Continual Learning: Uncertainity based Continual Learning with Adaptive Regularization

Amir Raza, Mohamed Abdelsalam

23 December 2019

1. Abstract

Neural Networks learn to accomplish tasks by updating weights to minimize the loss for each task. Often times neural network training is very data intensive and computationally expensive, utilizing these learned network parameters across new tasks is thus very 'desirable'. If we try to naively use the same network on new tasks, the network does not know which weights to retain and ends up updating all the weights without taking into account the previously learned tasks, effectively forgetting them, ie **Catastrophic forgetting**. 'Continual Learning' aims to solve this problem by finding a balance between 'plasticity' to a new task against 'stability' of remembering the old tasks. Uncertainty based Continual Learning with Adaptive Regularization Ahn et al. (2019) proposes an approach on remembering the 'important weights' for each of the tasks based on an uncertainty measure (variance of the nodes in this case). The paper borrows ideas from [Blundell et al. (2015), Hinton and Van Camp (1993), Graves (2011), Nguyen et al. (2017)], for learning the variational parameters with backpropagation.
In context of IFT 6269 Final Project, we 1) provide a reimplementation for the code, 2) Investigate if the results as reported in the paper can be reciprocated.
`https://github.com/MohamedAbdelsalam9/UCL_Reimplementation`

2. Introduction

Continual learning: Neural networks, in general, are not capable of learning tasks sequentially, as they suffer from the problem of catastrophic forgetting as well as capacity saturation. The goal of Continual learning is to try and learn tasks one after the other using the same network in a sequential manner. De Lange et al. (2019) in their survey of continual learning mention that due to the difficulty of the general continual learning problem most papers relax the general setting to an easier task incremental one.
The **task incremental setting** considers a sequence of tasks, where each task consists of a number of classes. It is assumed that it is known to which task each input belongs to, but the network should classify the input among the classes of this task. In this setting one task is received at a time along with its training data. An offline training is then performed until convergence. $(X(t), Y(t))$ are randomly drawn from a distribution $D(t)$, with $X(t)$ being a set of data samples for task t and $Y(t)$ being their corresponding ground truth labels. The goal is to control the statistical risk of all seen tasks given limited or no access to data $(X(t'), Y(t'))$ from previous tasks $t' < t$. **For the previously learnt tasks, the data is no longer available**. The learning algorithm should adapt to new tasks, while retaining what it has already learnt on the previous tasks. The goal is to have models capable of adapting to new tasks and expanding their behaviour. Current neural networks models suffer from what is called **catastrofic forgetting**, as when they are trained on a new task they ignore and forget the old tasks. This work tries to tackle this problem.

Uncertainty based Continual Learning (UCL) (Ahn et al., 2019) proposes the notion of node- wise uncertainty. They add regularization terms that enforce stability by penalizing changes to parameters that where important for past tasks, and allow plasticity by controlling the actively learning parameters for a new task. As a regularization based approach it does not need to expand the model for every task. Similar to EWC(Kirkpatrick et al., 2017) and VCL (Nguyen et al., 2017), it uses a Bayesian framework for learning the network parameters.

Our contribution was to re-implement the network from scratch, and try to replicate the results which were reported in the original paper(Ahn et al., 2019).

3. Related Work

3.1. How to do Continual Learning?

1. Replaying past samples.
2. Regularization based methods.
3. Parameter isolation

Replaying:In this approach Samples are stored in their raw format or compressed in a generative model. Stored samples from previous tasks are replayed while learning a new

task to alleviate forgetting. This requires storage capacity , and is not scalable.

Parameter isolation: Different subsets of the model parameters are dedicated to the different tasks. It attempts to freeze the set of parameters learned after each task and grow some new branches for the new tasks/

Regularization-based methods 'identify important weights'. While learning a new task, large updates on these weights are penalized. EWC Kirkpatrick et al. (2017) applies the Bayesian framework for neural networks, which allows to find a posterior distribution over the network parameters which are dealt with as random variables, instead of mere point estimates in the parameter space. Following sequential Bayesian estimation, the old posterior over the weights of the previous tasks $T_{1:n1}$ constitutes the prior over the parameters when learning the new task T_n.

Variational inference (Nguyen et al., 2017) attempts to learn approximation of posterior distribution on a model. As Exact posterior is usually intractable, Variational inference can try to approximate posterior with a more tractable distribution $q(W|\theta)$.

4. Network Model

4.1. Bayes Network

Instead of using a simple weight matrix , the authors chose to use Bayesian Neural Networks. In Bayesian Neural Networks the 'Weights' of the network are random variables, which are sampled from their posterior distribution. No sampling for the bias is done. We postulate that Bias was not sampled, to keep the later derivations for losses simple (later confirmed after communication with authors).

