A Comparative Analysis of Modeling Approaches for Totally-Looks-Like

[redacted] [redacted]

***Abstract*—In the evolving domain of computer vision, understanding the divergence between artificial intelligence and human visual perception holds significant real-world implications. This research addresses the problem of accurately evaluating the human-like perception of images through the Totally-Looks-Like challenge [3]. Focusing on the TLL dataset [1] – a diverse collection of image pairs with varying degrees of similarity in aspects like color, shape, texture, and facial expressions – we aimed to develop an algorithm that excels in identifying resemblances. Our innovative methodology integrates advanced computer vision techniques, such as deep learning models and feature extraction algorithms. The results indicate that our algorithm showcases an enhanced ability to identify visually similar image pairs from the TLL dataset. This advancement aligns with and expands upon existing research, offering implications for further algorithmic improvements and broader applications in visual recognition. This experimental report reveals the performance of both traditional image classification methods and innovative approaches for this specific task, offering insights and references for subsequent research.**

I. Introduction

The human visual system excels at identifying details and patterns in images. Despite advancements in computer vision, replicating this human perception remains a challenge. One of the most important challenges in this aspect is the task of discerning visual similarities between images, a task that humans often perform effortlessly.

The Totally-Looks-Like (TLL) challenge [3] and its dataset, sourced from the popular online platform Reddit, present a unique opportunity to explore this problem. The dataset contains a diverse collection of image pairs that users believe resemble to each other. These resemblances cover various dimensions, from color and shape to texture and facial expressions, making it a rich resource for studying visual similarity perception.

Convolutional Neural Networks (CNNs), with their hierarchical structure and ability to learn spatial hierarchies of features, have shown potential in various computer vision tasks. In this report, we leverage CNNs and a method of using adaptation function to assess image visual similarities [2] to tackle the TLL challenge, aiming to bridge the gap between computer vision and human perception of image similarity. By focusing on the TLL dataset, we not only aim to find an algorithm that can identify visual resemblances similar to human perception but also contribute to the research on how machines can be trained to "see" more like humans.

[1]https://www.reddit.com/r/totallylookslike/

This report presents an exploration of methodologies to address the TLL challenge. We begin with a baseline approach, using the VGG16 model. Building upon this foundation, we introduce a finetuned model, employing the ResNet50 architecture. The method of adaptation function (to assess image visual similarities) shows a better performance in this task. Through a series of experiments, we seek to identify the advantages and limitations of each approach, offering insights that can be helpful in this domain.

In the following sections, we outline our experiments and results. We aim to provide more experimental references.

II. Experiments

A. Data Preprocessing

In terms of predictions, we need to generate a dataset, with 20 images as candidates(right image) for the potential match of the target image(left image). These candidates include the ground truth image and 19 randomly selected images as negative samples.

B. Baseline Method

Our baseline model uses a VGG16 model with pre-trained ImageNet weights and extracts features from the first fully connected (FFC) layer. In deep learning models, the FFC layer plays a crucial role in classifying input data by utilizing the features learned from preceding layers [1]. They contain a substantial number of parameters, making it necessary to fine-tune the parameter set for effective model performance.

Subsequently, we save the features of the test images to a file. For each pair of a left image and a right image, we can load the corresponding features and compute the cosine similarity. The definition of cosine similarity is [6]:

$$\cos(\theta) = \frac{\mathbf{AB}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} \mathbf{A}_i \mathbf{B}_i}{\sqrt{\sum_{i=1}^{n} (\mathbf{A}_i)^2} \sqrt{\sum_{i=1}^{n} (\mathbf{B}_i)^2}} \quad (1)$$

Cosine similarity is an optimal choice for comparing visual similarities between images because it focuses on the direction of vectors instead of their magnitude. This ensures images with similar content but different brightness have high similarity. Suitable for high-dimensional data like images, it's less sensitive to dimensionality compared to some distance measures like Euclidean distance.

This method is straightforward and it is set as the baseline model of our experiments. With the Top-2 accuracy evaluation metric, the baseline model gets the accuracy of 0.461 in the

test dataset on the Kaggle [2] platform, indicating that there is potential to enhance the performance.

C. Finetuned Model

1) Idea: We intended to concatenate each pair of the left image and the right image together as the input of our model. Using this approach, we can convert the problem to a binary classification task: if they are the correct match, the label is set as 1, otherwise, the label is 0. Therefore, we can build a new dataset to train the model to label the image pairs.

2) Data Preprocessing: The original extended dataset has 20 candidates for the left image. To transform the dataset to image pairs, we need to recreate the test dataset and assign the label to image pairs. If the right image is the ground truth, the label column is 1; if the right image is a random foil, the label is 0.

