Just-In-Time Detection of Unknown Classes Using Deep Learning

Vazgen Tadevosyan
Graduate Student
Department of Software Engineering
vt7313@rit.edu

Akshat Garg
Graduate Student
Department of Software Engineering
ag2193@rit.edu

Abu Islam
Lead Staff Scientist
CIMS Center for Remanufacturing and Resource Recovery
asigis@rit.edu

***Abstract*—Convolutional Neural Networks have demonstrated exceptional performance in various tasks by training on a restricted set of categories. However, when faced with a novel data class, the model tends to incorrectly misclassify it as one of the existing classes. This may not always lead to severe consequences, but in industries like healthcare, where detecting new viruses is vital, or in autonomous vehicles, where recognizing new or deteriorating road signs can impact driving, it can be critical. Moreover, some application systems necessitate rapid predictions to prevent bottlenecks, production interruptions, and unforeseen expenses. As a result, our work aims to enhance prediction speed while maintaining high accuracy. In this study, we address these issues by employing a Siamese Network with Transfer Learning that can effectively differentiate between known and unknown classes while offering quick prediction times.**

This work also examines the impact of Triplet Loss and Contrastive Loss functions on Siamese Network training. Additionally, our results show that Euclidean Distance is a superior distance metric compared to Cosine Similarity for the given task.

***Index Terms*—Open Set Classification, Deep Learning, Siamese Neural Networks, Transfer Learning, Contrastive Loss, Triplet Loss**

I. Introduction

Training a classifier model on all possible real-life classes is nearly impossible. A more realistic approach to this problem is Open Set Classifier (OSC), where a classifier correctly labels the known classes during training and unknown classes as a "rejected/unknown" class.

Using Siamese Neural Networks (SNN) [1] is one approach to solving this problem. SNN has two inputs processed in the same neural network parallelly. In the final layer, we compare flattened layers using Cosine Similarity or Euclidean Distance. In an ideal case, a model should locate different instances far from each other and identical ones closer. It performs better than traditional neural networks in particular when there are few instances of each class. SNN uses distance scores to determine whether two images are dissimilar to one another and can be used to compare novel instances with instances of the known classes.

Contrastive Loss is a popular loss function for Siamese Network training and is used to train instances that differ slightly from one another. It is put into practice by sending two images concurrently through the same network, and it is trained so that similar images are placed close together while dissimilar ones are placed apart from one another [22].

Triplet Loss is another widely used for Siamese Network and is applied in different domains such as image retrieval [2], [3], face recognition [4], [5], and other classification tasks [6], [7]. Before discussing the following paper, it is necessary to provide background information for Triplet Loss. Google implemented Triplet Loss for facial recognition in 2015 [8]. In this case, three inputs are given to the model: anchor, positive, and negative. An input for reference is the anchor. The same category as anchor input includes positive input. Negative input is not part of the anchor class; instead, it is a random class. We use the Triplet Loss function to increase the distance between the anchor and the negative sample while minimizing the distance between the anchor and the positive sample.

Our main area of interest is the automotive industry. It is crucial to efficiently sort the products on conveyor belts in order to maintain a smooth workflow and prevent bottlenecks. Every process will come to a stop if there is even the slightest delay, which will raise the operational costs. To accurately and quickly identify we will implement advanced machine learning techniques like Siamese Network with Contrastive Loss.

II. Related Work

A. Open Set Classification

Many classification models do not consider that new classes might be present in the test datasets [9]. However, in real-life situations, it is not always possible to train on all the possible classes we encounter in the real world, and hence want to classify these classes as unknown classes. However, the classification models out there misclassify those unknown classes as known ones. Hence, Open Set Classifier (OSC) is used to reject a novel input from classes not seen during the model's training [9]. For the study of this paper [9], they want to classify inputs belonging to the same distribution as the trained set and reject all others as "unknown." A classifier achieves this by finding the confidence of input. Paper [9] has approached this problem by separating it into two strategies, inference method, and space feature representation methods. They are discussed in section II A.1, A.2. To formulate the problem,

$$\widehat{y} = \begin{cases} argmax_k F(X) & \text{if } S(X) \geq \delta \\ K+1 & \text{if } S(X) < \delta \end{cases} \tag{1}$$

1) Inference methods: Pre-trained models were used since the models are fine-tuned, and no changes to training have to be performed, with only modification of how the network outputs are used.

