Performance of Various Loss Functions for Siamese Neural Networks on Tiny-ImageNet

Abstract

Siamese neural network is effective for tasks involving image verification, where two images are identified to belong in the same category or not by passing through identical networks with shared weights. Popular loss functions for these parallel networks include binary cross-entropy, contrastive and triplet loss. In this paper, these three loss functions are implemented to verify 20 classes of images from Tiny ImageNet, based on a pre-trained ResNet50 model. Each model is first tuned by testing various embedding dimensions. Loss is investigated along with extent of clustering of images in embedding space to explain the difference in those loss functions. Experiments shows that binary cross-entropy loss performs best and that contrastive loss result in tighter clusters in the embedding space compared to triplet loss, which correlate well with mathematical interpretations of these loss functions.

1. Background

Siamese neural network is found to be a good architecture for image verification tasks such as signature verification [1] and face recognition [2]. A Siamese network generally includes two or three neural networks with shared weights. Pairs or groups of inputs each pass through a neural network and the output is processed by an energy function or loss functions to find the relationship between inputs. For the task of image verification where images are classified as belonging to the same category, or different categories, the output of each neural network can be regarded as an embedding. The concept of embedding is used commonly in language processing [3] and image processing [4] where a high dimension input is reduced to a lower dimension representation. Embedding outputs in Siamese networks are commonly processed by using a distance metric to find degree of similarity between inputs. Similar inputs, or inputs from the same class would have short distance between them and vice versa. These inputs exist in a manifold and the task of the neural network is to learn the manifold representation. There are various ways to construct loss function for learning. Binary cross-entropy, contrastive and triplet loss are three most popular loss functions used in Siamese networks.

Binary cross entropy loss used in the context of Siamese network is first proposed for the task of one-shot learning, where predictions is made based on a single example of a new class [5]. Similarity between inputs is ranked by weighted distance between two embeddings that connects to a sigmoid activation. Using binary labels, error is calculated with binary cross-entropy. This simple method allows models to generalize well to new data of unknown classes.

Similar to binary cross-entropy, contrastive loss uses pairs of images with binary labels; either an anchor image and a positive image of the same class, with label indicating the same class, or an anchor image with a negative image of different class, with label indicating so. The difference is that contrastive loss uses distance between embeddings directly to calculate loss, without weighted distance or sigmoid activation.

While contrastive loss uses pairs of labelled images, triplet loss uses a group of three images as input. An anchor image and another image from the same class as a positive image, as well as a negative image from a different class. The triplet loss aims to decrease the distance between anchor and positive embedding and increase the distance between anchor and negative embedding so that difference in these two distances is greater than a margin α.

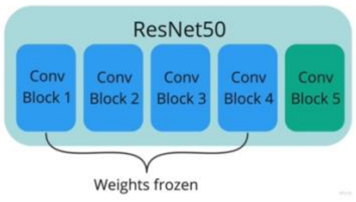
The purpose of all three loss functions is to pull embedding of images of the same class together and separate clusters of different classes so that embeddings of images of different classes are far apart. This allows generalizability of Siamese network which is data driven and can perform well to unseen categories without the need for engineered features like in some previous image recognition works [6][7].

There are many possibilities for the choice of architecture for network backbone. Convolutional neural network is a natural choice since convolution operation is shown to be effective in extracting increasing scale of features from images and are rotation and scale invariant [8]. The task in this paper is to verify classes of objects from Tiny ImageNet, which has 200 classes of images, each with 500 different images. To avoid underfitting, a deep model preferred. However, there is limited computing resources to train a model from scratch with a large amount of data. Thus, transfer learning is a better choice to improve performance of a model in a specific domain by transferring the features learned in a related domain [9]. By fine-tuning pre-trained models, training time is shortened, and less training data is required compared to training an identical model from scratch [10]. Tiny ImageNet is a subset of ImageNet, which makes models pre-trained on ImageNet good candidates [11]. Models trained on ImageNet include examples such as AlexNet [12], VGG [13] and ResNet [14]. This paper uses ResNet50 as pretrained model since residual connections reduce issue of vanishing gradients for deep networks and allows the model to find simpler mapping if necessary. This is pertinent since Tiny ImageNet has smaller and arguably simpler images than ImageNet, and a small subset of Tiny ImageNet is used. Thus, a simpler model might suffice.

