Multitask Learning with Generative Adversarial Networks Research Proposal

Ben Carr

May 29, 2017

Abstract

This document outlines my A.I. MSc dissertation proposal. In a nutshell, the dissertation will explore whether the benefits of multitask learning can be applied to generative adversarial networks. This proposal motivates the project by giving some background theory and suggests why a successful project might be beneficial to the field of machine learning. A completion criteria (including a discussion on evaluation methods) and work plan (including contingencies) is outlined. If the work plan is followed then the completion criteria should be met and the project deemed a success. A discussion on resource-feasibility is also included.

1 Background

Across any domain, there is vastly more unlabelled data available than labelled. Building A.I. systems from only labelled data, as done in supervised learning, is hugely limiting. Hence, unsupervised learning is a powerful tool in advancing A.I. systems. One example of successful unsupervised learning is the development of a human infants brain. A baby isn't told what a vertical line is, or given a label for each object that it can discern (sure, some of those objects may eventually be labelled by the parent but it's a tiny fraction). A huge proportion of the intelligence gained in those early years is through unsupervised learning. Thus, most experts believe unsupervised learning is a crucial step to one day building general artificial intelligence.

1.1 Introducing GANs and WGANs

Generative adversarial networks (GANs) are an exciting new paradigm within unsupervised learning, introduced by Goodfellow et al [4]. This novel approach pits two neural networks in opposition (hence the *adversarial* part of their name). One is called the generator, the other, the discriminator. We will shorthand these networks as G and D respectively. Given a dataset of images, D is trained to classify an image as either $real$ or $fake$ [1]. Whereas G is trained to generate fake images, which 'fool' D into believing are real. G knows how to fool D by using the error gradients of D's loss function when it updates its weights.

The common analogy is that of an art forger and police detective pitted in opposition. The art forger is training to generate realistic fakes that fool the detective, whereas the detective is training to become better at discerning between real paintings and forgeries.

The traditional GAN architecture has been shown numerous times to achieve the sate of the art in image generation [6] [3] [10]. However, one issue with it is the lack of objective measure to understand how well it is doing. Both G and D have loss functions, but since they are in opposition and their losses are negatively correlated, a low loss on either G or D is no indication of image quality.

A recent new development to GANs is the Wasserstein GAN (WGAN) introduced by Arjovsky et al [5]. It overcomes the 'unmeaningful-loss' problem that the original GANs have by adapting the training procedure slightly such that D and G are no longer in opposition (perhaps they should be called WGNs). D instead provides a kind of feedback to G which helps G get *close* to the real, underlying distribution of the data. Here, *close* refers to a distance over a metric (more precisely an approximation of a metric), the Wasserstein metric, otherwise known as the

[1] From this point on we will only consider the use of GANs in image processing but they can be applied across many of the classic machine learning domains (thus far, NLP being a big exception).

Earth Mover metric. Intuitively, this metric is the minimal amount of work needed to shape one distribution into another. For vanilla GANs, there is no clear metric that is being minimised over, that is, a metric between G's distribution and the real distribution. Theory suggests it is a quasi JS-divergence but it is not clear [4].

So, the loss of the generator in a WGAN is the earth mover distance between G's distribution and the real distribution that the images come from. This means we have a quantifiable way of showing improved image quality and model performance. WGANs have shown to generate images of qualitatively comparable, or even better quality than GANs and their training has been shown to be much more stable [5]. For these reasons, we will try to solely use WGANs for this project.

Another important development within GANs that we will rely on for this project, is that of conditional GANs [8]. This adaptation of the architecture incorporates the class information of the images into the training process. This allows the GAN to learn a class boundary across the multi-modal distribution of G and D. Thus, G knows how to generate a certain class just as D knows how to discern whether a certain class of image is real or not. There are a few ways to incorporate this class information into the G of a GAN, as shown by Mirza et al [8]. But so far, I haven't seen any literature on a way of incorporating this class information into the discriminator of a WGAN. That is, there is no 'conditional WGAN' paper, like Mirza et al's, as of yet. This is to be expected due to the recency of the seminal WGAN paper. Furthermore, there have been multiple success stories on ML forums, so I don't believe this will be a problem [7]. If it does turn out to be too challenging, the original GAN architecture will have to be used and model evaluation will have to rely on other means than loss scores, e.g using Amazon Mechanical Turk to survey image quality using human evaluation or using the learned features of the GAN to train a classifier on top of which can be empirically evaluated.

