Analysis and Implementation of a Semantic Auto-Encoder for Zero-Shot Learning

1st Arpad Voros
Department of Electrical Engineering
North Carolina State University
Raleigh, USA
aavoros@ncsu.edu

***Abstract*—A semantic auto-encoder (SAE) utilizes the encoder-decoder paradigm to be utilized in zero-shot learning (ZSL) for latent space projection and feature space reconstruction. In this document, a SAE is used to perform ZSL and classification on a handwritten character dataset. The last feature layer of a classification model is projected into latent space using a respective SAE and the resulting projection is used as a new baseline for classification. Features of a single unseen datapoint of each unseen class are projected into the latent space and used as the reference for classifying all other unseen datapoints to perform ZSL and classification. A rudimentary decoder is incorporated to enhance visual understanding of the latent space representation of each datapoint and a proposition is made further the effort in incorporating SAE into an unsupervised learning setting.**

I. Introduction

Zero-shot learning (ZSL) has been an area of growing interest in deep learning the past decade, where a models ability to predict classes, previously unseen to the model, is enhanced after the official training phase of the model. Though there are various methods to implement ZSL, the one chosen for this independent study was a semantic auto-encoder (SAE) [1] due to its simplicity and computational efficiency. Inspired by the usage of the Omniglot dataset in a paper using Siamese neural networks to perform one-shot learning [2], a similar handwritten character-set is utilized from NIST to implement a SAE for zero-shot classification (ZSC).

The purpose of this study is to analyze the benefits and downsides of using a SAE and to potentially exploit a SAE for cluster analysis in the latent space. Currently, SAE can be used for ZSC by minimizing the k-dimensional Euclidean distance in the latent embedded space between an embedded input to multiple reference points of known classes. However, it will be further emphasized that this is not zero-shot "learning" since the current model can not be improved with SAE[1]. Rather, the classification model stays stagnant while the SAE is utilized in junction for one-time computations of classes which are **known to be unknown** to the classification model. It is believed that SAE has large potential in applications such as unsupervised learning without any training on the original classifier, which is discussed more in the *Future Work* section.

[1]As interpreted by the authors definition of ZSL. A model should be able to improve itself and flexibly incorporate unseen classes rather than use it as a parasitic building block for temporary modular components respective to each new classification.

It is planned to have a continuation of this study, as this document merely serves as an official documented point of progress in the author's personal ambitions of exploring ZSL.

II. Methodology

A. Semantic auto-encoder

The idea behind the proposed SAE by Kodirova, Xiang, and Gong [1] is using a linear auto-encoder to project an input data vector from the feature space into a k-dimensional latent space using a projection matrix. Likewise, another projection matrix is used to return back from the latent space to the feature space. This can be formulated by minimizing the following

$$\min_{\mathbf{W},\,\mathbf{W}^*} ||\mathbf{x} - \mathbf{W}^*\mathbf{W}\mathbf{x}||^2 \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^{d\times 1}$ is the input, $\mathbf{W} \in \mathbb{R}^{k\times d}$ and $\mathbf{W}^* \in \mathbb{R}^{d\times k}$ are the respectively latent and feature projection matrices, and the embedded latent space vector $\mathbf{s} \in \mathbb{R}^{k\times 1}$ is calculated by $\mathbf{Wx}$. The vectors can be extended into matrices $\left(\mathbf{X} \in \mathbb{R}^{d\times N} \text{ and } \mathbf{S} \in \mathbb{R}^{k\times N}\right)$ to include multiple (N) datapoints. Similarly, $\mathbf{S}$ now equals $\mathbf{WX}$. The latent space to feature space projection matrix $\mathbf{W}^*$ is proposed to be $\mathbf{W}^\top$ and a computationally efficient derivative calculation is derived

$$\mathbf{AW} + \mathbf{WB} = \mathbf{C} \tag{2}$$

where $\mathbf{A} = \mathbf{SS}^\top$, $\mathbf{B} = \lambda\mathbf{XX}^\top$, $\mathbf{C} = (1+\lambda)\mathbf{SX}^\top$, and λ is a weighting coefficient [1]. In turn, $\mathbf{W}$ can be solved using the `sylvester` equation using MATLAB, and this SAE algorithm from (2) can be recursively called to calculate a projection matrix which minimizes the error between $\mathbf{X}$ and $\hat{\mathbf{X}} = \mathbf{W}^\top\mathbf{S} = \mathbf{W}^\top\mathbf{WX}$.

