MLP Coursework 3: Transfer Learning

G115 | s1791476 | s1546937

Abstract

Transfer learning has proven itself to be an incredibly useful area in deep learning, not only by facilitating those with limited access to resources such as data and time, but also by driving a deeper understanding of neural network features in general. Like all areas of machine learning these methods come with their own set of difficulties and heuristic methods in tow. In this paper we shall motivate and further detail a project to investigate the use of Bayesian methods to automatically optimize these heuristically tuned parameters and potentially alleviate some of the methods well known symptoms.

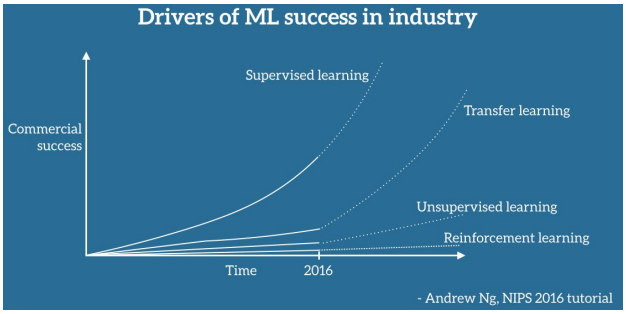

Figure 1. Andrew Ng's drivers of industry taken from his tutorial at NIPS 2016 and recreated by Sebastian Rudder The diagram illustrates the early exponential significance of supervised learning in industry, with transfer learning following a similarly impactful path later.

1. Introduction

The unprecedented rise of deep learning in the industrial sector has been in no small part due to the availability of data to accurately train networks upon. However, for companies and individuals who are not of the size of Google and Amazon the data required to train deep models is far from readily available, being simply infeasible to collect or to label. For instance in the medical imaging field, it may be extremely expensive to have a group of oncologists annotate the binary outcomes scans relating to brain tumors. This is precisely the issue that transfer learning methods are employed to solve.

It has been shown (Zeiler & Fergus, 2013) that in a convolutional neural network trained upon images, the more generic and therefore widely applicable we expect the features learned in the earlier stages of the network to be. For example, a feature which detects the edges of an object should be useful for object recognition whether the objects in question are animals or cars. Conversely, as we progress through the network the more customised the features will become to the data in question.

The transfer learning method then makes use of this of this insight in the following way, if we first fully train a model on a wealth of image data, the lower level weights should lie quite close to where they would be had our own data set been sufficient. As we progress toward the output layer, the more the weight sets should diverge. Therefore, by taking the pre-trained model, we should keep these lower layers fixed or *frozen* in weight space and allow the upper layers to tune slowly towards the new data.

This choice of which layers to hold fixed and which to finely tune is however non trivial and entirely heuristic. In fact each tutorial and resource online regarding this point suggests that we should take into account such factors as the similarity and relative sizes between the target and source datasets.These metrics are of course ill defined and highly variable across domains.

This decision we make, regarding the learning rates of each layer however could be viewed as a hyper-parameter of transfer learning itself. The cumbersome task of tuning hyper-parameters is of course a well established and highly researched area in deep learning. The work carried out by Snoek et al (Snoek et al., 2015)(Snoek et al., 2012), lays out precise guidelines for the utilization of Bayesian methods for automatic hyper-parameter tuning, either by the use of a Gaussian Process or Bayesian Neural network. These methods can provide useful tools for effective experimental planning and therefore an efficient use of resources.

In this paper we shall focus on this final point. We shall put forward questions about the use of Bayesian optimization methods for transfer learning and further establish objective tasks by which we may arrive at conclusions about these questions. We lay out the data to be used in the project and an examination of the initial experiments carried out thus far, which have provided a baseline to compare our later models against. Finally, we will present our interim conclusions about our motivation to carry out this research from

our initial investigation of the literature and experiments, before setting out a plan of work for the coming weeks.

2. Research Questions

As discussed in the previous section, transfer learning provides a great benefit to those with limited data, but methods for the approach are still highly ambiguous and are highly susceptible to issues such as overfit. We investigate the use of Bayesian methods to automatically optimize the hyper-parameters of the method and further analyze the performance of such an approach. Three research questions naturally arise from adopting this strategy, which range from basic to highly ambitious:

1. **Is it possible to translate the heuristics of transfer learning to hyper-parameters which we can automatically tune using Bayesian optimization?**

   Both transfer learning and Bayesian optimization are well established areas, but is it possible to engineer our model so that the choices of learning rates at each layer are optimized automatically.

