

Symbolic Autoencoder

Oryan Barta

Abstract

Developing effective unsupervised learning techniques is an essential stepping stone towards next generation machine learning models. Such models would no longer be bottlenecked by their dependence on massive labeled datasets which are often difficult or impossible to obtain. We propose a novel architecture for deep feature extraction from unlabeled data and intelligent labeling of data using an implicitly defined and learned symbolic language. The model can then be used in a semi-supervised context to reduce the amount of labeled data necessary for training.

1 INTRODUCTION AND BACKGROUND

Given some distribution \mathcal{D} over vectors $x \sim \mathcal{D}$, the task of an unsupervised model is to implicitly or explicitly find some approximation p_{model} to the density function $p_{\mathcal{D}}$ given some subset $S \subset \mathcal{D}$. Such a model can arguably be said to understand the intrinsic structure of the data domain \mathcal{D} .

1.1 UNSUPERVISED VERSUS SUPERVISED MODELS

In supervised learning, labels $y \sim \mathcal{L}$ are imposed on the vectors $x \sim \mathcal{D}$. These labels come from some artificial distribution \mathcal{L} with density function $q(x) = q(y(x))$, and ideally carry some human learned information about the deep structure of the vectors x . More formally, we hope the cross entropy between p and q is low, where we define this value as:¹

$$H(p, q) = - \sum_{x \in \mathcal{D}} p(x) \log q(x)$$

Low values of H means the structure of density function q of labels is similar to the structure of density function p of data points. The supervised model M is given tuples (x, y) from the joint distribution $(\mathcal{D}, \mathcal{L})$ and is now tasked with arriving at some density function p_{model} which minimizes $H(p_{model}, q)$.

The problem of minimizing $H(p_{model}, q)$ is usually easier than the unsupervised task of minimizing $H(p_{model}, p)$, as \mathcal{L} is a more abstracted domain than \mathcal{D} , however sampling (x, y) is usually restrictively expensive and thus supervised learning tasks are often limited or impossible. It is also possible that the density q is not a good proxy for p , and therefore unsupervised

models which are unshackled to preconceived human notions embedded in q can gain a better understanding of \mathcal{D} than their supervised counterparts.

1.2 OVERVIEW OF COMMON UNSUPERVISED MODELS

Generally unsupervised models come in one of two forms: explicit and implicit models.

Explicit models attempt to directly compute p_{model} . For example, PixelRNN or PixelCNN attempt to directly compute p_{model} using the formula $p_{model}(x) = \prod_{i=1}^n p(x_i | x_1, \dots, x_{i-1})$ for x_i being pixel i in an image, explicitly interpreting an image as a statement over a language of length 256 (by first quantizing pixel values) and then training an RNN (recurrent neural network) to learn this language. Note that no labels are required for training: the model objective is simply predicting the next pixel given all previous pixels.

Implicit models, on the other hand, attempt to implicitly encode p_{model} in the model weights without defining any tractable way of directly computing p_{model} for any given sample. This is done by training a generator function which learns to sample $x \sim p_{model}$. Examples of such models include autoencoders, variational autoencoders (VAEs) and generative adversarial networks (GANs) among others.

¹ For continuous distributions, $H(p, q) = -E_{x \sim p}(\log q) = H(p) + D_{KL}(p \parallel q)$, where D_{KL} is the Kullback-Leibler

divergence $D_{KL}(p \parallel q) = \int_{-\infty}^{\infty} p(x) \log \frac{p(x)}{q(x)} dx$ and $H(p)$ is the entropy of p , $H(p) = - \int_{-\infty}^{\infty} p(x) \log p(x) dx$

1.2.1 Naïve Autoencoders

Autoencoders work by trying to reconstruct images using a naïve l_2 loss between the input image and the generated image. However, they tend to be unable to learn deep features, instead blurring the input in inverse proportion to the feature vector size. The failure of naïve autoencoders can arguably be attributed to an ineffective metric, as a pixelwise comparison is clearly not optimal for learning deep features which are largely invariant to pixelwise changes.

1.2.2 Variational autoencoders

Instead of comparing image pixels directly, VAEs compare them through a mask provided by several moments of some chosen distribution. In other words, they attempt to directly learn the parameters of some distribution (artificially chosen, for example the two moments of the normal distribution), over the image pixels as they relate to a similarly learned distribution over the feature vector. Although an improvement, they still tend to produce blurred images with few sharp features, again due to the limitations of the chosen distribution which contains far less moments than necessary to sufficiently represent the image domain, and therefore an ineffective pixelwise comparison is that which ultimately drives training. See fig 1.

1.2.3 Generative adversarial networks

GANs [1] currently represent the cutting edge in the field of unsupervised learning. GAN training is in fact a minimax game between two players called the discriminator ($D: \mathbb{I} \rightarrow \{0,1\}$) and generator ($G: \mathbb{R}^n \rightarrow \mathbb{I}$), each usually implemented as several convolutional or deconvolutional layers respectively [2]. The generator produces images from some randomly sampled vector z , and the discriminator must learn to discriminate between the fake images produced by G and real images. The generator then learns to produce increasingly realistic images in an attempt to fool the discriminator, and thus both networks are driven to learn deeper features of the domain by this governing adversarial objective.