B. Incorporation for zero-shot classification

Given the set of all classes Ω, each trained class t belong in the set T and each untrained class u belong in the set U, where T and U are subsets of Ω. It is implied that for a practical application, $T \cup U = \Omega$. The evaluation metric used, unless otherwise stated, is a k-dimensional Euclidean distance of the latent space projection $\forall \mathbf{s} \in \mathbf{S}$ with respect to some reference $\mathbf{S}^{\text{ref}}$. It should be noted that set notation on matrices (e.g. $\mathbf{s} \in \mathbf{S}$) refers to each element $\mathbf{s}$ of set $\mathbf{S}$ is a row-wise vector of the matrix $\mathbf{S}$.

For all unknown classes U, single instance of the class is used as the reference for classification. Meaning, given the evaluation metric, the closer the selected reference datapoint is to the average of all data within that class, the better chance it has at being in the "center" of the latent cluster, thus resulting in a better classifier. If an outlier (visually irregular / distorted handwritten character) is selected to represent the 'semantic baseline' for all further ZSC's using SAE, the worse the classification accuracy. For all known classes T, an average of all datapoints of like-classes within the latent space is used as the semantic baseline.

A well-trained classifier using deep learning architecture to extract spatial characteristics, such as 2-dimensional convolutional neural networks (CNNs), can be utilized to streamline the efficiency and accuracy of an SAE. Utilizing the activations of the final convolutional layer of a classifier (trained only on datapoints in classes T) the features can be encoded w.r.t. their one-hot classification labels using a projection matrix $\mathbf{W}_l$ to be fed through an SAE. The latent space encoding for trained data $\mathbf{S}_T$ is initialized randomly and fed through SAE to determine the first latent projection matrix $\mathbf{W}$. The projection matrix is reused to calculate $\mathbf{S}_T = \mathbf{W}\mathbf{X}_T$. Once a near-zero error is reached for the iterative calculation of $\mathbf{W}$, the means of each latent cluster to be used as classification references are determined by

$$\mathbf{S}_T = \mathbf{W}\mathbf{X}_T \tag{3}$$

where

$$\mathbf{X}_c = \mathbf{W}_l \left(\text{layer}_f \left\{\mathbf{X}_c^{\text{input}}\right\}\right) \tag{4}$$

where c is placeholder for class-type (i.e. U, T, etc.), $\text{layer}_f\{\mathbf{G}\}$ are the activations of a feature layer within the classification network given inputs $\mathbf{G}$, and

$$\mathbf{S}_T^{\text{ref}}(m) = \sum_{t_m \in T} \frac{\mathbf{S}_{t_m}}{|t_m|} \tag{5}$$

where $m = (0, 1, \ldots, |T|)$, $\forall \cup t_m = T$, and $|t_m|$ represents the number of data elements within class t_m. It should be noted that the averaging done on the r.h.s. of (5) could derive the unexisting reference datapoints indicated by $\mathbf{X}_{T*}^{\text{ref}}$ (T^* since the datapoint itself is unlikely to be in the set of T unless there only exists one data element for a given class).

One sample for each class of U ($|U|$ samples) produce our reference matrice $\mathbf{X}_U^{\text{ref}}$. The untrained datapoints $\mathbf{X}_U$ is updated by removing the selected elements in the reference set $\mathbf{X}_U^{\text{ref}}$ and used to calculate the semantic baseline, $\mathbf{S}_U^{\text{ref}}$, to represent the center of the latent space encoding their respective classes

$$\mathbf{S}_U^{\text{ref}} = \mathbf{W}\mathbf{X}_U^{\text{ref}} \tag{6}$$

and similarly

$$\mathbf{S}_U = \mathbf{W}\mathbf{X}_U \tag{7}$$

The semantic baseline references $\mathbf{S}_T^{\text{ref}}$ for trained datapoints and $\mathbf{S}_U^{\text{ref}}$ for untrained datapoints are now used to map the semantic evaluations of $\mathbf{S}_T$ and $\mathbf{S}_U$ onto U, T, and Ω based off the k-dimensional distance between each element of $\mathbf{S}_c$ w.r.t. some reference $\mathbf{S}_c^{\text{ref}}$.

