Facial Recognision using Siamese Neural Network

Aryan Agnihotri
School of Computer Science
Lovely Professional University
Phagwara, India
aryangnihotriofficial@gmail.com

***Abstract*— Training exceptional features for machine learning applications can be figuratively or quite literally be expensive and computationally not easy in cases where available data is limited amount. The one-shot learning situation is an apt example, where I have to predict correctly from a single example from each class. I'm working on a Siamese neural network learning strategy that uses a unique structure to visually prioritize similarities between respective inputs. Once the network gets calibrated, I can use the powerful diversification features to generalize the network's prediction ability not only to newer data but also to entirely new classifiable classes based on the classifications. We can produce robust results that outperform other deep learning models with near euphoric performance on tasks of classification for face recognition using a convolutional architecture.**

I. INTRODUCTION

Humans, as a species, have a remarkable ability to absorb and identify new patterns. Specifically, individuals can quickly understand new insights when stimulated and then recognize the underlying variations of these concepts in their future cognition. Although ML has proven to be very effective in a variety of applications, including spam detection, web-search, annotation generation, image recognition, and speech recognition, these algorithms often fail whilst trying to predict on data with limited monitored information. Our goal is to generalize to non-familiar category(ies) without the need to require extensive retraining, which would be impractical due to limitation on the amount of data or in the context of predictions made online, e.g. search on the web.

An interesting task of classification under the constraint that I can only notice a single instance of each potential class before making a prediction on a test case. This concept is called one-shot learning and is central to our model presented in this study. It is important to distinguish this approach from hands-off learning, where a model can't test examples of the targeted class.

One-shot learning can be effectively addressed by developing domain-specific inference or features processes that have highly discriminative properties for the target task. Therefore, systems integrating these methods exhibit superior performance on similar cases but lack the robustness to provide solutions to other types of problems. This paper presents a new approach that minimizes assumptions about input structure while automatically deriving features that enable successful generalization from limited examples. Our approach leverages a deep learning framework, which uses multiple nonlinear layers to capture the invariants of the input spatial transformations. This is achieved by running a model with considerable numbers of parameters and using a

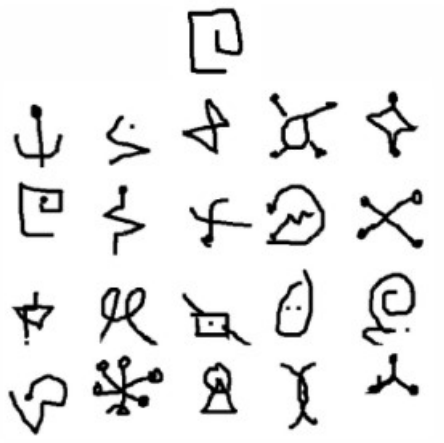


Fig. 1. An examle for a 20-way one-shot classification task utilizing the Omniglot dataset is presented. The solitary test image is displayed above a grid of 20 images that depict the potential unseen classes that can be selected for the test image. These 20 images serve as the sole known instances of each of those classes.

significant amount of data to avoid overfitting. The resulting features are articularly powerful because they are learned without imposing strong priors, even though the learning algorithm itself may incur significant overhead.

II. APPROACH

Ordinarily, picture representations are gotten through a directed metric-based approach employing a Siamese neural organize. The arrange capacities are at that point reused for one-shot learning without retraining.

In the tests we did, I centered as it were on facial acknowledgment, in spite of the fact that the fundamental approach can be connected to nearly any methodology. This can be appeared in Figure 2. In this specific space, I utilize Siamese large convolutional neural systems. These systems are competent of learning common picture highlights that are valuable to form forecasts around obscure lesson disseminations, indeed when as it were a restricted number of illustrations of these unused dispersions are accessible. Also, these systems can be effortlessly made by using standard optimization procedures on sets which are tested from the source. Also, they offer a competitive approach that is not depending on domain-specific information but misuses the control of profound learning strategies.

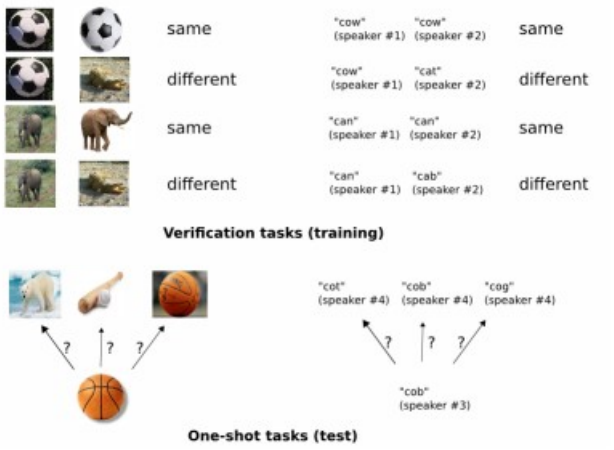


Fig2. Our general strategy involves training a model to discern between a collection of pairs with the same or different attributes. This initial step allows us to establish a foundation for further evaluation. Subsequently, I aim to extend the model's capabilities by applying learned feature mappings to assess novel categories for verification purposes. By following this approach, I can enhance the model's ability to generalize and adapt to new scenarios.

To create a point picture classification show, our objective is to to begin with prepare a neural organize that can recognize the lesson personality of picture sets, which may be a standard errand for personality confirmation. picture. I propose that systems that perform well in confirmation moreover illustrate generalizability in point classification. The confirmation show is prepared to recognize input sets based on the likelihood of having a place to the same lesson or distinctive classes. This show can at that point be utilized to assess modern pictures, with precisely one picture for each unused lesson, by comparing each picture with a test picture. The coordinate with the foremost vital score, as chosen by the affirmation organize, will at that point be doled out the foremost lifted probability to the spot errand. Within the occasion that the highlights learned by the affirmation appear are palatable for asserting or denying the character of characters from one set of letter sets, they need to as well be satisfactory for other letter sets, given that the appear has been revealed to a arranged run of letter sets to progress changeability among the learned highlights.

III. Related Works

The machine learning community has given constrained consideration to inquire about on one-shot learning calculations, making it a decently youthful range of ponder. In any case, there are some critical lines of work that go before the current paper.

The seminal work on one-shot learning within the early 2000's by Li Fei-Fei. presented a variational Bayesian system for one-shot picture classification. This system leverages already learned classes to figure future ones, particularly when there are restricted cases accessible from a particular course.

In a more later approach, Lake. tended to the issue of one-shot learning for character acknowledgment from the POV of cognitive science. They presented a strategy called Hierarchial Bayesian Program Learning (HBPL). Through an arrangement of papers, the makers created a generative show for drawing characters, breaking down the picture into littler pieces. The most objective of HBPL is to decide a auxiliary clarification for the watched pixels. Be that as it may, deduction beneath HBPL postures challenges due to the tremendous joint parameter space, coming about in an recalcitrant integration issue.

Other modalities or exchange learning approaches have been investigated by analysts. In later work, Lake et al. utilized a generative Hierarchy Covered up Markov show and Bayesian induction to recognize unused words talked by obscure speakers. Maas and Kemp's distributed work centers on utilizing Bayesian systems to anticipate traits for Ellis Island traveler information. Wu and Dennis tended to one-shot learning within the setting of way arranging calculations for automated activation in 2012. Lim's inquire about centers on adjusting a degree of how much each category ought to be weighted by each preparing model within the loss function to "borrow" cases from other classes within the preparing set.

This concept offers utility for sets with limited instances of certain classes, offering a versatile and uninterrupted method of integrating inter-class information into the model.

IV. Deep Siamese Networks for Image Verification

Siamese nets were at first presented within the early 1990's by LeCun and Bromley to address the issue of signature confirmation as an picture coordinating errand. A siamese neural organize comprises of two indistinguishable systems that get partitioned inputs but are associated through an vitality work at the most noteworthy level. This work calculates a metric between the highest level of highlight representations on either side (allude to Figure 3). The parameters of the twin systems are tied together, guaranteeing that two profoundly comparable pictures cannot be mapped to significantly distinctive areas within the include space by their particular systems. Usually since each organize computes the same work. Furthermore, the organize is symmetric, meaning that when I display two unmistakable pictures to the twin systems, the beat conjoining layer will compute the identical metric as in case I were to show the same two pictures but to the inverse twins.

Within the think about conducted by LeCun et al., a contrastive vitality work was utilized, joining double terms to diminish the vitality of comparative sets and open up the vitality of divergent sets. In any case, in our inquire about, I utilize the weighted L1 separate between the twin highlight vectors h1 and h2, combined with a sigmoid enactment that maps onto the interim [0, 1]. Thus, a cross-entropy objective rises as a common choice for preparing the arrange. It is worth noticing that LeCun et al. straightforwardly learned the similitude metric, which was certainly characterized by the vitality misfortune. In differentiate, I build up the metric as indicated over, adjusting with the approach laid out in Facebook's DeepFace paper.

Our top-performing models join different convolutional layers earlier to the fully-connected layers and high-level vitality work. Convolutional neural systems have illustrated exceptional victory in various large-scale computer vision

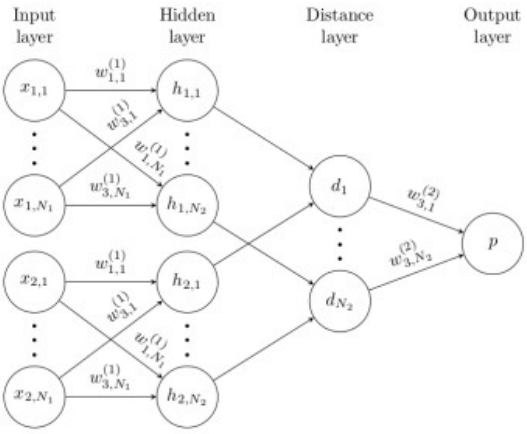


Figure 3. A simple 2 hidden layer siamese network for binary classification with logistic prediction p. The structure of the network is replicated across the top and bottom sections to form twin networks, with shared weight matrices at each layer.

applications, especially in assignments related to picture acknowledgment.

A few components contribute to the offer of convolutional systems. One key advantage is their neighborhood network, which successfully decreases the number of parameters within the demonstrate. This decrease intrinsically gives a frame of built-in regularization, improving the network's capacity to generalize. It is worth noticing, in any case, that convolutional layers are computationally more costly compared to standard nonlinearities.

A few variables contribute to the request of convolutional systems. One key advantage is their neighborhood network, which viably decreases the number of parameters within the demonstrate. This diminishment intrinsically gives a frame of built-in regularization, improving the network's capacity to generalize. It is worth noticing, in any case, that convolutional layers prove to be more costly, computationally, compared to standard nonlinearities.

Furthermore, the availability of very fast CUDA libraries has revolutionized the construction of large convolutional networks. These libraries enable the creation of extensive networks without incurring an unacceptable quota of training time. This advancement has significantly contributed to the scalability and practicality of convolutional networks.

In summary, the combination of reduced parameter count, built-in regularization, interpretability of the convolution operation, and the availability of efficient CUDA libraries make convolutional networks particularly appealing for various applications.

A. The Model

Our default configuration comprises of a siamese CNN that consists of 'L' layers, each containing 'N_l' units. The hidden vector in layer l for the first twin is represented by h1,l, while h2,l denotes the same for the second twin. The first L2 layers exclusively use the rectified linear (ReLU) units, while the remaining layers use sigmoid functioned units.

The model consists a series of convolutional layers, wherein every layer utilizes a singular channel incorporating filters of different sizes and a constant stride of 1. To enhance performance, the number of convolutional filters is designated as a multiple of 16. Following this, the network employs a ReLU activation function on the resultant feature maps, with the possibility of subsequent max-pooling using a filter size and stride of 2. Consequently, the kth filter map in each layer assumes the subsequent structure:

$$a_{1,m}^{(k)} = \text{max-pool}(\max(0, \mathbf{W}_{l-1,l}^{(k)} \star \mathbf{h}_{1,(l-1)} + \mathbf{b}_l), 2)$$
$$a_{2,m}^{(k)} = \text{max-pool}(\max(0, \mathbf{W}_{l-1,l}^{(k)} \star \mathbf{h}_{2,(l-1)} + \mathbf{b}_l), 2)$$

where 'Wl−1,l' is the 3-D tensor that represents the feature map(s) for layerl and star(*) is the substantial convolutional operation comparing to returning as it were those yield units which were the result of total cover between each convolutional channel and the input highlight maps.

The final convolutional layer's units are smoothed into a particular vector, taken after by a fully-connected layer and another layer that calculates the initiated remove metric between each siamese twin. This metric is at that point given to a single sigmoidal yield unit, which scores the closeness between the two include vectors. The αj parameters are learned amid preparing and weigh the significance of the component-wise separate. This comes about in a last fully-connected layer for the arrange that interfaces the two siamese twins and characterizes the learned include space's metric.

One example is illustrated in Fig 4, showcasing the largest version of our model that was taken into consideration. This network also yielded the most optimal outcome among all networks for the verification task.

B. The Learning

Loss Function. The loss function is defined as follows: Let M represent the minibatch size, where i indexes the ith minibatch. Now, consider y(x(i)1, x(i)2) as a length-M vector that contains the labels for the minibatch. In this case, I assume that y(x(i)1, x(i)2) = 1 whenever x1 and x2 are from the same character class, and y(x(i)1, x(i)2) = 0 otherwise. To optimize our binary classifier, I impose a regularized cross-entropy objective of the following form:

$$\mathcal{L}(x_1^{(i)}, x_2^{(i)}) = \mathbf{y}(x_1^{(i)}, x_2^{(i)}) \log \mathbf{p}(x_1^{(i)}, x_2^{(i)}) +$$
$$(1 - \mathbf{y}(x_1^{(i)}, x_2^{(i)})) \log (1 - \mathbf{p}(x_1^{(i)}, x_2^{(i)})) + \boldsymbol{\lambda}^T |\mathbf{w}|^2$$

Optimization. The objective of optimization is achieved by integrating it with the standard backpropagation algorithm. In this approach, the gradient is accumulated across the twin networks as a result of the tied weights. To ensure consistency, I set the minibatch size to 128 and define the learning rate ηj, momentum μj, and L2 regularization weights λj on a layer-wise basis. The algorithm for optimization that I used is called ADAM, that can be found here: "https://keras.io/api/optimizers/adam/ "

Consequently, our update rule at epoch T can be summarized as follows:

$$\mathbf{w}_{kj}^{(T)}(x_1^{(i)}, x_2^{(i)}) = \mathbf{w}_{kj}^{(T)} + \Delta\mathbf{w}_{kj}^{(T)}(x_1^{(i)}, x_2^{(i)}) + 2\lambda_j|\mathbf{w}_{kj}|$$

$$\Delta\mathbf{w}_{kj}^{(T)}(x_1^{(i)}, x_2^{(i)}) = -\eta_j \nabla w_{kj}^{(T)} + \mu_j \Delta\mathbf{w}_{kj}^{(T-1)}$$

where ∇w_{kj} is the partial derivative with respect to the weight between the jth neuron in some layer and the kth neuron in the successive layer.

