Google Landmark Recognition

Nirav Patel
nkpatel8@ncsu.edu

Peeyush Taneja
ptaneja@ncsu.edu

Sneha Aradhey
svaradhe@ncsu.edu

I. MOTIVATION

Instance Level Recognition is a visual task of identifying the specific instance of an object, rather than simply recognising the object class. For example, a car is an object class, and Tesla Model X is an instance of the object car. Instance level recognition extends to various domains like landmark, retail products and artwork recognition. We aim to comprehend this concept in more depth by solving Google Landmark Recognition Challenge hosted on Kaggle platform [1].

Our objective is to build a model to recognize correct landmarks in images. The main challenge lies in the fact that we have a large number of classes and the number of training examples per class varies significantly (long tailed class distribution), which makes it a much harder problem than trivial image classification. During the course of this project we expect to get a firm grip of concepts like image embedding space, deep metric learning, ensemble models and more.

II. DATASET DESCRIPTION

We used a cleaned subset of Google landmarks dataset version 2 [3] for this project. The dataset contains annotation with labels representing human-made and natural landmarks. There are almost 1.5M images present in the dataset associated with 81k different landmarks as shown in figure 1.

Fig. 1. sample images

Because of the limitation on computation power, we used only the subset of the data and randomly selected 1k different landmarks as our training dataset which gives us nearly 18k images to work with. The original dataset is highly imbalanced i.e number of images associated with one class varies from one to thousands. To illustrate, a famous landmark as Eiffel tower has nearly 6000 images while a non famous landmark has only 2 images. While sampling. we have maintained the original distribution of the images per class to make our model train on dataset similar to the original dataset. The distribution is shown in figure 2. This class imbalance presents a challenge for the model to learn the data properly.

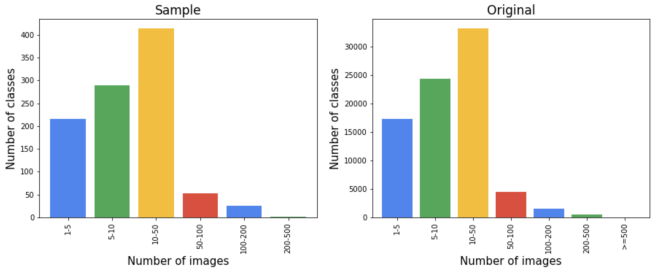

Fig. 2. Class distribution

Another challenge with the dataset is images are taken with different angle and exposure. For the images with same label, we have images from inside, outside, landscape, people in front of the structure as shown in figure 3 which makes this problem harder whether to classify the given image to a specific label or to non-landmark image.

Fig. 3. Images taken from different angles and exposure

III. METHODOLOGY

The basic idea behind the implementation of this project was to use explore the novel technique of metric learning and compare the results with a traditional loss functions. Unlike

cross entropy loss whose objective is to learn to predict directly a label from given input, metric loss predicts relative distance between inputs. This is often called Metric Learning and it is especially useful in cases where the number of classes are very large and training samples per class is limited, metric learning is further explained in the section IV

- Data Preprocessing - The images in the dataset are of different shapes and size. The ResNet model has been structured and works best for the input image size of 224*224*3. We thus pass our samples images through a preprocessing function to resize the image to the mentioned shape. Images for training and test dataset are resized, the model is then trained on these updated images using Image data generator.

- Data Augmentation - Given that the number of samples per class average around 18 for our problem dataset, data augmentation becomes a very important step. We did image augmentation with non-zero values for height and width shift, zoom, brightness, rotation and shear. We did not store the augmented images in our system, rather used them directly as an input to the model.

