COMP90086: Totally-Looks-Like Challenge

Aadesh Samdaria
1363757
asamdaria@student.unimelb.edu.au

Arya Araban
1439683
aaraban@student.unimelb.edu.au

***Abstract*—This report explores three approaches for the "Totally Looks Like" challenge in computer vision, involving image resemblance. The methods include Canny edge detection, pre-trained model use, and a Siamese Network with triplet loss. Evaluation is conducted using a model test dataset and the Kaggle competition. The results show limitations of edge detection, a 43.1% accuracy with pre-trained models, and a 47% accuracy with the Siamese Network. With the aim of attaining superior outcomes while acknowledging computational constraints, extensive experimentation was carried out in hopes to further optimize the results.**

I. Introduction

The "Totally Looks Like" competition poses an interesting computer vision challenge - finding highly similar pairs of images that depict different objects or people. This gives us an opportunity to explore methods for effectively comparing and matching unrelated images. For this project, we tested three different methods for the "Totally Looks Like" challenge. The first uses Canny edge detection to identify edges in the images. The second leverages a pre-trained image encoding model to measure overall image similarity. The third is a Siamese Network that joins the pre-trained model with Triplet loss.

Our main objective is to thoroughly evaluate the effectiveness of these methods. To achieve this, we carefully assess them using a dedicated test dataset that closely resembles the provided test candidates. This dataset was created based on the training dataset. The model test dataset consists of 2000 rows, with each row containing a left image and 20 potential right images. Among these options, one represents a genuine match, while the remaining 19 are false candidates.

Furthermore, we evaluated each obtained model on the Kaggle "Totally Looks Like" competition dataset. This allowed us to see how our approach compares to others participating in the competition. We aimed to perform well in the competition and to deepen our comprehension of the continuously evolving field of computer vision.

II. Methodology

A. Canny Edge Detection Algorithm based image similarity

For this approach, we used the Canny edge detection algorithm [1] with lower and upper threshold values of 50 and 150. As seen through experimentation summarized in Table 1, these values balanced edge detection, image details, and noise sensitivity. Lower thresholds missed important edges while higher values were too susceptible to noise. Ultimately, our selected thresholds resulted in the best performance.

To assess the similarity between edges, we compute cosine similarity between the left image edges and each of the 20 right candidate edges. This produces 20 similarity scores, one per candidate, with the top 2 highest scoring candidates being predicted as matches to the left images. This approach yielded a "top-2 accuracy" range of 9% to 11% on our model dataset. When the same procedure was applied to the test dataset, it resulted in an overall competition accuracy of 10.6%.

TABLE I
Performance Evaluation of Canny Edge Detector at Different Thresholds

Threshold 1	*Threshold 2*	*Model Test Accuracy (%)*
0	100	10.2
50	150	10.5
100	200	9.5

In Figure 1, we see a precise prediction in the model test dataset by method A. The left image "ada" matches the "yxn" image on the right. This is clear when comparing the edge maps of the left image with those of "right1" and "right2," particularly in terms of the wide open mouth edges.

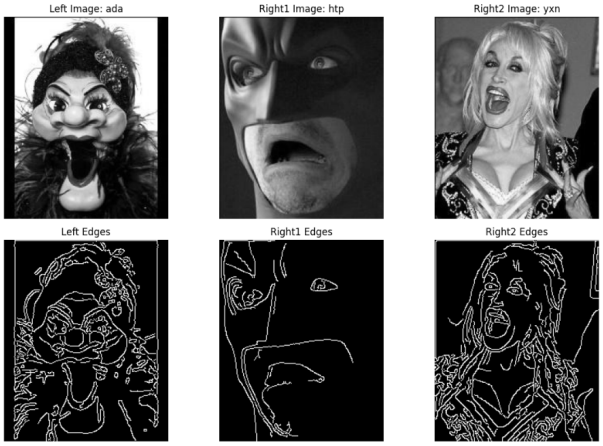

Fig. 1. Sample correct prediction of the model test dataset by Method A

Figure 2 depicts an incorrect prediction in the model test dataset. The model's top two predictions do not align with the correct match. For the left image "aaa," the accurate match is "osr." Visually, it's evident that the edges of the binoculars(circles) in the left image closely resemble the circles found in "right1" and "right2". This similarity likely

led to these two images being predicted as the closest pair for the left image.

