much better results than CNN based architectures. However, despite the 3rd place submission drawing the conclusion that Swin achieved the best retrieval performance, they kept CNN-based models. Their reasoning for this is that when comparing the model backbones, the greater the difference in model structure, the greater the complementarity of performance in the fusion stage. Conversely, the winning submission took another direction and presented two architectures that conceptually share a large part of an EfficientNet-based CNN encoder. The first is a model with a deep orthogonal fusion of local and global features (DOLG) using an EfficientNet backbone and a novel Hybrid-Swin-Transformer. The latter was created by appending a vision transformer to a CNN-encoder resulting in a Hybrid-CNN Transformer model.

Where the 3rd place submission has a solution that majority of the other submissions followed, the 1st place solution differentiates from it substantially.

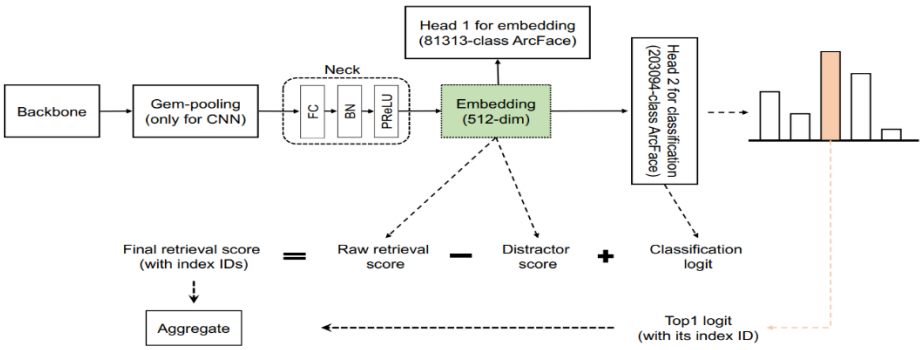


Figure 1: Model structure of 3rd place submission [7]

From Figure 1, the model structure that can be seen is a common structure followed by efficient submissions from the competition. The backbone outputs are aggregated through a Generalized-Mean (GeM) pooling layer. However, gem pooling is only used for CNN as each token output feature is the global representation because of the self-attention mechanism. It is then fed into an embedding neck layer which consists of Linear(512)+1D+BN+PReLU. The embeddings are then used to classify specific landmarks supervised.

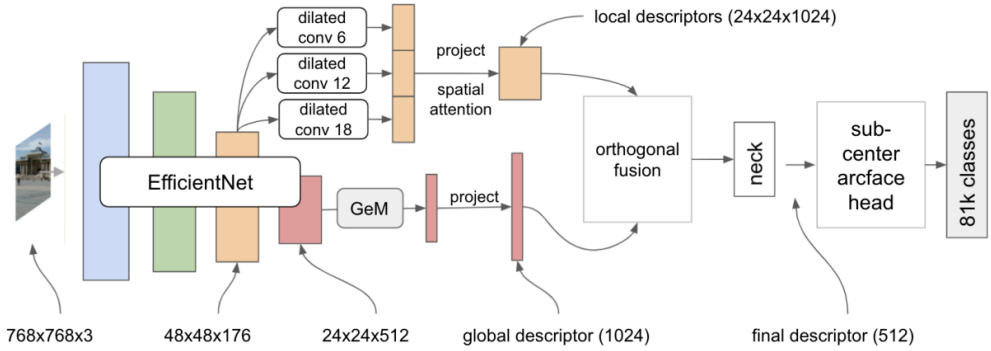
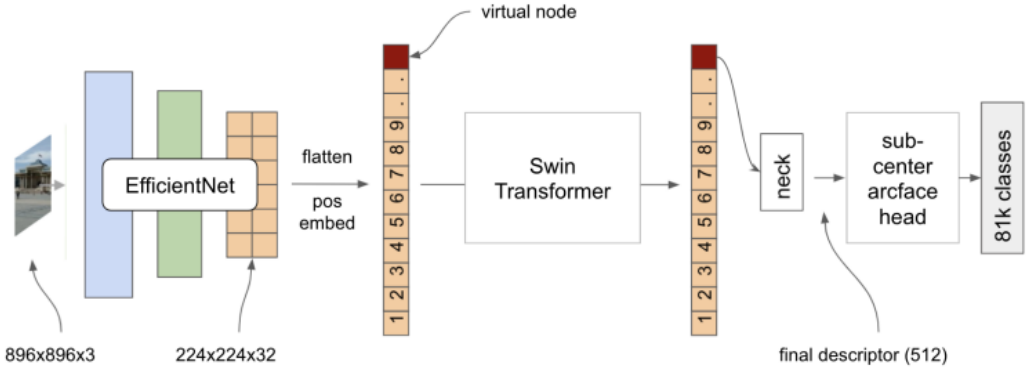


