MONCAE: MULTI-OBJECTIVE NEUROEVOLUTION OF CONVOLUTIONAL AUTOENCODERS

Daniel Dimanov, Emili Balaguer-Ballester
Faculty of Science and Technology, Bournemouth University
Bournemouth, BH12 5BB, UK
{ddimanov,eb-ballester}@bournemouth.ac.uk

Colin Singleton
CountingLab
Reading, UK
colin@countinglab.co.uk

Shahin Rostami
Data Science Lab, Polyra Limited
Bournemouth, UK
shahin@polyra.com

ABSTRACT

In this paper, we present a novel neuroevolutionary method to identify the architecture and hyperparameters of convolutional autoencoders. Remarkably, we used a hypervolume indicator in the context of neural architecture search for autoencoders, for the first time to our current knowledge. Results show that images were compressed by a factor of more than 10, while still retaining enough information to achieve image classification for the majority of the tasks. Thus, this new approach can be used to speed up the AutoML pipeline for image compression.

1 INTRODUCTION

Convolutional neural networks have achieved exceptional results for computer vision problems (Goodfellow et al., 2016), but they require expertise to be constructed. Thus, until recently, researchers relied mainly on trial and error to determine the best hyperparameters and network architecture (Stanley et al., 2019). Recently, there has been an increase in the number of effective approaches for automated convolutional network design, such as evolutionary algorithms (Lu et al., 2019; Miikkulainen et al., 2019; Real et al., 2019; Stanley et al., 2019), reinforcement learning (Qin & Wang, 2019; Tan & Le, 2019) and other automated approaches (He et al., 2021).These methods have been able to match and sometimes surpass expert-informed, manually designed architectures.

One of the most successful approaches for automated machine learning has been neuroevolution (NE) (Real et al., 2019; Lu et al., 2019), but its long training times makes it unfeasible for a plethora of problems (Real et al., 2019). Interestingly, it was recently shown that choosing a curated set of attributes or building new ones from the original features, can reduce the computational cost and even improve the performance (Charte et al., 2020).To overcome this drawback, this paper proposes a neural architecture search approach based on neuroevolution to approximate the Pareto-front of convolutional autoencoders. This method is designed to optimize the trade-off between reconstruction loss and image compression (calculated based on the size of the bottleneck layer). To the best of the authors' knowledge, this is the first attempt to do neural architecture search for convolutional autoencoders and to use multi-objective optimisation technique in this context.

2 RELATED WORK

Autoencoders In recent years, autoencoder neural networks have shown promise for addressing a range of supervised and unsupervised problems (Ballard, 1987; Schmidhuber, 2015). Autoencoders are NNs with symmetric input and output layers, termed encoder and decoder, that usually contain a bottleneck layer in between (Schmidhuber, 2015; Goodfellow et al., 2016). The task they solve is to minimise the reconstruction loss, which is calculated based on the difference between the original input and the reconstruction (output) (Baldi, 2012). This allows for various applications ranging from style transfer (Qian et al., 2019) and generative techniques (Dosovitskiy & Brox,

2016) to semi-supervised learning (Akcay et al., 2018), data denoising (Gondara, 2016) and image compression techniques (Charte et al., 2020) For more sophisticated datasets, convolutional layers can be incorporated instead of simple dense layers, these autoencoders are referred to as convolutional autoencoders (Azarang et al., 2019). In difference with simple autoencoders, they achieve the compression through the use of convolutional and pooling layers instead of simply having fewer neurons and the upsampling is achieved through deconvolutions (also known as upsampling layers) (Baldi, 2012). The state-of-the-art in autoencoders is composed of architectures designed by experts, which are intended to work for the specific scenarios (Charte et al., 2020). Currently, given the increasing interest on Automated machine learning(AutoML) (He et al., 2021), a range of automated approaches for designing and optimisation architectures have been proposed (Real et al., 2019; Lu et al., 2019; Yan et al., 2020).