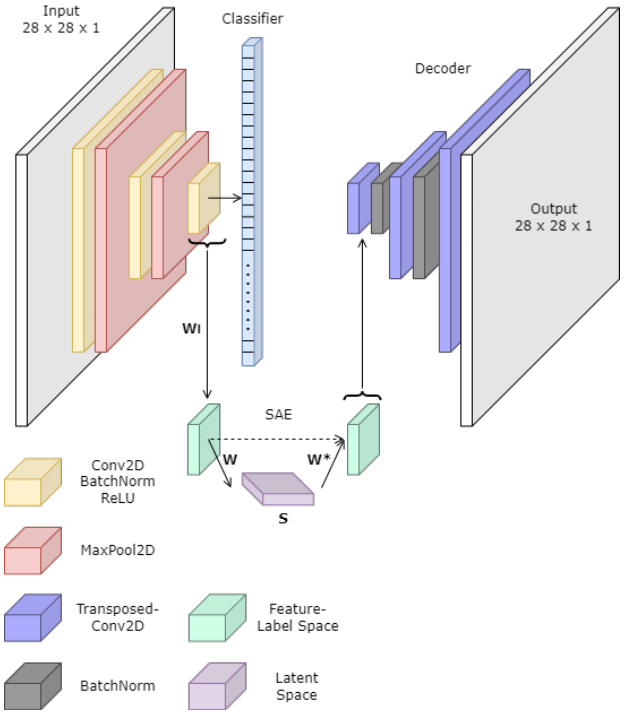

Fig. 1: Model architecture

$\mathbf{S}_U^{\text{ref}}$ is used to map results to U, $\mathbf{S}_T^{\text{ref}}$ is used to map results to T, and $\mathbf{S}_U^{\text{ref}}$ concatenated with $\mathbf{S}_T^{\text{ref}}$ is used to map results to Ω for classification.

A decoder is implemented independently from the ZSC algorithm described above to visualize the semantic references in the visual space. This is used to get a rough idea on how the linear combination of the closest known labels can 'construct' a never-seen-before class. In this report, the implemented decoder does not use the activations of the last feature layer of the classifier, but only the feature-label encoded variables $\mathbf{X}_c$. In *Future Work* it is discussed how using $\mathbf{x} \in \mathbf{X}$ as well as a constant $\mathbf{W}_l$ as inputs to the decoder network, better reconstruction can be achieved.

III. Implementation

The data used in this study consists of alphanumeric characters (0 through 9 and uppercase A through Z) are used from the original MNIST digit dataset [3] and the uppercase portion of the EMNIST dataset [4]. It should be noted this study uses a combined number of 35 classes instead of 36 to account for the similarity between 0 and an uppercase O. The data is converted to a binary format, where the threshold is half of the maximum value of the data/datatype (e.g. 0.5 if $[0, 1]$, 127 if `uint8`, etc.). This was not only done for decreasing storage dependency and increasing computational efficiency, but also for future incorporation of the Omniglot dataset, which consists only of binary data. More on this in *Future Work*.

All the networks were constructed using MATLABs deep learning library. A set number of classes are selected for

the seen and unseen datasets respectively. The classes are selected at random, and the datasets as well as labels are split accordingly.

The model architecture can be seen in Figure 1. The classifier was trained using the seen dataset and labels until a sufficiently high validation accuracy is reached. The activation's for the last hidden layer are extracted and used as inputs for the SAE. In theory, any feature layer can be selected, even the raw input. However, due to the nature of CNN classifiers, the hidden layers closer to the output embed more 'feature' information rather than spatial/temporal information. This can decrease iterations of the SAE in finding the latent projection matrix $\mathbf{W}$.

The latent space representations of data to be evaluated compute a k-dimensional Euclidean distance between itself and semantic baselines for different class sets given by $\mathbf{S}_c^{\text{ref}}$. In this study, k was limited to 256, though $k \in \mathbb{N}$. In addition, the parameter λ was usually 1.

The decoder used a transposed CNN architecture to decode feature-label space data $\hat{\mathbf{X}}_c = \mathbf{W}^\top \mathbf{W}\mathbf{W}_l \left(\text{layer}_f \left\{\mathbf{X}_c^{\text{input}}\right\}\right)$ for visuals only. This is intended to be used as a precursor to generative modeling as well as unsupervised learning (see *Future Work*), but for the time being the decoder is included purely for cosmetic purposes.

IV. Results

Run	$\lvert U\rvert$ / $\lvert\Omega\rvert$	% Trained	ZSC Mapping	Top 1	Top 3
R_0	9 / 35	82.60	$\mathbf{S}_U \longrightarrow U$	**83.8**	**97.0**
R_0	9 / 35	82.60	$\mathbf{S}_U \longrightarrow \Omega$	17.0	72.1
R_0	9 / 35	82.60	$\mathbf{S}_T \longrightarrow T$	93.3	99.3
R_0	9 / 35	82.60	$\mathbf{S}_T \longrightarrow \Omega$	92.1	98.9
R_0	9 / 35	82.60	$\mathbf{S}_\Omega \longrightarrow \Omega$	**84.2**	**96.7**
R_1	23 / 35	42.04	$\mathbf{S}_U \longrightarrow U$	**53.8**	**81.1**
R_1	23 / 35	42.04	$\mathbf{S}_U \longrightarrow \Omega$	23.4	70.2
R_1	23 / 35	42.04	$\mathbf{S}_T \longrightarrow T$	96.3	99.8
R_1	23 / 35	42.04	$\mathbf{S}_T \longrightarrow \Omega$	90.8	98.7
R_1	23 / 35	42.04	$\mathbf{S}_\Omega \longrightarrow \Omega$	**54.5**	**83.7**