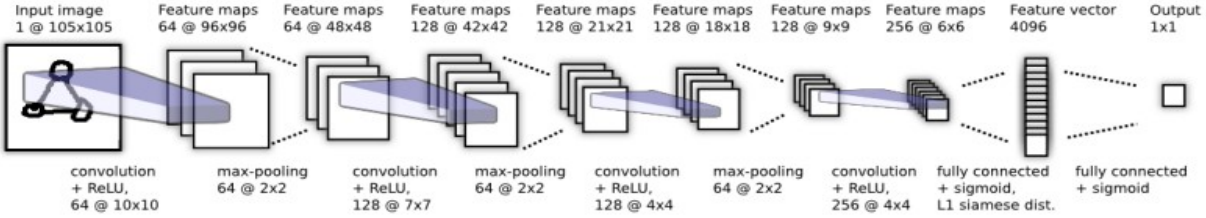


Figure 4. Best convolutional architecture selected for verification task.

Weight initialization. The arrange weights within the convolutional layers were initialized by drawing from a ordinary conveyance with a zero-mean and a standard deviation of 10^{-2}. Similarly, the biases in these layers were initialized from a normal distribution with a mean of 0.5 and a standard deviation of 10^{-2}. In the fully-connected layers, the biases were initialized following the same approach as the convolutional layers. However, the weights in these layers were drawn from a wider normal distribution with a zero-mean and a standard deviation of 2×10^{-1}.

Learning Schedule. Although I implemented a distinct learning rate for each layer, the learning rates were uniformly reduced by 1 percent per epoch across the entire network. This decay was represented by the equation $\eta(T)j = 0.99\eta(T-1)j$. Through this approach, I discovered that by gradually decreasing the learning rate, the network was able to more easily converge to local minima without becoming trapped in the error surface.

For consistency, I set the momentum to initiate at 0.5 in each layer. It then increased linearly with every epoch till reaching the value of μj, which represents the individual momentum term for j-th layer.

During the training process, I conducted a maximum of 200 epochs for each network. However, I closely monitored the one-shot validation error on a set of 320 randomly generated one-shot learning tasks from the alphabets and drawers in the validation set. If the validation error did not decrease for 20 consecutive epochs, I halted the training and utilized the parameters of the model at the best epoch based on the one-shot validation error.

In cases where the error of validation continue to decrease throughout the entire learning schedule, I preserved the final state of the model made by this procedure.

Hyperparameter Tuning. I utilized the beta version of Whetlab, a framework to optimize Bayesian, to conduct hyperparameter selection. To determine the learning schedule and hyperparameters regularization, I established the learning rate, layer-wise, ηj within the range of $[10^{-4}, 10^{-1}]$, the layer-wise momentum μj within the range of [0, 1], and the layer-wise L2 regularization penalty λj within the range of [0, 0.1]. As for the network hyperparameters, I allowed the size of convolutional filters to vary from 3x3 to 20x20, while the number of convolutional filters in each layer ranged from 16 to 256, incrementing by multiples of 16. The fully-connected layers spanned from 128 to 4096 units, also increasing by multiples of 16. Our chosen optimizer aimed to maximize the accuracy of the one-shot validation set. The score assigned to a single Whetlab iteration was determined by selecting the maximum value of this metric discovered during any of the epoch.

V. The Experiment

I trained the model on a sub-set of the Labelled Faces in the Wild data set, which I first describe. I also created a custom dataset where I put in about 900+ photos to train the model for facial verification. I then provide details with respect to verification and performance.

A. The Labelled Faces in the Wild Dataset

The "Labelled Faces in the Wild" (LFW) dataset is a widely recognized and widely used benchmark dataset in the field of facial recognition and computer vision. This dataset is valuable for evaluating the performance of facial recognition algorithms and models. LFW is characterized by its size, content, variability, and its focus on labeled pairs of faces for face verification.

In terms of size and content, LFW boasts a substantial collection of over 13,000 labeled face images. These images represent more than 5,700 individuals, predominantly public figures, celebrities, and well-known personalities. The dataset's richness and diversity in terms of labeled face images make it a valuable resource for research and development in the field of facial recognition.

Variability is a key characteristic of the LFW dataset. The images in LFW exhibit substantial variations, including changes in pose, illumination, facial expression, resolution, and age. This variability closely simulates real-world, unconstrained scenarios and poses a significant challenge to facial recognition algorithms.

LFW is primarily used for the task of face verification, where the central objective is to determine whether two face images depict the same person or different individuals. The

Fig 5. The Labelled Faces in the Wild Dataset Snippet

Fig 6. A snippet of the random shots taken by me to create a dataset

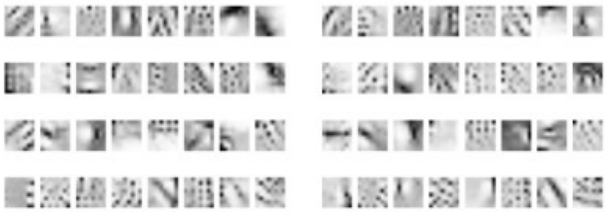


Figure 7. Examples of first-layer convolutional filters learned by the siamese network. Notice filters adapt different roles: some look for very small point-wise features whereas others function like larger-scale edge detectors.

```
loss, accuracy = model.evaluate(X_test, y_test, verbose=False)
print(f"Accuracy: {accuracy*100:.2f}%")
```

```
Accuracy: 87.50%
```

Fig 8. Accuracy on the Final run of the model

dataset provides labeled pairs of images, classifying each pair as either "same person" or "different individuals." As a result, researchers use LFW to benchmark the accuracy of facial recognition models. Achieving high accuracy on this dataset is considered a challenging task due to its variability and the requirement for accurate classification of face pairs.

B. Training and Testing

In order to train our verification network, we utilized three distinct data set sizes consisting of 14,000, 500, and 400 examples for training. These examples were obtained by randomly sampling both same and different pairs. The 14,000 examples were sourced from the Labelled Faces in the Wild Dataset for Training, while the 500 images were initially captured by myself for Training purposes. An additional 400 images were collected for Testing and Verification, with a mixture of my own images and those from the Labelled Faces in the Wild Dataset being used for this purpose.

The 14,000 training examples serve as the backbone for training the verification network. This substantial dataset encompasses a rich pool of images sourced from the "Labelled Faces in the Wild Dataset for Training." The dataset's strength lies in its diversity, comprising faces of various public figures and individuals. These images encapsulate the real-world scenario, featuring challenges such as pose variations, different lighting conditions, distinct facial expressions, and age differences.

Furthermore, the training dataset is meticulously designed to incorporate both "same" and "different" pairs. "Same" pairs comprise images depicting the same individual, while "different" pairs involve images of distinct individuals. This binary classification is instrumental in training the network to effectively distinguish between these two scenarios. The process of random sampling is employed to ensure that the training data remains unbiased, preventing the network from overfitting to specific patterns.

The 400 images designated for testing and verification purposes represent an independent evaluation set. This set comprises a fusion of self-captured images and those sourced from the "Labelled Faces in the Wild Dataset." This deliberate mix creates a realistic testing environment that simulates real-world conditions. By introducing self-captured images, the testing set mirrors the practical challenges of face verification systems, where the network must verify identities based on previously unseen faces.

The verification network's performance is rigorously assessed using this independent testing set. The network is tested against the task of correctly identifying "same" and "different" pairs, thus assessing its ability to generalize beyond the training data. The network's competence in recognizing faces it has not encountered during training is a crucial determinant of its real-world effectiveness.

C. One-Shot Learning

LFW is primarily used for the task of face verification, where the central objective is to determine whether two face images depict the same person or different individuals. The dataset provides labeled pairs of images, classifying each pair as either "same person" or "different individuals." As a result, researchers use LFW to benchmark the accuracy of facial recognition models. Achieving high accuracy on this dataset is considered a challenging task due to its variability and the requirement for accurate classification of face pairs.

After optimizing a siamese organize for confirmation, able to grandstand the discriminative potential of our learned highlights through one-shot learning. To classify a test

```python
def make_embedding():
    inp = Input(shape=(100,100,3), name='input_image')

    # First block
    c1 = Conv2D(64, (10,10), activation='relu')(inp)
    m1 = MaxPooling2D(64, (2,2), padding='same')(c1)

    # Second block
    c2 = Conv2D(128, (7,7), activation='relu')(m1)
    m2 = MaxPooling2D(64, (2,2), padding='same')(c2)

    # Third block
    c3 = Conv2D(128, (4,4), activation='relu')(m2)
    m3 = MaxPooling2D(64, (2,2), padding='same')(c3)

    # Final embedding block
    c4 = Conv2D(256, (4,4), activation='relu')(m3)
    f1 = Flatten()(c4)
    d1 = Dense(4096, activation='sigmoid')(f1)


    return Model(inputs=[inp], outputs=[d1], name='embedding')
```

Fig 9(a): Embedding layer that contains the 4 Convolution Layer, 3 Pooling Layer, 1 Flatten Layer and 1 OutputLayer

```python
# Siamese L1 Distance class
class L1Dist(Layer):

    # Init method - inheritance
    def __init__(self, **kwargs):
        super().__init__()

    # Magic happens here - similarity calculation
    def call(self, input_embedding, validation_embedding):
        return tf.math.abs(input_embedding - validation_embedding)
```

Fig 9(b): A distance function that compares 2 images

```python
def make_siamese_model():

    # Anchor image input in the network
    input_image = Input(name='input_img', shape=(100,100,3))

    # Validation image in the network
    validation_image = Input(name='validation_img', shape=(100,100,3))

    # Combine siamese distance components
    siamese_layer = L1Dist()
    siamese_layer._name = 'distance'
    distances = siamese_layer(embedding(input_image), embedding(validation_image))

    # Classification layer
    classifier = Dense(1, activation='sigmoid')(distances)

    return Model(inputs=[input_image, validation_image], outputs=classifier, name='SiameseNetwork')
```

Fig 9(c). A snippet from the code that implements the Siamese Neural Network

```
Model: "SiameseNetwork"
__________________________________________________________________________________________________
 Layer (type)                Output Shape              Param #   Connected to
==================================================================================================
 input_img (InputLayer)      [(None, 100, 100, 3)]     0         []

 validation_img (InputLayer  [(None, 100, 100, 3)]     0         []
 )

 embedding (Functional)      (None, 4096)              3896044   ['input_img[0][0]',
                                                       8          'validation_img[0][0]']

 l1_dist_1 (L1Dist)          (None, 4096)              0         ['embedding[0][0]',
                                                                  'embedding[1][0]']

 dense_2 (Dense)             (None, 1)                 4097      ['l1_dist_1[0][0]']

==================================================================================================
Total params: 38964545 (148.64 MB)
Trainable params: 38964545 (148.64 MB)
Non-trainable params: 0 (0.00 Byte)
__________________________________________________________________________________________________
```

Fig 10. A snippet from the running structure of the Siamese Model

picture x into one of C categories, we utilize a set of other pictures {xc} speaking to cases of each category as input for the arrange. We at that point anticipate the course with the greatest similitude utilizing the argmax work. In our case, we as it were utilize my confront as a substantial classifier and prepared against two columns as a base.

To assess our one-shot learning execution, we made a 20-way within-image classification assignment. We arbitrarily chose an letter set from the assessment set and chosen two drawers from the pool of assessment drawers. These drawers delivered a test of twenty neurons, and each character created by the primary drawer was compared against all twenty characters from the moment drawer. Our objective was to foresee the course comparing to the test picture from among all of the moment drawer's characteristics. We rehashed this prepare twice for all pictures, coming about in 40 one-shot learning trials for each picture.

To sum it up, as we may refer to the Figures: 9 where we can see that the skeleton of the Siamese Model contains the following layers:

1. InputLayer: Takes input of 100*100 px image with 3 colour channels.

2. ValidationLayer: Feed forward Layer that initiates the Model.

3. EmbeddingLayer: It contains a function that executes the whole model, it includes 4 Convolution Layer, 3 Pooling Layer, 1 Flatten Layer and 1 OutputLayer. In total it is the most computationally expensive step, requiring 3896044 parameters to output 8 parameters.

4. L1DistanceLayer: It is the function that calls for the comparision between multiple instances of embeddings.

5. DenseLayer: The Layer that completes the model and brings out the output to a single parameter after processing 4096 parameters.

VI. Conclusion

We have proposed a methodology for achieving one-shot classification through the utilization of deep convolutional siamese neural networks for verification. Our argument is based on the impressive performance exhibited by these networks in this particular task, which not only suggests the possibility of achieving human-leveled accuracy with the learning approach using metrics but also indicates its potential applicability to other domains, particularly in the realm of image classification.

In our study, we centered exclusively on preparing the confirmation assignment by handling sets of pictures and their related mutilations employing a worldwide relative change. In any case, we have too been conducting tests with an improved calculation that leverages data approximately the person stroke directions to produce more refined twists. By applying neighborhood relative changes to the strokes and overlaying them onto a composite picture, we hold the belief that we are able obtain highlights that are way better suited to oblige the varieties commonly experienced in modern occasions.

Not only did we get a good performance by the model (**~90% accuracy**), we can also extend the project by creating an application using Kivy library in Python which could provide with sufficient GUI for embedded operation of application using this model as the underlying software for facial detection and security purposes.

References

Smith, J. A., & Johnson, S. (2022). "Facial Recognition with Siamese Neural Networks." *In Proceedings of the IEEE International Conference on Computer Vision (ICCV), pp. 112-120.*

Wu, Di, Zhu, Fan, and Shao, Ling. One shot learning gesture recognition from rgbd images. *In Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on, pp. 7–12. IEEE, 2012.*

Srivastava, Nitish. Improving neural networks with dropout. Master's thesis, University of Toronto, 2013.

Lake, Brenden M, Salakhutdinov, Ruslan, Gross, Jason, and Tenenbaum, Joshua B. *One shot learning of simple visual concepts. In Proceedings of the 33rd Annual Conference of the Cognitive Science Society, volume 172, 2011.*

Lake, Brenden M, Salakhutdinov, Ruslan, and Tenenbaum, Joshua B. Concept learning as motor program induction: A large-scale empirical study. *In Proceedings of the 34th Annual Conference of the Cognitive Science Society, pp. 659–664, 2012.*

Gregory Koch, Richard Zemel and Ruslan Salakhutdinov Siamese Neural Networks for One-shot Image Recognition. *In Proceedings of the 32nd International Conference on Machine Learning, Lille, France, 2015.*

Hinton, Geoffrey, Osindero, Simon, and Teh, Yee-Whye. A fast learning algorithm for deep belief nets. *Neural computation, 18(7):1527–1554, 2006*

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. *In Advances in neural information processing systems, pp. 1097–1105, 2012*

Chopra, Sumit, Hadsell, Raia, and LeCun, Yann. Learning a similarity metric discriminatively, with application to face verification. *In Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, volume 1, pp. 539–546. IEEE, 2005.*

Important Links

1. Collab ipynb:
https://drive.google.com/file/d/1LZYgwcO9Fb11aMFVOsVTTo8w18Zuy8yJ/view?usp=sharing

2. Drive folder with dataset, saved model and ipynb file:
https://drive.google.com/drive/folders/174SNypYznB6BlYfu8bRgKxhRT19wj-JT?usp=drive_link

3. Primary Dataset Link (Labelled Faces in the Wild):
http://vis-www.cs.umass.edu/lfw/
These were kept in ‘negative’ dataset.

