MLP Coursework 4: One-shot learning with Omniglot

G81: s1312650, s1456537, s1747971

Abstract

One of the hallmarks of human intelligence and a true challenge for machines is the ability to effectively reapply skills from one context to another with as little retraining as possible. This paper investigates the performance limits of Siamese and Triplet networks with convolutional neural network (CNN) branches for the purposes of one-shot character recognition on the Omniglot dataset. We report and compare experimental results across different loss functions, input sampling methods, parameter sharing approaches and use of normalisation and regularisation. We are the first to apply Triplet networks on the Omniglot set for one-shot learning and we debut our own evaluation and training methodology inspired by the work of Koch et al. (2015). Our final model, a Triplet CNN with global loss, achieved one-shot test accuracy of 89.25%, beating the Siamese CNN baseline's 84.50% from the interim report, while underperforming Koch et al. (2015)' final model by 2%, which we explain with undertraining and more modest parameter tuning due to limited time and computational resources.

1. Introduction

One-shot learning (Fei-Fei et al., 2006) is a minimalistic mode of generalisation, originating from computer vision, whose main motivation is to mimic the way humans learn in order to enable machines to make classification predictions on a wide family of similar but novel problems. The core constraint of this type of task is that the algorithm should decide upon the class of a test instance after seeing just one training example from each unseen class in question. To achieve this, Fei-Fei et al. (2006) stated that each time when learning from a training sample, some general knowledge should be added to the model and this should not be limited to only the category the given object is classified as. Features and prior knowledge embedded in the model should then be used to make estimates about classification of new objects. One major drawback of systems trained to solve a one-shot task is that they often overfit and fail to generalise to an extent that makes them useless for practical purposes. We counter the aforementioned problem by employing a variety of deep learning techniques in the setting of several modifications of siamese (Bromley et al., 1994) and triplet architectures.

A siamese network consists of two identical computational subgraphs (called *legs* or *branches*) that share all parameters. The two branches produce embeddings for a pair of inputs and try to verify if the latter come from the same class by learning a distance measure. The triplet architecture is an extension to that but tries to learn embeddings between anchor, positive (same class) and negative (different class) triples of inputs instead. The models we considered were always convolutional neural networks and differed only in their size and hyperparameters.

Omniglot (Lake et al., 2011) is one of the most widely researched datasets in the one-shot learning community. It contains 20 binary images per each of 1623 character classes from over 50 alphabets. To make our approach comparable with previous results (Lake et al., 2011; Koch et al., 2015), we have split the data set identically: 40 alphabets in the background set (further subdivided into a training set of 30 alphabets and a validation set of 10 alphabets) and an evaluation set of 10 alphabets. We are only using the training set to train our models and the validation set to tune hyperparameters. We sample the validation set at the end of each epoch in order to estimate the accuracy of our model on the binary verification task and to report intermediate one-shot validation accuracy. We report our final one-shot learning test accuracy only on characters drawn from the evaluation set. The difference between the evaluation set and an ordinary test set is that the former consists entirely of classes (and not only samples) unseen during training by the model.

The main research objective of this paper is to investigate whether triplet networks (Hoffer & Ailon, 2014) constitute a significant improvement over their siamese equivalents for one-shot learning on Omniglot. The work can be seen as a logical continuation of prior research conducted by Koch et al. (2015). While in the previous stage of the the project, we exhaustively experimented with a siamese CNN inspired by Koch et al. (2015)'s model (G81, 2018), here we provide a detailed discussion of experimental results on the performance of triplet networks, and in particular on how different architectural modifications such as regularisation, activation functions, loss functions as well as sampling methods, affect their classification accuracy. We reuse our baseline from the interim report: a fine-tuned adaptation of the siamese convolutional neural network described in Koch et al. (2015). To the best of our knowledge, this is the first paper dedicated to the use of triplet networks in this exact context and therefore the report elaborates on implementation specifics of the triplet network and the possible ways to evaluate its performance on one shot tasks in a way that makes it comparable to the siamese baseline.

2. Models

2.1. Siamese convolutional neural network

A siamese network (Bromley et al., 1994) consists of two identical branches with shared parameters. For the purposes of one-shot learning on Omniglot, we can assume it consists of an input layer that takes a pair of images; two legs that map the images to embedding space and a final output layer that learns a similarity measure based on the L1 distance of the embeddings activated by a sigmoid. The model proposed by Koch et al. (2015) has 4 convolutional and max pooling layers, followed by a fully connected layer, and achieved a test accuracy of 92% on the Omniglot dataset. The leg's parameters are optimized using a regularized cross-entropy loss function. During training pairs of same-class images are oversampled to ensure the leg learns to verify same-class membership correctly.

2.2. Triplet network

While Koch et al. (2015)'s siamese network achieved considerable performance for Omniglot, triplet networks have been reported to outperform siamese networks throughout a number of other problems, ranging from deep ranking for image retrieval to face recognition (Wang et al. (2014); Hoffer & Ailon (2014); Schroff et al. (2015); Kumar et al. (2016)). One of the advantages of using triplets of input rather than pairs is that this accounts for the notion of context when considering similarity. For example, given a dataset of random objects, images of two different people could be classified as coming from the same category, whereas with a dataset consisting only of people's images, they should be viewed as dissimilar (Hoffer & Ailon, 2014). Instead of comparing pairs of inputs independently, triplet networks consider anchor, positive (same-class) and negative (different-class) 3-tuples, thus encouraging more contextualised optimisation (see Figure 1 and 2).

