"A principal diferença entre algo que pode dar errado e algo que não pode dar errado é que quando algo que não pode dar errado dá errado, geralmente acaba sendo impossível de ser consertado" (Douglas Adams - O Guia do Mochileiro das Galáxias).

Redes Siamesas

Paulo Ricardo Lisboa de Almeida

DSBD

UFPR

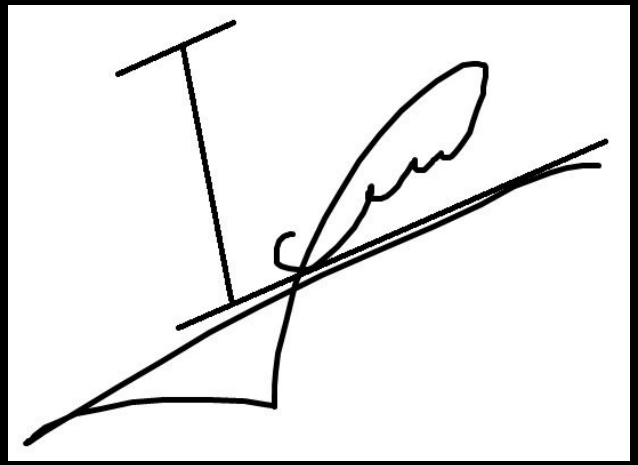
Considere

Considere o problema de identificar se duas imagens pertencem à mesma pessoa.

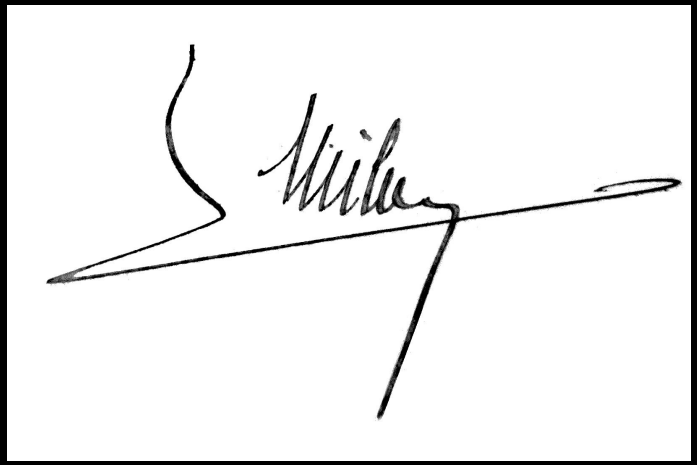
= ???

Considere

Ou ainda, o problema de identificar se duas assinaturas pertencem à mesma pessoa.



= ???



Responda

O que esses problemas têm em comum?

Responda

O que esses problemas têm em comum?

- Estamos comparando objetos.
- Possuem poucas amostras.
  - Exemplo: sua agência bancária ou cartório possui uma ou duas fotos ou assinaturas suas para comparar.
- É inviável treinar um modelo que reconheça todas as assinaturas ou fotos de pessoas possíveis.
  - Mesmo que tivéssemos amostras de todas pessoas do mundo, quando uma nova pessoa nascer, esse modelo se tornaria obsoleto.

Ideia

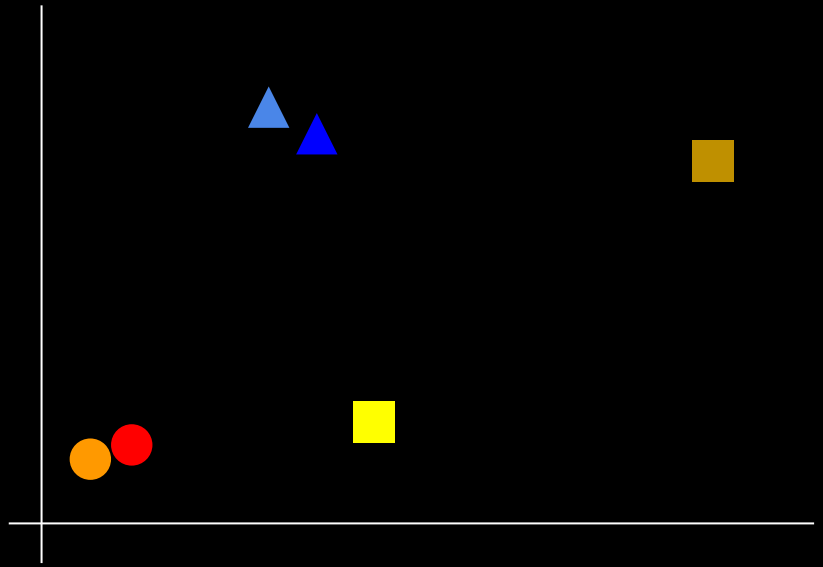
Para esse tipo de problema, a ideia é que objetos que “pertencem à mesma classe” estão próximos no espaço, enquanto objetos que não pertencem à mesma classe, estão distantes.

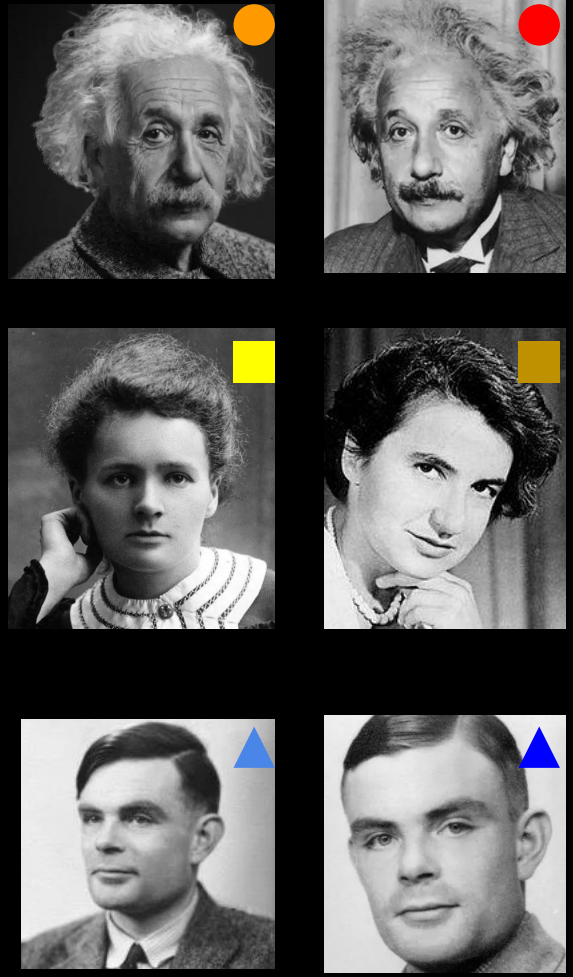
= ???