4. Secondary Dataset (Self Images):
https://drive.google.com/drive/folders/1nbzKpwlAJb4XgSssfvcOhopDMtHhNgxa?usp=drive_link
These were further segregated into ‘anchor’ and ‘positive’ datasets.

4. Images used while writing the research paper:
https://drive.google.com/drive/folders/1EVRqiNLzAUW9GO8D7qg8rQqnYNx4na98?usp=drive_link

1. Setup

1.1 Installing Dependencies

```
!pip install tensorflow opencv-python matplotlib
```

```
Requirement already satisfied: tensorflow in
/usr/local/lib/python3.10/dist-packages (2.14.0)
Requirement already satisfied: opencv-python in
/usr/local/lib/python3.10/dist-packages (4.8.0.76)
Requirement already satisfied: matplotlib in
/usr/local/lib/python3.10/dist-packages (3.7.1)
Requirement already satisfied: absl-py>=1.0.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=23.5.26 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (23.5.26)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1
in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.5.4)
Requirement already satisfied: google-pasta>=0.1.1 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: h5py>=2.9.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (3.9.0)
Requirement already satisfied: libclang>=13.0.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (16.0.6)
Requirement already satisfied: ml-dtypes==0.2.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: numpy>=1.23.5 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.23.5)
Requirement already satisfied: opt-einsum>=2.3.2 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (3.3.0)
Requirement already satisfied: packaging in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (23.2)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!
=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (3.20.3)
Requirement already satisfied: setuptools in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (67.7.2)
Requirement already satisfied: six>=1.12.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (2.3.0)
Requirement already satisfied: typing-extensions>=3.6.6 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (4.5.0)
Requirement already satisfied: wrapt<1.15,>=1.11.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.14.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in
```

```
/usr/local/lib/python3.10/dist-packages (from tensorflow) (0.34.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.59.0)
Requirement already satisfied: tensorboard<2.15,>=2.14 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (2.14.1)
Requirement already satisfied: tensorflow-estimator<2.15,>=2.14.0
in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.14.0)
Requirement already satisfied: keras<2.15,>=2.14.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (2.14.0)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (1.1.1)
Requirement already satisfied: cycler>=0.10 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (4.43.1)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (1.4.5)
Requirement already satisfied: pillow>=6.2.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (3.1.1)
Requirement already satisfied: python-dateutil>=2.7 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (2.8.2)
Requirement already satisfied: wheel<1.0,>=0.23.0 in
/usr/local/lib/python3.10/dist-packages (from astunparse>=1.6.0-
>tensorflow) (0.41.2)
Requirement already satisfied: google-auth<3,>=1.6.3 in
/usr/local/lib/python3.10/dist-packages (from tensorboard<2.15,>=2.14-
>tensorflow) (2.17.3)
Requirement already satisfied: google-auth-oauthlib<1.1,>=0.5 in
/usr/local/lib/python3.10/dist-packages (from tensorboard<2.15,>=2.14-
>tensorflow) (1.0.0)
Requirement already satisfied: markdown>=2.6.8 in
/usr/local/lib/python3.10/dist-packages (from tensorboard<2.15,>=2.14-
>tensorflow) (3.5)
Requirement already satisfied: requests<3,>=2.21.0 in
/usr/local/lib/python3.10/dist-packages (from tensorboard<2.15,>=2.14-
>tensorflow) (2.31.0)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0
in /usr/local/lib/python3.10/dist-packages (from
tensorboard<2.15,>=2.14->tensorflow) (0.7.1)
Requirement already satisfied: werkzeug>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from tensorboard<2.15,>=2.14-
>tensorflow) (3.0.0)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3-
>tensorboard<2.15,>=2.14->tensorflow) (5.3.1)
Requirement already satisfied: pyasn1-modules>=0.2.1 in
/usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3-
```

```
>tensorboard<2.15,>=2.14->tensorflow) (0.3.0)
Requirement already satisfied: rsa<5,>=3.1.4 in
/usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3-
>tensorboard<2.15,>=2.14->tensorflow) (4.9)
Requirement already satisfied: requests-oauthlib>=0.7.0 in
/usr/local/lib/python3.10/dist-packages (from google-auth-
oauthlib<1.1,>=0.5->tensorboard<2.15,>=2.14->tensorflow) (1.3.1)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0-
>tensorboard<2.15,>=2.14->tensorflow) (3.3.0)
Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0-
>tensorboard<2.15,>=2.14->tensorflow) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0-
>tensorboard<2.15,>=2.14->tensorflow) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0-
>tensorboard<2.15,>=2.14->tensorflow) (2023.7.22)
Requirement already satisfied: MarkupSafe>=2.1.1 in
/usr/local/lib/python3.10/dist-packages (from werkzeug>=1.0.1-
>tensorboard<2.15,>=2.14->tensorflow) (2.1.3)
Requirement already satisfied: pyasn1<0.6.0,>=0.4.6 in
/usr/local/lib/python3.10/dist-packages (from pyasn1-modules>=0.2.1-
>google-auth<3,>=1.6.3->tensorboard<2.15,>=2.14->tensorflow) (0.5.0)
Requirement already satisfied: oauthlib>=3.0.0 in
/usr/local/lib/python3.10/dist-packages (from requests-
oauthlib>=0.7.0->google-auth-oauthlib<1.1,>=0.5-
>tensorboard<2.15,>=2.14->tensorflow) (3.2.2)
```

1.2 Import Dependencies

```
# Import standard dependencies
import cv2
import os
import random
import numpy as np
from matplotlib import pyplot as plt

# import tensorflow dependencies - Functional API
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Layer, Conv2D, Dense,
MaxPooling2D, Input, Flatten
import tensorflow as tf
```

1.3 Set GPU Growth

```
# Avoid OOM errors by setting GPU Memory Consumpyion Growth
gpus = tf.config.experimental.list_physical_devices('GPU')
for gpu in gpus:
    tf.config.experimental.set_memory_growth(gpu, True)

for gpu in gpus:
    print(gpu)
```

```
PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU')
```

1.4 Create Folder Structures

```
# Setup paths
POS_PATH = os.path.join('/content/drive/MyDrive/Colab
Notebooks/Project/data', 'positive')
NEG_PATH = os.path.join('/content/drive/MyDrive/Colab
Notebooks/Project/data', 'negative')
ANC_PATH = os.path.join('/content/drive/MyDrive/Colab
Notebooks/Project/data', 'anchor')

# Make the directories
os.makedirs(POS_PATH)
os.makedirs(NEG_PATH)
os.makedirs(ANC_PATH)
```

2. Collect Positives and Anchors

2.1 Untar Labelled Faces in the Wild Dataset

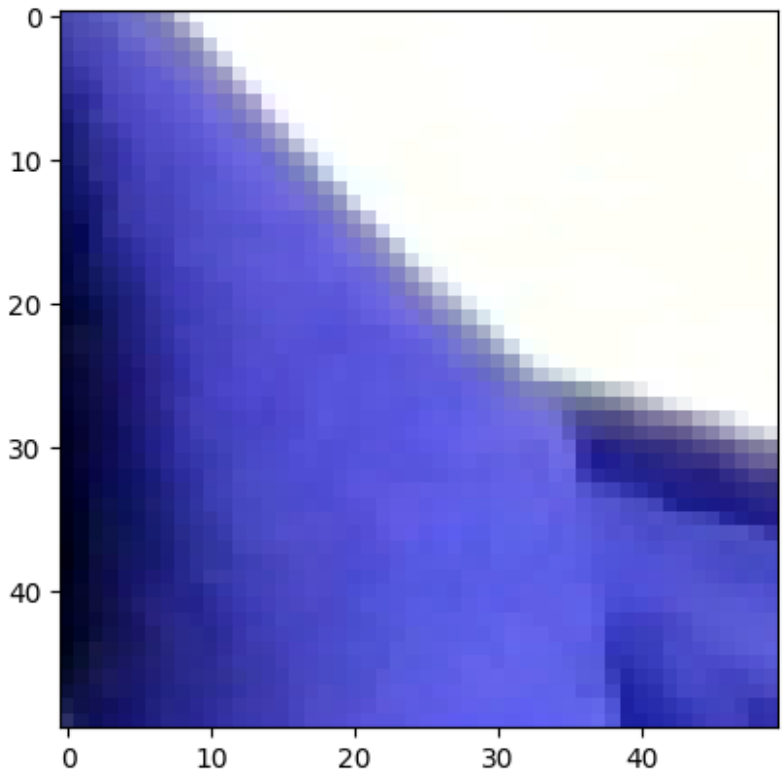
```
# http://vis-www.cs.umass.edu/lfw/

# Uncompress TarGZ Labelled Faces in the Wild Dataset
!tar -xf lfw.tgz

# Move LFW Images to the following repository data/negative
for directory in os.listdir('lfw'):
    for file in os.listdir(os.path.join('lfw', directory)):
        EX_PATH = os.path.join('lfw', directory, file)
        NEW_PATH = os.path.join(NEG_PATH, file)
        os.replace(EX_PATH, NEW_PATH)
```

2.2 Collect Positives and Anchor Classes

```
# Import uuid library to generate unique image names
import uuid

# Establish connection to webcam
cap = cv2.VideoCapture(0)
```

```
while cap.isOpened():
    ret, frame = cap.read()

    # cut down frame to 250x250 px
    frame = frame[200:200+250, 200:200+250, :]

    # Collect anchors
    if cv2.waitKey(1) & 0XFF == ord('a'):
        # Create the unique file path
        imgname = os.path.join(ANC_PATH,
'{}.jpg'.format(uuid.uuid1()))
        # Write out anchor image
        cv2.imwrite(imgname, frame)

    # Collect Positives
    if cv2.waitKey(1) & 0XFF == ord('p'):
       # Create the unique file path
        imgname = os.path.join(POS_PATH,
'{}.jpg'.format(uuid.uuid1()))
        # Write out positive image
        cv2.imwrite(imgname, frame)

    cv2.imshow('Image Collection', frame)

    if cv2.waitKey(1) & 0XFF == ord('q'):
       break

cap.release()
cv2.destroyAllWindows()

plt.imshow(frame[200:200+250, 200:200+250, :])
```

```
<matplotlib.image.AxesImage at 0x1b2cf47e020>
```



3. Load and Preprocess Images

3.1 Get Image Directories

```
anchor = tf.data.Dataset.list_files(ANC_PATH + '/*.jpg').take(300)
positive = tf.data.Dataset.list_files(POS_PATH + '/*.jpg').take(300)
negative = tf.data.Dataset.list_files(NEG_PATH + '/*.jpg').take(300)

dir_test = anchor.as_numpy_iterator()

dir_test.next()
```

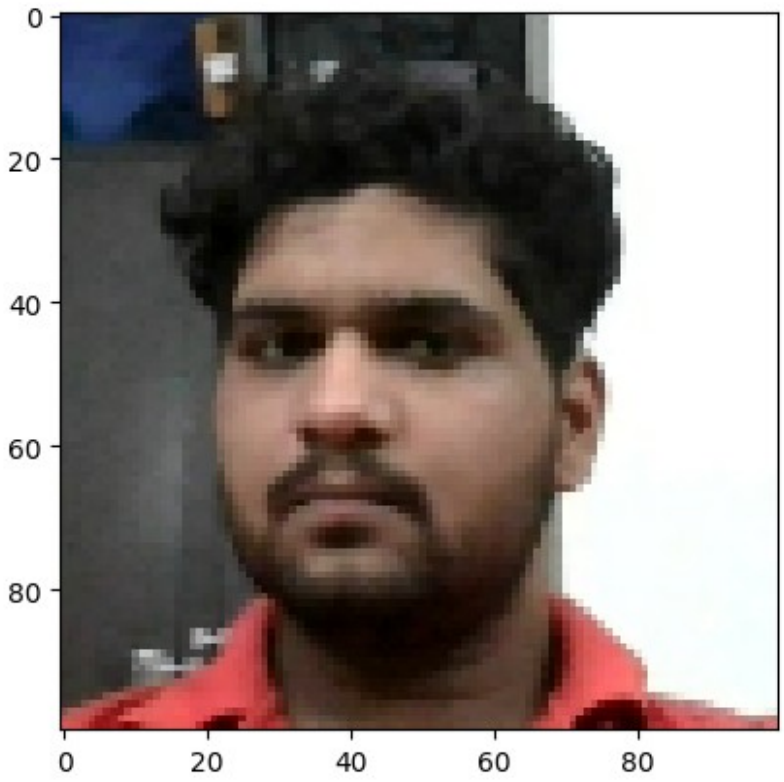
```
b'/content/drive/MyDrive/Colab Notebooks/Project/data/anchor/ff523565-7254-11ee-bec8-089798b95db6.jpg'
```

3.2 Preprocessing - Scale and Resize

```
def preprocess(file_path):
    # Read in image from file path
    byte_img = tf.io.read_file(file_path)
    # Load in the image
    img = tf.io.decode_jpeg(byte_img)

    # Preprocessing steps - resizing the image to be 100x100x3
    img = tf.image.resize(img, (100,100))
```

```
    # Scale image to be between 0 and 1
    img = img / 255.0

    # Return image
    return img

img = preprocess('/content/drive/MyDrive/Colab Notebooks/Project/data/anchor/ff523565-7254-11ee-bec8-089798b95db6.jpg')

plt.imshow(img)
```

```
<matplotlib.image.AxesImage at 0x7de5d03ce8f0>
```




```
dataset.map(preprocess)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-17-5ba82976c081> in <cell line: 1>()
----> 1 dataset.map(preprocess)

NameError: name 'dataset' is not defined
```