Weights are sampled from a Gaussian with mean μ and standard deviation σ

$$p(\boldsymbol{W}) = \prod_{j=1} \mathcal{N}(w_j \mid \mu, \sigma^2) \tag{1}$$

Where the σ is obtained through ρ to ensure that is is always positive:

$$\sigma = \log(1 + e^{\rho}) \tag{2}$$

The parameters ρ and μ are learnt while back-propagating through the loss objective. The weights are sampled using the reparametrization trick (Ahn et al., 2019) to allow for backpropagation to be used.

4.2. Loss term

For learning a Bayesian neural network in the general setting, the original Loss term (Free energy) is (Ahn et al., 2019):

$$\begin{aligned} F(D, \theta) =& E_{q(\boldsymbol{W}|\theta)}[-\log p(\boldsymbol{D} \mid \boldsymbol{W})] + \\ & D_{KL}\big(q(\boldsymbol{W} \mid \theta)||p(\boldsymbol{W} \mid \alpha)\big) \end{aligned} \tag{3}$$

Applied to Continual Learning, we have it in the form:

$$\begin{aligned} F(D_t, \theta_t) =& E_{q(\boldsymbol{W}|\theta_t)}[-\log p(\boldsymbol{D_t} \mid \boldsymbol{W})] + \\ & D_{KL}\bigg(q(\boldsymbol{W} \mid \theta_t)||q(\boldsymbol{W} \mid \theta_{t-1})\bigg) \end{aligned} \tag{4}$$

Where the previous task posterior over the weights act as the new task prior (hence the first task has only the likelihood term $\log p(\boldsymbol{D_t} \mid \boldsymbol{W})$)

Using Gaussian Mean Field assumption, the above Loss can be reduced in terms of means and variances of the layers as :

$$\frac{1}{2}\sum_{l=1}^{L}\left[||(\frac{{\mu_t}^l - {\mu_{t-1}}^l)}{\sigma_{t-1}^l}||_2^2 + \mathbf{1}^T[(\frac{{\sigma_t}^l}{{\sigma_{t-1}}^l})^2 - \log(\frac{{\sigma_t}^l}{{\sigma_{t-1}}^l})^2]\right] \tag{5}$$

If we observe this loss term, we can see that there is a difference between ${\mu_t}^l$ and ${\mu_{t-1}}^l$ which is divided by the factor: ${\sigma_{t-1}}^l$. Intuitively this implies that while learning the new task, the change from μ_{old} to μ_{new} depends on how important the parameter is. ${\sigma_{t1}}^l$ is the uncertainty measure for corresponding mean weight of ${\mu_{t-1}}^l$,

Key assumption made by the paper is that importance of a parameter is inversely proportional to its variance . That is:

$$importance \propto \frac{1}{\sigma}$$

. If a parameter is important, it should have low variance.

4.3. Regularization

4.3.1. Finding important nodes from importance of weights

Memory is limited. Storing all the parameters μ and σ is storage expensive. To reduce the number of parameters required, authors suggest finding out 'the important' nodes from the learnt wes. The idea behind selecting an important node is that if any connection in layer L or layer (L-1) is strong then the node is important. The authors penalize updates for strong connections, as they don't want updates on these learnt parameters. Working on nodes also makes sense, because nodes give the final outputs we care for instead of the weight parameters, which are involved in the intermediate steps. **Weight gets high regularization strength** when either of the node it connects has low uncertainty (Fig 1).

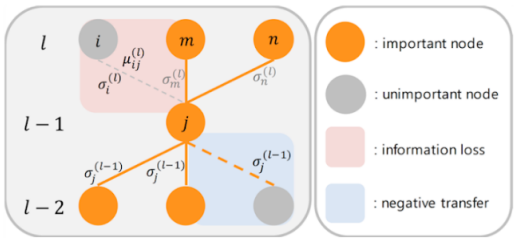

Figure 1: Node importance: Reducing from weights to Nodes

Here we identify strong node connections using ${\Lambda_{i,j}}^l$ parameter :

$$ {\Lambda_{i,j}}^l = max\left(\frac{{\sigma_{init}}^l}{{\sigma_{t-1,i}}^l}, \frac{{\sigma_{init}}^{l-1}}{{\sigma_{t-1,j}}^{l-1}}\right) \tag{6} $$