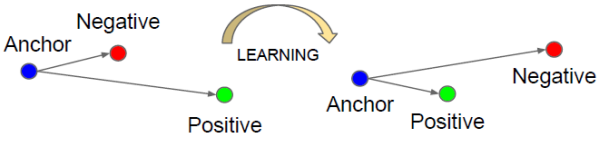

Figure 1. The learning aim of triplet networks is to minimise the distance between anchor-positive examples and maximise the distance between anchor-negative examples (figure taken from Schroff et al. (2015)).

2.2.1. Triplet sampling

An effective sampling technique can result in increased performance of the model, as shown by Wu et al. (2017). They emphasise that appropriate sampling leads to more informative tuples and faster convergence, and this can prove significant within tasks such as one-shot learning where only a limited subset of samples per class can be used due to combinatorial explosion. Additionally for triplet networks, we obtain the embeddings of the distances between

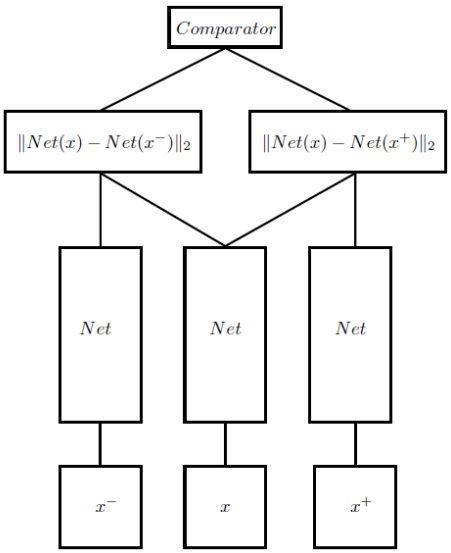

Figure 2. Triplet network defined by Hoffer & Ailon (2014). The subnetworks denoted as *Net* share parameters and each take one example from the triplet of input and propagate it through the network to construct their embeddings.

the anchor and the positive and negative examples (denoted as positive and negative distances respectively). These are used for comparison within the loss function, which makes it possible to influence its effectiveness by how we sample the images of the two branches.

Random sampling. The simplest way to sample from labeled classes is to randomly select the triplets (Hoffer & Ailon, 2014). After selecting an anchor image, the positive image is selected randomly from the same class of the anchor (excluding the anchor image) and the negative is randomly selected among all of the other classes. For this case, image samples' similarity levels are determined strictly based on the class labels. While such random sampling is computationally inexpensive, it will frequently lead to triplets that satisfy (minimise) the defined loss function (Schroff et al., 2015). Consequently, they will not contribute to actively training the network and will result in slower convergence.

Hard negative/positive mining. An alternative approach is to enforce hard negative or positive mining for each of the negative and positive branches respectively (Schroff et al., 2015). For hard negative mining, we obtain the closest image to the anchor which is not in the same class. On the other hand for positive mining, we obtain the furthest image to the anchor which is classified within the same class. However, when dealing with large datasets it is computationally and time-inefficient to compare against all other images.

Offline vs online input sampling. There are two main general approaches to dealing with this issue, either using an online approach or an offline one to generate the required samples (Schroff et al., 2015). The online algorithms limit their scope of hard negative/positive mining search only within a specific minibatch for each training iteration. Con-

versely, offline algorithms compute how close or faraway the images in a subset of the dataset are after a specific number of epochs. How the closeness is defined is determined from problem to problem; one of the approaches is to simply use the Euclidean distance between two images.

2.2.2. Loss functions

The loss functions used for triplet networks differ significantly from those used in Siamese networks and in supervised learning in general. Because the samples are hard-coded to always contain a positive and a negative image in relation to an anchor image, the usage of the actual labeled output becomes redundant (Schroff et al., 2015), and so the triplet network's focus is on distance metric learning. The purpose of loss functions used within the triplet networks is to reduce the distance between the anchor and the positive image, while also increasing the distance between the anchor and the negative image within a minimum margin of difference (Schroff et al., 2015). Then comparing the two relative distances one can learn a distance metric, which the network can use to update the weights of the branches (Schultz & Joachims, 2004). Thus, for our network to learn we want to satisfy the following (Schroff et al., 2015):

$$\|f(x_i^a) - f(x_i^p)\|_2^2 + m < \|f(x_i^a) - f(x_i^n)\|_2^2, \tag{1}$$

where f is the output embedding of the triplet network, a is the anchor image, p is the positive image, n is the negative image, i is the ith triplet sample and m is the margin, which prevents the distances from being pushed to 0 by the network. A different distance metric can be used other than just the squared Euclidean distance, which can be rewritten more generally as:

$$d(f(x_i^a) - f(x_i^p)) + m < d(f(x_i^a) - f(x_i^n)), \tag{2}$$

where d is the distance metric.

The loss functions that we will explore are: two variations of basic triplet losses (Schroff et al., 2015; Kumar et al., 2016), global loss, global plus triplet loss (Kumar et al., 2016) and a SoftMax loss (Hoffer & Ailon, 2014).

A basic form of the triplet loss function, as defined by Schroff et al. (2015) for their *FaceNet* model, can be written as:

$$L(x^a, x^p, x^n) = \frac{1}{N}\sum_{i=1}^{N}(max(0, \|f(x_i^a) - f(x_i^p)\|_2^2 - \|f(x_i^a) - f(x_i^n)\|_2^2 + m)), \tag{3}$$

where N is the batch size. This will be referred to as *Schroff's triplet loss* later in this paper.