Dissimilaridade

O conceito de dissimilaridade é projetar os objetos em um espaço em que a distância entre objetos parecidos é pequena, e a distância de objetos diferentes é grande.

Dissimilaridade

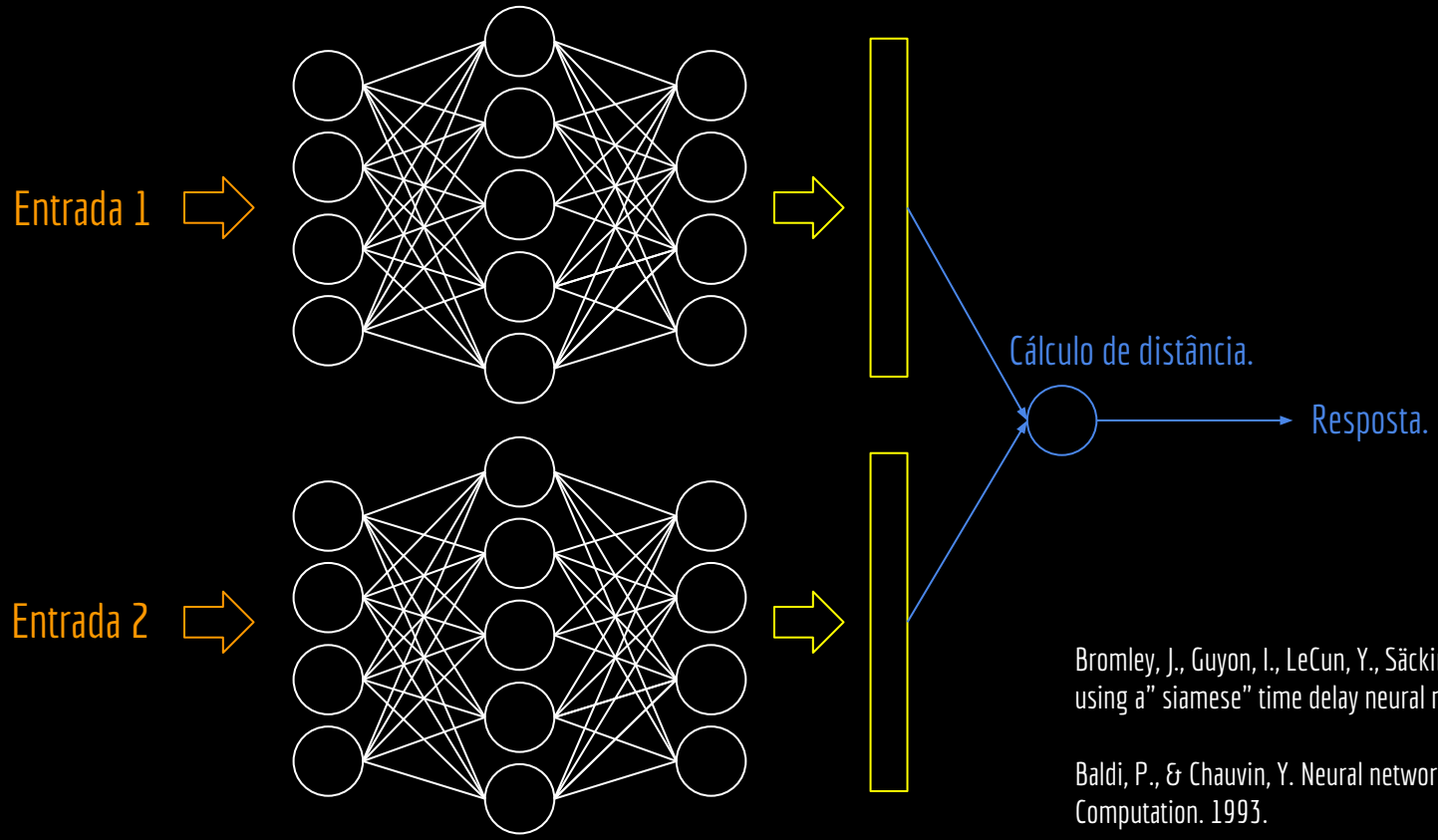




Ideia Geral

Redes de mesma estrutura
para extrair a representação.

Embedding.



Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., & Shah, R. Signature verification using a" siamese" time delay neural network. Neurips. 1993.

Baldi, P., & Chauvin, Y. Neural networks for fingerprint recognition. Neural Computation. 1993.

Na prática

Poderíamos ter “dois braços” idênticos da rede na memória, e computar as imagens em paralelo.

Rápido, mas ocupa o dobro da memória.

Na prática

Poderíamos ter “dois braços” idênticos da rede na memória, e computar as imagens em paralelo.

Rápido, mas ocupa o dobro da memória.

Uma alternativa é manter somente um lado da rede, e mandar as imagens em sequência para ela.

