1 Problem statement

In this task, we consider the problem of character recognition from different alphabets. The settings of the character recognition task is described as follows. Given a query image x_q and a set of 5 candidate images $\{x_0, x_1, x_2, x_3, x_4\}$, the constructed model should return the image that is the most similar to the query image from the set of candidate images. To this end, we consider the Omniglot data set, which is composed of 20 images for each of the approximately 1600 handwritten characters from 50 different alphabets. The provided data consists of a training dataset for the purpose of training an appropriate model and a query dataset for testing the performance of the constructed model. The training dataset contains 18,800 images of handwritten characters of size 28x28. Each pixel takes a value of either 0 or 1 and each image depicts one of 940 different characters from 29 different alphabets. Additionally, each image is accompanied with a label y which indicates the character depicted in the image, denoted with as an integer. The query dataset contains 100 sets of 6 28x28 images of handwritten characters which are not present in the training dataset. Each test set consists of 1 query image and 5 candidate images. The objective is to develop a model that is capable of selecting the image depicting the same character as the query image, i.e., the anchor image out of 5 candidate images.

Given that the number of classes is large (i.e., 940) and we only observe very few data points per class (i.e., 20), making a prediction becomes challenging as it leaves very little room for the right pattern to be correctly distinguished from the rest of the information in the data. This is essentially a problem under the class of *one-shot learning*, where we aim to train a model that can learn the specific characteristics and patterns for the handwritten characters and then introduce the downstream task of detection instead of detecting a specific class for each image. For this purpose, we employ the metric learning technique which learns a distance metric for which the images from the same class are 'closer' together than the images from different classes. Note that the goal of this task is to select the closest image to a query image out of 5 unseen candidate images. Thus, the metric learning technique is deemed appropriate for this setting as the model can develop features that differentiate the characters and use them to generalize given a very few observed examples. One of the most robust network architecture for metric learning is the Triplet network, which employs in its training set up three data points and a three-component loss function. Due to its robustness, this network architecture has become our choice of model for the current task at hand.

2 Approach

2.1 Network Architecture

2.1.1 Embedding Net

Note that in metric learning, we aim to learn a latent space where examples of different classes can be distinguished using a metric distance. The representations of examples inside this latent space would be one-dimensional latent vectors, which are also called embeddings. These embeddings can be obtained by a convolutional neural network (CNN) whose architecture is shown in Figure 1. There are 3 convolution blocks,

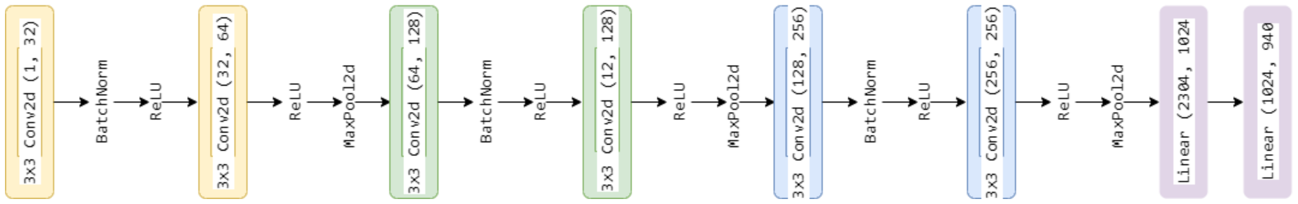

Figure 1: CNN architecture for obtaining the embeddings of input images.

each of which includes: 2 convolutional layers with ReLU activation. Each convolutional layer is followed by batch normalization to normalize the activations of the previous layer within each batch. The number of channels in each convolution is the same and is twice that of the previous convolution. MaxPooling with a kernel size of 2, stride is 0 is applied to reduce overfitting. Zero padding is added to prevent the input tensor from shrinking too fast. The last layers are linear layers, which results in one-dimensional vector of size 940.

2.1.2 Triplet Network

In the setting of the Triplet network, three input images are required, namely the anchor image x_a, the positive image x_p and the negative image X_n. First, an anchor image x_a is selected, then a positive image is selected from the same class as the anchor image x_a whereas the negative image x_n is from a different class. With this settings, the Triplet network can learn both positive and negative distances simultaneously.

Let f denote the function parameterized by the CNN, i.e., the embedding described in Section 2.1.1. The objective is to learn embeddings such that the anchor is closer to the positive image than it is to the negative image by some margin value α:

$$||f(x_p) - f(x_a)||^2 - ||f(x_n) - f(x_a)||^2 + \alpha \leq 0.$$

The triplets are formed in a way that for the images in a minibatch during training, we exhaust all anchor-positive pairs. However, for choosing the negative image, an image is selected uniformly at random from the classes that are not the class of the anchor-positive pairs. This is regarded as a Random Triplet Selector. With the triplets created by the Random Triplet Selector, we can define the Triplet loss as follows.

$$L_{triplet}(x_a, x_p, x_n) = max(0, \alpha + ||f(x_a) - f(x_p)||^2 - ||f(x_a) - f(x_n)||^2).$$

For our model, the margin α is set to 1. The purpose is to update the weights of model parameters during the training phase such that the Triplet loss is minimized, i.e., minimizes the distance between an anchor and a positive which are from the same class while maximizes the distance between the anchor and a negative which belongs to different classes. Here, the Euclidean distance is used to calculate the distance between two latent vectors in the latent space. The architecture of the Triplet network is depicted in Figure 2.