By using conditional WGANs this is no longer true unsupervised learning. However, GAN development is usually framed within the context of unsupervised learning and the significance of a positive result in this project could extend to other types of MTL methods which aren't reliant on labelling. Thus, I feel the best context to situate this work within is still unsupervised learning.

1.2 Introducing Multitask Learning

One issue that all neural network architectures can face is overfitting. This applies to generative models as well. The model can effectively memorise the images and not find the more general features we are after. To counteract overfitting in neural networks, many regularisation techniques have been discovered, e.g, dropout [11] or data augmentation. These can theoretically be applied to GANs to help them find more general representations of the data. This paper focuses on applying an interesting regularisation technique, *multitask learning* (MTL), to the WGAN architecture.

MTL is the technique of training a single model on multiple, related tasks in parallel (a thorough treatment is given in Caruana's PhD thesis [1]). By learning in this way, the model can learn more 'task-general' features, as opposed to 'task-specific' features. For neural networks this means that a certain subset of the weights are shared. The hope is that these shared weights capture more general representations of the data. There are many conceivable ways of finding multiple, related tasks for an image generation model. For example, one could define a model that defines tasks on multiple datasets. For simplicities sake, we will apply a very simple form of MTL, defining only two tasks on a single dataset. In terms of architecture, it makes sense to share these weights within a subset of a networks layers. Since the inputs are the same for each task but the outputs are different, it makes sense to consider an architecture where the first l layers are shared where after the network bifurcates, splits in two, and gives two branches with different outputs. The two tasks will differ in the type of labels that are associated to each image. One type of labelling will be more precise, e.g a *terrior*, whereas the other type will be more coarse, e.g a *dog*. Each of the more precise labels will belong to a single coarse label. Hence, we refer to the two types as *classes* and *super-classes* respectively. The two tasks that they correspond to will be referred to as the *main-task* and *auxiliary-task* respectively. This exact type of task selection (two tasks with class hierarchies) has been shown to work very successfully on a variety of image datasets for classification by Teterwak et al [2].

1.3 Bringing MTL and WGANs together

The main goal of this project is to see if the benefits of MTL can be applied to conditional WGANs (giving us a MTL-WGAN). To do this, G and D need to be defined for both tasks. Both G and

D need to share layers and bifurcate at some point. For simplicities sake, the bifurcation point l will be kept the same for both G and D. Training will be carried out by alternating randomly between the tasks, training on a batch at a time. The effects of the choice of l will be explored and compared to a baseline conditional WGAN which doesn't have MTL applied to it. These comparisons will be made through the loss of the generator for the main task. The best l will be selected and used to define a best MTL-WGAN model. A deeper evaluation will then be carried out on this model. Resource permitting, this will include a qualitative analysis via Amazon Turk along with an indirect evaluation via a classification task trained on the top of the pretrained features of the MTL-WGAN. This classification task evaluation could be carried out on a different dataset to gain a better understanding of how general the features are. Another important step is to show that the network is not memorising samples from the data, a nearest neighbour analysis will be done on generated samples.

Taking lead from Teterwak et al's paper, other results that would be interesting (again, resource permitting): the effect of shrinking the dataset size and the effect of changing the hierarchy structure of the classes. By staying close to the approaches of Teterwak et al, a positive result in this work which agreed with their results would not only strengthen their conclusions but would indicate that the benefits of MTL are shared across both supervised and unsupervised learning.

On this note, many aspects of that paper will be followed. This includes the choice of dataset to work on. CIFAR was chosen in particular, over the other datasets used in that paper, because the image size is smaller (32 by 32) so easier to compute with, and there is a natural hierarchy of classes given to us with the dataset so the task selection is a natural one [12]. Another reason is that I already have worked with CIFAR and have it easily accessible.