So after these modifications the loss becomes: **Final Loss**:

$$ -\log p(\boldsymbol{D_t} \mid W) + \sum_{l=1}^{L}\left[\frac{1}{2}||\Lambda^l \circ ({\mu_t}^l - {\mu_{t-1}}^l)||_2^2 + (\sigma_{init}{}^l)^2||(\frac{{\mu_{t-1}}^l}{{\sigma_{t-1}}^l})^2 \circ ({\mu_t}^l - {\mu_{t-1}}^l)||_1 + \frac{\beta}{2}.\mathbf{1}^T[(\frac{{\sigma_t}^l}{{\sigma_{t-1}}^l})^2 - \log(\frac{{\sigma_t}^l}{{\sigma_{t-1}}^l})^2 + (\sigma_t{}^l)^2 - \log(\sigma_t{}^l)^2]\right] \tag{7} $$

The third term , $(\sigma_t{}^l)^2 - \log(\sigma_t{}^l)^2)$ is added to make sure that σ parameter can still learn and does not become fixated with already learnt values. β is a hyper-parameter here. See (Ahn et al., 2019) for the detailed derivation.

5. Experiments and Results

5.1. Datasets for Continual Learning tasks

The datasets for continual learning aim to have several tasks with new classes added one after the other.
Split MNIST Ten digits are separated into pairs of two digits at a time. Tasks are numbered from 1 to 5. Task 1 is classification between (0,1), Task 2 is classifying between (1,2) and so on, till Task 5.
Split NotMNIST It deals with pairs of characters instead of digits(and hence the name not mnist). Each task is a classification between pair of characters taken at a time. So Task 1 is (A,B) classification, Task 2 is (B, C) classification and so on till Task 5. This dataset is much more challenging than Mnist, as the images are more varied.

Plot description Model is trained on all five tasks. We check for accuracy after training on each new task (x axis). Therefore it makes sense that the model accuracy on a specific task drops after learning on the subsequent tasks. Final plot shows the average of the accuracies across all the observed tasks at each time step. Our comparisons have been made between a Vanilla network(which only trains on the data likelihood with no regularization) and the UCL implementation.

5.2. MNIST

We observed that vanilla network is not able to remember previous tasks. The accuracies keep falling down for .While UCL still performs well on the previously learnt tasks.

5.2.1. Comparison with Paper results

Refer to figure 2 Our model obtains an Average accuracy of 98.9%, while the Vanilla model obtains an accuracy of 96.9%. The reported results in the paper are 98.7% for VCL (Nguyen et al., 2017), and 99.6% for UCL. Hence the difference between our implementation and the paper implementation is very small.

Even though the vanilla model has high accuracy, it should be noted that UCL was able to achieve higher accuracy on MNIST, which is a significant achievement.

5.3. NotMNIST

5.3.1. Comparison with Paper results

Refer to figure 3
Bottom Line: Average accuracy UCL 92.9%, Vanilla 75%
Reported Paper Results: Average Accuracy EWC 84%, VCL 90.1%. UCL 95.7%

Better results were observed when we did not do the division of the regualrization term with mini batch size.

6. Discussion and Conclusion

We observed that the regularizer added to the loss term is important for retaining accuracy on previously learnt tasks. Dividing the regularization term with minibatch size or removing the regularizer altogether, degraded the task accuracies. The current model has some scope for extension, as Bias parameters are not being sampled currently to simplify the problem. We also found many discrepancies between the value of parameters reported in the paper, and the actual values used. The Hyperparameter values not mentioned in the paper include (scheduler, regularization weight). Our opinion after running the model implementation is that the paper achieves what it set out to. Even though the model needs fine-tuning, but it is able to achieve the task of continual learning on SplitMnists and NotMnist data sets.