TABLE I: ZSC Accuracy from the A-Z, 0-9 MNIST dataset

Two runs were completed with a different number of trained classes in each. Both require separate classifiers, SAE projection matrices, as well as decoders. The first run trained the model with a randomly selected majority (26 / 35) of all classes (82.60% of data), leaving 9 unseen classes: $U = \{$ 2, 6, 7, 9, E, L, V, W, Y $\}$. The second run trained the model with a randomly selected minority (12 / 35) of all classes (42.04% of data), leaving 23 unseen classes: $U = \{$ 0 or O, 1, 2, 3, 4, 6, 7, 8, A, B, D, E, H, I, J, K, L, M, P, V, W, X, Y $\}$

The bolded accuracies do not refer to the best results, since there is no other ZSL model being compared. Rather, the bolded accuracies refer to results of interest. The first row w.r.t. each run in Table I refer to the evaluation metric used by Kodirova, Xiang, and Gong [1], where only latent embedded datapoints from unseen classes $\mathbf{S}_U$ are evaluated by mapping the likelihood of $\mathbf{s}_U$ being in the set of unseen classes U. However, this is only realistic in an application in which a new class (known to applicant of the model and unknown to the model) is introduced to the model, and only new classes

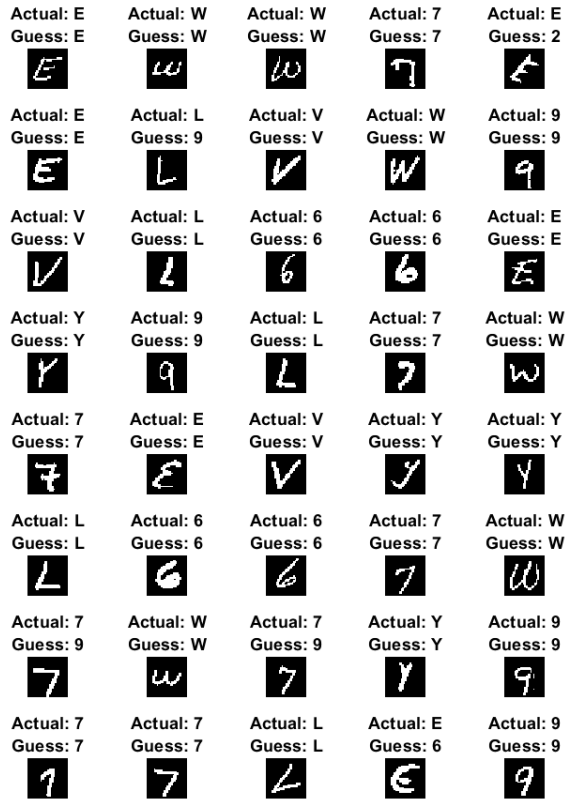

Fig. 2: R_0: $\mathbf{S}_U \rightarrow U$ for 9 unseen + 26 trained classes

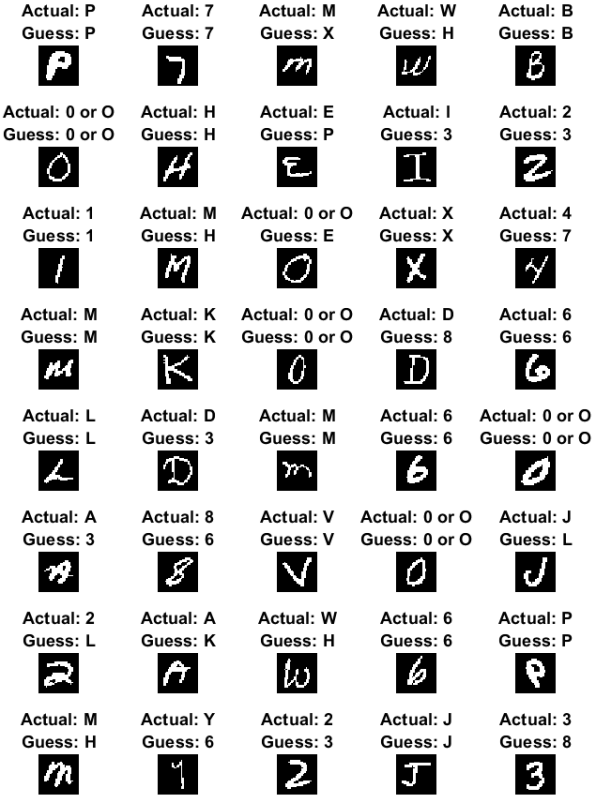

Fig. 3: R_1: $\mathbf{S}_U \rightarrow U$ for 23 unseen + 12 trained classes

can be evaluated and classified as elements of U. This is unrealistic, because it implies that a different method of classification (semantic baseline $\mathbf{S}_c^{\text{ref}}$) is used in regards to the class type c of the input data. This further implies that the applicant must differentiate and separate $\mathbf{X}_\Omega^{\text{input}}$ into $\mathbf{X}_T^{\text{input}}$ and $\mathbf{X}_U^{\text{input}}$ for evaluation. A better approach would be updating the model to incorporate U so that an overarching semantic baseline of $\mathbf{S}_\Omega^{\text{ref}}$ can map any latent embedded input data $\mathbf{S}_\Omega$ onto Ω for classification. This metric is used to represent applications in which a model needs to blindly evaluate some data $\in \Omega$ after being introduced to U. The ZSC accuracies for this mapping is the last row w.r.t. each run in Table I.

