A Comparison of Deep Generative Models for Semi-Supervised Learning

David Hayes, Guenole Silvestre, Chris Bleakley
School of Computer Science
University College Dublin
Dublin, Ireland

***Abstract*—Deep generative models provide a plausible approach to learning semantic representations of large datasets. Such representations have potential applications in semi-supervised learning in the presence of limited quantities of hand-labeled data. This paper seeks to compare the semi-supervised learning capabilities of the most prominent generative models across large image datasets.**

I. Introduction

Deep neural networks have become an integral block of the modern machine learning pipeline, particularly in the field of computer vision. After training such models on massive datasets of image-label pairs, these networks easily adapt to a variety of visual tasks, achieving impressive results on image classification tasks on smaller data sets [1].

However, such results rely on a supervisory signal from a large-scale dataset of hand-labeled data. Relying on such a supervisory signal during training means ignoring much of the useful information present in the structure of the data itself. For datasets where only a handful of labeled examples present themselves, these results become significantly less impressive.

In addition, the process of increasing the number of labels present is generally not a simple task. The acquisition of hand-labeled data is expensive at best, and impractical at worst. In contrast, unlabeled data has become increasingly abundant. From both perspectives, a semi-supervised approach to classification appears to represent a more desirable approach.

In the proposed semi-supervised approach, a feature extraction network, or inference network, undergoes unsupervised pre-training using the entirety of the vast amount of unlabeled data present. This inference network, or encoder, learns a mapping from the pixel-space representation of an image to a more abstract representation, which captures the underlying structure of the data. The learned encoder's parameters can then be frozen and utilized as the basis of an image classification network, which is trained in typical supervised fashion, using the small number of labeled samples present. Such an approach promises superior classification results to purely supervised classifiers in the presence of such small quantities of labeled examples.

Learning such an inference network is an unsupervised learning problem. The classic approaches to feature learning in this regard are such algorithms as Principal Components Analysis [2] or clustering techniques, k-means [3] being one example. These methods fundamentally suffer from the fact that they capture a non-distributed representation of the data, where the features learned fail to interact on many different levels [4]. Such representations are too simple to capture the highly complex and hierarchical representations which characterize most large datasets.

The promise of deep learning is to discover rich, hierarchical models that represent probability distributions over such data as natural images [5]. Deep generative models propose a novel framework for learning a parametric encoder that maps to such a distributed representation, presenting a credible approach to extracting rich feature representations of complex data distributions, without the need for hand-labeled data.

These generative models have been successfully applied in many domains such as the generation of realistic images, the up-sampling of super-resolution signals and text-to-image synthesis. Given the success of these models in terms of sample generation, by intuition, it seems plausible that such models must learn a deep understanding of the many explanatory sources of variation present within the data. Deep generative models thus propose the unsupervised learning of semantic latent representations for large datasets.

II. Models for Comparison

Auto-encoder and adversarial frameworks are the most prominent approaches to generative modeling by deep neural networks. These approaches model the data distribution as a transformation of a fixed latent distribution.

A. Auto-encoders

Auto-encoders [6] have the purpose of reconstructing its input after it passes through a low-dimensional 'bottleneck' layer, with the aim of obtaining a compact latent representation. The encoder maps the inputs to latent representations while the decoder reconstructs the inputs from this compressed latent vector. The encoder and decoder can be trained together by minimizing reconstruction error, the discrepancy between the data and its reconstruction over training examples.

The auto-encoder variations that will be explored in this paper are:

1) Basic Auto-encoder: The simple framework described above. The difficulty with this simple model is found in extracting useful features, as the reconstruction process could simply copy the input or try to maximize mutual information, which could allow perfect reconstruction without learning a semantically meaningful representation.

2) De-noising Auto-encoder: In an attempt to combat this deficiency, de-noising auto-encoders [7], or DAEs, are trained to reconstruct a clean, 'repaired' input from a noise-corrupted version of the input. Essentially, it is learning a reconstruction function that corresponds to a vector field pointing towards high-density regions, i.e. the manifold where data examples concentrate. In general however, the corruption of DAE inputs does not require much semantic information to undo.

3) Adversarial Auto-encoders: Adversarial Auto-encoders (AAEs) [8] add an adversarial component to the typical training objective of auto-encoders in order to place a constraint on the distribution of the encoder representation being learned. The loss function of AAEs consists of a pixel-space reconstruction term, and a second term accounting for an adversarial loss determined by a discriminative model which learns to distinguish between samples from the learned encoder and the imposed prior distribution.

