LLAMA: Learning Latent trAnsforMations for generative style trAnsfer

William Hu
william.hu@yale.edu
Yale University

Sydney Thompson
sydney.thompson@yale.edu
Yale University

Nathan Tsoi
nathan.tsoi@yale.edu
Yale University

1 ABSTRACT

We present a method and a tool, called StyleApp (Figure 1), for smooth interpolation between images of different styles. Our technique gives the user control over the visual properties of style and works even when only one sample of a given style is provided. We also explore different architectures and techniques to facilitate realistic generation of handwriting styles. In particular, we first train two types of variational autoencoders on EMNIST [4] to learn character representations and then fine-tune the models on samples of our own handwriting to create person-specific networks for style. After creating our individualized style networks, we investigate latent space clustering and linear transformations as potential methods for extracting semantic meaning from our learned representations. Though our application currently uses labeled data, we show that unsupervised methods of learning semantics from the compressed representation of images is possible and hope that our findings will enable future work.

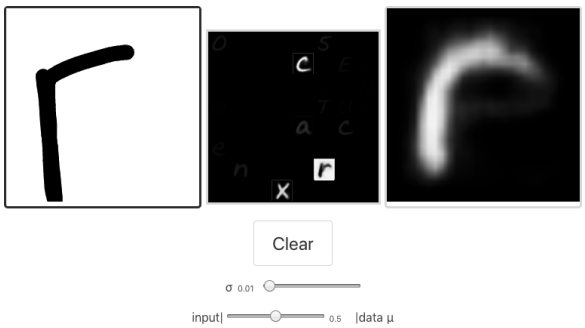


Figure 1: StyleApp: a tool for visualizing latent representations.

2 INTRODUCTION

Handwriting style — a phenomenon shaped by personal, emotional, intentional, and circumstantial factors [5] — varies from person to person. Despite its high complexity, previous work has shown great promise in modeling handwriting style from stroke information [1]. However, little work has been done in learning handwriting style from pixel information. This is likely because strokes provide more information than static images, making it easier for the model to learn during training.

For style transfer, the models first learn low-level representations of characters or text and then apply the learned parameters to new images. Style transfer is a well-explored area of machine learning, but methods often resort to "cleverly balancing optimization strategies" [6] to interpolate between different styles. One class of generative models that have proven effective in style transfer is variational auto-encoders (VAEs). For example, Oord, et al. [18] use VAEs to perform style transfer on audio data. VAEs specialize in the unsupervised learning of complex distributions by representing inputs via latent normal distributions parameterized by mean μ and standard deviation σ, as shown in Figure 2. The latent space can then be used for sampling with variation to generate content in a learned style.

This project aims to build a generative model for recreating person-specific handwriting styles from only character images. We explore the potential of learning low dimensional representations of alphanumeric characters for use in style transfer. This is significantly more difficult than learning style from stroke information because our model is not given any temporal data. However, learning from images is more applicable because stroke information is not always readily available. Through our work, we aim to understand the nature of handwritten character representation in order to facilitate interpolation between styles. This would allow users to modify the characteristics of their handwriting from combining with other styles. In addition, our model framework is general, so it can be extended to any domain of VAE style transfer.

3 RELATED WORK

3.1 Handwriting Style Transfer

Previous work in handwriting style transfer has relied on stroke information to recreate realistic handwriting samples [1], [16]. Most notably, Aksan, et al. [1] used stroke information captured from a tablet to learn sentence-level representations of text. However, handwriting style transfer using only images is still relatively unexplored. This is likely because learning from pixel data is a more challenging task, as stroke information always has implicit temporal data encoded. In other words, learning from stroke information is easier because stroke length, frequency, and order capture extra data relevant to style. Nevertheless, we believe that learning from pixel data has broader applications for style transfer since stroke information is not always available. While research has been done on extracting strokes from images [7], [19], there is no guarantee that the learned stroke order is identical. In addition, learning style transfer from character images can give us a better understanding of character representations.

3.2 Model Selection

3.2.1 Convolutional Neural Networks. Convolutional neural networks (CNNs) have been the go-to model for image style transfer [8], [11]. This is because CNNs are effective at extracting meaningful features from images, including features that are relevant to style. In particular, Gatys, et al. [8] build a model to learn style

transfer for paintings, combining the style of one painting with the content of another.

3.2.2 Generative Style Transfer. For generative style transfer, CNNs are often used as layers in generative models that seek to learn image reconstructions for a set of given styles. Variational autoencoders (VAEs) are one such generative model that have shown promise in style transfer [2], [18], [21]. As shown in Figure 2, VAEs are a class of models that learn latent representations of inputs via Gaussian means μ and standard deviations σ. They are structurally similar to autoencoders, but they support generative sampling of outputs from their learned latent space. In 2017, Oord, et al. [18] developed a vector quantized-VAE (VQ-VAE) that could learn latent representations of phonemes to transfer voices from one source of audio to another. Their model uses convolutional layers to both extract features and generate reconstructions, producing realistic audio clips in different voices. Other applications of style transfer from VAEs include control [21] and music [2].