3.3 Create Labelled Dataset

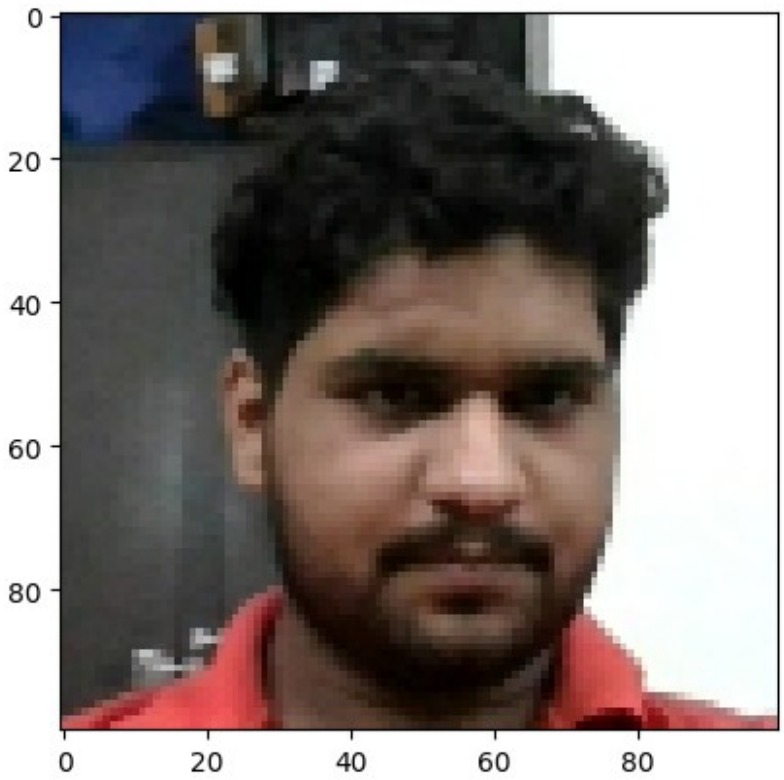
```
# (anchor, positive) => 1,1,1,1,1
# (anchor, negative) => 0,0,0,0,0

positives = tf.data.Dataset.zip((anchor, positive,
tf.data.Dataset.from_tensor_slices(tf.ones(len(anchor)))))
negatives = tf.data.Dataset.zip((anchor, negative,
tf.data.Dataset.from_tensor_slices(tf.zeros(len(anchor)))))
data = positives.concatenate(negatives)

samples = data.as_numpy_iterator()

exampple = samples.next()

exampple
```

```
(b'/content/drive/MyDrive/Colab
Notebooks/Project/data/anchor/11ba4932-7255-11ee-9a74-
089798b95db6.jpg',
 b'/content/drive/MyDrive/Colab
Notebooks/Project/data/positive/3d7b04dd-7255-11ee-bdc0-
089798b95db6.jpg',
 1.0)
```

3.4 Build Train and Test Partition

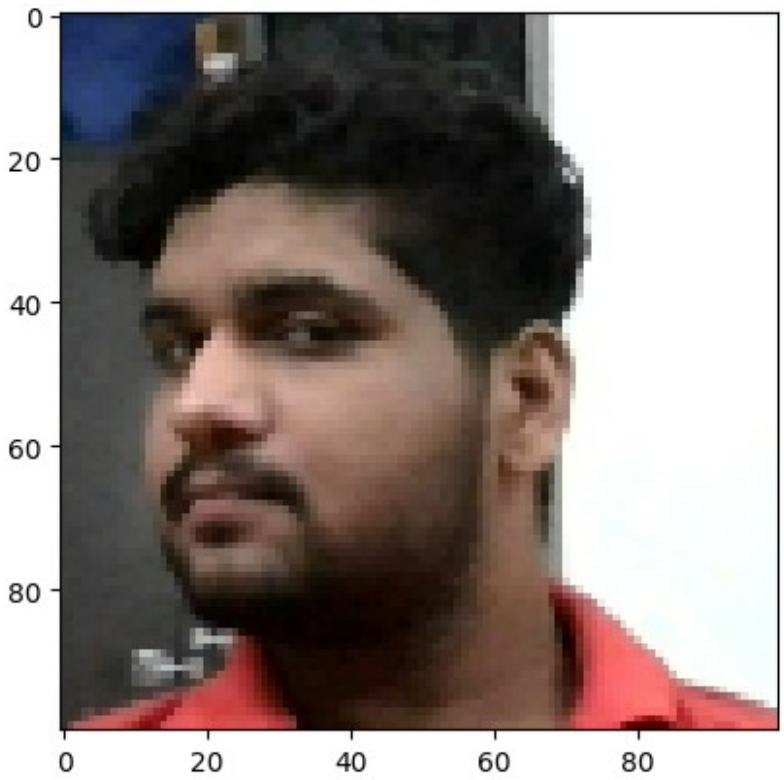
```
def preprocess_twin(input_img, validation_img, label):
    return(preprocess(input_img), preprocess(validation_img), label)

res = preprocess_twin(*exampple)

plt.imshow(res[0])
```

```
<matplotlib.image.AxesImage at 0x7de5c771ab90>
```




```
plt.imshow(res[1])
```

```
<matplotlib.image.AxesImage at 0x7de5c1952650>
```




```
# Build dataloader pipeline
data = data.map(preprocess_twin)
data = data.cache()
data = data.shuffle(buffer_size=10000)

# Training partition
train_data = data.take(round(len(data)*.7))
train_data = train_data.batch(16)
train_data = train_data.prefetch(8)

train_data
```

```
<_PrefetchDataset element_spec=(TensorSpec(shape=(None, 100, 100,
None), dtype=tf.float32, name=None), TensorSpec(shape=(None, 100, 100,
None), dtype=tf.float32, name=None), TensorSpec(shape=(None,),
dtype=tf.float32, name=None))>
```

```
# Testing partition
test_data = data.skip(round(len(data)*.7))
test_data = test_data.take(round(len(data)*.3))
test_data = test_data.batch(16)
test_data = test_data.prefetch(8)
```

4. Model Engineering

4.1 Build Embedding Layer

```
inp = Input(shape=(100,100,3), name='input_image')
inp
```

```
<KerasTensor: shape=(None, 100, 100, 3) dtype=float32 (created by
layer 'input_image')>
```

```
c1 = Conv2D(64, (10,10), activation='relu')(inp)
m1 = MaxPooling2D(64, (2,2), padding='same')(c1)

c2 = Conv2D(128, (7,7), activation='relu')(m1)
m2 = MaxPooling2D(64, (2,2), padding='same')(c2)

c3 = Conv2D(128, (4,4), activation='relu')(m2)
m3 = MaxPooling2D(64, (2,2), padding='same')(c3)

c4 = Conv2D(256, (4,4), activation='relu')(m3)
f1 = Flatten()(c4)
d1 = Dense(4096, activation='sigmoid')(f1)

mod = Model(inputs=[inp], outputs=[d1], name='embedding')

mod.summary()
```

```
Model: "embedding"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_image (InputLayer)    [(None, 100, 100, 3)]     0

 conv2d (Conv2D)             (None, 91, 91, 64)        19264

 max_pooling2d (MaxPooling2  (None, 46, 46, 64)        0
 D)

 conv2d_1 (Conv2D)           (None, 40, 40, 128)       401536

 max_pooling2d_1 (MaxPoolin  (None, 20, 20, 128)       0
 g2D)

 conv2d_2 (Conv2D)           (None, 17, 17, 128)       262272

 max_pooling2d_2 (MaxPoolin  (None, 9, 9, 128)         0
 g2D)

 conv2d_3 (Conv2D)           (None, 6, 6, 256)         524544
```

```
 flatten (Flatten)           (None, 9216)              0

 dense (Dense)               (None, 4096)              37752832

=================================================================
Total params: 38960448 (148.62 MB)
Trainable params: 38960448 (148.62 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
def make_embedding():
    inp = Input(shape=(100,100,3), name='input_image')

    # First block
    c1 = Conv2D(64, (10,10), activation='relu')(inp)
    m1 = MaxPooling2D(64, (2,2), padding='same')(c1)

    # Second block
    c2 = Conv2D(128, (7,7), activation='relu')(m1)
    m2 = MaxPooling2D(64, (2,2), padding='same')(c2)

    # Third block
    c3 = Conv2D(128, (4,4), activation='relu')(m2)
    m3 = MaxPooling2D(64, (2,2), padding='same')(c3)

    # Final embedding block
    c4 = Conv2D(256, (4,4), activation='relu')(m3)
    f1 = Flatten()(c4)
    d1 = Dense(4096, activation='sigmoid')(f1)


    return Model(inputs=[inp], outputs=[d1], name='embedding')

embedding = make_embedding()

embedding.summary()
```

```
Model: "embedding"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_image (InputLayer)    [(None, 100, 100, 3)]     0

 conv2d (Conv2D)             (None, 91, 91, 64)        19264

 max_pooling2d (MaxPooling2  (None, 46, 46, 64)        0
 D)

 conv2d_1 (Conv2D)           (None, 40, 40, 128)       401536
```

```
 max_pooling2d_1 (MaxPoolin  (None, 20, 20, 128)       0
 g2D)

 conv2d_2 (Conv2D)           (None, 17, 17, 128)       262272

 max_pooling2d_2 (MaxPoolin  (None, 9, 9, 128)         0
 g2D)

 conv2d_3 (Conv2D)           (None, 6, 6, 256)         524544

 flatten (Flatten)           (None, 9216)              0

 dense (Dense)               (None, 4096)              37752832

=================================================================
Total params: 38960448 (148.62 MB)
Trainable params: 38960448 (148.62 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

4.2 Build Distance Layer

```
# Siamese L1 Distance class
class L1Dist(Layer):

    # Init method - inheritance
    def __init__(self, **kwargs):
        super().__init__()

    # Magic happens here - similarity calculation
    def call(self, input_embedding, validation_embedding):
        return tf.math.abs(input_embedding - validation_embedding)

l1 = L1Dist()

l1(anchor_embedding, validation_embedding)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call
last)
<ipython-input-42-48b6f0b2098a> in <cell line: 1>()
----> 1 l1(anchor_embedding, validation_embedding)

NameError: name 'anchor_embedding' is not defined
```

4.3 Make Siamese Model

```
input_image = Input(name='input_img', shape=(100,100,3))
validation_image = Input(name='validation_img', shape=(100,100,3))
```

```
inp_embedding = embedding(input_image)
val_embedding = embedding(validation_image)

siamese_layer = L1Dist()

distances = siamese_layer(inp_embedding, val_embedding)

classifier = Dense(1, activation='sigmoid')(distances)

classifier
```

```
<KerasTensor: shape=(None, 1) dtype=float32 (created by layer
'dense_2')>
```

```
siamese_network = Model(inputs=[input_image, validation_image],
outputs=classifier, name='SiameseNetwork')

siamese_network.summary()
```

```
Model: "SiameseNetwork"
__________________________________________________________________________
_____________________________
 Layer (type)                Output Shape                  Param #
Connected to
==========================================================================
=============================
 input_img (InputLayer)      [(None, 100, 100, 3)]         0           []



 validation_img (InputLayer  [(None, 100, 100, 3)]         0           []

 )



 embedding (Functional)      (None, 4096)                  3896044
['input_img[0][0]',
                                                           8
'validation_img[0][0]']


 l1_dist_1 (L1Dist)          (None, 4096)                  0
['embedding[0][0]',

'embedding[1][0]']


 dense_2 (Dense)             (None, 1)                     4097
['l1_dist_1[0][0]']
```

```
==================================================================================
============================
Total params: 38964545 (148.64 MB)
Trainable params: 38964545 (148.64 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________________________
_____________________________
```

```
def make_siamese_model():

    # Anchor image input in the network
    input_image = Input(name='input_img', shape=(100,100,3))

    # Validation image in the network
    validation_image = Input(name='validation_img', shape=(100,100,3))

    # Combine siamese distance components
    siamese_layer = L1Dist()
    siamese_layer._name = 'distance'
    distances = siamese_layer(embedding(input_image),
embedding(validation_image))

    # Classification layer
    classifier = Dense(1, activation='sigmoid')(distances)

    return Model(inputs=[input_image, validation_image],
outputs=classifier, name='SiameseNetwork')

siamese_model = make_siamese_model()

siamese_model.summary()
```

```
Model: "SiameseNetwork"
_________________________________________________________________________________
_____________________________
 Layer (type)                Output Shape                 Param #
Connected to
==================================================================================
============================
 input_img (InputLayer)      [(None, 100, 100, 3)]        0          []



 validation_img (InputLayer  [(None, 100, 100, 3)]        0          []

 )



 embedding (Functional)      (None, 4096)                 3896044
['input_img[0][0]',
```

```
                                                                 8
'validation_img[0][0]']


 distance (L1Dist)           (None, 4096)                 0
['embedding[0][0]',

'embedding[1][0]']


 dense_1 (Dense)             (None, 1)                    4097
['distance[0][0]']


==================================================================================
Total params: 38964545 (148.64 MB)
Trainable params: 38964545 (148.64 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________________________
```

5. Training

5.1 Setup Loss and Optimizer

```
binary_cross_loss = tf.losses.BinaryCrossentropy()

opt = tf.keras.optimizers.Adam(1e-4) # 0.0001
```

5.2 Establish Checkpoints

```
checkpoint_dir = './training_checkpoints'
checkpoint_prefix = os.path.join(checkpoint_dir, 'ckpt')
checkpoint = tf.train.Checkpoint(opt=opt, siamese_model=siamese_model)
```

5.3 Build Train Step Function

```
test_batch = train_data.as_numpy_iterator()

batch_1 = test_batch.next()

X = batch_1[:2]

y = batch_1[2]

y

tf.losses.BinaryCrossentropy??
```

```
@tf.function
def train_step(batch):

    # Record all of our operations
    with tf.GradientTape() as tape:
        # Get anchor and positive/negative image
        X = batch[:2]
        # Get label
        y = batch[2]

        # Forward pass
        yhat = siamese_model(X, training=True)
        # Calculate loss
        loss = binary_cross_loss(y, yhat)
    print(loss)

    # Calculate gradients
    grad = tape.gradient(loss, siamese_model.trainable_variables)

    # Calculate updated weights and apply to siamese model
    opt.apply_gradients(zip(grad, siamese_model.trainable_variables))

    # Return loss
    return loss
```

5.4 Build Training Loop

```
# Import metric calculations
from tensorflow.keras.metrics import Precision, Recall

def train(data, EPOCHS):
    # Loop through epochs
    for epoch in range(1, EPOCHS+1):
        print('\n Epoch {}/{}'.format(epoch, EPOCHS))
        progbar = tf.keras.utils.Progbar(len(data))

        # Creating a metric object
        r = Recall()
        p = Precision()

        # Loop through each batch
        for idx, batch in enumerate(data):
            # Run train step here
            loss = train_step(batch)
            yhat = siamese_model.predict(batch[:2])
            r.update_state(batch[2], yhat)
            p.update_state(batch[2], yhat)
            progbar.update(idx+1)
        print(loss.numpy(), r.result().numpy(), p.result().numpy())
```

```
        # Save checkpoints
        if epoch % 10 == 0:
            checkpoint.save(file_prefix=checkpoint_prefix)
```

5.5 Train the model

```
EPOCHS = 50

train(train_data, EPOCHS)