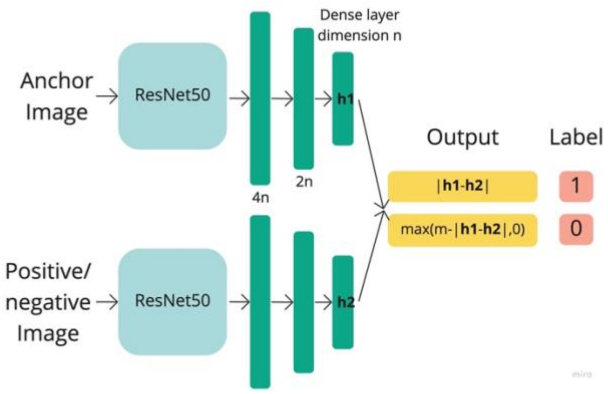
[15] found that first three layers AlexNet contain generalizable features, beyond which the features are more specific to the source dataset. This paper uses pretrained ResNet50 for transfer learning with the first 4 convolution blocks frozen and last convolution block fine-tuned. Furthermore, it is common to add fully connected layers and regularizers to reduce uncertainty of predictions [9]. Thus, this paper adds three dense and batch normalization layers after ResNet50 backbone.

Despite large amount of research on transfer learning using ImageNet trained models, and various improvements on classic Siamese network models, there is lack of comparison of binary cross-entropy, contrastive and triplet loss functions, specifically tested on Tiny ImageNet. Motivated by this gap in literature, this paper aims to evaluate these three loss functions, using a model pretrained on ResNet50 and tested on Tiny ImageNet. This paper will first talk about image pre-processing procedures, model architecture design, then model tuning process. Lastly, comparison between loss functions will be made in terms of mathematical difference that results different embedding.

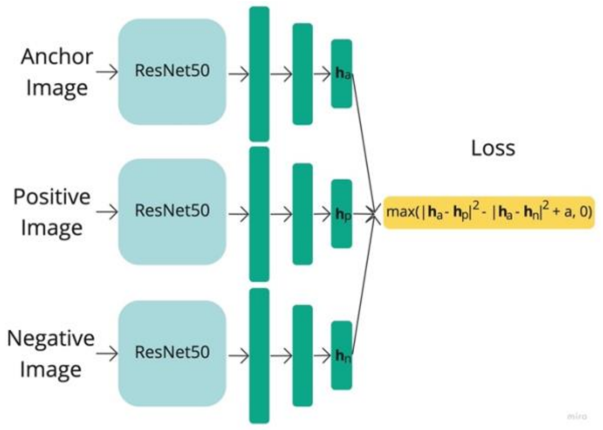
Experiments shows that binary cross-entropy loss performs best with 69.32% validation accuracy and that contrastive loss result in tighter clusters in the embedding space compared to triplet loss, which correlate well with mathematical interpretations of these loss functions.



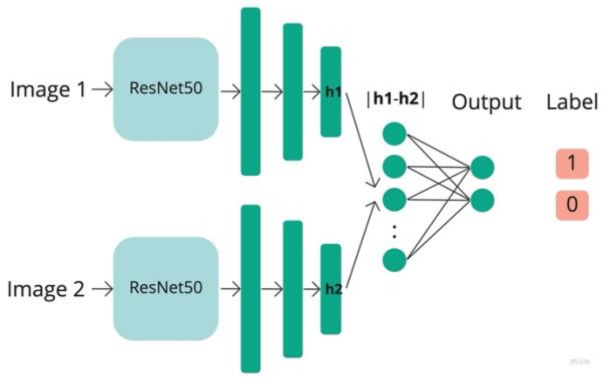
(a) ResNet50



(b) Contrastive loss function



(c) Triplet loss function



(d) Binary cross-entropy loss function

Figure 1. Various architectures used in this paper. (a) All models use ResNet50 as a backbone. Weights of first 4 convolution blocks are frozen. Weights in the last convolution block are fine tuned. (b,c,d) Two or three back bones are used. 3 additional fully connected layers follow ResNet50, each dimension is half of the previous.