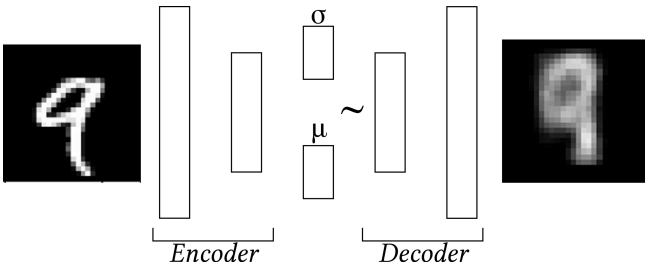


Figure 2: VAE architecture showing the input image and reconstrution via latent mean and standard deviation vectors. The ~ indicates a sampling operation from the normal distribution parameterized by the latent μ, σ variables.

3.3 Unsupervised Learning of Semantics from Latent Representations

Though the definition of the term "disentanglement" is still under debate, it is generally believed that a compressed, factorial latent space z leads to interpretable and semantically meaningful representations [3]. Recent work in representation learning has shown skepticism towards the idea of unsupervised learning of disentangled representation without inductive biases [13], and in parallel, there has been research on jointly optimizing dimensionality reduction and clustering in [20]. Like previous work, we take an approach to extracting structure and semantic meaning from our learned latent representations. Our focus, however, is not on the end task of classification, as in [20], but to better understand structure in z to perform style transfer via interpolation between meaningful points in latent space.

3.4 Style Loss

To evaluate style transfer, the loss is generally divided into two components: content loss and style loss. Gatys, et al. [8] formulates the loss function as follows:

$$\mathcal{L} = \alpha \mathcal{L}_{content} + \beta \mathcal{L}_{style}, \tag{1}$$

where α and β are the weights for the content and style losses respectively. Here, the content loss $\mathcal{L}_{content}$ is defined as the L2 norm of the difference between feature maps at a particular layer in the CNN. That is,

$$\mathcal{L}_{content} = \frac{1}{2} \sum_{i=1}^{N_l} \sum_{i=1}^{M_l} (F_{ij}^l - P_{ij}^l)^2 \tag{2}$$

where F_{ij}^l is the tensor of feature maps for layer l of the neural network corresponding to the stylized image and P_{ij}^l is the tensor of feature maps for layer l of the neural network corresponding to the content image. N_l is the number of feature maps within the layer l, and M_l is the height multiplied by the width of the feature map.

The style loss, on the other hand, is computed as a weighted average over all layers, where w_l is the weight of layer l:

$$\mathcal{L}_{style} = \sum_l w_l \mathcal{L}_{style}^l \tag{3}$$

The style loss of an individual layer works to minimize the L2 norm of the difference between the Gram matrices of the stylized image and the target style image.

$$\mathcal{L}_{style}^l = \frac{1}{4N_l^2 M_l^2} \sum_{i=1}^{N_l} \sum_{j=1}^{M_l} (G_{ij}^l - A_{ij}^l)^2 \tag{4}$$

The Gram matrix encodes information about the style of the image without coupling too tightly with the content of the image. In this way, the L2 norm of the gram matrices of the stylized and target style image provides a reasonable candidate for style loss:

$$G_{ij}^l = \sum_{k=1}^{M_l} F_{ik}^l F_{jk}^l. \tag{5}$$

The Gram matrix A_{ij}^l is constructed in the same manner as G_{ij}^l, but with the feature maps corresponding to the target style image.

4 METHOD

The following describes our exploration method for better understanding the compressed representation of character images. This includes preliminary analysis via clustering of the latent dimensions and t-SNE visualization [15]. Following preliminary analysis we implement latent traversal, style loss, clustering, and network fine-tuning. We explore these methods on the BetaVAE and InfoVAE networks. For all experiments, we use the same encoder and decoder architecture and vary elements such as components of the loss and latent dimension size (from 2 to 32). We also describe in this section the method of style interpolation used in our final product: StyleApp.

4.1 Preliminary Exploration

To model handwriting style, we first trained a VAE on the EMNIST dataset [4] to learn character reconstructions. We chose to use a VAE because once we learned the latent representations of an individual's style, we coule resample from the latent space to generate realistic variation. In addition, we could explore the latent space to better understand the character representations in the scope of the

generative model. We considered two different kinds of VAEs for reconstructions: InfoVAE [22] and BetaVAE [9].

In our preliminary exploration, we trained one InfoVAE [22] model with 20 latent dimensions and many BetaVAE [9] models with a range of latent dimensions between 2 and 64. All models were implemented with Tensorflow, and we experimented training on MNIST [12] and EMNIST [4] independently. For each dataset, we held 20% for validation and trained on the remaining 80%. MNIST [12] only contains the 10 digits [0-9] and was used to determine whether the model could actually reconstruct character images. Extended MNIST (EMNIST)[4], on the other hand, contains all of the alphanumeric characters and is the main dataset we used to train our models.

