Unified Classification and Generation Networks for Co-Creative Systems

Kunwar Yashraj Singh, Nicholas Davis, Chih-Pin Hsiao, Ricardo Macias, Brenda Lin, Brian Magerko
College of Computing
Georgia Institute of Technology
Atlanta, GA, USA
{kysingh, ndavis35, chsiao9, rmacias3, blin15, magerko}@gatech.edu

Abstract

This paper reports on a new deep machine learning architecture to classify and generate input for co-creative systems. Our approach combines the generational strengths of Variational Autoencoders with the image sharpness typically associated with Generative Adversarial Networks, thereby enabling a generative deep learning architecture for training co-creative agents called the Auxiliary Classifier Variational Autoencoder (AC-VAE). We report the experimental results of our network's classification accuracy and generational loss on the MNIST numerical image dataset and TU-Berlin sketch data set. Results indicate our technique is effective for classifying and generating sketched object images, with larger sizes. We also describe how our network is particularly useful for co-creative agents since it can generate diverse concepts, as well as transform and morph user generated sketches while maintaining their concept identity.

Introduction

Three distinct trends are emerging in the field of computational creativity that form a spectrum of computational intervention in the creative process of users. On one end of the spectrum, intelligent systems assist users to achieve their creative goals with creativity support tools (CSTs). At the other end of the spectrum, users control different algorithmic parameters and deploy fully autonomous systems that generate artworks (i.e. procedural and algorithmic art generation). Computer colleagues represent the midpoint on this spectrum since these creative systems introduce co-creative agents in the user's creative process as a collaborator or partner that can modify and add to a shared creative product.

Collaboration is a powerful way to inspire and support creativity. During creative improvisational collaborations, a new form of distributed creativity arises that can lead to emergent, dynamic, and unexpected meaning to support creativity in new ways (Sawyer and DeZutter 2009). We have seen evidence of how collaboration leads to dynamic and emergent meaning structures that inspire novel ideas during empirical studies of pretend play (Davis, Comerford, et al. 2015) and collaborative drawing (Davis, Hsiao, Yashraj Singh, et al. 2016). We have also found how collaborators often provide unexpected ideas and thus lead to surprising results. The same dynamism and flexibility that make creative collaboration so effective also make it challenging to implement autonomous co-creative agents.

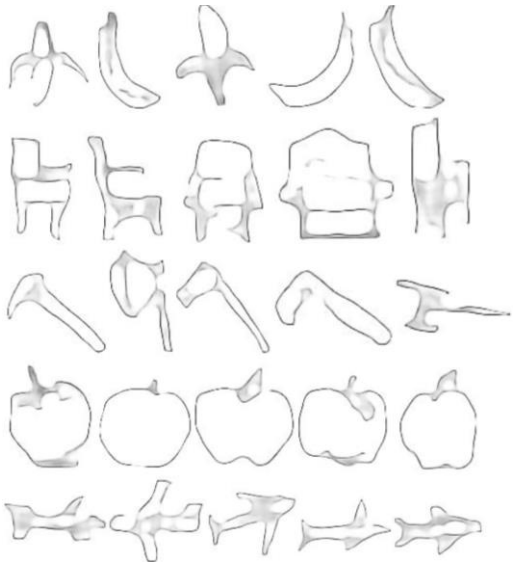

Figure 1. Generated sketches from the TU-berlin sketch data test set using the AC-VAE network.

We designed the Drawing Apprentice as a co-creative drawing partner to help explore what technical approaches and interaction designs are effective for facilitating creative collaboration in drawing. The system analyzes the user's input and responds with its own contribution on a shared canvas. Our findings highlight the importance of classification and generative models for such systems, in order to recognize what type of object the user is drawing and generate an object in response. This creative dialogue of progressively adding related objects to a canvas can help the user generate more novel ideas and stay motivated to continue adding to the drawing. However, training a generative model with highly variable data is problematic in our case. The open-ended nature of drawing means users can introduce virtually any idea, and they expect real time responses. Furthermore, users are more interested in engaging in the drawing activity than explicitly training a system. These factors impede interactive machine learning and place a higher burden on the

interface and UX design to convince users to provide enough feedback to meaningfully train the system.

Creative domains, such as drawing, have two big knowledge engineering challenges: 1) The domain is *open-ended*, *dynamic*, and *highly improvisational*, which means the system needs to both classify and generate sketches in real time; and 2) creative domains do not have a lot of publicly available datasets. Furthermore, only a subset of the publicly available creativity data include data about the process of their creation. For example, in the domain of drawing, the TU-berlin sketch dataset is one of the only publicly available datasets of human sketched objects that contains stroke-level information.