- ***Confidence thresholding*** is a simple approach to OSC for threshold the output of the model by using the τ-softmax function to find the probability of each class.
- ***Distance metrics*** are based detection of out-of-distribution data using distance metrics. Euclidean Distance is the most common measure for OSC for small dimensional feature spaces containing many classes [9].
- ***One-class Classification*** is another technique for determining outlier data from the distribution.
- ***Extreme value theory recognizes*** novel inputs by getting probabilities of occurrence than the previous one.

2) Feature Space Representation Methods: This method alters the network's architecture, enabling better OSC performance. There are a few ways to do so,

- ***One-vs-rest Classifiers*** Softmax has an issue classifying outlier data because of normalization, detecting the wrong class. One-vs-rest eliminates this problem by replacing each class with a logistic sigmoid function [9], reducing the risk of the wrong classification by setting each class threshold value.
- ***Background class regularization*** is a prevalent technique for object detection algorithms [9]. A newer development in this method has shown that it can be used to determine if the dataset is uniformly distributed or not [9].

We will use the idea of Confidence thresholding in this work to detect Unknown Classes.

B. Siamese Network with Transfer Learning

High automated visual inspection performance, including handling defective, new, and unseen instances, is crucial for meeting industrial standards in many fields, such as railway, automotive product manufacturing, casting, and even health care. Besides classifying already known classes, handling anomalies and new instances are essential. In the paper [10], they were interested in deep learning approaches that are useful for developing a comprehensive framework that addresses some issues mentioned above. [10] proposed new Siamese Network architecture outperformed other similar Neural Networks. Their model's other key advantage is that it easily tackles unseen new instances without being re-trained. One of the two datasets they used is relative to our project, a dataset of metal disk-shape castings with and without defects, and the other was a collection of traffic signs with and without distortion. Even though their proposed method outperformed Fine Tuned VGG16 [11] and the Siamese Network, it did worse than Resnet [11]. They tested their method on unseen data with 25 new traffic sign classes. Recall that the objective was not to predict classes but defect type. Even though their method outperformed other traditional SNNs, their work had a limitation, as there were significant variations in accuracy for different error types. In our project, we hope to get higher accuracy, and we are concentrated only on predicting unseen classes, not their defects or distortions. For us, it is also essential for time efficiency.

C. Triplet Loss

Triplet Loss is widely used for Siamese Networks and is applied in different domains such as image retrieval [2], [3], face recognition [4], [5], and other classification tasks [6], [7]. Before discussing the following paper, it is necessary to provide background information for Triplet Loss. Google implemented Triplet Loss for facial recognition in 2015 [8]. In this case, three inputs are given to the model: anchor, positive, and negative. An input for reference is the anchor. The same category as anchor input includes positive input. Negative input is not part of the anchor class; instead, it is a random class. We use the Triplet Loss function to increase the distance between the anchor and the negative sample while minimizing the distance between the anchor and the positive sample.

Let's denote anchor, positive and negative classes as a,p, and n respectively. So, need to have this $d(a,p) - d(a,n) < 0$ where function $d(x,y)$ is Euclidean Distance metric. The distance between the anchor and the negative should be more than the positive and the anchor. Modifying this we can have the following loss:

$$LossL = max(d(a,n) - d(a,p), 0) \tag{2}$$

A parameter' margin' is introduced to increase the separation between positive and negative further.

$$L = max(d(a,n) - d(a,p) + margin, 0) \tag{3}$$

Hard triplets are helpful because they accelerate training time [4]. It is essential to have paired a, n, and p similar to each other while training thus, the model will be more robust. If it is trained on the only objects where instances n and p are straightforward to differentiate, it will not perform well where instances are close to each other. To reduce search space, some researchers proposed Online Hard Negative Mining [12]–[15], thus getting good samples for training in batches. However, a large amount of training data is required because of the many pairs of classes [16]. "As the number of identities becomes extremely large, the training will suffer from bad local minima because effective hard triplets are difficult to be found" [**?**].

The study suggests training triplet networks with subspace learning to address the issue. The proposed method divides the space of all instances into subspaces that only include similar and tricky instances. Along with subspace learning, they also worked on robust noise removal and effective image retrieval, combined to provide state-of-the-art performance in the MS-Celeb-1M competition (without external data in Challenge1) [16].

Some researchers used softmax in their research [14], [17]–[19] even with a combination of triplet and softmax. However, they clearly stated the reasons for choosing Triplet Loss over softmax. The fully-connected layer that links to softmax loss

will likewise grow significantly as the number of classes increases, making the GPU memory cost high with a typical batch size while making training time too long with a small batch size.