In an effort to understand these preliminary models, we ran a t-Distributed Stochastic Neighbor Embedding (t-SNE) [15] model. If m is the number of dimensions of the latent space, then this model transforms the latent mean μ from $\mathbb{R}^m$ into $\mathbb{R}^2$ to enable visualization. The t-SNE model [15] works by minimizing the KL divergence between the conditional probability of pairs of samples in the original 16-dimensional latent means and the conditional probability of pairs of the 2-dimensional samples transformed by the t-SNE model [15]. This allows us to visualize results with μ's of more than two dimensions, while maintaining some of the important structure from the high-dimensional setting. We used this tool as an additional heuristic measure of how well our model was training.

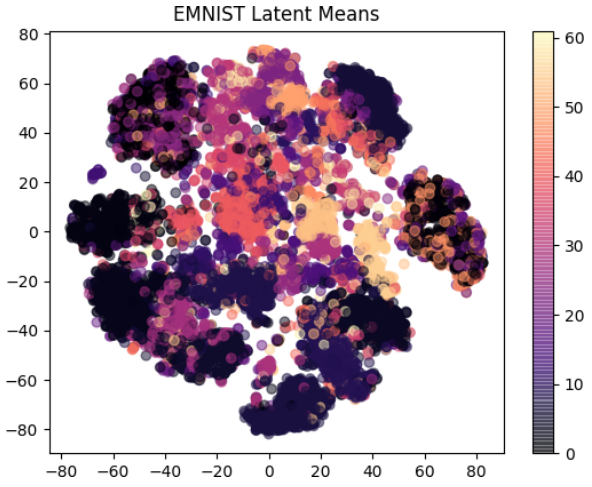


Figure 3: t-SNE [15] Embeddings for all characters [a-z,A-Z,0-9] in the EMNIST dataset. These latent means were generated from a VAE with 16 latent dimensions.

Note that in Figure 3, there are pronounced clusters representing the digits. For clusters representing letters, the distinction is less clear, but this seems to indicate that the model could be learning latent representations that encode semantic meaning for the letters. We investigated this further in section 5.3 with various clustering techniques on the latent character embeddings.

Once we trained our initial model, we fine-tuned it using individual handwriting samples. As a proof of concept, we demonstrated that fine-tuning the network works with one of our group member's handwriting style. To collect this individualized data, we each completed several grid-spaced 8.5 x 11" worksheets, as shown in Figure 4. Each member completed 3 uppercase and 3 lowercase worksheets, and all worksheets contained the 10 digits. This results in 216 usable characters per user after preprocessing, done via an automated image processing pipeline that we built. Note that while we converted our handwriting samples to EMNIST [4] format, we didn't initially remove extra white space around our characters, as recommended by [4]. This allowed us to preserve scale information for users with smaller or larger handwriting. We removed the white space in later experiments. Approximately 72% of the handwritten characters are alphabetic and the remaining 28% are numeric. Our scripts to processes the input sheets yielded 216 characters for at least 1 user and a few less for other users due to a problem with scanning quality. We expect re-scanning these sheets will resolve the issue.

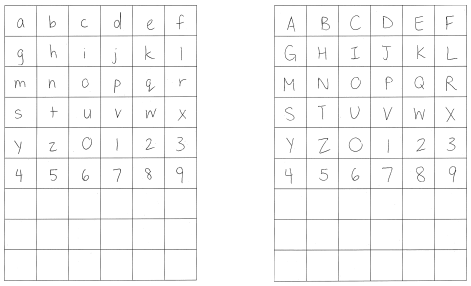


Figure 4: Sample data collection worksheets that were automatically processed into EMNIST [4] format using OpenCV [10].

4.2 Architecture search

Following our preliminary investigation, we performed a limited architecture search for both InfoVAE [22] and BetaVAE [9] before settling on a 2-layer convolutional encoder along with a similar deconvolutional decoder network. We searched over networks with latent dimensions ranging from 2 to 32, batch sizes between 1 and 4096, both the Adadelta and Adam optimizer (learning rate from 5e-3 to 1e-10).

4.3 Latent Traversal

Throughout training and experimentation we relied on latent traversal to examine how a network is learning. In the case of networks with only 2 latent dimensions, we simply iterated, in some preset interval, along each dimension and plotted the reconstruction results along the two axes of a two-dimensional graph. An example is shown in Figure 7. However, two latent dimensions were not enough to encode the EMNIST dataset [4]. We therefore developed a different way to visualize the networks evolution of weights and biases.

