CV Report

Zhewen Zhang 1320186 Hengrui Qu 1398834

Abstract

In the dynamic realm of computer vision, algorithms that emulate intricate human visual perception are vital. This project explores the Totally-Looks-Like Challenge, inspired by a platform where users match visually alike image pairs, thus redefining traditional AI-driven image recognition and similarity standards. Despite advanced algorithms excelling in tasks like object classification or face recognition, mirroring human abstract image reasoning remains a puzzle. The challenge tasks us to pinpoint similar images from pairs, enhancing existing models by considering factors like color, shape, and texture. This not only demands deep image feature understanding but also a rigorous evaluation of model efficacy. Through this, we aim to understand the disparity between model-driven and human image similarity perceptions, shedding light on future computer vision research directions.

1 Introduction

Advances in computer vision have changed the way we understand visual data and have applications in areas such as healthcare and autonomous driving. One such challenge is the total similarity (TLL) challenge, derived from pairs of similar images on the Reddi platform. This challenge focuses not only on surface similarity, but also on details between two seemingly different images, such as color and texture. Despite the progress made, machines still fall short of humans in understanding visual similarities. The dataset used for the project contained 2,000 training image pairs and 2,000 test image pairs, with no validation set, which further complicates algorithm development.

The report emphasizes training algorithms to recognize image similarities intended to be similar to human judgments. We suggest fine-tuning models for image analysis that specialize in this task to help them detect subtle similarities between images, thus simulating human-like similarity perception. Subsequent sections cover related work, our approach, experiments, results, and potential directions for future exploration.

2 Literature Review

2.1 Dataset

In this experiment, each image pair of the dataset is divided into "left" and "right" image components. The dataset is further divided into training and testing subsets, each containing 2000 pairs, to achieve a balanced distribution for model training and evaluation. The resolution of each image is normalized to 200x245 pixels. The test subset is created by pooling each "left" image with a pool of 20 potential "right" match candidates. Each candidate pool encapsulates the true "right" image and 19 foil images that were indiscriminately selected from the test subset, thus embedding a layer of complexity and ambiguity in the matching task [1].

2.2 VGG

VGG16 whose powerful feature extraction capabilities can be used to extract multi-scale features from images is useful when analyzing different types of similarities (e.g., color, texture, and shape) in the dataset images [2]. Meanwhile, through migration learning, using VGG16's pre-trained weights on ImageNet, the model can be enabled to proficiently extract meaningful features from the TLL dataset, which provides assistance in detecting subtle similarities between different images and meets the challenge's requirement of perceiving different visual similarities such as color and texture.VGG16's convolutional layer is capable of extracting local features from images through a small local receptive field, which is also useful in analyzing different types of similarities such as color and shape in the dataset images. extract local features of an image, which may also help in recognizing local similarities in images in the TLL challenge [3].

Figure 1: The figure shows the architecture of Resnet50 Model

2.3 Resnet

ResNet or Residual Network, which mitigates the gradient vanishing/exploding problem through its residual learning approach, has deep capabilities in handling deep learning challenges. We consider that the depth of ResNet helps in extracting hierarchical features from images ranging from low-level details to high-level semantics, which is crucial for recognizing and correlating visual commonalities within and between different image pairs in the TLL challenge [4].

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 2: The figure shows the architecture of Resnet50 Model

3 Methodology

The algorithm we designed consists of two parts, the first part is to extract the feature vectors of the left and right images through a model such as vgg, and then in the second part the differences between the images are calculated using an adaptive loss function that can compute the similarity of the images as shown in the figure below.

3.1 Model design and feature extraction

The data format of the model input from the dataset has already been described in 2.1, using a preprocessing method with positive and negative sampling that learns the differences with the remaining 19 uncorrelated images (negative samples) while comparing the differences between LEFT and GROUND TRUTH RIGHT (as positive samples). These will be explained in detail later in 3.2, before that we will focus on how to design the model used for feature extraction. For the model selection we considered vgg and resnet, as both models have won sota in imagenet competitions, so these two models should perform well for local feature extraction. Since we want to compare the similarity by the feature vectors of the images, we should skip the fully-connected classification layer after convolution as in the vgg model and instead add several layers as shown in the following figure:

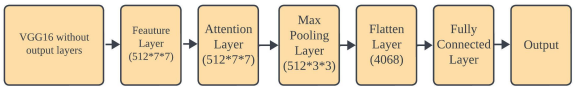


Figure 3: The diagram shows the layers we added after the model

Features Layer extracts basic visual information and features of similarity between images from the original image. Then in Attention Layer, by assigning different weights to different features, the attention mechanism can help the model to recognize and focus on the features that are more important for the similarity judgment, so as to improve the performance and accuracy of the model, and enhance the computational efficiency. The Max pooling layer then reduces the size of the feature map to minimize the computational effort while retaining the most important information (the maximum value of each small region) as shown in the figure. It also provides a local invariance that makes the model more robust to small spatial variations and helps the model to recognize and compare similarities between images. Finally the feature vectors are passed through the Flatten Layer and Fully Connected Layer to transform the feature map into a one-dimensional vector and map it to a new space.

3.2 Adaptive loss function

Triplet Loss operates on three samples: the anchor point (left image), the positive sample (which has a high similarity to the anchor point), and the negative sample (which does not have a high similarity to the anchor point). The goal is to make sure that the anchor point will be close to the positive value and far from the negative value. It helps the model to learn semantically meaningful relationships between embeddings, pushing similar items closer together and different items farther apart in the embedding space. To some extent, it improves the generalization ability of the model with better robustness.

$$L_{triplet} = \max(0, D(a,p) - D(a,n) + \alpha) \quad (1)$$

While cosine similarity measures the cosine of the angle between two non-zero vectors. This value ranges between -1 and 1, where the value 1 means the vectors are identical, 0 means they are orthogonal (uncorrelated), and -1 means they are diametrically opposed. Cosine similarity effectively measures how similar two embeddings are in terms of direction [5].

$$Similarity(A,B) = \frac{(A * B)}{||A|| * ||B||} \quad (2)$$

Since cosine similarity is only related to the direction and not to the magnitude of the vectors, it makes the embeddings scale-invariant. This is crucial for the task of comparing similarity of feature vectors of different magnitudes as in our experiments. Incorporating the cosine into the triple loss ensures that the network does not collapse all representations to a single point and maintains a certain distribution in the embeddings, leading to more stable training.

3.3 Hyper Parametric design

The main hyperparameters involved in this experiment are epoch, batch size, learning rate, and weight decay. epoch size is set to 10 in order to avoid overfitting the model with too many training rounds.

The batch size is set to 10 for the vgg16 model and 6 for the resnet50 model. This is due to the fact that we are training on an NVIDIA 4090 graphics card, which is limited to 24 Gigabytes of memory, and because the dimensionality of the input dataset is very large (20 right images with an input dimension of [20, 245, 200]) the batch size is already set at 10 for both batch sizes. batch size choices are the maximum that can be set within the 24G memory limit. While a smaller batch size can learn more detailed information, it also introduces more noise, which we can corroborate in a later section.

The learning rate is set to 1e-4, considering that too large a learning rate may cause the gradient of the model to drop too fast and not learn enough local information. However, the actual performance shows that the loss is basically 0 in the later rounds of training in vgg16, so the learning rate can be increased appropriately, and the weight decay used to limit the learning rate can be reduced at the same time.

4 Result and Discussion

4.1 Result

The accuracy score on Kaggle is evaluated with reference to the Top-2 accuracy. It reflects the ability of the model to rank truly similar images in the top-2 of the ranked similarity score. In other words it evaluates the difference between the similarity of the best two left-right image pairs and the standard answer for each data set.

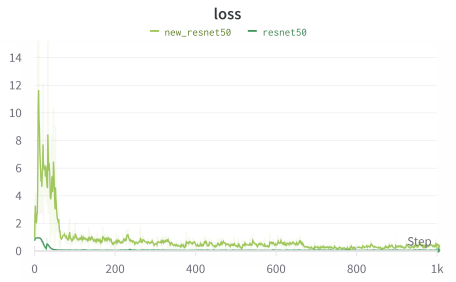
Before the algorithm was improved, the score of VGG16 was 0.178, while the score of ResNet50 was 0.249. After the improvement, the score of VGG16 soared to 0.569, while the score of ResNet50 was only 0.419. The Figure 4 and 5 shows the performance of the VGG16 and Resnet50 models, before and after we implement the new algorithmic design.