2. Method

2.1. Image pre-processing

Out of the 200 classes of objects in Tiny ImageNet, 20 random classes are used for training. Not all classes are used due to limited computing resources. For binary cross-entropy and contrastive loss, 1000 pairs of images are generated. 500 pairs of images of the same class with label 1, and 500 pairs of different classes with label 0. For triplet loss, 500 groups of images are generated with two images of the same class and one of a different class. For any pair of images from the same class, two random images are drawn out of all the 500 hundred available images. For a pair from different classes, a random image is drawn from the baseline class, and another randomly from 1 of 19 other classes. Thus, training images come from a pool of 20,000 images. Training and validation split is 8:2 meaning 16,000 images are used for training and 2,000 for testing. A further 5 random classes are used for testing. Testing set all come from these 5 random classes. They are not grouped with training classes. Thus, the classes in testing images have never been seen during training.

Each image is upsampled to be 200x200 in order to fit ResNet50 better, since ImageNet generally contains larger images. Each image is then converted from RGB to BGR. Each colour channel is zero-centered with respect to ImageNet, which is done through tenforflow's preprocess_input function [16]. No augmentation is done since enough input data is available and overfitting is not likely.

2.1 Model baseline

Pretrained ResNet50 is used as a baseline model. ResNet50 has 5 convolution blocks. Other than the first block, each block has 3, 4 or 6 convolution units. Each unit is made up of 1x1 and 3x3 kernels. The weights in the first 4 convolution blocks are frozen. The last block is trained with 3 additional dense layers with $4n$, $2n$ and n nodes with Relu activation. Between each dense layer is a batch normalization layer. The baseline model takes in a 200x200 image with 3 channels, and output a n

dimensional embedding on top of which various loss architectures are implemented. All three architectures are visualized in Figure 1. Various embedding dimensions are tested in order to find the best dimension for each loss function.

All models tested in the following experiments use learning rate 0.0002, and are trained until validation accuracy stops increasing.

2.2 Binary cross entropy loss

The binary cross entropy loss used in this paper follows the architecture proposed by [5]. The author used weighted distance between two embeddings that connects to a sigmoid activation so that the output is between 0 and 1. Thus binary labels can be used directly for back propagation. The weighted distance is learned as a dense layer that connects the output node and the distance between embeddings. Loss is given by

$$L = y log P(x_1, x_2) + (1 - y)\, log\big(1 - P(x_1, x_2)\big)$$

$$P(x_1, x_2) = \sigma(\boldsymbol{w}^T |\boldsymbol{h}_1 - \boldsymbol{h}_2|)$$

where σ is the sigmoid activation function. $\boldsymbol{w}$ is the weight that connects the second last layer to the final scalar output. y is binary label where 1 represent two images of the same class; 0 represent different classes. $\boldsymbol{h}_1$ and $\boldsymbol{h}_2$ are output embedding vectors. $|\boldsymbol{h}_1 - \boldsymbol{h}_2|$ is the L_1 distance between them.

$$|\boldsymbol{h}_1 - \boldsymbol{h}_2| = (|\boldsymbol{h}_1^1 - \boldsymbol{h}_2^1|, |\boldsymbol{h}_1^2 - \boldsymbol{h}_2^2|, \cdots, |\boldsymbol{h}_1^N - \boldsymbol{h}_2^N|)^T$$

N is the embedding dimension.

2.3 Contrastive loss

Contrastive loss is originally proposed by [17] to solve the general problem of learning a globally coherent nonlinear function that casts similar high-dimension inputs to close points on a low-dimension embedding manifold. The nonlinear function can be a neural network and high-dimension input can be images. Loss is given by

$$L = y\|\boldsymbol{h}_1 - \boldsymbol{h}_2\| + (1 - y) max(0, m - \|\boldsymbol{h}_1 - \boldsymbol{h}_2\|)$$

where m is the margin between two embeddings and set to 1. $\|\boldsymbol{h}_1 - \boldsymbol{h}_2\|$ is the L_2 or Euclidean distance between $\boldsymbol{h}_1$ and $\boldsymbol{h}_2$

$$\|\boldsymbol{h}_1 - \boldsymbol{h}_2\| = \sqrt{(\boldsymbol{h}_1^1 - \boldsymbol{h}_2^1)^2 + (\boldsymbol{h}_1^2 - \boldsymbol{h}_2^2)^2 + \cdots (\boldsymbol{h}_1^n - \boldsymbol{h}_2^n)^2}$$

Similar images have low loss if label is 1; images of different classes have low loss if label is 0. The difference from binary cross-entropy loss the lack of logarithm taken over the Euclidean distance between two embeddings and addition of a max function which serves as Relu activation.