Fig. 2. Sample false prediction of the model test dataset by Method A

In Figure 3, a Principle Component Analysis (PCA) plot [2] shows the edge embeddings of the left image and 20 possible right images for the prediction in Figure 2. While the left and correct right images appear close, random samples share similarities with the left image, placing them between the two. Consequently, the top two predictions were incorrect.

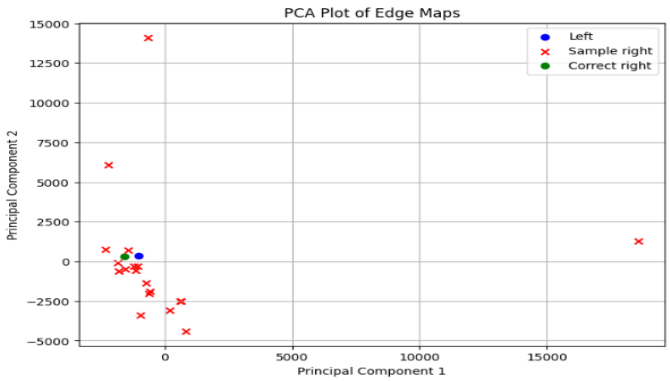

Fig. 3. PCA plot - Edge embeddings of the left image and 20 possible right images of the sample in fig 2

Challenges encountered when employing the Canny edge detection algorithm include its intrinsic limitations. The Canny algorithm's design predominantly centers on edge detection, with a notable focus on structural elements in images. This design choice unintentionally sidelines other vital visual attributes, such as texture and color, which can play a pivotal role in distinguishing images with subtle resemblances. This limited scope renders the model highly susceptible to variations within images, which, in turn, adversely impacts its performance in tasks like the "Totally Looks Like" competition.

Despite these acknowledged shortcomings, the Canny edge detection model serves as a valuable benchmark and stepping stone in the pursuit of accurate image comparison. By highlighting its limitations, it offers insights into areas of emphasis for the development and improvement of alternative methods.

B. Pre-Trained Model

In our efforts to overcome the limitations of our previous approach, we've introduced a significant enhancement. We've seamlessly integrated the use of the pretrained Xception model, as detailed in Chollet's research [3], which was rigorously pretrained on the extensive 'ImageNet' dataset [4]. This strategic incorporation empowers us to directly extract image embeddings through the capabilities of the Xception model. As a result, we effectively preserve and encapsulate all the intricate visual features contained within the image.

Following this enhancement, we proceed to assess the similarity between the "left" image and a variety of potential "right" images by employing the cosine similarity metric. This advanced approach has yielded remarkable results, achieving an impressive top-2 accuracy of 42% on the model test dataset.

In Figure 4, we can observe a situation where a correct top-2 prediction is made for an image that Method A had previously identified as incorrect. The right image "osr" is indeed the correct match for the left image "aaa," with a cosine similarity of 0.35.

Fig. 4. Sample correct prediction of the model test dataset by Method B

In Figure 5, we can observe an example of an incorrect prediction made by method B. Although the correct match for the left image "abh" is the image "cdx," the model instead predicts the closest matches as "bbd" and "vrz" with cosine similarity scores of 0.39 and 0.42, respectively.

Fig. 5. Sample false prediction of the model test dataset by Method B

Figure 6 displays a PCA plot that visually represents the image embeddings created by the Xception model for the prediction of the images displayed in Figure 4. The alignment between the correct right image and the left image embeddings becomes evident when considering the utilization of the cosine similarity metric. With this metric capturing the angle between vectors rather than their magnitude.

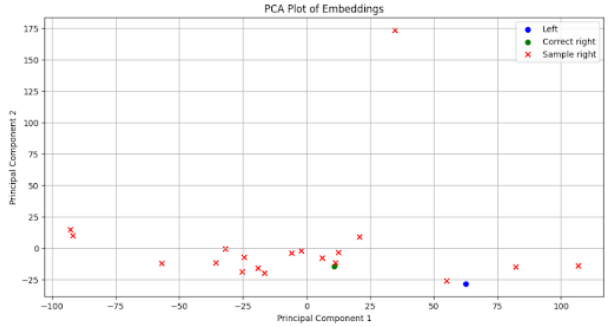

Fig. 6. PCA plot - Xception model image embeddings of the left image and 20 possible right images of the sample in fig 4

We applied the same method to the test_candidates dataset and submitted the scores to Kaggle and achieved an accuracy of 43.1%. This result is impressive considering our model relies solely on pre-trained embeddings and cosine similarity calculations, without any additional training. It showcases the potential of this approach and establishes a strong foundation for our final approach.