D. Contrastive Loss

Contrastive Loss is another Objective function used for training Siamese Networks. It was introduced in the early 1990s in the context of Siamese Networks. One of the earliest works that employed the Contrastive Loss function in Siamese Networks is the paper by [24]. This is the original paper that introduced Siamese Networks. Although it does not explicitly use the term "Contrastive Loss," it proposes a similar loss function for training a time delay neural network (TDNN) on pairs of signatures. The trained Siamese Network is used to verify whether two signatures belong to the same person.

$$L = \frac{1}{2N}\sum_{i=1}^{N} y_i d(a_i, p_i)^2 + (1-y_i)\max(0, \text{margin} - d(a_i, p_i))^2 \tag{4}$$

The Contrastive Loss function is designed to minimize the distance between similar data points (e.g., images of the same person) while maximizing the distance between dissimilar images. Here, in contrast to Triplet Loss we only. The Contrastive Loss function L is defined for a batch of N pairs, where a_i is the anchor data point, p_i is the positive or negative data point, y_i is the binary label indicating whether the pair (a_i, p_i) is similar ($y_i = 0$) or dissimilar ($y_i = 1$), $d(a_i, p_i)$ is the distance metric between the anchor and the positive/negative data point, and "margin" is the predefined threshold for separating similar and dissimilar pairs. The loss is averaged over the batch by dividing the summation by $2N$. In this work, [22] the authors propose a method for learning a similarity metric between pairs of data points using a Contrastive Loss function within a Siamese Network. The Contrastive Loss helps the model learn a low-dimensional representation that maps similar data points close together and dissimilar data points far apart. Another paper [23] presents a one-shot learning approach using Siamese Networks and Contrastive Loss for image recognition. The authors train the network on pairs of images to learn a similarity metric. The learned metric enables the network to perform one-shot classification, where the model recognizes objects from new classes based on just one or very few examples.

E. Prototype Embedding

Many traditional few-shot learning methods involve comparing test images with individual training embeddings which is a commonly used approach. [25] the authors introduce Matching Networks which employs an attention mechanism to compare a test image with individual training embeddings. The approach computes a weighted sum of the labels of training images based on the similarity between their embeddings and test image embedding. [26] propose Relation Networks, a few-shot learning approach that learns to compare a test image with each of the support images in a few-shot task. The method computes a relation score between the test image and each supported image, which is then used for classification. However, there are also other methods that do not require comparing the test image with all individual training images:meta-learning and prototype-based methods. [**?**], [28] This paper introduces prototypical networks, which are designed for few-shot learning. The authors use the distance between an image's embedding and the prototype embedding of each class to classify the image. They also experiment with different metrics such as Euclidean Distance and Cosine Similarity and showed the latter was not a good choice for distance. However, in contrast to our work, which employs a Siamese Network, they utilized meta-learning. In their approach, they create prototypes before training, while in our method, we generate prototypes after training. Our goal is to construct an encoder that embeds images such that images belonging to the same class are close to each other and distinct from images of other classes.

III. Datasets

We will talk about the datasets we used in this section. For the experiment, we used a portion of the lfw dataset [27], and to fine-tune the model, we used a dataset we had created for automotive remanufacturing parts.

A. LFW dataset

[27] introduced the LFW dataset. There were initially more than 13000 images in the dataset. With just 10 classes and 31 images per class in training and 10 images per class in testing, we created a custom dataset using this dataset. We reduced the dataset's size in order to test how well the model works with smaller datasets. During the testing process, we also introduced a few new random novel classes. Since few images had more than two faces, which could cause incorrect prediction, we also made the background neutral for each image to make our dataset comparable to the automotive dataset.

B. Automotive dataset

We adopted the same split as before but we had a few more instances per class during testing. We randomly selected 40 images instance per class for training and 10 images per class for validation and the rest being testing images. Similar to the LFW dataset, we included a few instances of new classes during testing to check how well it performs to identify unknown instances.

IV. Approach

In this project, we aim to use Convolution Neural Networks to train models capable of dealing with unknown classes in real time. As discussed in our dataset section, we work on automotive parts and the LFW dataset. Some parts may only be introduced during testing time. Algorithms that cannot learn those labels will misclassify them as one of the existing classes. In this section, we describe our approach. Figure 1 explains our approach in 3 steps.

