Siamese Networks

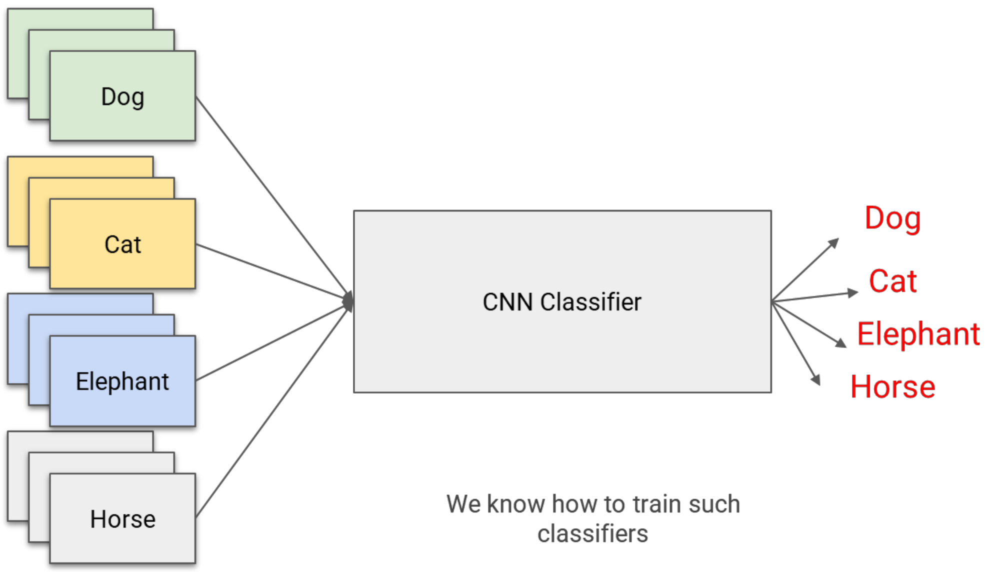
Topics Covered

1. Understanding Siamese Network and One-Shot Learning
2. Siamese Neural Networks
3. Applications And Usage Scenarios
4. Loss Function for Siamese Network
5. Data Preparation
6. Summary

Understanding Siamese Network and One-Shot Learning

Deep Convolutional Neural Networks have become the state-of-the-art methods for image classification tasks. However, one of the biggest limitations is they require a lot of labelled data. In many applications, collecting this much data is sometimes not feasible. One Shot Learning aims to solve this problem.

In case of standard classification, the input image is fed into a series of layers, and finally at the output we generate a probability distribution over all the classes (typically using a Softmax). For example, if we are trying to classify an image as cat or dog or horse or elephant, then for every input image, we generate 4 probabilities, indicating the probability of the image belonging to each of the 4 classes.

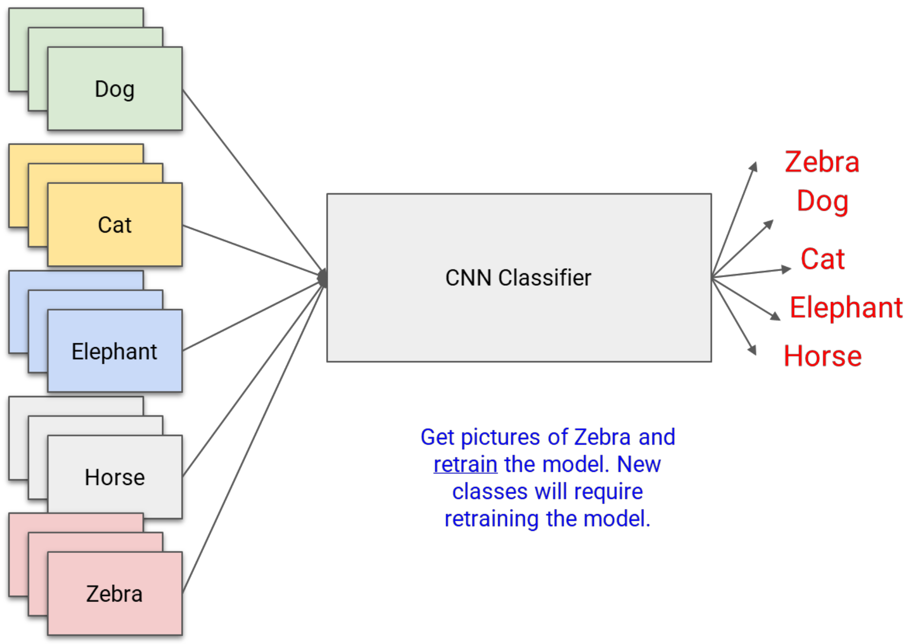


Two important points must be noticed here.