C. Siamese Network

In our final approach, we employ a Siamese network [5]. While the pre-trained Xception model has performed rather well, it's not tailored to our dataset. By training a specialized neural network which learns to differentiate between image pairs, we aim to create a model that's more attuned to our specific task. This approach, combined with an appropriate loss function, has high promise of improved accuracy.

Our model employs a Siamese architecture consisting of three shared Xception networks customized for our data. We start with an Xception base pre-trained on ImageNet, leaving the initial convolutional blocks frozen while we only unfreeze the final few convolutional modules. This balances transferring general representations with specializing low-level filters for our task. Additional dense layers are appended, enabling learning of high-level patterns. Figure 7 shows the architecture of the shared base model.

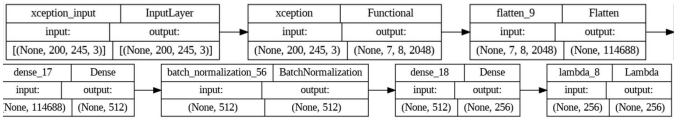

Fig. 7. Shared Base Model Utilized in Siamese Network

To enhance the Siamese network's performance in distinguishing between matching and non-matching image pairs, a triplet loss function was utilized. This involves modifying the training dataset so that the left image serves as the anchor, the corresponding right image is the positive example, and a randomly selected negative sample is chosen for each row. The negative sample is selected to ensure that it is different from both the anchor and positive example, thus allowing effective training of the triplet loss function. In Figure 8, one such example of these triplets can be seen.

Fig. 8. An example of a Triplet Used for Training

By employing the triplet loss and incorporating a final L2 distance layer, our siamese network model minimizes the distance between anchor and positive embeddings while maximizing the distance between anchor and negative embeddings. This optimization process leads to a specialized representation. Figure 9 illustrates the final structure of our siamese network model.

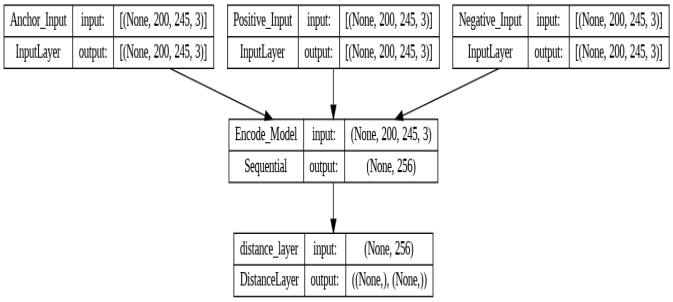

Fig. 9. Siamese Network Architecture

To optimize our data, after creating the triplets using our train data, we split them into training and validation sets. In the training set, we enhance generalization by duplicating each triplet multiple times while sampling a different negative for each duplication. For validation, we format it to be similar to the 'test_candidates' file by adding an additional 19 random negative images per row which are specifically sampled over the remaining validation images, thus creating a structure similar to the competition's test set. The model's top 2 accuracy is evaluated on this formatted validation set, and the triplet structure is maintained for validation loss assessment.

In this particular section, we will refrain from employing our model test set for the purpose of assessing the Siamese network. This is due to the fact that the model test set is derived from the same training data that we possess. Consequently, to address this concern, we implement adjustments to the validation procedure, as previously indicated. It is also worth noting that In order to do inference, we specify a separate model which takes in a pair of images and passes each through the encoder portion of the trained Siamese network to generate encoded feature vectors. It then computes the cosine similarity between these encoded vectors to evaluate the similarity of the two input images. This model allows comparing new pairs of

images using the encodings learned by the Siamese network without any additional training.

Figure 10 illustrates the progression of the triplet loss over multiple epochs for both the training and validation triplets. In the training phase, the triplet loss demonstrates a notable convergence towards zero after only two epochs. During the validation phase, the triplet loss exhibits a consistent decline up until the fourth epoch. Subsequently, it maintains a relatively stable value between the fourth and fifth epochs.