It can be observed that the accuracies for classifying unseen classes of R_0 performed significantly better than R_1 due to R_0's exposure to more classes. There was a 83.8% Top 1 classification rate for unseen classes in R_0 when restricting the classification domain to U seen in Figure 2. There was a 53.8% Top 1 classification rate for unseen classes in R_1 when restricting the classification domain to U seen in Figure 3. Both Figures 2 and 3 evaluate randomly selected $\mathbf{x}_U^{\text{input}} \in \mathbf{X}_U^{\text{input}}$ of runs R_0 and R_1, respectively. Refer to Tables II and III for the ranked distributions of Top 1 classifications.

It should be noted that all mapping accuracies incorporating U (all except $\mathbf{S}_T \rightarrow T$) depend on the selection of semantic baselines of unseen classes $\mathbf{S}_U^{\text{ref}}$. Since the baselines were chosen at random, there exists a $\mathbf{S}_U^{\text{ref}}$ which can yield a better or worse accuracy in each category. Demos of the results with data and algorithms can be found at https://github.com/arpadav/zsl_sae_matlab

A decoder was used for visualization only gain greater insight about the latent space as well as how the model and SAE perform. Figure 4 shows U on the left half and T on the right half with the following information:

- U:
  - $\mathbf{X}_U^{\text{ref, input}}$ - The input image (used as the semantic baseline for the given unseen class)
  - $\text{layer}_f \left\{ \mathbf{X}_U^{\text{ref, input}} \right\}$ reconstructed through feature decoder - The features of the input image (used as the semantic baseline for the given unseen class) reconstructed through a feature decoder
  - $\mathbf{X}_U^{\text{ref}}$ reconstructed through decoder - The latent space baseline for the given unseen class, transformed **out of** latent space and reconstructed through a decoder.
  - Random $\mathbf{x}_U$ reconstructions through decoder - the higher similarity to baseline means higher chance corrent classification.
- T:
  - $\mathbf{X}_{T*}^{\text{ref, input}}$ - Not possible to reconstruct
  - $\text{layer}_f \left\{ \mathbf{X}_{T*}^{\text{ref, input}} \right\}$ - Not possible to reconstruct
  - $\mathbf{X}_T^{\text{ref}}$ reconstructed through decoder - The latent space baseline for the given seen class, transformed **out of** latent space and reconstructed through a decoder.
  - Random $\mathbf{x}_T$ reconstructions through decoder - the higher similarity to baseline means higher chance corrent classification.

