Totally Looks Like

Hao Xu 1224853 Chaoyi Shi 1069250

October 2023

Abstract

Contrastive learning techniques have shown promising results in various machine learning tasks. This paper presents a novel Siamese neural network model that leverages the power of two diverse pre-trained models ResNet50 and VGG-19 to perform similarity learning tasks. The primary objective is to examine how the concatenation or averaging of features extracted by these base models impacts the overall performance. To this end, a custom contrastive loss function is implemented and optimized using the Adam optimizer.

Our experimental setup involves using TensorFlow for model construction and TPU-based training for scalability. The model is trained and validated on a custom dataset, and the training is facilitated through Model-Checkpoint to save the best model based on the validation accuracy. The results indicate that the proposed model architecture provides a balanced feature representation, achieving top-K categorical accuracy during validation. The codebase, saved models and experiment logs are available for further research and development.

1 INTRODUCTION

In the era of data-driven decision-making, deep learning methods have radically transformed various application domains, including natural language processing, healthcare, and, notably computer vision. As these techniques continue to advance, there is a burgeoning interest in developing specialized neural network architectures to address the unique challenges posed by specific tasks in these domains.

Among such specialized architectures, Siamese neural networks have emerged as a powerful solution for assessing the similarity between instances in a feature space [6], relevant for applications like face verification, anomaly detection, and document clustering. Recognizing the effectiveness of Siamese networks in capturing complex relationships, this work focuses on improving the network's ability to generalise by using feature extraction from multiple pre-trained models.

While Siamese networks have been the subject of extensive research, a relatively unexplored area is the integration of features from diverse neural network architectures. Existing works have typically utilized single pre-trained models for feature extraction, thereby limiting the range and diversity of features that can be captured.

This paper aims to enhance Siamese networks' efficiency by combining features from two different pre-trained models: ResNet50[4] and VGG19[9]. The central question addressed is whether this fusion of features can improve the network's performance in identifying similar images by utilizing the Totally Looks Like dataset introduced by (Rosenfeld et al., n.d.) in their paper [8]. Our results indicate a promising improvement in these areas.

(a) Similar Pair

(b) Dissimilar Pair

Figure 1: Randomly picked pairs from the Totally Looks Like dataset

The paper is structured as follows: related work, methodology, experiments, and finally results and discussions for future research avenues.

2 RELATED WORK

A key application of effective image similarity lies in the realm of image retrieval. Babenko [1] and colleagues demonstrate that fine-tuning a pre-trained Convolutional Neural Network (CNN) with domain-specific data can enhance its performance on related datasets. Similarly, Chandrasekhar et al [2]. put forth an evaluation of Fisher Vectors and CNN frameworks for the task of image retrieval, revealing that their combined use outperforms each method individually on certain datasets. While these approaches also leverage CNNs like ours, they differ in objectives and network architecture and are not designed to directly manage image pairs. Interestingly, Babenko et al [1]. have suggested in their concluding remarks the potential utility of Siamese networks for direct feature learning for image matching and retrieval, an area which has yet to be extensively explored.

The recent study by Zbontar and LeCun [11] primarily concentrates on a technique for comparing small image patches to extract depth information. This method employs convolutional networks and minimizes a hinge loss function, exhibiting superior performance on the KITTI stereo evaluation dataset. However, the method's limitation lies in its focus on very small patches (9x9 pixels), thereby narrowing its scope of applicability. In contrast, our work aims to address a more comprehensive range of image variations, making our approach suitable for a wider array of applications.

3 METHOD

Our objective is to devise a robust similarity function that can effectively distinguish between similar and dissimilar images within a given dataset. To achieve this, we introduce an approach built upon two deep convolutional neural networks, specifically leveraging the proven architectures of VGG19[9] and ResNet50[4], which have shown remarkable performance in image classification tasks.