4) Variational Auto-encoder: Variational Auto-encoders (VAEs) [9], are auto-encoder models in which the latent space representation learned is in the form of a posterior probability distribution. The loss function of VAEs is similar to that of AAEs consisting of both a pixel-space reconstruction term and a term constraining its distribution. However, the latter term in VAEs is determined by the Kullback-Liebler divergence between the true posterior distribution and the inferred distribution. In both models, the parameters of a probability distribution modeling the data is learned rather than an arbitrary function, as is the case with other auto-encoders.

B. Generative Adversarial Networks

Generative Adversarial Networks (GANs) [5] lead the adversarial approach to generative modeling. A generator network learns to generate data from an arbitrary latent distribution, which resembles the distribution of the training dataset in question as closely as possible, while a discriminator network learns to distinguish between the real data and the generator data. The learning process of the generator and discriminator ideally continues until a convergence point is reached where the discriminator becomes unable to distinguish between the real data and that produced by the generator.

The GAN framework, however, lacks an efficient inference mechanism. While it proposes an approach to learning the mapping from the latent space to data, the inverse mapping, required for use in semi-supervised learning, is unclear. A number of proposed methods for learning this inverse mapping, with the addition of an extra deep network to the standard GAN topology, are considered here:

1) Latent Regressor: The latent regressor (LR) model, as suggested in [1], represents the simplest method of learning the encoder mapping, with the generator and discriminator trained in the traditional manner. Once this training is complete, the encoder model can be trained to minimize a simple loss function representing the difference between samples from the imposed prior distribution and their reconstructions using generator and encoder networks, the former's parameters of which are frozen.

2) Bi-directional GANs: Bi-directional GANs [1], or Bi-GANs, explore a method of learning the inverse mapping where both the generator and encoder networks train together, and in order to fool the discriminator, the encoder must learn the inverse mapping of the generator. The BiGAN discriminator differs from the traditional GAN discriminator in that it discriminates not only in the data space, but jointly in the data and latent spaces, where the latent component is either an encoder output or a generator input.

3) Partial BiGAN: This model, as suggested in [10], is a slight variation to the BiGAN approach in which the encoder is still trained in conjunction with the BiGAN discriminator but only after the generator model has first been fully trained in typical fashion, and its parameters held constant for the duration of the encoder's training process.

III. Results & Discussion

Across all datasets and modes of comparison, the same approach to choosing the architecture and training hyper-parameters of the models is followed. All models are implemented in Keras with TensorFlow back-end and trained using a TitanX GPU.

The complications involved in model training for the basic GAN framework is well-documented [11]. The additional encoder network only increases the difficulties encountered. For this reason, a BiGAN model that performs acceptably, in terms of generated images and image reconstruction, is first developed for the dataset in question. The encoder and decoder architectures from this model are then used for all other models in order to ensure as fair a comparison as possible. In the case of the other GAN models, the discriminators are modified to receive solely an image input, rather than a concatenation of image and latent vectors. A tanh activation is employed for the activation of the generator output layers for the GAN models, while a sigmoid activation is utilized in the auto-encoder models.

The hyper-parameters of the training process for each model are chosen to optimize the validation performance of the model with respect to its own performance metrics, e.g. minimal reconstruction loss for the simple auto-encoder. A Gaussian prior is used for all models which place constraints on the learned encoder mapping.

A. Semi-Supervised Learning

The usefulness of the latent representation learned by each model for classification, where the ratio of unlabeled to labeled examples available is small, is investigated. The dataset used is the MNIST dataset [12], a collection of 28x28 pixel images of hand-written digits. All images are gray-scale and labeled as a number between 0 and 9. MNIST contains 60,000 examples for training, with a further 10,000 available for test purposes. Obtaining a semi-supervised classifier involves two stages.

The first is the unsupervised pre-training of an encoder model. A fixed latent dimension of 100 is used for all models. In each case, the encoder network undergoes unsupervised training on the full training set of 60,000 examples. The

TABLE I: Semi-Supervised Classification Results for MNIST

Encoder Model	Labeled Examples					
	200	***500***	***1,000***	***2,000***	***5,000***	***10,000***
Basic AE	**72.43**	81.54	86.33	90.53	92.89	95.05
DAE	71.73	82.58	**87.49**	**91.03**	93.22	**95.10**
AAE	72.29	81.50	87.08	90.46	**93.36**	94.91
VAE	59.79	74.94	83.56	87.60	92.48	94.98
LR	72.26	82.34	87.32	89.41	92.14	94.58
BiGAN	70.74	81.67	86.97	89.64	92.78	94.89
Partial BiGAN	72.71	**83.29**	86.83	89.64	92.42	94.78
Fully-Supervised Classifier	72.13	81.96	85.87	90.46	93.12	94.82

learned encoder is then used as a feature extractor which can form part of a classifier.