GAN networks, unlike autoencoders and VAEs, can produce remarkably sharp images approaching photorealism on some domains (see fig. 3), however these models exhibit significant problems including unstable training with no convergence measure, mode collapse problems (inability to deal with multiple modes in the data), and uninvertibility: in other words, GAN's define a method to train a decoder but not its matching encoder, arguably the more useful side of the transformation. As a result, GANs are essentially able to

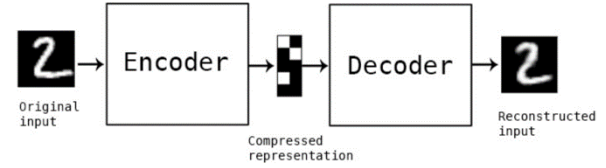


Figure 2 Blurry results from VAE

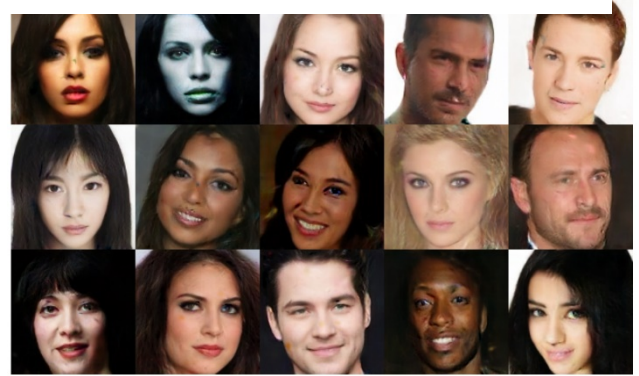


Figure 3 State of the art results from GAN

produce impressive artwork, however, without the inverse transform they lack clear practical application.

2 RELATED WORK

Hundreds if not thousands of papers have been published attempting to tackle these issues. PPGN [3] (plug-n-play) models introduce “conditioning networks” and use activity maximization techniques to get a handle on generator outputs. WGAN [4] attempts to improve stability and reduce GAN dependence on batch normalization by modifying the loss to effectively minimize the more continuous “earth mover” distance instead of D_{KL} divergence. AEGAN [5] and BiGAN [6]

and many others attempt to learn the inverse mapping in diverse and often complicated ways. InfoGAN [7] attempts to impose some interpretability on the latent space mapping by forcing the generator to express certain elements of its latent input. BEGAN [8], the most similar model to the currently proposed model, defines the discriminator as an autoencoder and by adding regulating constraints to the loss successfully offers a convergence measure and stabilizes training.

3 PROPOSED METHOD

We propose a relatively simple solution which resolves some of these issues.

The proposed model develops an internal and implicitly defined language which converges to represent the given data domain. The language consists of symbols represented as vectors on the n –sphere, where we interpret this mapping onto the sphere as a language due to its gradually being learned by two disjoint systems tasked with communicating between them – in other words with inventing a language that best represents encountered stimuli.²

An objective function is defined which encourages the two systems to agree on similar stimuli (images) while disagreeing on dissimilar stimuli. For each data sample, each system produces a symbol $s_{1,2} \in S_n$, and in essence they attempt to minimize $\langle s_1, s_2 \rangle$. We call this the *syncretic objective* between the two *units* of the bicameral network or *bi-net*,

$$B = (D_{\theta_1}: \mathbb{I} \rightarrow \mathbb{R}^n, D_{\theta_2}: \mathbb{I} \rightarrow \mathbb{R}^n)$$

where θ_i denotes trainable parameters.

In tandem a single generator function $G_{\theta_G}: \mathbb{R}^n \rightarrow \mathbb{I}$ produces fake stimuli, and an *adversarial objective* drives deep feature extraction using a mechanism similar to the GAN. The generator produces two symbols $\tilde{s}_{1,2} \in S_n$ by producing two fake images and running them through each unit independently, with the objective of minimizing $\langle \tilde{s}_1, s_2 \rangle$ and $\langle \tilde{s}_2, s_1 \rangle$ respectively, while the bi-net has the opposite objective. The input to the generator is the symbolic output of the bi-net $s_{1,2}$ from which it produces $\tilde{s}_{2,1}$. This marks a significant shift from GANs in that no noise is required for training. The

cyclic input ensures that the input language and output language become identical, in other words at convergence we expect:

$$D_i = G^{-1}$$

In figure 4 (next page) a high level schematic is shown of the model computation, including syncretic and adversarial forces.