In a simplified view, as depicted in Figure 2, our model is comprised of twenty-one interconnected branches. During the training phase, a batch of twenty-one images is processed through this network. The computed features are then passed to a specialized loss layer designed to minimize the cosine similarity for dissimilar image pairs while maximizing it for similar pairs.

The succeeding sections will delve into our custom loss function, explaining its role and implementation in our framework. A more detailed account of the model's architecture is also provided in a dedicated section.

3.1 Contrastive loss

To optimize the proposed network, we utilized a cost function that is capable of distinguishing between pairs. More precisely, it encourages similar examples to be close, and dissimilar ones to have distances of at least margin m from each other. To implement this, we studied the margin-based contrastive loss function proposed in [3] which is defined as follows:

$$L = \frac{1}{2} l D^2 + \frac{1}{2}(1-l) max(0, m - D)^2$$

We then made our own modifications based on this function. Within the twenty right images, we select the most similar image the model predicts then minus the similarity score between this with our correct right image, which will be our loss. The loss L for a single instance can be defined as:

$$L = \max(0, \max(D[:, 1:]) - D[:, 0] + m)$$

The final loss is the then average loss over all instances N:

$$CustomContrastiveLoss == \frac{1}{N} \sum_{i=1}^{N} L_i$$

where,

- L is the overall loss calculated over N instances in the batch.
- $D_{positive,i}$ is the distance for the positive pair for the i^{th} instance.
- $D_{negative,i}$ is the set of distances for the negative pairs for the i^{th} instance.
- $\max(D_{negative,i})$ is the maximum distance among the negative pairs for the i^{th} instance.
- m is the margin, a hyperparameter usually set to 1.0.

3.2 Input Image Preparation

Given that the similar image pairs are already labelled, the remaining challenge is the selection of the additional nineteen images to be compared with the 'left' image. The choice of these images can significantly impact the model's learning capacity; essentially, the closer these images are in resemblance to the left image, the more nuanced the model becomes in distinguishing subtle differences.

To address this, we employ a randomized selection approach that draws nineteen diverse images from the entire dataset. This strategy not only enriches the variety of images the model is exposed to, but also simplifies the preprocessing stage, as the only parameter needed is the number of random images to be selected. Consequently, this method facilitates both rapid and efficient data preprocessing, enabling the model to focus on learning the critical features that distinguish similar images from dissimilar ones.

3.3 Architecture

The core of our system is built around a Siamese Network architecture designed for image similarity tasks. We have followed a similar approach as [6]. The network takes one 'left' image and a set of twenty 'right' images as inputs. The Siamese Network employs dual feature extraction stages which are different to [6], where each stage utilizes a pre-trained model -EfficientNetV2S[10], VGG19[9], ReseNet50[4] or MobileNetV3Large[5] followed by fully connected layers and a similarity computation unit.

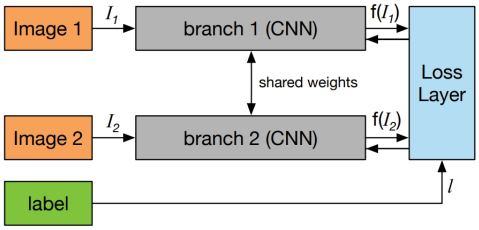


Figure 2: Model Structure from [6] However with our model we are using 21 images through the network

3.4 Feature Extraction

For feature extraction, we used those pre-trained models to encapsulate various aspects of the visual data. Each base model processes the left and right images separately, generating feature maps. The feature maps were then flattened to form feature vectors. The feature vectors obtained from the two different base models are combined. We explored both concatenation and average strategies to integrate the feature vectors and produce a comprehensive representation. After feature fusion, the resulting feature vector goes through a sequence of fully connected layers.

3.5 Similarity Computation

Finally, the similarity between the left image's feature vector and each of the right images' feature vectors is computed using cosine similarity[7]. The raw similarity scores are normalized and converted to fall within the range of [0,1]

3.6 Evaluation Metirc

We employ Top-2 K categorical Accuracy as our evaluation metric, which considers the model's prediction correct if any of the top two most similar 'right' images are indeed the correct match.