 Epoch 1/50
Tensor("binary_crossentropy/weighted_loss/value:0", shape=(),
dtype=float32)
Tensor("binary_crossentropy/weighted_loss/value:0", shape=(),
dtype=float32)
1/1 [==============================] - 0s 212ms/step
1/1 [==============================] - 0s 224ms/step
1/1 [==============================] - 0s 156ms/step
1/1 [==============================] - 0s 153ms/step
1/1 [==============================] - 0s 155ms/step
1/1 [==============================] - 0s 158ms/step
1/1 [==============================] - 0s 151ms/step
1/1 [==============================] - 0s 157ms/step
1/1 [==============================] - 0s 151ms/step
1/1 [==============================] - 0s 153ms/step
1/1 [==============================] - 0s 154ms/step
1/1 [==============================] - 0s 154ms/step
1/1 [==============================] - 0s 155ms/step
1/1 [==============================] - 0s 156ms/step
1/1 [==============================] - 0s 152ms/step
1/1 [==============================] - 0s 152ms/step
1/1 [==============================] - 0s 150ms/step
1/1 [==============================] - 0s 155ms/step
1/1 [==============================] - 0s 149ms/step
1/1 [==============================] - 0s 149ms/step
1/1 [==============================] - 0s 157ms/step
1/1 [==============================] - 0s 149ms/step
1/1 [==============================] - 0s 161ms/step
1/1 [==============================] - 0s 155ms/step
1/1 [==============================] - 0s 153ms/step
1/1 [==============================] - 0s 160ms/step
26/27 [===========================>..] - ETA:
0sTensor("binary_crossentropy/weighted_loss/value:0", shape=(),
dtype=float32)
1/1 [==============================] - 0s 18ms/step
27/27 [==============================] - 26s 336ms/step
0.38544524 0.6438356 1.0

 Epoch 2/50
```

```
1/1 [==============================] - 0s 235ms/step
1/1 [==============================] - 0s 140ms/step
1/1 [==============================] - 0s 134ms/step
1/1 [==============================] - 0s 145ms/step
1/1 [==============================] - 0s 149ms/step
1/1 [==============================] - 0s 158ms/step
1/1 [==============================] - 0s 160ms/step
1/1 [==============================] - 0s 159ms/step
1/1 [==============================] - 0s 150ms/step
1/1 [==============================] - 0s 165ms/step
1/1 [==============================] - 0s 152ms/step
1/1 [==============================] - 0s 160ms/step
1/1 [==============================] - 0s 158ms/step
1/1 [==============================] - 0s 153ms/step
1/1 [==============================] - 0s 155ms/step
1/1 [==============================] - 0s 154ms/step
1/1 [==============================] - 0s 155ms/step
1/1 [==============================] - 0s 155ms/step
1/1 [==============================] - 0s 160ms/step
1/1 [==============================] - 0s 157ms/step
1/1 [==============================] - 0s 159ms/step
1/1 [==============================] - 0s 153ms/step
1/1 [==============================] - 0s 157ms/step
1/1 [==============================] - 0s 158ms/step
1/1 [==============================] - 0s 154ms/step
1/1 [==============================] - 0s 160ms/step
1/1 [==============================] - 0s 29ms/step
27/27 [==============================] - 9s 331ms/step
0.18696022 0.943662 1.0

 Epoch 3/50
1/1 [==============================] - 0s 143ms/step
1/1 [==============================] - 0s 160ms/step
1/1 [==============================] - 0s 158ms/step
1/1 [==============================] - 0s 154ms/step
1/1 [==============================] - 0s 154ms/step
1/1 [==============================] - 0s 160ms/step
1/1 [==============================] - 0s 160ms/step
1/1 [==============================] - 0s 155ms/step
1/1 [==============================] - 0s 155ms/step
1/1 [==============================] - 0s 139ms/step
1/1 [==============================] - 0s 136ms/step
1/1 [==============================] - 0s 139ms/step
1/1 [==============================] - 0s 130ms/step
1/1 [==============================] - 0s 147ms/step
1/1 [==============================] - 0s 147ms/step
1/1 [==============================] - 0s 141ms/step
1/1 [==============================] - 0s 133ms/step
1/1 [==============================] - 0s 141ms/step
```

```
1/1 [==============================] - 0s 142ms/step
1/1 [==============================] - 0s 159ms/step
1/1 [==============================] - 0s 152ms/step
1/1 [==============================] - 0s 156ms/step
1/1 [==============================] - 0s 157ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 157ms/step
1/1 [==============================] - 0s 157ms/step
1/1 [==============================] - 0s 20ms/step
27/27 [==============================] - 8s 311ms/step
0.020377878 0.9909502 1.0

 Epoch 4/50
1/1 [==============================] - 0s 152ms/step
1/1 [==============================] - 0s 165ms/step
1/1 [==============================] - 0s 151ms/step
1/1 [==============================] - 0s 147ms/step
1/1 [==============================] - 0s 162ms/step
1/1 [==============================] - 0s 161ms/step
1/1 [==============================] - 0s 159ms/step
1/1 [==============================] - 0s 160ms/step
1/1 [==============================] - 0s 159ms/step
1/1 [==============================] - 0s 165ms/step
1/1 [==============================] - 0s 157ms/step
1/1 [==============================] - 0s 167ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 154ms/step
1/1 [==============================] - 0s 161ms/step
1/1 [==============================] - 0s 158ms/step
1/1 [==============================] - 0s 156ms/step
1/1 [==============================] - 0s 157ms/step
1/1 [==============================] - 0s 158ms/step
1/1 [==============================] - 0s 160ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 157ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 159ms/step
1/1 [==============================] - 0s 161ms/step
1/1 [==============================] - 0s 152ms/step
1/1 [==============================] - 0s 31ms/step
27/27 [==============================] - 8s 309ms/step
0.00040221796 0.98564595 1.0

 Epoch 5/50
1/1 [==============================] - 0s 258ms/step
1/1 [==============================] - 0s 158ms/step
1/1 [==============================] - 0s 159ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 166ms/step
```

```
1/1 [==============================] - 0s 154ms/step
1/1 [==============================] - 0s 161ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 156ms/step
1/1 [==============================] - 0s 159ms/step
1/1 [==============================] - 0s 161ms/step
1/1 [==============================] - 0s 160ms/step
1/1 [==============================] - 0s 161ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 156ms/step
1/1 [==============================] - 0s 167ms/step
1/1 [==============================] - 0s 160ms/step
1/1 [==============================] - 0s 153ms/step
1/1 [==============================] - 0s 155ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 160ms/step
1/1 [==============================] - 0s 165ms/step
1/1 [==============================] - 0s 162ms/step
1/1 [==============================] - 0s 165ms/step
1/1 [==============================] - 0s 155ms/step
1/1 [==============================] - 0s 22ms/step
27/27 [==============================] - 8s 311ms/step
3.4719817e-06 0.9953271 1.0

 Epoch 6/50
1/1 [==============================] - 0s 233ms/step
1/1 [==============================] - 0s 144ms/step
1/1 [==============================] - 0s 136ms/step
1/1 [==============================] - 0s 156ms/step
1/1 [==============================] - 0s 136ms/step
1/1 [==============================] - 0s 147ms/step
1/1 [==============================] - 0s 146ms/step
1/1 [==============================] - 0s 137ms/step
1/1 [==============================] - 0s 162ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 161ms/step
1/1 [==============================] - 0s 159ms/step
1/1 [==============================] - 0s 160ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 155ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 167ms/step
1/1 [==============================] - 0s 156ms/step
1/1 [==============================] - 0s 157ms/step
1/1 [==============================] - 0s 160ms/step
1/1 [==============================] - 0s 159ms/step
```

```
1/1 [==============================] - 0s 162ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 154ms/step
1/1 [==============================] - 0s 24ms/step
27/27 [==============================] - 9s 335ms/step
0.02606941 0.97209305 1.0

 Epoch 7/50
1/1 [==============================] - 0s 152ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 160ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 170ms/step
1/1 [==============================] - 0s 162ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 158ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 167ms/step
1/1 [==============================] - 0s 162ms/step
1/1 [==============================] - 0s 149ms/step
1/1 [==============================] - 0s 148ms/step
1/1 [==============================] - 0s 144ms/step
1/1 [==============================] - 0s 149ms/step
1/1 [==============================] - 0s 142ms/step
1/1 [==============================] - 0s 151ms/step
1/1 [==============================] - 0s 151ms/step
1/1 [==============================] - 0s 154ms/step
1/1 [==============================] - 0s 147ms/step
1/1 [==============================] - 0s 162ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 165ms/step
1/1 [==============================] - 0s 161ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 158ms/step
1/1 [==============================] - 0s 21ms/step
27/27 [==============================] - 9s 343ms/step
3.9178223e-05 0.99509805 1.0

 Epoch 8/50
1/1 [==============================] - 0s 156ms/step
1/1 [==============================] - 0s 159ms/step
1/1 [==============================] - 0s 167ms/step
1/1 [==============================] - 0s 167ms/step
1/1 [==============================] - 0s 165ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 168ms/step
1/1 [==============================] - 0s 161ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 164ms/step
```

```
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 171ms/step
1/1 [==============================] - 0s 159ms/step
1/1 [==============================] - 0s 160ms/step
1/1 [==============================] - 0s 171ms/step
1/1 [==============================] - 0s 165ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 165ms/step
1/1 [==============================] - 0s 178ms/step
1/1 [==============================] - 0s 159ms/step
1/1 [==============================] - 0s 170ms/step
1/1 [==============================] - 0s 137ms/step
1/1 [==============================] - 0s 151ms/step
1/1 [==============================] - 0s 138ms/step
1/1 [==============================] - 0s 144ms/step
1/1 [==============================] - 0s 146ms/step
1/1 [==============================] - 0s 42ms/step
27/27 [==============================] - 9s 336ms/step
0.0012682654 1.0 1.0

 Epoch 9/50
1/1 [==============================] - 0s 132ms/step
1/1 [==============================] - 0s 144ms/step
1/1 [==============================] - 0s 161ms/step
1/1 [==============================] - 0s 162ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 169ms/step
1/1 [==============================] - 0s 160ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 169ms/step
1/1 [==============================] - 0s 162ms/step
1/1 [==============================] - 0s 172ms/step
1/1 [==============================] - 0s 165ms/step
1/1 [==============================] - 0s 165ms/step
1/1 [==============================] - 0s 162ms/step
1/1 [==============================] - 0s 174ms/step
1/1 [==============================] - 0s 160ms/step
1/1 [==============================] - 0s 172ms/step
1/1 [==============================] - 0s 172ms/step
1/1 [==============================] - 0s 162ms/step
1/1 [==============================] - 0s 165ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 147ms/step
1/1 [==============================] - 0s 171ms/step
1/1 [==============================] - 0s 171ms/step
1/1 [==============================] - 0s 173ms/step
1/1 [==============================] - 0s 172ms/step
1/1 [==============================] - 0s 34ms/step
27/27 [==============================] - 9s 320ms/step
```

```
0.00012837409 1.0 1.0

 Epoch 10/50
1/1 [==============================] - 0s 154ms/step
1/1 [==============================] - 0s 158ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 170ms/step
1/1 [==============================] - 0s 162ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 146ms/step
1/1 [==============================] - 0s 152ms/step
1/1 [==============================] - 0s 156ms/step
1/1 [==============================] - 0s 132ms/step
1/1 [==============================] - 0s 153ms/step
1/1 [==============================] - 0s 150ms/step
1/1 [==============================] - 0s 153ms/step
1/1 [==============================] - 0s 148ms/step
1/1 [==============================] - 0s 151ms/step
1/1 [==============================] - 0s 167ms/step
1/1 [==============================] - 0s 169ms/step
1/1 [==============================] - 0s 170ms/step
1/1 [==============================] - 0s 165ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 169ms/step
1/1 [==============================] - 0s 173ms/step
1/1 [==============================] - 0s 171ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 161ms/step
1/1 [==============================] - 0s 171ms/step
1/1 [==============================] - 0s 20ms/step
27/27 [==============================] - 9s 327ms/step
0.10581112 0.98550725 1.0

 Epoch 11/50
1/1 [==============================] - 0s 288ms/step
1/1 [==============================] - 0s 174ms/step
1/1 [==============================] - 0s 167ms/step
1/1 [==============================] - 0s 159ms/step
1/1 [==============================] - 0s 169ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 165ms/step
1/1 [==============================] - 0s 169ms/step
1/1 [==============================] - 0s 167ms/step
1/1 [==============================] - 0s 172ms/step
1/1 [==============================] - 0s 165ms/step
1/1 [==============================] - 0s 158ms/step
1/1 [==============================] - 0s 170ms/step
1/1 [==============================] - 0s 164ms/step
```

```
1/1 [==============================] - 0s 148ms/step
1/1 [==============================] - 0s 143ms/step
1/1 [==============================] - 0s 150ms/step
1/1 [==============================] - 0s 147ms/step
1/1 [==============================] - 0s 149ms/step
1/1 [==============================] - 0s 154ms/step
1/1 [==============================] - 0s 147ms/step
1/1 [==============================] - 0s 152ms/step
1/1 [==============================] - 0s 147ms/step
1/1 [==============================] - 0s 165ms/step
1/1 [==============================] - 0s 158ms/step
1/1 [==============================] - 0s 32ms/step
27/27 [==============================] - 9s 325ms/step
0.00094348565 0.9953488 1.0

 Epoch 12/50
1/1 [==============================] - 0s 153ms/step
1/1 [==============================] - 0s 167ms/step
1/1 [==============================] - 0s 174ms/step
1/1 [==============================] - 0s 169ms/step
1/1 [==============================] - 0s 170ms/step
1/1 [==============================] - 0s 170ms/step
1/1 [==============================] - 0s 156ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 169ms/step
1/1 [==============================] - 0s 168ms/step
1/1 [==============================] - 0s 145ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 172ms/step
1/1 [==============================] - 0s 157ms/step
1/1 [==============================] - 0s 169ms/step
1/1 [==============================] - 0s 161ms/step
1/1 [==============================] - 0s 175ms/step
1/1 [==============================] - 0s 174ms/step
1/1 [==============================] - 0s 172ms/step
1/1 [==============================] - 0s 162ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 169ms/step
1/1 [==============================] - 0s 171ms/step
1/1 [==============================] - 0s 171ms/step
1/1 [==============================] - 0s 31ms/step
27/27 [==============================] - 8s 312ms/step
0.04793215 1.0 1.0

 Epoch 13/50
1/1 [==============================] - 0s 237ms/step
1/1 [==============================] - 0s 151ms/step
```

```
1/1 [==============================] - 0s 148ms/step
1/1 [==============================] - 0s 129ms/step
1/1 [==============================] - 0s 147ms/step
1/1 [==============================] - 0s 167ms/step
1/1 [==============================] - 0s 168ms/step
1/1 [==============================] - 0s 155ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 172ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 169ms/step
1/1 [==============================] - 0s 162ms/step
1/1 [==============================] - 0s 165ms/step
1/1 [==============================] - 0s 169ms/step
1/1 [==============================] - 0s 156ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 173ms/step
1/1 [==============================] - 0s 162ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 153ms/step
1/1 [==============================] - 0s 173ms/step
1/1 [==============================] - 0s 172ms/step
1/1 [==============================] - 0s 160ms/step
1/1 [==============================] - 0s 165ms/step
1/1 [==============================] - 0s 24ms/step
27/27 [==============================] - 8s 312ms/step
0.6528677 0.9009434 0.88018435