The solution we propose is a deep learning architecture that can learn the latent distribution of example images to effectively classify and generate diverse instances of the learned concept. Our approach is called the Auxiliary Classifier Variational Autoencoder (AC-VAE). Our approach has two main benefits: 1) it enables the design of deeper Variational Autoencoders, and 2) it allows a training methodology to ensure that these networks do not collapse when trained on data that has high variance, such as sketched objects. The network employs a greedy training process for optimization, which ensures that these deeper networks do not collapse when training on data that has higher amount of variations.

We demonstrate how this network can effectively use variational autoencoding on large highly variable sketched image inputs and represent variations in the data by matching it to a unit Gaussian. This means if the latent vector is varied by sampling from a normal distribution, the examples smoothly morph to form each other. In the case of Drawing Apprentice, the network can take a sketch input, convert it to the latent variable, and then transform it to something else (within the same concept) by sampling on the latent vector. Our method uses one deep unified network to achieve high classification accuracy and low generation loss of image based data. This paper describes the AC-VAE network and the details of experimental results showing its efficacy on standard ML datasets and sketched object data. We also describe how the algorithm fits into the Drawing Apprentice architecture and how users will interact with it in the real-time application.

Related Work

Recent advances in deep machine learning enabled powerful classification and generation capabilities. Machine learning has been applied to traditional CSTs by improving the classification of the user's actions to provide better contextual support (Hsiao 2015). Similarly, deep learning has been applied to generative systems to produce extremely detailed and aesthetically pleasing artistic products, as demonstrated by the proliferation of neural style blending applications (Gatys, Ecker, and Bethge 2016). However, designing deep learning architectures for co-creative agents presents unique and interesting challenges as mentioned. These systems need to be able to learn from diverse examples on the fly during improvisation to help facilitate a more seamless collaboration experience.

There have been some creative approaches for generating datasets in creative domains, such as crowdsourcing and pulling creative content from the web (Chen et al. 2014; Colton, Goodwin, and Veale 2012; Veale 2012). While these methods can be successful in generative computational creativity systems, co-creative systems have to actively improvise with users that can generate responses in real time with which the system is completely unfamiliar. This type of improvisational learning requires generalizing from a set of training examples that may have a high variability within the examples.

Recent advances in generative models have enabled algorithms to learn a distribution from input examples and generate new examples from them, which can help train deep neural networks. Some popular methods for generating examples include Autoencoders (Vincent et al. 2010; Bengio et al. 2013), Generative Adversarial Networks (GAN) (Radford, Metz, and Chintala 2015), and Variational Autoencoders (VAE) (Kaae Sønderby et al. 2016; Johnson et al. 2016; Walker et al. 2016). Autoencoders are unsupervised networks that consists of an encoder and decoder stitched together that learn to reconstruct the inputs provided to the network. Generative Adversarial Networks learn the input distribution by training in an adversarial fashion. GANs consists of a discriminator and generator module that attempt to constantly fool each during the training process in order to learn the distribution (Goodfellow et al. 2014). While GANs have gained in popularity recently due to their effectiveness at learning the input distribution to generate realistic images, their stability remains an open question as training them effectively requires a lot of tuning (Arjovsky and Bottou 2017; Salimans et al. 2016). These networks are difficult to train on large image sizes, and often do not converge on large and complex image inputs (Goodfellow et al. 2014).

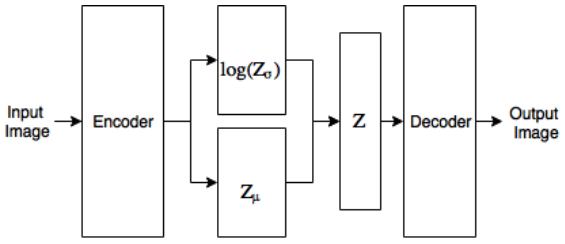

Figure 2. A typical Variational Autoencoder

Variational Autoencoders are a variant of autoencoders that learn a compact representation of the input space, referred to as the latent space (Sønderby et al. 2016). The latent space learned by VAEs are rich in a sense that they also encode various properties of the image implicitly, which is useful for performing vector arithmetic to generate new images (Sønderby et al. 2016). The downside of this approach

is that the images are blurry when compared to GAN due to the mean loss being optimized. Additionally, it is not feasible to train a VAE that is deep and works with large image sizes (Arjovsky and Bottou 2017; Salimans et al. 2016). Recent techniques such as Ladder Variational Autoencoders (LVAEs), Importance Weighted Autoencoders, and Matryoshka Networks have tried to address this problem by utilizing warm up training to introduce the variational term gradually (Bachman 2016; Sønderby et al. 2016; Burda, Grosse, and Salakhutdinov 2015). There has also been success using hybrid top down and bottom up networks (Bachman 2016). In this paper, we build on the finding of LVAE and Matryoshka architecture to overcome some of the limitations of VAEs. This approach enables training deeper and larger VAEs and opens the possibility of handling large input images and leading to sharper output results. We will describe the overall system components before discussing the details of our approach and how it can fit into the system.