1) Training the network
2) Prototype embeddings
3) Classification

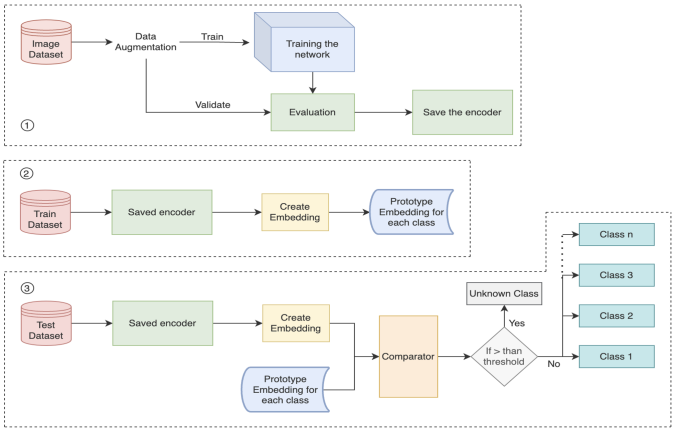


Fig. 1. Approach Diagram

A. Training the network

State-of-the-art models for Image classification tasks are trained on millions of images. The ImageNet dataset contains 1000 categories and about 1.2 million images. Because of our project's narrow domain and relative novelty, we won't deal with such big datasets. Transfer Learning is how a model created for one task is applied to another. It's only necessary to train only the last couple of layers of the model while keeping early layers frozen, as they can already detect low-level features such as edges, cycles, and so on. This approach helps to modify already trained powerful models to adapt to specific tasks.

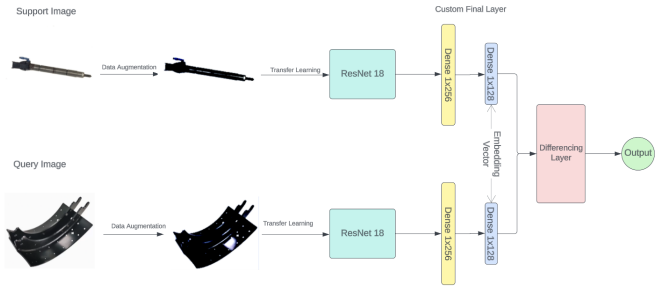


Fig. 2. Architecture Diagram

Training the network is divided in the following subsections:

- Data Augmentation
- Encoder Training

1) Data Augmentation: We applied data augmentation techniques needed to improve the performance of deep networks to create a powerful image classifier with less training data. In this step, we first resized our image to 224*224 and then normalized the images.

2) Encoder Training: We use Siamese Network with Transfer Learning. We send two images parallelly which are then processed by a trained Resnet-18 model. We unfreeze the last two layers and update the weights of only those two layers. This led to a decrease in training time and worked great with a small dataset. Figure 2 explains the architecture of the model. Once we evaluate our model using the validation set we save our encoder.

B. Prototype Embedding

In order to significantly reduce our prediction time, we implemented prototype embedding. One way to think of prototype embedding is as the centroid of clusters of instances from various classes. Assuming that the training set contains m instances of each of n classes, Figure 3. To accurately predict which class the given test image belongs to, we must compare each test image n*m times. Prototype embeddings allow us to cut down the comparison to just n times. This significantly boosts the network prediction time.

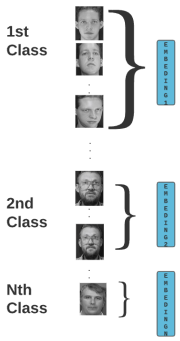


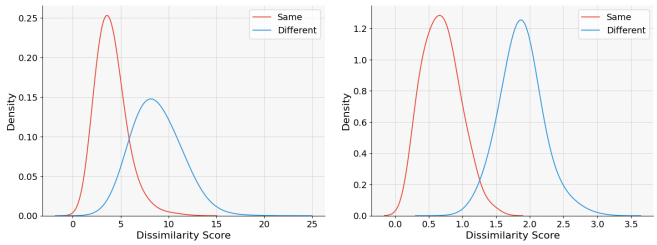
Fig. 3. Example of how we created prototype embeddings

Using the saved encoder from the previous step, we then create class embeddings for the training dataset. These embeddings are 128-dimensional vectors. Generally, in Siamese Network, we compare each test image with all training images to accurately predict, but this is a time-consuming process as explained earlier. To reduce we take the mean of embeddings in each class and store it as mean prototype embedding. Now test images will only be compared to newly created mean prototype embeddings.