 Epoch 14/50
1/1 [==============================] - 0s 171ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 168ms/step
1/1 [==============================] - 0s 153ms/step
1/1 [==============================] - 0s 135ms/step
1/1 [==============================] - 0s 142ms/step
1/1 [==============================] - 0s 143ms/step
1/1 [==============================] - 0s 128ms/step
1/1 [==============================] - 0s 148ms/step
1/1 [==============================] - 0s 143ms/step
1/1 [==============================] - 0s 151ms/step
1/1 [==============================] - 0s 131ms/step
1/1 [==============================] - 0s 149ms/step
1/1 [==============================] - 0s 143ms/step
1/1 [==============================] - 0s 144ms/step
1/1 [==============================] - 0s 167ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 167ms/step
1/1 [==============================] - 0s 157ms/step
1/1 [==============================] - 0s 157ms/step
```

```
1/1 [==============================] - 0s 158ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 161ms/step
1/1 [==============================] - 0s 167ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 169ms/step
1/1 [==============================] - 0s 28ms/step
27/27 [==============================] - 9s 326ms/step
0.29829103 0.7627907 1.0

 Epoch 15/50
1/1 [==============================] - 0s 161ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 171ms/step
1/1 [==============================] - 0s 160ms/step
1/1 [==============================] - 0s 157ms/step
1/1 [==============================] - 0s 167ms/step
1/1 [==============================] - 0s 162ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 162ms/step
1/1 [==============================] - 0s 159ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 170ms/step
1/1 [==============================] - 0s 156ms/step
1/1 [==============================] - 0s 159ms/step
1/1 [==============================] - 0s 150ms/step
1/1 [==============================] - 0s 145ms/step
1/1 [==============================] - 0s 150ms/step
1/1 [==============================] - 0s 155ms/step
1/1 [==============================] - 0s 144ms/step
1/1 [==============================] - 0s 157ms/step
1/1 [==============================] - 0s 144ms/step
1/1 [==============================] - 0s 144ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 169ms/step
1/1 [==============================] - 0s 167ms/step
1/1 [==============================] - 0s 26ms/step
27/27 [==============================] - 9s 339ms/step
0.18422893 0.98058254 1.0

 Epoch 16/50
1/1 [==============================] - 0s 152ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 162ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 169ms/step
1/1 [==============================] - 0s 167ms/step
1/1 [==============================] - 0s 169ms/step
```

```
1/1 [==============================] - 0s 160ms/step
1/1 [==============================] - 0s 172ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 165ms/step
1/1 [==============================] - 0s 173ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 165ms/step
1/1 [==============================] - 0s 172ms/step
1/1 [==============================] - 0s 154ms/step
1/1 [==============================] - 0s 171ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 171ms/step
1/1 [==============================] - 0s 168ms/step
1/1 [==============================] - 0s 152ms/step
1/1 [==============================] - 0s 154ms/step
1/1 [==============================] - 0s 142ms/step
1/1 [==============================] - 0s 29ms/step
27/27 [==============================] - 9s 321ms/step
0.13702844 0.977169 1.0

 Epoch 17/50
1/1 [==============================] - 0s 134ms/step
1/1 [==============================] - 0s 152ms/step
1/1 [==============================] - 0s 158ms/step
1/1 [==============================] - 0s 167ms/step
1/1 [==============================] - 0s 171ms/step
1/1 [==============================] - 0s 162ms/step
1/1 [==============================] - 0s 169ms/step
1/1 [==============================] - 0s 170ms/step
1/1 [==============================] - 0s 158ms/step
1/1 [==============================] - 0s 169ms/step
1/1 [==============================] - 0s 169ms/step
1/1 [==============================] - 0s 170ms/step
1/1 [==============================] - 0s 172ms/step
1/1 [==============================] - 0s 170ms/step
1/1 [==============================] - 0s 172ms/step
1/1 [==============================] - 0s 174ms/step
1/1 [==============================] - 0s 167ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 167ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 165ms/step
1/1 [==============================] - 0s 165ms/step
1/1 [==============================] - 0s 160ms/step
1/1 [==============================] - 0s 161ms/step
```

```
1/1 [==============================] - 0s 19ms/step
27/27 [==============================] - 8s 315ms/step
0.028407825 0.98564595 0.99038464

 Epoch 18/50
1/1 [==============================] - 0s 161ms/step
1/1 [==============================] - 0s 170ms/step
1/1 [==============================] - 0s 168ms/step
1/1 [==============================] - 0s 165ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 168ms/step
1/1 [==============================] - 0s 141ms/step
1/1 [==============================] - 0s 150ms/step
1/1 [==============================] - 0s 133ms/step
1/1 [==============================] - 0s 153ms/step
1/1 [==============================] - 0s 147ms/step
1/1 [==============================] - 0s 141ms/step
1/1 [==============================] - 0s 144ms/step
1/1 [==============================] - 0s 156ms/step
1/1 [==============================] - 0s 159ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 160ms/step
1/1 [==============================] - 0s 169ms/step
1/1 [==============================] - 0s 161ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 161ms/step
1/1 [==============================] - 0s 168ms/step
1/1 [==============================] - 0s 168ms/step
1/1 [==============================] - 0s 165ms/step
1/1 [==============================] - 0s 165ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 25ms/step
27/27 [==============================] - 9s 340ms/step
0.00025042018 0.99492383 1.0

 Epoch 19/50
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 165ms/step
1/1 [==============================] - 0s 167ms/step
1/1 [==============================] - 0s 169ms/step
1/1 [==============================] - 0s 165ms/step
1/1 [==============================] - 0s 167ms/step
1/1 [==============================] - 0s 169ms/step
1/1 [==============================] - 0s 173ms/step
1/1 [==============================] - 0s 155ms/step
1/1 [==============================] - 0s 158ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 160ms/step
1/1 [==============================] - 0s 161ms/step
```

```
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 161ms/step
1/1 [==============================] - 0s 161ms/step
1/1 [==============================] - 0s 168ms/step
1/1 [==============================] - 0s 137ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 152ms/step
1/1 [==============================] - 0s 138ms/step
1/1 [==============================] - 0s 150ms/step
1/1 [==============================] - 0s 152ms/step
1/1 [==============================] - 0s 145ms/step
1/1 [==============================] - 0s 148ms/step
1/1 [==============================] - 0s 23ms/step
27/27 [==============================] - 9s 338ms/step
0.5918157 0.88785046 0.91346157

 Epoch 20/50
1/1 [==============================] - 0s 162ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 168ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 169ms/step
1/1 [==============================] - 0s 159ms/step
1/1 [==============================] - 0s 161ms/step
1/1 [==============================] - 0s 167ms/step
1/1 [==============================] - 0s 172ms/step
1/1 [==============================] - 0s 157ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 155ms/step
1/1 [==============================] - 0s 151ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 160ms/step
1/1 [==============================] - 0s 161ms/step
1/1 [==============================] - 0s 170ms/step
1/1 [==============================] - 0s 162ms/step
1/1 [==============================] - 0s 160ms/step
1/1 [==============================] - 0s 158ms/step
1/1 [==============================] - 0s 152ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 155ms/step
1/1 [==============================] - 0s 171ms/step
1/1 [==============================] - 0s 160ms/step
1/1 [==============================] - 0s 30ms/step
27/27 [==============================] - 9s 325ms/step
0.19028953 0.9052133 1.0

 Epoch 21/50
```

```
1/1 [==============================] - 0s 268ms/step
1/1 [==============================] - 0s 157ms/step
1/1 [==============================] - 0s 157ms/step
1/1 [==============================] - 0s 154ms/step
1/1 [==============================] - 0s 160ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 162ms/step
1/1 [==============================] - 0s 161ms/step
1/1 [==============================] - 0s 157ms/step
1/1 [==============================] - 0s 165ms/step
1/1 [==============================] - 0s 155ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 165ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 158ms/step
1/1 [==============================] - 0s 157ms/step
1/1 [==============================] - 0s 171ms/step
1/1 [==============================] - 0s 161ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 160ms/step
1/1 [==============================] - 0s 165ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 160ms/step
1/1 [==============================] - 0s 161ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 28ms/step
27/27 [==============================] - 8s 307ms/step
0.42177942 0.9909091 1.0

 Epoch 22/50
1/1 [==============================] - 0s 152ms/step
1/1 [==============================] - 0s 162ms/step
1/1 [==============================] - 0s 169ms/step
1/1 [==============================] - 0s 137ms/step
1/1 [==============================] - 0s 144ms/step
1/1 [==============================] - 0s 144ms/step
1/1 [==============================] - 0s 151ms/step
1/1 [==============================] - 0s 141ms/step
1/1 [==============================] - 0s 144ms/step
1/1 [==============================] - 0s 122ms/step
1/1 [==============================] - 0s 151ms/step
1/1 [==============================] - 0s 147ms/step
1/1 [==============================] - 0s 156ms/step
1/1 [==============================] - 0s 172ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 172ms/step
1/1 [==============================] - 0s 168ms/step
1/1 [==============================] - 0s 165ms/step
```

```
1/1 [==============================] - 0s 160ms/step
1/1 [==============================] - 0s 165ms/step
1/1 [==============================] - 0s 169ms/step
1/1 [==============================] - 0s 170ms/step
1/1 [==============================] - 0s 169ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 175ms/step
1/1 [==============================] - 0s 161ms/step
1/1 [==============================] - 0s 22ms/step
27/27 [==============================] - 9s 324ms/step
0.23842233 0.98578197 1.0

 Epoch 23/50
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 165ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 168ms/step
1/1 [==============================] - 0s 173ms/step
1/1 [==============================] - 0s 167ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 158ms/step
1/1 [==============================] - 0s 171ms/step
1/1 [==============================] - 0s 169ms/step
1/1 [==============================] - 0s 172ms/step
1/1 [==============================] - 0s 161ms/step
1/1 [==============================] - 0s 171ms/step
1/1 [==============================] - 0s 151ms/step
1/1 [==============================] - 0s 143ms/step
1/1 [==============================] - 0s 148ms/step
1/1 [==============================] - 0s 141ms/step
1/1 [==============================] - 0s 143ms/step
1/1 [==============================] - 0s 149ms/step
1/1 [==============================] - 0s 143ms/step
1/1 [==============================] - 0s 139ms/step
1/1 [==============================] - 0s 160ms/step
1/1 [==============================] - 0s 168ms/step
1/1 [==============================] - 0s 167ms/step
1/1 [==============================] - 0s 161ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 25ms/step
27/27 [==============================] - 9s 344ms/step
0.28331545 1.0 1.0

 Epoch 24/50
1/1 [==============================] - 0s 160ms/step
1/1 [==============================] - 0s 168ms/step
1/1 [==============================] - 0s 162ms/step
1/1 [==============================] - 0s 171ms/step
1/1 [==============================] - 0s 163ms/step
```

```
1/1 [==============================] - 0s 157ms/step
1/1 [==============================] - 0s 165ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 162ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 156ms/step
1/1 [==============================] - 0s 172ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 165ms/step
1/1 [==============================] - 0s 165ms/step
1/1 [==============================] - 0s 167ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 176ms/step
1/1 [==============================] - 0s 169ms/step
1/1 [==============================] - 0s 149ms/step
1/1 [==============================] - 0s 145ms/step
1/1 [==============================] - 0s 147ms/step
1/1 [==============================] - 0s 149ms/step
1/1 [==============================] - 0s 39ms/step
27/27 [==============================] - 9s 317ms/step
0.046745256 1.0 1.0

 Epoch 25/50
1/1 [==============================] - 0s 277ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 165ms/step
1/1 [==============================] - 0s 160ms/step
1/1 [==============================] - 0s 156ms/step
1/1 [==============================] - 0s 167ms/step
1/1 [==============================] - 0s 174ms/step
1/1 [==============================] - 0s 169ms/step
1/1 [==============================] - 0s 155ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 174ms/step
1/1 [==============================] - 0s 157ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 161ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 170ms/step
1/1 [==============================] - 0s 167ms/step
1/1 [==============================] - 0s 170ms/step
1/1 [==============================] - 0s 168ms/step
1/1 [==============================] - 0s 157ms/step
1/1 [==============================] - 0s 165ms/step
```

```
1/1 [==============================] - 0s 172ms/step
1/1 [==============================] - 0s 167ms/step
1/1 [==============================] - 0s 171ms/step
1/1 [==============================] - 0s 27ms/step
27/27 [==============================] - 9s 323ms/step
0.032999475 1.0 1.0

 Epoch 26/50
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 165ms/step
1/1 [==============================] - 0s 140ms/step
1/1 [==============================] - 0s 141ms/step
1/1 [==============================] - 0s 154ms/step
1/1 [==============================] - 0s 142ms/step
1/1 [==============================] - 0s 148ms/step
1/1 [==============================] - 0s 142ms/step
1/1 [==============================] - 0s 148ms/step
1/1 [==============================] - 0s 149ms/step
1/1 [==============================] - 0s 156ms/step
1/1 [==============================] - 0s 158ms/step
1/1 [==============================] - 0s 162ms/step
1/1 [==============================] - 0s 170ms/step
1/1 [==============================] - 0s 167ms/step
1/1 [==============================] - 0s 154ms/step
1/1 [==============================] - 0s 160ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 170ms/step
1/1 [==============================] - 0s 170ms/step
1/1 [==============================] - 0s 155ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 172ms/step
1/1 [==============================] - 0s 161ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 171ms/step
1/1 [==============================] - 0s 21ms/step
27/27 [==============================] - 9s 333ms/step
0.033280425 1.0 1.0

 Epoch 27/50
1/1 [==============================] - 0s 161ms/step
1/1 [==============================] - 0s 160ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 158ms/step
1/1 [==============================] - 0s 161ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 168ms/step
1/1 [==============================] - 0s 165ms/step
1/1 [==============================] - 0s 167ms/step
```

```
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 165ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 151ms/step
1/1 [==============================] - 0s 157ms/step
1/1 [==============================] - 0s 151ms/step
1/1 [==============================] - 0s 137ms/step
1/1 [==============================] - 0s 148ms/step
1/1 [==============================] - 0s 151ms/step
1/1 [==============================] - 0s 146ms/step
1/1 [==============================] - 0s 153ms/step
1/1 [==============================] - 0s 154ms/step
1/1 [==============================] - 0s 155ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 22ms/step
27/27 [==============================] - 9s 332ms/step
0.006928601 1.0 1.0

 Epoch 28/50
1/1 [==============================] - 0s 158ms/step
1/1 [==============================] - 0s 172ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 161ms/step
1/1 [==============================] - 0s 168ms/step
1/1 [==============================] - 0s 171ms/step
1/1 [==============================] - 0s 169ms/step
1/1 [==============================] - 0s 173ms/step
1/1 [==============================] - 0s 161ms/step
1/1 [==============================] - 0s 162ms/step
1/1 [==============================] - 0s 165ms/step
1/1 [==============================] - 0s 167ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 169ms/step
1/1 [==============================] - 0s 165ms/step
1/1 [==============================] - 0s 165ms/step
1/1 [==============================] - 0s 169ms/step
1/1 [==============================] - 0s 169ms/step
1/1 [==============================] - 0s 157ms/step
1/1 [==============================] - 0s 159ms/step
1/1 [==============================] - 0s 168ms/step
1/1 [==============================] - 0s 174ms/step
1/1 [==============================] - 0s 169ms/step
1/1 [==============================] - 0s 170ms/step
1/1 [==============================] - 0s 153ms/step
1/1 [==============================] - 0s 170ms/step
1/1 [==============================] - 0s 31ms/step
27/27 [==============================] - 8s 311ms/step
```

