Similarity Detection Models Through Human Lens

Lang (Ron) Chen*
Student ID: 1181506

Anhui Situ*
Student ID: 1173962

***Abstract*—Computer Vision's progress has produced advanced Deep Architectures which beats humans on rigid tasks like classification, but human vision systems and the way humans process visual signals are much richer than the traditional objectives Computer Vision models train to achieve. In this work, the way humans detect similarity is modelled using a Siamese Network with Triplet Learning to solve the Totally-Looks-Alike similarity classification problem, proposing a model with 61.7% top2-accuracy when classifying similarity over 20 candidate images and building towards more human-like CV-ML applicable to robotics and digital content monitoring.**

***Index Terms*—Computer Vision, Contrasitive Learning, Triplet Learning, Machine Learning**

I. Introduction

The astonishing development of Deep Learning in the field of Computer Vision (CV) has seen Convolutional Neural Networks (CNNs) surpass human performance on tasks such as image classification, with these incredible developments advancing works in engineering fields such as robotics and automatic driving. However, beating humans on tasks with rigid and clearly defined labels still holds CV models a long way off from being close to 'human-like', a gap which the field of Natural Language Processing has recently been able to narrow with the inception of ChatGPT3.5. Humans, with brilliant minds and creative imagination, can interpret what they see in more ways than just a strict label - for example seeing similarities in two totally unrelated objects which classification CV models will be trained to interpret as two different classes.

Successfully modelling how humans interpret visual similarity is one of the tasks which can instill 'humanness' into robots, bringing closer to reality the prospect of truly human-like robots. More practical applications for CV similarity detectors include automatic detection of well-disguised threatening or abusive content online, which are increasingly doctored to be more abstract to bypass the surveillance of traditional classification models (i.e. visual content which violate platform policies posted in comment sections constructed by using characters as pixels).

The Totally-Looks-Alike (TLA) Image Similarity Scoring problem is one which leads solution CV models to learn how humans judge similarity. The objective is to give likelihoods to each of 20 candidate images for each query image, and score the correct match to be the highest or second highest scoring. In this work, a Siamese Learning framework using an embedded Densenet CNN encoder is proposed as the solution, with a 61.7% top2 accuracy performance on the TLA data-set.

II. Methodology

A. Convolutional Neural Network and Transfer Learning

Deep Learning is a widely-applicable model framework for CV problems, with State-of-the-Arts Convolutional Neural Network able to stack multiple Convolutional Layers that each automatically extracts a wide range of features from the data while optimising for a certain task. Notable networks include Res-net [3] - which uses residual layers to allow the network to stack hundreds of layers while bypassing the vanishing gradient problems, GoogLeNet [7] - which extracts features at multiple scales with an Inception module that contains kernels of different sizes, and DenseNet [4], which densely connects the output of each convolutional layer onto future layers as inputs to encourage reuse of low-level features from earlier layers at higher levels. These architecture often have inbuilt mechanisms that prevents overfit, thus allowing for tens or even hundreds of convolutional layers to be stacked without losing generalisability on unseen data. The same model with different depth are referenced distinctively by adding an integer after the model name (i.e. 'Densenet201'), with deeper networks having greater feature extraction capacity.

CNNs can be used on other CV tasks by considering the Convolutional Layers as a collective module embedded into bespoke networks. Further, a 2012 paper [5] demonstrated that it was beneficial to task performance to embed the Convolutional Layers from a CNN trained on large amounts of data for one CV task into a network for another task, and then fine-tuning the latter with data for that problem. Most commonly, the original CNN would be trained to classify the 'ImageNet' data-set, with this feat - commonly deemed Transfer Learning - allowing models to effectively learn from more data than the image data curated for the specific problems (typically much, much less than ImageNet). In this work, we use **Densenet201** with **weights pretrained on ImageNet as our model architecture's embedded CNN encoder**. The **pretrained final linear layer is decapitated**, as per common practice in Transfer Learning, and **all the CNN weights are subject to fine-tuning**.