C. Classification

1) Optimal Threshold: Now when we have trained a Siamese Network for few-shot learning to learn a distance metric between images, we can compare all image pairs and see their dissimilarity score distributions. We compare the distribution of dissimilarity scores for pairs of images belonging to the same and different classes. The model's confidence can be indicated by minimizing the intersection of image pairs for both the same and different classes. A smaller intersection number suggests better model performance. For example, in Figure 4a, the intersection is relatively large, indicating lower confidence, whereas in Figure 4b, the intersection is smaller, signifying improved performance. In short, image pairs falling

into an intersection might be misclassified depending on the threshold. In our research to determine the optimal threshold score, we devised the following solution. We plotted the number of images belonging to the same class but misclassified as different classes, and vice versa, for various thresholds ranging from -0.5 to 1.5. At this stage, it is equally important for us to address both metrics. Therefore, we chose the point where the two lines intersect, where they are closest to their global minimum. We chose 0.59 as the optimal threshold See Figure 5 where intersection happens where the threshold is equal to 0.59.



(a) (b)

Fig. 4. Distribution of image pair distance belonging to same and different classes

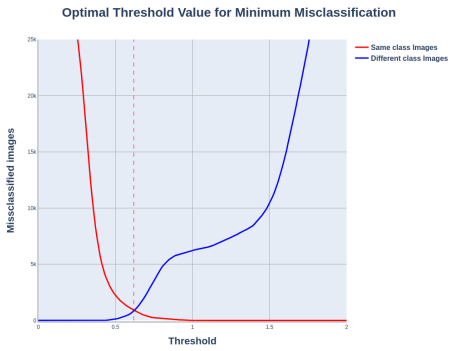


Fig. 5. Optimal Threshold for Minimum Misclassification

2) Prediction: The prototype embeddings are 128-dimensional vectors that represent the average feature vectors of the training samples for each class. These prototypes serve as a reference point to classify new test images. When a new test image is introduced, the Siamese Network's encoder converts the image into a 128-dimensional vector. This embedding is then compared with all the prototype embeddings to measure similarity, which can be done using distance metrics like Euclidean or cosine distance. If the smallest distance between the test image embedding and the prototypes exceeds a designated threshold, the system classifies the test image as unknown, indicating that it does not belong to any of the known classes. Otherwise, the test image is assigned to the class whose prototype has the minimum distance to the test image embedding. This method is an example of few-shot learning because it can generalize and classify new examples based on a limited number of training samples per class, leveraging the Siamese Network's ability to learn meaningful feature representations and the prototype-based classification approach.

D. Deep Learning Classifier

After having datasets for training and testing, we will try different algorithms and compare their results and performance. The methodologies that are applied to the issue of unknown classes are Siamese Network and Open Set Classification. Siamese Network, unlike traditional CNNs, takes two inputs into the network, and the dissimilarity score is the output. So we're going to use Siamese Network with Contrastive Loss and Triplet Loss to compare which performs better for our task. Another methodology is to compare distance metrics, mainly Cosine Similarity and Euclidean Distance since the Siamese Network works on the basis of distances. So those are things we will research and implement in our domain set.

E. Validation

To evaluate the performance of our models, we will focus on three aspects. First and foremost, it's essential to have a robust model predicting labels with high accuracy. We aim to have more than 90% accuracy on the automotive parts dataset. Secondly, To address our main problem, unknown classes, we will evaluate the models' robustness on unseen classes. The satisfying model will differentiate unknown classes from known classes. Last, we also seek an efficient and fast model for real-time prediction. We aim to have a prediction time from the image first seen to classification to be less than 10 seconds.

V. Results

A. Prototype Embedding

We compared the performance of our model before and after applying prototype embedding to predict test cases. Figure 6 displays the results of the initial model without prototype embedding and the prototype model with prototype embedding. Before the prototype embedding, our model compared each test image with all other images, resulting in a time-consuming process. However, with prototype embedding, although there was a slight increase in error, the benefits of prototype embedding were significant as we achieved a 25-fold acceleration in the prediction time. Therefore, the prototype model with its enhanced efficiency, represents a substantial improvement in our model's performance.

B. Contrastive Loss vs Triplet Loss

One of our goals was to compare how Siamese Network worked on Contrastive Loss and Triplet Loss and which loss performs better in our task. From Figure 7 it is visible that triplet loss does not converge properly during training. This is also evident from the results in table I, which clearly shows in all the categories Triplet Loss performs comparatively worst than Contrastive Loss.