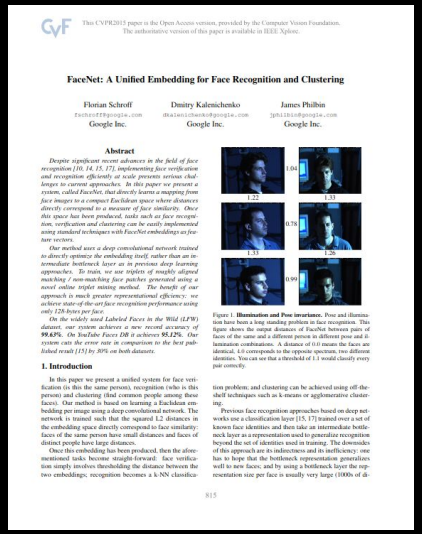
Assume-se que os dois braços da rede possuem a mesma **estrutura e pesos**.

```
def __ativacao_euclidiana(self, x, y):
       return torch.sqrt(F.relu(torch.sum((x - y) ** 2, dim=1, keepdim=True)))

def forward(self, input1, input2):
   #Mantém apenas um “braço da rede” na memória. Manda as duas imagens para ele
   #em sequência. Depois, calcula a função de ativação que combina as respostas.
   tower1 = self.__embedding_network(input1)
   tower2 = self.__embedding_network(input2)
   return self.__ativacao_euclidiana(tower1, tower2)
```

Um exemplo

O exemplo a seguir é uma rede que usa a combinação das estruturas de rede dos seguintes artigos.



FaceNet: A Unified Embedding for Face Recognition and Clustering

Florian Schroff — fschroff@google.com — Google Inc.
Dmitry Kalenichenko — dkalenichenko@google.com — Google Inc.
James Philbin — jphilbin@google.com — Google Inc.

Abstract

Despite significant recent advances in the field of face recognition [10, 14, 15, 17], implementing face verification and recognition efficiently at scale presents serious challenges to current approaches. In this paper we present a system, called FaceNet, that directly learns a mapping from face images to a compact Euclidean space where distances directly correspond to a measure of face similarity. Once this space has been produced, tasks such as face recognition, verification and clustering can be easily implemented using standard techniques with FaceNet embeddings as feature vectors.

Our method uses a deep convolutional network trained to directly optimize the embedding itself, rather than an intermediate bottleneck layer as in previous deep learning approaches. To train, we use triplets of roughly aligned matching / non-matching face patches generated using a novel online triplet mining method. The benefit of our approach is much greater representational efficiency: we achieve state-of-the-art face recognition performance using only 128-bytes per face.

On the widely used Labeled Faces in the Wild (LFW) dataset, our system achieves a new record accuracy of **99.63%**. *On YouTube Faces DB it achieves* **95.12%**. *Our system cuts the error rate in comparison to the best published result [15] by 30% on both datasets.*

1. Introduction

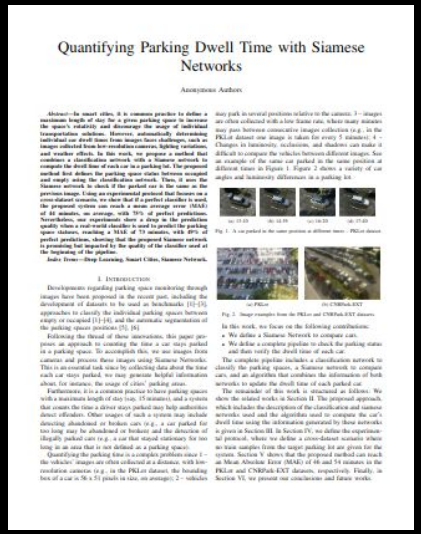
In this paper we present a unified system for face verification (is this the same person), recognition (who is this person) and clustering (find common people among these faces). Our method is based on learning a Euclidean embedding per image using a deep convolutional network. The network is trained such that the squared L2 distances in the embedding space directly correspond to face similarity: faces of the same person have small distances and faces of distinct people have large distances.

Once this embedding has been produced, then the aforementioned tasks become straight-forward: face verification simply involves thresholding the distance between the two embeddings; recognition becomes a k-NN classification problem; and clustering can be achieved using off-the-shelf techniques such as k-means or agglomerative clustering.

Previous face recognition approaches based on deep networks use a classification layer [15, 17] trained over a set of known face identities and then take an intermediate bottleneck layer as a representation used to generalize recognition beyond the set of identities used in training. The downsides of this approach are its indirectness and its inefficiency: one has to hope that the bottleneck representation generalizes well to new faces; and by using a bottleneck layer the representation size per face is usually very large (1000s of di-

Figure 1. **Illumination and Pose invariance.** Pose and illumination have been a long standing problem in face recognition. This figure shows the output distances of FaceNet between pairs of faces of the same and a different person in different pose and illumination combinations. A distance of 0.0 means the faces are identical. 4.0 corresponds to the opposite spectrum, two different identities. You can see that a threshold of 1.1 would classify every pair correctly.

Schroff, F., Kalenichenko, D., & Philbin, J. Facenet: A unified embedding for face recognition and clustering. CVPR. 2015.



Quantifying Parking Dwell Time with Siamese Networks

Anonymous Authors

Abstract—**In smart cities, it is consensus practice to define a maximum length of stay for a given parking space to increase the space's rotativity and discourage the usage of individual transportation schemes. However, automatically determining individual car dwell time from images faces challenges, such as images collected from low-resolution cameras, lighting variations, and weather effects. In this work, we propose a method that combines a classification network with a Siamese network to compute the dwell time of each car in a parking lot. The proposed method first defines the parking space status between occupied and unoccupied. Then, occupied spaces are fed to the Siamese network to check if the parked car is the same as the previous image. Using an experimental protocol that focuses on a cross-dataset scenario, we show that if a perfect classifier is used, the proposed system can reach a mean average error (MAE) of 41 minutes, on average, with 70% of perfect predictions. Nevertheless, our experiments show a drop in the prediction quality when a real-world classifier is used to predict the parking space statuses, reaching a MAE of 73 minutes, with 49% of perfect predictions, showing that the proposed Siamese network is promising but impacted by the quality of the classifier used at the beginning of the pipeline.**

Index Terms—**Deep Learning, Smart Cities, Siamese Network.**

I. Introduction

Developments regarding parking space monitoring through images have been proposed in the recent past, including the development of datasets to be used as benchmarks [1]–[3], approaches to classify the individual parking spaces between empty or occupied [1]–[4], and the automatic segmentation of the parking spaces positions [5], [6].