- Baseline Model - As a baseline model, we used a ResNet50 model pre-trained on Imagenet dataset as feature extractor to get the embeddings vectors. The weights for the last block of the ResNet model along with additional dense layers are retrained for our landmark images dataset using Transfer Learning which allows us to use the available trained model weights for general image classification. The model which is connected to a linear layer is then fed to a cross entropy loss function.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
resnet50 (Functional)        (None, 7, 7, 2048)        23587712
_________________________________________________________________
global_average_pooling2d (Gl (None, 2048)              0
_________________________________________________________________
flatten (Flatten)            (None, 2048)              0
_________________________________________________________________
batch_normalization (BatchNo (None, 2048)              8192
_________________________________________________________________
dense (Dense)                (None, 1024)              2098176
_________________________________________________________________
dropout (Dropout)            (None, 1024)              0
_________________________________________________________________
batch_normalization_1 (Batch (None, 1024)              4096
_________________________________________________________________
dense_1 (Dense)              (None, 512)               524800
_________________________________________________________________
batch_normalization_2 (Batch (None, 512)               2048
_________________________________________________________________
dense_2 (Dense)              (None, 943)               483759
=================================================================
Total params: 26,708,783
Trainable params: 18,089,903
Non-trainable params: 8,618,880
_________________________________________________________________
```

Fig. 4. Baseline Model

- Proposed Solution : We proposed on using a triplet loss based model with SE-ResNeXt101 as backbone architecture. Using pre-trained SE-ResNeXt101 as a global descriptor we obtained embedding representation of the input. These embeddings are passed through a linear layer which is trained using the triplet loss function. The triplet loss function forms cluster for different classes which makes the classification easier. To give the final class landmarks for images we have used KNN classifier.

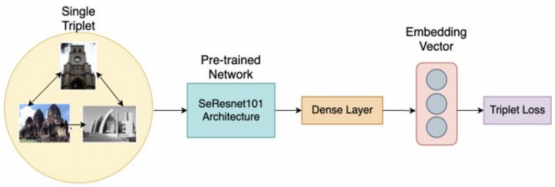

Fig. 5. Proposed Model

IV. MODEL SELECTION

- ResNet50 [4]: For our baseline model we have used resnet50 as our backbone model. Resnet short for residual networks which was winner of imagenet challange in 2015. The main advantage of this architecture is it allow to train very deep neural network such as 150+ successfully. Adding new layers in deep neural network simply does not work because of when the gradient is propagated to previous layers, repeated multtiplication make gradient very small (also known as vanishing gradient problem). Resnet architecture solves this issue with the idea of skip

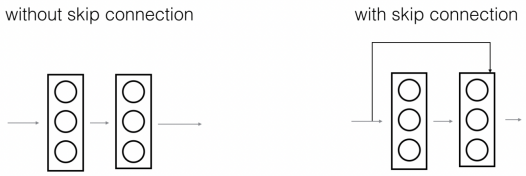

Fig. 6. resnet skip connection