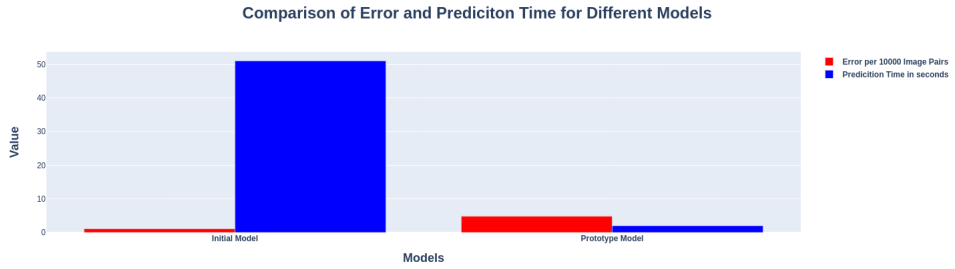
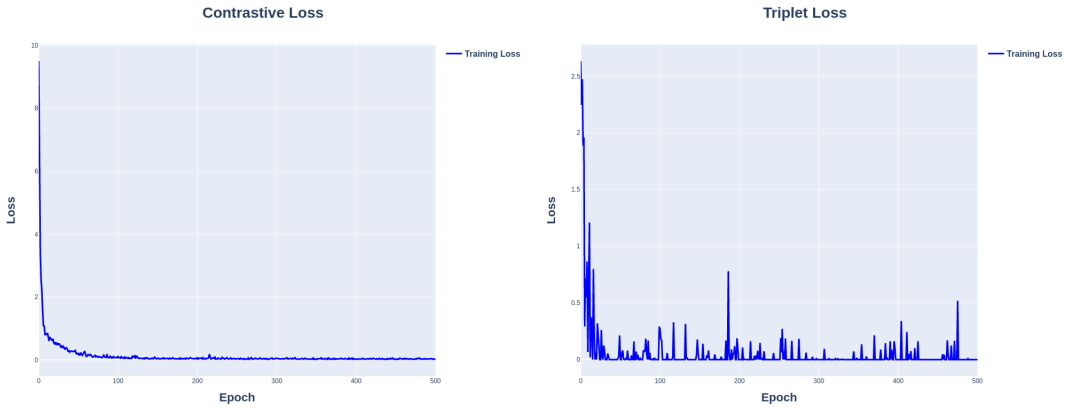


Fig. 6. Before and after prototype embedding applied



(a) Contrastive Loss (b) Triplet Loss

Fig. 7. Comparing Contrastive Loss vs Triplet Loss losses during training

Objective Function	Accuracy	F1-Score	Accuracy of Unknown Class
Contrastive Loss	**92%**	**92%**	**82%**
Triplet Loss	85%	85%	65%

TABLE I
COMPARING ACCURACY, F1-SCORE AND ACCURACY OF UNKNOWN CLASS FOR CONTRASTIVE LOSS AND TRIPLET LOSS

C. Comparison of Distance Metrics

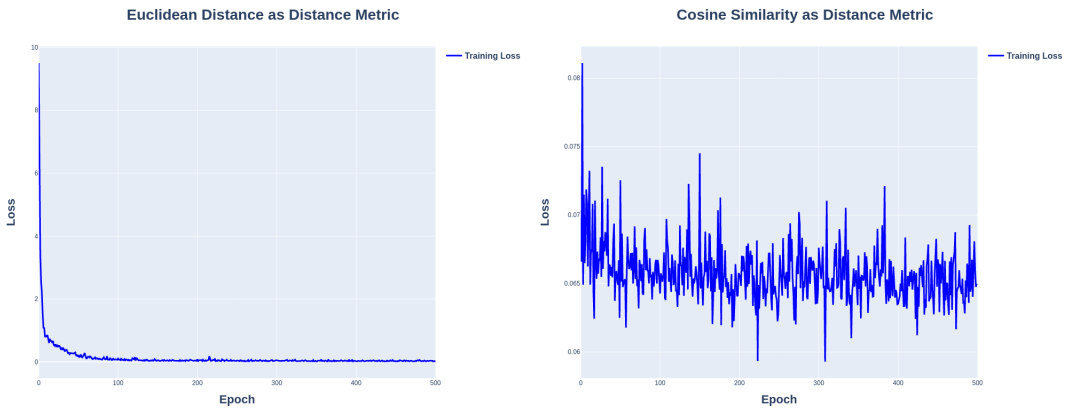
Another goal was to compare how the Siamese Network performed using different distance metrics. As discussed earlier, the Siamese Network works on the principle of finding how two images are dissimilar by calculating the distance between them. So, we wanted to compare two commonly used distance metrics, Euclidean Distance, and Cosine Similarity. In our experiment, we found out that Cosine Similarity did not converge at all and was actually looking more like a noise plot, as shown in Figure 8. So Euclidean Distance performs better distance metric than Cosine Similarity.