Following the thread of these innovations, this paper proposes an approach to counting the time a car stays parked at a parking space. To accomplish this, we use images from cameras and process these images using Siamese Networks. This is an essential task since by collecting data about the time each car stays parked, we may generate helpful information about, for instance, the usage of cities' parking areas.

Furthermore, it is a common practice to have parking spaces with a maximum length of stay (say, 15 minutes), and a system that counts the time a driver stays parked may help authorities detect offenders. Other usages of such a system may include detecting abandoned or broken cars (e.g., a car parked for too long may be abandoned or broken) and the direction of illegally parked cars (e.g., a car that stayed stationary for too long in an area that is not defined as a parking space).

Quantifying the parking time in a complex problem since 1– the vehicles' images are often collected at a distance, with low-resolution cameras (e.g., in the PKLot dataset, the bounding box of a car is 56 x 51 pixels in size, on average); 2– vehicles stay park in several positions relative to the camera; 3 – images are often collected with a low frame rate, where many minutes may pass between consecutive images collection (e.g., in the PKLot dataset one image is taken for every 5 minutes); 4 – Changes in luminosity, occlusions, and shadows can make it difficult to compare the vehicles between different images. See an example of the same car parked in the same position at different times in Figure 1. Figure 2 shows a variety of car angles and luminosity differences in a parking lot.

(a): 17-19 (b): 12-19 (c): 16-33 (d): 17-33

Fig. 1. A car parked in the same position at different times – PKLot dataset.

(a) PKLot (b) CNRPark-EXT

Fig. 2. Image examples from the PKLot and CNRPark-EXT datasets.

In this work, we focus on the following contributions:

- We define a Siamese Network to compare cars.
- We define a complete pipeline to check the parking status and then verify the dwell time of each car.

The complete pipeline includes a classification network to classify the parking spaces, a Siamese network to compare cars, and an algorithm that combines the information of both networks to update the dwell time of each parked car.

The remainder of this work is structured as follows: We show the related works in Section II. The proposed approach, which includes the description of the classification and siamese networks used and the algorithm used to compute the car's dwell time using the information generated by those networks is given in Section III. In Section IV, we define the experimental protocol, where we define a cross-dataset scenario where no train samples from the target parking lot are given for the system. Section V shows that the proposed method can reach an Mean Absolute Error (MAE) of 66 and 54 minutes in the PKLot and CNRPark-EXT datasets, respectively. Finally, in Section VI, we present our conclusions and future works.

Ribas, Benedet, Zanlorensi, Oliveira, Almeida. Quantifying Parking Dwell Time with Siamese Networks. ICMLA. 2024.

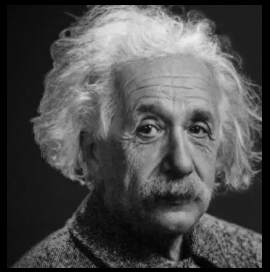
Ideias: Facenet -› Triplet Loss

Triplet Loss.

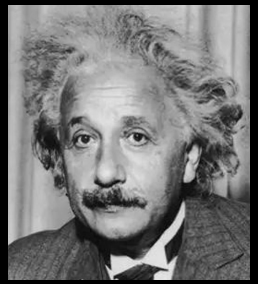
Introduzido no Artigo da Facenet.

Cada amostra de treinamento é uma tripla (triplet) contendo uma âncora, uma imagem positiva, da mesma classe da âncora, e uma imagem negativa, de uma classe diferente.

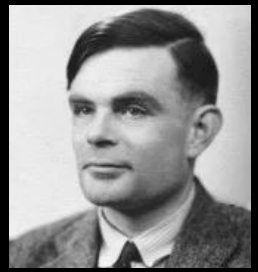
âncora



positivo



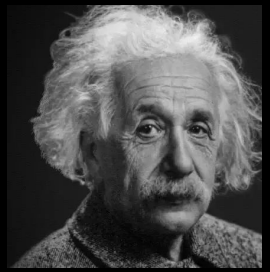
negativo



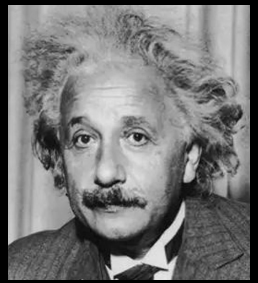
Triplet Loss

O objetivo da rede é projetar as imagens em um espaço as âncora e os exemplos positivos estão próximos, ao passo que as âncoras e os exemplos negativos estão distantes.

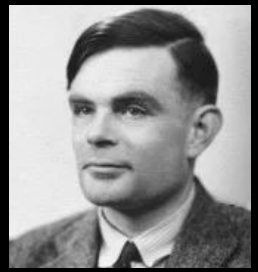
âncora



positivo



negativo



Triplet Loss

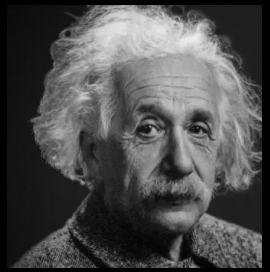
Desejamos que:

$$||x_i^a - x_i^p||_2^2 + \alpha < ||x_i^a - x_i^n||_2^2, \forall (x_i^a, x_i^p, x_i^n) \in \tau$$

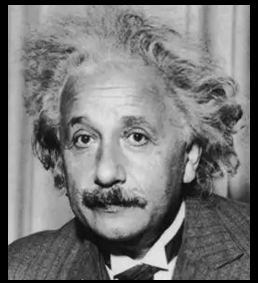
Onde:

- x^a é a âncora.
- x^p é a imagem positiva.
- x^n é a imagem negativa.
- $||\cdot||_2$ é a norma L2.
- $\mathcal{T}$ é o conjunto de todas triplets, e tem tamanho *N*.
- α é uma margem mínima entre pares positivos e negativos.