4 EXPERIMENTS

In this section, we delve into the empirical results to critically assess the efficacy of our proposed methodology. Specifically, we aim to answer two pivotal questions:

(a) Does the integration of two distinct pre-trained deep learning models yield superior performance compared to a singular pre-trained model?

The initial approach involved the utilization of a single pre-trained model, namely VGG19, ResNet50, MobileNet, and EfficientNet. However, the results from these individual models did not meet our expectations, with the highest accuracy achieved by ResNet50 being approximately 0.4.

Consequently, we made the decision to enhance model complexity and improve accuracy by combining two different models. In this approach, we integrated the ResNet50 model as our primary choice due to its superior performance when compared to the other individual models. The second model consists of the remaining models in our selection. In order to determine whether the combination model is superior to the single model, we performed five different permutations of the combination model. These include ResNet50/VGG19, ResNet50/MobileNetV3Large, ResNet50/EfficientNetV2S, ResNet50/InceptionV3, and ResNet50/MobileNetV3small. According to Figure 3, the single model has an average accuracy of approximately 0.37, while the combination model has an average accuracy of 0.61(figure 4). It is evident that the combination model performs better than the single model.

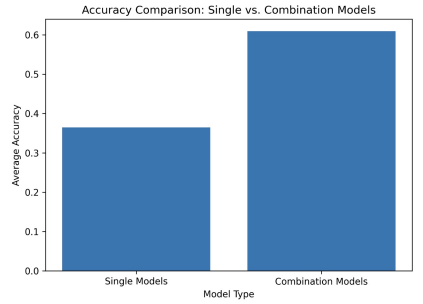


Figure 3: Average accuracy for single model and combination models

(b) Among the combinations of two models, which pair is most effective at feature extraction for the purpose of identifying image similarity?

Through these queries, we aspire to provide a comprehensive evaluation that not only validates the advantages of our dual-model approach but also identifies the optimal model pair for image similarity detection.

By evaluating different permutations of those combination models, we set the batch size, embedding model, epoch, and learning rate as the same to guarantee the consistency and fairness of our experiments. As a result, any variation in performance can be attributed to the inherent differences between the model combinations, rather than external factors. The combinations of ResNet50/VGG19 and ResNet50/MobileNetV3Large emerged as the top performers, achieving accuracies surpassing 0.67(figure 4). While we experimented with a variant of MobileNet called MobileNetV3Small, its performance did not surpass that of MobileNetV3Large. Therefore the ResNet50/VGG19 and ResNet50/MobileNetV3Large seemed as most effective at feature extraction which pair achieves the highest accuracy.