B. Contrastive Learning and Triplet Loss

Siamese Networks is an architecture for semi-supervised similarity learning, under Contrastive Learning [2]. Siamese Networks takes in pairs of images, and under a specific definition of distance tries to maximise the gap between 'dissimilar' pairs while minimising 'similar' ones. In essence, it is trying to map images to one high-dimensional embedding space

where similar images have smaller distance and dissimilar images have bigger distance. Common distances are Euclidean Distance or Cosine Similarity.

Though traditionally optimised with the contrastive loss which is a function on two embeddings, Contrastive Learning more recently advanced to using the Triplet Learning framework[8], whereby each training input is a triplet of an anchor image, a similar image (positive) and a dissimilar image (negative), and uses the triplet loss which repels the distance between anchor and positive to be smaller than that of anchor and negative beyond a positive margin:

$$L(E_a, E_p, E_n) = \max(0,\, d(E_a, E_p) - d(E_a, E_n) + margin) \tag{1}$$

where $margin > 0 \in R$ is a model hyper-parameter, $d(E_a, E_p)$ is the distance between the anchor and positive, while $d(E_a, E_n)$ is that between anchor and negative.

The framework for the solution architecture use Contrastive learning with Triplet Loss. Given the nature of the data in TLA, we **also propose an alternative Double Triplet Loss** which will be experimented alongside Triplet Loss:

$$\begin{aligned} L(E_a, E_p, E_n) = &\max(0,\, d(E_a, E_p) - d(E_a, E_n) + \text{margin}_a) \\ &+ \max(0,\, d(E_a, E_p) - d(E_p, E_n) + \text{margin}_p) \end{aligned} \tag{2}$$

where two separate hyper-parameters $margin_a$ and $margin_p$ exist to give different importance to the separation of negative to anchor and to positive.

C. Siamese Network Architecture

The architecture used for this work is a Siamese network, which consists a twin embedding structure where weights are shared so all images are projected onto the same field (see figure 1). Mathematically, the Siamese architecture used in this work is as follows:

$$E_a, E_p, E_n = \text{Siamese}(x_a, x_p, x_n); \tag{3}$$

$$E_a = \text{OutMLP}(\text{LinearLayers}(\text{CNN}(x_a))); \tag{4}$$

$$E_p = \text{OutMLP}(\text{LinearLayers}(\text{CNN}(x_p))); \tag{5}$$

$$E_n = \text{OutMLP}(\text{LinearLayers}(\text{CNN}(x_n))); \tag{6}$$

where x_a, x_p and x_n are the anchor, positive and negative images, E_a, E_p and $E_n \in \mathbb{R}^{EmbeddingDimension}$ are embeddings, CNN is the embedded Densenet201 encoder architecture, OutMLP is an MLP that changes the hidden dimension to the chosen embedding size, and LinearLayers are stacked individual layers that takes in the flattened output of the last CNN layer defined as:

$$LinearLayer(X) = Dropout(\sigma(MLP(X)) \tag{7}$$

where Dropout is the dropout layer, σ is the Rectified Linear Unit (ReLU) activation function.

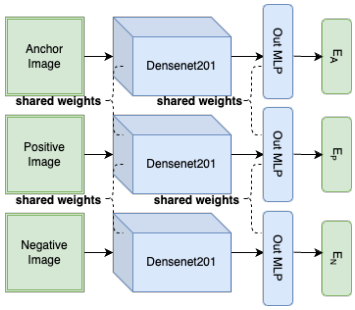


Fig. 1: Siamese Network Architecture

D. Training Scheme and Hard Negative Mining

As the data (see subsection III-A) contain a list of pairs of similar image with no internal ordering within the pair, **all images from both sides of the pairing are able to be used as anchors** with their counterpart as a natural positive. As triplet loss is not symmetrical (i.e. the positive and negatives don't repel each other), it is important for both images to be used as anchors to stabilise their position within the plane, and also be able to repel their own dissimilar images' embedding away rather than relying on their counterpart to do so on their behalf.