For networks with more than two latent dimensions we recorded both the class label c and the compressed representation z_i, for each i-th character's n-dimensional representation seen in a training epoch. After completing all steps through an epoch of training, we computed the mean latent representation, z_μ, for each character

component-wise. We then visualized these latent mean representations via the decoder component of the Variational Auto Encoder. We performed the same process on the validation set over each epoch. An example output image is shown in Figure 5.

Figure 5: An example image generated from mean values via latent traversal of a 32 latent-dimension InfoVAE [22] network (validation set).

4.4 Style Loss

To determine a qualitative baseline for style transfer, we adapted the Keras implementation [1] of style loss using the VGG19 architecture [17]. Note that because the style loss function is formulated specifically for the VGG family of models, we decided not to incorporate it into our VAE models.

4.5 Clustering

Many aspects of our current approach, from visualization of the latent means to the implementation of the StyleApp, described below, depend on labeled data and the ability to accurately classify unlabeled samples. Our hypothesis was that clustering the disentangled latent representation of a VAE should allow us to avoid the need for labeled data and a supervised learning paradigm. We also hypothesized that learning meaningful latent representations corresponds with proximity of embeddings of characters of the same class within latent space.

We experimented with both affinity propagation and k-means clustering by encoding all inputs in our dataset and recording their latent representations z_i along with their character class c_i. The z_i's were then clustered using both affinity propagation and k-means (with $k = 62$). Clustering efficacy was determined by computing the most common character in each cluster, assigning the cluster that label, and comparing cluster members against their ground truth label c_i.

[1] https://github.com/keras-team/keras/blob/master/examples/neural_style_transfer.py

4.6 Network Fine-tuning

In an attempt to perform style transfer with very few handwriting samples, we trained both BetaVAE [9] and InfoVAE [22] networks with varied hyperparameters on the full EMNIST dataset [4]. We then substantially decreased the learning rate and trained for between 1 and 10 epochs on our collected datasets. We experimented with fine-tuning procedures by allowing weight updates to the encoder and decoder as well as allowing weight updates to occur in the decoder only.

4.6.1 Transform loss. To facilitate transferring between styles, we attempted to use one of the methods outlined in [14]. That is, if we consider a VAE network X, denote the encoder as E_X, the decoder as D_X, the latent distribution l_X, samples from this latent distribution $z_X \sim l_X$, and reconstructions of the input x as $\hat{x}$. Let G be the general character-reconstructing VAE trained on the EMNIST data [4], S denote the fine-tuned style network, and let A be a linear transformation where $A : \mathbb{R}^m \to \mathbb{R}^m$ where m is the dimension of the latent space. Our goal was to jointly train the style network S and the linear transformation A so that we could use the EMNIST [4] reference images and trained weights from the general network G to construct new samples in the handwritten style. We initialized S with the weights from G.

To train the style network S and transformation A, given an example of character c from the training data x_t and an example of the same character from the handwriting sample x_h, we computed the following:

$$
\begin{aligned}
(l_G)_{x_t} &= E_G(x_t) \\
(z_G)_{x_t} &\sim (l_G)_{x_t} \\
(z_S)_{x_t} &= A(z_G)_{x_t} \\
\hat{x}_t &= D_S((z_S)_{x_t})
\end{aligned}
$$

$$
\begin{aligned}
(l_S)_{x_h} &= E_S(x_h) \\
(z_S)_{x_h} &\sim (l_S)_{x_h} \\
\hat{x}_h &= D_S((z_S)_{x_h})
\end{aligned}
$$

Letting $BCE(x, y)$ be the binary cross-entropy between x and y, $MMD(x, z)$ to be the Maximum Mean Discrepancy [22] between samples x and distribution z, then to train we optimized the following loss function, $\mathcal{L}$:

$$
\begin{aligned}
\mathcal{L}_{\text{recon}} &= BCE(\hat{x}_t, \hat{x}_h) \\
\mathcal{L}_{\text{VAE}} &= BCE(x_h, \hat{x}_h) \\
\mathcal{L}_{\text{MMD}_t} &= MMD(A(l_G)_{x_t}, \mathcal{N}(0, 1)) \\
\mathcal{L}_{\text{MMD}_s} &= MMD((l_S)_{x_s}, \mathcal{N}(0, 1)) \\
\mathcal{L}_{\text{means}} &= BCE(A(l_G)_{x_t}, (l_S)_{x_s}) \\
\mathcal{L} &= \alpha\mathcal{L}_{\text{recon}} + \beta\mathcal{L}_{\text{VAE}} + \mathcal{L}_{\text{MMD}_t} + \mathcal{L}_{\text{MMD}_s} + L\mathcal{L}_{\text{means}}
\end{aligned}
$$

If this transformation A is invertible, then we could also use this method to train multiple style networks and interpolate from style to style by applying these transformations in sequence.