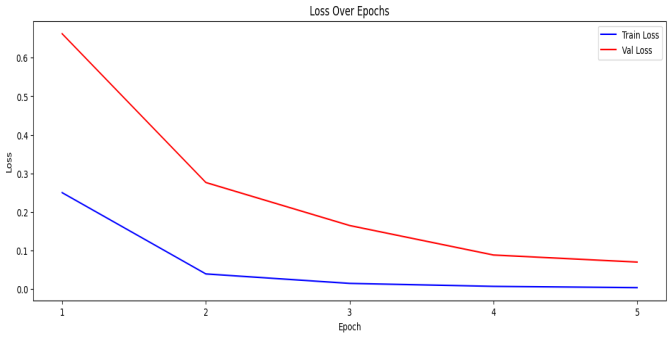

Fig. 10. Training and Validation loss Over Epochs

Figure 11, On the other hand illustrates the top 2 accuracy on the formatted validation set. The inference model encodes each left image and its 20 candidates through the Siamese encoder and compares their cosine similarities to predict the top 2 matches. As can be seen, the accuracy experiences a sharp increase between the first and second epoch, followed by a consistent upward trend.

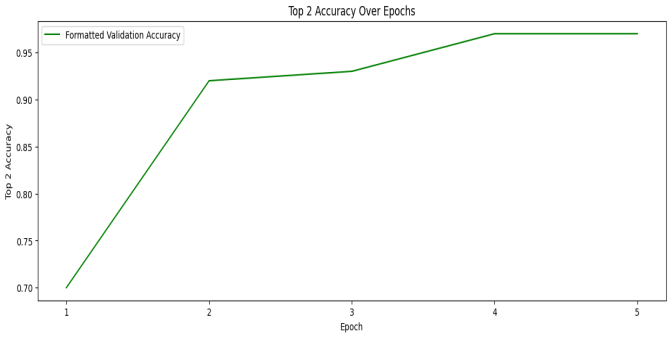

Fig. 11. Accuracy of Formatted Validation Set Over Epochs

In the end, Our model achieved an accuracy of 47% on the competition dataset, representing a considerable improvement compared to previous approaches. However, the variance in accuracy between the validation set and the competition dataset may suggest that the formatted validation set aligns with the training data distribution, while the competition dataset may exhibit variations and distinct data characteristics that impact performance. Thus, there seems to be an evident opportunity for further enhancement through meticulous tuning of both data and model hyperparameters. As an attempt to achieve this goal, we conducted several experiments aimed at pushing accuracy higher.

Training on unmodified triplet data yielded 42.8% accuracy, indicating the model struggled to learn effectively, prompting the dataset modifications we made. Attempting to increase difficulty by ensuring high similarity between positive and negative images surprisingly reduced accuracy to 30%, demonstrating the ineffectiveness of this technique. Finally, we explored modifications to the structure of the Siamese network's base model. However, we faced limitations due to computational constraints on the GPU resources and inference time available in Colab. For instance, one approach involved removing all layers following the base Xception model and increasing the number of unfrozen layers to 32. Despite not being limited by resources, this approach it did not yield satisfactory results, with a final accuracy of only 43.2%. Another strategy we attempted was to unfreeze all layers in the Xception base model and remove any leading layers. This approach, however, led to runtime crashes due to high RAM usage before results could be measured. These experiences underscore the potential for further accuracy gains through hyperparameter tuning on optimized infrastructure.

III. Conclusion

In this project, we addressed the Totally-Looks-Like challenge through three distinct methods: Canny edge detection, pre-trained model inference, and a Siamese Network employing triplet loss. While we achieved satisfying results, the progression of our outcomes was constrained by computational resource limitations. Expanding our model tuning efforts with additional computational resources holds the promise of significantly enhancing our results.

References

[1] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 6, pp. 679–698, 1986.

[2] M. Ringnér, "What is principal component analysis?," *Nature Biotechnology*, vol. 26, pp. 303–304, Mar 2008.

[3] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[4] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

[5] G. Koch, R. Zemel, R. Salakhutdinov, *et al.*, "Siamese neural networks for one-shot image recognition," in *ICML deep learning workshop*, vol. 2, Lille, 2015.