COMP6248 REPRODUCIBILITY CHALLENGE
YOU ONLY TRAIN ONCE: LOSS-CONDITIONAL TRAINING OF DEEP NETWORKS

Atul Joshi, Hanuragav Muthiah Giri & Kamsiriochukwu Chukwuma Ojiako
Department of Electronics & Computer Science
University of Southampton
{aj4n21, hmg1u21, kco1e20}@soton.ac.uk

ABSTRACT

In this coursework we attempt to reproduce the experiments of the paper titled 'YOU ONLY TRAIN ONCE: LOSS-CONDITIONAL TRAINING OF DEEP NETWORKS' by Dosovitskiy et al, published in 2020. We give a detailed explanations of the challenges faced, experimental assumptions and techniques implemented to reproduce the results. The paper is reproducible to a degree, the code was implemented from scratch as the authors did not release the official implementation.

1 INTRODUCTION

The paper reflects on certain issues faced by machine learning practitioners on the choice of quality of the model. That is the trade-off between a model attaining high performance and the ability to generalise well on new data. A similar case is discussed in the paper highlighting the challenges faced by practitioners in optimising multiple losses. A prime example is training a model for image compression where the trade-off is between quality and size of the compressed image. The paper proposes a straightforward and widely applicable method for dealing with multi-term loss functions. They train one model that covers the whole space of alternative loss weightings, rather than iteratively training a single model for each parameter combination of the loss. This is accomplished in two steps by a) sampling a loss parameter from a particular distribution and b) conditioning the layers of the network on the loss parameter sampled.

2 BACKGROUND

The YOTO(You Only Train Once) technique is widely implementable for various machine learning models and architectures (Dosovitskiy & Djolonga, 2020). The paper implements YOTO technique on three different models namely, a) β- variational autoencoder, b) Learned image compression, and c) Style transfer. In this coursework we re-implement the β-VAE equipped with YOTO and recreate the results. The β-VAE (Higgins et al., 2017) is an extension of the Variational Autoencoder with an additional hyper-parameter β. The β-VAE is stable while training, makes minimal assumptions about the data, and only requires modifying one hyperparameter.

3 IMPLEMENTATION OVERVIEW

The paper proposes that training the β-VAE for each single parameter can be cumbersome and time consuming. They overcome this by training the model on by re-sampling the β parameter from a specific distribution for every training point. The distribution used is a log-uniform distribution. A single positive parameter β in the loss trades off reconstruction quality and divergence between the estimated posterior and the prior. This parameter is then used to condition the layers with the help of FiLM(Feature-wise Linear Modulation). The authors test the YOTO technique on two datasets, 1) CIFAR10 and 2) Shapes3D.

3.1 Architecture

Both datasets use different architectures for their for the encoder and decoder. THe CIFAR10 uses 3 groups of 2 convolutional layers with the number of channels as 8,8, 16,16,32,32. For the encoder, each convolutional layer uses a kernel size of 3 and stride 2. This is followed by 2 fully connected layers size 256 and 512. The decoder is a symmetric to the encoder with convolutional up-sampling. The last layer of each group had a conditioning layer(FiLM). A similar architecture was employed for the Shapes3D dataset with notable differences in the number of convolutional layers. It possesses 4 convolutional layers with channels as 8,16,32,64 followed by 2 dense layers. The CIFAR10 and Shapes3D architecture uses a latent space of 256 and 20 respectively. This architecture was implemented with few modifications as the input size did not match with the kernel size and strides of each layer. An varying padding sequence was employed to overcome this for each architecture.

3.2 FiLM (Feature-wise Linear Modulation)

FiLM (Perez et al., 2017) layers use a basic feature-wise affine transformation based on conditioning information to influence neural network computation. These layers could be as simple as a neural network. In this case it is a single hidden layer MLP. This is implemented in Pytorch by obtaining the loss parameter β from a distribution at each training point. It was broadcast into a 256 vector and was used as input to the neural network. This outputs 2 vectors (μ, σ) of size equal to the number of channels in the corresponding conditioned layer. The outputs contain a sigmoid activation function to squish the values between (0,1). Each channel of the β-VAE is then modified with the corresponding values in the FiLM network.

$$\hat{f} = \sigma f + \mu$$

That is the each value is broadcast into the size (height and width) of the output feature and stacked. This is then used to shift and multiply with the feature outputs with the corresponding broadcast μ and σ.

4 Experiments

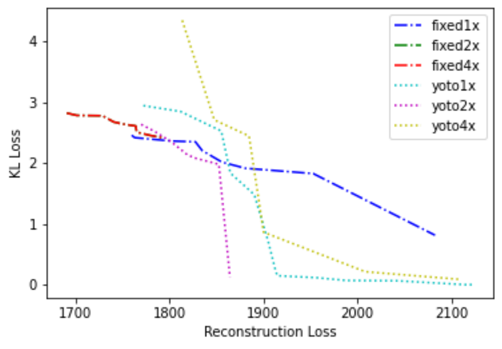
The paper performs experiments on both datasets with different architectures. We attempt to recreate the Reconstruction loss vs KL loss graph for a multitude of experiments. They plot the frontier losses, which is the Pareto front of both the losses plotted against each other. They run the experiments for fixed weight, where the loss is a fixed β value. The experiments are run for ”wider” models, i.e for architectures with increased number of channels. The sequence of experiments contain 1x, 2x and 4x wider models for both the fixed weight and YOTO implemented architectures. Each model was run for 100 epochs. Figure 1(a) shows the implementation for CIFAR10 dataset and a similar plot to the original graph is obtained.