The pairs in the **data-set are first split into proportions of 70%-15%-15% for training, validation and testing respectively**. When sampling negative instances for training or for empirical evaluation of the validation and test set (see section II-D), **sampling are restricted to be strictly within-set** to prevent data leakage leading to inaccurate estimations of model performance.

The way negative samples are selected can change the convergence of the model. The most basic form of sampling negative instances is random selection, but other works [1] have identified tricks to improve this by sampling the negative samples with smallest distance to each anchor image after each epoch or batch, with benefits being the forced separation of difficult samples to yield a better discriminative boundary which betters generalisability. The 2015 FaceNet paper [6] introduced an alternative scheme of semi-hard sampling where the most similar negative instances are used provided that they are further away than the positive, with one benefit being greater training stability. It also mixed randomly sampled negatives along with semi-hard instances to further this effect.

For this problem, **negative mining is performed such that in the first epoch negative images are sampled randomly**; while **from the second epoch onward hard-negative or semi-hard negative mining schemes are employed** by first producing all embeddings of images using the model from this epoch, before calculating every pair-wise distance and than re-ranking and re-sampling negatives for every image.

The model was optimised using the **AdamW optimiser**, and the optimal hyperparamters are set to the combination that produces the validation set's top-2 accuracy (with 19 randomly sampled candidates for each validation set pair) from the model trained on the training set. **While typically the**

(a) Map of an Island and Dragon, sharing similarities in their shape

(b) Cartoon Figure and Man, sharing similarities in their beard

Fig. 2: Examples of pairs of similar images in Data-Set

model's loss on the validation set is used to control early stopping and re-scheduler, for this problem the validation score will be used, because the optimum of the loss and the accuracy may be mismatched due to the evaluation metric being top2accuracy rather than top1. The test score on the early-stopped optimal-hyper-parameter model will be used the final estimation of model performance on unseen data.

E. Preprocessing

All images were loaded in in their original dimensions (245, 200, 3), and each pixel is **normalised to between [0, 1] by dividing by 255**.

III. Results

A. Data-set

The data-set used in to train models is scraped from a forum where users post pairs of images they believe look alike despite this relationship generally considered to be unthinkable - unexpectedly curating data that captures human perception of similarity, with two examples presented in figure 2. The data-set consists 2000 pairs of similar images, making it semi-labeled but classless.

B. Evaluation

The evaluation metric for this task is the top2accuracy when a 'left' image is to be matched up with 20 'right' images, meaning that if the predicted pseudo-likelihood of the correct 'right' image is within top 2 out of 20, then similarity detection is considered accurate.

When performing inference on the validation and test set, 19 non-similar images are randomly sampled for every pair to produce a 'one left 20 right' setting, and the pseudo-likelihood is taken to be softmax($\frac{1}{d+\epsilon}$) where ϵ is a small constant.

C. Model Optimisation

The model was optimised on the hyper-parameters displayed in table I.

D. Results

The results for experiments are presented in table II. All results were obtained from experiments conducted on a single 12-core NVIDIA GeForce RTX 4090 24GB GPU on the AutoDL platform, using Python 3.9 and the PyTorch library.

Hyperparameter	Value
Hidden Dimension of MLP	512
Embedding Dimension	128
Loss Margin	1
Dropout	0.1
MLP Layers	2
Init Learning Rate	10^{-5}
Number of Triplets Per Pair	9
Batch Size	16
Patience	5

TABLE I: Optimal Hyperparameters

Model	Top2Accuracy
Triplet(Euclidean) Semi-Hard Negative Mining	0.617
Triplet(Euclidean) Random & Semi-Hard Negative Mining	0.553
Triplet(Euclidean) Hard Negative Mining	0.593
Triplet(Euclidean) Random Negative Sampling	0.553
Triplet(CosSim, Margin=0.2) Semi-Hard Negative Mining	0.583
DoubleTriplet(Eucliean, MarginA=1, MarginP=0.5) Semi-Hard Negative Mining	0.573