Figure 2: Model architecture DOLG-EfficientNet-B5 of 1st place position [5]

In Figure 2, we can see the implementation of the EfficientNet backbone, After the backbone, the first-place solution uses 3 dilated convolutions [10] with differing dilation parameters per model. The concatenation happens in the feature dimensions and is self-attended using spatial-2d-



attention. Following this, the local features are then fused orthogonally with the global feature vector coming from the GeM pooling. Furthermore, the fused features are aggregated using average pooling. For the neck, their final solution uses FC and BN and PReLU like the 3rd place solution.

Figure 3: Hybrid-Swin-Transformer, exemplary shown for EfficientNet-B5-Swin-Base224 [5]

Following the previous backbone, the 1st place submission also had a different model architecture for their Hybrid-Swin transformer as can be seen in Figure 3. Their submission leverages recent advances in using transformers for computer vision problems. They created the Hybrid-CNN-Transformer model, they append a vision transformer to a CNN-encoder. The Swin-Transformer is used because of its flexibility at various scales. Following this, the structure of the neck is the same as the previous architecture.

Looking at the similarities that the top submissions had, we can see all of them decided to use ArcFace loss function instead of SoftMax loss function for training the models. ArcFace loss is superior to SoftMax loss in certain situations because it can better handle cases where there are a large number of classes or where the data is highly imbalanced. This is because ArcFace loss uses a margin to separate classes, which allows it to better handle situations where the data is not linearly separable. Additionally, ArcFace loss can also incorporate additional information, such as the angle between feature vectors, which can improve the performance of the model. In this particular image recognition competition, the top submissions all used ArcFace Margin loss for the training of the models. However, what separated the first-place submission, and the rest is the use of a sub-center ArcFace head for training with dynamic margins. Sub-center ArcFace is a method for training deep learning models that are used for facial recognition tasks. It is a variant of the ArcFace method. The sub-center variant is designed to improve the performance of the model by using additional information about the location of facial features within an image. This can help the model better distinguish between faces and improve its overall accuracy. Looking at the paper [12] which introduced the dominance of sub-center ArcFace, we can see that margin-based deep face recognition methods have achieved great success in unconstrained face recognition. The intra-class constraint of ArcFace is relaxed to improve the robustness to label noise.

Finally, the 1st place model used an ensemble of three DOLG models and Hybrid-Swin-Transformer models with differently adjusted Efficient-net backbones and input image sizes. The winning submission also trained two pure EfficientNet models where the models are trained on the full GLDv2. Whereas the 2nd place uses 4 models in their final ensemble. These consist of SwinBase, SwinLarge, CSWinLarge, and EfficientNet B7. To add diversity, they extracted features using different image sizes and many Test Time Augmentation (TTA). TTA is an application of data augmentation to the test dataset and can be used to make better predictions. The final ensemble for the 2nd place submission is complete with a total of 11 types of features used. On the other hand, the 3rd place does not add more models, and to blend each which they previously used, they l2-normalized each individually, concatenated them, and again applied l2-normalized to conduct ensembled retrieval. L2 normalization is used to normalize feature vectors

so that they can be compared on an equal basis. The 1st place submission similarly used l2-normalization for their ensemble retrieval.