2.4 Triplet loss

Triplet loss used in this paper is proposed by [18]. It is originally created for the task of face verification, where the model needs to recognize if two faces belong to the same person. A certain threshold classifies the pair, below which two faces are the same person, and above which two faces are distinct. The task of verifying images of various objects is similar to face verification. Triplet loss trains different images of the same object to occupy embedding spaces close to each other. This is different from contrastive loss where images of the same object are encouraged to occupy the same point in embedding.

To implement triplet loss, three images are used, an anchor a, a positive image p belonging to the same class as the anchor, and a negative image n, from a different class. Loss is given by

$$\mathrm{L} = \max\left(\|\mathbf{h}_\mathrm{a} - \mathbf{h}_\mathrm{p}\|^2 - \|\mathbf{h}_\mathrm{a} - \mathbf{h}_\mathrm{n}\|^2 + \alpha, 0\right)$$

where α is the threshold that separates same or different class classification. α is set to 1. $\mathbf{h}_\mathrm{a}, \mathbf{h}_\mathrm{p}$ and $\mathbf{h}_\mathrm{n}$ are the embedding of anchor, positive and negative images.

Performance of each model is assessed by their testing and validation accuracy. For binary cross-entropy and contrastive loss, the output of the loss functions can be compared to corresponding binary label directly to obtain accuracy. However, triplet loss does not use labels and do not output anything that can be used directly for prediction. Thus, simple logistic regression is added for inference. After training is finished, images are paired up again to each go through the model backbone to output two embedding values. Euclidean distance between them is stored along with corresponding label. Logistic regression is used to maximize the probability that prediction from Euclidean distances matches label. The loss function then becomes exactly the same as binary cross-entropy function. Thus, two training sessions are required for triplet loss. Firstly, train the backbone by using triplet loss, then train a logistic head with binary cross-entropy loss. Similarly, for validation and testing, pairs of images go through fine-tuned ResNet50 and dense layers to produce Euclidean distances between them. This distance pass through trained logistic head to produce binary labels.

2.5 Embedding distance

All three loss functions aim to maximize distance between different images and minimize distance between similar images. However, the way they do so is different. Contrastive loss does so by pushing images of the same loss toward the same point so that $y\|\boldsymbol{h}_1 - \boldsymbol{h}_2\|$ is minimized. Triplet loss is more forgiving. If there is sufficient distance between positive pairs and negative pairs, loss will be small. Thus, triplet loss allows more variance between classes and outliers in class clusters. Additionally, if the distance between positive pairs is far, triplet loss will not reduce this distance if no negative sample interfere with it, or if distance from negative sample is large enough to exceed the margin. Whereas contrastive loss always aims to reduce distance between positive pairs, and only uses the margin for negative pairs. Thus, a hypothesis can be drawn from this mathematical difference that embedding is clustered more tightly for contrastive loss than triplet loss.

To verify this, embedding for all training images for each loss function are produced from trained models. In each class, average location of all image embeddings is found, and the distance of each embedding to the average location. The average distance indicates extent of clustering for each class. If each cluster is treated as a ball, average radius is the average distance of each embedding to the centre. More tightly clustered classes would have smaller radius, and vice versa.

$$\text{Cluster radius} = \frac{1}{N}\sum_{i=1}^{N}\left\|\boldsymbol{h}_i - \overline{\boldsymbol{h}}\right\| \quad \text{, where } \overline{\boldsymbol{h}} = \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{h}_i$$

$N = 500$ is the number of images in each class. Cluster radius for each class is plotted for each loss function to compare their difference. Relationship between cluster radius and accuracy is also compared, since smaller cluster radius might result in larger distance between clusters and higher prediction accuracy. Accuracy for each class combines accuracy identification of positive pairs and negative pairs.