2. **How would such a model perform against a manually tuned model?**

   How would such a model perform against one which we manually train by using the current best practices available. Perhaps the Bayesian model outperforms our standard model but so that the computational increase is unwarranted for the size of improvement.

3. **What does the ideal function of learning rate over network layers look like?**

   As previously discussed, current guidelines suggest that earlier layers in the network are frozen and later layers then finely tuned. This notion of 'freezing' layers equates to setting the learning rate to zero and fine tuning equates to a low learning rate. Therefore, if we model the learning rate as a function of depth in the network, this would be in essence a step function. A further ambitious task in our investigation will be to relax the restriction on earlier layers and see how the resulting optimized function might be, perhaps it will be a smoother increase or perhaps it will closely resemble the original.

3. Objectives

In order to address and consequently answer the research questions which we have proposed, we will lay out a set of objectives for the coming weeks. These objectives are task which naturally arise from the questions which produce metrics to allow accurate comparison, they once again increase consecutively in ambition, with the final task lying outside the scope of our two core objectives.

1. **Baseline 2.0**

   Our first step will be to follow the current guidelines and best practices to create a transfered model from the CIFAR100 source to CIFAR10_Percent datasets which will be manually tuned. This new baseline model will allow us comparison against our previous baseline VVG17 model on the limited data and against our future VGG17 model with Bayesian optimization incorporated. We aim to run six experiments at this stage, with three settings for the number of frozen layers and two learning rates for each of these settings.

2. **Bayesian Transfer Learning**

   Next we shall re-engineer our convolution network architecture TensorFlow so that current state of the art Bayesian optimization packages ((Snoek et al., 2012)) can automatically tune the defined hyper-parameters. At this stage we hope to experiment with both Gaussian Process and Bayesian Neural Network models. We will also need to experiment across different hyper-parameters of the Bayesian models such as the scale and bandwidth of the Gaussian Process model.

3. **Unfreezing Layers**

   Additionally, once we have found our preferred optimized model, we wish to relax the assumption of early frozen layers and investigate how the optimal learning rate increases with the depth of the model. We expect this model to be of much higher computational cost than previous models and therefore will be an additional goal. We will first experiment with setting pairs of layers to the same value, so that we have approximately eight hyper-parameters to tune.

4. Data

We are using two dataset, one is CIFAR-100 and other is CIFAR-10. Both the dataset have same attributes like dimensions, color bands and pixels, and only differ in number of classes. This makes them work on the same model architecture. The dataset we have used for training our model is CIFAR-100. This dataset has 100 classes containing 600 images each, which is further divided into 500 training and 100 testing images for each class. These 100 classes are grouped into 20 super-classes. Every image is labeled with one of the 100 class (known as fine labels) and 20 super-class (known as coarse labels). The size of each images is 32x32, yielding 1024 pixels x3 for each band, so 3027 pixels per sample.

The trained model on CIFAR-100 dataset will be transferred to the 10 percent of CIFAR-10 dataset. CIFAR-10 dataset has same properties as CIFAR-100, the only difference is it has 10 classes compared to 100 classes of CIFAR-100. Each class contains 6,000 images each, which are further divided into 5,000 training and 1,000 testing images for each class. Since we are using 10 percentage of training data of CIFAR-10, so the training set will be reduced to 5,000 (which is 500 for each class) and the testing sets will remain the same. We preprocessed the CIFAR-10 dataset,

by taking the random sampling from the training set, which limit the data into 10 percent, approximately dividing into equal set of each of the 10 classes. We only limited the training set, not the testing set as this will help in better generalization of the model.

Here are the current best results from both CIFAR-100 and CIFAR-10 dataset. The paper Fast and Accurate Deep Network Learning(Clevert et al., 2015) uses ELU(Exponential Linear Units) on CIFAR-100 dataset, which increases the performance on the deep neural networks and lead to higher accuracy of 75.72 percent. ELU reduces the vanishing gradient by identity of positive values, as opposed to other activation functions like rectified linear units (ReLUs), leaky ReLUs (LReLUs) and parameterized ReLUs (PReLUs).

The paper Fractional Max-Pooling(Graham, 2014) uses stochastic max-pooling on CIFAR-10 dataset, which reduces overfitting and lead to higher accuracy of 96.53 percent. The author used fractional max pooling in a spatially-sparse convolutional network. According to his spatially-sparse convolutional network design, he actually extended the image input spatial size by padding zeros. Additionally, fractional max pooling downsizes the input by a factor of pooling ratio which is often less than 2. These two combined together allowed stacking more convolutional layers than using regular max pooling and hence building a deeper network.

