### 02456 DEEP Learning, DTU Compute, AUTUMN 2020

### SEMI-SUPERVISED LEARNING USING VARIATIONAL AUTOENCODERS

*Uffe Poul Hansen s203264, Axel Bregnsbo* *s152672*

Det Danske Klasselotteri A/S, Demant A/S

#### Abstract

We have implemented the two Variational Auto-Encoder (VAE) classifiers from "Semi-supervised Learning with Deep Generative Models", by Kingma et al [1]. The M2 classifier is using latent space augmented with a discreet classification variable. With this classifier, using 40,000 un-labelled data and only 100 labelled data, we achieved a, so far undisputed, record breaking 94.3% accuracy.

The high M2 performance is due to

* Using a higher alpha = 0.5 for the conventional classifier loss in eq. (9) [1], compared to the quoted 0.1.
* Combining forward pass for both labelled and un-labelled data before doing back propagation and coefficient update.

**1. Introduction**

This project explores digit classification on pictures from MNIST, using a VAE based classifier capable of accepting both labelled and un-labelled data, thus doing semi-supervised learning. The main purpose is to familiarize us with generative VAE’s. To this end we are building the M1 and M2 VAE classifiers described in "Semi-supervised Learning with Deep Generative Models", by Kingma et al [1]. To ensure the quality of the constructed models we have compared these with a conventional FF classifier.

For a fair comparison between FF, M1 and M2 we have used the same the number of weights in the networks involved in the classification, as this is the part that will be used in a deployment of these networks.

We have also used the same size for the M1 and M2 VAE latent space. There is no reason to think these should be equal, as the architecture of M1 and M2 is different. Ideally one should find the optimal latent-space size of M1 and M2 independently, while keeping total number of classifier coefficients constant, and use that in a comparison. Due to time constraints we did not perform such a scan, and have kept the sizes the same in lack of more information.

**2. The Models - Theory**

Figure 1 depicts the M1 classifier from [1]. This is classic VAE which is trained on un-labelled data. As a loss function we are using (1), which is a lower bound on the proper loss function but computationally feasible. After the VAE has been trained on un-labelled data, we train a classifier using labelled data. The classifier uses the latent variables as inputs, instead of the raw picture, and is trained with a conventional cross-entropy loss.

Figure 2 depicts the M2 classifier, when given labelled data, and Figure 3 when M2 is given unlabelled data. The idea in M2 is that we have an additionally to the normal latent variable we have 10-dimensional multinomial distribution modelling the number.

Figure 3 depicts the M2 classifier when given un-labelled data. We calculate a and a and associated loss for each of the digits 0 to 9. These losses are then weighted with the output of the network. In the M2 presented in [1] the y prior only influences the my for the q(z|x,y) posterior, but not the sigma. In our M2 y influences both my and sigma.

**3. Building The Models**

Our main focus was getting working VAE classifiers, and only secondary optimizing these. A decision which turned out to be wise. Especially M2 was tricky to get working. We scanned the most important hyper parameters:

* size of the latent space vector
* alpha weight in M2 loss formula (3)

Originally, we also had dropout with p = 0.5 on all layers. We tried different values and found that is almost all cases this made results worse, and it was therefore disabled.

Remaining hyper parameters were left unchanged, these being:

* Sub-networks calculating posteriors in encoder and decoder are all FF, aka MLP
* 3 hidden layers
* ReLU activation functions
* BatchNorm1d on hidden layers
* Adam optimizer with learning rate 1e-3
* Batch size of 64

The models are summarized in Table 1

<table of models>

**3.1 Back-propagation and alpha**

With the first functioning M2, that did not blow up with NaN, we got 78% accuracy. This was achieved by repeating the small labelled training set e.g. 50 times before running a single unlabelled one. We guessed that labelled and unlabelled data were pulling the coefficients in different (opposite?) directions. Based on this assumption we changed the Backward Propagation and coefficient Updating (BPU), so instead of doing a separate BPU after labelled and another after unlabelled data, we made a forward pass with labelled followed by 10 forward passes the unlabelled, and only then do a single BPU which now takes both loss functions into account. Together with using a higher alpha our accuracy results jumped to above 90%.

**3.2 Coding pitfalls**

We fell into quite some PyTorch and model-building bumps during our journey. Especially M2 was tricky

to get to work. Some of the pain points:

* Wrongly having ReLU activation function on the network calculating the posterior my,sigma
* M2 training ending in NaN after a few epochs due to sub-parts of the loss function having wrong sign
* Applying softmax() along wrong dimension
* M1 encoder weights were not frozen during classifier training.
* Getting too good classifier results because, because a fresh set of labelled training set were wrongly introduced when doing an incremental training.

**4. Results**

**4.1 Inside the VAE**

**4.2 Classifier Results**

**4.3 Style Transfer**

**5. Conclusion**

The parameter in eq. (9) and the cross-entropy loss it controls, seems to be given a cavalier treatment in [1], only mentioning that the value 0.1 was being used. In our case it had a large impact on achieving the 94.3% accuracy. We suspect [1] did not combine labelled and unlabelled forward passes before doing BPU, and therefore did not see a large impact of alpha.

In our work influence of most hyper parameters were investigated. E.g. the current 520,000 weight network can surely be reduced in size by changing the first layers in the FFNN network to CNN types.

**6. References**

[1] Kingma et al. (2014). *Semi-supervised Learning with Deep Generative Models*, arxiv.org/abs/1406.5298v2