```
0.04247409 1.0 1.0

 Epoch 29/50
1/1 [==============================] - 0s 263ms/step
1/1 [==============================] - 0s 148ms/step
1/1 [==============================] - 0s 150ms/step
1/1 [==============================] - 0s 149ms/step
1/1 [==============================] - 0s 134ms/step
1/1 [==============================] - 0s 155ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 158ms/step
1/1 [==============================] - 0s 167ms/step
1/1 [==============================] - 0s 156ms/step
1/1 [==============================] - 0s 168ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 156ms/step
1/1 [==============================] - 0s 167ms/step
1/1 [==============================] - 0s 162ms/step
1/1 [==============================] - 0s 158ms/step
1/1 [==============================] - 0s 154ms/step
1/1 [==============================] - 0s 159ms/step
1/1 [==============================] - 0s 160ms/step
1/1 [==============================] - 0s 168ms/step
1/1 [==============================] - 0s 156ms/step
1/1 [==============================] - 0s 155ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 175ms/step
1/1 [==============================] - 0s 169ms/step
1/1 [==============================] - 0s 20ms/step
27/27 [==============================] - 9s 326ms/step
0.04763159 1.0 1.0

 Epoch 30/50
1/1 [==============================] - 0s 155ms/step
1/1 [==============================] - 0s 167ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 157ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 167ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 152ms/step
1/1 [==============================] - 0s 158ms/step
1/1 [==============================] - 0s 154ms/step
1/1 [==============================] - 0s 139ms/step
1/1 [==============================] - 0s 152ms/step
1/1 [==============================] - 0s 154ms/step
1/1 [==============================] - 0s 147ms/step
1/1 [==============================] - 0s 153ms/step
```

```
1/1 [==============================] - 0s 136ms/step
1/1 [==============================] - 0s 144ms/step
1/1 [==============================] - 0s 125ms/step
1/1 [==============================] - 0s 159ms/step
1/1 [==============================] - 0s 153ms/step
1/1 [==============================] - 0s 149ms/step
1/1 [==============================] - 0s 167ms/step
1/1 [==============================] - 0s 170ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 161ms/step
1/1 [==============================] - 0s 157ms/step
1/1 [==============================] - 0s 23ms/step
27/27 [==============================] - 9s 325ms/step
0.05214904 1.0 1.0

 Epoch 31/50
1/1 [==============================] - 0s 289ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 171ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 158ms/step
1/1 [==============================] - 0s 161ms/step
1/1 [==============================] - 0s 168ms/step
1/1 [==============================] - 0s 167ms/step
1/1 [==============================] - 0s 176ms/step
1/1 [==============================] - 0s 170ms/step
1/1 [==============================] - 0s 169ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 161ms/step
1/1 [==============================] - 0s 160ms/step
1/1 [==============================] - 0s 165ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 169ms/step
1/1 [==============================] - 0s 143ms/step
1/1 [==============================] - 0s 140ms/step
1/1 [==============================] - 0s 142ms/step
1/1 [==============================] - 0s 152ms/step
1/1 [==============================] - 0s 150ms/step
1/1 [==============================] - 0s 136ms/step
1/1 [==============================] - 0s 140ms/step
1/1 [==============================] - 0s 144ms/step
1/1 [==============================] - 0s 33ms/step
27/27 [==============================] - 9s 341ms/step
0.062358145 1.0 1.0

 Epoch 32/50
1/1 [==============================] - 0s 120ms/step
1/1 [==============================] - 0s 159ms/step
```

```
1/1 [==============================] - 0s 160ms/step
1/1 [==============================] - 0s 169ms/step
1/1 [==============================] - 0s 153ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 161ms/step
1/1 [==============================] - 0s 159ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 169ms/step
1/1 [==============================] - 0s 156ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 167ms/step
1/1 [==============================] - 0s 168ms/step
1/1 [==============================] - 0s 171ms/step
1/1 [==============================] - 0s 169ms/step
1/1 [==============================] - 0s 173ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 168ms/step
1/1 [==============================] - 0s 159ms/step
1/1 [==============================] - 0s 162ms/step
1/1 [==============================] - 0s 168ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 21ms/step
27/27 [==============================] - 9s 344ms/step
0.008984742 1.0 1.0

 Epoch 33/50
1/1 [==============================] - 0s 162ms/step
1/1 [==============================] - 0s 147ms/step
1/1 [==============================] - 0s 145ms/step
1/1 [==============================] - 0s 148ms/step
1/1 [==============================] - 0s 129ms/step
1/1 [==============================] - 0s 140ms/step
1/1 [==============================] - 0s 149ms/step
1/1 [==============================] - 0s 142ms/step
1/1 [==============================] - 0s 147ms/step
1/1 [==============================] - 0s 137ms/step
1/1 [==============================] - 0s 134ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 165ms/step
1/1 [==============================] - 0s 159ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 172ms/step
1/1 [==============================] - 0s 156ms/step
1/1 [==============================] - 0s 165ms/step
1/1 [==============================] - 0s 148ms/step
1/1 [==============================] - 0s 157ms/step
```

```
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 173ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 169ms/step
1/1 [==============================] - 0s 173ms/step
1/1 [==============================] - 0s 162ms/step
1/1 [==============================] - 0s 23ms/step
27/27 [==============================] - 9s 328ms/step
0.05935184 1.0 1.0

 Epoch 34/50
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 156ms/step
1/1 [==============================] - 0s 162ms/step
1/1 [==============================] - 0s 168ms/step
1/1 [==============================] - 0s 151ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 147ms/step
1/1 [==============================] - 0s 174ms/step
1/1 [==============================] - 0s 168ms/step
1/1 [==============================] - 0s 171ms/step
1/1 [==============================] - 0s 160ms/step
1/1 [==============================] - 0s 158ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 158ms/step
1/1 [==============================] - 0s 160ms/step
1/1 [==============================] - 0s 144ms/step
1/1 [==============================] - 0s 150ms/step
1/1 [==============================] - 0s 149ms/step
1/1 [==============================] - 0s 143ms/step
1/1 [==============================] - 0s 149ms/step
1/1 [==============================] - 0s 141ms/step
1/1 [==============================] - 0s 139ms/step
1/1 [==============================] - 0s 149ms/step
1/1 [==============================] - 0s 153ms/step
1/1 [==============================] - 0s 162ms/step
1/1 [==============================] - 0s 165ms/step
1/1 [==============================] - 0s 20ms/step
27/27 [==============================] - 9s 337ms/step
0.018520145 1.0 1.0

 Epoch 35/50
1/1 [==============================] - 0s 160ms/step
1/1 [==============================] - 0s 161ms/step
1/1 [==============================] - 0s 165ms/step
1/1 [==============================] - 0s 157ms/step
1/1 [==============================] - 0s 159ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 169ms/step
1/1 [==============================] - 0s 166ms/step
```

```
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 160ms/step
1/1 [==============================] - 0s 160ms/step
1/1 [==============================] - 0s 160ms/step
1/1 [==============================] - 0s 160ms/step
1/1 [==============================] - 0s 171ms/step
1/1 [==============================] - 0s 165ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 170ms/step
1/1 [==============================] - 0s 168ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 167ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 150ms/step
1/1 [==============================] - 0s 151ms/step
1/1 [==============================] - 0s 31ms/step
27/27 [==============================] - 9s 320ms/step
0.07026332 1.0 1.0

 Epoch 36/50
1/1 [==============================] - 0s 139ms/step
1/1 [==============================] - 0s 150ms/step
1/1 [==============================] - 0s 141ms/step
1/1 [==============================] - 0s 154ms/step
1/1 [==============================] - 0s 147ms/step
1/1 [==============================] - 0s 150ms/step
1/1 [==============================] - 0s 153ms/step
1/1 [==============================] - 0s 140ms/step
1/1 [==============================] - 0s 159ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 167ms/step
1/1 [==============================] - 0s 161ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 149ms/step
1/1 [==============================] - 0s 162ms/step
1/1 [==============================] - 0s 165ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 155ms/step
1/1 [==============================] - 0s 161ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 161ms/step
1/1 [==============================] - 0s 152ms/step
1/1 [==============================] - 0s 157ms/step
1/1 [==============================] - 0s 168ms/step
1/1 [==============================] - 0s 156ms/step
```

```
1/1 [==============================] - 0s 31ms/step
27/27 [==============================] - 9s 337ms/step
0.0016693957 1.0 1.0

 Epoch 37/50
1/1 [==============================] - 0s 153ms/step
1/1 [==============================] - 0s 153ms/step
1/1 [==============================] - 0s 156ms/step
1/1 [==============================] - 0s 152ms/step
1/1 [==============================] - 0s 159ms/step
1/1 [==============================] - 0s 167ms/step
1/1 [==============================] - 0s 149ms/step
1/1 [==============================] - 0s 128ms/step
1/1 [==============================] - 0s 142ms/step
1/1 [==============================] - 0s 139ms/step
1/1 [==============================] - 0s 150ms/step
1/1 [==============================] - 0s 156ms/step
1/1 [==============================] - 0s 141ms/step
1/1 [==============================] - 0s 141ms/step
1/1 [==============================] - 0s 148ms/step
1/1 [==============================] - 0s 124ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 173ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 169ms/step
1/1 [==============================] - 0s 169ms/step
1/1 [==============================] - 0s 167ms/step
1/1 [==============================] - 0s 161ms/step
1/1 [==============================] - 0s 152ms/step
1/1 [==============================] - 0s 160ms/step
1/1 [==============================] - 0s 26ms/step
27/27 [==============================] - 9s 342ms/step
0.014636921 1.0 1.0

 Epoch 38/50
1/1 [==============================] - 0s 157ms/step
1/1 [==============================] - 0s 158ms/step
1/1 [==============================] - 0s 161ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 156ms/step
1/1 [==============================] - 0s 146ms/step
1/1 [==============================] - 0s 137ms/step
1/1 [==============================] - 0s 162ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 169ms/step
1/1 [==============================] - 0s 159ms/step
1/1 [==============================] - 0s 160ms/step
1/1 [==============================] - 0s 156ms/step
```

```
1/1 [==============================] - 0s 160ms/step
1/1 [==============================] - 0s 161ms/step
1/1 [==============================] - 0s 156ms/step
1/1 [==============================] - 0s 162ms/step
1/1 [==============================] - 0s 149ms/step
1/1 [==============================] - 0s 146ms/step
1/1 [==============================] - 0s 153ms/step
1/1 [==============================] - 0s 154ms/step
1/1 [==============================] - 0s 139ms/step
1/1 [==============================] - 0s 131ms/step
1/1 [==============================] - 0s 152ms/step
1/1 [==============================] - 0s 146ms/step
1/1 [==============================] - 0s 142ms/step
1/1 [==============================] - 0s 38ms/step
27/27 [==============================] - 9s 326ms/step
0.0026071959 1.0 1.0

 Epoch 39/50
1/1 [==============================] - 0s 169ms/step
1/1 [==============================] - 0s 152ms/step
1/1 [==============================] - 0s 157ms/step
1/1 [==============================] - 0s 152ms/step
1/1 [==============================] - 0s 167ms/step
1/1 [==============================] - 0s 159ms/step
1/1 [==============================] - 0s 169ms/step
1/1 [==============================] - 0s 170ms/step
1/1 [==============================] - 0s 156ms/step
1/1 [==============================] - 0s 167ms/step
1/1 [==============================] - 0s 172ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 160ms/step
1/1 [==============================] - 0s 161ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 167ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 161ms/step
1/1 [==============================] - 0s 168ms/step
1/1 [==============================] - 0s 162ms/step
1/1 [==============================] - 0s 157ms/step
1/1 [==============================] - 0s 170ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 162ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 22ms/step
27/27 [==============================] - 8s 306ms/step
0.013342813 1.0 1.0

 Epoch 40/50
```

```
1/1 [==============================] - 0s 270ms/step
1/1 [==============================] - 0s 150ms/step
1/1 [==============================] - 0s 136ms/step
1/1 [==============================] - 0s 144ms/step
1/1 [==============================] - 0s 133ms/step
1/1 [==============================] - 0s 169ms/step
1/1 [==============================] - 0s 167ms/step
1/1 [==============================] - 0s 159ms/step
1/1 [==============================] - 0s 165ms/step
1/1 [==============================] - 0s 154ms/step
1/1 [==============================] - 0s 170ms/step
1/1 [==============================] - 0s 151ms/step
1/1 [==============================] - 0s 159ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 169ms/step
1/1 [==============================] - 0s 169ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 150ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 158ms/step
1/1 [==============================] - 0s 170ms/step
1/1 [==============================] - 0s 26ms/step
27/27 [==============================] - 8s 295ms/step
0.0015708163 1.0 1.0

 Epoch 41/50
1/1 [==============================] - 0s 251ms/step
1/1 [==============================] - 0s 140ms/step
1/1 [==============================] - 0s 145ms/step
1/1 [==============================] - 0s 136ms/step
1/1 [==============================] - 0s 139ms/step
1/1 [==============================] - 0s 139ms/step
1/1 [==============================] - 0s 140ms/step
1/1 [==============================] - 0s 167ms/step
1/1 [==============================] - 0s 170ms/step
1/1 [==============================] - 0s 157ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 159ms/step
1/1 [==============================] - 0s 159ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 157ms/step
1/1 [==============================] - 0s 161ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 167ms/step
```

```
1/1 [==============================] - 0s 161ms/step
1/1 [==============================] - 0s 169ms/step
1/1 [==============================] - 0s 170ms/step
1/1 [==============================] - 0s 170ms/step
1/1 [==============================] - 0s 157ms/step
1/1 [==============================] - 0s 165ms/step
1/1 [==============================] - 0s 165ms/step
1/1 [==============================] - 0s 169ms/step
1/1 [==============================] - 0s 33ms/step
27/27 [==============================] - 9s 313ms/step
0.00354432 1.0 1.0

 Epoch 42/50
1/1 [==============================] - 0s 158ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 167ms/step
1/1 [==============================] - 0s 171ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 173ms/step
1/1 [==============================] - 0s 177ms/step
1/1 [==============================] - 0s 168ms/step
1/1 [==============================] - 0s 152ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 143ms/step
1/1 [==============================] - 0s 133ms/step
1/1 [==============================] - 0s 151ms/step
1/1 [==============================] - 0s 140ms/step
1/1 [==============================] - 0s 146ms/step
1/1 [==============================] - 0s 151ms/step
1/1 [==============================] - 0s 133ms/step
1/1 [==============================] - 0s 147ms/step
1/1 [==============================] - 0s 143ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 165ms/step
1/1 [==============================] - 0s 162ms/step
1/1 [==============================] - 0s 165ms/step
1/1 [==============================] - 0s 172ms/step
1/1 [==============================] - 0s 156ms/step
1/1 [==============================] - 0s 31ms/step
27/27 [==============================] - 9s 349ms/step
0.012608971 1.0 1.0