Neural Architecture Search, AutoML and Neuroevolution The rise in popularity of AutoML is s natural response to address the high demand for expertise and computational resources for deep learning in industry and academia (He et al., 2021; Stanley et al., 2019). The automation involves not the NN training, but also the design of architectures, which is known as Neural Architecture Search (NAS) (Stanley et al., 2019; Elsken et al., 2019). Multiple different algorithms and approaches have been proposed with the first significant one for modern NAS being the work of (Zoph & Le, 2016). As the foundations were laid, various reinforcement learning (Qin & Wang, 2019; Tan & Le, 2019), Neuroevolution (Lu et al., 2019; Miikkulainen et al., 2019; Real et al., 2019; Stanley et al., 2019) and other approaches started to emerge.

A significant breakthrough in the field occurred in 2017, when Real et al. (2019)'s AmoebaNET reached state-of-the-art performance, outperforming architectures designed by experts. The AmoebaNet is a Neuroevolution algorithm (Real et al., 2019). In such approaches, decoding the NN (the individual in an evolutionary algorithm context) is added as an extra step before the evaluation, whilst encoding the individual back to genotype is added after the evaluation (Stanley et al., 2019). Recently, indirect encodings (Hadjiivanov & Blair, 2016) (closely inspired by how DNA works (Miikkulainen et al., 2019)) have shown promising results. Neuroevolution has been successfully used for autoencoder optimization in a range of studies (Charte et al., 2020; Sereno, 2018; Alvernaz & Togelius, 2017); however, a limitation of these approaches is that they typically focus on evolving a NN with dense activation layers (e.g., in Charte et al. (2020)'s EvoAAA) instead of convolutional and pooling layers, which have shown great potential for computer vision (Gu et al., 2018).

3 MONCAE

MONCAE [1] is built on top of DEvol (Davison, 2017)[2], which is a neuroevolution algorithm for image classification. We modified this algorithm to allow for searching optimal convolutional autoencoders, and to evaluate the population using multi-objective optimisation. Thus, to the best of our knowledge, this is the first attempt of using neuroevolution for automating the design of convolutional autoencoders; whilst incorporating multi-objective optimisation for their automated NAS. As we use DEvol as the backbone of this approach the evolutionary algorithm behind it is a genetic algorithm, which uses tournament selection and both crossover and mutation as variation operators.

A severe limitation of AutoML approaches (and neuroevolution in particular) is that these algorithms take a tremendous amount of time to discover optimal solutions (He et al., 2021). In the case of AmoebaNET, the time exceeds 50000 GPU hours for grayscaled CIFAR-10 (Appendix A),

which makes it unfeasible for many modern-day scenarios (Lebedev & Lempitsky, 2018). There is a plethora of different attempts for solving this problem, focusing both in the 'data space' (Wang et al., 2018; Singh & Lee, 2017) as well as in the 'algorithmic space' (Hinton et al., 2015; Frankle & Carbin, 2018). Recently, researchers have demonstrated that one promising approach is to split the problem into two stages: first feature engineering and then using the extracted features to solve the problem at hand (Sereno, 2018). To achieve that, Sereno (2018) and Charte et al. (2020) have showcased the possibility of using an automatically discovered autoencoder to tackle this.

[1] *URL will be provided for the camera-ready version

[2] https://github.com/joeddav/devol

We followed a similar methodology, but instead of using dense layers for our architecture, we allowed the neuroevolution algorithm to search for convolutional and pooling layers too. Moreover, we implemented the hypervolume indicator (HVI) (Equation 2) (Rostami & Neri, 2016; Guerreiro et al., 2020) to find a trade-off between two main objectives: 1) the level of compression, for which we use a metric, which is roughly based on Bayesian information criterion (Watanabe, 2013) (Equation 1) and 2) the reconstruction loss.

Remarkably, with our approach, further objectives can be added. In Equation 2, $\boldsymbol{X}$ refers to the matrix containing all performance metrics for the whole population set, $\mathbf{r}$ refers to the vector of reference points for each objective (f^{ref} per objective) , m refers to the number of objectives, and f_m refers to the objective function for objective m.

$$Level\ of\ compression(\mathbf{k}) = log_{10}(\prod_{i=0}^{dimensions}(\mathbf{k}_i)) \tag{1}$$

$$HVI(\mathbf{X}, \mathbf{r}) = \Lambda\left(\bigcup_{\mathbf{x}_n \in X}[f_1(\mathbf{x}_n, f_1^{ref})] \cdots [f_m(\mathbf{x}_n, f_m^{ref})]\right) \tag{2}$$