System Description

Drawing Apprentice System

The Drawing Apprentice is implemented as a web application (He et al. 2016; Davis, Hsiao, Yashraj Singh, et al. 2016; Davis, Hsiao, et al. 2015) with a client-server architecture that enables multiple people to collaborate on the same drawings as well as the co-creative agent. It was designed for use with stylus- or touch-based interactions, but a mouse can also be used. To briefly summarize its function, the system takes user input lines, transforms those lines based on the sketch recognition and generation algorithms, and outputs new lines onto the same canvas. Unique and defining features of input line sets are determined by clustering the data points and sending that them into the neural network. This allows the neural network to derive its own classifications scheme based on the data it has been given.

As shown in Figure 3, the system was seeded with various algorithms: (1) line transformation functions, such as translation, scaling, rotation (Davis et al. 2011); (2) line morphing techniques that change the individual features of the input lines to create new lines that retain a similarity to the input lines (Davis, Hsiao, et al. 2015; Davis et al. 2014); and (3) recognizing the sketching object and generating similar object in response (Davis, Hsiao, Singh, et al. 2016). During drawing collaboration, the user may begin at any point, which can lead to synchronous collaboration. When the user pauses the drawing, the agent will recognize it as a "turn", and adopt one of the algorithms to generate the response lines. When the user is not satisfied with the agent's drawing actions, she could provide feedback on them to optimize the system's model by clicking on the up and down voting buttons. To simulate the dynamism and embodied nature of real-time human collaboration, the Drawing Apprentice character draws lines dynamically, meaning lines do not appear at once in full, but are gradually animated through until their completion. Dynamic line drawing is meant to provide a sense that the system is going through the embodied act of creating a line. This presents an interesting opportunity where improving the user experience design might potentially improve the performance of the machine learning algorithms (since more feedback helps train the system).

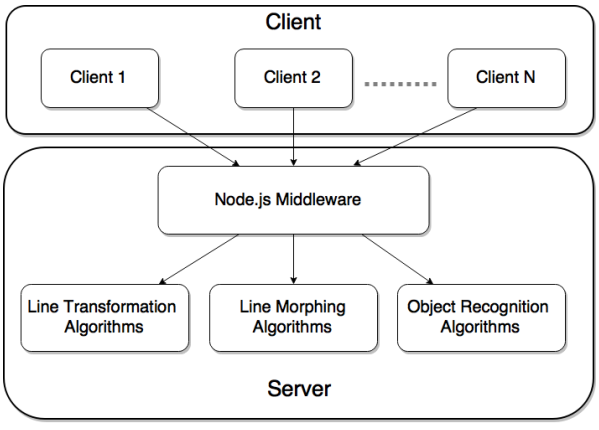

Figure 3. The Drawing Apprentice system overview.

One of the primary limitations of the Drawing Apprentice system that requires the proposed architecture is the ability to generate diverse instances of learned objects. Before integrating the new architecture, the system's line generation capabilities were restricted to reacting to the user's line or recognizing groups of lines as objects and selecting a related object from its databased of known concepts and drawing the selection directly. The system never actively generated new instances of concepts that it learned. The new AC-VAE network learns the latent distribution of example inputs, which allows sampling in that space to generate new outputs based on that distribution. We define two types of sampling for our use case.

Zero-Sampling: when the latent vector of the input image is sampled close to the mean having 0 standard deviation.

Tail-Sampling: when the latent vector of the input image is sampled way from the mean towards the tail of the distribution, having standard deviation close to 1.

Based on the two sampling types we can indicate the amount of variation required on the input. In case of zero-sampling, the input image is reconstructed as close to itself as possible whereas in tail-sampling, the input is varied significantly while still retaining some of the input features.

The following sections describe the design decision and experimentations related to the new network architecture.