4.2 Result Analysis

The enhancement of our model's performance comes from a dual algorithmic design: extracting features from models such as VGG, and using an adaptive loss function tailored to image similarity.

1. Feature extraction via VGG and ResNet: both VGG and ResNet were considered because of their established performance in the ImageNet competition, which demonstrated their adeptness at local feature extraction. Bypassing the fully-connected classification layer common after convolution in VGG, the model is augmented to merge multiple layers, such as the feature layer for basic visual information extraction, the attention layer for weighing and focusing on salient features, and the maximal pooling layer efficiency for computation while retaining important information [3]

(a) VGG16 Loss Comparison



(b) Resnet50 Loss Comparison

2. Hyperparameter tuning: The significant effect of hyperparameters becomes apparent during the experimental phase. For example, while a learning rate of 1e-4 was initially set to ensure that the model does not ignore localized complexity during gradient descent, the near-zero loss observed in the latter training period of VGG16 suggests the possibility of a more aggressive learning rate. Additionally, given the memory limitations of the NVIDIA 4090 GPU, different batch sizes for VGG16 and ResNet50 were optimized, balancing the trade-off between detail preservation and noise introduction.

4.3 Discussion

By looking at the results of the algorithm enhancements, it can be seen that both VGG and ResNet exhibit improved performance. However, the performance of VGG is significantly superior. The potential reasons for this are explored in depth:

1. Model complexity: ResNet's complex architecture, while deep, may not always translate into better performance in specific tasks like image similarity.The consistent architecture of VGG16 may provide a more stable optimization environment.

2. Feature Vector Dimension: The feature vectors derived from these models serve as the basis for similarity comparisons. it seems reasonable that VGG16's feature vectors would inherently have properties that are better suited to this particular task.

3. Model Improvements: Attention to feature extraction, especially the integration of the attention layer and the max pooling layer, is critical. The attention layer assigns different weights to different features, allowing the model to identify and prioritize more relevant attributes. On the other hand, max pooling ensures computational efficiency without compromising salient features [5].

4.4 Error Analysis

During the model refinement process, some design choices, while theoretically sound, did not produce the expected benefits:

1. Black border removal: while the logic of removing black borders from the image may seem reasonable, the inherent pixel representation of black is zero, making this step ineffective and of no real benefit to the model's performance.

2. Grayscale conversion: an attempt was made to simplify the parameter space of the model by converting the image to grayscale, but unintentionally obfuscated some of the color context cues and complicated the task of interpreting the model.

5 Conclusion

In this paper we discuss how to approximate the human perception level on the task of image similarity. We propose an algorithmic model that combines feature extraction and adaptive loss function, which also demonstrates very good similarity judgement after fine-tuned training. The original model using only VGG16 or resnet50 does not perform well in similarity perception. Traditional computer vision single models do not have the complexity to combine different semantic, texture, structure, colour and other information in a way that is more similar to human beings. While our current design demonstrates good results, there may be greater potential for dual-network mechanisms such as twin networks. Its design focuses on comparing pairs of inputs, so it has inherent advantages for image similarity tasks. In conclusion, while our innovative approach has achieved good results in bridging the gap between human perception and machine interpretation of image similarity, we believe that more comprehensive and generalised large models that can be applied to different kinds of computer vision tasks are the way forward.

References

[1] Amir Rosenfeld, Markus D. Solbach, and John K. Tsotsos. “Totally Looks Like - How Humans Compare, Compared to Machines”. In: *ACCV*. 2018.

[2] Karen Simonyan and Andrew Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: *arXiv preprint arXiv:1409.1556* (2015).

[3] Kai Han, Andrea Vedaldi, and Andrew Zisserman. “Learning to Discover Novel Visual Categories via Deep Transfer Clustering”. In: *arXiv preprint arXiv:1908.09884* (2019).

[4] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *arXiv preprint arXiv:1512.03385* (2015).

[5] Olivier Risser-Maroix et al. “Learning an Adaptation Function to Assess Image Visual Similarities”. In: *arXiv preprint arXiv:2206.01417* (2022).