# One-Shot Learning Using Gaussian Mixture Models and Variational Autoencoders

**Chelsea Zou** [1] **Connor Jennings** [1] **David Sanders** [1]

## Abstract

We introduce a generative, probabilistic, cluster-based approach for one-shot learning to model single examples of each class by forming an ensemble of their subclusters using Gaussian Mixture Models (GMMs). Sampling from the inferred parameters of the Gaussian clusters generates new cluster representations of the class, which is what we propose as Abstracted Gaussian Prototype (AGPs). Using AGPs, we can synthetically increase the size of the training set to employ a variational autoencoder (VAE) to learn a continuous latent space over the AGPs to generate new variants of different classes.

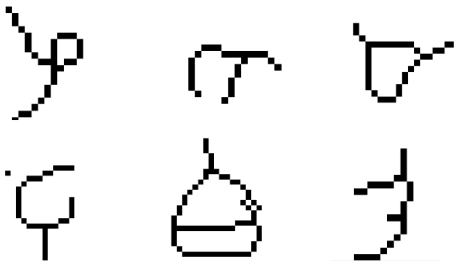

*Figure 1.* Newly generated characters from our AGP-VAE pipeline.

## 1. Introduction

The ability of humans to acquire novel concepts after only minimal exposure to examples is an important constituent of general intelligence. Humans have the remarkable ability to quickly abstract concepts and extrapolate from a few prior examples (Lake et al., 2015), allowing for efficient and adaptable learning. On the contrary, current machine learning architectures require large amounts of data to learn from. As a result, the performance of these models significantly diminishes when data is scarce. Hence, a key computational challenge is to understand how an intelligent system can acquire novel classes, given a modest amount of data. This emerged the field of few-shot learning (Wang et al., 2020; Kadam & Vaidya, 2020), which seeks to computationally mimic human reasoning and learning with limited data.

The objective of few-shot learning is to create a model that can learn classes while only being allowed a small number of training data for each class. Here, we address the challenge of one-shot learning, where only a *single* training instance per class is allowed to be used. Specifically, we focus on a generative task on the Omniglot Dataset (Lake et al., 2019), which is a large image dataset of handwritten characters. This generative task we will be focusing on involves creating entirely new variants of handwritten characters, given only one example per class from a set of characters. The goal is to have the model learn and generate new characters that look plausible.

We propose a novel approach that only utilizes a single instance per class to learn effectively, by leveraging the probabilistic framework of Gaussian Mixture Models (GMMs). GMMs are unsupervised clustering models that have the ability to model complex data distributions by inferring a combination of its simpler distributions (McLachlan & Basford, 1988). Particularly, these simpler distributions correspond to distinct clusters, which are the mixture model's Gaussian components, parameterized by unique means and standard deviations (Yu et al., 2015). Essentially, the overall data distribution is represented as a mixture of its individual components. To be concise, note that we will use the terms *components*, *subparts*, *clusters*, and *segments* to refer to the same idea. While GMMs are traditionally used for discriminative clustering tasks, they can also be used to generate new data by sampling from the learned parameters of the inferred distributions (Liang et al., 2022; Reynolds et al., 2009).This can be particularly useful in generating meaningful data in the realm of one-shot learning, where data is limited and confined to one example.

Our approach leverages GMMs as the fundamental tool to create new data prototypes of each class. A separate GMM is used to model each class, where each Gaussian component represents a distinct topological subpart of the class. By sampling from the inferred parameters of each cluster,

[1]Binghamton University, NY, US. Correspondence to: <>.

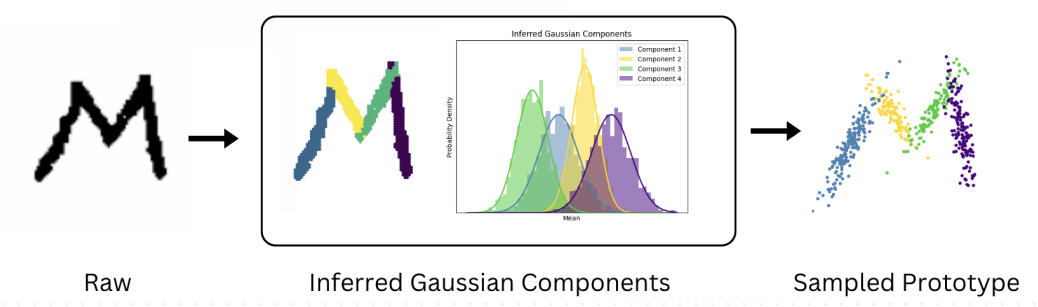


*Figure 2.* The raw image is shown on the left. The inferred clusters of the GMM is shown in the middle. Finally, the newly generated abstracted prototype is sampled from the inferred parameters.

probabilistically augmented subparts are generated. The collective ensemble of these subparts form what we propose as the Abstracted Gaussian Prototype (AGP), see Figure 2. AGPs provide a way to generate new data belonging to a particular class.