4.7 StyleApp

For the final portion of the project we created a web application [2], shown in Figure 6, to demonstrate our method of style transfer. The application accepts either physical-touch or mouse-pointer based input on a 256-by-256 pixel square canvas. When the user draws a character on the canvas, the canvas is rendered and converted into an black and white image which is then re-scaled and passed as input to a convolutional neural network, trained on the EMNIST dataset [4] to a validation accuracy of 72%, assigning one of 62 character classes to the input image.

Below the drawing pad are two sliders: the top slider controls the standard deviation associated with sampling from the VAE, and the bottom slider controls the amount of style interpolation λ between the handwriting sample and the reconstructions of each character class' latent mean z_μ. The interpolated character's latent representation is then computed by the formula

$$z = z_{pad} + \lambda \left(z_\mu - z_{pad} \right),$$

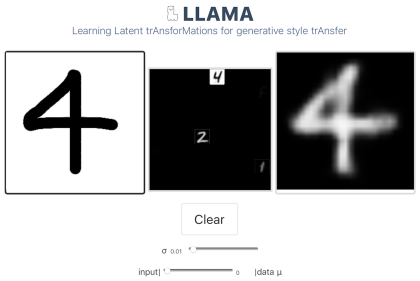
where z_{pad} is the encoded representation of the drawn input. The style-transferred output image is created by decoding the interpolated z value. In addition, the characters in the center matrix can be selected to allow the user to override the classifier's prediction and to visualize interpolation between two different characters. In practice, this smooths the input character towards the mean character chosen in the matrix.

StyleApp is designed as a single-page static website. The entire application runs in the browser, and the assets are provided as static files for the browser to consume. There is no server-side processing component. We use the TensorflowJS [3] library for client-side operations on our trained networks. To do this we convert the saved Keras models from training, both classifier and InfoVAE, to TensorflowJS compatible JSON files, which are then loaded by our webapp.

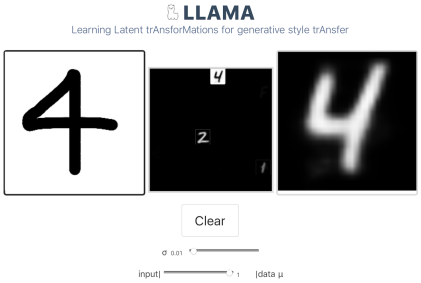
5 EXPERIMENTS

In our preliminary experimentation, we implemented a BetaVAE [9] with $\beta = 1$. We tested both 2 and 16 latent dimensions. We trained this model on both MNIST [12] and EMNIST [4]. As seen in Figure 7, two dimensions were insufficient to adequately represent all alphanumeric characters. In our architecture search, we investigated the effect of increasing the number of dimensions used in the latent representation.

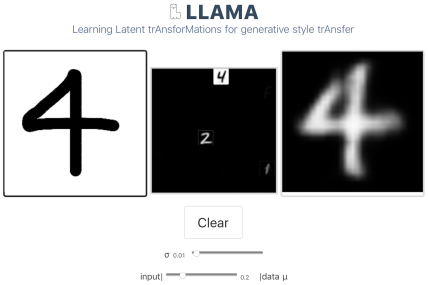
We evaluated our style-transfer approach by fine-tuning the VAE trained on EMNIST [4] with 2 latent dimensions. Again, we used the same network and hyper-parameters, except for batch size, which was reduced to accommodate our much smaller style data set of approximately 200 samples.



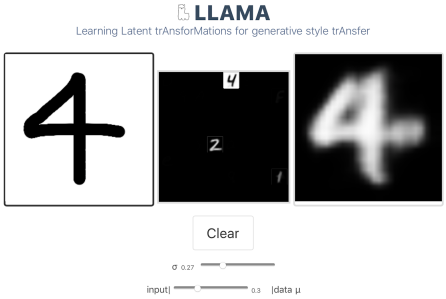
(a) Input and reconstructed image.



(b) Input and latent mean image.



(c) Interpolation between the input and latent mean image.



(d) Interpolation with variance.

Figure 6: Images from our style app. The interpolation image is a mixture of the reconstructed handwriting sample in (a) and the latent mean reconstruction of the character class in (b).