Experimental Setup and Methodology

To determine our experimental model's efficiency, we compared our submission results with the top competitors of the Kaggle competition. Using the Google Landmark Recognition 2021's dataset, we developed our model through the Kaggle website for consistency across our members' platforms and due to the 105Gb dataset size. Our environmental setup uses Kaggle's GPU T4 x2 accelerator for better training performance and python as the language of choice.

In order to process the dataset, the data was preprocessed into TensorFlow DataFrame to allow for parallel caching when processing another image in the epoch. Images were processed such that the image dimensions are normalized to 256x256 with 3 channels. These images were then set up with their respective labels for training and testing if available, and then packaged into DataFrames. Images are then shuffled and later manipulated in training with random flips and cropping.

The model we chose to develop, and run is a Shifted Window (Swin) Transformer based of the Keras implementation with modification. Similar to other Transformers, the model uses a transformer encoding section with multi-headed self-attention and uses Vision Transformer's (ViT) patch embedding system. The patch embedding system is similar to that of ViT's where an image is divided into several set patch sizes that are flattened to be used for the multi-headed attention layer. The key difference in the Swin Transformer that helps it maintain linear time opposed to quadratic is that it first embeds the patches with smaller initial sizes that are merged at deeper layers of the transformer. These embeddings are computed using self-attention only within local windows rather than in global space and the outputs are afterwards merged. These windows are not fixed and instead are shifted with respect to the previous layers. The process repeats itself for the number of layers of the Transformer until no more patches can be constructed.

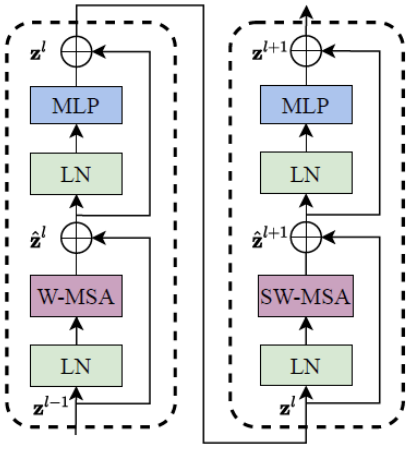


Figure 4: Two blocks of Shifted Window Transformer (Liu et al.)[3]

The model architecture consists of passing input through randomized cropping and flipping before patch encoding begins. The randomly modified images pass through patch encoding by first extracting the patches and then embedding the patches in initial 4 x 4 patches with 64 dimensions. The next step is two blocks of the Swin Transformers that each apply the same patch dimensions and with an 8 x 8 window size that is shifted only in the second block. The transformers themselves apply a sequential MLP layer that consists of several dense layers followed by a Gaussian Error Linear Unit (GELU) activation – a form of activation that combines ReLU and dropout regularization qualities -, and then several dropout and dense layers again. The transformer patches are then merged and a 1D global average pooling is used to extract classifications from the feature map. The model uses the Adam optimizer with a learning rate of 0.001 and a weighted decay of 0.0001. Lastly the loss is calculated using a sparse categorical cross entropy and evaluated using an accuracy and top 5 accuracy.