Network Architecture

When initially tackling the sketch generation problem using Bayesian Program Learning (BPL)(Lake, Salakhutdinov, and Tenenbaum 2015), we found that the quantification of the semantic representation of differing sketch types is an intractable problem due to the variations in size, detail, and

primitives inherent in each sketch. The large number of combinations of stroke orderings, along with their semantic significance for each sketch archetype is very difficult to represent in a way BPL can generate examples. The mathematical representation of sketch archetypes would be the ideal basis to use when generating new sketch examples. But due to the intractability of the problem, we decided that a more suitable framing of the problem would be an approximation of a family of posterior distributions. These distributions should be similar to a Gaussian, so that variation between sketches of the same type can be captured. The goal is to then sample from these distributions, which will also be representative of the vector space from which sketches can be generated, with a given standard deviation to produce a variation on a given image. We wish to represent distributions that are similar enough to the true generative distribution so that generated images are semantically coherent, but not so closely related that a classifier being trained on a generated example would receive little increase in classification accuracy. Therefore, it is necessary to use a model that can approximate these distributions based on the hidden representation of the salient variables present in each sketch, and this narrowed down the family of algorithms that would be appropriate for the task quite a bit.

We noticed that we could build a modified Variational Autoencoder that would be suitable for the task of approximating the family of posterior distributions because we could build an architecture which uses hierarchies of conditional random variables to represent them (Sønderby et al. 2016; Bachman 2016). The salient aspects of each sketch, represented in the latent space, are used to condition the distributions.

A typical VAE architecture consists of an encoder network and a decoder network. The encoder network takes in the input image and maps it to a latent vector Z of predefined length. This latent vector is then used by the decoder to reconstruct the input image given the information contained in it (Doersch 2016). The variational part comes into play when the encoder does not directly encode the image but splits it into two vectors of same length, namely Z_μ and $log(Z_\sigma)$.Then, the final encoding, which is the latent vector Z, is formed by sampling on the learned mean and standard deviation (Doersch 2016; Nowozin, Cseke, and Tomioka 2016). For example: $Z = Z_\mu + e^{log(Z_\sigma)} * \varepsilon$ where $\varepsilon \sim N(0,1)$ where N is the standard normal distribution. This separation of latent space into two components is referred to as the 'reparameterization' trick (D. P. Kingma and Welling 2013; Doersch 2016) that helps in applying KL-divergence in order to evaluate how well the latent variable matches the unit Gaussian (D. P. Kingma and Welling 2013; Nowozin, Cseke, and Tomioka 2016; Doersch 2016). Hence, the loss function that VAE tries to optimize as its objective is (1) The reconstruction loss and (2) the KL-divergence (D. P. Kingma and Welling 2013).

$$Loss_{reconstruction} = BinaryCrossEntropy(Actual, Generated)$$

$$Loss_{KL} = -\frac{1}{2} Mean(1 + log(Z_\sigma) - {Z_\mu}^2 - e^{log(Z_\sigma)})$$

$$Loss = Loss_{reconstruction} + Loss_{KL}$$

Previous work has shown that smaller networks are able to optimize the loss function easily but cap off after a point due to the difficulty smaller networks face when attempting to capture highly non-linear relationships in data. Deeper networks on the other hand tend to get stuck in a local maxima and collapse when trained on large image sizes (Sønderby et al. 2016; Bachman 2016; Burda, Grosse, and Salakhutdinov 2015). The LVAE architecture demonstrated that training deeper Multilayer Perceptron layers requires gradually introducing the KL term in the loss function during the training process. However, this method had limitations, particularly in knowing at what epoch the KL term should be introduced and on what scale (Sønderby et al. 2016). In the literature, these decisions are usually determined by experimentation and fine-tuning.

The network discussed in this paper was built to generate as well as classify images using the learned features. We call this architecture Auxiliary Classifier Variational Autoencoder (AC-VAE) that consists of an auxiliary classifier as part of the encoder along with the latent vector. We build upon findings from LVAE and use deep residual learning to construct a very deep model that can work with large image sizes and train efficiently in order to generate and classify images. Classification networks built using residual blocks are known for being very deep (>30 layers). Their ability to adjust themselves depend on the degree of linearity of the data (He et al. 2016) eases the training process. These residual blocks, as shown in Figure 4, form an integral part of our network and are symmetrical for the encoder and decoder network. They make use of Batch Normalization, which speeds up training and can be incorporated into deeper networks without running the risk of overfitting. The diagram shown in Figure 5 shows the overall network architecture of AC-VAE using residual blocks and identity mappings between the convolutional layers for the sketch dataset.