connection. The neural network learns several features at the end of its layers. In residual learning, instead of trying to learn some features, we try to learn some residual. Residual can be simply understood as subtraction of feature learned from input of that layer. ResNet does this using shortcut connections (directly connecting input of nth layer to some (n+x)th layer. The skip connection works because it tackles vanishing gradient issue by allowing alternate path for gradient to flow through in back propagation. Addititionally, it also make sure that the higher layer will perform at least as good as the lower layer. And this is why it is very widely used as a backbone in many computer vision tasks.

- SE-ResNet [5]:
  In our proposed model, we have used seresnext101 as our backbone model. The central building block of convolutional neural networks (CNNs) is the convolution operator, which enables networks to construct informative features by fusing both spatial and channel-wise information within local receptive fields at each layer. SENet is build with the idea of "Squeeze-and-Excitation" (SE) block that adaptively recalibrates channel-wise feature

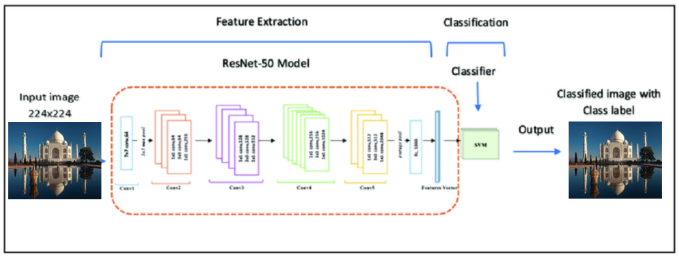

Fig. 7. Baseline Model

responses by explicitly modelling inter dependencies between channels.

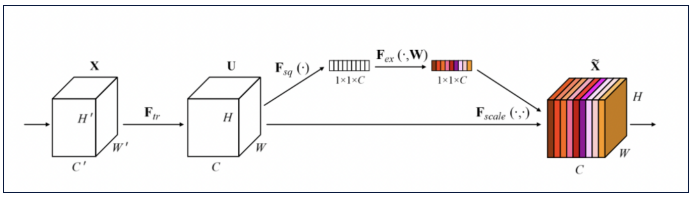

Fig. 8. SE block

Basically with the squeeze-and-excitation block, the neural nets are better able to map the channel dependency along with access to global information. As shown in figure 8 once convolution is applied on input image to get features, squeeze operation is performed to get a single value for each channel. The main idea behing squeeze operation is to extract global information from each of the channels of an image which convolution operation does not provide since it is a local operation. After we perform excitation operation on the outpur of squeeze operation, we get per channel weights. It is easy to integrate this SE block with other state of the art methods. This is shown in the figure 9.

- Triplet Loss Function [6]: The main idea of using triplet loss is to learn distributed embeddings representation of input data in a way that in the embedding space, data points belonging to the same class are projected in the near-by region whereas different class images are projected far away from each other. For Triplet Loss, the objective is to build triplets consisting of an anchor image, a positive image (which is similar to the anchor image), and a negative image (which is dissimilar to the anchor image). There are different ways to define similar and dissimilar images. If you have a dataset having multiple labels as the target class, Then images of the same class can be considered as similar, and images across different classes can be considered as dissimilar. The loss objective is that the distance between the anchor sample and the negative sample $d(e_a, e_n)$ is greater (and bigger than a margin m) than the distance between the anchor and positive representations $d(e_a, e_p)$. We can represent the loss with the below equation

$$L(e_a, e_b, e_c) = max(0, m + d(e_a, e_p) - d(e_a, e_n))]$$

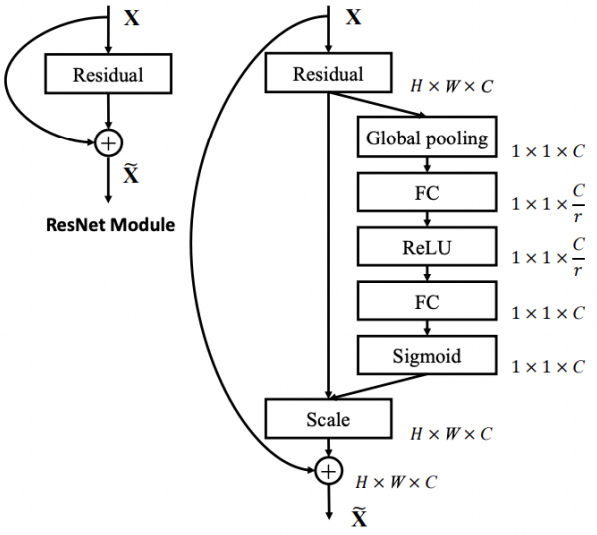

Fig. 9. seresnext module

Fig. 10. Baseline Model

V. EVALUATION

This section present results from the experiments which provide an extensive comparison between the baseline (ResNet and Cross Entropy) and our proposed model (SE-ResNetXt101 and Triplet Loss). Fig 7 shows the baseline model accuracy curve for training and validation set. It is clear from the graph that the baseline model starts overfitting around epoch 20 which is kind of what we expected. Fig 8 visualize the accuracy curve of our proposed model. We observe that validation curve closely follows the training accuracy curve which suggest that the model is not overfitting.