However, AGPs can only represent one class at a time. In order to generate new diverse variants of entirely new classes, we employ a variational autoencoder (VAE) to learn a continuous latent space that encapsulates a probabilistic distribution over all the generated AGPs. VAEs can learn meaningful representations across classes through an encoder-decoder architecture that maps input features into a lower-dimensional, continuous latent space (Kingma et al., 2019). Through variational inference, the encoder approximates the true posterior distribution with a variational distribution, typically assumed to be Gaussian (Kingma & Welling, 2013). Therefore, the latent space is not confined to discrete categories, but instead enables the model to probabilistically sample between learned concepts and reconstruct new variations. Our formulation of this novel AGP-VAE pipeline interpolates between subclusters of the AGPs by sampling from a global feature space that encapsulates the local features of different classes.

## 2. Background

In this section, we provide the mathematical background underlying GMMs and VAEs which are fundamental to our approach.

### 2.1. Gaussian Mixture Models

A GMM is a probabilistic clustering model that assumes the data is generated from a combination of multiple Gaussian distributions. Each Gaussian component $k \in K$ represents a cluster in the dataset, and is characterized by its unique parameters mean $\mu$, standard deviation $\sigma$, and a weight $\pi$. A univariate Gaussian probability density function (PDF) for random variable $X$, which represents the probability of observing the given datapoint, is defined as the following:

$$P(X|\mu,\sigma) = \tag{1}$$

$$\mathcal{N}(\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{(X-\mu)^2}{2\sigma^2}\right)$$

Similarly, a multivariate Gaussian PDF is defined as:

$$\mathcal{N}(\mu,\Sigma) = \tag{2}$$

$$\frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(X-\mu)^T\Sigma^{-1}(X-\mu)\right)$$

where $X$ is now a d-dimensional feature vector, $\mu$ is a d-dimensional vector representing the means of the distribution, and $\Sigma$ is a d x d covariance matrix. Finally, a finite GMM can be expressed by a weighted sum over K components:

$$P(X|\pi,\mu,\Sigma) = \sum_{k=1}^{K} \pi_k * \mathcal{N}(X|\mu_k,\Sigma_k) \tag{3}$$

satisfying the condition where:

$$\sum_{k=1}^{K} \pi_k = 1 \tag{4}$$

### 2.2. Variational Autoencoders

A variational autoencoder (VAE) is a type of neural network architecture that has gained significant attention in generative modeling, due to their ability to learn continuous representations of discrete input classes (Kingma & Welling, 2013). Variational inference is a probabilistic framework used to approximate complex posterior distributions by optimizing a simpler, parameterized distribution. In VAEs, the goal is to infer the latent variables given the observed data. This is achieved by introducing a probabilistic encoder that maps data points to a distribution in the latent space. The

encoder outputs both the mean and the variance of the approximate posterior distribution, which is typically assumed to be Gaussian.

**Encoder:** The encoder of a VAE maps input data $x$ to a latent space variable $z$, and is defined by an approximate posterior distribution:

$$q_\phi(z|x) = \mathcal{N}(z; \mu_z(x), \sigma_z(x)^2) \tag{5}$$

where $\mu_z(x)$ and $\sigma_z(x)$ are the mean and standard deviation of the approximate posterior learned by the encoder's weights and biases $\phi$.