3) Model Definition: We fine-tuned a model built upon ResNet50, utilizing pre-trained ImageNet weights. The top classification layers of the original ResNet50 were omitted. To preserve the integrity of the pre-trained weights, we iteratively froze the layers of ResNet50, ensuring they remain unchanged during subsequent training. We also implemented a checkpoint callback to store the optimal model weights. Our enhancements began with the addition of a Global Average Pooling layer, which averages the feature maps from ResNet50's output. This was followed by a fully connected layer activated by ReLU. Ultimately, for binary classification, we introduced a dense layer with a single unit, activated by a sigmoid function, which effectively maps the output value between 0 and 1. [4]. The definition is:

$$\sigma(x) = \frac{1}{1 + exp(-x)} \tag{2}$$

4) Result: The performance of this model was not as good as we expected. On the Kaggle platform, the accuracy is 0.248, which is worse than our base model. Reflecting on the method, there are several issues that need to be addressed.

The first issue is that the new test dataset contains imbalanced data. The number of image pairs in the negative class (label 0) significantly outweighs the number of image pairs in the positive class (label 1). This imbalance makes it difficult for the model to learn the ground truth and results in high accuracy and low recall. To solve this issue, we reduced the number of negative data to the same as the positive class. The recall increased to 0.547 and the accuracy is 0.713 as shown in the TABLE I. However, the original task is to compare the similarity between the left images and 20 candidates. Changing the dataset modifies the definition of the task and therefore the model can only recognize the ground truth image when there is one candidate.

There exists a disparity between the model's objective and our actual task's goal. The model aims to determine the class to which an image pair belongs. In contrast, our task involves measuring the similarity between a target image and 20 candidate images. This implies that the model must execute its task twenty times to approximate a single prediction in our actual task, which increases the likelihood of the model's failure. The evaluation metric is top-2 accuracy and it indicates that if there are two images in the 20 candidates incorrectly labeled, the task fails and the accuracy drops. This results in low accuracy on the Kaggle platform (0.248).

[2]https://www.kaggle.com/competitions/comp90086-2023/overview

	Imbalanced data	**Balanced data**
Accuracy	0.682	0.713
Recall	0.197	0.547

TABLE I
ACCURACY AND RECALL OF VALIDATION DATA

Notes: 1. *Imbalanced data*: 1 positive sample : 19 negative samples. 2. *Balanced data*: 1 positive sample : 1 negative sample.

The second issue is that when the left image and the right image are concatenated, image concatenation assumes that the relevant information from both images can be effectively represented by combining them horizontally. If the two images contain related information, this approach may work. However, for the two images that are not matched, the important features and information may not be extracted correctly when they are concatenated. For example, in the concatenated image below Fig. 1, the left image and the right image both have faces. However, none of the colors, shapes, and textures in the two images have anything in common.

Fig. 1. An example of concatenated image

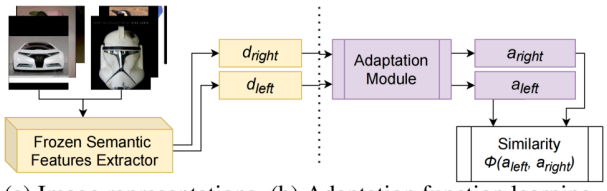
The third issue is about the dimensionality. Image concatenation can significantly increase the dimensionality of the input data, which may lead to increased computational complexity. Our model is not sophisticated enough to handle the multiple dimensions of concatenated images.

D. Adaptation function

Human perception intuitively identifies image similarities, which is a process not fully replicated by computer vision due to a complex cognitive basis. We explored cutting-edge techniques which include deep learning methods and comparing image feature vectors. These methods have a good performance with semantically related images but struggle with those visually alike yet semantically different.

1) Idea: In this paper [2], an adaptive function approximating correspondence with the primate IT cortex is solely learned by means of a metric learning framework. This approach inspired us to change our perspective. Previously, our fine-tuning method aimed to adjust features to minimize the loss with a fixed computational rule. Drawing from this, we propose training a scorer (essentially an adaptation function) to rate the similarity between a pair of images. By feeding features

into our trained scorer (a simplistic model comprising a single fully connected layer), the scorer assigns specific weights to particular features, indicating which attributes it prioritizes. This process facilitates the computation of similarity scores.



(a) Image representations (b) Adaptation function learning

Fig. 2. Developing a pipeline to get the visual similarity of image pairs by learning an adaptation function. Reference [2]

The Fig 2 illustrates a structured procedure involving image representation and adaptation function learning. In the initial phase, images are processed through a 'Frozen Semantic Features Extractor', which captures high-level features, yielding vectors d_left and d_right for the corresponding images. This extractor remains static with its parameters unaltered during learning. Importantly, the left and right images are paired as positive pairs. Conversely, all other right images serve as negative pairs for a specific target left image. The subsequent phase involves repetitively processing these representations through the 'Adaptation Module' during training to minimize the loss. This loss function is tied to the cosine similarity matrix and is framed as a Cross-Entropy Loss. The objective of this training is to refine the 'Adaptation Module', enabling it to identify a relatively higher similarity for each positive pair(matrix diagonal data), while concurrently increasing the distance between all the negative image pairs(matrix non-diagonal data).