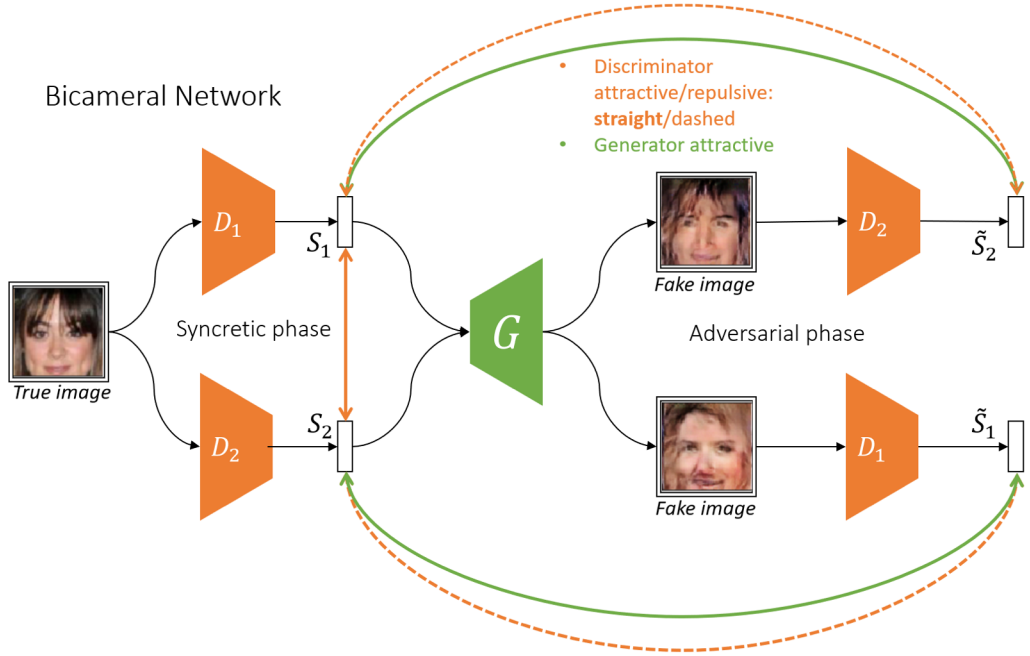
The combination of adversarial and syncretic training constitutes a new paradigm which we call the *symbolic autoencoder* (SAE). A possible interpretation is as follows: In order to produce a label for a given image, each unit simply assumes the other unit knows the label, and adversarial training proceeds under this assumption. These complementary assumptions are expressed in the figure by the “crossed” pathways of the symbols (note that D_1, D_2 exchange places between the first and second half of the figure). We find that this arrangement is necessary for stable convergence and best results.

3.1 COMPARISON WITH GAN

SAE relies on principles of GAN to drive generalized and deep feature extraction, however while the GAN model is defined such that the encoder (discriminator) network is degenerate – meaning it outputs only a single number and therefore lacks utility outside of training – the SAE allows simultaneous training of both the decoder and encoder. Ideally SAE should be able to extract most if not all deep features from the data during unsupervised training, and then be composed with a thin

² We allow that “language” is simply a set of abstract objects (symbols) which are used both as proxies for stimuli and as conventions facilitating communication.

Figure 4 Bicameral network computation: A true image is run through the two discriminators which produce normalized vector outputs $s_i \in S_n$. These are then used as seeds for two fake images produced by the generator, which are then further passed into the two units (in a crossed fashion – necessary for stabilization) in order to produce two further symbols $\tilde{s}_{1,2}$. Syncretic and adversarial forces then act on the produced symbols to drive training.



network which would learn from a drastically reduced set of labeled data how to translate the internal language to human readable form.

SAE also presents an effective and elegant solution to the “mode collapse” problem plaguing GAN networks, which makes it difficult or impossible for GANs to converge on multimode datasets.³ The dimensionality expansion of the SAE discriminator unit allows diverse loci of adversarial training to stabilize on the n –sphere, enabling network nonlinearities to more effectively mask and disentangle gradient flow from different modes of the input (explained more in the section “disentanglement of modes” below).

3.2 COMPARISON WITH VAE

Unlike other autoencoders such as the variational autoencoder (VAE), at no point in the SAE loss are image pixels compared. This is significant in that while other autoencoders tend to blur stimulus in proportion to the size of their information bottleneck (the encoded feature vector), SAE tends to *generalize* stimulus. This is because the Euclidean loss between image pixels causes VAE to prefer *shallow features* (precise positioning of pixels) over *deep features* (the object being reconstructed).

3.3 BICAMERAL ASYMMETRY: “LEFT AND RIGHT BRAIN”

One of the basic ideas behind SAE is that the “optimal” language, meaning a language with symbolic structure and interrelationships that most accurately reflect a particular domain, might be defined by an equilibrium between two opposing intellectual forces: that of generalization and that of diversification. If the symbols produced are either too diverse or too general, corresponding to the identity or zero transform respectively, it could be argued that no deep feature extraction took place. Therefore, the two systems mentioned above divide the work between them: one is responsible for learning general features and one for learning specific features of the data, and in attempting to communicate with the other each serves as a stabilizing and moderating force, preventing the system as a whole from degeneration.

This idea is similar to that used by the Boundary Equilibrium GAN (BEGAN) [8] mentioned above, which uses proportional control theory to balance these two forces and stabilize convergence. Our system does not require the additional “adaptive term” used in BEGAN, as instead the generator can strike this balance

³ For example a dataset of faces is relatively unimodal in contrast to a dataset of handwritten digits, as changes

between faces are relatively continuous compared to changes between different digits.