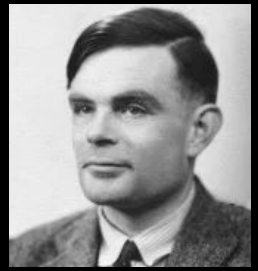
âncora



positivo



negativo



Triplet Loss

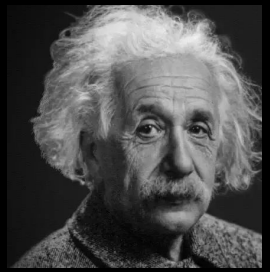
Logo, a função de Loss fica:

$$L = \sum_{i}^{N} ||x_i^a - x_i^p||_2^2 - ||x_i^a - x_i^n||_2^2 + \alpha$$

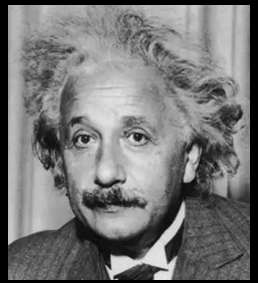
Onde:

- x^a é a âncora.
- x^p é a imagem positiva.
- x^n é a imagem negativa.
- $||\cdot||_2$ é a norma L2.
- $\mathcal{T}$ é o conjunto de todas triplets, e tem tamanho *N*.
- α é uma margem mínima entre pares positivos e negativos.

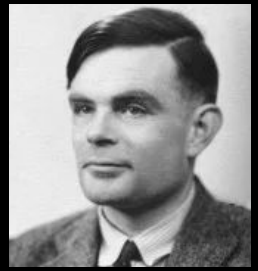
âncora



positivo



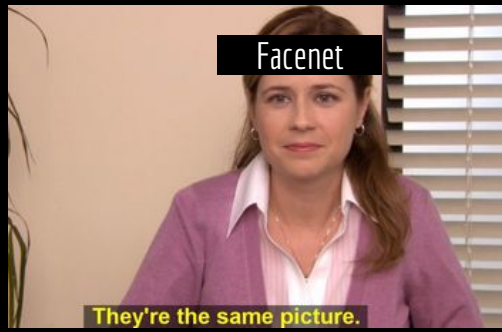
negativo



Facenet

Existem ainda outros detalhes no artigo, como garantir que os dados residem em uma hiperesfera de raio 1, e que as triplets são “difíceis” para melhorar a convergência e evitar o colapso da rede na origem.

Veja os detalhes nos artigos.



Parking Dwell Time

Como usar redes neurais siamesas para cronometrar automaticamente o tempo de permanência de veículos parados?

ICMLA 2024 - Miami, Estados Unidos.

Aceito para publicação.