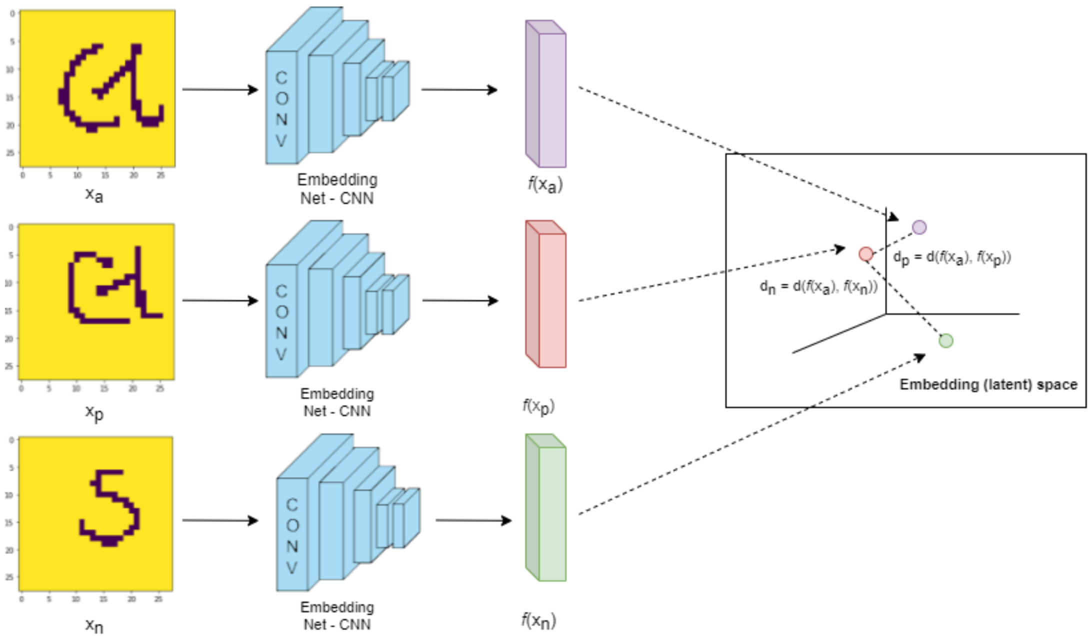

Figure 2: Triplet network for character recognition task.

2.2 Model training

The raw data consists of binary images, thus as a preliminary step we convert the data to float data type. Next, for the purpose of training, the training dataset is stratified-split further into a training set (80%) and a validation set (20%). Furthermore, a balance batch sampler is adapted, which returns a minibatch containing 15 classes and 25 samples for each class. The model is trained for 200 epochs, using the Triplet loss and the SGD optimizer with a learning rate $\eta = 0.001$ and momentum is 0.9.

3 Results

The training loss and validation loss are calculated at each of the 200 epochs, as shown in Figure 3.

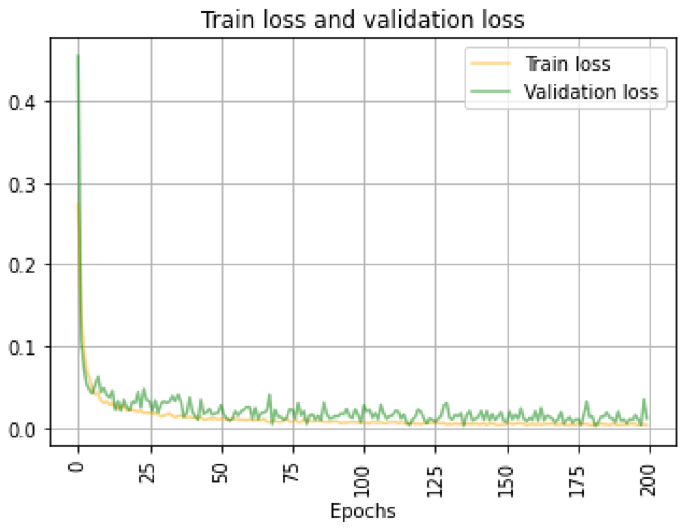

Figure 3: Training loss and validation loss for 200 epochs.

As can be observed, both training loss and validation loss decrease drastically around the first 50 epochs and converge after 100 epochs. This implies that the model is learning quite well and quite fast. The validation loss is quite close to the training loss of the same epoch, though do slightly fluctuate over time.

As the goal of the task is to correctly recognize the candidate image that is most similar to the query image out of 5 candidates, the performance of the model is assessed based on the top-1 and top-3 accuracy. To facilitate this assessment, the following steps are taken: for each test set (i.e., a query with the corresponding 5 candidates), the embeddings of the query image and the 5 candidate images are extracted. We then calculate the Euclidean distance between the query image and each candidate and sort the candidate images by distance in ascending order. The list of corresponding indices for these candidates are then used to calculate the top-1 accuracy and top-3 accuracy. Top-1 accuracy represent the number of times the true label of the query matches the first element of the returned list of indices. Top-3 accuracy represent the number of times the true label of the query belongs in the top 3 element of the aforementioned list. Table 1 presents the top-1 and top-3 accuracy achieved by our model after training for 200 epochs.