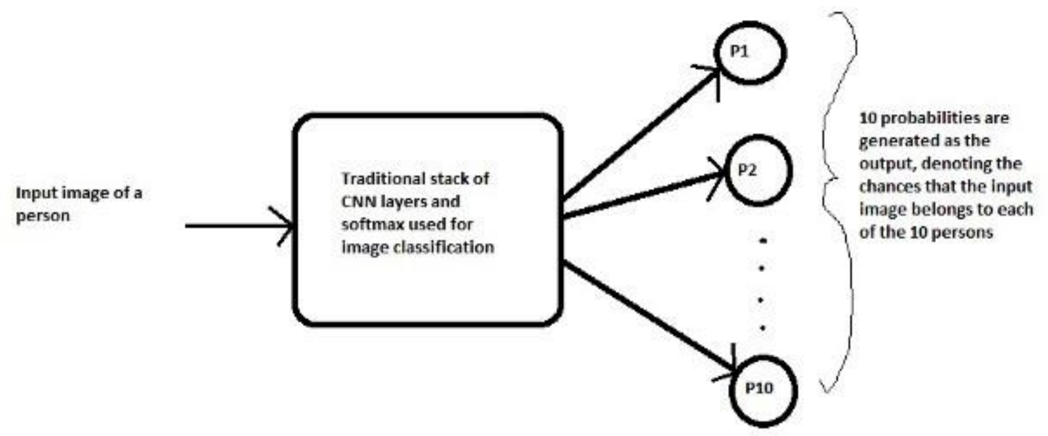
First, during the training process, we require a large number of images for each of the class (cats, dogs, horses and elephants).

Second, if the network is trained only on the above 4 classes of images, then we cannot expect to test it on any other class, example “zebra”.

If we want our model to classify the images of zebra as well, then we need to first get a lot of zebra images and then we must **re-train** the model again. There are applications wherein we neither have enough data for each class and the total number classes is huge as well as dynamically changing. Thus, the cost of data collection and periodical re-training is too high. On the other hand, in a **one shot classification**, we require only one training example for each class.


Dog
Cat
Elephant
Horse
Zebra
CNN Classifier
Zebra
Dog
Cat
Elephant
Horse
Get pictures of Zebra and retrain the model. New classes will require retraining the model.

Assume that we want to build face recognition system for a small organization with only 10 employees (small numbers keep things simple). Using a traditional classification approach, we might come up with a system that looks as below:


Input image of a person
Traditional stack of CNN layers and softmax used for image classification
P1
P2
P10
10 probabilities are generated as the output, denoting the chances that the input image belongs to each of the 10 persons

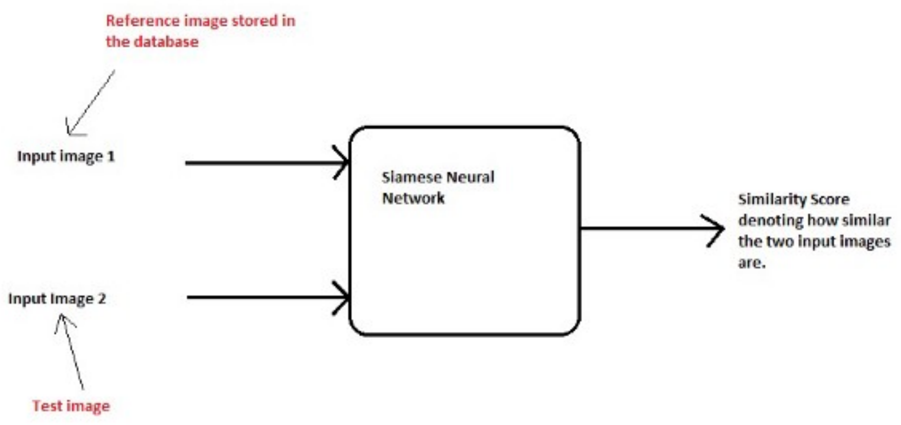
Standard classification using CNN

Problems:

a) To train such a system, we first require a lot of **different** images of each of the 10 persons in the organization which might not be feasible. (Imagine if you are doing this for an organization with thousands of employees).