Marcelo

Heloisa

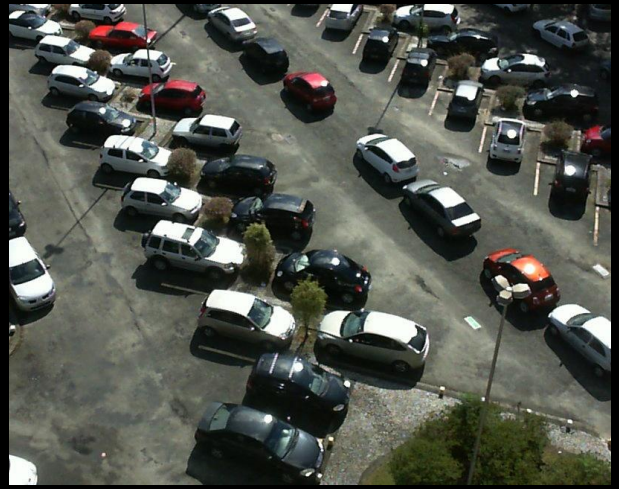
Paulo

Luiz

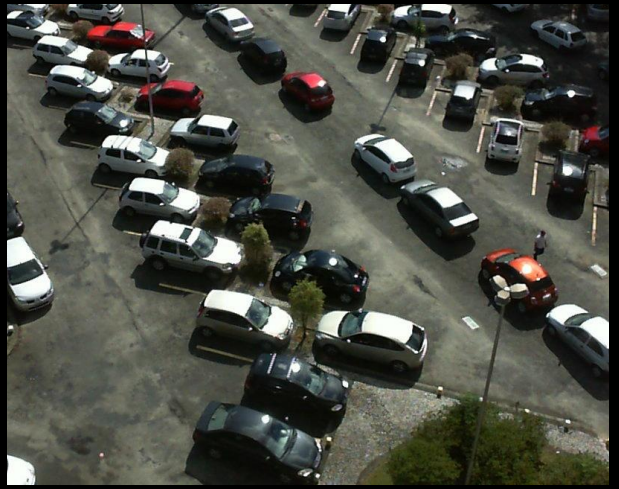
Luiz

Parking Dwell Time

9h40



9h45

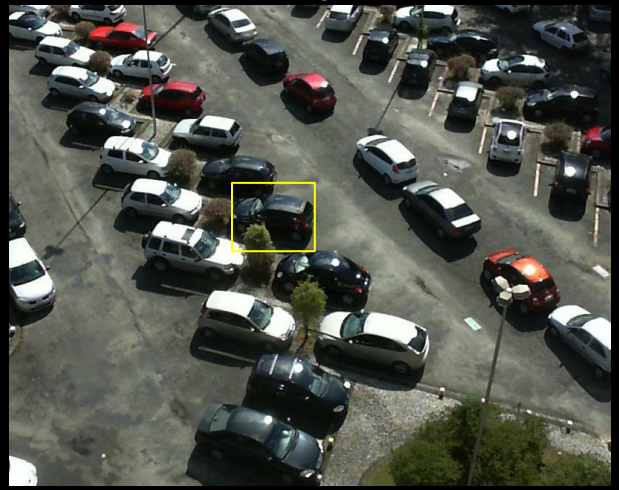


9h50

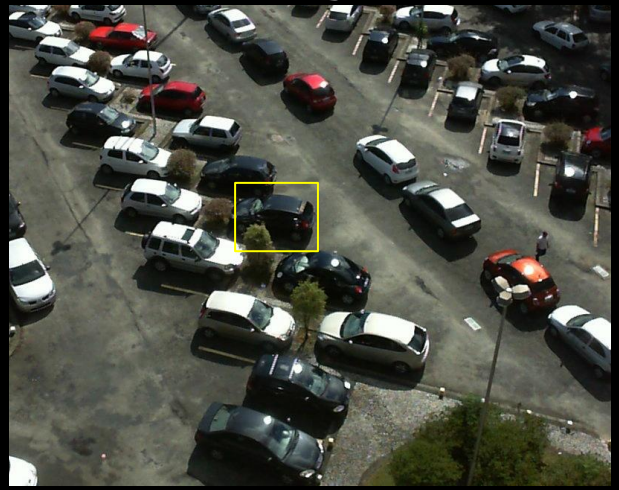
Duas fases: 1 - Encontrar os veículos nas imagens.

Parking Dwell Time

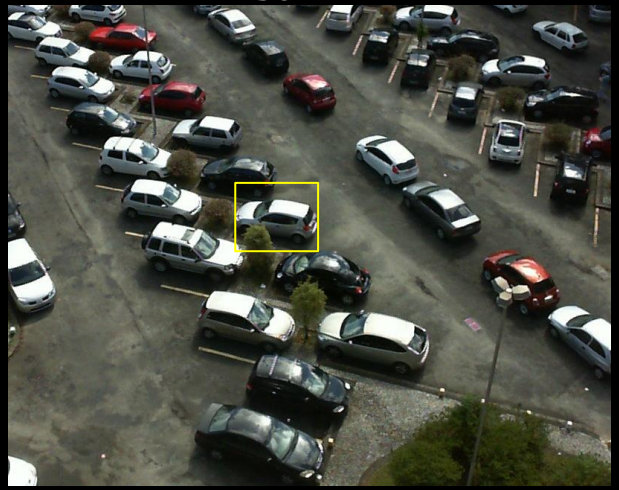
9h40



9h45



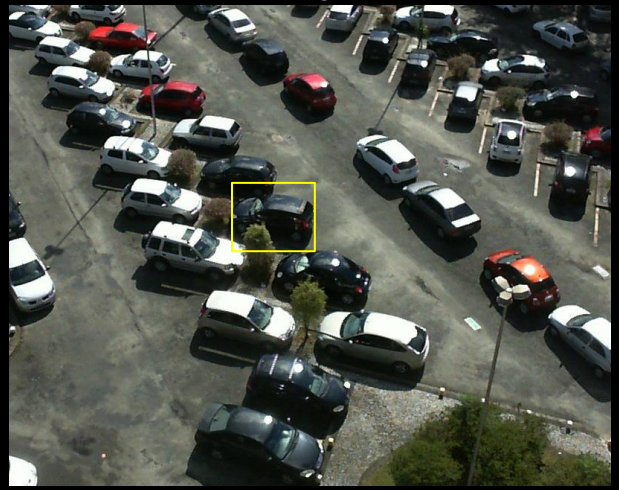
9h50



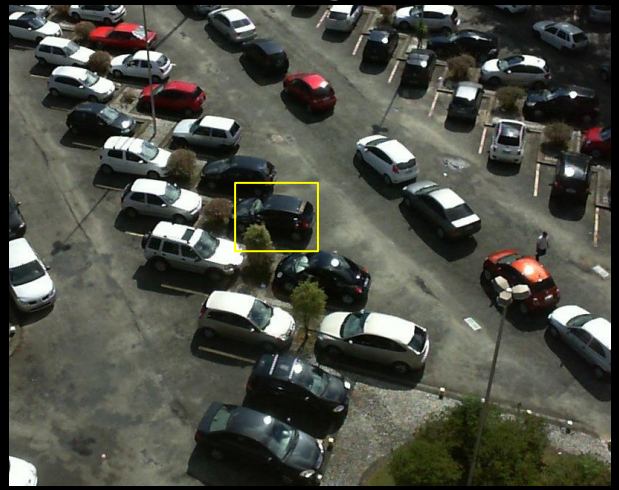
Duas fases: 1 - Encontrar os veículos nas imagens.