1.4 Project feasibility

An important, but not critical, step to successfully completing this project is to show MTL working on classification on CIFAR. Although the end goal is not classification, it is an important step to show that my implementation of MTL is correct and that the main-tasks and auxiliary-tasks are an appropriate choice. During the machine learning practical course this semester I've implemented both a WGAN architecture and a MTL classification architecture on CIFAR. The WGAN architecture has worked well, generating reasonably high quality images and loss curves. However, the MTL architecture hasn't shown any improvement over the baseline. This has been discussed with my project supervisor (Peter Bell) and a few approaches were advised, number one being to strip back the baselines complexity and do a simpler comparison since the current baseline might be overfitting to the evaluation set. Another thing mentioned was the fact that a negative result does not necessarily mean that the features learned are not more general. Thus, a deep evaluation of the features of the MTL-WGAN is necessary to understand if MTL has worked.

None of the architectures that have been discussed (here or in the referenced literature) are massively deep. Therefore I think that for most experiments my NVIDIA 1070 GPU should suffice. The longest experiment I have had so far is training a vanilla WGAN, this took 10 hours and appeared close to convergence. If this does become a problem there is the Edinburgh Uni MSc cluster or cheap services offered by Amazon, Google etc. which can be fallen back on for extra compute. These resources are either free or can be personally financed by myself.

Using the Amazon Mechanical Turk service to obtain human-evaluated results has its pitfalls (need lots of data, very variable based on the wording of the task as presented to the human), it would however be a good addition to have for the model evaluation. This is because different evaluations of generative models have been shown (for example by Theis et al. [14]) to often be independent of each other. A ballpark figure for the pricing for Mechanical Turk is ≈ 1000 labels per pound and for time spent, ≈ 1000 labels per hour [13]. This evaluation method could be costly in both price, time and effort. It depends on how many labels are needed to gather meaningful results (it's hard to say at this point the number of labels required, it will require analysis). This evaluation is not critical to the project. Thus, it will be dropped if there are constraints.

There are no other obvious resource constraints, all software (tensorflow, PyCharm etc.) is open source and the data is free to use.

Due to the precursor work I have done, the clear structure of the project that has been outlined and the flexibility in the completion criteria (as described in the next section), I believe that this project can be feasibly achieved by myself within 12 weeks.

2 Completion Criteria

- Presentation of a working implementation of MTL-WGAN as described above. This will require a thorough explanation of the background theory, methods used and evidence that the model is training and MTL is implemented correctly. The relationship between the improving loss and generated image quality should be displayed.
- G-loss comparison of different bifurcation points for MTL-WGAN against the G-loss of a WGAN baseline, testing for statistical significance. A 'best' MTL-WGAN should be defined. A table of results along with a discussion should be displayed.
- A nearest neighbour analysis on the best MTL-WGAN to show the model isn't memorising samples.
- (if time) Use the pre-trained features of the best MTL-WGAN to train a classification model and compare the test score performance of that model against an appropriate baseline. This could possibly be done on a dataset other than CIFAR.
- (if time and money) Compare the best MTL-WGAN against the baseline WGAN with an Amazon Mechanical Turk survey to get a human-evaluated measure of performance.
- (if time) Explore the effect of changing the super-class hierarchy on the best MTL-WGAN to see if the results are consistent with Teterwak et al.
- (if time) Explore the effect of shrinking the dataset size on the best MTL-WGAN to see if the results are consistent with Teterwak et al.

3 Planning

Here is an outline of the main milestones required to reach the completion criteria. Contingencies for potential problems are discussed in the bullet points.