The second stage involves the supervised training of a classifier on the feature vector learned by the encoder in question. The parameters of the encoder are frozen for this step. The classifier used is consistent for all models, a simple multi-layer perceptron (MLP) of two fully-connected layers with a hidden layer dimension of 256 and dropout of 0.5. The classifier is trained on a subset of the 60,000 images to minimize a categorical cross-entropy loss function, with a training/validation split of 4:1 employed. A training limit of 100 is placed on the number of epochs and training is stopped early if no decrease in validation loss is observed over the course of 10 epochs. A batch size of 100 is used, along with an Adadelta optimizer.

The performance of each semi-supervised classifier is measured by its classification accuracy on the set of 10,000 test images. The number of examples available from the training set for supervised training of the classifier is varied and the performance of each model is observed. The final classification accuracy figures quoted in Table I are the average test accuracy values observed for 10 separate random initializations of the MLP. While covering all models mentioned in this paper, this table also contains classification accuracy figures for a fully supervised classifier, with the same encoder architecture as the unsupervised models, with the modification that its encoder is randomly initialized and trained fully supervised in conjunction with the MLP. This is done in order to show the merits of semi-supervised training over purely supervised training.

The highest test accuracy obtained for each subset of the full training set is highlighted in Table I. From these results, a few observations are made:

- In all cases, the strongest semi-supervised classifier outperforms the fully supervised classifier, highlighting the merits of the former approach.
- Of the semi-supervised classifiers, auto-encoder models exhibit the strongest levels of performance, accounting for the highest accuracy figure for all but one of the subsets of training data.
- The DAE is the strongest performing of all models, especially in the subset sizes of 1,000 and 2,000, which are probably more indicative than the end values of 100 and 10,000.
- There does not appear to be any clear best performer across the GAN variants.

B. Latent-Space Visualization

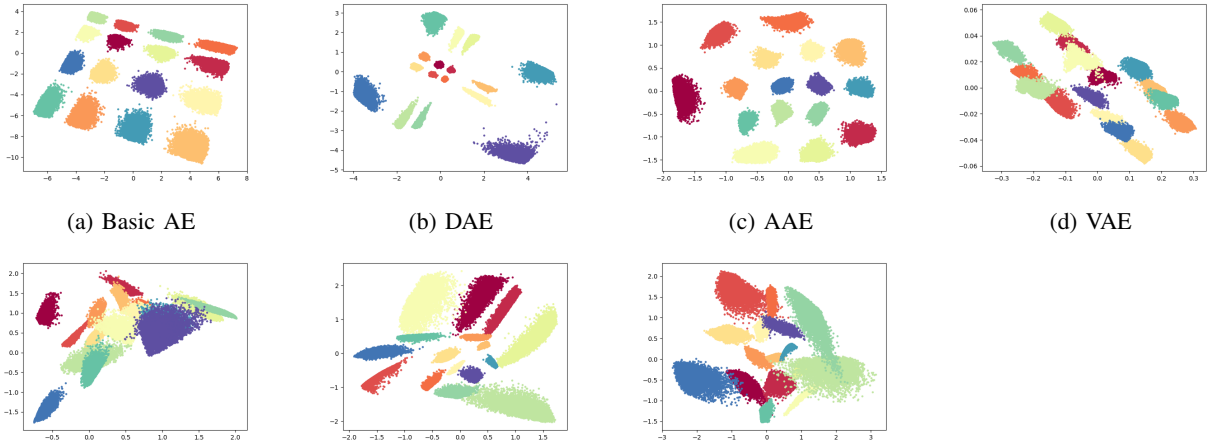
In general, for large datasets it is very difficult to visualize the latent representation. As large datasets have extremely complex underlying distributions, a 2D or 3D latent space cannot realistically capture the semantics of the entire dataset. For example, even a relatively simple dataset like MNIST requires a dimension of approximately 100 in order to capture a somewhat semantic representation.

For this reason, a synthetic dataset is created that can be semantically captured by a 2D latent space and visualized to provide further insight about how various models structure their latent space. The dataset consists of 2x2 gray-scale images, spanning 16 classes. The training set consists of 80,000 examples, with approximately 5,000 examples per class. All examples are distorted by 0.1 Gaussian noise. Fig. 1 shows the 2D latent space produced by the learned encoder models for the full set of training data. Each color corresponds to a particular class label from the dataset.

