Summative Assessment 1 Accompanying Report

Introduction

The data pipeline and model I have ended up submitting is one of many, many iterations. I experimented with all sorts of loading methods and model architectures. Towards the end of this process, I realised that my decisions were often driven by the engineering practicality of the solution rather than the (validation) performance of the method itself, as poor performance of the former prevented assessment of the latter.

Data Processing and Model Walkthrough

Data Loading

The data loading process begins with the function `build_label_to_filepaths`. This creates a mapping between the identity labels and the filepaths to their images. This function is important because it is far more efficeint to work with the address of a large object than the object itself. I found that out the hard way... After this, two helper functions are defined, `decode_and_resize` and `load_decode_batch`, which pretty much do what their names suggest. The actual data loading logic occurs in the following function `build_triplet_compatible_dataset`, which has some important features: the output of the function is a `_PrefetchDataset` object, allowing for more efficient memory utilisation and some mitigation of I/O bottlenecks; the function picks a specified number of identities (at random) and subsequently samples a specified number of (random) images from this identity. This random sampling method is a little different to the common approach of batching using the entire dataset available, and was chosen as a balance between optimising I/O efficiency and using a larger dataset.

Circle Loss

The circle loss function is defined as requested in the question, making use of log operations for numerical stability.

Triplet Selection

The triplet selection mechanism was taken from Schroff et al (2015)'s FaceNet paper (https://arxiv.org/pdf/1503.03832). This method involves creating triplets for all anchor-positive combinations for each identity, and selecting the negative according to one of two methods:

- Hard negative mining: where the most similar ("most wrong") negative image is chosen given each triplet's anchor
- Semi-hard negative mining: where the most similar ("most wrong") negative image is chosen given each triplet's anchor **subject to** being at least as dissimilar as a given margin (m from the circle loss).

The FaceNet paper suggests that solely emplopying hard negative mining can lead to degenerate weight distributions, so an approach of initially using semi-hard negative mining (to more softly guide the model toward a good distribution) for a period before switching to hard negative mining (to continually challenge the model) during later training stages.

Model Architecture

The model architecture employed is a tweaked version of the Inception-ResNetV2 model described in the FaceNet paper. This model was transcribed into Keras from the description given in the paper (and reference GitHub pages) and then had the number of contiguous residual blocks reduced in order to create a more lightweight model and streamline training.

Several model architectures were experimented with and tested, but unfortunately due to computational and time constraints, hyperparemeter tuning was only conducted for the final modified Inception-ResNetV2 architecture. Because of this, it is plausible that other candidate architectures (which may be in fact better suited to the task) were initially discarded as a result of unimpressive performance when perhaps they could eventually outperform the final architecture (after hyperparameter tuning etc).

Training Loop

The training loop is structured with metrics and checkpointing in mind. Each training step (batch) is wrapped in a `tf.GradientTape()` context so that gradient descent can be applied after each forward pass. Validation metrics are computed every 10 (or otherwise specified number of) epochs. Checkpoints are saved every epoch.

Hyperparameters

Hyperparameter tuning was done via a gridsearch on γ and m. 10 epochs were run over the product of the ranges $\gamma = [8,16,32,64,128,256]$ and $m = [0.15,0.2,0.25,0.3,0.35]$. It was discovered that high values of γ can lead to computational instability. The hyperparameters $\gamma=16$ and $m=0.15$ were eventually selected. Other hyperparameters, such as batch and triplet selection hyperparameters, dropout rate and embedding dimension, were experimented with in a more ad-hoc fashion, but were not analysed rigorously.

Under greater computational facility and with more time, a wider and more granular range of hyperparameters should be tested, including hyperparameters not considered in the gridsearch, perhaps using a more intelligent optimisation algorithm such as Bayesian optimisation. As well as this, the computational issues mentioned regarding γ should be ironed out.

Next Steps and potential improvements

Methodological improvements:

- More rigorous hyperparam search, in tandem with in-depth analysis of model architecture decisions
- Perhaps experimenting with different loss functions
- Further experimentation with data augmentation (crops, flips, zooms, brightness/contrast)
- Considering implementing a "bounding box" face identification step on training and test data to help standardise inputs
- Experimentation with training the network as a binary classifier in the first place (i.e., a two-input model with a fully connected final layer which does the classification task)

Computational improvements:

- Numerical stability in the loss function (trivially solved by allowing greater floating point precision, enabled by a RAM upgrade. Can also be solved via cleverer transformations of the data e.g., working in the log domain as much as possible)

- Larger batch sizes (enabled by a RAM upgrade)
- Deeper architecture (enabled by a RAM and processor upgrade)
- Longer training time with adaptive (self-tuning) hyperparameters,based on given loss and validation metrics (enabled by a processor upgrade and longer time constraint)

References:

Schroff et al (2015) FaceNet: (https://arxiv.org/pdf/1503.03832)

Sun et al (2020) Circle Loss: (https://openaccess.thecvf.com/content_CVPR_2020/papers/Sun_Circle_Loss_A_Unified_Perspective_of_Pair_Similarity_Optimization_CVPR_2020_paper.pdf)