5. Methodology

Transfer learning methods have been in use with considerable success since the 1980s (Thrun, 1996). However the first successful application of the method in the domain of object recognition with the use of convolutional neural networks was by (Donahue et al., 2013) in 2013. This paper however described a one-to-one vanilla transfer of the network and as the authors noted, was extremely susceptible to becoming overfit.

(Yosinski et al., 2014) took this approach a step further by, in the first case, investigating the transferability of the features of these networks. It was here we see a layer by layer effect on the transfer model and learning just how valuable fine tuning in the method can be. (Oquab et al., 2014) further refine a general approach to the problem, detailing the benefit of adding some linear fully connected layers to the last stages of the transferred model before training on the target dataset.

Motivated by the this, we shall widely combine the approach taken in these papers. The model which we use is an adapted version of the MLP course GitHub code for a VGG style convolutional neural network which itself is an adaption of the original model specification (Simonyan & Zisserman, 2014). Our model consists of four convolutional blocks, each consisting of three convolutional layers and a single dimension reduction layer. The convolutional layers perform the standard convolution as used in the last semester, where the stride is one. The dimensional reduction layer is also a convolution however with a stride of two. This is widely favored over max-pooling in that it discards less information and is a learn-able filter. As described in the data section, our input is a batch of 32×32 images, each with three dimensions for RGB image bands. As a result of the stride two dimension reduction at each block the image dimension will be reduced by a factor of two and the number of feature maps increased also by a factor of two so that at the final fully connected layer we shall have batch-size $\times 4 \times 4 \times 1024$ connections. In line with the literature we shall add a second fully connected layer atop of the network when transferring the model from source to target.

We shall then optimize the our parameters for the transfer network in the method set out in the papers by (Snoek et al., 2012; 2015). In using Bayesian optimisation, we place a prior distribution over functions of performance of our model, for which our hyperparameters are inputs. By updating the posterior with the likelihood that our prior was correct we can better assess the set of parameters which will yield the best results for our next experiment. The packages we will use do this work automatically.

6. Experiments

The baseline experiments were performed using CIFAR-100 dataset on VGG(Visual Geometry Group) model. The VGG model uses 3x3 filters with stride and pad of 1, along with 2x2 max-pooling layers with stride 2.

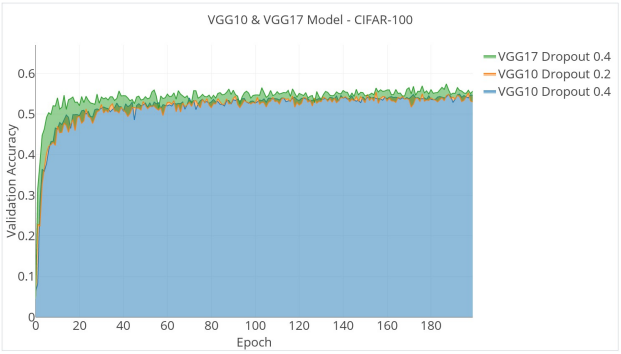

Figure 2. Classification Accuracy on CIFAR-100 dataset on VGG10 and VGG 17 for dropout : 0.2 and 0.4.

We performed the experiment for two dropouts 0.2 and 0.4 on the 10-layer VGG(VGG10) and 17-layer VGG(VGG17), and run the model for 200 epochs. The classification accuracy on the VGG10 model for dropout 0.2 was 53.8% and for dropout 0.4 was 54.5%.

The similar experiments we performed for VGG17 model on dropout 0.4, as 0.4 dropout was giving better results for VGG10 model. Classification accuracy of 57.4% was given by the VGG17 model, and is the baseline model for further experiments.

We did similar experiments on CIFAR-10 dataset with dropout 0.4. One was done on the entire CIFAR-10 dataset using VGG17 model, and other was done on the reduced data, which is 10 percentage of the CIFAR-10 training data(CIFAR-10_percent). Classification accuracy on the CIFAR-10 was 87.2% and for the CIFAR-10_percent was 55.6%

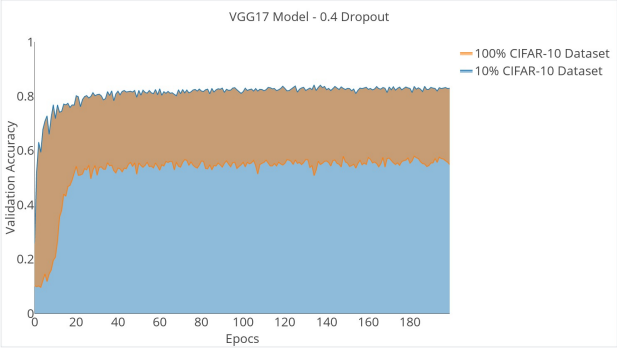

Figure 3. Classification Accuracy on 10 percent and 100 percent CIFAR-10 dataset on VGG17 model for dropout : 0.4.