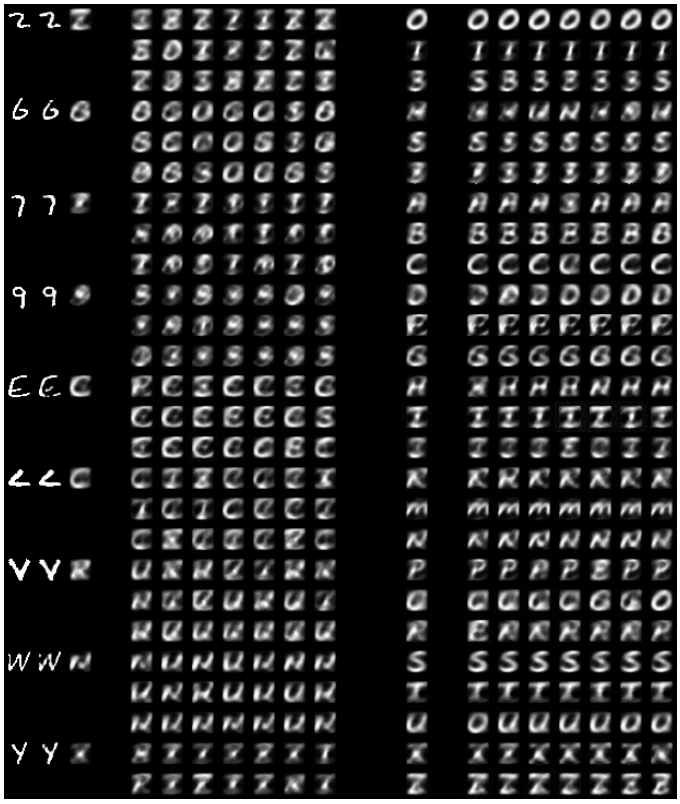

Fig. 4: Reconstruction of R_0 classes. U on left, T on right

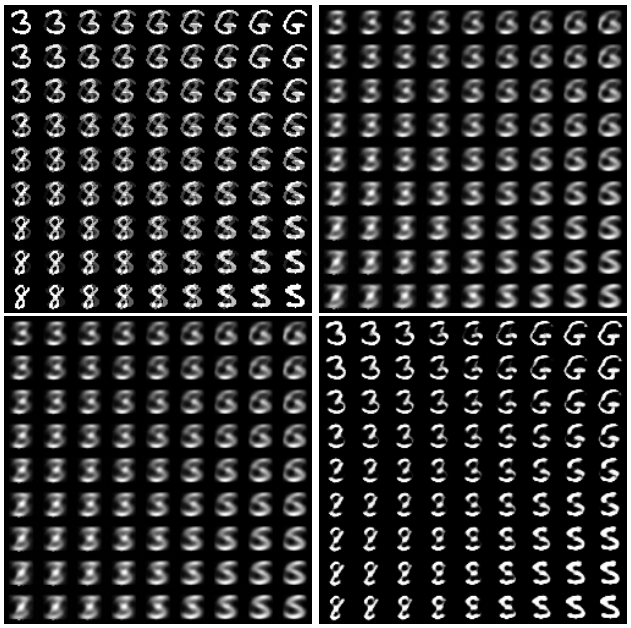

Fig. 5: Bilinear interpolation in different spaces for T of R_0

Figure 5 shows bilinear interpolation of the sampled characters in $\{$ 3, G, 8, S $\} \subset T$ in run R_0. Top left of Figure 5 shows what this looks like in the spatial domain, where the drawings fade into one another. Top right of Figure 5 shows interpolation in the latent space and then reconstructed. Bottom right of Figure 5 shows interpolation in the one-hot

encoded space (transformed by projection matrix $\mathbf{W}_l$), which looks near-identical to interpolation in the latent space. Bottom left of Figure 5 shows interpolation of the features given by the activations of the feature layer, decoded through a different (feature) decoder.

It can be seen that there is a significant amount of information loss when transforming the feature space representation using projection matrix $\mathbf{W}_l$ (addressed in *Future Work*), since $\mathbf{W}_l$ optimized to minimize the error between the features of classes in T and the respective one-hot encodings which represent T.This means that decoding using **any** stage after the transformation of $\mathbf{W}_l$ results in information only given by this 'label' space. This shows how SAE is powerful when coming to classification, but it severely lacks potential when it comes to ZSL generative tasks in its current state.

V. Future Work

For unsupervised learning, the definition of $T \cup U = \Omega$ is completely accurate in describing the current system. It is unknown whether the addition of the set of untrained classes U completes the set Ω. Rather, it can be formulated that the only information the applicant as well as the model have to work with is $T \cup U^* \subseteq \Omega$, where $U^* \subseteq U$ and U^* is the ***observed*** set of classes unseen by the model. The model must be able to interpret any new $u^* \in U^*$ and perform the merging of sets $T \leftarrow T \cup U^*$. Note that this does not mean the model must retrain using traditional training methods, it simply means the model must account for unseen classes with a high enough accuracy to be considered on-par with T using some ZSL method.

As mentioned throughout the report, there is a significant amount of information lost when embedding the features of the input images in the latent space to extract some semantic meaning. This is because implementation of the SAE works in a rigid manner where the T (and therefore $|T|$) stays constant, and there is a increased emphasis on ZSC rather than ZSL. Therefore, I would like to further analyze the mathematics to formulate a method which incorporates SAE in the one-shot updating of the weights and biases after the feature layer. The key components at play are projection matrix $\mathbf{W}_l$, latent space projection matrix $\mathbf{W}$, and the final dense layer of the network. $\mathbf{W}_l$ correlates feature information with classification information, and $\mathbf{W}$ inherently portrays information about how any input is interpret ted as a linear combination of the known outputs. Given a new class, the k-Euclidean distance evaluation used for SAE can give enough statistically significant information that a new class is present. Therefore, action can be taken in updating the classification layer to include the extra class, and SAE can be rerun to incorporate the newly added class. This can easily be done, it's just a question of which methods yield the 'proper technique'.

