Similarity Learning and Data Generators

Reed Graff[1]

[1] Undergraduate Student at The University of Texas at Austin
Rangergraff@gmail.com

Abstract

Applying similarity learning methods in classification problems allows for extremely accurate classification with the downside of longer classification times, however, the tools for supporting such methods of classification are few and far between. This paper explores many different methods in which data generators may be developed for similarity learning algorithms, primarily focused on the Siamese Neural Network architecture.

Background

As a point of reference, this paper will be focusing on siamese neural networks and its respective data generators, however, this data paper may still hold value to other fields of research, void of any connection to siamese neural networks. This paper has utilized some common terms, idiomatic expressions, and lingo specific to both data generation and machine learning, which will be addressed now:

Table 1: *Common Terms and Definitions*

Short	Long
AI/ML	Artificial Intelligence/Machine Learning
SiNN	Siamese Neural Network
Scrape	To aggregate information

What Is AI/ML?

AI was originally coined by John McCarthy in 1955, before being further defined as "the construction of computer programs that engage in tasks that are currently more satisfactorily performed by human beings because they require high-level mental processes such as: perceptual learning, memory organization and critical reasoning"[1, 2].

Now AI is a broad term used to describe just about any learning process being completed by a computer, however, in simplest terms it is a computer task that creates a function which is trained to predict outputs based on inputs.

What Is Deep Learning?

Deep Learning is a more specific branch of AI that is typically associated with multi-layer neural networks (more than 3) also with the purpose of training values to better predict outputs dependent upon inputs.

What Is Similarity Learning?

Similarity is a very specific branch of ML that is used for determining the similarity between different data sets[3].

What Are Siamese Neural Networks?

A SiNN is a kind of similarity learning approach to comparing images (or other 2D data), and determining their similarity. SiNNs leverage one neural network which is used to classify each individual image, which can then be used to find the euclidean distance and the overall similarity of the images.

Introduction

Requirements of The Data Generator

Many libraries exist now for generating, changing, and augmenting data, however, there has been some amount of underdevelopment in the area of SiNNs, which can in large part be attributed to the lack of data generation for such architecture.

The purpose of this paper is to expose possible methods of data generation for SiNNs, both as a stand alone generator (one that isn't dependent on other AI/ML libraries), as well as one that may interface with the Tensorflow library.

Table 2: *Tensorflow arguments for image_dataset_from_directory()*

Short	Long
directory	Directory where the data is located.If labels is "inferred", it should containsubdirectories, each containing images for a class.Otherwise, the directory structure is ignored.
labels	Either "inferred"(labels are generated from the directory structure),None (no labels),or a list/tuple of integer labels of the same size as the number ofimage files found in the directory. Labels should be sorted accordingto the alphanumeric order of the image file paths(obtained via os . walk(directory) in Python).
label _ mode	String describing the encoding of labels . Options are: 'int': means that the labels are encoded as integers(e.g. for sparse_categorical_crossentropy loss). 'categorical' means that the labels areencoded as a categorical vector(e.g. for categorical_crossentropy loss). 'binary' means that the labels (there can be only 2)are encoded as float32 scalars with values 0 or 1(e.g. for binary_crossentropy). None (no labels).
class _ names	Only valid if "labels" is "inferred". This is the explicitlist of class names (must match names of subdirectories). Usedto control the order of the classes(otherwise alphanumerical order is used).
color _ mode	One of "grayscale", "rgb", "rgba". Default: "rgb".Whether the images will be converted tohave 1, 3, or 4 channels.
batch _ size	Size of the batches of data. Default: 32.If None , the data will not be batched(the dataset will yield individual samples).
image _ size	Size to resize images to after they are read from disk,specified as (height , width) . Defaults to (256 , 256) .Since the pipeline processes batches of images that must all havethe same size, this must be provided.
shuffle	Whether to shuffle the data. Default: True.If set to False, sorts the data in alphanumeric order.
seed	Optional random seed for shuffling and transformations.
validation _ split	Optional float between 0 and 1,fraction of data to reserve for validation.
subset	Subset of the data to return.One of "training", "validation" or "both".Only used if validation _ split is set.When subset="both" , the utility returns a tuple of two datasets(the training and validation datasets respectively).
interpolation	String, the interpolation method used when resizing images.Defaults to bilinear . Supports bilinear , nearest , bicubic , area , lanczos3 , lanczos5 , gaussian , mitchellcubic .
follow _ links	Whether to visit subdirectories pointed to by symlinks.Defaults to False.
crop _ to _ aspect _ ratio	If True, resize the images without aspectratio distortion. When the original aspect ratio differs from the targetaspect ratio, the output image will be cropped so as to return thelargest possible window in the image (of size image _ size) that matchesthe target aspect ratio. By default (crop _ to _ aspect _ ratio=False),aspect ratio may not be preserved.
*kwargs	Legacy keyword arguments.

Arguments / Inputs To The Generator

Regarding the development of the generator, this paper seeks to mimic the pre-existing tensorflow function "tf.keras.utils.image_dataset_from_directory", and match the arguments (Table 2) that are supported by the aforementioned function[4] in the Tensorflow integration explained later on.