An alternative form of the loss function is to constrain the loss value to be between 0 and 1 (Kumar et al., 2016):

$$L(x^a, x^p, x^n) = \frac{1}{N}\sum_{i=1}^{N}(max(0, 1 - \frac{\|f(x_i^a) - f(x_i^n)\|_2^2}{\|f(x_i^a) - f(x_i^p)\|_2^2 + m})) \tag{4}$$

This will be referred to as *Kumar's triplet loss* later in this paper.

The loss functions defined in equations 3 and 4 limit their scope to each individual triplet sample at a time. Kumar et al. (2016) proposed two additional loss functions, global and global plus triplet loss, which attempt to minimise the global error for all of the classes in the training set. The main motivation of this is to improve the network's generalisation capabilities. Additionally, it should help overcome difficulties with sampling for a triplet networks. It avoids the necessity to sample all possible triplets for the network and limits the dependency of fine tuning a subsampling method. The global loss function obtains the mean and variances of the distance embeddings of the network. Thus it will try to minimise the variance of both the distances of the positive and negative against the anchor and the mean distance between the anchor-positive and maximise the mean distance between the anchor-negative. The global loss function for the triplet networks is defined as (Kumar et al., 2016):

$$\begin{aligned} L(x^a, x^p, x^n) = {} & Var(\|f(x^a) - f(x^p)\|_2^2) + \\ & Var(\|f(x^a) - f(x^n)\|_2^2) + \\ & l * max(0, mean(\|f(x^a) - f(x^p)\|_2^2) - \\ & mean(\|f(x^a) - f(x^n)\|_2^2) + t), \end{aligned} \tag{5}$$

where l is the constant that balances the term importance and t is the margin of between the positive and negative distances to the anchor.

The equation in 5 has been further improved by adding the original triplet loss function as defined in equation 4 (Kumar et al., 2016), with the global plus triplet loss function having the following equation:

$$\begin{aligned} L(x^a, x^p, x^n) = {} & g * mean(max(0, \\ & 1 - \frac{\|f(x_i^a) - f(x_i^n)\|_2^2}{\|f(x_i^a) - f(x_i^p)\|_2^2 + m})) + \\ & Var(\|f(x^a) - f(x^p)\|_2^2) + \\ & Var(\|f(x^a) - f(x^n)\|_2^2) + \\ & l * max(0, mean(\|f(x^a) - f(x^p)\|_2^2) - \\ & mean(\|f(x^a) - f(x^n)\|_2^2) + t), \end{aligned} \tag{6}$$

where g is a constant that indicates the importance of the original triplet loss function.

Another approach, as defined by Hoffer & Ailon (2014) in the original paper that introduces the triplet networks, is using SoftMax. With the usage of SoftMax we obtain a ratio measure between the positive and negative distances to the anchor. This method reduces the need for an appropriate sampling method for the triplets and, unlike the previous loss functions, it does not have any parameters to be fine tuned. The SoftMax result for the distance between the positive and the anchor is obtained with the following equation:

$$d_p = \frac{e^{\|f(x^a)-f(x^p)\|_2^2}}{e^{\|f(x^a)-f(x^p)\|_2^2} + e^{\|f(x^a)-f(x^n)\|_2^2}}, \tag{7}$$

and respectively for the negative to the anchor distance:

$$d_n = \frac{e^{\|f(x^a)-f(x^n)\|_2^2}}{e^{\|f(x^a)-f(x^p)\|_2^2} + e^{\|f(x^a)-f(x^n)\|_2^2}}, \tag{8}$$

with the final loss function obtaining the mean squared error of the two distances:

$$L(x^a, x^p, x^n) = \|d_p, d_n - 1)\|_2^2 \tag{9}$$

2.2.3. Layer parameter sharing

Considering the triplet network purely as an extension to the Siamese network suggests using full sharing of layer parameters. An alternative approach is using disjoint parameters for the anchor branch (Ponti et al., 2017). They recommend keeping parameters disjoint when the image branches come from different domains (e.g. sketch vs. photo in sketch-based image retrieval); using a hybrid half-shared model when the input domains are similar (e.g. sketch and image's edgemap) and sharing parameters fully when the domains are close such as face recognition. We report results for triplet nets with both shared and disjoint weights.

2.2.4. Normalisation

Input normalisation. Throughout the body of research on neural networks, various normalisation approaches have been proposed to introduce improvements in training. (Sola & Sevilla, 1997) suggest that input data normalisation is an important technique to ensure good training results and fast convergence. They propose normalising the inputs to three different intervals: (0.1, 0.9), (0.05, 0.95) and (0, 1), and show comparable results for all of them. Subsequently, we consider input normalisation to (0, 1) in some of our further experiments.

Local response normalisation. Local response normalisation (LRN) is an approach introduced in Krizhevsky et al. (2012) and applied in their ImageNet model. Authors postulate that CNNs using ReLUs (Nair & Hinton, 2010) do not require input normalisation but can benefit from this local normalisation scheme to improve the model's generalisation capacity. LRN is defined as follows:

$$b^i_{x,y} = a^i_{x,y} / \left(k + \alpha \sum_{j=max(0,i-n/2)}^{min(N-1,i+n/2)} (a^j_{x,y})^2\right)^\beta \tag{10}$$

where $a^j_{x,y}$ is the activation of a neuron at position (x, y) after applying kernel i and then applying ReLU, $b^j_{x,y}$ is that same value after response normalisation, and the sum runs over n "adjacent" kernel maps at the same spatial position and N is the number of kernels in the layer. The default hyper-parameter values here are $k = 2, n = 5, \alpha = 10^{-4}, \beta = 0.75$. LRN is inspired by biological mechanisms of local inhibition; it diminishes responses that are uniformly large in a given neighbourhood, while making large local activations more pronounced. Although ImageNet applies LRN just after ReLU, FaceNet (Schroff et al., 2015) features LRN layers after pooling layers. As FaceNet is closer in architecture to our model, we will consider FaceNet's placement after pooling layers in our further experiments.