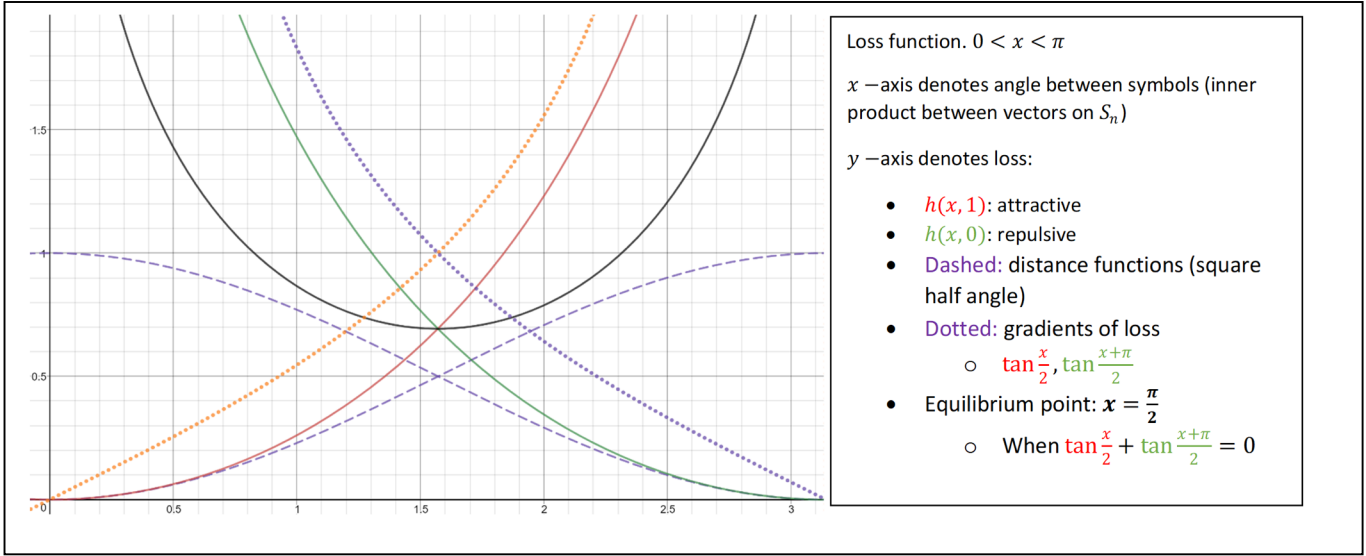
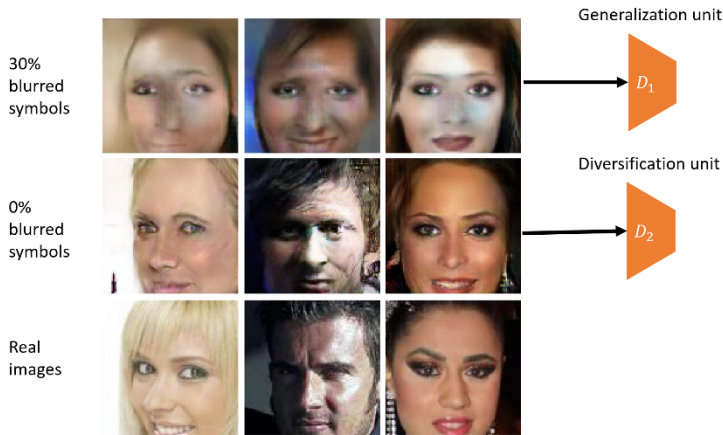


Figure 5

naturally: if it fails on one unit more than the other, the gradients coming from that unit will be greater causing G to prefer to correct itself in the quality it lacks.

3.4 SEMANTIC BLURRING

To support the generalization process, we add white noise to the symbol seeding the generalizing unit's fake image. This causes an effect we call *semantic blurring*: the generator learns to associate lower frequencies in symbol space with more general features in image space. With faces, for example, blurring the symbol causes the face produced to lose shallower features, such as hairstyle, but retain deeper features, such as gender. This noise is also helpful to stabilize training. This feature is still in infancy – there is much here that needs to be developed.



3.5 LOSS FUNCTION

The base function for our loss is the sigmoid cross entropy on the square half angle formula, identical to the GAN loss though instead of sigmoid arguments we use a scaled inner product on S_n :

$$h(x, p) = -p \cdot \log\left(\frac{1+x}{2}\right) - (1-p) \cdot \log\left(\frac{1-x}{2}\right)$$

The argument $\frac{1+x}{2}$ is the square of the half angle formula given $x = \cos \theta$,⁴ which takes values in $[0, 1]$.

The formula is illustrated in fig. 5.

3.5.1 The Syncretic Objective

In figure 6 can be seen a map of the symbolic sphere (the mapping D_1, D_2 onto S_3) as it trains on 3 modes of mnist, color coded according to digit type. The inner and outer rings are samplings from D_1 and D_2 respectively. The figure illustrates how different modes naturally disentangle during training.