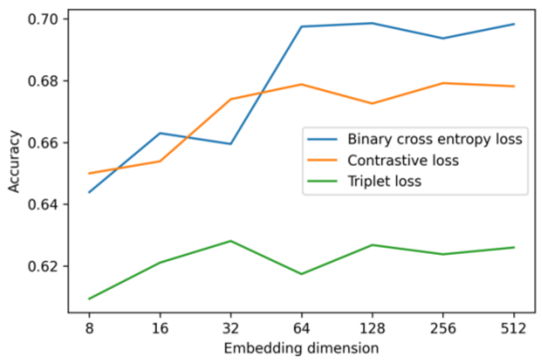
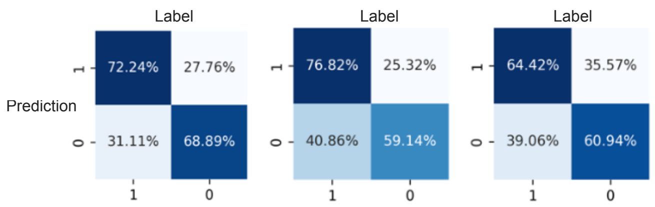


Figure 2. Output embedding dimension affects accuracy of each model.



(a) Binary cross-entropy loss (b) Contrastive loss (c) Triplet loss

Figure 3. True positive, false positive, true negative, false negative rate of each loss function.

	Embedding dimension	Training accuracy	Validation accuracy	Test accuracy
Binary cross entropy	64	0.6932	0.6986	0.5584
Contrastive	64	0.6700	0.6788	0.5432
Triplet	32	0.7124	0.6268	0.5257

Table 3. Summary of performance of each loss function

3. Results and evaluation

Firstly, model for each loss function is tuned by testing various embedding dimensions. Figure 2 shows how embedding dimension affects accuracy. For binary cross-entropy loss function, embedding dimension larger than 64 performs best. For contrastive and triplet, embedding dimension larger than 32 result in good performance. One can observe that once past a certain threshold, larger dimension does not yield further accuracy increase. For all loss functions, values in embedding larger than 64 dimensions become sparse, where many values are 0. Thus, the models do not fully utilize larger dense layers but use fewer nodes since 32 or 64 dimensions is enough to represent image embedding.

Figure 3 shows the of accuracy rate of each prediction against each label. All three models perform better in identifying same image class than different image class. This is especially severe for contrastive loss. Its true positive rate is the highest and true negative rate is the lowest of all three. High true positive rate can be explained by small distance between positive images, which contrastive loss do minimize. However, there might be not enough separation between classes, resulting in low true negative rate. Triplet loss allows larger distance between positive pairs, which may contribute to its low true positive rate. There is not enough separation between positive and negative pairs.

Table 1 shows summary of all three models. Test accuracy on unseen classes is a lot worse for all three models. However, binary cross-entropy and contrastive loss functions generalize well from training data to validation data. This is not the case for triplet loss. Training accuracy is much higher than validation accuracy.

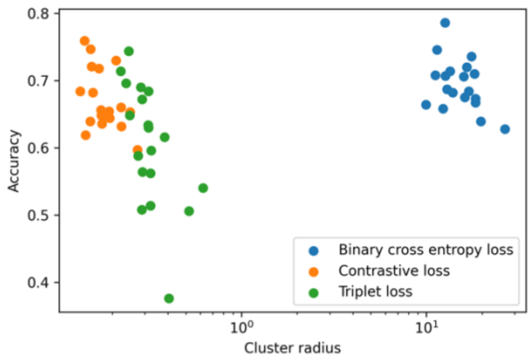


Figure 4. Plot of accuracy for each image class against class radius or cluster radius. Cluster radius is the average distance of each class's training image embedding's distance to the centre of the cluster.

Figure 4 shows that triplet loss has higher cluster radius than contrastive loss, which confirms the hypothesis that contrastive loss function's positive distance minimizing property reduces its cluster radius. In comparison, cluster radius for binary cross-entropy loss is a lot bigger. This is because sigmoid activation occurs after embedding difference is calculated. Thus, embedding can take large values. For contrastive and triplet loss, embedding distance is forced between 0 and 1 due to threshold being 1. Thus, their cluster radius is a lot smaller.