The table shown in Fig 10 present results in a quantative manner. It tells that triplet loss with SE-ResNeXt101 architecture performs much better than Cross entropy with ResNet in both F1 score and accuracy value. We also calculated the top5 accuracy which checks that whether the correct class is present in top 5 predictions by the model. It is used frequently in image search related task.

As the number of classes increases and sample per class decreases cross entropy fails to produce good results.

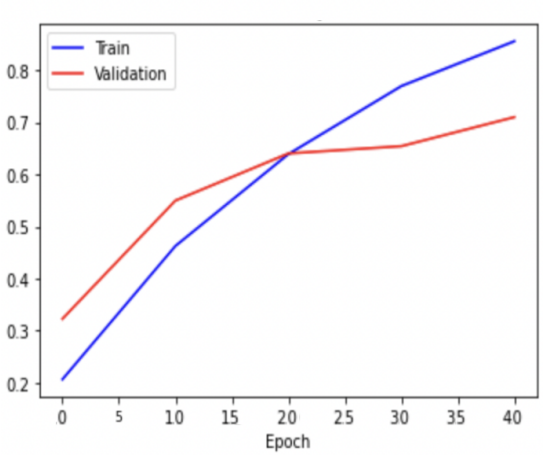

Fig. 11. Validation Accuracy curve for the baseline model. X axis represents number of epochs and Y represents accuracy

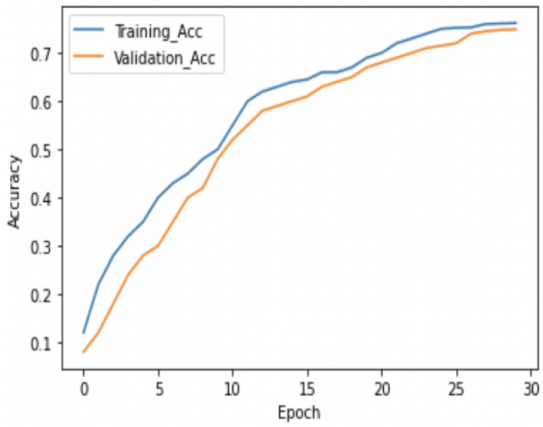

Fig. 12. Validation Accuracy curve for the proposed model. X axis represents number of epochs and Y represents accuracy

Validation	F1 Score Top1	Accuracy Score Top1	Accuracy Score Top5
Base Model	62.3%	65%	78%
Proposed Model	72%	75.8%	90.4%

Fig. 13. Result Comparison between Baseline model and Proposed model

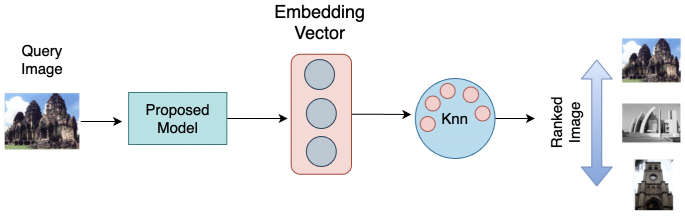

Fig. 14. Inference pipeline

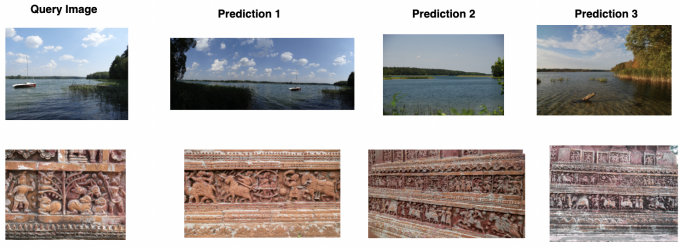

Fig. 15. Inference pipeline for the given query image where k=3

A. *Inference Pipeline*

We are going to briefly discuss the process of getting results from test images (query images). Whenever we get a query image to process, the first step is to get an embedding vector from the SE-ResNetXt101 model for the corresponding image. After that we input the query embedding vector into the KNN classifer. The search space for kNN classifer is build using the training images (reference images). The L2 distance between query embedding and reference embeddings are then used to get the K nearest neighbor. Fig provides example of thee inference pipeline. For this example we have the value of k = 3.

VI. Future Scope

We've implemented model which was proposed and from the results it is seen it performs better than the baseline model. As part of literature reading we explored some more loss functions used in metric learning and different backbone architecture. Due to time constraint we could not implement and test it. From the discussions on kaggle and other blog posts, arc loss can be replaced as a loss function instead of triplet loss. Also currently our backbone model only consists of SEResnext101. We want to try more variants of resnet,seresnet and efficientnet or rather ensemble of these models. Also currently we have trained our model on the subset of the data only. We would like to perform training of our model on the whole dataset given.

References

[1] Google Landmarks Recognition 2020 https://www.kaggle.com/c/landmark-recognition-2020/overview
[2] Christof Henkel and Philipp Singer: "Supporting large-scale image recognition with out-of-domain samples "
[3] Google Landmarks Dataset v2 https: //github.com/cvdfoundation/google-landmark.
[4] Jie Hu, Li Shen, Samuel Albanie, Gang Sun, Enhua Wu: "Squeeze-and-Excitation Networks"
[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun :"Deep Residual Learning for Image Recognition"
[6] Florian Schroff, Dmitry Kalenichenko, James Philbin: "FaceNet: A Unified Embedding for Face Recognition and Clustering"