Top-1 accuracy	Top-3 accuracy
0.73	0.96

Table 1: Top-1 accuracy and top-3 accuracy.

4 Analysis and conclusions

For the task at hand, we employ a Triplet network using Triplet loss with a random selector as our model of choice. The model is shown to learn effectively where the losses significantly drop over time and converge quickly. The top-1 accuracy and top-3 accuracy, which are the main assessors of the model performance are reported to be very high (0.94 and 1.00, respectively), indicating that most of the time the model is able to recognize correctly the candidate that is the most similar to the query image. However, this does not necessary mean that the model is utterly perfect. For more challenging cases in which the patterns and attributes of the handwritten characters depicted in the query and the candidate images are similar, the model is likely to make mistake. The results of 5 random query from the given task is shown in Figure 4.

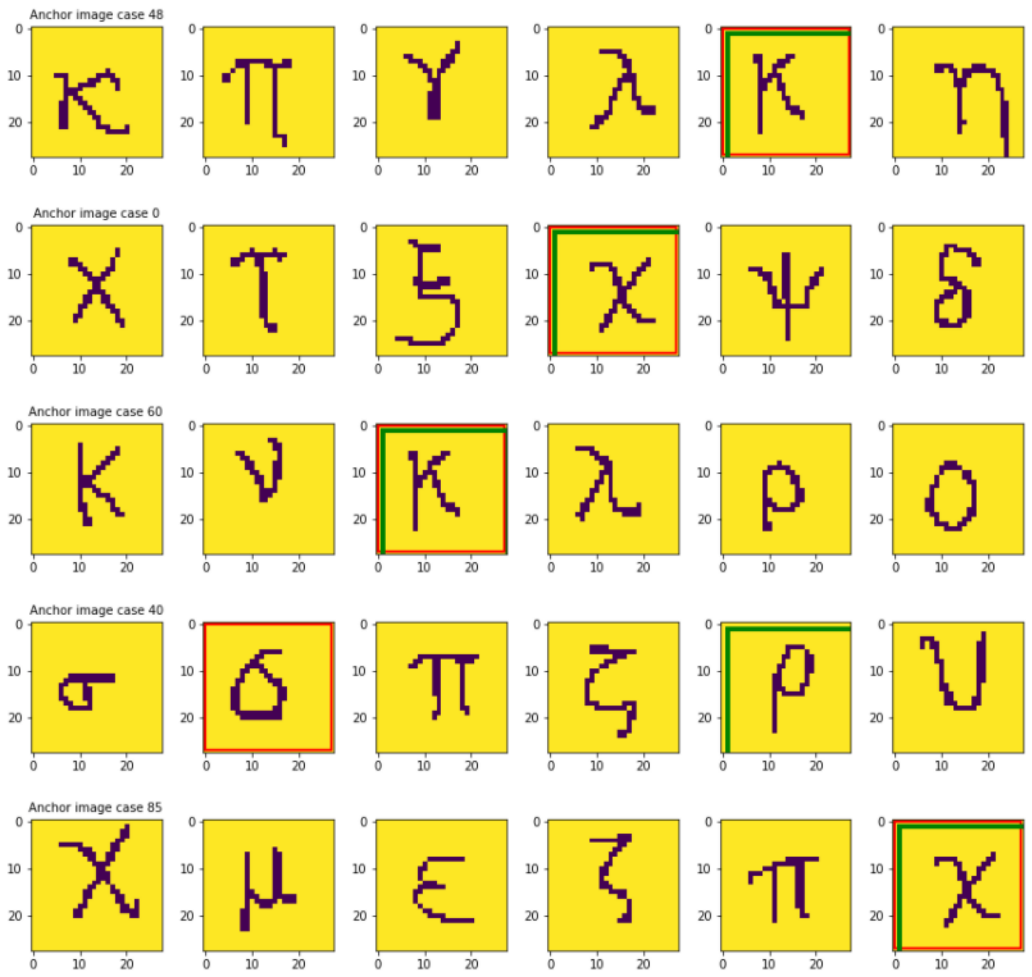

Figure 4: The results of 5 random query returned by the model where the image in the red boundary is the target image and the image in the green boundary is the image deemed to be the closest to the query image by the model.

The first column corresponds to the query images (anchor images) of each of these 5 cases. All other images are the candidate images from which the task is to recognize the anchor image. The image enclosed in the red boundary denotes the target image that belongs to the same class as the anchor image. The image enclosed in the green boundary indicates the image deemed to be the anchor image by the model. Desirably, the red boundary and the green boundary would be overlapping, indicating that the model's verdict matches the ground truth, which is shown to be the majority of the cases. However, as can be seen in the 4^{th} row of Figure 4, the model has incorrectly identified the target image due to the extreme similarity between the image enclosed in the green boundary and the anchor image.