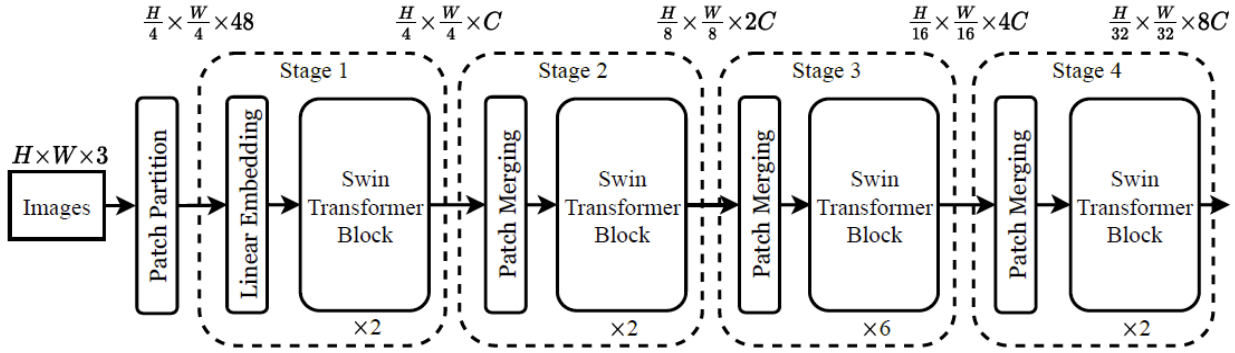


Figure 5: SWIN Architecture (Liu et al.)[3]

Training of the model used GLR 2021's training dataset that consists of 1.5 million images and over 81,000 classification labels. Even though the images were reduced to 256x256x3, to stay within the competition's timeframe, we trained the model using multiple sized datasets and number of epochs to determine optimal accuracy and max training time. We split the datasets into dataset sizes of 200,000, 100,000, and 50,000 of the 1.5mil images with epoch sizes of 20, 30, and 40 respectively. The datasets were each given a 25% validation split and consisted of batch sizes of 32. After individually training the datasets, we used the optimal model to evaluate the test dataset for submission.

Discussion & Results

After training the model on separate dataset sizes, we found that the optimal accuracy was achieved when using 50,000 images with 40 epochs. The model was able to achieve an average accuracy of 0.3325 and a validation accuracy of 0.2771. Comparatively, the 1st, 2nd, and 3rd place submissions were able to achieve accuracies of 0.51272, 0. 49076, and 0.48962 respectively. Across all 383 teams this would rank the current submission amongst the top 30 submissions.

Epochs/Dataset	Loss	Accuracy	Top 5 Accuracy	Validation Loss	Validation Accuracy	Validation Top 5 Accuracy
40/50,000	3.4092	0.3325	0.5574	4.2932	0.2771	0.4633
30/100,000	3.8798	0.2816	0.4911	4.7026	0.2305	0.4030
20/200,000	5.0154	0.1826	0.3412	5.5032	0.1695	0.3091

The model was able to achieve comparatively good accuracies in part due to Swin Transformers efficiency with large datasets. The transformer's ability to run linearly with respect to the patches allows it to achieve great performances with little accuracy loss when self-attention is done on local rather than global scales.

Though the model's performance was better than others, the implementation of this model is subject to overfitting. The GLD has over 81,000 labels with many of the labels only being used a single time which means when the model was forced to train on divisions of 50,000 random images, it never is statistically exposed to many of the available labels. As such the model will over fit due to the 50,000 available labels it trains on. This is supported by the model's validation accuracy which as the dataset size decreased, so did the gap between mean accuracy and validation accuracy increase.

Conclusion & Future Work

In this paper we discussed how the problem proposed by the GLD helps facilitate the advancement of machine learning algorithms. Images in the GLD are made difficult to identify by its large amount of input data, distinct categories and images not included in the training data. Self-attention works better on large datasets like the GLD, propelling transformer-based architectures past CNNs in their results.

Aside from using transformer-based architecture, the top three models had many similarities in their implementation. They all used images from the 2019 GLD test set as part of their validation sets. The top model differed from the others by excluding index images that did not match any query images, shortening the computation time. All three submissions also used multiple backbone models to extract embeddings, the top submission, however, differed by using two different architectures sharing an encoder. Finally, they all used ArcFace as their loss function in place of SoftMax. The top model varied by using sub-center ArcFace in place of the standard function.