D. Fine-tuned Model

Leveraging the pre-trained ResNet-18 model, we fine-tuned it on an automotive dataset. We incorporated the optimization techniques discussed earlier, such as optimal threshold and prototype embedding, to enhance our model. The model's performance was also tested on the automotive dataset for validation. Furthermore, a few hyperparameters were optimized using the validation set. Figure 9 demonstrates that our model does not suffer from overfitting. As shown in Table II, after applying the aforementioned techniques, our approach achieved an accuracy of 96% with a prediction time of just 0.04 seconds.

VI. CONCLUSION

In this paper, we presented our method for the real-time detection of unknown classes and instances in small datasets. There are no thorough studies that address the issues we have with real-time automotive part prediction while experiencing OSR issues. Our research also indicated that Euclidean Distance is a superior distance metric to compare images in a Siamese Network when compared to Cosine Similarity, and Contrastive Loss performs better than Triplet Loss in Siamese Networks. Additionally, our improved model demonstrates that our method is the workable and faster method for categorizing remanufactured automotive parts so that they can be sorted out



(a) Euclidean Distance Loss

(b) Cosinie Similarity Loss

Fig. 8. Comparing distance metrics loss during training

Dataset	Threshold Score	Accuracy	F1-score	Accuracy of Unknown Class	Prediction Time in seconds
LFW dataset	1	92%	92%	82%	0.007
automobile dataset-before optimization	1	83%	78%	10%	0.056
automobile dataset-after optimization	0.59	**96%**	97%	88%	**0.038**

TABLE II
COMPARING TUNING RESULTS ON AUTOMOBILE DATASET

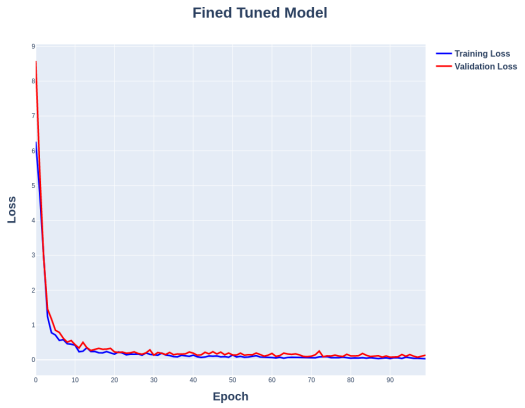


Fig. 9. Train vs Validation loss on automobile dataset

quickly based on known classes, and if a new class instance is introduced, it can categorize as an unknown class.

VII. FUTURE WORK

One of our current limitations is that even in unknown classes, we can have multiple classes, and all are currently categorized under one class, "unknown class." To overcome these, we plan to implement a clustering-based algorithm to cluster classes in unknown classes. Another future plan is to deploy this model in the industry to see how it performs in real-life situations and improve its limitation. We also plan to find the minimum instances needed per class without model accuracy and prediction time. This further helps us to implement our model with fewer instances during training and decreases training time.

REFERENCES

[1] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. 1993. Signature verification using a "Siamese" time delay neural network. In Proceedings of the 6th International Conference on Neural Information Processing Systems (NIPS'93). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 737–744.

[2] Song, Hyun Oh, et al. "Deep Metric Learning via Lifted Structured Feature Embedding." 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016. Crossref, https://doi.org/10.1109/cvpr.2016.434.

[3] Fang Zhao, et al. "Deep Semantic Ranking Based Hashing for Multi-Label Image Retrieval." 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2015. Crossref, https://doi.org/10.1109/cvpr.2015.7298763.

[4] Schroff, Florian, et al. "FaceNet: A Unified Embedding for Face Recognition and Clustering." 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2015. Crossref, https://doi.org/10.1109/cvpr.2015.7298682.

[5] Simonyan, Karen and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition." CoRR abs/1409.1556 (2014): n. pag.

[6] Chen, Weihua & Chen, Xiaotang & Zhang, Jianguo & Huang, Kaiqi. (2017). A Multi-task Deep Network for Person Re-identification.

[7] Hermans, Alexander et al. "In Defense of the Triplet Loss for Person Re-Identification." ArXiv abs/1703.07737 (2017): n. pag.

[8] He, Kaiming et al. "Deep Residual Learning for Image Recognition." 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015): 770-778.