Based on these two objectives and prespecified reference points for each one, we first calculated the HVI of the population at each generation, and then evaluated the contribution of each individual to the overall HVI (CHVI, Equation 3), which used as the fitness score. The process follows the general Neuroevolution cycle; details for each step can be found in Appendix B.

$$CHVI_{\mathbf{x}_i} = HVI(\mathbf{X} \cup \{\mathbf{x}_i\}) - HVI(\mathbf{X} \setminus \mathbf{x}_i) \tag{3}$$

Experimental Setup We evaluate MONCAE on three datasets: MNIST (LeCun et al., 1999), Fashion-MNIST (Xiao et al., 2017) and CIFAR-10 (Krizhevsky et al., 2014) (see dataset details in Appendix A). For each dataset, we do 10 runs with different random seeds (Appendix D) for 20 generations and a population size of 20, a maximum of 5 epochs and using early stopping. After the process is complete, we finetune the final population by training for an extra 20 epochs using the parameters, hyperparameters, optimiser and architecture discovered by the algorithm. Experiments were implemented in python, TensorFlow and a single RTX 3090 NVidia GPU with CUDA 11.1.

Using multi-objective optimisation with a population-based approach such as neuroevolution allows us to examine the whole final approximation set of produced solutions, instead of picking one. This can allow the stakeholders to choose the model that best fits the context of the problem at hand. Moreover, setting the reference points can also induce bias towards certain solutions and enable preference articulation to be added to the optimisation process.

4 RESULTS

Average results over the whole population would undermine our purpose of producing an approximation set from which to choose. Thus, we present the results from the runs having latent representations which compress the input by at least a factor of 2, since more accurate, but larger representations are not the focus of this work. The results in Table 1 show averaged results for models that have a level of compression below a certain threshold (2.5 for MNIST and F-MNIST and 3.1 for CIFAR-10) in the final population.

A sample population from one of the runs can be seen it Appendix C, which showcases the variety of available models produced. The total time in the table is the time for the whole algorithm to do 1 run with the constraints specified above. As expected, the reconstruction and classification error is the lowest for MNIST. The average level of compression is around 1.7 (approximately a 4x4x3 representation), which means that the autoencoder managed to compress the data by a factor of almost **16** on average. The 85.8% accuracy for MNIST is not state-of-the-art level, but, remarkably, these architectures were discovered using only 20 generations with a population size of 20 for around 1.5 GPU hours on average. A more detailed look at one of the best performing models for MNIST is available in Figure 1. The same can be found for F-MNIST and CIFAR-10 in Figure 2.

Table 1: Summarised results averaged over 10 runs and all individuals with compression below 3.1 for CIFAR-10 and 2.5 for MNIST and F-MNIST dasets.

	Rec loss	LOC	Cl loss	Cl acc	Total time w/o ft (min)	Finetuning time (s)
MNIST	0.141	1.699	0.409	0.858	93.23	1160
F-MNIST	0.316	1.639	0.524	0.803	91.19	889
CIFAR-10	0.596	2.15	1.693	0.402	123.04	2752

Figure 1: MNIST autoencoder with bottleneck of 4x4x4 and reconstruction loss of 0.0817

Figure 2: Left: Fashion-MNIST autoencoder with bottleneck of 4x4x4 and reconstruction loss of 0.289. Right: CIFAR-10 autoencoder with bottleneck of 8x8x8 and reconstruction loss of 0.564

However, the fact that the algorithm managed to compress the dataset to representations that are (2x2x4) while still retaining around 50% of the data, shows that the approach is indeed promising. With larger population sizes, more epochs, more generations and expanding the search space, this approach can be scaled to larger datasets. Nevertheless, to scale the approach to multi-channel inputs remains a challenge (Figure 2).

Figures 1 and 2 present some of the best results obtained for all of the three datasets, showcasing the balance between compression and reconstruction loss. From the images, it can be seen that the MNIST reconstructions are hardly distinguishable from the originals. The results for F-MNIST are also still recognisable, but most of the details are lost during compression. Images for CIFAR-10, on the other hand, barely resemble the original images for the most part, which illustrates how adding the extra colour channels and complex objects increases the complexity of the problem, and hence limits the extent to which a dataset can be compressed in lower-dimensional representations.