[2] https://styleapp.netlify.com
[3] https://tensorflow.org/js

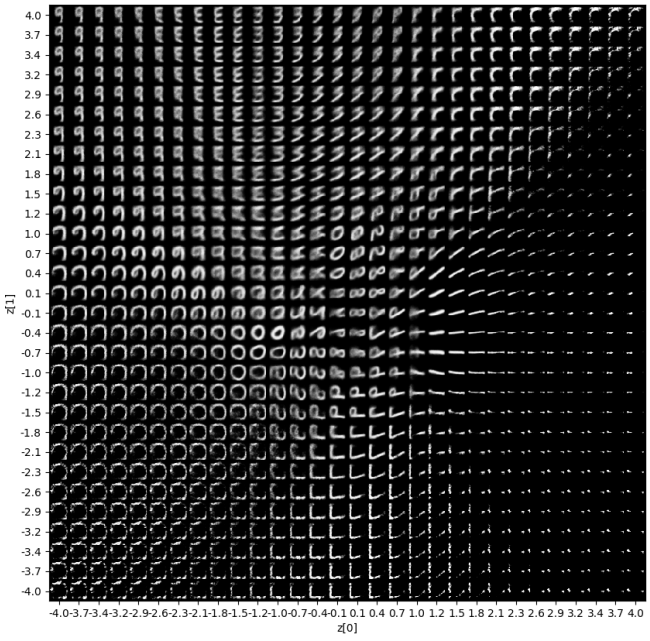
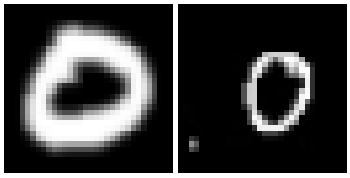


Figure 7: Reconstructions via latent traversal from a VAE trained on EMNIST [4] with 2 latent dimensions.



(a) Base image. **(b) Style image.**

(c) Progression of the combined image.

Figure 8: The VGG19 [17] model's style and content extractors are tuned to produce realistic combined images. After just a few iterations, the combined image shows properties of both the base and the style images. While the overall shape of the combined image is closer to that of the style image, the weight of the character is between the base and style images.

5.1 Architecture Search

Following our preliminary experimentation, we performed a limited architecture search. This revealed that the best reconstruction results could be obtained reliably with an InfoVAE [22] with 32 latent dimensions, trained over 40 epochs with a batch size of 2048 and Adam optimizer with a learning rate of 5e-3. It also appeared the batch size is not crucial when training a network with this size of compressed representation.

While we did have some success training both InfoVAE [22] and BetaVAE [9] with 8 latent dimensions, learning was highly unpredictable. As we made small, necessary changes to the loss function, batch size, learning rate and initialization we saw significant changes in the output of the network. These factors all played a large role in convergence and therefore majority of the time this network would not learn, collapse into (incorrectly) reconstructing a single output image for all input.

5.2 Style Loss

After training our base networks, we built a VGG19 model [17] with style loss to determine a qualitative baseline for style transfer. We observed that the VGG19 model [17] with style loss could generally combine the content and style from two images and interpolate between them. As shown in Figure 8, the 'o' at the end of the progression is a combination of both the base and the style images. The shape and thickness of the 'o' suggests that the model is in fact learning from both of the starting images.

5.3 Clustering

Our efforts to cluster the latent representations of the EMNIST [4] data demonstrated the usefulness of our latent representations. Clustering the raw images using k-means ($k = 62$) resulted in a classification accuracy of 48%, which served as a baseline for comparing the usefulness of the latent representations. We then compared the results of using k-means clustering on the latent representations from both InfoVAE [22] and BetaVAE [9]. These had a classification accuracy of 57% and 14%, respectively. For the InfoVAE [22], this is an 18% improvement over the raw images, which indicates that the latent representations are at least somewhat meaningful. It is also worth noting that training the k-means clustering algorithm on the full images required $24x24 = 576$ inputs while training on the latent representations required only 32-dimensional input - a 94% decrease in input size. This result indicates that clustering on latent representations of images can be faster and potentially more meaningful than clustering on raw images. In practice, the difference would be even more pronounced on datasets with much larger images.

Our results from applying k-means to the latent representations also suggest that the latent space generated from the InfoVAE [22] was reasonably well-separable. By contrast, the low accuracy of the BetaVAE [9] suggests that characters of the same class are not mapped closely together in latent space. This is consistent with the analysis in [22], indicating that the architecture of the BetaVAE [9] suffers from learning that occurs only in the decoder instead of creating meaningful latent variables from a properly trained encoder. This demonstrates that the InfoVAE [22] was significantly more effective in creating meaningful latent representations than the BetaVAE [9] and therefore more effective towards our goal of style transfer.

We see in Figure 9 that the digits are cleanly separated into distinct clusters. However, non-numeric characters are not as cleanly separated among clusters, possibly due to the large data imbalance in the EMNIST dataset [4]. Approximately 34% of the dataset is

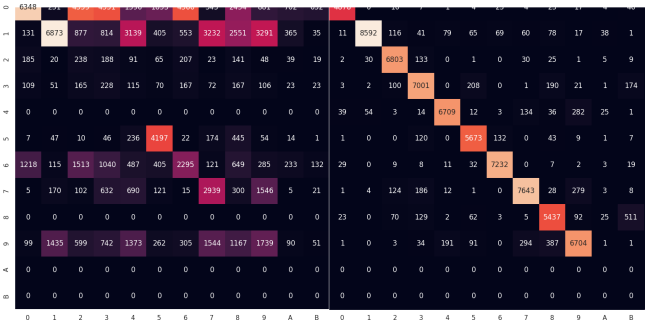


Figure 9: A confusion matrix of the distribution of characters within clusters for a k-means clustering algorithm with $k = 62$. The y-axis denotes the most frequent character within each cluster and the x-axis shows the character count per ground truth label class. This matrix shows the breakdown of a subset of characters within a subset of the clusters from the k-means clustering. The clusters not shown contain insignificant numbers of samples in the BetaVAE [9] clusters.

composed of the 10 digits and the remaining 52 characters are represented in only 66% of the total samples.