Training deeper convolutional networks to optimize the VAE loss requires significant tuning and usually the KL-term becomes very large in the first few epochs to account for variations in the data (Sønderby et al. 2016; Doersch 2016). Hence, we came up with a training strategy that could potentially overcome these limitations and train the deep network effectively. To do so, we reexamined the objective that the network was trying to optimize and found something similar to findings mentioned by LVAEs but the strategy had to be changed. We found that if we minimized just the reconstruction error before we moved to minimize the overall loss, the network would be able to learn good features prior to minimizing the variational error. This scheme enabled the network to partially encode the data distribution before moving on to learning the variations. Apart from this,

we also introduced an auxiliary classifier that shared the

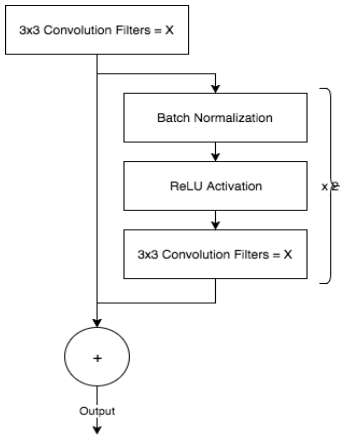

Figure 4. Residual block with identity mapping, used as the basis of our network where X denotes the number of filters used in each convolutional layer.

learned features to classify the input data. Therefore, the overall loss term that AC-VAE minimizes is:

$$Loss_{classification} = CategoricalCrossEntropy(Labels_{true}, Labels_{predicted})$$

$$Loss_{AC-VAE} = Loss_{reconstruction} + Loss_{KL} + Loss_{classification}$$

Furthermore, we noticed that the $Loss_{KL}$ term acts as regularizer for our network, as mentioned in (Sønderby et al. 2016; Bachman 2016; D. P. Kingma and Welling 2013), and helps the entire network counter overfitting. The process below outlines the way the network is trained.

Method: **Train AC-VAE**
For each epoch:

1. Train the network once to minimize $Loss_{reconstruction} + Loss_{classification}$
2. Train the network for k epochs to minimize $Loss_{AC-VAE}$

This method of training facilitates the network to learn good weights before trying to learn the variation across the image batch and this is particularly helpful for deeper network that have large number of parameters. The $Loss_{KL}$term acts as a regularization term (D. P. Kingma and Welling 2013; Doersch 2016) and counters the network from overfitting to the training data. A more intuitive way to reason about the training process is that the first sub-epoch tries to match the training distribution by training the network as a deterministic auto-encoder whereas, the second sub-epoch tries to pull the network away from the distribution by introducing variational cost. This enables the network account for variations in the training data and represent it within the latent vector. This way of greedily minimizing the pieces of the overall loss within each iteration stops the network from collapsing when there are large deviations in the training data and when the minimization process suddenly introduces large variances in the KL-term.

Experiments and Evaluation

MNIST

MNIST dataset contains a collection of handwritten digit images from 0 to 9 (image size 28 by 28 pixels) with 60,000 images in the training set and 10,000 in the test set. (LeCun, Cortes, and Burges 2010). This dataset is widely used to benchmark generative as well as classification networks. For our experiments with MNIST, we created a network of 2 residual blocks as the encoder and 2 for the decoder to account for the non-linearity.

The auxiliary classifier was attached to the intermediate dense layer right before the latent layer and it consisted of another residual block as shown in Figure 4. The network mentioned in the paper was trained with Adam optimizer (D. Kingma and Ba 2014), using no data augmentation and with a learning rate of 0.001 as used during training LVAE (Sønderby et al. 2016). We ran the experiments for different dimensions of latent vectors from 2, 50 and 100 for 50 epochs.

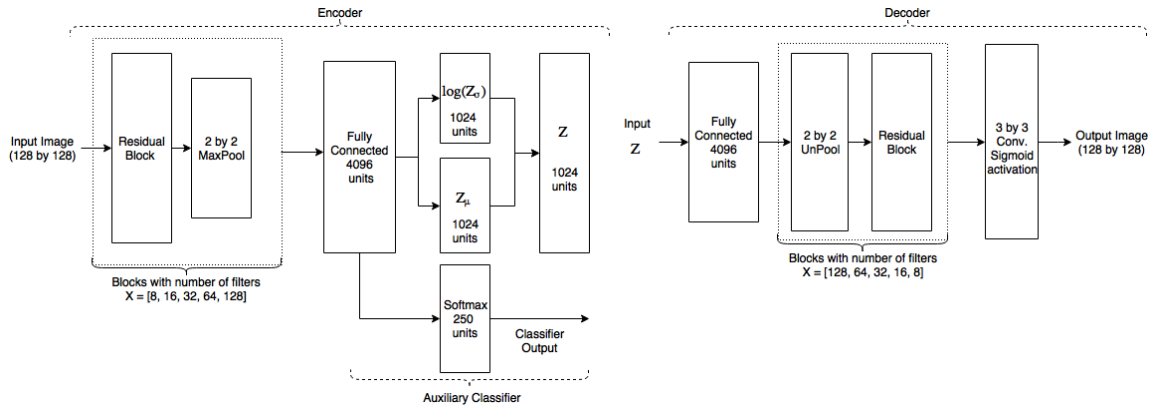

Figure 5. AC-VAE used for sketch data. The auxiliary classifier uses the features extracted by the encoder to classify.