TABLE II: Densenet201 Siamese Network Results

IV. Discussion

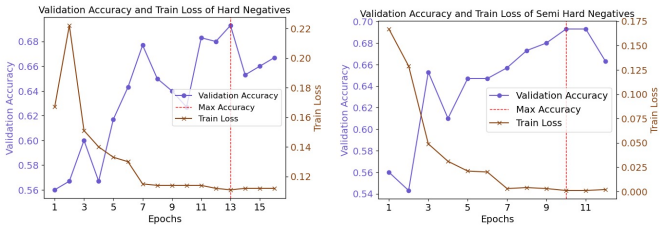
From table II, Semi-Hard Negative Mining with Triplet Euclidean Loss demonstrated the top performance overall out of the different losses and training tricks that were attempted. The second best performing was Hard Negative Sampling (Triplet Euclidean Loss). For the same loss, Semi-Hard Negative Sampling with Random Samples performed third and Random Sampling worst. This shows the effectiveness Soft Negative Sampling for this problem.

Triplet Loss with $1 - CosSim$ as the difference function performed worse than Triplet Euclidean Loss, a rather peculiar result as this contradicts the curse of dimensionality for Euclidean Distance operating on 128 dimensional space; one hypothesis for this observation is that the majority of the training set are faces, so vector embedding would fall in similar parts of the field and hence the direction vector from the plane's origin to each embedding are not sufficient to separate images.

Other CNN frameworks were trailed as encoders in less formal experiments but were abandoned as they did not perform nearly as well as Densenet201.

A. Triple loss and Hard Negatives Sampling

As discussed, all forms of non-random negative sampling gave better performances random negative sampling. The reason for this lies within the definition of Triplet Loss - when the distance between the anchor and positive is smaller than the distance of the anchor to negative by a threshold (which is the margin), the loss of this instance is 0 and it contributes no impact to the gradient and hence the training. Thus, after the first few epochs, it is likely that only few of the sampled negatives trigger a positive loss that trains the network. Meanwhile, selecting negative samples that will have the greatest chance of having a non-zero gradient will enforce learning in every epoch, as well as recursively pushing away dissimilar images that are close to each other to converge at



(a) Hard Negative Mining (b) Semi-Hard Negative Mining

Fig. 3: Training Loss and Validation Accuracy over epochs in different Negative Mining Strategies

a better discriminative boundary, which in random sampling would only be achieved if these highly impactful instances were sampled by chance.

However, a downside of mining the hardest negative is that they move the gradients too much in each epoch, and that some dissimilar images are noisy and there might exist images that are not of the same pair but realistically look alike (i.e. same image appearing twice in different pairs in training set). Semi-hard negative sampling solves both problems by selecting the closest pairs conditioned on having a greater distance than the positive, which means the loss is guaranteed to be less than the margin and also if the same image appears twice or two unpaired-images naturally look-alike in the data-set, they will not be sampled as those instances' euclidean distance is guaranteed to be smaller than that of the anchor and positive. As seen in figure 3, the validation accuracy of the semi-hard negative mining grows more steadily than hard-negative mining's validation loss and also converges faster at 10 epochs.

B. Double Triplet Loss

Regrettably, the attempted novelty in the problem-tailored Double Triplet Loss function did not enhance the performance over Triplet Loss. The intuition behind the loss defined by equation 2 was to attempt to penalise the closeness of the negative embedding to the positive as well as the closeness of negative and anchor, as it is assumed each pair of images in the 'data label' are uniquely similar to each other, so repelling negatives from the positive makes sense. The smaller $margin_p$ is designed so the gradients are still focused on separating the anchor from negatives. Nonetheless, this failure could be due to positives not always being the ground-truth closest image in the whole data-set to the anchor, as the data-set wasn't curated strictly abiding by the aforementioned assumption.