In terms of the quality of the latent space, there are two important indicators. The first being that examples are cleanly clustered with other examples of the same label, with the clusters of similar classes positioned closer together. A second is how well the latent space is covered, which determines its quality in terms of its use as a generative model, where the latent distribution can be sampled from to produce new examples. Large gaps in the latent space are undesirable in this regard. From this perspective, it is important to note the scale on each axis.

Some observations of the latent spaces are as follows:

- All models apart from the basic AE and DAE have a training objective that in some way prompts its latent space to approximate a Gaussian distribution, hence the reason that their resulting spaces are much less sparse.
- The basic AE generates a latent space that is very well clustered. However, the latent space is very sparse and large holes are present between clusters, and hence it is not useful as a generative model.
- The DAE latent representation is similarly well-clustered to the basic framework. It suffers from the same difficul-



(a) Basic AE (b) DAE (c) AAE (d) VAE

(e) LR (f) BiGAN (g) Partial BiGAN

Fig. 1: Visualization of 2D Latent Spaces for Synthetic Dataset

ties of the presence of holes in the latent space. In fact, it is the sparsest representation of all models considered.

- The AAE approximates its Gaussian prior and structures its clusters very cleanly. It leaves large gaps in the latent space, however, more so than other models considered, with similar Gaussian prior distributions.
- The VAE structures its latent space well, but there is overlap between class clusters observed.
- The LR does not produce a strong latent space from a visual point of view. It firstly struggles to recover a Gaussian distribution. In addition, it is weak at clustering according to labels, with many instances of inter-class overlap observed.
- The partial BiGAN recovers the Gaussian prior well while also demonstrating strong coverage of the latent space. This comes at a cost however, as there is evidence of cluster overlap in more than one instance.
- The BiGAN exhibits a strong latent representation, displaying a neat balance between latent space coverage and homogeneous clusters, while also making a fair approximation of a Gaussian distribution.

In terms of generative modeling, the BiGAN probably presents the strongest latent space structure. However, it is interesting to note that the models which show apparent deficiencies in this regard, in terms of leaving holes in the latent space are the strongest performing in the semi-supervised tasks, the DAE being the best example of this.

IV. Conclusion

The merits of a series of the most prominent deep generative models for semi-supervised learning were compared in terms of classification accuracy on large image datasets in the presence of small numbers of hand-labeled examples. The latent space learned by the models in question was visualized in 2-dimensional space, utilizing a synthetic dataset of simple images.

Deep generative modeling is an area of much development, with numerous recent papers claiming improvements on existing models and plausible new models. In future work, a similar analysis on such models can be performed to test their semi-supervised learning credentials, while testing can be extended to multiple datasets.

REFERENCES

[1] Donahue, J., Krhenbhl, P. and Darrell, T., 2016. Adversarial feature learning. ArXiv preprint arXiv:1605.09782.

[2] Tipping, M.E. and Bishop, C.M., 1999. Probabilistic principal component analysis. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 61(3), pp.611-622.

[3] Hartigan, J.A., 1975. Clustering algorithms.

[4] Bengio, Y., Courville, A. and Vincent, P., 2013. Representation learning: A review and new perspectives. IEEE Transactions on Pattern Analysis and Machine Intelligence, 35(8), pp.1798-1828.

[5] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y., 2014. Generative adversarial nets. In Advances in Neural Information Processing Systems (pp. 2672-2680).

[6] Hinton, G.E. and Salakhutdinov, R.R., 2006. Reducing the dimensionality of data with neural networks. Science, 313(5786), pp.504-507.

[7] Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y. and Manzagol, P.A., 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. Journal of Machine Learning Research, 11(Dec), pp.3371-3408.

[8] Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I. and Frey, B., 2015. Adversarial autoencoders. ArXiv preprint arXiv:1511.05644.

[9] Kingma, D.P. and Welling, M., 2013. Auto-encoding variational bayes. ArXiv preprint arXiv:1312.6114.

[10] Dumoulin, V., Belghazi, I., Poole, B., Mastropietro, O., Lamb, A., Arjovsky, M. and Courville, A., 2016. Adversarially learned inference. ArXiv preprint arXiv:1606.00704.

[11] Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A. and Chen, X., 2016. Improved techniques for training gans. In Advances in Neural Information Processing Systems (pp. 2234-2242).

[12] LeCun, Y., 1998. The MNIST database of handwritten digits. http://yann.lecun.com/exdb/mnist/.