We wanted to check the result of training the model on the reduced dataset. As in our future experiments we will be doing transfer learning, using the CIFAR-100 VGG17 model on the CIFAR-10_percent dataset. We will be looking into using the knowledge gained while training the CIFAR-100 dataset and applying it to the CIFAR-10_percent dataset, which is a related tear down dataset.

Data set	Model	Dropout Rate	Accuracy (%)
CIFAR-10_percent	VGG17	0.4	55.6
CIFAR-10	VGG17	0.2	87.2
CIFAR-100	VGG10	0.2	53.8
CIFAR-100	VGG10	0.4	54.5
CIFAR-100	VGG17	0.4	57.4

Table 1. Classification accuracies for VGG10 and VGG17 dataset on CIFAR-10 and CIFAR-100 data sets.

7. Interim Conclusions

In this paper we have motivated a project for the use of Bayesian methods to automatically tune hyperparameters of a transfer learning problem, namely, the number of layers which should be frozen in the model. We have further outlined the datasets to be used as source and target data and the methods and architectures of convolutional neural networks used to perform the image classification upon them. Finally we have outlined a project plan for the coming weeks along with mitigations to any potential risks which may derail the project.

8. Plan

Timeline

Figure 4 on the final page displays how we intend to reach our objectives over time in the coming weeks. The overlap of objectives in the diagram corresponds with the uncertainty in the time estimate in each task and also where certain tasks can be completed in parallel. A high level description of our project plan is as follows;

- **Weeks 1 & 2**

  The initial two weeks of the project shall focus upon setting up our environments and models in anticipation of the experiments in the latter three weeks. First, for our Bayesian optimisation, we have the choice of both Gaussian process model in the Spearmint package, which is our primary choice, and also a PyTorch implementation of a Bayesian neural network. We will investigate the viability of both methods in this time and also test each of them with online tutorials. If this initial stage moves quickly we will also begin testing the general transfer learning method and manually tuning the model for a baseline improvement from our interim results. In this week we shall also generate and set aside graphs and other results from experiments for later use in reporting.

- **Week 3** Beginning in week three and possibly before we shall begin to create our standard approach to the transfer learning problem. Our first task is to adapt the output layers of the network to handle ten classes instead of one hundred and also to add the extra fully connected layer. We then adapt the TensorFlow code to pass a parameter list specifying which layers are to be trained and which are to be frozen. We initialize then transfer the weights from our CIFAR100 model and carry out our experiments. We have decided to complete this baseline task after our initial testing with the Bayesian methods since they are more high risk so that we can adapt our plan earlier if needs be.

- **Week 4** Considering everything has gone smoothly up until this point we would like to further adapt our code so that each layer or at least each block has a variable learning rate which we can tune automatically and then investigate how this resulting network performs. We shall also begin drafting our final report in this week since we should have our results and conclusions from our primary objectives.

- **Week 5** In this week we will be primarily drawing together our final results for a complete comparison and finishing the details on our final report for submission.

Risks

We have considered the following risks for the project and possible approaches for their mitigation:

- **Underestimation of difficulty**

  We can estimate quite accurately how quickly our implementation of convolutional neural networks since we have experience in them however we do not have practical experience in Bayesian optimisation. This is one reason as to why we are investing two different Bayesian methods.

- **Underestimation of compute time.** Bayesian optimisation is notoriously slow, if this is a problem we shall reduce the size of our architecture.

  If any of the above or some unforeseen critical problem arises such that we cannot complete our set out project plan our backup plan shall be to take the standard transfer learning model created and investigate it's performance on another dataset such as Painters by Numbers where the task is more dissimilar.

References

Clevert, Djork-Arné, Unterthiner, Thomas, and Hochreiter, Sepp. Fast and accurate deep network learning by exponential linear units (elus). *CoRR*, abs/1511.07289, 2015. URL http://arxiv.org/abs/1511.07289.