The syncretic objective is made of three components: the *shell loss*, the *kernel loss*, and the *self loss*. The latter two losses are necessary for maintaining the concentric ring structure, which we found to be essential to stabilize and regularize training, and for the need to find an equilibrium between generalizing and diversifying forces explained above. The *self loss* ensures that one unit constricts its output and the other extends its output, while the *kernel loss* ensures that the units are

⁴ $\cos^2\left(\frac{\theta}{2}\right) = \frac{1+\cos \theta}{2}$

concentric. The *shell loss* then causes D_i to try and agree on identical images. This causes each meridian on the sphere, or radii on the plot, to be a site of adversarial training on a particular image.

3.5.2 The Adversarial Objective

The objective of G is to cause $D_1(G(s_2)) = \tilde{s}_1 \rightarrow s_2$, and $D_2(G(s_1)) = \tilde{s}_2 \rightarrow s_1$, in other words to produce images which cause each unit to produce symbols approaching the symbol the real image would cause each to approach. Note that the generator is restricted to the mapped domain of each unit respectively, so it cannot force the discriminator to produce a symbol it hasn't mapped (this is a consequence of the continuity of gradient descent optimization). So the objective of G is to produce the *closest* symbol in D_1 's range to s_2 (and vice versa). This symbol is naturally s_1 , as this is the symbol D_1 tries to bring closer to s_2 (and s_2 in the other unit's case).⁵ D_1 's adversarial loss component then causes it to try and disentangle s_1 from \tilde{s}_1 (as their gradients point in opposite directions – one towards s_2 and the other away). To do this it needs to learn features which differentiate between the real and fake image. As the fake image becomes increasingly realistic, it will need to extract increasingly deeper features to successfully differentiate.⁶ Note that the importance of the shell loss is in providing a stable *direction* for these opposing gradients, the direction being dependent on the *mode* of the data (discussed below), and we don't expect the shell loss to necessarily decrease over time as the symbols are constantly pulled back and forth as the generator and discriminator in turn learn deeper features.

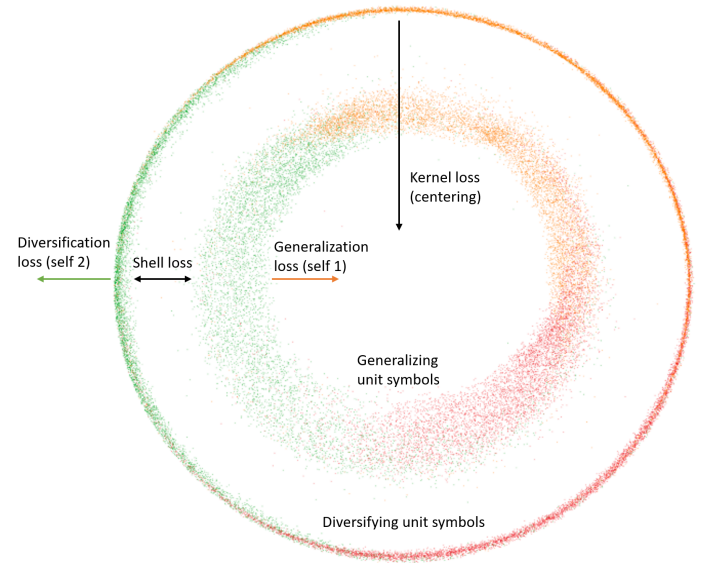


Figure 6 Projection (lambert azimuthal aligned to the center of mass of the first unit) of the symbolic sphere S_3 ; the generalizing unit and diversifying unit are the inner and outer rings respectively. Attractive force between the rings combined with adversarial training drives feature extraction simultaneous with mode disentanglement.

3.5.3 Disentanglement of Modes

At first, the only modes the system need learn are “real” and “fake,” similar to the original GAN model: The discriminators can use simple features which do not differentiate between modes in the real distribution. However, as training progresses and the generator begins producing more realistic images spanning multiple modes,⁷ the discriminators are forced to begin discriminating not just between the real and fake domains but within the real domain itself: in other words, to begin labeling the different modes of the data. D 's objective is to pull apart real and fake symbols, however if the fake symbols are close enough to real symbols than as a side effect it also learns to differentiate real from real. This leads to clustering of the data around the surfaces of two concentric spheres of dimension $m - 1$ (m being the symbol dimension), which can be seen in figure 6 with $m = 3$. The clustering is done using ever deepening features of the data, perhaps making it possible to refer to this process as “automatic labeling.”

⁵ The reason we use this “crossed” objective and don't simply set G 's objective to be s_1 (or s_2 for the second unit) is for stability reasons: such an objective would cause s_1 to be constantly “running away” from \tilde{s}_1 , which is undesirable.

⁶ The limit of this process is in the size of the symbol, or in the degree of noise masking the generator input.

⁷ Note that G has sufficient domain to span this range of modes even without noise input as s_2 (the seed of \tilde{s}_1) is more spread out than s_1 by the diversification loss (second unit's self loss).

3.6 ALGORITHM

Let $I \sim \mathcal{D}$ be a batch of size n of sampled images, and $D(I) = s \in \mathbb{R}^{n \times m}$ where m is the symbol dimension.

Let $\theta_{D_{0,1}}, \theta_G$ be trainable parameters of the differentiable functions $D_{0,1}, G$ described above.