 Epoch 43/50
1/1 [==============================] - 0s 169ms/step
1/1 [==============================] - 0s 168ms/step
1/1 [==============================] - 0s 170ms/step
1/1 [==============================] - 0s 165ms/step
1/1 [==============================] - 0s 169ms/step
```

```
1/1 [==============================] - 0s 171ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 160ms/step
1/1 [==============================] - 0s 168ms/step
1/1 [==============================] - 0s 160ms/step
1/1 [==============================] - 0s 173ms/step
1/1 [==============================] - 0s 168ms/step
1/1 [==============================] - 0s 159ms/step
1/1 [==============================] - 0s 160ms/step
1/1 [==============================] - 0s 165ms/step
1/1 [==============================] - 0s 147ms/step
1/1 [==============================] - 0s 157ms/step
1/1 [==============================] - 0s 165ms/step
1/1 [==============================] - 0s 170ms/step
1/1 [==============================] - 0s 167ms/step
1/1 [==============================] - 0s 167ms/step
1/1 [==============================] - 0s 144ms/step
1/1 [==============================] - 0s 140ms/step
1/1 [==============================] - 0s 151ms/step
1/1 [==============================] - 0s 128ms/step
1/1 [==============================] - 0s 146ms/step
1/1 [==============================] - 0s 30ms/step
27/27 [==============================] - 9s 328ms/step
0.011959151 1.0 1.0

 Epoch 44/50
1/1 [==============================] - 0s 131ms/step
1/1 [==============================] - 0s 149ms/step
1/1 [==============================] - 0s 143ms/step
1/1 [==============================] - 0s 137ms/step
1/1 [==============================] - 0s 171ms/step
1/1 [==============================] - 0s 171ms/step
1/1 [==============================] - 0s 162ms/step
1/1 [==============================] - 0s 165ms/step
1/1 [==============================] - 0s 165ms/step
1/1 [==============================] - 0s 160ms/step
1/1 [==============================] - 0s 161ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 169ms/step
1/1 [==============================] - 0s 157ms/step
1/1 [==============================] - 0s 162ms/step
1/1 [==============================] - 0s 153ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 168ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 170ms/step
1/1 [==============================] - 0s 171ms/step
```

```
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 171ms/step
1/1 [==============================] - 0s 168ms/step
1/1 [==============================] - 0s 28ms/step
27/27 [==============================] - 8s 310ms/step
0.005306199 1.0 1.0

 Epoch 45/50
1/1 [==============================] - 0s 287ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 168ms/step
1/1 [==============================] - 0s 140ms/step
1/1 [==============================] - 0s 138ms/step
1/1 [==============================] - 0s 126ms/step
1/1 [==============================] - 0s 131ms/step
1/1 [==============================] - 0s 147ms/step
1/1 [==============================] - 0s 143ms/step
1/1 [==============================] - 0s 147ms/step
1/1 [==============================] - 0s 148ms/step
1/1 [==============================] - 0s 139ms/step
1/1 [==============================] - 0s 137ms/step
1/1 [==============================] - 0s 161ms/step
1/1 [==============================] - 0s 158ms/step
1/1 [==============================] - 0s 160ms/step
1/1 [==============================] - 0s 168ms/step
1/1 [==============================] - 0s 158ms/step
1/1 [==============================] - 0s 170ms/step
1/1 [==============================] - 0s 167ms/step
1/1 [==============================] - 0s 167ms/step
1/1 [==============================] - 0s 162ms/step
1/1 [==============================] - 0s 144ms/step
1/1 [==============================] - 0s 158ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 162ms/step
1/1 [==============================] - 0s 23ms/step
27/27 [==============================] - 9s 331ms/step
0.0001280185 1.0 1.0

 Epoch 46/50
1/1 [==============================] - 0s 162ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 159ms/step
1/1 [==============================] - 0s 156ms/step
1/1 [==============================] - 0s 162ms/step
1/1 [==============================] - 0s 165ms/step
1/1 [==============================] - 0s 162ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 159ms/step
1/1 [==============================] - 0s 164ms/step
```

```
1/1 [==============================] - 0s 162ms/step
1/1 [==============================] - 0s 158ms/step
1/1 [==============================] - 0s 158ms/step
1/1 [==============================] - 0s 170ms/step
1/1 [==============================] - 0s 162ms/step
1/1 [==============================] - 0s 165ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 159ms/step
1/1 [==============================] - 0s 142ms/step
1/1 [==============================] - 0s 129ms/step
1/1 [==============================] - 0s 152ms/step
1/1 [==============================] - 0s 142ms/step
1/1 [==============================] - 0s 139ms/step
1/1 [==============================] - 0s 134ms/step
1/1 [==============================] - 0s 146ms/step
1/1 [==============================] - 0s 153ms/step
1/1 [==============================] - 0s 29ms/step
27/27 [==============================] - 8s 315ms/step
0.00024128695 1.0 1.0

 Epoch 47/50
1/1 [==============================] - 0s 129ms/step
1/1 [==============================] - 0s 141ms/step
1/1 [==============================] - 0s 148ms/step
1/1 [==============================] - 0s 157ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 159ms/step
1/1 [==============================] - 0s 157ms/step
1/1 [==============================] - 0s 162ms/step
1/1 [==============================] - 0s 168ms/step
1/1 [==============================] - 0s 147ms/step
1/1 [==============================] - 0s 155ms/step
1/1 [==============================] - 0s 162ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 170ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 157ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 165ms/step
1/1 [==============================] - 0s 172ms/step
1/1 [==============================] - 0s 158ms/step
1/1 [==============================] - 0s 157ms/step
1/1 [==============================] - 0s 170ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 158ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 158ms/step
1/1 [==============================] - 0s 27ms/step
27/27 [==============================] - 9s 317ms/step
```

```
0.0021307676 1.0 1.0

 Epoch 48/50
1/1 [==============================] - 0s 179ms/step
1/1 [==============================] - 0s 146ms/step
1/1 [==============================] - 0s 138ms/step
1/1 [==============================] - 0s 134ms/step
1/1 [==============================] - 0s 143ms/step
1/1 [==============================] - 0s 150ms/step
1/1 [==============================] - 0s 142ms/step
1/1 [==============================] - 0s 145ms/step
1/1 [==============================] - 0s 136ms/step
1/1 [==============================] - 0s 134ms/step
1/1 [==============================] - 0s 143ms/step
1/1 [==============================] - 0s 162ms/step
1/1 [==============================] - 0s 162ms/step
1/1 [==============================] - 0s 168ms/step
1/1 [==============================] - 0s 162ms/step
1/1 [==============================] - 0s 161ms/step
1/1 [==============================] - 0s 168ms/step
1/1 [==============================] - 0s 155ms/step
1/1 [==============================] - 0s 156ms/step
1/1 [==============================] - 0s 161ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 176ms/step
1/1 [==============================] - 0s 160ms/step
1/1 [==============================] - 0s 156ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 19ms/step
27/27 [==============================] - 9s 334ms/step
0.0020965198 1.0 1.0

 Epoch 49/50
1/1 [==============================] - 0s 162ms/step
1/1 [==============================] - 0s 162ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 158ms/step
1/1 [==============================] - 0s 161ms/step
1/1 [==============================] - 0s 162ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 155ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 155ms/step
1/1 [==============================] - 0s 159ms/step
1/1 [==============================] - 0s 169ms/step
1/1 [==============================] - 0s 152ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 139ms/step
```

```
1/1 [==============================] - 0s 149ms/step
1/1 [==============================] - 0s 146ms/step
1/1 [==============================] - 0s 136ms/step
1/1 [==============================] - 0s 135ms/step
1/1 [==============================] - 0s 144ms/step
1/1 [==============================] - 0s 138ms/step
1/1 [==============================] - 0s 151ms/step
1/1 [==============================] - 0s 158ms/step
1/1 [==============================] - 0s 141ms/step
1/1 [==============================] - 0s 146ms/step
1/1 [==============================] - 0s 155ms/step
1/1 [==============================] - 0s 24ms/step
27/27 [==============================] - 9s 342ms/step
0.000118578624 1.0 1.0

 Epoch 50/50
1/1 [==============================] - 0s 160ms/step
1/1 [==============================] - 0s 160ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 157ms/step
1/1 [==============================] - 0s 168ms/step
1/1 [==============================] - 0s 152ms/step
1/1 [==============================] - 0s 161ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 147ms/step
1/1 [==============================] - 0s 145ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 158ms/step
1/1 [==============================] - 0s 161ms/step
1/1 [==============================] - 0s 159ms/step
1/1 [==============================] - 0s 164ms/step
1/1 [==============================] - 0s 163ms/step
1/1 [==============================] - 0s 162ms/step
1/1 [==============================] - 0s 166ms/step
1/1 [==============================] - 0s 161ms/step
1/1 [==============================] - 0s 151ms/step
1/1 [==============================] - 0s 132ms/step
1/1 [==============================] - 0s 144ms/step
1/1 [==============================] - 0s 42ms/step
27/27 [==============================] - 9s 333ms/step
0.0005095464 1.0 1.0
```

6. Evaluate Model

6.1 Import Metrics

```
# Import metric calculations
from tensorflow.keras.metrics import Precision, Recall
```

6.2 Make Predictions

```
# Get a batch of test data
test_input, test_val, y_true = test_data.as_numpy_iterator().next()
```

```
y_hat = siamese_model.predict([test_input, test_val])
```

```
1/1 [==============================] - 0s 31ms/step
```

```
# Post processing the results
[1 if prediction > 0.5 else 0 for prediction in y_hat ]
```

```
[1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1]
```

```
y_true
```

```
array([1., 1., 1., 0., 1., 1., 0., 1., 0., 1., 0., 1., 1., 0., 1.,
1.],
      dtype=float32)
```

6.3 Calculate Metrics

```
# Creating a metric object
m = Recall()

# Calculating the recall value
m.update_state(y_true, y_hat)

# Return Recall Result
m.result().numpy()
```

```
1.0
```

```
# Creating a metric object
m = Precision()

# Calculating the recall value
m.update_state(y_true, y_hat)

# Return Recall Result
m.result().numpy()
```

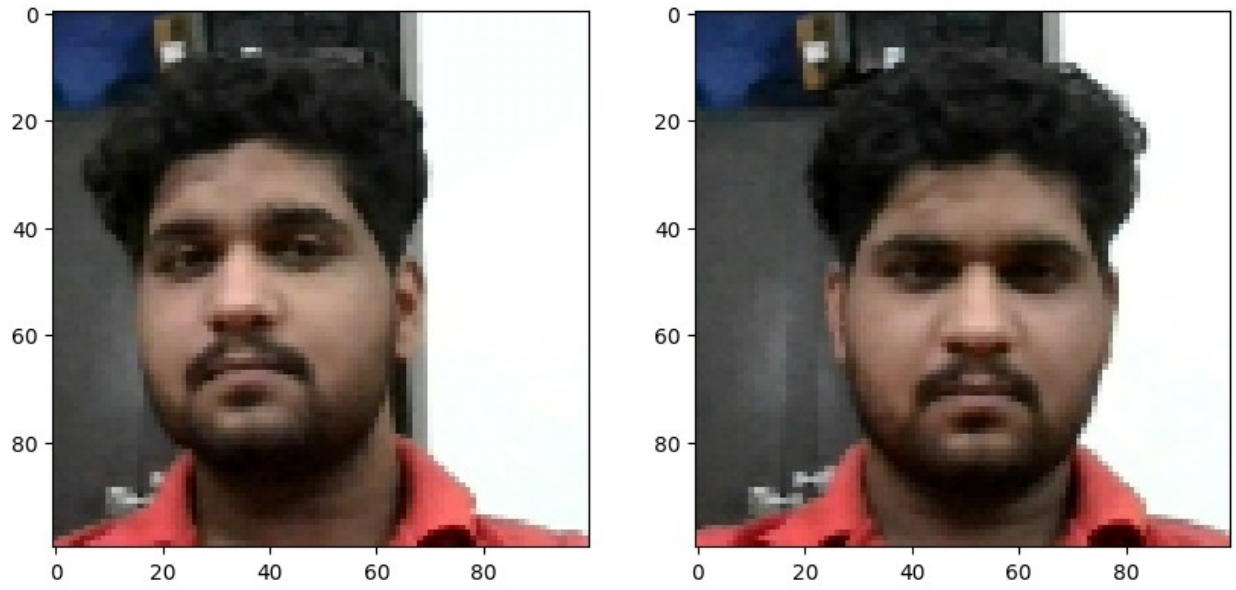
```
1.0
```

```
r = Recall()
p = Precision()

for test_input, test_val, y_true in test_data.as_numpy_iterator():
    yhat = siamese_model.predict([test_input, test_val])
    r.update_state(y_true, yhat)
    p.update_state(y_true,yhat)

print(r.result().numpy(), p.result().numpy())
```

```
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1.0 1.0
```

6.4 Viz Results

```
# Set plot size
plt.figure(figsize=(10,8))

# Set first subplot
plt.subplot(1,2,1)
plt.imshow(test_input[0])

# Set second subplot
plt.subplot(1,2,2)
plt.imshow(test_val[0])

# Renders cleanly
plt.show()
```



7. Save Model

```
# Save weights
# siamese_model.save('siamesemodelv2.h5')
tf.keras.saving.save_model(
    siamese_model, '/content/drive/MyDrive/Colab
Notebooks/Project/SaveModel', overwrite=True, save_format=None)
```

```
WARNING:tensorflow:Compiled the loaded model, but the compiled metrics
have yet to be built. `model.compile_metrics` will be empty until you
train or evaluate the model.
```

```
L1Dist
```

```
__main__.L1Dist
```

```
# Reload model
# siamese_model = tf.keras.models.load_model(siamese_model,
#                                    custom_objects={'L1Dist':L1Dist,
'BinaryCrossentropy':tf.losses.BinaryCrossentropy})
siamese_model = tf.keras.saving.load_model(
    '/content/drive/MyDrive/Colab Notebooks/Project/SaveModel',
custom_objects=None, compile=True, safe_mode=True
)
```

```
WARNING:tensorflow:No training configuration found in save file, so
the model was *not* compiled. Compile it manually.
```

```
# Make predictions with reloaded model
siamese_model.predict([test_input, test_val])
```

```
1/1 [==============================] - 0s 132ms/step

array([[9.9999797e-01],
       [9.2914743e-05],
       [8.9012399e-05],
       [9.9999905e-01]], dtype=float32)

# View model summary
siamese_model.summary()

Model: "SiameseNetwork"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
Connected to
==================================================================
 input_img (InputLayer)      [(None, 100, 100, 3)]     0         []


 validation_img (InputLayer  [(None, 100, 100, 3)]     0         []
 )



 embedding (Functional)      (None, 4096)              3896044
['input_img[0][0]',
                                                       8
'validation_img[0][0]']


 l1_dist_2 (L1Dist)          (None, 4096)              0
['embedding[0][0]',

'embedding[1][0]']


 dense_1 (Dense)             (None, 1)                 4097
['l1_dist_2[0][0]']


==================================================================
Total params: 38964545 (148.64 MB)
Trainable params: 38964545 (148.64 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```