Parking Dwell Time

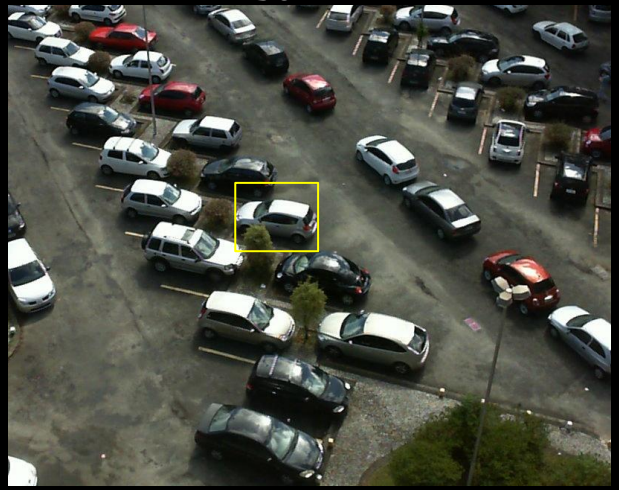
9h40



9h45



9h50



=

≠

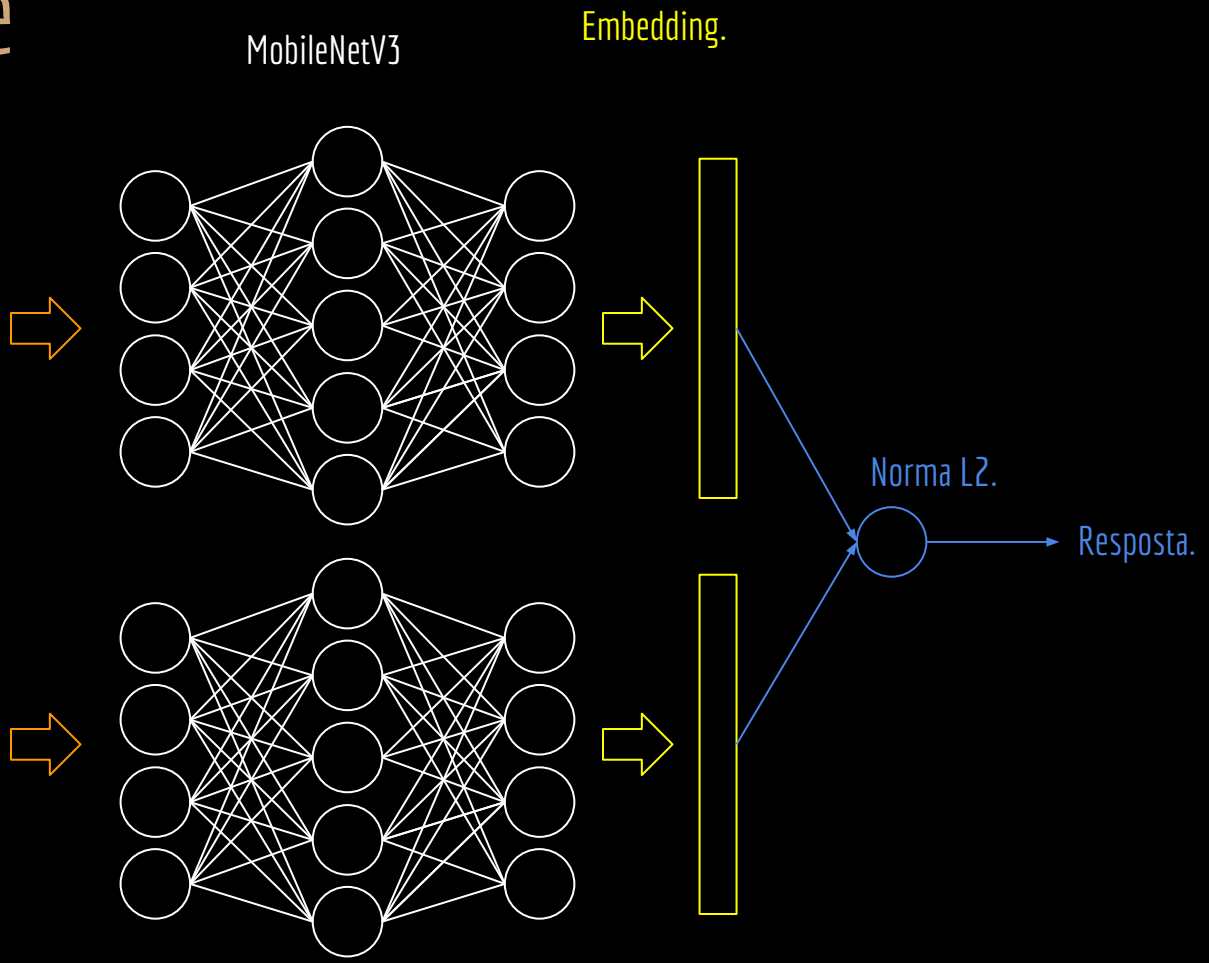
Duas fases: 2 - Comparar usando uma rede siamesa. Usar o resultado para cronometrar o tempo de permanência.

Parking Dwell Time

Rede Siamesa que usa uma MobileNetV3 pré-treinada na ImageNet.

Fine-Tuning para comparar carros.

Contrastive Loss com função de perda.