The training process use the training methodology as outlined in this paper and we set $k = 2$. During the training process, we monitored the log-likelihood to report the generation accuracy in addition to the classification accuracy measured. Table 1 below reports the best generation loss achieved in addition to the auxiliary classification accuracy on the MNIST test set and compares it to the other state-of-the-art methods.

We can see that AC-VAE can minimize the loss better than the previously employed methods, this is because network can adjust itself to be as linear or nonlinear as possible depending on the training data. To add to it, the high number of convolutional feature extraction layers work well with spatial data, have more parameters and can better represent the distribution. Apart from the generative mode, we can see that the features learned by the intermediate layer of the network are even useful for classification purposes and the auxiliary classifier reaches a good accuracy of 99.31 percent, which is comparable to the state-of-the-art classification results on MNIST dataset without any data augmentation.

Model	**Generation Log Likelihood**	**Classification Accuracy**
VAE, 2-layer + VGP (Sønderby et al. 2016; Tran, Ranganath, and Blei 2015)	-81.90	-N/A
LVAE, 5-Layer + fine-tuning (Sønderby et al. 2016)	-81.84	-N/A
LVAE, 5-Layer + fine-tuning + IW=10 (Sønderby et al. 2016)	-81.74	-N/A
MATNET (Bachman 2016)	-80.5	-N/A
AC-VAE, Latent size = 50	-55.31	99.22%
AC-VAE, Latent size = 100	-52.63	99.31%

Table 1. Results on the MNIST test set. Generation accuracy is measured using log-likelihood as used in previous works.

TU-Berlin Sketch Dataset

The TU-Berlin dataset is a collection of human drawn sketches of objects from everyday life. It is one of the first datasets to contain several exemplar sketches for a wide variety of human-drawn concepts. The dataset contains 250 categories, with each category containing 80 distinct sketches for a total of 20,000 images. This dataset was selected because it aligns with the scope of Drawing Apprentice project, and it is one of the largest collection of human drawn sketches. We were motivated to test the network and see how well the network could understand and generate

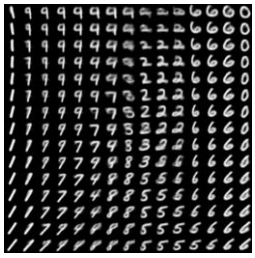

Figure 6. 2D Latent space representation of MNIST test images. Images are generated by sampling points from [0, 0] (top left) to [1, 1] (bottom right). Each row goes from zero-sampling to tail sampling on the right.

variations of the sketches present in this dataset. If successful, this would help the program generate unique variations to the sketches drawn by the user depending on how far the algorithm samples from the distribution, adding transformational qualities to the agent.

Model	**Generation Log Likelihood**	**Classification Accuracy**
DRAW + VGP (Tran, Ranganath, and Blei 2015) (300 epochs)	-423.9	-
AC-VAE (100 epochs)	-887	39%

Table 2. Results on the TU-Berlin test set where generation accuracy is measured using log-likelihood.

For our purpose, the sketch images were resized to 128 by 128 pixels, and we tweaked the network to handle larger images by using 5 residual blocks for the encoder and 5 for the decoder. The dense layers that bridged the encoder and decoder consisted of 4096 neurons in the intermediate layer and 1024 in the latent layer. Overall, the network consisted of 45 layers and was trained using ADAM optimizer with a learning rate of 0.00001 for 100 epochs.

To test our model, we split the 20,000 images into training and test set with 18,000 images in the training set and the remaining 2,000 in the test set. This split was chosen because previous evaluation methods used the same split to benchmark the models. The above table reports the generation and classification accuracy we achieved using our model and compare it to DRAW + VGP as used in D. Tran et al. (Tran, Ranganath, and Blei 2015). Though, DRAW + VGP generated using window sizes, the method cannot be directly compared to ours as it uses a sliding window approach instead of generating end to end. Table 2 above shows the accuracy for 300 epochs for DRAW + VGP and for 100 epochs for AC-VAE. Few of the generated images are also presented in figure 1 and 7 where figure 7 highlights the variational part of the generative network where the first image is the input image used to obtain the initial latent vector and is sampled upon to get the generated images. The

sampling is closer to mean on the left and away from the mean as we move right.

Discussion

Our experimental results show that AC-VAE is effective for classifying and generating sketched object images as required by the Drawing Apprentice co-creative system. This approach combines the strength of VAEs with the image sharpness typically associated with GANs. The network leverages the strength of VAEs to represent variations in the data by matching it to a unit Gaussian. This means if the latent vector is varied by sampling from a normal distribution, the examples smoothly morph to form each other as seen in figure 6. This continuous representation of the conceptual space enables the system to generate diverse examples of that concept during the co-creation with the user, such as helping designers explore the conceptual space of their design.