Noteworthy, some of the objects in the last two images of 2 look similar. In the presented model the algorithm managed to compress the representations of these images by a factor of 12 for MNIST (Figure 1) and F-MNIST (Figure 2), while all useful information (except the sandal class for F-MNIST) seems to have been retained. As per CIFAR-10, the images were decreased by a factor of 6, while retaining some of the information, but classification performance dropped excessively.

5 Conclusion and Future Work

This paper presents a novel approach for automatically constructing convolutional autoencoders using neural architecture search, which utilizes neuroevolution to approximate the Pareto-front of solutions. While some of our results, especially for CIFAR-10, are still sub-optimal, this study is designed to be a stepping stone towards automating the process and making it considerably faster. Our goal is to speed up AutoML and allow decision-makers to articulate preference, offering them a set of models to choose from, instead of just a single one. We believe that the generations and population size we chose are insufficient to explore such an enormous search space. In future studies, we aim to scale the algorithm for significantly larger datasets and expand the search space available by adding new hyperparameters, as well as potentially achieving end-to-end neuroevolution. We are also interested in adding extra objectives in our multi-objective optimisation (e.g., explainability).

REFERENCES

Samet Akcay, Amir Atapour-Abarghouei, and Toby P Breckon. Ganomaly: Semi-supervised anomaly detection via adversarial training. In *Asian conference on computer vision*, pp. 622–637. Springer, 2018.

Samuel Alvernaz and Julian Togelius. Autoencoder-augmented neuroevolution for visual doom playing. In *2017 IEEE Conference on Computational Intelligence and Games (CIG)*, pp. 1–8. IEEE, 2017.

Arian Azarang, Hafez E Manoochehri, and Nasser Kehtarnavaz. Convolutional autoencoder-based multispectral image fusion. *IEEE Access*, 7:35673–35683, 2019.

Pierre Baldi. Autoencoders, unsupervised learning, and deep architectures. In *Proceedings of ICML workshop on unsupervised and transfer learning*, pp. 37–49. JMLR Workshop and Conference Proceedings, 2012.

Dana H Ballard. Modular learning in neural networks. In *AAAI*, pp. 279–284, 1987.

Francisco Charte, Antonio J Rivera, Francisco Martínez, and María J del Jesus. Evoaaa: An evolutionary methodology for automated neural autoencoder architecture search. *Integrated Computer-Aided Engineering*, (Preprint):1–21, 2020.

Joe Davison. Devol - deep neural network evolution, Feb 2017. URL `https://github.com/joeddav/devol`.

Alexey Dosovitskiy and Thomas Brox. Generating images with perceptual similarity metrics based on deep networks. *arXiv preprint arXiv:1602.02644*, 2016.

Thomas Elsken, Jan Hendrik Metzen, Frank Hutter, et al. Neural architecture search: A survey. *J. Mach. Learn. Res.*, 20(55):1–21, 2019.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

Lovedeep Gondara. Medical image denoising using convolutional denoising autoencoders. In *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, pp. 241–246. IEEE, 2016.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Gang Wang, Jianfei Cai, et al. Recent advances in convolutional neural networks. *Pattern Recognition*, 77:354–377, 2018.

Andreia P Guerreiro, Carlos M Fonseca, and Luís Paquete. The hypervolume indicator: Problems and algorithms. *arXiv preprint arXiv:2005.00515*, 2020.

Alexander Hadjiivanov and Alan Blair. Complexity-based speciation and genotype representation for neuroevolution. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, pp. 3092–3101. IEEE, 2016.

Xin He, Kaiyong Zhao, and Xiaowen Chu. Automl: A survey of the state-of-the-art. *Knowledge-Based Systems*, 212:106622, 2021.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. The cifar-10 dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html*, 55:5, 2014.

Vadim Lebedev and Victor Lempitsky. Speeding-up convolutional neural networks: A survey. *Bulletin of the Polish Academy of Sciences. Technical Sciences*, 66(6), 2018.