Note also that $s \in S_m$, so $\langle s_1, s_2 \rangle = \cos \theta_{12}$.

Step by step explanation of iteration:

1. Sample true symbols
2. Sample fake images
3. Sample fake symbols.
4. True intra-unit correlation – attractive & repulsive
5. True inter-unit correlation – attractive only
6. Fake inter-unit correlation – repulsive
7. Fake inter-unit correlation – attractive
8. Discriminator intra unit self loss for specialization of units
9. Discriminator inter unit attractive kernel loss for centralization of units
10. Discriminator inter unit repulsive kernel loss for adversarial training
11. Discriminator elementwise inter unit loss for syncretic training
12. Generator elementwise inter unit loss for adversarial training
13. -16. Sum the losses and update weights.

	For $t = 0 \dots \infty$:
	$s_{0,1} \leftarrow D_{0,1}(I)$
1	$\tilde{I}_{0,1} \leftarrow G(s_{0,1})$
2	$\tilde{s}_{0,1} \leftarrow D_{0,1}(\tilde{I}_{1,0})$
3	$A^i \leftarrow h(s_i^T s_i, i)$
4	$B \leftarrow h(s_1^T s_2, 1)$
5	$C^i \leftarrow h(s_j^T \tilde{s}_i, 0)$
6	$\tilde{C}^i \leftarrow h(s_j^T \tilde{s}_i, 1)$
7	
8	$l_{self}^k \leftarrow \frac{1}{n^2} \sum_{ij}^n A_{ij}^k$
9	$l_{kernel} \leftarrow \frac{1}{n^2} \sum_{ij}^n B_{ij}$
10	$\tilde{l}_{kernel}^k \leftarrow \frac{1}{n^2} \sum_{ij}^n C_{ij}^k$
11	$l_{shell} \leftarrow \frac{1}{n} \text{tr}(B)$
12	$\tilde{l}_{shell}^k \leftarrow \frac{1}{n} \text{tr}(\tilde{C}^k)$
13	$l_B \leftarrow l_{shell} + l_{kernel} + \sum_{k \in \{0,1\}} l_{self}^k + \tilde{l}_{kernel}^k$
14	$l_G \leftarrow \sum_{k \in \{0,1\}} \tilde{l}_{shell}^k$
15	$\theta_{D_k} \leftarrow \theta_{D_k} - \mu \frac{\partial l_B}{\partial \theta_{D_k}}$
16	$\theta_G \leftarrow \theta_G - \mu \frac{\partial l_G}{\partial \theta_G}$

4 IMPLEMENTATION

Implementation and testing were done using Tensorflow for python. We pulled from the DCGAN implementation as a basic framework to build on. D_i and G were implemented as 5 layer convolutional and deconvolutional networks with kernel size 5 and stride 2, identical to DCGAN. Relu nonlinearities were used except for the final activation on D_i where we used l_2 normalization. Batch normalization was used to stabilize gradient flow, though we noted this was not strictly necessary (in contrast to the DCGAN model). ADAM optimizer was used for the update step with $\beta_1 = 0.5$, $\mu = .0002$.

4.1 TESTING

We tested the model on 3 different datasets: mnist, celebA, and music notation. We observed faster convergence by several orders of magnitude over DCGAN on mnist, and the ability to generate near photorealistic images approaching state of the art on the celebrity faces dataset.

We found that within 5 minutes (a few thousand iterations) our model exceeded the quality of DCGAN after many hours of training (over 100,000 iterations).

We also found much faster convergence on celebA dataset, with image quality approaching photorealism even on large image sizes (108×108), another difficulty for the original GAN model (although BEGAN resolved this issue, but using a much larger architecture than DCGAN and ours).

Results can be found below.

5 CONCLUSION & NEXT STEPS

The SAE model although built on GAN is more different than similar to its antecedent: it is not dependent on noise input, meaning the basic GAN paradigm of latent space mapping does not apply, and the main product (in terms of usability) of the SAE trainer is an encoder instead of a decoder. The main use of SAE is to resolve the need for copious quantities of labeled data in neural network training, however it is unclear if this goal was achieved as the model has not yet been tested in a semi-supervised setting; this would be an immediate next step.

There is much work yet to be done: developing a mathematical understanding of the dynamics which arise