[9] Pornntiwa Pawara, Emmanuel Okafor, Marc Groefsema, Sheng He, Lambert R.B. Schomaker, Marco A. Wiering, One-vs-One classification for deep neural networks, Pattern Recognition, Volume 108, 2020, 107528, ISSN 0031-3203, https://doi.org/10.1016/j.patcog.2020.107528.

[10] Nagy A. and Czúni L. (2021). Detecting Object Defects with Fusioning Convolutional Siamese Neural Networks.In Proceedings of the 16th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 4: VISAPP, ISBN 978-989-758-488-6, pages 157-163. DOI: 10.5220/0010263301570163

[11] Russakovsky, O., Deng, J., Su, H. et al. ImageNet Large Scale Visual Recognition Challenge. Int J Comput Vis 115, 211–252 (2015). https://doi.org/10.1007/s11263-015-0816-y

[12] Omkar M. Parkhi, Andrea Vedaldi and Andrew Zisserman. Deep Face Recognition. In Xianghua Xie, Mark W. Jones, and Gary K. L. Tam, editors, Proceedings of the British Machine Vision Conference (BMVC), pages 41.1-41.12. BMVA Press, September 2015.
[13] Schroff, Florian, et al. "FaceNet: A Unified Embedding for Face Recognition and Clustering." 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2015. Crossref, https://doi.org/10.1109/cvpr.2015.7298682.
[14] Wang, Jiang et al. "Learning Fine-Grained Image Similarity with Deep Ranking." 2014 IEEE Conference on Computer Vision and Pattern Recognition (2014): 1386-1393.
[15] Hermans, Alexander et al. "In Defense of the Triplet Loss for Person Re-Identification." ArXiv abs/1703.07737 (2017): n. pag.
[16] Wang, Chong, et al. "How to Train Triplet Networks with 100K Identities?" 2017 IEEE International Conference on Computer Vision Workshops (ICCVW), Oct. 2017. Crossref, https://doi.org/10.1109/iccvw.2017.225.
[17] He, Kaiming, et al. "Deep Residual Learning for Image Recognition." 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016. Crossref, https://doi.org/10.1109/cvpr.2016.90.
[18] Hoffer, Elad, and Nir Ailon. "Deep Metric Learning Using Triplet Network." Lecture Notes in Computer Science, 2015, pp. 84–92. Crossref, https://doi.org/10.1007/978-3-319-24261-3.
[19] Sankaranarayanan, Swami, et al. "Triplet Probabilistic Embedding for Face Verification and Clustering." 2016 IEEE 8th International Conference on Biometrics Theory, Applications and Systems (BTAS), Sept. 2016. Crossref, https://doi.org/10.1109/btas.2016.7791205.
[20] Shubham Shinde, Ashwin Kothari, Vikram Gupta, YOLO based Human Action Recognition and Localization, Procedia Computer Science, Volume 133, 2018, Pages 831-838, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2018.07.112, (https://www.sciencedirect.com/science/article/pii/S1877050918310652)
[21] Koch, G., Zemel, R., Salakhutdinov, R. (2015). Siamese Neural Networks for One-shot Image Recognition. In ICML Deep Learning Workshop. Retrieved from https://www.cs.cmu.edu/ rsalakhu/papers/oneshot1.pdf
[22] R. Hadsell, S. Chopra and Y. LeCun, "Dimensionality Reduction by Learning an Invariant Mapping," 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06), New York, NY, USA, 2006, pp. 1735-1742, doi: 10.1109/CVPR.2006.100.
[23] Koch, Gregory R.. "Siamese Neural Networks for One-Shot Image Recognition." (2015).
[24] Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., Shah, R. (1994). Signature Verification using a "Siamese" Time Delay Neural Network. NIPS 1993.
[25] Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., others. (2016). Matching Networks for One Shot Learning. In Advances in Neural Information Processing Systems (NIPS) 2016.
[26] Sung, F., Yang, Y., Zhang, L., Xiang, T., Torr, P. H. S., & Hospedales, T. M. (2018). Learning to Compare: Relation Network for Few-Shot Learning. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2018.
[27] Huang, G., Ramesh, M., Berg, T. & Learned-Miller, E. Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments. (University of Massachusetts, Amherst,2007,10)
[28] Snell, J., Swersky, K. & Zemel, R. Prototypical Networks for Few-shot Learning. (2017)
[29] Chen, W., Liu, Y., Kira, Z., Wang, Y. & Huang, J. A Closer Look at Few-shot Classification. (2020)