By utilizing the Cross-Entropy Loss with the cosine similarity matrix, this problem is essentially transformed into a multi-class classification task. This addresses some issues highlighted in the previous experiment and represents an improvement based on our baseline model. On the Kaggle platform, this method achieved a Top-2 accuracy of 0.63 on the test dataset.

2) Improvement and result: In this section, we will discuss details about our design options that make the test evaluation performance increase by 11%.

In the experiments, We split data into train data and validation data with a ratio of 75:25.

MODEL: The TABLE II presents loss values for various models on a validation dataset after training. Of the models listed, "clip_rn50x4" has the lowest loss values, both smoothed (3.2853) and raw (3.2849).

Based on this data, we selected models from the CLIP family for further investigation. Specifically, "clip_rn50x4" was chosen as the base model for further optimization due to its superior performance.

DATA AUGMENTATION: The Fig. 3 illustrates two trajectories, representing the performance of models with and without data augmentation. As can be seen from the figure,

Models	Smoothed	Loss Value
ViT	4.7055	4.7083
barlow	4.339	4.3392
clip_rn101	3.309	3.309
clip_rn50x4	3.2853	3.2849
clip_vit32_b	3.5185	3.52
dino_resnet50	4.3802	4.3809
facenet	5.1056	5.1053
resnet50	4.6524	4.6549
resnext101_32x48d_wsl	4.3058	4.3057

TABLE II
LOSS VALUES IN THE VALIDATION DATASET WITH DIFFERENT MODELS AFTER TRAINING.

Note: At the lowest point - when epoch = 50

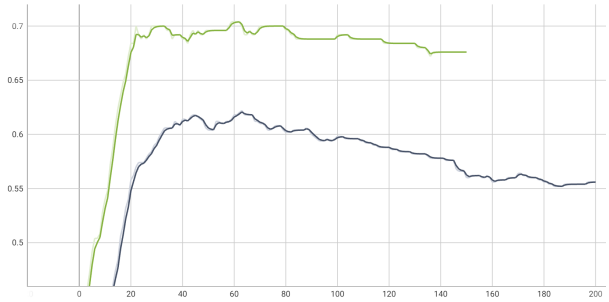


Fig. 3. Performance(Top-2 accuracy) on validation dataset for different options - data augmentation(RandomHorizontalFlip(p=1)) or not

the model using data augmentation (represented by the lower curve) consistently outperforms the one without data augmentation (represented by the higher curve) throughout the training process. Potential explanations for this phenomenon include overfitting caused by data augmentation. Furthermore, data augmentation can result in minor changes in the orientation, size, or angle of images, which can be detrimental to our specific task. Such augmentations could potentially undermine the model's learning capacity. For instance, an image of a vertical skyscraper and a pen, when rotated or flipped, might appear distinct. However, with data augmentation, the model is still trained to recognize them as similar, leading to potential confusion and worse performance.

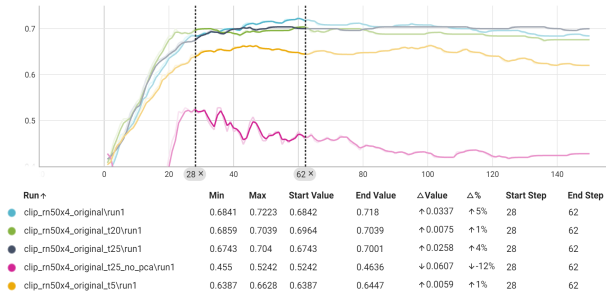


Fig. 4. Performance(Top-2 accuracy) on validation dataset for different options, focus on "cpa" and "temperature" parameters

CPA AND TEMPERATURE: In the Fig. 4, the red line represents a model tested on the validation dataset without PCA, while the other lines indicate models trained with

PCA(with 256 dimensions). Evidently, PCA improves performance. It's also worth noting that parameter tuning, such as adjusting the temperature, can further enhance performance. ("t20" means temperature equals to 20)

PCA as a dimensionality reduction technique can give us the following benefits: **Reduced Overfitting:** By reducing the dimensionality of the dataset, it can help with mitigating the risk of overfitting, especially when the dataset has a high number of features relative to the number of observations. **Improved Computational Efficiency:** Reducing the dimensionality of the data can speed up training times and reduce computational costs. **Noise Reduction:** PCA can help filter out noise by retaining only the most significant variance in the data, which can lead to improved model performance.