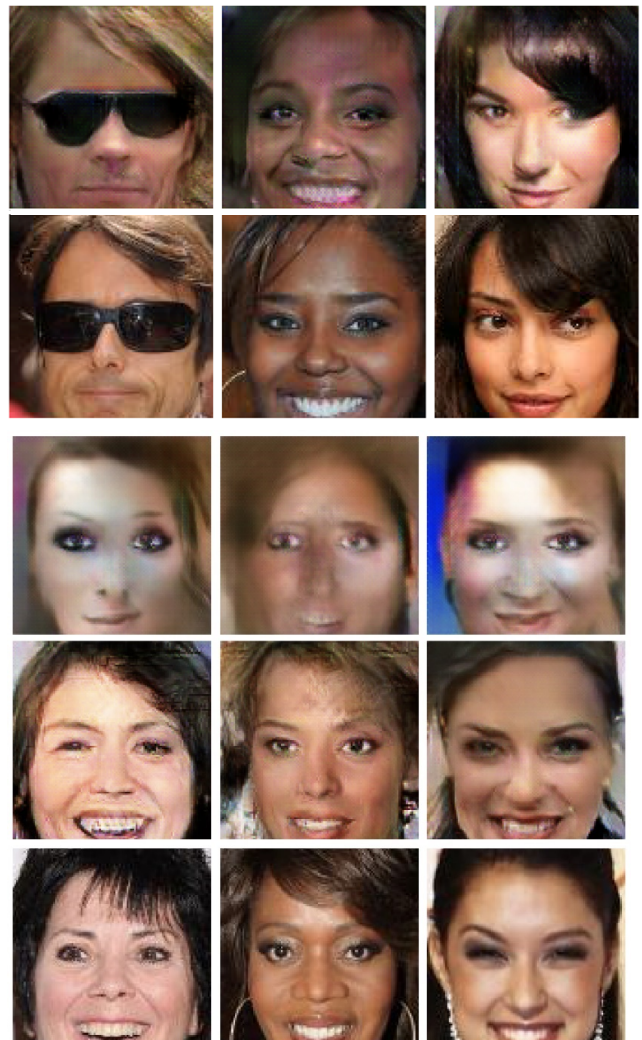
during training, stabilizing the “semantic blurring” technique and arriving at a convergence measure.

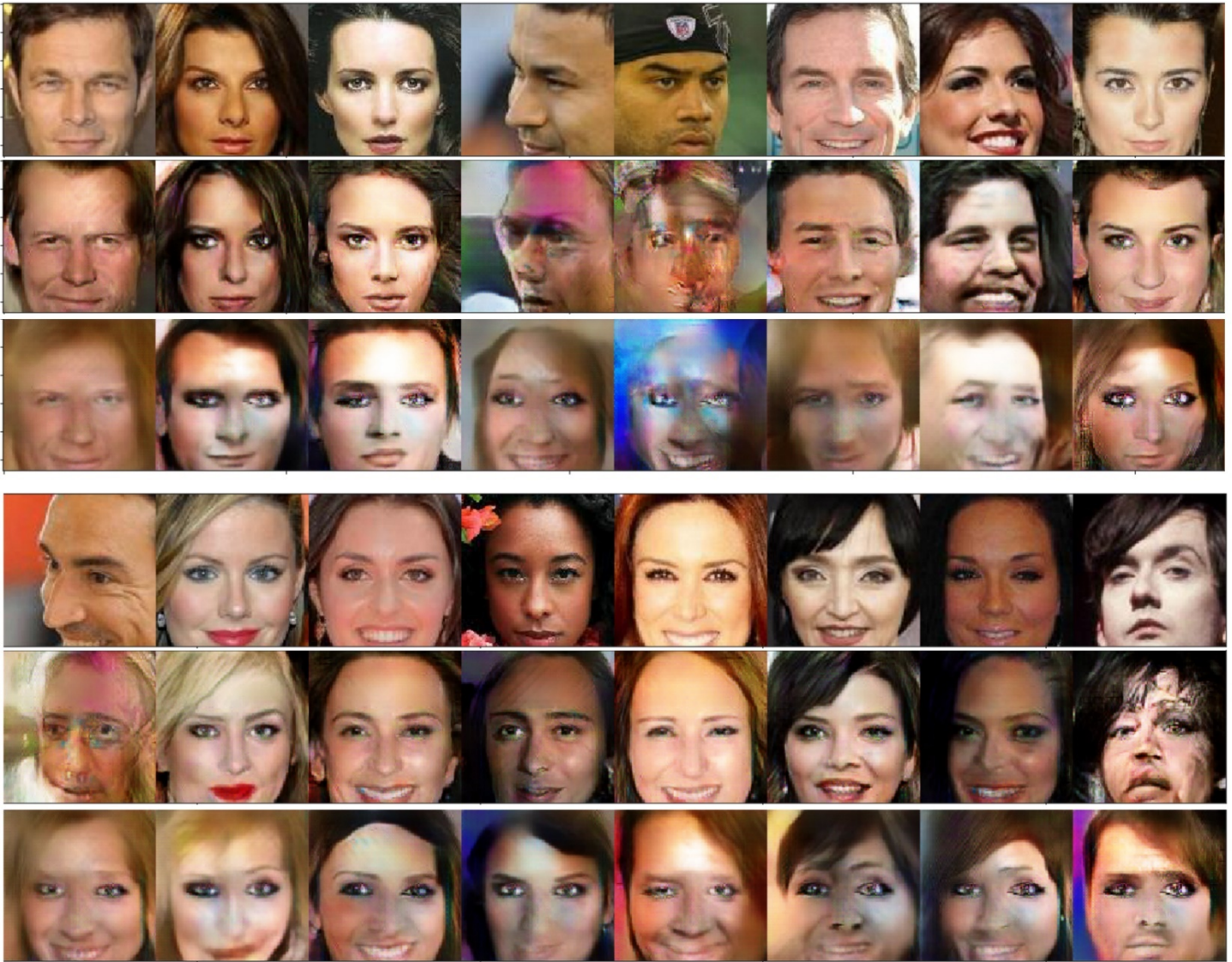
Finally, if we posit that our model constructs an implicitly defined language, a natural evolution of the model would involve grammar, in other words a state machine. We believe understanding how to incorporate an RNN into the computation is the next overarching step for this project.