We believe our results show promise in finding interesting or even semantic (e.g. character classes in this case) points in latent space in an unsupervised manner. While our unsupervised classification technique via clustering under-performs when compared to the the CNN we trained (57% accuracy vs 72% accuracy), it does not depend on labeled data, providing a method to learn semantics in an unsupervised context. We hypothesize that more powerful clustering techniques combined with a more powerful encoder and decoder could yield better results, both on smaller datasets, like EMNIST [4], and possibly even on datasets of greater complexity.

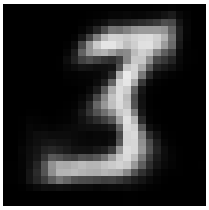
5.4 Network Fine-tuning

In our experimentation with various types of network fine-tuning for both BetaVAE [9] and InfoVAE [22]. We found that updating the weights in both the encoder and decoder was effective in terms of reconstruction error. In this configuration, we were easily able to update the encoder and decoder to accurately reconstruct a given users' input using only 200 samples in a single epoch of training.

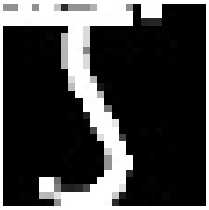
We then attempted to maintain a similar latent representation between the encoder trained on all of ENMIST [4] and the encoder for a given user by disallowing weight updates to the decoder during fine tuning. In this configuration the network was able to reconstruct only a few characters classes. While this warrants further exploration, it appears that reconstruction only succeeds in this case when the sample for a user is close enough to the samples in the EMNIST dataset [4] and can therefore be enocoded using the pretrained encoder.

In practice, we found that image pre-processing was the most important factor when encoding a user's input. Cropping and the thickening of lines in a sample image was required to make the encoder work effectively.

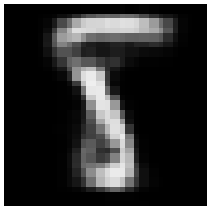
(a) Mean reconstruction for each character class from the fine-tuned network for an individual's handwriting. The means are computed from style-network latent representations of each character class.



(b) Mean reconstruction of 'J' from the general network.



(c) Sample of handwritten character 'J' from user.



(d) Mean reconstruction of 'J' from the style network.

Figure 10: The reconstructions of styled handwriting samples using the style network trained as described in Section 4.6.1. This grid is created by computing the mean latent embedding for each class and then visualizing the reconstruction.

5.4.1 Transform loss. The style network successfully incorporated artifacts of the sampled handwriting and took approximately 40 seconds to train on an Nvidia GeForce RTX 2070. Samples of each character from the style network are found in the grid in Figure 10a.

We can see in Figure 10b, Figure 10c, and Figure 10d that the styled network does learn artifacts of the handwriting. These examples of the character 'J' from the styled network are much more similar to the style target 'J' than the original 'J'. This method creates samples that are slightly blurry, but shows empirically that the network is learning handwriting style from the provided samples.

Unfortunately, as we can see in Figure 11, the linear transform does not appear to learn a meaningful mapping between the general VAE G and the style network S. This inhibited our ability to

Figure 11: Mean reconstructions for the first 8 character classes from the network fine-tuned on an individual's writing. The mean is computed from latent representations of each class from general network G and transformed using A to latent space for style network S, then sampled and reconstructed using D_S.

test this method with style transform from one handwriting to another. This collapse occurred even when the transform was trained independently of the style network. This could suggest that the transformation from the general latent space to the style-specific latent space may not be well-approximated by a linear transformation. Future exploration in different network architectures for this transformation may produce better results.

6 CONCLUSION

Our work allowed us to create meaningful low-dimensional representations of EMNIST [4] images to facilitate generative style transfer. This was most successful with online interpolation between handwritten digits in the StyleApp. Our application allows users to smoothly interpolate between their own handwriting and mean characters from the trained VAE. In practice, this technique could be used to create more readable versions of a user's handwriting while still maintaining elements of their personal style.

We also saw success in the inverse problem of generating handwriting styles. Though imperfect, with only a few examples of a user's handwriting, the network can generate characters that clearly demonstrate artifacts of the user's style. Future work can focus on improving the input images to create cleaner reconstructions.

Our approach of clustering the latent representation to find semantically meaningful points for interpolation shows promise towards better interpretability of learned representations with more than 2 dimensions, where simple traversal and visualization is not possible. We hope to explore applications of this technique on other datasets in future work.