Batch normalisation. Introduced in (Ioffe & Szegedy, 2015), batch normalisation was suggested by the authors as an alternative to LRN, and as a result, has largely replaced the usage of LRN in CNNs. Due to FaceNet's successful use of LRN, we believed it is noteworthy to compare these two competing normalisation techniques to validate batch normalisation's superiority.

3. Methodology

3.1. Training, validation and evaluation process

Having to sample triplets from a data sample of 32460 from the training set pushes the possible examples to $O(n^3)$. As with the previous experiments with the Siamese network (G81, 2018), because of computational and time limitations it makes it infeasible to use all possible triplets per epoch. Additionally, having all possible triplets is redundant as a triplet network should easily be able to map most trivial triplets (Schroff et al., 2015). We have set the same training settings as with the Siamese network (G81, 2018), being for each epoch we have 500 iterations with each iteration performing gradient descent on a batch of 50 triplets. And so a total of 25,000 triplets are iterated over for each epoch. We report validation accuracy on the verification task (of size 10,000 triplets) and a one-shot validation accuracy (320 one-shot tasks) after each epoch. Unless specified otherwise, all experiment were conducted for a duration of 50 epochs.

Our anchor image is sampled each time uniformly to maintain equal representation. While for the positive and negative images, as mentioned previously (2.2.1), we can sample using a random method or by hard positive/negative mining when training the network. When validating or testing, the two branches are always sampled randomly to not skew the results. For our base model, we initially follow a random selection of the positive and negative images. Here, we define the closeness of the images to be whether they are from the same class or not. The positive image is randomly selected from the remaining images from the same class as the anchor. While the negative image, its class and within example are selected randomly, with an exclusion from selecting the anchor class. Additionally, we explore an online approach in hard positive and semi-hard negative mining of images. Because we have a large number of available classes, we define a minibatch (hence semi-hard) to limit our scope by randomly selecting one of the classes (except the anchor class) for the negative branch before finding the most similar image to the anchor. While for the positive branch, the batch is defined to be the same as the anchor class (excluding the anchor image) to find the most dissimilar image against the anchor image. When comparing the

similarity of images we defined how close one is to another by computing their structural similarity. This uses a sliding window with a Gaussian kernel, as defined by Wang et al. (2004).

Wanting to maintain the same 20-way within-alphabet classification evaluation method used in the previous paper (G81, 2018) and defined by Lake et al. (2015) (*given a test image x and N labelled example images of previously unseen classes, we want to classify x into the correct character class by pairing it with each example and predicting class attribute according to maximum similarity*), we adapted the way we pass triplet samples to our network. We first select an alphabet from the evaluation set, as well as 21 characters from it (uniformly and randomly), where one of the characters is repeated twice, making it the anchor image whose target class needs to be predicted. We then set as our negative images the remaining 20 character examples (which one is a positive image). The distance of the positive to the anchor branch will be disregarded for the one-shot task and so any image can be used, which for simplicity we have just set the positive image the same as the anchor image. This is done to maintain a strict similarity between our previous sampling method for the siamese network (G81, 2018) so the two models can be easily comparable. To consider a one-shot task successful, we must obtain the smallest distance between the anchor image and the single positive image in the negative branch.

The general themes explored in our experiments were: different activation functions, layer parameter sharing approaches, loss functions, regularisation and normalisation schemes. Due to constrained access to computational resources, we performed these strands of research in parallel to optimally utilise available compute power instead of following the common coordinate descent approach of sequentially building on the results of experiments on different aspects. Unless specified otherwise, experiments described in Section 4 build on top of the architecture outlined in Section 3.2.

3.2. Model implementation

The triplet model implemented is based on the Siamese network built by Koch et al. (2015) and our adaptation of it in the previous report (G81, 2018). The architecture used consists of a total of seven layers, excluding the output layer and input layer, which takes an image of size 105x105 pixels. The first five layers are mirrored as part of each of the three branches, of which the first four are convolutional layers, followed by a fully connected layer that flattens the output of the previous layer. The convolutional layers have a filter size of 10x10, 7x7, 4x4 and 4x4 respectively and a stride of 1. Each of the branches is then connected to a fully connected dense layer of size 4096, which flattens out the output. Additionally, max pooling was added after each of the three first convolutional layers to reduce the size, which each have a size of 2 and a stride of 2. The first five layers use a rectified linear unit (ReLu) for our base model. The sixth layer is a simple non-trainable L1 norm layer that finds the distances between the embeddings of the anchor image against the positive and the negative, merging the branches of the network from three to two. Finally, the last layer is a non-trainable L2 norm layer which calculates the metric distance for each of the branches, which summarises the output to a size of 1. The weights are then updated through backpropagation based on the defined loss function, which takes as input the calculated distances of the positive and negative branches against the anchor. As a default setting, we incorporated Kumar's triplet loss function.