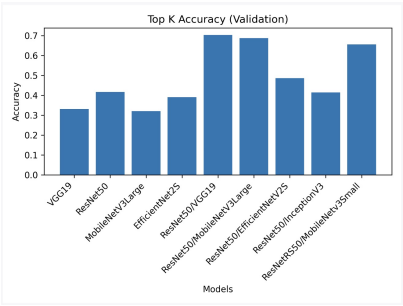


Figure 4: Top K Validation Accuracy

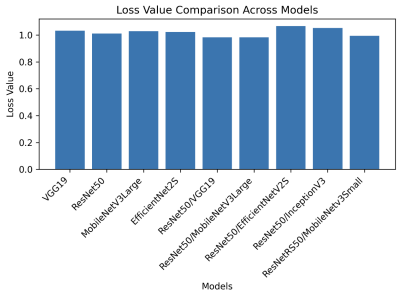


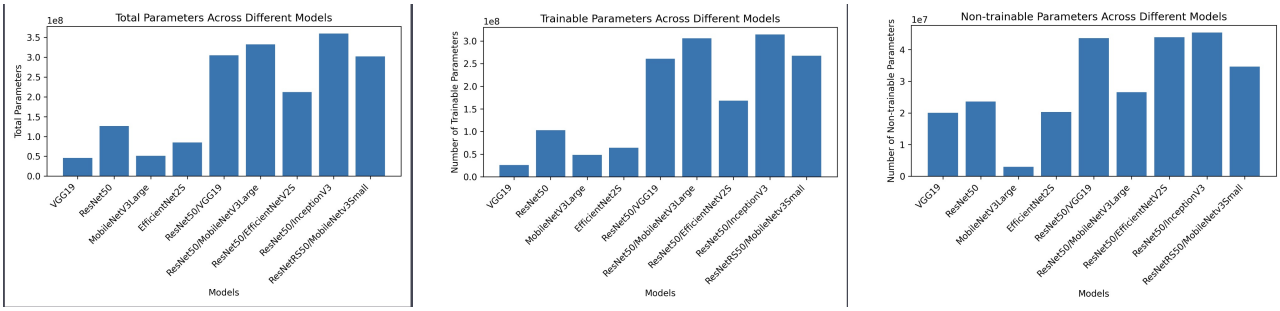
Figure 5: Loss value each model

5 RESULTS AND DISCUSSION

In this section, we will discuss the results of experiments with various permutations of pre-trained models. Based on deep learning techniques, we compare the results of the following approaches:

Comparing the Figure 4 model accuracy and Figure 5 loss value for each model, we can conclude that the loss values of all the models are similar, hovering around 1.0. Generally, a lower loss value indicates better predictions. As a result of the uniformity of loss values across all models, it is difficult to determine which model is the best based solely on this metric.

As a single model may not be sufficient for our Siamese Network, we chose to use two models for feature extraction, which were then combined to produce varied results. Our primary choices, ResNet50/MobileNetV3Large and ResNet50/VGG19, demonstrated the most promising performance. As a result, they achieved the highest validation accuracy of approximately 0.68 (figure 4), along with test accuracy of 0.631 and 0.623, respectively.



(a) Total Parameters (b) Trainable Parameters (c) Non-trainable Parameters

Figure 6: Parameter Statistics

Upon comparing Figure 6, which illustrates the total number of parameters, with Figure 4 model accuracy. This observation suggests that an increase in parameter count may lead to an improvement in model accuracy. However, an excessive number of parameters may result in overfitting, which will have a negative impact on the accuracy of the model. For example, ResNet50/InceptionV3 had the biggest number of parameters, but it. And its performance is bad. And it training time is much longer than our model. Therefore, this analysis underscores the imperative of balanced model complexity, ensuring a trade-off between accuracy and efficiency.

As a result, combined models tend to have a higher trainable parameter set than their independent counterparts. It may mean that they are more expressive, but it may also mean that they require more training data and more time

References

[1] Artem Babenko, Anton Slesarev, Alexandr Chigorin, and Victor Lempitsky. Neural codes for image retrieval, 2014.

[2] Vijay Chandrasekhar, Jie Lin, Olivier Morere, Hanlin Goh, and Antoine Veillard. A practical guide to cnns and fisher vectors for image instance retrieval, 2015.

[3] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742, 2006.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[5] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. Searching for mobilenetv3, 2019.

[6] Iaroslav Melekhov, Juho Kannala, and Esa Rahtu. Siamese network features for image matching. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 378–383, 2016.

[7] Gang Qian, Shamik Sural, Yuelong Gu, and Sakti Pramanik. Similarity between euclidean and cosine angle distance for nearest neighbor queries. In *Proceedings of the 2004 ACM Symposium on Applied Computing*, SAC '04, page 1232–1237, New York, NY, USA, 2004. Association for Computing Machinery.

[8] Amir Rosenfeld, Markus D. Solbach, and John K. Tsotsos. Totally looks like - how humans compare, compared to machines. version: 3.

[9] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

[10] Mingxing Tan and Quoc V. Le. Efficientnetv2: Smaller models and faster training, 2021.

[11] Jure zbontar and Yann LeCun. Stereo matching by training a convolutional neural network to compare image patches, 2016.