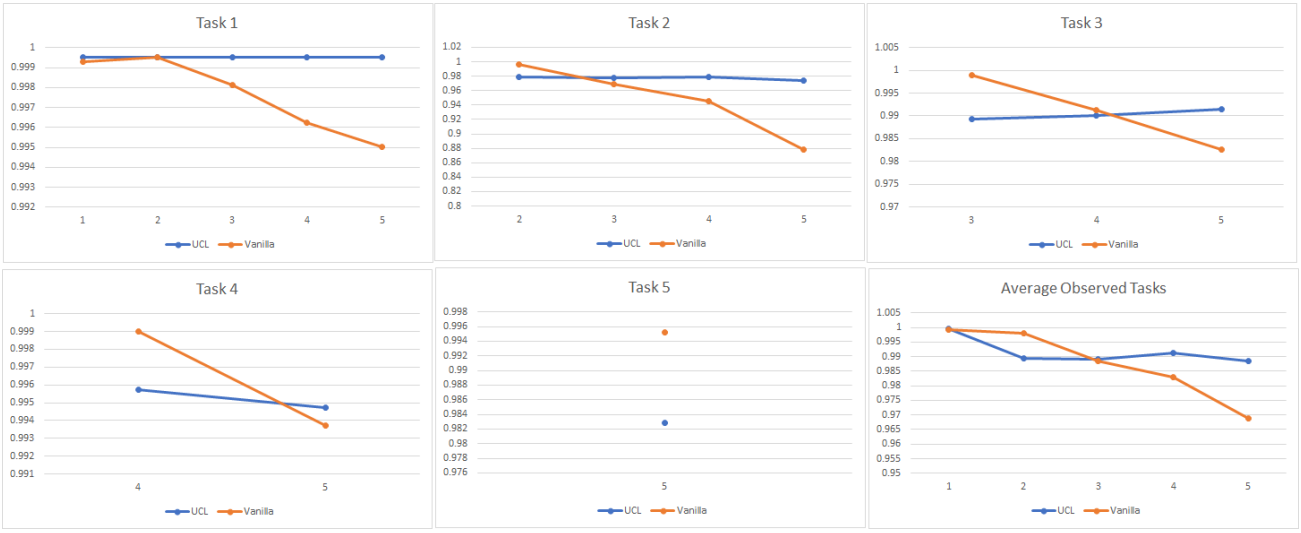

Figure 2: MNIST: Accuracies vanilla and UCL network

References

H. Ahn, S. Cha, D. Lee, and T. Moon. Uncertainty-based continual learning with adaptive regularization. In *Advances in Neural Information Processing Systems*, pages 4394–4404, 2019.

C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015.

M. De Lange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, and T. Tuytelaars. Continual learning: A comparative study on how to defy forgetting in classification tasks. *arXiv preprint arXiv:1909.08383*, 2019.

A. Graves. Practical variational inference for neural networks. In *Advances in neural information processing systems*, pages 2348–2356, 2011.

G. Hinton and D. Van Camp. Keeping neural networks simple by minimizing the description length of the weights. In *in Proc. of the 6th Ann. ACM Conf. on Computational Learning Theory*. Citeseer, 1993.

J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

C. V. Nguyen, Y. Li, T. D. Bui, and R. E. Turner. Variational continual learning. *arXiv preprint arXiv:1710.10628*, 2017.

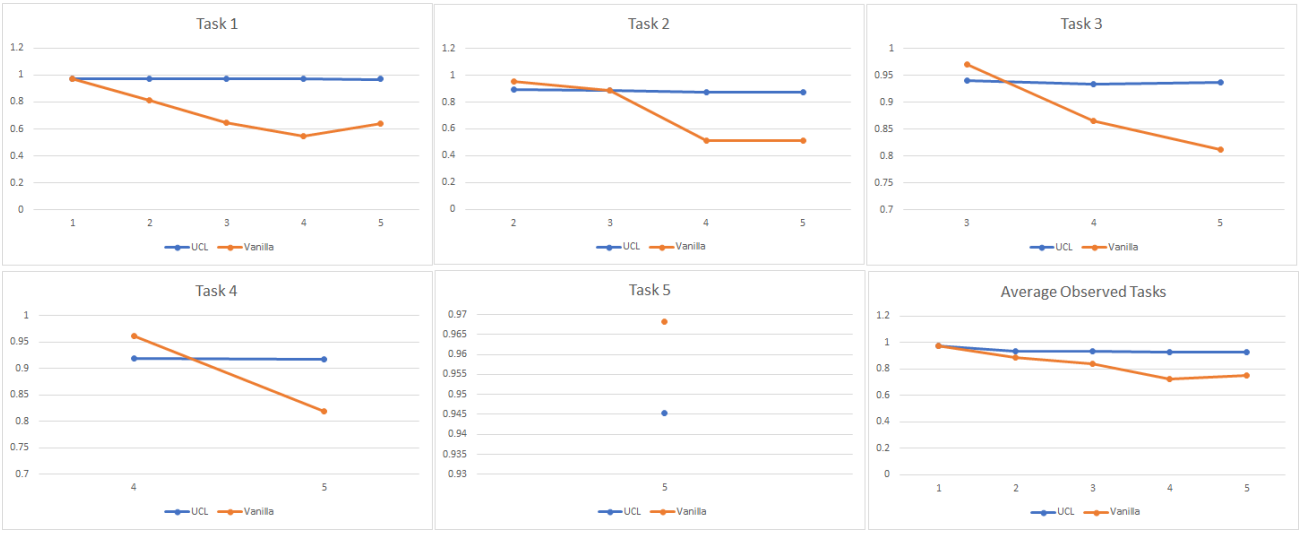

Figure 3: Not MNIST: Accuracies vanilla and UCL network