The second experiments were attempted to recreate the Full loss difference vs the $\log_2$ of Beta. This is run for the same sequence of wide architectures. Here we train the fixed weight models for the loss parameter value β = 1, 16, 64, 256 and obtain the corresponding test loss. This is subtracted with the the baseline model which is the fixed weight model trained with 1x wide. The difference between the test loss of baseline and the YOTO implemented models with varying width and beta is plotted in Figure 1(b).

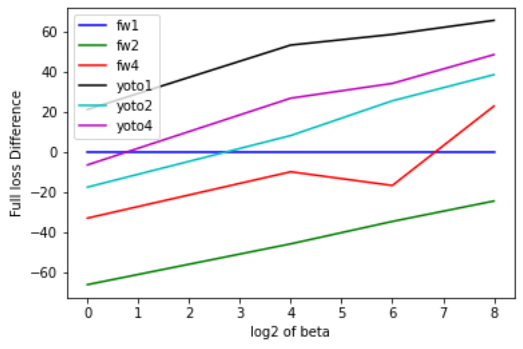
The third attempt is to recreate the Shapes3D visualisation. We train the Shapes3d architecture with β = 1, 128, and 512 for 1x wide architecture. The Shapes3D dataset was sufficiently large and we used a subset to train the model. The subset was split into train and test. Figure 2 shows the test set visualisations for the fixed weight and corresponding YOTO implemented model for different β values.

5 Results

Although the paper provides the code repository, it is in-accessible and the code was implemented from scratch using Pytorch 1.11.0. The authors give the architectures for the two datasets, but the kernel size and strides mentioned are not practically implementable without sufficient padding as this was not mentioned in the paper. While progressing through the network, the kernel size becomes bigger than the height and width of the feature output.



(a) Frontier Loss



(b) Full Loss Difference vs Beta

Figure 1: CIFAR10 Loss plots

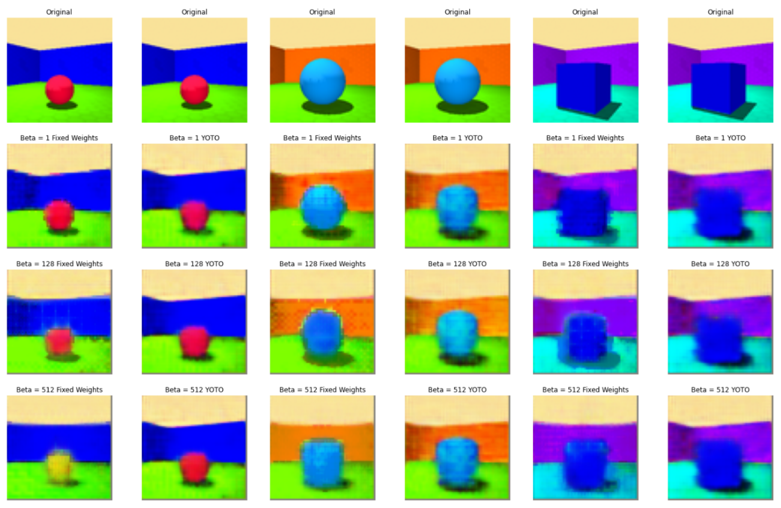


Figure 2: Shapes3D Test Set Visualisations

The authors do not explicitly mention the implementation of the conditioning layers(FiLM). They were implemented on the assumption that the loss parameter β has an influence on the feature outputs. When plotting the Reconstruction Loss with the Kulback Liebler Divergence, the graph is erratic. The authors overcome this by using the pareto front is a collection of non-dominated solutions in which each aim is seen as equally valuable. Hence the 'Frontier' term relates to the Pareto Front. The authors perform the experiments with at least 1000 epochs. Due to computational limits we train all models with 100 epochs. From Figure 1(a) we can see that there exist an inverse trend between the KL-loss and Reconstruction loss as expected from the original paper. From Figure 1(b) the full loss difference shows that as β increases the Full loss difference increases. Since these were the results obtained from the CIFAR10, our experiments do not follow the same trends as shown in the original paper for the CIFAR10 dataset. But it follows the trends obtained from the Shapes3D dataset of the original paper. This is because of architectural differences between our implementation and a smaller training duration (100 epochs). Figure 2 shows the test set visualisations for the Shapes3D dataset. Although they are not similar to the results obtained in the original paper, expected trends do hold. As the loss parameter increases, the is reconstructed images lose its quality. Upon close inspection there are subtle changes obtained with in the images reconstructed with different loss parameters for the YOTO implemented models compared to the fixed weights.

6 CONCLUSION

The original paper gives the reader a full overview of the theory that underpins the paper's operation. The supplement also provided additional technical data about the model itself that were not critical to the theory, such as specifics regarding testing and the network architecture employed, although certain sections were left to our own interpretation. The combination of paper and supplement was insufficient to replicate the results entirely. Finally, making full use of a variety of sources enabled effective reimplementation and replication of results.

REFERENCES

Alexey Dosovitskiy and Josip Djolonga. You only train once: Loss-conditional training of deep networks. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=HyxY6JHKwr`.

Irina Higgins, Loïc Matthey, Arka Pal, Christopher P. Burgess, Xavier Glorot, Matthew M. Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *ICLR*, 2017.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron C. Courville. Film: Visual reasoning with a general conditioning layer. *CoRR*, abs/1709.07871, 2017. URL `http://arxiv.org/abs/1709.07871`.

(Dosovitskiy & Djolonga, 2020) (Higgins et al., 2017) (Perez et al., 2017)