**Sampling with Reparametrization Trick:** To obtain a sample $z \sim q_\phi(z|x)$ from the latent space learned by the encoder, the reparametrization trick is used to maintain differentiability for backpropagation:

$$z = \mu_z(x) + \sigma_z(x) \odot \epsilon \tag{6}$$

where $\epsilon$ is typically sampled from a fixed standard normal distribution:

$$\epsilon \sim \mathcal{N}(0, 1). \tag{7}$$

**Decoder:** The decoder of a VAE with parameters $\theta$ maps a sample $z$ from the latent space back to the original feature space, generating a reconstruction $\hat{x}$ from the conditional likelihood distribution:

$$p_\theta(x|z) = \mathcal{N}(\hat{x}; \mu_x(z), \sigma_x(z)^2) \tag{8}$$

where $\mu_x(z)$ and $\sigma_x(z)$ are the mean and standard deviation of the reconstructed data.

**Loss Function:** The training objective for VAEs is to maximize the Evidence Lower Bound (ELBO). The ELBO is defined by:

$$\mathcal{L}(\phi, \theta) = \tag{9}$$
$$E_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}(q_\phi(z|x)||p(z))$$

where the first term is the expected value of the log-likelihood of $x$ given $z$ and the second term is the Kullback-Leibler (KL) divergence between the approximate posterior and the prior distribution. The KL divergence measures the difference between two probability distributions and acts as a latent space regularization term that encourages the learned approximate posterior space to be close to the true posterior. Let $J$ be the dimensionality of $z$. For the case of two Gaussian distributions, it is defined as

$$D_{KL}(q_\phi(z|x)||p_\theta(z)) = \tag{10}$$
$$\frac{1}{2}\sum_{i=1}^{J}\left(\mu_i^2 + \sigma_i^2 - \log(\sigma_i^2) - 1\right)$$

# 3. Approach

In this section, we formalize our approach to the generative tasks of one-shot learning on the Omniglot Challenge. The Omniglot dataset consists of 1623 hand-written characters taken from 50 different alphabets, with 20 examples for each class. Given the limited number of available examples for each class, it is often used to evaluate few-shot learning approaches.

First, a separate GMM is used to model each concept, where each cluster of the model is assumed to represent a unique subpart of the concept. Next, we generate newly abstracted subparts of each cluster by probabilistically sampling from its fitted parameters. The collective ensemble of these generated subparts form what we refer to as the AGP, denoted $\mathcal{P}$. Overall, this can be thought of as a generative scheme which produces new prototypes $\mathcal{P}$ for each class. Instead of relying solely on the single provided examples for each class, we can now generate multiple prototypes to synthetically increase the size of the training set. The details are discussed below.

#### 3.0.1. Abstracted Gaussian Prototype Generation

In a one-shot learning task, there is a set of $N$ classes, denoted as $\mathcal{C} = \{c_1, c_2, \ldots, c_N\}$, and one available instance of each class. The given set of these single instances is denoted as $\mathcal{X} = \{x_1, x_2, \ldots, x_N\}$. Each instance of a class is provided as a binary image of pixels. Under the probabilistic framework of a GMM, let us define each sampled pixel as the realization of a random variable, characterized by its PDF. Each instance of a concept in $\mathcal{X}$ is first segmented into its unique sub-parts using a GMM, where $\mathcal{G} = \{g_1, g_2, \ldots, g_k\}$ represents the set of different segments in each instance and $k$ is a hyperparameter controlling the number of components in a GMM. Here, $\mathcal{G}$ represents the mixture of Gaussian components of each instance, which allows the GMM to sample from the fitted distribution for each component $g_i$ and generate new augmented subparts $p_i$. We define the ensemble of these subparts as the prototype $\mathcal{P}$ of the class, where $\mathcal{P} = \{p_1, p_2, \ldots, p_k\}$.

### 3.1. Generative Tasks

There are three major steps to generate new variations of exemplars or classes given only a single instance of each class. First, we harness the ability of GMMs to generate various prototypes to synthetically increase the amount of training data available. This introduces more variation and diversifies the training set. Second we leverage a VAE that trains across all the synthetic data to generate continuous new variations amongst the starting set. Finally, we use a post-processing topological skeleton technique (Lee et al., 1994; Zhang, 1997) to refine the generated outputs. The