We compared our model's efficiency to that of the top models through the Kaggle website. We first preprocessed the data into TensorFlow DataFrame before developing a Shifted Window (Swin) Transformer. We chose this transformer type with the aim of maintaining linear time through embedding its patches in deeper layers of the transformer with reduced image sizes. Input images were randomly modified by our model. Patches were then extracted from the images and then given to the transformers which applied a sequential MLP layer. We used 1D average pooling, the Adam Optimizer and sparse categorical cross entropy.

We used 50 000 images in our model with 40 epochs. We achieved an average accuracy of 33.25% which is in the 92nd percentile of submissions, with the best model being 51.272%. Our accuracy was bolstered by the Swin transformer we used allowing local self-attention which lowers accuracy loss. The largest issue we had is that our model was quite computationally expensive, requiring us to use a relatively small image set for training. This in turn caused overfitting.

Going forward we aim to further improve our model through the implementation of several new features. Firstly, we want to train our model the same way as the top models by using GLD 2019 or 2020. We also want to use larger image sizes to improve our accuracy and lower the number of epochs to reduce overfitting. Lastly, we want to use a feature extractor to preprocess the image as this should make the model more efficient.

References

[1] Keras Team, "Image classification with Swin Transformers," *Keras*, Sep. 08, 2021. https://keras.io/examples/vision/swin_transformers/ (accessed Dec. 12, 2022).

[2] T. Weyand, A. Araujo, B. Cao, and J. Sim, "Google Landmarks Dataset v2 -- A Large-Scale Benchmark for Instance-Level Recognition and Retrieval," *arXiv*, Nov. 02, 2020. https://arxiv.org/abs/2004.01804 (accessed Dec. 12, 2022).

[3] Z. Liu *et al.*, "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows," *arXiv*, Aug. 17, 2021. https://arxiv.org/abs/2103.14030 (accessed Dec. 12, 2022).

[4] Google Landmark Recognition 2021 Competition. https://www.kaggle.com/ c/landmark-recognition-2021/ overview/iccv-2021.

[5] C. Henkel, "Efficient large-scale image retrieval with deep feature orthogonality and Hybrid-Swin-Transformers," *arXiv*, Oct. 27, 2021. https://arxiv.org/pdf/2110.03786.pdf (accessed Dec. 12, 2022).

[6] S. Dai, "2nd Place Solution to Google Landmark Recognition Competition 2021," *arXiv*. https://arxiv.org/pdf/2110.02638.pdf (accessed Dec. 12, 2022).

[7] C. Xu *et al.*, "3rd Place Solution to Google Landmark Recognition Competition 2021," *arXiv*, Oct. 07, 2021. https://arxiv.org/pdf/2110.02794.pdf (accessed Dec. 12, 2022).

[8] A. Arora, "GeM Pooling Explained with PyTorch Implementation and Introduction to Image Retrieval," *Committed towards better future*, Aug. 30, 2020. https://amaarora.github.io/2020/08/30/gempool.html (accessed Dec. 12, 2022).

[9] Y. Gu, "ATTENTION-AWARE GENERALIZED MEAN POOLING FOR IMAGE RETRIEVAL," *arXiv*. https://arxiv.org/pdf/1811.00202.pdf (accessed Dec. 12, 2022).

[10] S.-H. Tsang, "Review: DilatedNet — Dilated Convolution (Semantic Segmentation)," *Towards Data Science*, Nov. 17, 2018. https://towardsdatascience.com/review-dilated-convolution-semantic-segmentation-9d5a5bd768f5 (accessed Dec. 12, 2022).

[11] C. Henkel and P. Singer. Supporting large-scale image recognition with out-of-domain samples. CoRR, abs/2010.01650, 2020.

[12] J. Deng, J. Guo, T. Liu, M. Gong, and S. Zafeiriou, "Sub-center ArcFace: Boosting Face Recognition by Large-scale Noisy Web Faces,' ECVA, 2020. https://www.eva.net/papers/eccv_2020/papers_ECCV/papers/123560715.pdf (accessed Dec. 12, 2020)