C. Error Analysis

An analysis on two example pairs that failed to be classified as top-2-similar (see figure 4) allows qualitative examination of model performance. In the first example, the ground-truth similarity is in the anchor's hair showing similar shape to right's noodles. However, as our dataset was dominated by face matching pairs, unsurprisingly one of the top2 predictions was a mole with afro as the highly similar hair features were matched. The second example highlighted that skin colour

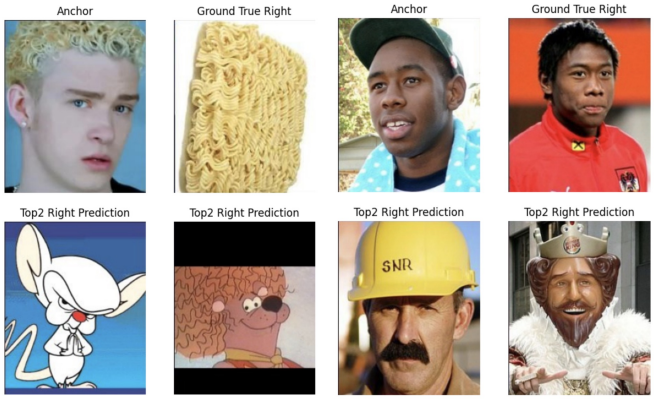


Fig. 4: Examples of Failed Top2 Similarity Detection

seemed to be a key feature for recognising similarity, as the top 2 predicted similarity had dark skin naturally or were tanned. This was also the case in the first example where the one incorrect prediction was a white mouse which matched the pale skin of the man in the anchor. This raises concerns for racial bias in the model which would need to be investigated further before real-world deployment.

Nonetheless, the ground truth of the mis-classified pairs have very obscure similarity even to a human, so the model is still rather well-performing despite the 38.3% error rate.

V. Conclusion and Future Works

Though this work has performed with 51.7% higher top2-accuracy compared to the random baseline (10%), the framework is not without limitations, including but not limited to the aforementioned potential ethnicity bias.

No pre-possessing or augmentation were capable of improving the performance of models, despite repeated experiment attempts. We have experimented offline pre-processing, including adjusting contrast and applying gaussian blur, and online augmentation applied in mini-batch such as random rotation, horizontal flip and shift. It is believed that this is due to the vastly different visual similarities that humans can see and also the already-abstract nature of the images, any single pre-processing or augmentation scheme would take out important signals for comparison between some pairs. However, the use of feature extractors overlayed on the original image may be an area of research that can produce enhancements on the existing architecture.

Though our attempted designed loss failed, manipulation of losses and also potentially to the architecture (including but not limited to introducing interaction between the images before the last layer) could yield enhancements.

In summary, this work proposed a solution to similarity modelling using Siamese Architecture and Triplet Learning and trained with Semi-Hard Negative Mining, producing 61.7% performance on top2-accuracy over 20 candidate images for the TLA problem; a small steps towards CV models that has greater human qualities.

References

[1] Ross Girshick et al. “Rich feature hierarchies for accurate object detection and semantic segmentation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2014.

[2] Raia Hadsell, Sumit Chopra, and Yann LeCun. “Dimensionality Reduction by Learning an Invariant Mapping”. In: *In Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)* (2006), pp. 1735–1742.

[3] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016).

[4] Gao Huang et al. “Densely Connected Convolutional Networks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017).

[5] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Advances in Neural Information Processing Systems* 25 (2012).

[6] Florian Schroff, Dmitry Kalenichenko, and James Philbin. “FaceNet: A Unified Embedding for Face Recognition and Clustering”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015).

[7] Christian Szegedy et al. “Going Deeper with Convolutions”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition* (2015).

[8] Eric Taigman et al. “Learning a Similarity Metric Discriminatively, with Application to Face Verification”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2014.