b) What if a new person joins or leaves the organization? You need to take the pain of collecting data again and re-train the entire model again. This is practically not possible specially for large organizations where recruitment and attrition is happening almost every week.

Now let's understand how do we approach this problem using one shot classification which helps to solve both of the above issues:


Reference image stored in the database
Input image 1
Input Image 2
Test image
Siamese Neural Network
Similarity Score denoting how similar the two input images are.

One Shot Classification

Instead of directly classifying an input(test) image to one of the 10 people in the organization, this network instead takes an extra reference image of the person as input and will produce a similarity score denoting the chances that the two input images belong to the same person. Typically the similarity score is squished between 0 and 1 using a sigmoid function; wherein 0 denotes no similarity and 1 denotes full similarity. Any number between 0 and 1 is interpreted accordingly.

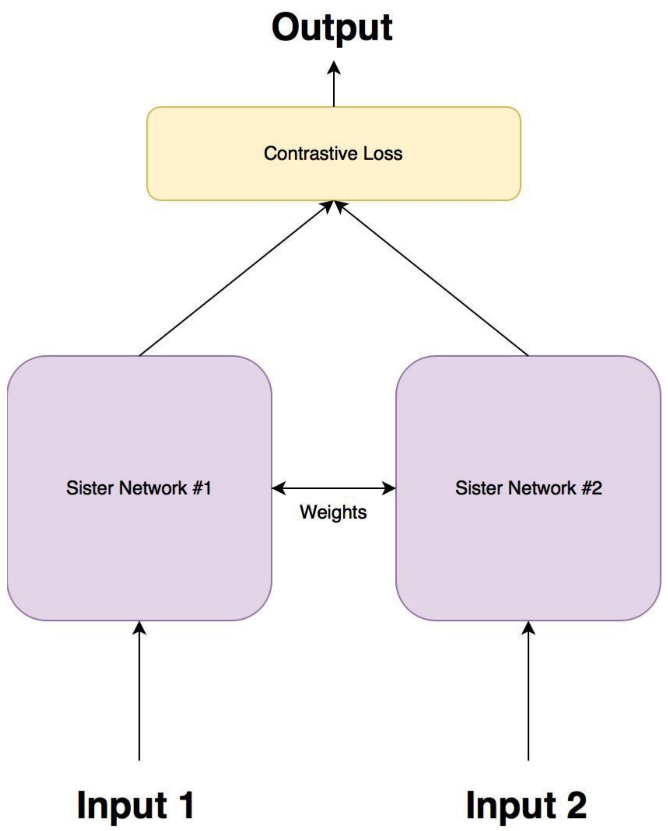
Notice that this network is not learning to classify an image directly to any of the output classes. Rather, it is learning a **similarity function**, which takes two images as input and expresses how similar they are.

It solves the two problems we discussed above as below
a) In a short while we will see that to train this network, you do not require too many instances of a class and only few are enough to build a good model.
b) But the biggest advantage is that, let's say in case of face recognition, we have a new employee who has joined the organization. Now in order for the network to detect his face, we only require a **single** image of his face which will be stored in the database. Using this as the reference image, the network will calculate the similarity for any new instance presented to it. Thus, we say that network predicts the score in **one shot**.

Siamese Neural Networks

A Siamese neural network (sometimes called a twin neural network) is an artificial neural network that contains two or more identical subnetworks which means they have the same configuration with the same parameters and weights. Usually, we only train one of the subnetworks and use the same configuration for other sub-networks. These networks are used to find the similarity of the inputs by comparing their feature vectors.