The weight and bias initialisation of the network is maintained the same as that recommended by Koch et al. (2015), as indicated by our previous experiments to be optimal for this task (G81, 2018). Specifically, the weights for all of the trainable layers were set from a normal distribution with a mean of 0.0 and standard deviation of 0.002, with an exception to the dense layer which was initialised with a mean of 0.0 and standard deviation of 0.2. Also, the bias was randomly initialised with a mean of 0.5 and standard deviation of 0.01. Additionally, we maintain the same L2 regularisation within the layers are proposed by Koch et al. (2015) with a value of 0.0002 for the convolutional layers and for the fully-connected layer we use a value of 0.001. Finally, for the learning rule we, again based on our previous experiments (G81, 2018), use Adam learning rule (Kingma & Ba, 2014). However, for the learning rate we determined that a value of 0.0001 is more suitable based on experiments we conducted, rather than the previous value of 0.00006.

4. Experimental results

4.1. Activations

We decided to start by experimenting with different convolutional layer activation functions on top of our default model: a triplet CNN with learning rate 0.0001, triplet loss, fully shared weights, random sampling and a ReLU activation. We report one-shot validation accuracy of 74.375% for ReLU, 75% for SELU and 76.25% for ELU. Later on, when we combined ELU with different loss functions, it became apparent that it underperforms ReLU significantly: ReLU with global loss and triplet loss achieved 90.625% accuracy vs. 87.1875% for ELU; ReLU with just global loss performed at 90.625% vs. 88.4375% for ELU. In addition ReLU with our best dropout rate of 0.1 beats ELU by a margin of 3% (79.480% vs 76.256%). Therefore we decided to proceed using ReLU as default activation function from now on.

4.2. Layer parameter sharing

To validate which layer parameter sharing approach is better, we compared a homogeneous model that used shared parameters for all three branches (anchor, positive, negative; labelled as *full share* in Figure 3), as well as one using a disjoint parameter set for the anchor branch, and shared parameters between the negative and positive branches (*disjoint anchor*). Our experimental results show a huge dif-

ference between models using these two parameter sharing approaches in terms of their one-shot validation accuracy. Fully sharing parameters clearly helps in performance from the very beginning, with the the *disjoint anchor* approach achieving a disappointing 9.06% best one-shot validation accuracy against 74.375% accuracy of the *full share* model.

Indeed, this was somewhat expected as the three branches considered inputs from a broadly similar general category of written characters, and the relative similarity between them justifies mirroring the parameters in the triplet network. As noted in Ponti et al. (2017), it is more common to use a heterogeneous network architecture to learn embeddings in cases where inputs to each of the branches originate from different domains, such as one being a sketch of a duck and another being a photo of it. In addition, Bui et al. (2016) note that achieving convergence is more difficult in the *disjoint anchor* architecture compared to the *full share* case. We believe that the issue here is not with problematic convergence but simply due to the former architecture not being obviously applicable to the problem at hand. Informed by this result, all of our further experiments utilise a homogeneous triplet network architecture.

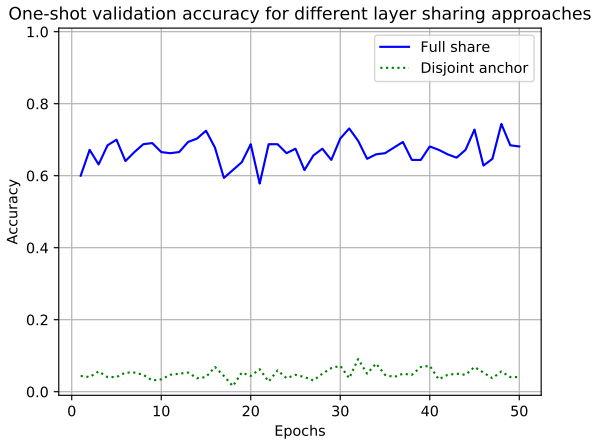

Figure 3. One-shot validation accuracy for different layer sharing approaches.

For our disjoint model, we observed that the value of the loss function was decreasing (results omitted) with, however, the one-shot validation maintaining low accuracy. We can hypothesise that there may be also an underlying issue of vanishing gradients, as suggested by Bui et al. (2016). The triplet loss function in combination with the disjoint network, minimised the loss by pushing the two distances closer together rather than reducing the positive and increasing the negative. They presume that this happens when the distances of the positive and negative are naturally close.

To check the validity of this hypothesis We also compared the disjoint anchor approach with a hard positive sampling method for 15 epochs, because of the computationally demanding usage of hard mining, as seen in Figure 4. The results obtained when using hard sampling have improved drastically, with a one-shot validation accuracy of 73.44%.

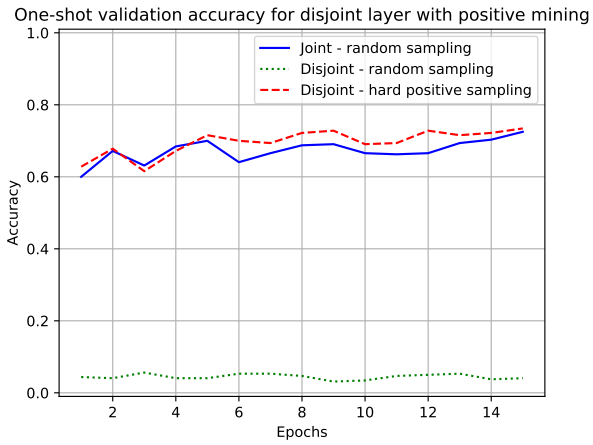

Figure 4. One-shot validation accuracy for disjoint with random and hard positive sampling against joint random sampling.