The AC-VAE generation capabilities are useful for co-creative agents because it can generate concepts representing different degrees of variations within the overall concept. For example, the network can be used to intelligently alter user's sketched objects by converting it to the latent variable and then transforming it to something else within the same concept by sampling on the latent vector.

Future work can also explore combining the latent vectors of two objects into a new space to enable object blending. For example, combining the concepts of tree and airplane may produce interesting tree airplanes when sampling the latent vector. This type of object blending can be used in the real time co-creative system by turning one object into another resulting in a conceptual shift for the user. These types of conceptual shifts are a unique part of the creative process, and are particularly relevant to collaboration as different perspectives often reveal new ways of seeing problems and opportunities in the environment.

Conclusions

This paper reported on a new deep machine learning architecture to classify and generate input for co-creative systems. This network combines the generational strengths of Variational Autoencoders with the image sharpness typically associated with Generative Adversarial Networks, thereby enabling a generative deep learning architecture for training co-creative agents called the Auxiliary Classifier Variational Autoencoder (AC-VAE).

Our approach has two benefits: 1) it enables the design of deeper Variational Autoencoders, and 2) it allows a training methodology to ensure that these networks do not collapse when trained on data that has high variance, such as sketched objects. We reported the experimental results of our network's classification accuracy and generational loss on the MNIST numerical image dataset and TU-Berlin sketch data set. Results indicate our technique is effective for classifying and generating sketched object images, eve n with larger image size. We also described how our network is particularly useful for co-creative agents since it can generate diverse concepts, as well as transform and morph user generated sketches while maintaining their concept identity.

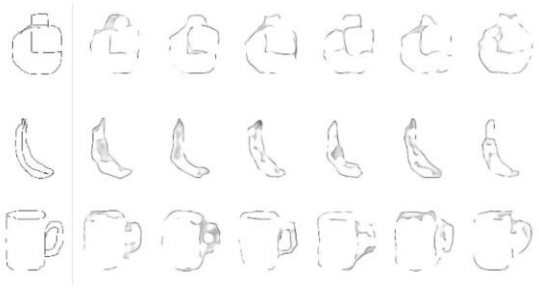

Figure 7. Variations of examples from Tu-Berlin test data generated using AC-VAE. Image on the far left represents the sketch image that was fed into the system. The variations are generated by randomly sampling on the latent variable of size 1024 used in the experiment.

Acknowledgements

This work was supported by NSF Grant IIS-1320520.

References

Arjovsky, Martin, and Léon Bottou. 2017. "Towards Principled Methods for Training Generative Adversarial Networks." In *NIPS 2016 Workshop on Adversarial Training. In Review for ICLR*. Vol. 2016.

Bachman, Philip. 2016. "An Architecture for Deep, Hierarchical Generative Models." In *Advances in Neural Information Processing Systems*, 4826–4834.

Bengio, Yoshua, Li Yao, Guillaume Alain, and Pascal Vincent. 2013. "Generalized Denoising Auto-Encoders as Generative Models." In *Advances in Neural Information Processing Systems*, 899–907.

Burda, Yuri, Roger Grosse, and Ruslan Salakhutdinov. 2015. "Importance Weighted Autoencoders." *arXiv Preprint arXiv:1509.00519*.

Chen, Quanze, Chenyang Lei, Wei Xu, Ellie Pavlick, and Chris Callison-Burch. 2014. "Poetry of the Crowd: A Human Computation Algorithm to Convert Prose into Rhyming Verse." In *Second AAAI Conference on Human Computation and Crowdsourcing*.

Colton, Simon, Jacob Goodwin, and Tony Veale. 2012. "Full Face Poetry Generation." In *Proceedings of the Third International Conference on Computational Creativity*, 95–102.

Davis, Nicholas, Margeaux Comerford, Mikhail Jacob, Chih-Pin Hsiao, and Brian Magerko. 2015. "An Enactive Characterization of Pretend Play." In *Proceedings of the 2015 ACM SIGCHI Conference on Creativity and Cognition*, 275–284. ACM.

Davis, Nicholas, Ellen Yi-Luen Do, Pramod Gupta, and Shruti Gupta. 2011. “Computing Harmony with PerLogicArt: Perceptual Logic Inspired Collaborative Art.” In *Proceedings of the 8th ACM Conference on Creativity and Cognition*, 185–194. ACM.

Davis, Nicholas, Chih-Pin Hsiao, Kunwar Yashraj Singh, Lisa Li, Sanat Moningi, and Brian Magerko. 2015. “Drawing Apprentice: An Enactive Co-Creative Agent for Artistic Collaboration.” In *Proceedings of the 2015 ACM SIGCHI Conference on Creativity and Cognition*, 185–186. ACM.