A Siamese network consists of two identical neural networks, each taking one of the two input images. The last layers of the two networks are then fed to a contrastive loss function, which calculates the similarity between the two images. Refer below illustration to help explain this architecture.



There are two sister networks, which are identical neural networks, with the exact same weights.

Each image in the image pair is fed to one of these networks.

Applications and Usage Scenarios

Siamese networks have wide-ranging applications. Here are a few of them:

- **One-shot learning.** In this learning scenario, a new training dataset is presented to the trained (classification) network, with only one sample per class. Afterwards, the classification performance on this new dataset is tested on a separate testing dataset. As siamese networks first learn discriminative features for a large specific dataset, they can be used to generalize this knowledge to entirely new classes and distributions as well. In (*Koch, Gregory, Richard Zemel, and Ruslan Salakhutdinov. "Siamese neural networks for one-shot image recognition." ICML Deep Learning Workshop. Vol. 2. 2015.*) the authors use this capability to do one-shot learning on the MNIST dataset using a network trained on the Omniglot dataset (an entirely different image dataset).

- **Pedestrian tracking for video surveillance** (*Leal-Taixé, Laura, Cristian Canton-Ferrer, and Konrad Schindler. "Learning by tracking: Siamese cnn for robust target association." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. 2016.*). In this work, a siamese CNN network is combined with size and position features of image patches to track multiple persons in the field-of-view of the camera by detecting their position in each video frame, learning the associations between multiple frames and computing the trajectories.

- **Cosegmentation** (*Mukherjee, Prerana, Brejesh Lall, and Snehith Lattupally. "Object cosegmentation using deep Siamese network." arXiv preprint arXiv:1803.02555 (2018).*).

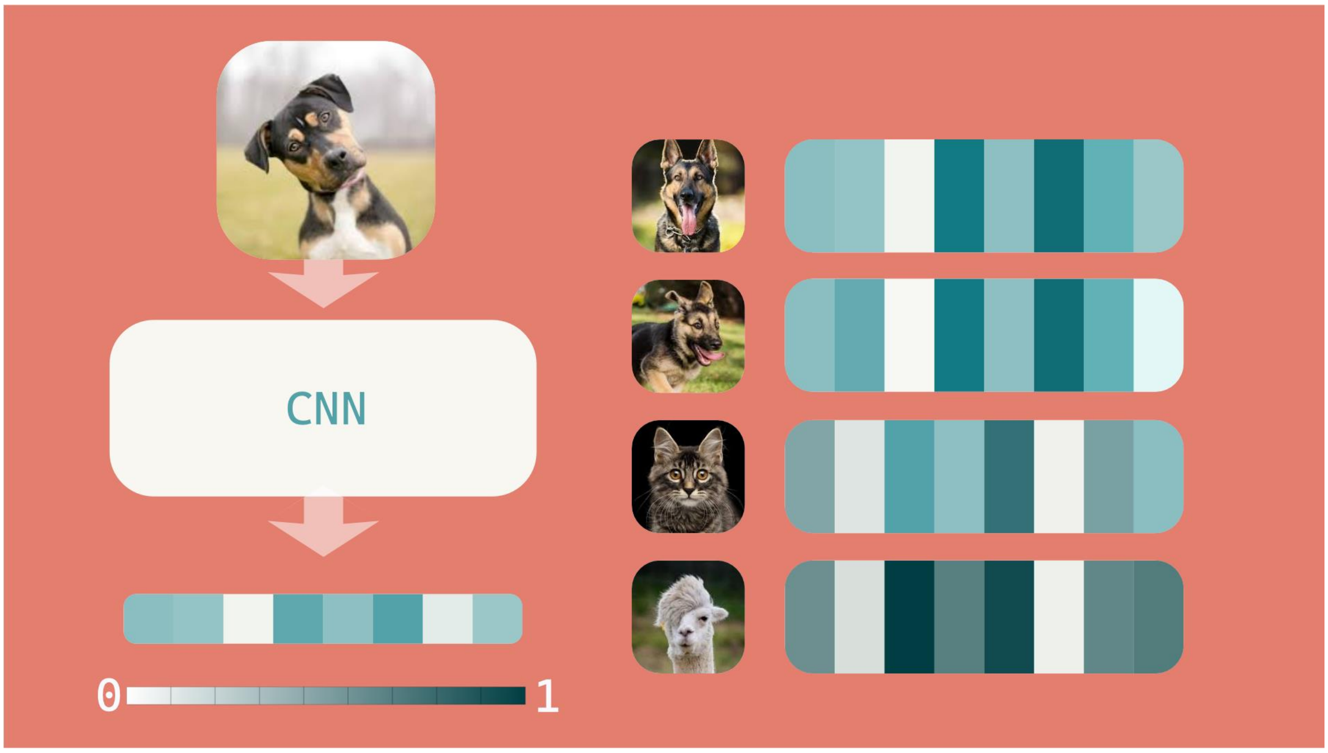
- **Matching resumes to jobs** (*Maheshwary, Saket, and Hemant Misra. "Matching Resumes to Jobs via Deep Siamese Network." Companion of the The Web Conference 2018 on The Web Conference 2018. International World Wide Web Conferences Steering Committee, 2018.*). In this exotic application, the network tries to find matching job postings for applicants. In order to do this, a trained siamese CNN network extracts deep contextual information from both the postings and the resumes and computes their semantic similarity. The hypothesis is that matching resume posting pairs will rank higher on the similarity scale than non-matching ones.