This method outperforms, at least for the first 15 epochs, our joint network that uses random sampling which obtains a 72.5% accuracy on the one-shot validation task.

4.3. Sampling

To investigate which input sampling method is the most appropriate, we compared random sampling, hard positive mining and hard negative mining. Using hard positive mining results in the best one-shot validation accuracy, achieving its maximum value of 77.18% at epoch 49. Interestingly, hard negative mining results in worse performance compared to random sampling, with hard negative mining achieving 73.75% and random sampling attaining 74.38% one-shot validation accuracy.

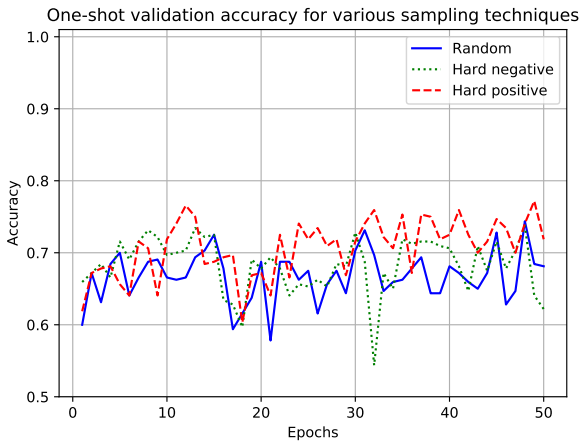

Figure 5. One-shot validation accuracy for various sampling approaches.

Looking at Figure 5, one might notice that there is an initial pattern of better performance of hard negative mining compared to random sampling up until around epoch 15, and subsequent decline in accuracy. This unexpected worse performance of the hard negative mining model can perhaps be attributed to optimisation being stuck in a suboptimal

area. This claim was also given by Schroff et al. (2015), that doing an aggressive hard-mining can lead to a bad local minimum at the early stages of training. In terms of the learning curves, it is not very clear that hard negative mining and hard positive mining actually contribute to faster convergence. As such, we were unable to confirm our expectations guided by claims from Wu et al. (2017). Still, one can argue that hard positive mining did in fact result in more informative triplets being used as inputs to training, as there is almost 3 percentage point increase in one-shot validation accuracy compared to random sampling.

4.4. Loss functions

For the following experiments, we set all the parameters of the loss functions to the ones used by Kumar et al. (2016). The values are as following: Kumar and Schroff's triplet losses $m = 0.01$, global loss $l = 0.8$ and $t = 0.4$ and for global plus triplet loss $m = 0.01$, $l = 0.8$, $t = 0.4$ and $g = 1.0$. The best performance of the losses are summarised in Table 1.

Our experiments with different loss functions contradict Bui et al. (2016)'s claim that using softmax loss is necessary to achieve convergence when training. In fact, the model with softmax loss performed the worst; this could potentially be explained by noticing that this approach reduces the classification task to a 2-class problem, resulting in unsatisfactory generalisation capacity of one-shot learning. Models using Kumar and Schroff's triplet losses achieve better one-shot validation accuracy but clearly get stuck in optimisation, with their learning curves oscillating within certain bounds throughout the whole training period. On the other hand, models based on global loss surpass them at around epoch 9-10 and continue to show upward trajectory in one-shot validation accuracy up until the end of training.

Comparing against the model using hard positive mining identified in Section 4.3, we confirm that using global loss not only makes it no longer necessary to employ a tailored sampling approach but also that a suitable loss function can considerably improve performance without the additional computational cost of non-random sampling. These results conform the findings by Kumar et al. (2016), that using global or global plus triplet loss allows for better convergence against other functions and more reliably, specifically against softmax.

Model	One shot accuracy
Hard positive mining	77.18 %
Kumar loss	74.38 %
Schroff loss	75.00 %
Softmax loss	7.82 %
Global loss	90.63 %
Global + triplet loss	90.63 %

Table 1. Comparison of models using different loss functions and the best sampling method – hard positive mining, as evaluated on one-shot validation accuracy.

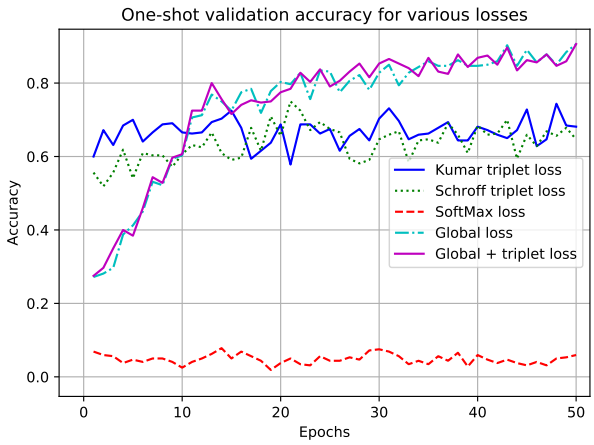

Figure 6. One-shot validation accuracy for various loss functions.