As stated in the paragraph above, the information loss resulting in the heavy emphasis on classification using the naïvely implemented SAE means the trained decoder has no feature information to work with. Since this decoder is essentially using one-hot labels as inputs and a vast number of intricate outputs, it's by no surprise that it does such a poor job in reconstructing never-seen-before classes in an accurate manner. Rather than training a decoder with only $\mathbf{X}_T$ or $\mathbf{S}_T$ as inputs, I would like to incorporate $\mathbf{W}_l$ as a *constant* input to the decoder. This provide the decoder with feature transformation information in an attempt to pseudo-reverse the transformation by means of deep learning. Once back in the feature domain (though different than the original one) the decoder can be more successful in reconstruction (as seen by the feature-only decoder in left-most columns of Figure 4 and the lower right of Figure 5). This will increase the robustness of the decoder, which can allow for generation of never-seen-before classes embedded within the learned latent-space without any prior reference or observation.

Lastly, this study plans to continue with expanding the dataset to include the Omniglot dataset. There is a signficant increase in the number of classes with overwhelmingly less individual datapoints within each class. This makes it an ideal candidate to test ZSL methods.

VI. Acknowledgment

Special thanks to Dr. Tianfu (Matt) Wu for overseeing this study and introducing me to the topic of ZSL.

References

[1] E. Kodirova, T. Xiang, and S. Gong, "Semantic autoencoder for zero-shot learning," *IEEE CVPR 2017*, July 2017.

[2] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," 2015.

[3] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010.

[4] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik, "Emnist: an extension of mnist to handwritten letters," *arXiv preprint arXiv:1702.05373*, 2017.

	Character	1^{st}	%	2^{nd}	%	3^{rd}	%	4^{th}	%	5^{th}	%	Rank
U	2	Z	43.46	3	20.17	J	9.99	Q	8.60	8	6.29	7
U	6	**6**	**40.68**	G	34.48	B	5.10	1	4.99	8	4.99	1
U	7	**7**	**41.89**	4	24.24	9	12.99	3	8.14	1	6.14	1
U	9	4	52.21	**9**	**38.29**	8	2.90	Q	2.64	3	1.62	2
U	E	**E**	**34.84**	F	22.38	G	14.87	C	7.17	Z	6.21	1
U	L	**L**	**55.25**	Z	14.24	1	10.80	I	6.99	C	5.76	1
U	V	U	49.57	**V**	**29.22**	Y	8.78	K	3.08	N	3.04	2
U	W	**W**	**58.18**	U	23.11	N	13.87	H	1.34	M	1.06	1
U	Y	**Y**	**36.11**	X	31.65	1	11.40	4	3.72	P	3.46	1
T	0 or O	**0 or O**	**86.63**	D	9.29	Q	2.06	6	0.84	B	0.26	1
T	1	**1**	**98.67**	I	0.36	J	0.32	3	0.23	4	0.18	1
T	3	**3**	**97.94**	J	0.52	B	0.35	8	0.34	5	0.21	1
T	4	**4**	**95.05**	H	1.77	U	1.17	A	0.92	1	0.34	1
T	5	**5**	**92.76**	S	3.17	3	0.82	J	0.71	8	0.65	1
T	8	**8**	**94.30**	B	3.59	1	0.31	9	0.29	P	0.25	1
T	A	**A**	**97.56**	H	0.69	R	0.40	M	0.37	1	0.26	1
T	B	**B**	**97.25**	8	0.95	3	0.62	R	0.23	A	0.20	1
T	C	**C**	**95.79**	L	1.91	G	0.75	T	0.23	U	0.21	1
T	D	**D**	**96.10**	J	1.07	B	0.84	0 or O	0.35	M	0.22	1
T	F	**F**	**99.66**	J	0.17	G	0.09	T	0.09	-	-	1
T	G	**G**	**96.23**	B	1.09	Q	0.90	6	0.24	C	0.23	1
T	H	**H**	**95.28**	4	1.05	N	0.96	A	0.57	K	0.47	1
T	I	**I**	**96.70**	J	2.14	1	0.45	2	0.27	Z	0.18	1
T	J	**J**	**96.61**	T	1.20	1	0.58	U	0.29	I	0.26	1
T	K	**K**	**96.98**	X	0.86	H	0.70	R	0.45	1	0.14	1
T	M	**M**	**98.14**	H	1.00	N	0.34	A	0.21	W	0.12	1
T	N	**N**	**95.18**	W	1.98	H	0.76	M	0.44	J	0.26	1
T	P	**P**	**96.72**	D	0.68	F	0.52	1	0.38	R	0.37	1
T	Q	**Q**	**96.20**	P	0.71	9	0.65	G	0.58	R	0.57	1
T	R	**R**	**94.71**	A	1.54	K	0.99	B	0.42	Q	0.41	1
T	S	**S**	**87.31**	5	7.90	J	2.83	G	0.52	8	0.39	1
T	T	**T**	**97.95**	F	0.72	1	0.33	7	0.17	4	0.12	1
T	U	**U**	**97.29**	4	0.57	M	0.26	H	0.23	W	0.21	1
T	X	**X**	**96.38**	K	2.23	Y	0.32	Z	0.24	R	0.18	1
T	Z	**Z**	**97.78**	2	0.66	3	0.25	I	0.21	X	0.18	1