However, prior to this there will be a stand alone generator developed for the same purpose. The focus, however, of the standalone is to provide a more fundamental understanding of the generator and will use the following limited list arguments:

- directory
- batch_size

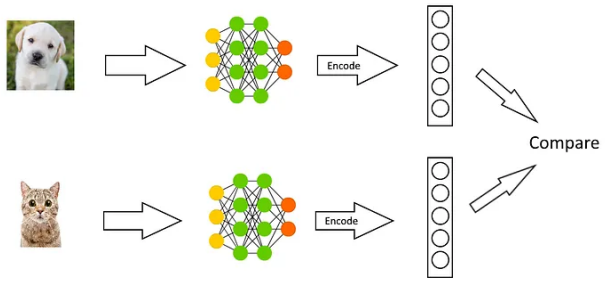

Figure 1: *The basic structure of a Siamese Neural Network, illustrating its parallel nature. Source: [5].*

Output Of The Generator

For the generator which this paper will contribute to Tensorflow, it will match as closely as possible to the already existing "image_dataset_from_directory". This existing function has an output which is dependant upon arguments which we will not be taking for the stand alone generator, and will thus be different in terms of output.

Table 3: *Output of image_dataset_from_directory*

A tf.data.Dataset object.
If label_mode is None, it yields float32 tensors of shape (batch_size, image_size[0], image_size[1], num_channels), encoding images (see below for rules regarding num_channels). Otherwise, it yields a tuple (images, labels), where images has shape (batch_size, image_size[0], image_size[1], num_channels), and labels follows the format described below.

The output of the stand alone generator will them be the following: (images, labels), where images has shape (batch_size, learning_size, image_size[0], image_size[1], num_channels), and where labels has shape (batch_size, learning_size - 1)

The key difference can be found with "learning_size" in the tensor shape of "images", where "learning_size" represents the number of images being used in parallel for the learning process as seen in (Figure 1). For example, it would makes sense for a triplet loss similarity learning architecture, the learning_size would be 3, as there are 3 images being used in parallel for the learning process. However, this variable allows for other architectures to be supported as well.

Additionally, "labels" is changed to a different shape to also allow for other architectures to be supported.

Figure 2: *An illustration of triplet loss for SiNNs, resulting in a "learning_size" = 2. Source: [5].*

For example, if the architecture is a triplet loss similarity learning architecture, the labels would be of shape (batch_size, 2), where the first column represents the similarity of the first image (the anchor) to the second image, and the second column represents the similarity of the first image (the anchor) to the third image. Meanwhile in a case where the learning_size is 2, the labels would be of shape (batch_size, 1), where the first column represents the similarity of the first image (the anchor) to the second image.

Initial Attempt

The initial attempt towards tackling this challenge was anchored in the idea of iterating through the directory, and producing every possible combination of images, this is accomplished by the code in Figure 3.

However, this approach is not scalable.

Firstly, this approach will lead to data heavily biased towards dissimilar images due to the nature of permutations. For example, look at a directory database that follows the structure shown in Table 4.

Table 4: *Example structure of a tree directory dataset.*

Class		
Number	Identification	Data Count
0	Red Oak	10
1	Cedar	10
2	Dogwood	10
3	Maple	10
4	Hickory	10

Assuming that within the main directory we see 5 folders representing classes, each with 10 images; With a Tuplet loss, and using 1 singular image of a class as an anchor, we can see that there may only be 10 possible positive cases (including pairing the anchor with itself), and 40 possible negative cases. Of course, as we translate across the dataset we will acquire additional positive cases, however, the issue of misrepresentation of the dataset is only exacerbated.

```
slots = data["files & structure"].keys()
for slot_1 in slots:
    print("Generating data for slot: " + slot_1)
    for file_1 in data["files & structure"][slot_1]:
        for slot_2 in slots:
            for file_2 in data["files & structure"][slot_2]:
                if slot_1 == slot_2:
                    im_1 = np.asarray(Image.open(input_folder_path + slot_1 + "/" + file_1).convert('RGB').resize((64, 64))).tolist()
                    im_2 = np.asarray(Image.open(input_folder_path + slot_2 + "/" + file_2).convert('RGB').resize((64, 64))).tolist()
                    X_train.append([im_1, im_2])
                    Y_train.append(1)
                else:
                    im_1 = np.asarray(Image.open(input_folder_path + slot_1 + "/" + file_1).convert('RGB').resize((64, 64))).tolist()
                    im_2 = np.asarray(Image.open(input_folder_path + slot_2 + "/" + file_2).convert('RGB').resize((64, 64))).tolist()
                    X_train.append([im_1, im_2])
                    Y_train.append(0)
```

Figure 3: *An illustration of triplet loss for SiNNs, resulting in a "learning_size" = 2. Source: [5].*

Additionally, in the case of smaller datasets, it will lead to a lot of unnecessary loops, especially, if we are interested in getting a unique pair of images, and not one the model has already seen.

References

[1] Gil Press. "A Very Short History Of Artificial Intelligence". In: *Forbes* (Oct. 2022). URL: https://www.forbes.com/sites/gilpress/2016/12/30/a-very-short-history-of-artificial-intelligence-ai/?sh=a3401fc6fba2.

[2] 2014. URL: https://www.coe.int/en/web/artificial-intelligence/history-of-ai#:~:text=The%20term%20%22AI%22%20could%20be,because%20they%20require%20high%2Dlevel.

[3] 2023. URL: https://www.aiforanyone.org/glossary/similarity-learning.

[4] 2023. URL: https://www.tensorflow.org/api_docs/python/tf/keras/utils/image_dataset_from_directory.

[5] Eric Craeymeersch. *One Shot learning, Siamese networks and Triplet Loss with Keras*. Sept. 2019. URL: https://medium.com/@crimy/one-shot-learning-siamese-networks-and-triplet-loss-with-keras-2885ed022352.