Week 10 Practical

Dr Simon Denman
CAB420: Machine Learning

This weeks practical will focus on multi-task learning. It builds on existing lecture examples and practical examples, and explores how we can add different tasks to networks and how this can impact performance. In this practical, you are encouraged to work off the suggested lecture or practical examples to simplify the implementation of your solutions.

For both questions, we are continuing to explore the problem of face recognition. In previous weeks we've seen that different methods generalise to different extents. This week, we'll explore if multi-task learning can help improve generalisation.

NOTE: Don't expect adding extra tasks to the network to suddenly make everything work perfectly, and don't be surprised if things get worse. These questions really aren't aimed at trying to make the most accurate face recognition system possible, but to explore multi-task learning. With that in mind, you may try things that make sense, and don't work. You may also find that parameters such as loss weights have a huge impact on performance. Don't focus too much on the overall accuracy, but instead explore how doing different things changes the accuracy relative to other networks or parameter settings.

Problem 1. Face Recognition with an AutoEncoder. The Semi-Supervised Learning lecture example (`CAB420_Encoders_and_Decoders_Example_3_Semi_Supervised_Learning.ipynb`) uses a network which is an autoencoder, with a class-based output also included, and resulting from a couple of dense layers connected to the bottleneck. This network allows the autoencoder to class specific information, as the bottleneck layer must be useful for both reconstruction, and classification.

Extend this idea to face recognition. In doing so you should:

1. Use the YaleB dataset to train the model. Look at the Week 7 practical solutions for code to prepare YaleB (and other face recognition datasets) for use in a DCNN.
2. Modify the class-based output of the network to classify the face into the target identities.
3. Use the trained network to extract bottleneck features, and train a simple classifier (i.e. a CKNN) for face recognition. Feel free to reuse code from earlier weeks (i.e. the week 7 practical solution) to achieve this.
4. Evaluate the same network on the ORL and Yale face datasets. Consider how well the network trained on YaleB generalises to these other datasets, and compare this to the results observed in previous practicals that have dealt with this data (Week 6 and 7).

Problem 2. Multi-Task Learning with Siamese Networks. Siamese networks can also be configured for multi-task learning. There are many ways to do this, but a common way is to consider a secondary task on one of the inputs. Using the Week 7 practical solution as a starting point, add a second task to the network. Possible tasks include:

- A classification output, which uses the embedding from one image (such as the anchor image) to classify the face into the identity using a softmax output.
- An image output, that adds a decoder to the embedding to try and reconstruct the original image.

For both of these, you may also wish to consider appropriate loss weights. Any selection of loss weights should consider both the importance of the tasks, and also the scale of the losses (i.e. are both losses typically in the same range? Or is one much larger or smaller than the other?).

Using the modified network, evaluate the face recognition performance on YaleB, ORL and Yale. As with Question 1, use a simple classifier and feel free to borrow from code such as the week 7 practical solutions. Compare the generalisation ability of the multi-task network to the original single task network.