TABLE II: R_0 Top 1 $\mathbf{S}_\Omega \rightarrow \Omega$ classifications rates for all available data

	Character	1^{st}	%	2^{nd}	%	3^{rd}	%	4^{th}	%	5^{th}	%	Rank
U	0 or O	Q	45.26	**0 or O**	**25.25**	C	13.87	2	2.73	G	2.39	2
U	1	**1**	**66.97**	7	17.34	T	9.90	Y	1.55	L	0.83	1
U	2	Z	67.17	3	8.48	**2**	**8.15**	Q	4.98	L	1.53	3
U	3	5	35.79	Z	28.15	**3**	**23.25**	9	4.89	S	2.51	3
U	4	9	60.73	Y	6.81	8	6.42	**4**	**6.14**	W	5.04	4
U	6	**6**	**33.58**	G	33.01	5	18.94	8	8.93	U	1.02	1
U	7	9	73.10	**7**	**23.52**	Z	1.52	T	0.44	1	0.40	2
U	8	5	30.84	9	20.41	B	13.58	**8**	**8.64**	3	6.21	4
U	A	R	40.51	K	23.13	P	10.14	**A**	**8.99**	G	4.74	4
U	B	**B**	**37.49**	G	24.93	Z	6.95	K	6.63	R	5.61	1
U	D	Q	29.57	**D**	**14.67**	2	13.29	Z	10.92	3	6.65	2
U	E	F	33.65	C	18.95	G	17.91	B	8.27	Z	6.31	8
U	H	**H**	**34.37**	M	14.87	N	13.22	X	11.93	K	8.09	1
U	I	Z	45.80	**I**	**28.39**	3	9.20	J	8.12	L	4.02	2
U	J	**J**	**26.03**	I	21.16	3	13.91	S	11.87	L	11.29	1
U	K	R	36.64	**K**	**25.41**	A	15.65	X	13.33	C	1.89	2
U	L	C	55.45	**L**	**22.15**	Z	14.85	E	1.78	R	1.23	2
U	M	N	55.71	**M**	**19.31**	H	8.93	X	5.16	T	3.27	2
U	P	F	57.94	**P**	**24.96**	T	11.19	R	1.49	Q	0.99	2
U	V	W	30.34	U	23.41	Y	18.84	**V**	**18.20**	N	6.98	4
U	W	**W**	**43.27**	H	34.58	N	16.74	V	1.83	M	1.21	1
U	X	**X**	**66.85**	N	4.74	Y	4.42	K	4.24	M	3.84	1
U	Y	1	32.78	T	30.54	Y	16.23	N	5.00	4	2.21	3
T	5	**5**	**89.88**	S	5.58	8	2.96	9	0.98	6	0.17	1
T	9	**9**	**97.97**	5	0.79	4	0.43	S	0.23	8	0.13	1
T	C	**C**	**98.39**	G	0.67	E	0.21	F	0.15	Z	0.10	1
T	F	**F**	**99.40**	G	0.17	S	0.17	T	0.17	I	0.09	1
T	G	**G**	**98.16**	6	0.26	C	0.24	4	0.16	Q	0.16	1
T	N	**N**	**96.51**	H	2.37	X	0.23	M	0.19	Z	0.08	1
T	Q	**Q**	**94.98**	G	2.32	P	0.58	4	0.41	0 or O	0.36	1
T	R	**R**	**94.48**	A	1.31	P	1.19	K	1.07	C	0.43	1
T	S	**S**	**96.54**	5	1.68	3	0.70	G	0.41	J	0.24	1
T	T	**T**	**98.87**	F	0.51	1	0.16	I	0.14	Z	0.10	1
T	U	**U**	**91.30**	V	3.03	Y	2.45	W	2.13	E	0.17	1
T	Z	**Z**	**99.18**	T	0.25	L	0.23	2	0.08	3	0.07	1

TABLE III: R_1 Top 1 $\mathbf{S}_\Omega \rightarrow \Omega$ classifications rates for all available data