6 RESULTS

6.1 CELEBA RESULTS

Note how images produced retain *general* features (such as hair style, skin color, facial expression) without retaining shallow features (such as precise positioning of hair or number of wrinkles etc.) in contrast to other autoencoders which tend to blur images to minimize pixelwise l_2 loss between images:


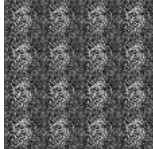

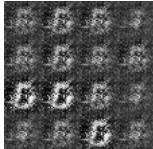
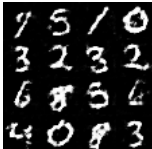
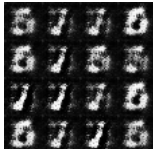

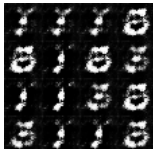


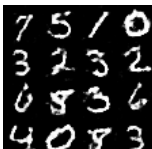

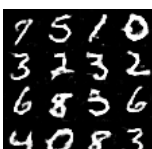
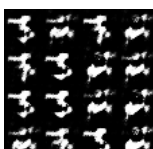
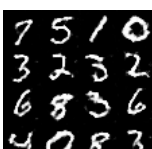
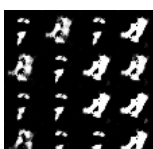




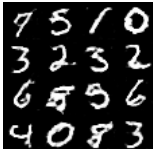
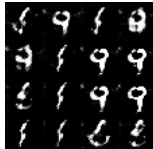


6.2 MNIST RESULTS



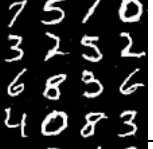

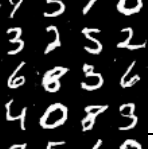

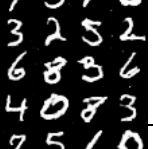

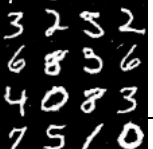



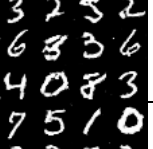
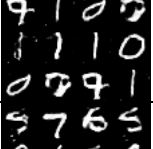
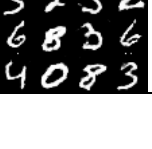
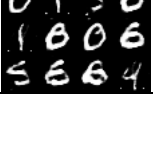
Mnist training results progress comparison:



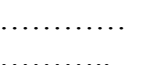
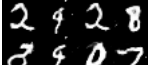
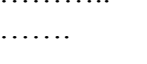
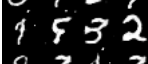
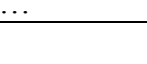

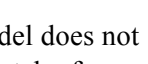
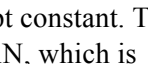
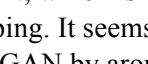
Iterations 0 to 1000

Iteration	New model	Elapsed minutes	Original Model
100		0.5	
200		1	
300		1.5	
400		2	
500		2.5	
600		3	
700		3.5	
800		4	

900		4.5	
1000		5	

Iterations 2000 to 10000

Iteration	New model	Elapsed minutes	Original Model
2000		13	
3000		20	
4000		27	
5000		34	
6000		41	
7000		48	
8000		55	
9000		62	

10000		69	
			
			
			
107500		
		
		
	...		

Note how the sampled batch on our model does not change as it is seeded from a constant batch of samples, while the original model sample batch is constantly changing although the latent seed is kept constant. This is a symptom of mode instability of GAN, which is constantly shifting its latent space mapping. It seems our model outdoes 107500 iterations of DCGAN by around iteration 3000, an improvement of at least 2 degrees of magnitude.

7 REFERENCES

-
- [1] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, et al. **Generative adversarial nets**. In *NIPS*, 2014.
 - [2] Alec Radford, Luke Metz, Soumith Chintala, **Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks**, conference paper at ICLR 2016
 - [3] Anh Nguyen, Jeff Clune, Yoshua Bengio, Alexey Dosovitskiy, Jason Yosinski, **Plug & Play Generative Networks: Conditional Iterative Generation of Images in Latent Space**, IEEE 2017
 - [4] Martin Arjovsky, Soumith Chintala, Léon Bottou, **Wasserstein GAN**, arxiv 2017
 - [5] Junyu Luo, Yong Xu, Chenwei Tang, Jiancheng Lv, **Learning Inverse Mapping by Autoencoder based Generative Adversarial Nets**, arxiv 2017
 - [6] Jeff Donahue, Philipp Krähenbühl, Trevor Darrell, **Adversarial Feature Learning**, arxiv 2017
 - [7] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, Pieter Abbeel, **InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets**, arXiv 2016
 - [8] David Berthelot, Thomas Schumm, Luke Metz, **BEGAN: Boundary Equilibrium Generative Adversarial Networks**, arxiv.org, May 2017