4.5. Normalisation and regularisation

Additionally, we decided to investigate how different kinds of layer normalisation methods affect accuracy (See Figure 7). All of the models below are also based on the configuration from subsection 3.2, whose one-shot validation accuracy was 74.38%. We start off by applying input normalisation (73.44%) and batch normalisation after each pooling layer (72.50%). Combining input normalisation with batch normalisation as described above yields 67.50% one-shot validation accuracy. Thus, neither of these combinations of normalisation approaches with our default model improve the model's performance but only decrease it. Applying local response normalisation to the default model configuration scores 74.38% one-shot validation accuracy, and so does not yield any improvements over the default model. When LRN is combined with input normalisation it performs slightly worse (73.44%) but at the same level as with just input normalisation. Relating all these results back to claims from 2.2.4, we could confirm Krizhevsky et al. (2012)'s proposition that LRN outperforms input normalisation when both are used independently; but when both are combined, this fares worse than just using LRN. On the other hand, using batch normalisation proved to not only underperform against LRN but also against the default model without batch normalisation. Our empirical results show that using LRN is more appropriate than batch normalisation, with the former also being the preferred choice of normalisation scheme for FaceNet (Schroff et al., 2015). Still, none of the normalisation schemes explored actually improved our default model performance – the best one (LRN) only performed equally well as the default model. As model parsimony is a generally preferred quality, we decided not to employ any normalisation in our final model.

We also experimented with various dropout rates on top of our default configuration from Section 3.2. We report one-shot validation accuracy for the following rates: 0.1 (79.06%), 0.3 (77.19%), 0.5 (76.88%), 0.7 (66.25%), 0.9 (50.94%). We then applied our best dropout rate (0.1) on the default model with global loss (88.75%) and with global

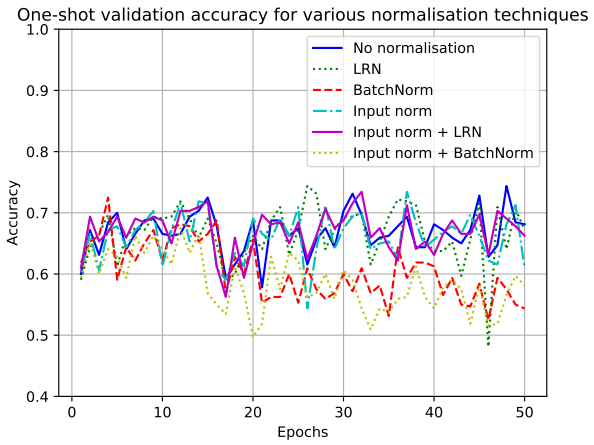

Figure 7. One-shot validation accuracy for various normalisation schemes.

+ triplet loss (90%). Clearly, adding dropout did not improve the performance of our models trained with global loss and global + triplet loss (cf. Table 1), which is why we did not apply dropout in our final model. Although in general featuring dropout can help combat the effects of overfitting and consequently improve the model's generalisation capacity, we could not demonstrate its benefits in the context of our triplet network.

5. Test results

Our final triplet CNN differed from the default model outlined in Section 3.2 only by the use of global loss instead of Kumar's triplet loss. We favoured global against global plus triplet loss, because it resulted in overall better performance from epoch to epoch. We compared this model's one-shot accuracy on the evaluation set against the baseline siamese network developed in G81 (2018) (see Table 2). Both of our siamese CNN and triplet CNN perform worse on the evaluation set, as could be expected given the fact that the evaluation set comprises completely novel alphabets, unseen by either of the models. However, the gap between validation and evaluation one-shot accuracy is much smaller for the triplet CNN than the siamese CNN. It could be argued that this signifies the triplet CNN's better generalisation capacity between these two datasets, and specifically, better ability in one shot learning.

Model	Validation Acc	Evaluation Acc
Siamese CNN	87.81 %	84.50 %
Triplet CNN	90.31 %	89.25 %

Table 2. Comparison of our Siamese CNN baseline with the final Triplet CNN, as evaluated on validation and evaluation set one shot classification accuracy.

6. Conclusions

Through our empirical results, we demonstrated that triplet CNNs can be successfully applied to one-shot learning for Omniglot, and that they have a potential to outperform siamese CNNs. Unfortunately, our triplet CNN was not able to surpass Koch et al. (2015)'s siamese CNN's one-shot accuracy for a similarly sized dataset size. Given that Koch et al. (2015) were able to train for a longer duration (200 epochs and not 50 as in our experiments with triplet CNNs), and also perform much more exhaustive Bayesian optimisation for their hyperparameters, we believe our results are still noteworthy.

We postulate that global loss allows for better generalisation than other loss functions. The triplet CNN achieved 1.06% lower accuracy on the evaluation set compared to the validation set, while the siamese network demonstrated a larger difference of 3.31% when using a cross entropy loss function.

7. Future work

To ensure best performance of the global loss function proposed by Kumar et al. (2016), authors suggest finding the mean and variance against all possible triplets. However, because of limited time and GPU memory on our local machines we were constrained to using a relatively small batch size of 50 triplets. It would be interesting for the future, when the resources become available, to perform equivalent experimentation using a set of much larger batch sizes. Based on the above mentioned paper, we should expect to have better convergence and generalisation capabilities and in turn better one-shot accuracy, for both the validation and evaluation set, as the batch size increases.

We have seen that using Kumar's triplet loss function together with hard positive mining (section 4.3) improved results against the default model. However, because of its high computational demand of comparing each image against all others it is not viable to be used. There are suggestions to incorporate hard mining for each batch passed within the loss function. One suggestion by Schroff et al. (2015) is to use a triplet loss with semi-hard negative mining. Another proposed method by Balntas et al. (2016) is using SoftPN loss function which incorporates a soft positive and negative mining. They claim that these methods offer the benefits of having a more refined sampling method without compromising training speed.