- This model can also be used for signature forgery detection & face detection

Loss Function for Siamese Network

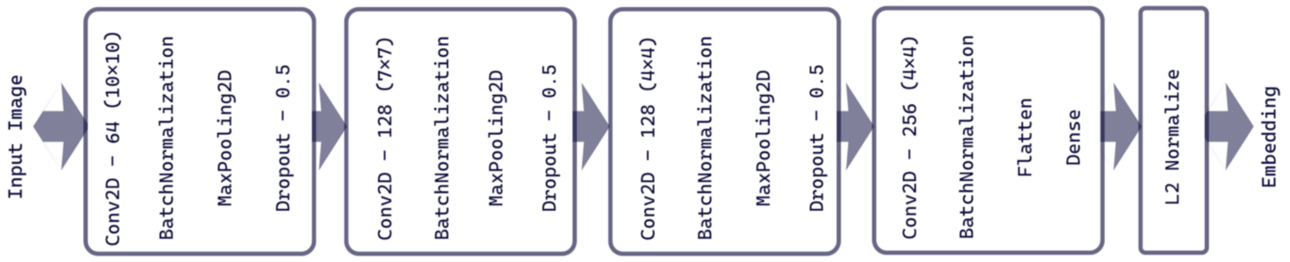
Understanding embedding

Suppose, we use n-dimensional space to map our image, where each dimension corresponds to value of a particular trait/or pattern. Each dimension narrates a unique visual feature of the input image. For example in the case of different animal images, an output embedding might look like this(intensity of colour denotes its value between 0 to 1) —



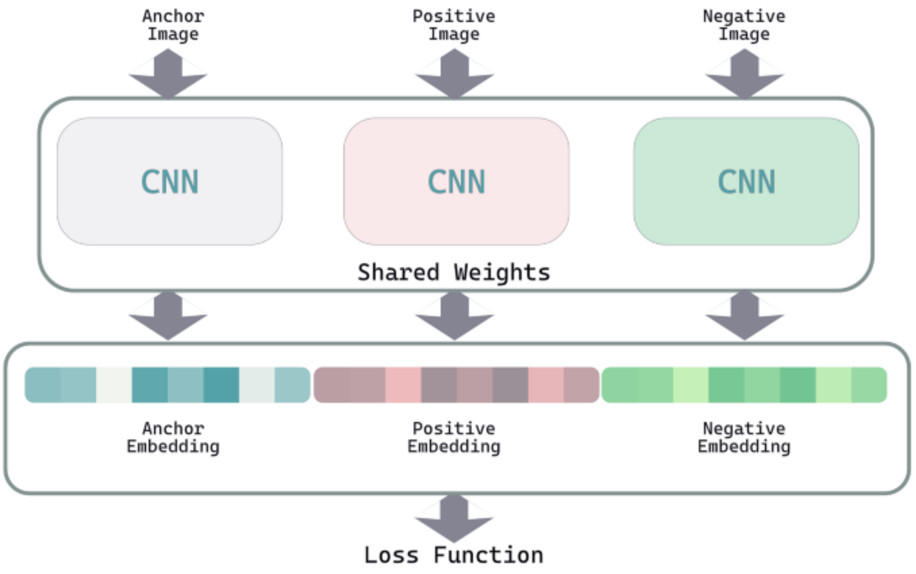
First two images of dog, outputs similar embedding, whereas third and fourth output embedding of cat and llama is very different from a dog because they have a very different bag of visual features.