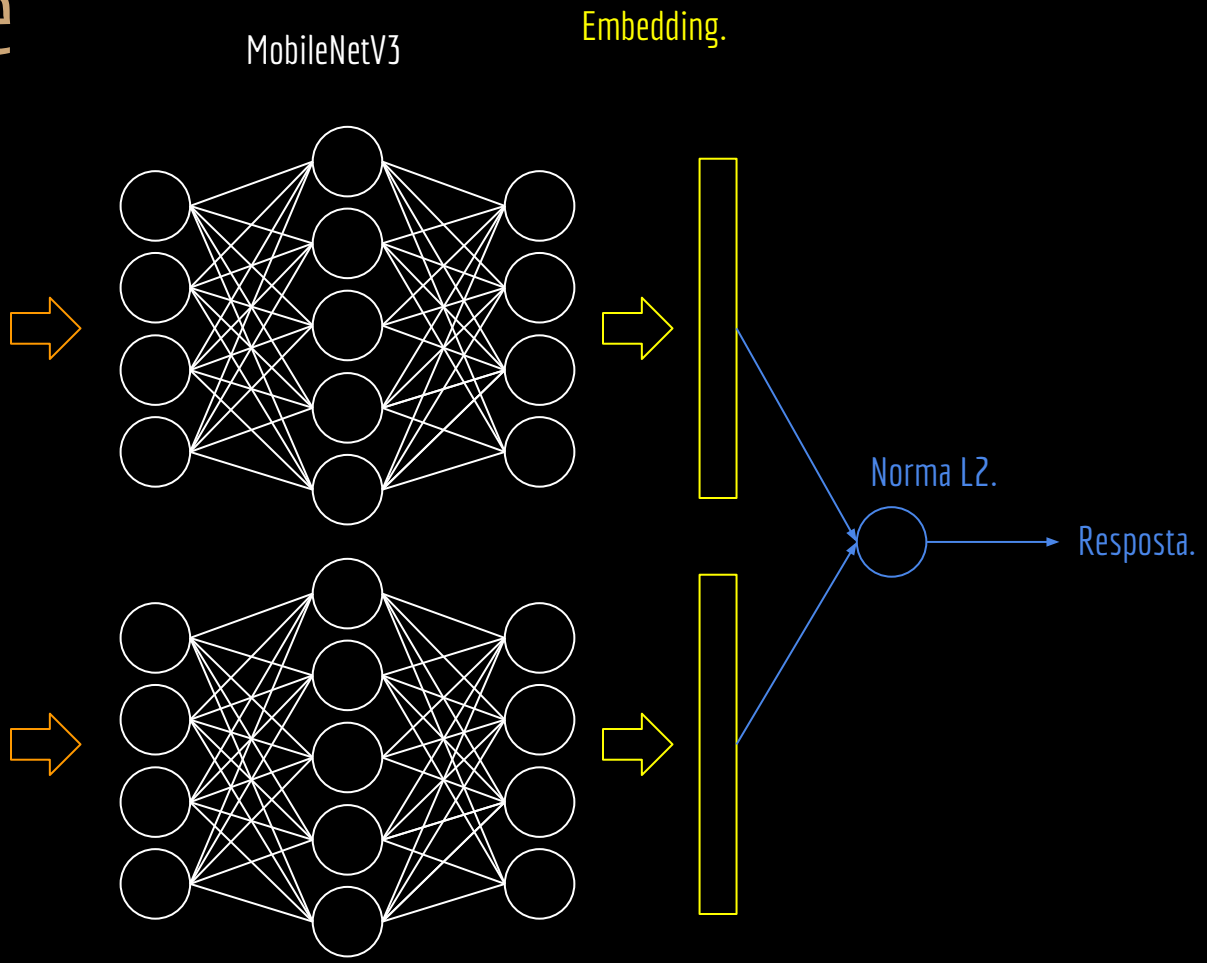
Parking Dwell Time

Rede Siamesa que usa uma MobileNetV3 pré-treinada na ImageNet.

Fine-Tuning para comparar carros.

Contrastive Loss com função de perda.

No exemplo disponibilizado, trocamos pelo triplet loss.



Faça você mesmo

Execute e entenda o exemplo disponibilizado.

Exercícios

1. A rede treinada usa um limiar de decisão arbitrário.
   a. Estude como calcular o Equal Error Rate (EER) para problemas binários utilizando a curva RoC.
   b. Calcule o EER nos dados de validação. Utilize o valor como novo limiar de decisão.
2. Troque a triplet loss pela contrastive loss (usada no paper original), e faça os ajustes necessários.
   a. O algoritmo de treinamento precisará ser alterado. A seguir é dada a função que faz o cálculo do contrastive loss, e a nova função de forward.

```
def __euclidean_distance(self, x, y):
    return torch.sqrt(F.relu(torch.sum((x - y) ** 2, dim=1, keepdim=True)))

def forward(self, input1, input2):
    tower1 = self.__embedding_network(input1)
    tower2 = self.__embedding_network(input2)
    return self.__euclidean_distance(tower1, tower2)
```

```
class ContrastiveLoss(torch.nn.Module):
    def __init__(self, margin=1.0):
        super(ContrastiveLoss, self).__init__()
        self.__margin = margin

    def forward(self, y_true, y_pred):
        square_pred = y_pred ** 2
        margin_square = torch.relu(self.__margin - y_pred) ** 2
        return torch.mean((1 - y_true) * square_pred + y_true * margin_square)
```

Referências

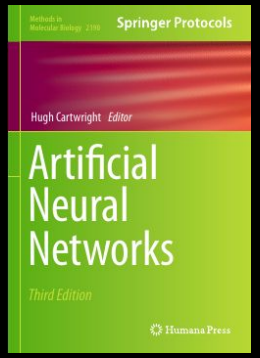
Hugh Cartwright (Editor). Artificial Neural Networks. Springer New York. 2017.

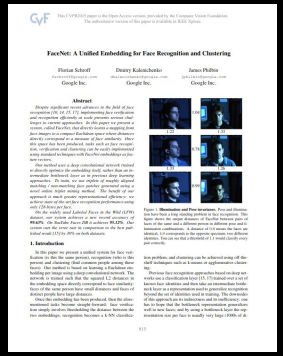
Schroff, F., Kalenichenko, D., & Philbin, J. Facenet: A unified embedding for face recognition and clustering. CVPR. 2015.

Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., & Shah, R. Signature verification using a" siamese" time delay neural network. Neurips. 1993.

Baldi, P., & Chauvin, Y. Neural networks for fingerprint recognition. Neural Computation, 5(3), 402-418. 1993.

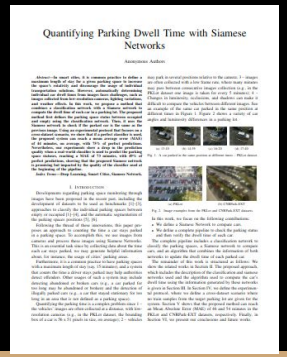
Ribas, Benedet, Zanlorensi, Oliveira, Almeida. Quantifying Parking Dwell Time with Siamese Networks. ICMLA. 2024.




FaceNet: A Unified Embedding for Face Recognition and Clustering


Signature Verification using a "Siamese" Time Delay Neural Network


Neural Networks for Fingerprint Recognition


Quantifying Parking Dwell Time with Siamese Networks

Licença