Y LeCun, C Cortes, and C Burges. The mnist dataset of handwritten digits (images). *NYU: New York, NY, USA*, 1999.

Zhichao Lu, Ian Whalen, Vishnu Boddeti, Yashesh Dhebar, Kalyanmoy Deb, Erik Goodman, and Wolfgang Banzhaf. Nsga-net: neural architecture search using multi-objective genetic algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 419–427, 2019.

Risto Miikkulainen, Jason Liang, Elliot Meyerson, Aditya Rawal, Daniel Fink, Olivier Francon, Bala Raju, Hormoz Shahrzad, Arshak Navruzyan, Nigel Duffy, et al. Evolving deep neural networks. In *Artificial intelligence in the age of neural networks and brain computing*, pp. 293–312. Elsevier, 2019.

Vinay Uday Prabhu and Abeba Birhane. Large image datasets: A pyrrhic win for computer vision? *arXiv preprint arXiv:2006.16923*, 2020.

Kaizhi Qian, Yang Zhang, Shiyu Chang, Xuesong Yang, and Mark Hasegawa-Johnson. Autovc: Zero-shot voice style transfer with only autoencoder loss. In *International Conference on Machine Learning*, pp. 5210–5219. PMLR, 2019.

Xu Qin and Zhilin Wang. Nasnet: A neuron attention stage-by-stage net for single image deraining. *arXiv preprint arXiv:1912.03151*, 2019.

Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pp. 4780–4789, 2019.

Shahin Rostami and Ferrante Neri. Covariance matrix adaptation pareto archived evolution strategy with hypervolume-sorted adaptive grid algorithm. *Integrated Computer-Aided Engineering*, 23 (4):313–329, 2016.

Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

David Martins Duarte Sereno. *Automatic Evolution of Deep AutoEncoders*. PhD thesis, Universidade de Coimbra, 2018.

Krishna Kumar Singh and Yong Jae Lee. Hide-and-seek: Forcing a network to be meticulous for weakly-supervised object and action localization. In *2017 IEEE international conference on computer vision (ICCV)*, pp. 3544–3553. IEEE, 2017.

Kenneth O Stanley, Jeff Clune, Joel Lehman, and Risto Miikkulainen. Designing neural networks through neuroevolution. *Nature Machine Intelligence*, 1(1):24–35, 2019.

Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pp. 6105–6114. PMLR, 2019.

Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A Efros. Dataset distillation. *arXiv preprint arXiv:1811.10959*, 2018.

Sumio Watanabe. A widely applicable bayesian information criterion. *Journal of Machine Learning Research*, 14(Mar):867–897, 2013.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Shen Yan, Yu Zheng, Wei Ao, Xiao Zeng, and Mi Zhang. Does unsupervised architecture representation learning help neural architecture search? *arXiv preprint arXiv:2006.06936*, 2020.

Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

A DATASETS

MNIST MNIST is a dataset containing 70000 images of handwritten digits in total, which are split among 10 classes. Each class represents an Arabic digit (0-9) and the images are 28x28 pixels and only have 1 colour channel, where the white value is specified as pixel intensity between 0-255 (LeCun et al., 1999).

F-MNIST Fashion MNIST is a similar dataset to MNIST with the same number of images as well as image shape, but the only difference is that the 10 classes are not digits, but instead 10 different pieces of clothing (Xiao et al., 2017).

CIFAR CIFAR is a labelled subset of the Tiny Images Dataset (Prabhu & Birhane, 2020). There are two main versions of the CIFAR dataset and these are CIFAR-10 (with 10 classes) and CIFAR-100 (with 100 classes) (Krizhevsky et al., 2014). The main difference with MNIST and F-MNIST is that this dataset is severely more complex and not only the images are 32x32, but they also have 3 (RGB) channels, so the total pixel values are 3072, compared to the 784 of the other two datasets.