Davis, Nicholas, Chih-Pin Hsiao, Kunwar Yashraj Singh, and Brian Magerko. 2016. “Co-Creative Drawing Agent with Object Recognition.” In *Twelfth Artificial Intelligence and Interactive Digital Entertainment Conference*.

Davis, Nicholas, Chih-Pin Hsiao, Kunwar Yashraj Singh, Lisa Li, and Brian Magerko. 2016. “Empirically Studying Participatory Sense-Making in Abstract Drawing with a Co-Creative Cognitive Agent.” In *Proceedings of the 21st International Conference on Intelligent User Interfaces*, 196–207. ACM.

Davis, Nicholas, Yanna Popova, Ivan Sysoeven, Dingtian Zhang, and Brian Magerko. 2014. “Building Artistic Computer Colleagues with an Enactive Model of Creativity.” In *International Conference on Computational Creativity, Ljubljuana Slovenia*. AAI.

Doersch, Carl. 2016. “Tutorial on Variational Autoencoders.” *arXiv Preprint arXiv:1606.05908*.

Gatys, Leon A, Alexander S Ecker, and Matthias Bethge. 2016. “Image Style Transfer Using Convolutional Neural Networks.” In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2414–2423.

Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. “Generative Adversarial Nets.” In *Advances in Neural Information Processing Systems*, 2672–2680.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. “Deep Residual Learning for Image Recognition.” In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 770–778.

Hsiao, Chih-Pin. 2015. “SolidSketch: Toward Enactive Interactions for Semantic Model Creation.” In *Proceedings of the 2015 ACM SIGCHI Conference on Creativity and Cognition*, 329–330. ACM.

Johnson, Matthew J, David Duvenaud, Alexander B Wiltschko, Sandeep R Datta, and Ryan P Adams. 2016. “Structured VAEs: Composing Probabilistic Graphical Models and Variational Autoencoders.” *arXiv Preprint arXiv:1603.06277*.

Kaae Sønderby, Casper, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. 2016. “How to Train Deep Variational Autoencoders and Probabilistic Ladder Networks.” *arXiv Preprint arXiv:1602.02282*.

Kingma, Diederik, and Jimmy Ba. 2014. “Adam: A Method for Stochastic Optimization.” *arXiv Preprint arXiv:1412.6980*.

Kingma, Diederik P, and Max Welling. 2013. “Auto-Encoding Variational Bayes.” *arXiv Preprint arXiv:1312.6114*.

Lake, Brenden M, Ruslan Salakhutdinov, and Joshua B Tenenbaum. 2015. “Human-Level Concept Learning through Probabilistic Program Induction.” *Science* 350 (6266): 1332–1338.

LeCun, Yann, Corinna Cortes, and Christopher J.C. Burges. 2010. “The MNIST Database of Handwritten Digits.” http://yann.lecun.com/exdb/mnist/.

Nowozin, Sebastian, Botond Cseke, and Ryota Tomioka. 2016. “F-GAN: Training Generative Neural Samplers Using Variational Divergence Minimization.” In *Advances in Neural Information Processing Systems*, 271–279.

Radford, Alec, Luke Metz, and Soumith Chintala. 2015. “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks.” *arXiv Preprint arXiv:1511.06434*.

Salimans, Tim, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. 2016. “Improved Techniques for Training Gans.” In *Advances in Neural Information Processing Systems*, 2226–2234.

Sawyer, R Keith, and Stacy DeZutter. 2009. “Distributed Creativity: How Collective Creations Emerge from Collaboration.” *Psychology of Aesthetics, Creativity, and the Arts* 3 (2): 81.

Sønderby, Casper Kaae, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. 2016. “Ladder Variational Autoencoders.” In *Advances in Neural Information Processing Systems*, 3738–3746.

Tran, Dustin, Rajesh Ranganath, and David M. Blei. "The variational Gaussian process." arXiv preprint arXiv:1511.06499 (2015).

Veale, Tony. 2012. “From Conceptual Mash-Ups to Bad-Ass Blends: A Robust Computational Model of Conceptual Blending.” In *Proceedings of the Third International Conference on Computational Creativity*, 1–8.

Vincent, Pascal, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. 2010. “Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion.” *Journal of Machine Learning Research* 11 (Dec): 3371–3408.

Walker, Jacob, Carl Doersch, Abhinav Gupta, and Martial Hebert. 2016. “An Uncertain Future: Forecasting from Static Images Using Variational Autoencoders.” In *European Conference on Computer Vision*, 835–851. Springer.