In our code, the temperature acts as a pivotal scaling factor for the logits prior to the softmax operation in the computation of the cross-entropy loss. By adjusting the temperature, we can finely regulate the granularity of the probability distributions: a heightened temperature produces more diffuse distributions, whereas a reduced one yields more distinct ones. By softening the probability distribution, temperature can act as a form of regularization.

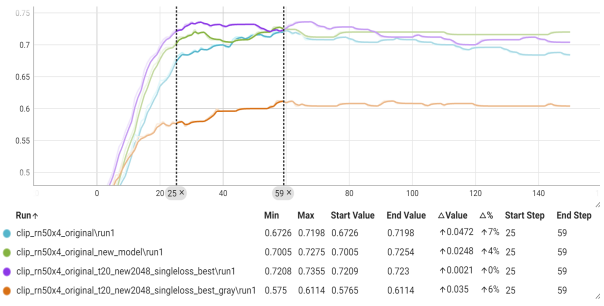


Fig. 5. Performance(Top-2 accuracy) on validation dataset for different options, focus on the variations of "adaptation function model", "loss function design" and "color or gray image as input"

LOSS FUNCTION: In our experiment, we restructured the loss function, and the results are illustrated by the purple line in the Fig. 5. It is obviously helpful to enhance the performance. Previously employing a bidirectional cross-entropy approach, we transitioned to a unidirectional variant, where the similarity is calculated for every left image against all right images. The Fig. 5 clearly illustrates the performance implications of this modified loss function, offering an insightful comparison against other methods. The equation is shown in Eq. (3)

$$L_{\text{left to right}} = \text{CrossEntropy}\left(\text{softmax}\left(\phi(a_{\text{left}}, a_{\text{right}})\right) \cdot \alpha, 1\right) \tag{3}$$

In Eq. (3), we used a parameter σ commonly known as the 'temperature parameter' in literature.

We know from [5], that human perceptions of similarity often exhibit asymmetry. For instance, many people would agree that a zucchini resembles a penguin more than a penguin resembles a zucchini. In our case, we only focus on identifying the correct right image for each left image. A bidirectional loss function might introduce confusion to the model.

PREPROCESS IMAGE: The orange curve illustrates the performance of a model trained on grayscale images. From the graph, it's evident that its performance is notably inferior compared to models trained on non-grayscale images.

Grayscale processing converts colored images into black and white, effectively stripping away color information and retaining only the luminance. However, in tasks determining the similarity between two images, color serves as a critical reference. The absence of color information through grayscale processing might cause the model to overlook some vital differences, resulting in decreased performance.

By integrating all the above-mentioned improvements, we made predictions on the test dataset. Ultimately, we achieved a Top-2 accuracy score of 0.699 on the Kaggle platform, marking a notable increase from the original 0.63.

III. Conclusion And Future Work

In conclusion, we made a comprehensive comparative analysis of modeling approaches for the Totally-Looks-Like (TLL) challenge. We explored various methodologies and models to bridge the gap between computer vision and human perception of image similarity.

We began with a baseline approach using the VGG16 model, followed by a fine-tuned model using the ResNet50 architecture. The fine-tuned model faced challenges due to imbalanced data and the failure to effectively represent concatenated images. To address these issues, we introduced an innovative adaptation function approach, inspired by a metric learning framework. This method involved training a scorer to rate the similarity between pairs of images, addressing some of the limitations in previous models.

Throughout the experiments, we identified several key insights and improvements. These included the impact of data augmentation, dimensionality reduction using PCA, temperature parameter tuning, and a modified loss function. The result showed that these methods led to significant improvements in model performance, achieving a Top-2 accuracy score of 0.699, which increased from the original 0.63 achieved with the adaptation function.

Our research has showcased the significance of integrating human perception into algorithmic design. There is great potential for further development in visual recognition.

References

[1] Chopra, P.: Progressivespinalnet architecture for fc layers. arXiv preprint arXiv:2103.11373 (2021)

[2] Risser-Maroix, O., Kurtz, C., Loménie, N.: Learning an adaptation function to assess image visual similarities. In: 2021 IEEE International Conference on Image Processing (ICIP). pp. 2498–2502. IEEE (2021)

[3] Rosenfeld, A., Solbach, M.D., Tsotsos, J.K.: Totally looks like-how humans compare, compared to machines. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. pp. 1961–1964 (2018)

[4] Sharma, S., Sharma, S., Athaiya, A.: Activation functions in neural networks. Towards Data Sci **6**(12), 310–316 (2017)

[5] Tversky, A.: Features of similarity. Psychological review **84**(4), 327 (1977)

[6] Xia, P., Zhang, L., Li, F.: Learning similarity with cosine similarity ensemble. Information sciences **307**, 39–52 (2015)