There is weak correlation between accuracy and cluster radius for each model. As seen in Table 2, cluster radius for best performing classes is lower than worst performing classes. However, there is no strong correlation

		Binary cross entropy loss	
		Accuracy	Cluster radius
Top accuracy	School bus	0.786	12.60
	Jellyfish	0.746	11.40
	Pizza	0.736	17.54
	Espresso	0.720	16.53
	Dining table	0.714	13.41
Bottom accuracy	Frying pan	0.667	18.42
	Cliff, drop	0.664	9.951
	Sock	0.658	12.29
	Umbrella	0.639	19.67
	Computer keyboard	0.628	26.53

		Contrastive loss	
		Accuracy	Cluster radius
Top 5 accuracy	Monarch butterfly	0.759	0.141
	Jellyfish	0.747	0.153
	School bus	0.730	0.209
	Pizza	0.721	0.154
	Espresso	0.718	0.169
Bottom 5 accuracy	Cliff, drop	0.639	0.151
	Sock	0.636	0.176
	Umbrella	0.632	0.224
	Egyptian cat	0.619	0.143
	Computer keyboard	0.597	0.273

		Triplet loss	
		Accuracy	Cluster radius
Top 5 accuracy	Tarantula	0.744	0.245
	Cliff, drop	0.714	0.221
	Monarch butterfly	0.696	0.237
	Sock	0.690	0.285
	Egyptian cat	0.684	0.314
Bottom 5 accuracy	Computer keyboard	0.540	0.618
	Umbrella	0.514	0.322
	School bus	0.508	0.288
	Rocking chair, rocker	0.506	0.520
	Frying pan	0.376	0.404

Table 4. Top and bottom performing classes for each loss function

4. Future improvement

Figure 3 shows that negative pairs are a lot harder to classify than positive pairs. This issue could be that same number of positive and negative pairs are used during training. However, more negative pairs are necessary to fully separate clusters. 500 pairs of positive images are used for each class and 500 negative images from 19 classes, which makes 26 negative pairs for each combination of classes. This is not enough to fully learn the distance between all combinations of negative pairs. Using more negative pairs for training could improve negative pair classification.

Cluster distance is chosen as a measure for extent of clustering. It can show inter-class distance but not intra-class distance, which is important for understanding negative pair classification. Difference classes should have larger distance between them, and more frequently misclassified classes may have more overlap with other classes. This study attempted to set embedding dimension to 2 for visualization. However, models failed to learn with such small dimension. In [17], the authors visualized embedding of MNIST digit 4 and 9 in 2D by training models on two classes of images. A similar model can be developed for future study where fewer classes consist of selected high accuracy and low accuracy classes are trained on 2-dimension embedding. Alternatively, t-SNE can be used to visualize high dimension data [19].

In this paper, first 4 convolution blocks are frozen and only the last convolution block is trained, along with additional dense layers. However, a study by [20] found that when fine-tuning AlexNet for Tiny ImageNet with fewer than 500 training examples per image class, it is better to freeze first 2 to 3 blocks and fine tune the rest. AlexNet contains 7 blocks which makes freezing less than half of the layers ideal. Furthermore, studies such as [21][22][23] show that fine-tuning all layers is better than fine-tuning only the output layers for in-distribution classification where the testing data is from the same distribution as training data. Since Tiny ImageNet is drawn from ImageNet, fine-tuning all layers should be the better choice according to literature. It is worthy to make this change in future studies given more computing resources. However, the aim of this paper is to compare and explain the differences in various loss functions. Thus, since the same baseline architecture is used, results for comparison is still meaningful.

Conclusion

This study aims to compare performances and explain the difference in binary cross-entropy, contrastive and triplet loss functions for image verification task. 20 classes from Tiny ImageNet are used for fine-tuning ResNet50 for transfer learning. Embedding dimension larger than 32 or 64 are ideal. Binary cross-entropy achieves highest validation accuracy at 69.32%. Contrastive loss achieves 67.88% accuracy and triplet loss achieves 62.68% accuracy. All models generalize poorly to unseen image classes. Inter class embedding distances for each class shows that triplet loss produces less tightly clustered embeddings than contrastive loss, and that smaller cluster radius generally correlates with higher accuracy.

References

[1] J. Bromley, I. Guyon, Y. LeCun, E. Sackinger, R. Shah. (1993). Signature Verification using a "Siamese" Time Delay Neural Network. International Journal of Pattern Recognition and Artificial Intelligence. pp. 737-744.