Donahue, Jeff, Jia, Yangqing, Vinyals, Oriol, Hoffman, Judy, Zhang, Ning, Tzeng, Eric, and Darrell, Trevor. Decaf: A deep convolutional activation feature for generic visual recognition. *CoRR*, abs/1310.1531, 2013. URL http://arxiv.org/abs/1310.1531.

Graham, Benjamin. Fractional max-pooling. *CoRR*, abs/1412.6071, 2014. URL http://arxiv.org/abs/1412.6071.

Oquab, M., Bottou, L., Laptev, I., and Sivic, J. Learning and transferring mid-level image representations using convolutional neural networks. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1717–1724, June 2014. doi: 10.1109/CVPR.2014.222.

Simonyan, Karen and Zisserman, Andrew. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. URL http://arxiv.org/abs/1409.1556.

Snoek, J., Larochelle, H., and Adams, R. P. Practical Bayesian Optimization of Machine Learning Algorithms. *ArXiv e-prints*, June 2012.

Snoek, J., Rippel, O., Swersky, K., Kiros, R., Satish, N., Sundaram, N., Patwary, M. M. A., Prabhat, and Adams, R. P. Scalable Bayesian Optimization Using Deep Neural Networks. *ArXiv e-prints*, February 2015.

Thrun, Sebastian. Is learning the n-th thing any easier than learning the first? In *Advances in Neural Information Processing Systems*, pp. 640–646. The MIT Press, 1996.

Yosinski, Jason, Clune, Jeff, Bengio, Yoshua, and Lipson, Hod. How transferable are features in deep neural networks? *CoRR*, abs/1411.1792, 2014. URL http://arxiv.org/abs/1411.1792.

Zeiler, M. D and Fergus, R. Visualizing and Understanding Convolutional Networks. *ArXiv e-prints*, November 2013.

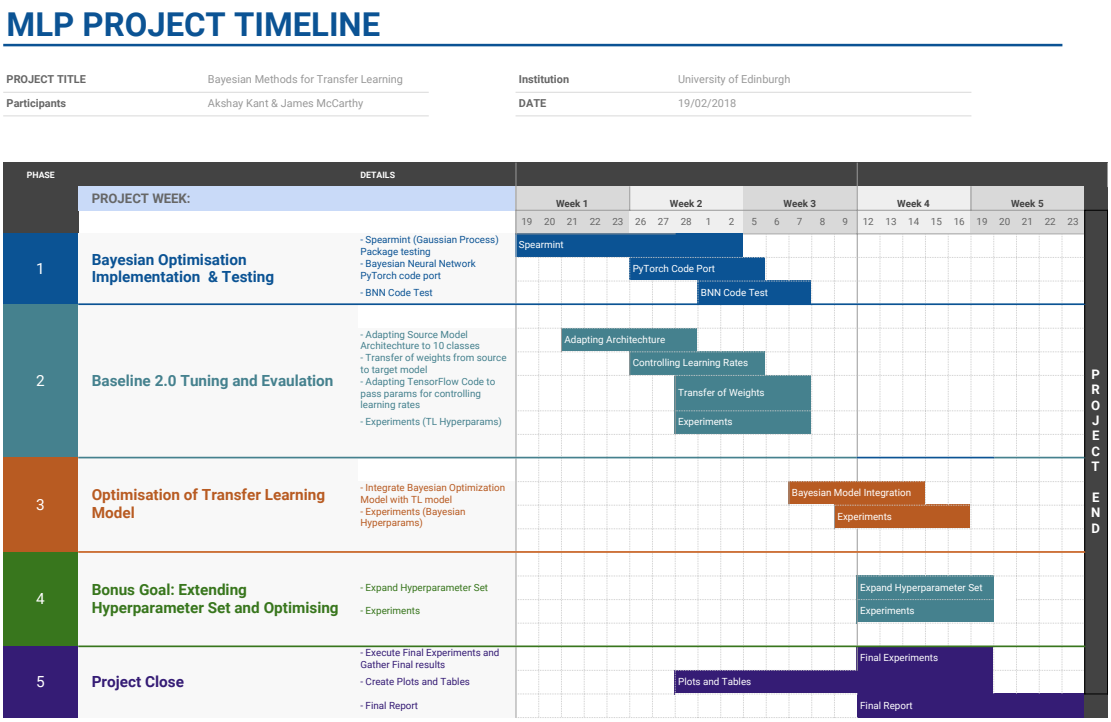

Figure 4. Gantt Diagram illustrating the time-line of our project