1. Get working environment organised (ensure tensorflow working with GPU, git repo set up, etc.)
2. Set up and train a classification baseline then evaluate its performance (this baseline could match the discriminator since showing MTL working on the same architecture is a good sign it'll work for MTL-WGAN).
3. Set up and train a MTL classifier that matches the baseline defined in 2. Evaluate this MTL architecture on CIFAR. Ideally, a statistically significant improvement over the baseline would be shown. This would show the feasibility of the task selection and give evidence that the implementation is correct.
   - As discussed in section 1.4, there is a significant risk here that a positive result for this step might not be found. I would rate this step as the one with the highest risk. There are a variety things to try: hyperparameter tuning, attempting to reproduce the exact results from Teterwak et al. [2] or even trying it on an entirely new dataset. However, even if no benefits of MTL for classification are found at this step, it could be because of the way we are evaluating the model. MTL might in fact be working well and finding much more general features that don't beat this particular baseline which could have been overfitted to the evaluation set. Thus, a positive result here is not crucial.
4. Set up and train a WGAN on CIFAR (generating qualitatively good images and with a sensible looking loss quality, show the correspondence between loss curve and image quality). This will ensure the implementation is correct and will hopefully give evidence that the loss curve is indicative of image quality.
5. As in the last step, set up and train a conditional WGAN on CIFAR. This is the WGAN baseline.

- As discussed in section 1.1, there is only literature on doing this with conditional GANs. However, there are success stories on forums where source code has been shared [7]. If this step is too challenging, a clear way of mitigating the issue is to just use vanilla GANs instead of WGANs. This would require we drop the G-loss evaluations and focus on the other evaluation methods discussed. I would rank this step as having the second highest risk.

6. Set up MTL-WGAN architecture, show it training on both tasks. Ensure implementation is correct by running sanity checks on weight distributions, activation scarcities, etc.

    - This is an essential step of the project so there are no real contingency plans. If the MTL-WGAN is not training then it is very likely due to implementation problems and so the implementation should be checked and tested repeatedly until it works.

7. Evaluate the MTL-WGAN G-loss with different bifurcation points. Compare these against the baseline WGAN. Test for statistical significance of results. Define a 'best' MTL-WGAN.

8. Test that the best MTL-WGAN is not memorising samples by performing a nearest neighbour search between generated samples from G and CIFAR.

    - I have never performed an analysis of this sort before. Saying this, there are plenty of open source algorithms to achieve this and the theory is simple so I do not think there is much chance of failure at this step. Worst case and it is too challenging to perform, this analysis will be omitted from the dissertation. Then, references to the literature will have to suffice as argument that the network is not memorising.

9. Re-evaluate the best MTL-WGAN against the baseline WGAN by using the Amazon Mechanical Turk service. This will shed more light into whether MTL is in fact working.

    - Although I have never done this before, this step looks fairly straightforward to set up. Time and money could be constraints here. It could potentially take too long or be too expensive for the survey to be completed since a lot of samples must be used to get a meaningful result. Anyhow, this step is not crucial and can be omitted if found to be too challenging or expensive (in terms of time or money).

10. Re-evaluate the best MTL-WGAN against the baseline WGAN by training a classification model on top of their features and obtaining test performance results. This could possibly be done on other datasets than CIFAR. This will shed more light into whether MTL is in fact working.

    - Although I have never done this before, this step looks fairly trivial using tensorflows *save* and *restore* methods. Anyhow, this step is not crucial and can be omitted if found to be too challenging or take too long.

11. Evaluate the G-loss for the MTL-WGAN with different class hierarchies. These results would be nice to have to compare against Teterwak et al's related results [2].

    - Time constraints might be a problem. For each hierarchy (there will be ≈ 3), the baseline must be retrained. This means a total of 6 MTL-WGAN/WGAN experiments (≈ 120 hours). Anyhow, this step is not crucial and can be omitted if found to be too challenging or take too long.

12. Evaluate the G-loss for the MTL-WGAN with differing sizes of training data. These results would be nice to have to compare against Teterwak et al's related results [2].

    - Again, the time constraint might be a problem. For each different dataset size (≈ 4), the baseline must be retrained. This means a total of 8 MTL-WGAN/WGAN experiments (≈ 160 hours). Anyhow, this step is not crucial and can be omitted if found to be too challenging or take too long.