[2] Chatterjee, R., Roy, S., Roy, S. (2022). A Siamese Neural Network-Based Face Recognition from Masked Faces. In:

Woungang, I., Dhurandher, S.K., Pattanaik, K.K., Verma, A., Verma, P. (eds) Advanced Network Technologies and Intelligent Computing. ANTIC 2021. Communications in Computer and Information Science, vol 1534. Springer, Cham. https://doi.org/10.1007/978-3-030-96040-7_40

[3] Wang. C., Nulty. P., Lillis. D. (2020). A Comparative Study on Word Embeddings in Deep Learning for Text Classification. NLPIR. pp.37-46. https://doi.org/10.1145/3443279.3443304.

[4] Frome. A., Corrado. G.S., Shlens. J., Bengio. S., Dean. J., Ranzato. M., Mikolov. T. (2013). DeViSE: A deep visual-semantic embedding model. NIPS'13, vol. 2, pp. 2121-2129

[5] G. Koch, R. Zemel, R. Salakhutdinov. (2015). Siamese Neural Networks for One-shot Image Recognition. Proceedings of the 32nd International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37.

[6] Y. Sun, X. Wang, and X. Tang. Deeply learned face representations are sparse, selective, and robust. CoRR, abs/1412.1265, 2014. 1, 2, 5, 8

[7] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. IEEE Conf. CVPR, 2014. 1, 2, 5, 7, 8, 9

[8] O'Shea. K., Nash. R. (2015). An Introduction to Convolutional Neural Networks. ArXiv e-prints.

[9] Zhuang. F., Qi. Z., Duan. K., Xi. D., Zhu. Y., Zhu. H., Xiong. H., He. Q. (2020). A Comprehensive Survey on Transfer Learning. Proceedings of the IEEE. pp. 1-34. 10.1109/JPROC.2020.3004555.

[10] Pan, S. J. and Yang, Q. A survey on transfer learning. IEEE Transactions on knowledge and data engineering, 2010.

[11] M. Huh, P. Agrawal, A. A. Efros. (2016). What makes ImageNet good for transfer learning? arXiv:1608.08614

[12] A. Krizhevsky, I. Sutskever, G. E. Hinton. (2012). Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems. pp. 1097–1105.

[13] K. Simonyan, A. Zisserman. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. http://arxiv.org/abs/1409.1556

[14] He. K., Zhang. X., Ren. S., Sun. J. (2016). Deep Residual Learning for Image Recognition. 770-778. 10.1109/CVPR.2016.90.

[15] Yosinski, J., Clune, J., Bengio, Y., Lipson, H. (2014). How transferable are features in deep neural networks? In: Advances in Neural Information Processing Systems. pp. 3320–3328

[16] TensorFlow v2.11.0, www.tensorflow.org/api_docs/python/tf/keras/applications/resnet50/preprocess_input. Accessed 12 Jan. 2023.

[17] Hadsell, R., Chopra. S., LeCun. Y. (2006). Dimensionality Reduction by Learning an Invariant Mapping. CVPR'06 10.1109/CVPR.2006.110

[18] F. Schroff, D. Kalenichenko, J. Philbin. (2015). FaceNet: A Unified Embedding for Face Recognition and Clustering. arXiv:1503.03832v3

[19] van der Maaten, L.J.P.; Hinton, G.E. (Nov 2008). Visualizing Data Using t-SNE. Journal of Machine Learning Research. 9: 2579–2605.

[20] Soekhoe. D., Putten. P., Plaat. A. (2016). On the Impact of Data Set Size in Transfer Learning Using Deep Neural Networks. 50-60. 10.1007/978-3-319-46349-0_5.

[21] S. Kornblith, J. Shlens, Quoc V. Le. (2019). Do better imagenet models transfer better? Computer Vision and Pattern Recognition (CVPR)

[22] X. Zhai, J. Puigcerver, A. Kolesnikov, P. Ruyssen, C. Riquelme, M. Lucic, J. Djolonga, A. S. Pinto, M. Neumann, A. Dosovitskiy, L. Beyer, O. Bachem, M. Tschannen. A large-scale study of representation learning with the visual task adaptation benchmark. arXiv, 2020

[23] K. He, H. Fan, Y. Wu, S. Xie, R. Girshick. Momentum contrast for unsupervised visual representation learning. Computer Vision and Pattern Recognition (CVPR), 2020.