7 FUTURE WORK

Our results in style interpolation and clustering show promise in understanding the latent representations of handwritten characters and provide interesting avenues for future work. One area that deserves more exploration is the understanding of the latent space generated from the encoders. As we have shown, there is benefit to using unsupervised methods of representation learning to compress and analyze high-dimensional input for generating new data and interpolating between interesting points in latent space. We can continue to improve this work by further investigating the interpretability of low dimensional representations in larger, more complex datasets, with the same fundamental approach but larger and more powerful parameterizations of our technique.

This work could be further advanced by improving the networks used to perform style transfer. Areas of improvement include refining the style network via better transformations from the general network to the style network, incentivizing crisp edges for more realistic reconstruction, and creating an interface for users to upload scans of their handwriting so that the StyleApp can perform interpolation on larger handwriting sample sets.

REFERENCES

[1] Emre Aksan, Fabrizio Pece, and Otmar Hilliges. 2018. DeepWriting: Making Digital Ink Editable via Deep Generative Modeling. In *SIGCHI Conference on Human Factors in Computing Systems (CHI '18)*. ACM, New York, NY, USA.

[2] Gino Brunner, Andres Konrad, Yuyi Wang, and Roger Wattenhofer. 2018. MIDI-VAE: Modeling Dynamics and Instrumentation of Music with Applications to Style Transfer. arXiv:cs.SD/1809.07600

[3] Tian Qi Chen, Xuechen Li, Roger B Grosse, and David K Duvenaud. 2018. Isolating sources of disentanglement in variational autoencoders. In *Advances in Neural Information Processing Systems*. 2610–2620.

[4] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. 2017. EMNIST: an extension of MNIST to handwritten letters. *arXiv preprint arXiv:1702.05373* (2017).

[5] J. . Crettez. 1995. A set of handwriting families: style recognition. In *Proceedings of 3rd International Conference on Document Analysis and Recognition*, Vol. 1. 489–494 vol.1. https://doi.org/10.1109/ICDAR.1995.599041

[6] Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. 2016. A learned representation for artistic style. *arXiv preprint arXiv:1610.07629* (2016).

[7] Yaroslav Ganin, Tejas Kulkarni, Igor Babuschkin, S. M. Ali Eslami, and Oriol Vinyals. 2018. Synthesizing Programs for Images using Reinforced Adversarial Learning. arXiv:cs.CV/1804.01118

[8] Leon A Gatys, 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.

[9] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. 2017. beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. *ICLR* 2, 5 (2017), 6.

[10] Itseez. 2015. Open Source Computer Vision Library. https://github.com/itseez/opencv.

[11] Yongcheng Jing, Yezhou Yang, Zunlei Feng, Jingwen Ye, Yizhou Yu, and Mingli Song. 2017. Neural Style Transfer: A Review. arXiv:cs.CV/1705.04058

[12] Yann LeCun, Corinna Cortes, and CJ Burges. 2010. MNIST handwritten digit database. *ATT Labs [Online]. Available: http://yann. lecun. com/exdb/mnist* 2 (2010).

[13] Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Rätsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. 2018. Challenging common assumptions in the unsupervised learning of disentangled representations. *arXiv preprint arXiv:1811.12359* (2018).

[14] Stephen Lombardi, Jason Saragih, Tomas Simon, and Yaser Sheikh. 2018. Deep appearance models for face rendering. *ACM Transactions on Graphics* 37, 4 (2018).

[15] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, Nov (2008), 2579–2605.

[16] Omar Mohammed, Gerard Bailly, and Damien Pellier. 2018. Style Transfer and Extraction for the Handwritten Letters Using Deep Learning. arXiv:cs.CV/1812.07103

[17] Karen Simonyan and Andrew Zisserman. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv:cs.CV/1409.1556

[18] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. 2017. Neural Discrete Representation Learning. arXiv:cs.LG/1711.00937

[19] Ning XIE, Hirotaka HACHIYA, and Masashi SUGIYAMA. 2013. Artist Agent: A Reinforcement Learning Approach to Automatic Stroke Generation in Oriental Ink Painting. *IEICE Transactions on Information and Systems* E96.D, 5 (2013), 1134–1144. https://doi.org/10.1587/transinf.e96.d.1134

[20] Bo Yang, Xiao Fu, Nicholas D Sidiropoulos, and Mingyi Hong. 2017. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 3861–3870.

[21] Ya-Jie Zhang, Shifeng Pan, Lei He, and Zhen-Hua Ling. 2018. Learning latent representations for style control and transfer in end-to-end speech synthesis. arXiv:cs.CL/1812.04342

[22] Shengjia Zhao, Jiaming Song, and Stefano Ermon. 2017. InfoVAE: Information Maximizing Variational Autoencoders. *CoRR* abs/1706.02262 (2017). arXiv:1706.02262 http://arxiv.org/abs/1706.02262