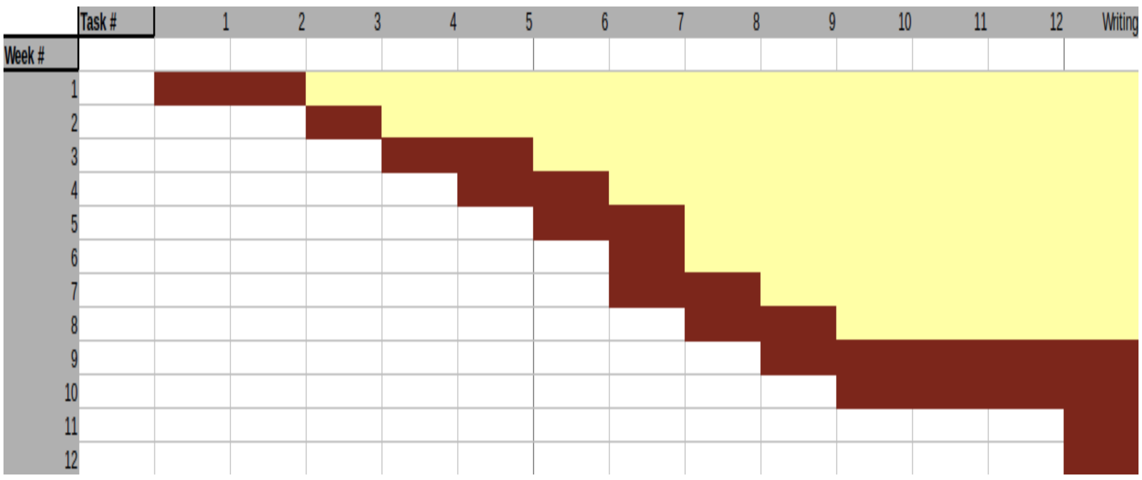

Figure 1: *A rough task schedule. Yellow for low priority and red for high priority. Week 1 starts on Friday the 26th of May and week 12 ends on Friday the 18th of August.*

References

[1] Caruana: Multitask Learning
`http://www.cs.cornell.edu/ caruana/mlj97.pdf`

[2] Teterwak & Torresani: Shared Roots: Regularizing Neural Networks through Multitask Learning,
`http://www.cs.dartmouth.edu/reports/TR2014-762.pdf`

[3] Alec Radford & Luke Metz: Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks,
`https://arxiv.org/pdf/1511.06434.pdf`

[4] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio: Generative Adversarial Nets,
`https://arxiv.org/pdf/1406.2661.pdf`

[5] Martin Arjovsky, Soumith Chintala and Leon Bottou: Wasserstein
`https://arxiv.org/pdf/1701.07875.pdf`

[6] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung: Improved Techniques for training GANs
`https://arxiv.org/pdf/1606.03498.pdf`

[7] Conditional WGAN discussion with link to source code [13/04/17]:
`https://github.com/martinarjovsky/WassersteinGAN/issues/16`

[8] Mehdi Mirza, Simon Osindero: Conditional Generative Adversarial Nets
`https://arxiv.org/pdf/1411.1784.pdf`

[9] Zhang, Han and Xu, Tao and Li, Hongsheng and Zhang, Shaoting and Huang, Xiaolei and Wang, Xiaogang and Metaxas, Dimitris: StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks
`https://arxiv.org/abs/1612.03242`

[10] Emily Denton, Soumith Chintala, Arthur Szlam, Rob Fergus: Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks
`https://arxiv.org/abs/1506.05751`

[11] Nitish Srivastava, Geoffey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov: Dropout: A Simple Way to Prevent Neural Networks from Overfitting
`https://www.cs.toronto.edu/ hinton/absps/JMLRdropout.pdf`

[12] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton: CIFAR dataset [12/04/17]
`https://www.cs.toronto.edu/ kriz/cifar.html`

[13] Amazon Mechanical Turk Pricing [12/04/17]
`https://requester.mturk.com/pricing`

[14] Lucas Theis, Aaron van den Oord, Matthias Bethge: A Note on the Evaluation of Generative Models
`https://arxiv.org/pdf/1511.01844.pdf`