A similar natural extension, as we saw between siamese and triplet networks, is to evaluate how quadruplet networks (Chen et al., 2017; Dong et al., 2017) perform on the one-shot task. These types of networks utilise even more information about patterns between anchor-positive and anchor-negative similarities. For example, Chen et al. (2017)'s quadruplet network uses a quadruplet loss that not only aims to minimise positive distances and maximise negative distances but also ensures that all positive distances are smaller than negative distances.

References

Balntas, Vassileios, Johns, Edward, Tang, Lilian, and Mikolajczyk, Krystian. Pn-net: Conjoined triple deep network for learning local image descriptors. *CoRR*, abs/1601.05030, 2016. URL http://arxiv.org/abs/1601.05030.

Bromley, Jane, Guyon, Isabelle, LeCun, Yann, Säckinger, Eduard, and Shah, Roopak. Signature verification using a "siamese" time delay neural network. In Cowan, J. D., Tesauro, G., and Alspector, J. (eds.), *Advances in Neural Information Processing Systems 6*, pp. 737–744. Morgan-Kaufmann, 1994.

Bui, Tu, Ribeiro, Leonardo, Ponti, Moacir, and Collomosse, John. Generalisation and sharing in triplet convnets for sketch based visual search. *arXiv preprint arXiv:1611.05301*, 2016.

Chen, Weihua, Chen, Xiaotang, Zhang, Jianguo, and Huang, Kaiqi. Beyond triplet loss: a deep quadruplet network for person re-identification. *CoRR*, abs/1704.01719, 2017. URL http://arxiv.org/abs/1704.01719.

Dong, Xingping, Shen, Jianbing, and Porikli, Fatih. Quadruplet network with one-shot learning for visual tracking. *CoRR*, abs/1705.07222, 2017. URL http://arxiv.org/abs/1705.07222.

Fei-Fei, Li, Fergus, R., and Perona, P. One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):594–611, April 2006. ISSN 0162-8828. doi: 10.1109/TPAMI.2006.79.

G81. MLP Coursework 3: One-shot learning with Omniglot. 2018.

Hoffer, Elad and Ailon, Nir. Deep metric learning using triplet network. *CoRR*, abs/1412.6622, 2014. URL http://arxiv.org/abs/1412.6622.

Ioffe, Sergey and Szegedy, Christian. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015. URL http://arxiv.org/abs/1502.03167.

Kingma, Diederik P. and Ba, Jimmy. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL http://arxiv.org/abs/1412.6980.

Koch, Gregory, Zemel, Richard, and Salakhutdinov, Ruslan. Siamese neural networks for one-shot image recognition. 2015.

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.

Kumar, BG, Carneiro, Gustavo, Reid, Ian, et al. Learning local image descriptors with deep siamese and triplet convolutional networks by minimising global loss functions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5385–5394, 2016.

Lake, B.M., Salakhutdinov, R, Gross, J, and Tenenbaum, J.B. One shot learning of simple visual concepts. 01 2011.

Lake, Brenden M., Salakhutdinov, Ruslan, and Tenenbaum, Joshua B. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015. ISSN 0036-8075. doi: 10.1126/science.aab3050. URL http://science.sciencemag.org/content/350/6266/1332.

Nair, Vinod and Hinton, Geoffrey E. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pp. 807–814, USA, 2010. Omnipress. ISBN 978-1-60558-907-7. URL http://dl.acm.org/citation.cfm?id=3104322.3104425.

Ponti, Moacir Antonelli, Ribeiro, Leonardo Sampaio Ferraz, Nazare, Tiago Santana, Bui, Tu, and Collomosse, John. Everything you wanted to know about deep learning for computer vision but were afraid to ask. In *2017 30th SIBGRAPI Conference on Graphics, Patterns and Images Tutorials (SIBGRAPI-T)*, pp. 17–41. IEEE, 2017.

Schroff, Florian, Kalenichenko, Dmitry, and Philbin, James. Facenet: A unified embedding for face recognition and clustering. *CoRR*, abs/1503.03832, 2015. URL http://arxiv.org/abs/1503.03832.

Schultz, Matthew and Joachims, Thorsten. Learning a distance metric from relative comparisons. In Thrun, S., Saul, L. K., and Schölkopf, B. (eds.), *Advances in Neural Information Processing Systems 16*, pp. 41–48. MIT Press, 2004. URL http://papers.nips.cc/paper/2366-learning-a-distance-metric-from-relative-comparisons.pdf.

Sola, J. and Sevilla, J. Importance of input data normalization for the application of neural networks to complex industrial problems. *IEEE Transactions on Nuclear Science*, 44(3):1464–1468, Jun 1997. ISSN 0018-9499. doi: 10.1109/23.589532.

Wang, Jiang, Song, Yang, Leung, Thomas, Rosenberg, Chuck, Wang, Jingbin, Philbin, James, Chen, Bo, and Wu, Ying. Learning fine-grained image similarity with deep ranking. *CoRR*, abs/1404.4661, 2014. URL http://arxiv.org/abs/1404.4661.

Wang, Zhou, Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, April 2004. ISSN 1057-7149. doi: 10.1109/TIP.2003.819861.

Wu, Chao-Yuan, Manmatha, R., Smola, Alexander J., and Krähenbühl, Philipp. Sampling matters in deep embedding learning. *CoRR*, abs/1706.07567, 2017. URL http://arxiv.org/abs/1706.07567.