Working idea of Siamese networks



CNN architecture

Our CNN outputs a 1-D array of the desired size of embedding. We can see that last layer *performs* L2 normalization, this will normalize the output vector and map it to the surface of n-dimensional hyper-sphere of radius 1. This is done to make sure that the value of similarity between images can be compared by calculating distance between two embeddings, as all the embedding will reside on the surface that will give a better result. Our model has three of these CNN and all shares the same weight. This will help our model to learn one similarity and one dissimilarity in each sample. Each sample will contain triplet constituting anchor, positive *and* negative sample image.



Final Siamese Network Architecture

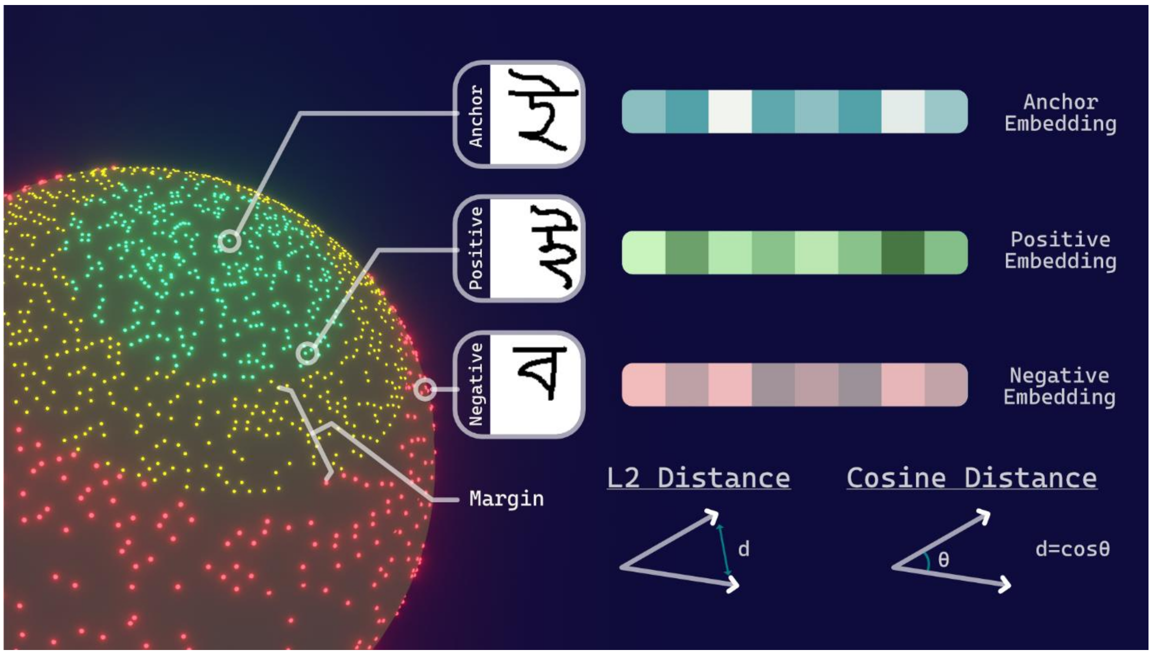
Triplet loss

Now comes the most important part of building this net. Here, we will write the definition of how it compares output embedding and understand similarity/dissimilarity between them, and thus fulfil the task of meta-learning.