B MONCAE NEUROEVOLUTION STEP DETAILS

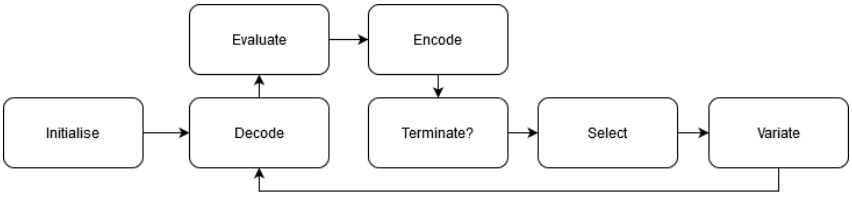

Figure 3: High-level flowchart of Neuroevolution with all steps from general evolutionary algorithms with Decode and Encode operators added. The Variation operator can be further split into Crossover and Mutation in the case of Genetic algorithms.

First, some algorithm hyperparameters are set. These are the number of generations, the population size, maximum number of convolutional (and pooling) layers, maximum filter size in a convolutional layer and the maximum epochs (used when each architecture is evaluated). The general steps of the algorithm follow the neuroevolution process depicted in Figure 3.

For the **initialisation** we create a random population (of the specified size). All individuals are by default **encoded** in a conveniently created python class, which stores the genome, which is composed of a long 2-dimensional list, which contains the information about all layers of the model (including a boolean value specifying if the layer is turned on or off, how many nodes/filters are there in the layer, what is the type of layer (convolutional, pooling, dense etc.), what activation function is the layer using, is there a batch normalisation following the layer, dropout etc.) as well as other hyperparameters, which include the optimiser the model is using and the learning rate. After the population is initialised, the individuals are **decoded** into TensorFlow models and are further **evaluated** by training for 5 epochs and then verifying the results by running the model on a predefined validation set that all individuals use. Here we tried various ways of reporting the fitness of each individual, but since we were aiming to minimise the latent vectors in the bottleneck as well as minimise reconstruction loss we chose to use the hypervolume indicator as a metric with the contributing hypervolume of each individual being its fitness score. So, the first objective is straightforward, which is the reconstruction loss, but for the size of the latent representations, we chose to name this "level of compression" and the way we calculate the level of compression is roughly based on Bayesian information criterion (Watanabe, 2013) (Equation 1). The $\mathbf{l}$ represent the latent representation as each element in the vector is the size of a dimension.

The next step, (**Terminate**), simply checks if the termination criterion is met and if it is, then the algorithm is stopped (in the context of these experiments, this criteria is the number of generations). For the **selection** a certain per cent of the best individuals survive and are kept in the next generation. For other individuals to be selected they have to have a higher fitness score than a random score

(which has a bound of 0 and the maximum score for the generation). The last step, before the cycle repeats, is to **variate** and the population and for variation, this approach uses both crossover and mutation, which is done on the selected individuals to form the next generation.

In our experiments, the reference points are set to 4 and 12 for the error and level of compression correspondingly. Thus, the algorithm might be biased to pick models with better compression than ones with lower error. In all training stages, if not otherwise specified, we used a batch size of 256 and trained for 20 epochs.

C SAMPLE POPULATION FOR MNIST

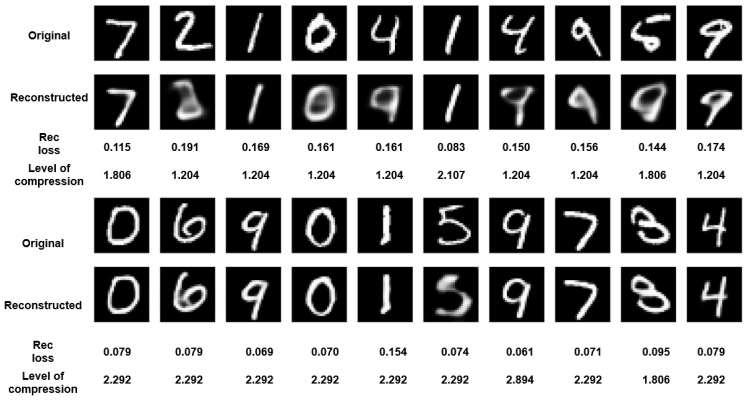

Figure 4: A population of solutions for MNIST. A single digit is used as example of the performance of each single model from a run picked from the authors. For each model an example of reconstruction of a single digit can be seen as well as the achived reconstruction loss and level of compression.

D USED SEEDS

[46, 29, 12, 8, 68, 44, 32, 91, 85, 61]