Since we have our output embedding mapped onto the surface using earlier discussed *L2* Normalization, we can either use L2 distance or Cosine Similarity. Using triplet loss will allow our model to map two similar images close to one another, and far from dissimilar sample image. This approach is implemented by feeding triplet constituting:

1. ***Anchor Image*** — This is a sample image.
2. ***Positive Image*** — This is just another variation of the anchor image. This helps model learns the similarities between the two images.
3. ***Negative Image*** — This is a different image from above two similar images. This helps our model learn dissimilarities with anchor image.

To increase the distance between similar and dissimilar output vector, and further map similar images close to one another, we introduce one more term called margin. This increases the separation between out similar and dissimilar vector, and also eliminate output of any trivial solution. Since it is difficult for humans to imagine an embedding mapped onto N-dimensional sphere and thus understanding how this loss is working, we made the following render to build intuition of how this thing works for N=3(our much familiar 3D).



This illustration describes our model output after training. Similar images marked as *green* points are mapped closed to one another and dissimilar images marked as *red* mapping are far apart with the least distance of the size of margin show as *yellow*. In an ideal case after training, no point should be mapped in the yellow region of the anchor image. Nevertheless, points often tend to have an overlapping region, since there is much-limited space on the surface. More on this is discussed later.

This similarity/dissimilarity is defined by the distance between two vectors using L2 distance and cosine distance.

So our loss is defined as follows —

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

$$d(a,p) \ \longrightarrow distance\ between\ anchor\ and\ positive\ image$$
$$d(a,n) \ \longrightarrow distance\ between\ anchor\ and\ negative\ image$$
$$margin \ \longrightarrow hyperparameter,\ indicates\ how\ far\ dissimilarities\ should\ be$$

Data Preparation

For Triplet Loss, the objective is to build triplets <anchor, positive, negative> consisting of an anchor image, a positive image (which is similar to the anchor image), and a negative image (which is dissimilar to the anchor image).

There are different ways to define similar and dissimilar images. If you have a dataset having multiple labels as the target class, then images of the same class can be considered as similar, and images across different classes can be considered as dissimilar. If We have a dataset of Geological images having 6 different classes. To generate triplets, first, 2 classes are selected randomly. Then, two images are selected from one class and one image is selected from the other one.

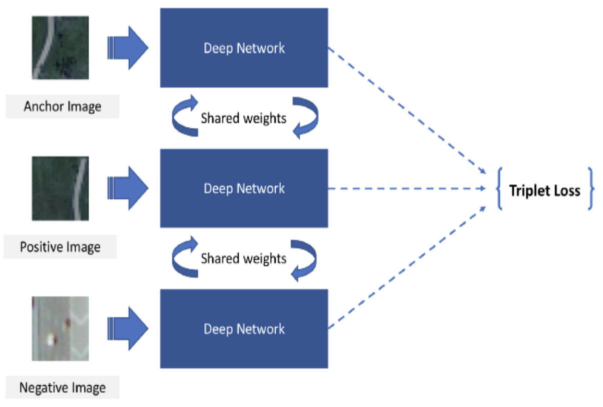
Now, images of the same classes are considered similar, so one of them is used as an anchor and the other one as positive whereas images from the other class is considered a negative image.
Likewise, for every batch, a set of n number of triplets are selected.

Model Architecture

The idea is to have 3 identical networks having the same neural net architecture and they should share weights. I repeat all the networks should share underlying weight vectors.

The last layer of the Deep Network has D-number of neurons to learn D-dimensional vector representation.

Anchor, Positive and Negative images are passed through their respective network and during backpropagation weight vectors are updated using shared architecture. During prediction time, any one network is used to compute the vector representation of input data.



Summary

- A Siamese neural network (sometimes called a twin neural network) is an artificial neural network that contains two or more identical subnetworks which means they have the same configuration with the same parameters and weights.
- Each sample of Siamese network will contain triplet constituting anchor*,* positive *and* negative sample image.
- Anchor Image — This is a sample image.
- Positive Image — This is just another variation of the anchor image. This helps model learns the similarities between the two images.
- Negative Image — This is a different image from above two similar images. This helps our model learn dissimilarities with anchor image.