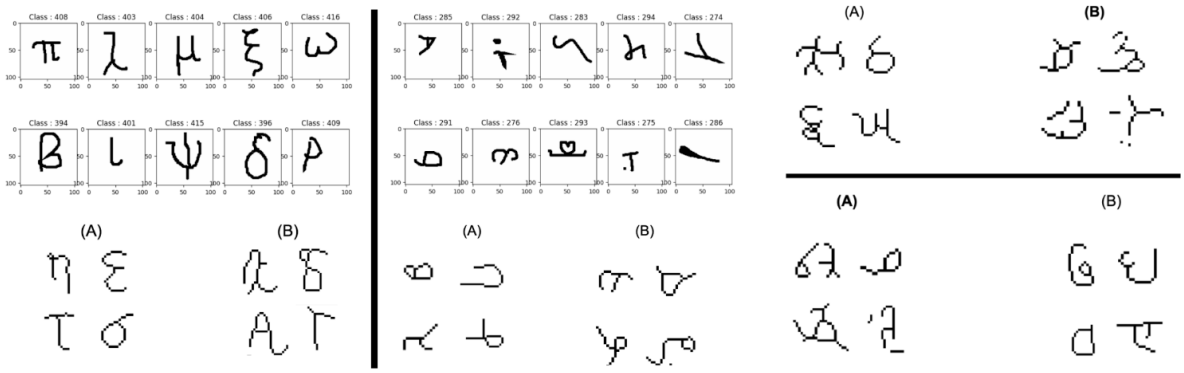


*Figure 3.* Visual turing tests of the output characters generated from our AGP-VAE pipeline. The set of characters drawn by the model is (B, B) from left to right and (B, A) from top to bottom

pseudocode is shown in Algorithm 1.

#### 3.1.1. AGP Training Set

The approach from section 3.0.1 is used to generate more prototypes to synthetically increase the size of a training set $\Phi$. This training set consists of a larger set of prototypes where $\Phi = \{\mathcal{P}_1, \mathcal{P}_1, \ldots, \mathcal{P}_D\}$, containing $D$ new prototype variants for each class. Generating a variety of prototypes for each concept increases the diversity of the training data, which is essential in improving the generalization capabilities of models trained with limited samples.

#### 3.1.2. VAE Interpolation

After each GMM generates $D$ new prototypes for each class to create the new train set $\Phi$, the next step is to create continuous variations among these prototypes. To accomplish this, a single VAE model trains across $\Phi_i$ for $i \in N$ classes to learn a latent space representation that captures the underlying structures of these abstracted prototypes. The key advantage is that this latent space continuous, allowing for coherent and semantically meaningful interpolation between subparts of the discrete prototypes created by the GMMs. The latent variables $z$ are sampled accordingly to encourage semantic mixing between prototypes, which are then decoded into the reconstructed variant images.

#### 3.1.3. Topological Skeleton Refinement

The final step in this process involves a post-processing technique based the work of (Lee et al., 1994; Zhang, 1997) on topological skeletons. Skeletonization is used often in image processing and computer vision to reduce the thickness of a binary object to a one-pixel-wide representation, while still preserving the topological properties of the object. This step further refines the reconstructed output images generated by the VAE, and emphasizes the stroke-like properties of Omniglot characters. After each reconstructed image from the VAE is skeletonized, the final result is a collection of generated variants of characters for each task, see Figure 1.

**Algorithm 1** Generating New Variants

**Input:**
$\mathcal{X} \leftarrow$ set of single instances from $N$ classes
GMM, VAE $\leftarrow$ trainable models
Skel $\leftarrow$ topological skeleton function
**Output:**
$\Phi \leftarrow$ set of $\mathcal{P}$ prototypes
$V \leftarrow$ final generated variants

**for** *i in N* **do**
  train a GMM$_i$(x$_i$)
  $\mathcal{P} \leftarrow$ sample from GMM$_i$
  $\Phi \leftarrow \Phi \cup \{\mathcal{P}\}$ append $\mathcal{P}$ to set of prototypes
  train VAE($\Phi$) across $\Phi$
  $z \leftarrow$ sample and reconstruct from VAE latent space
  $V \leftarrow$ Skel($z$) postprocess reconstruction
**end**

## 4. Results

### 4.1. Generative Tasks

A "visual turing test", as described in (Lake et al., 2013) is used to assess the quality of the generative outputs of the model. In this test, a set of characters produced by a human is displayed next to a set produced by the model, see Figure 3. Human judges then try to identify which set was drawn by a human, and which set was generated by the model. Our generative approach is evaluated based on the identification

| | Identification Accuracy | Preference |
|---|---|---|
| Mean | 52.25% | 55.25% |
| SD | 8.13% | 8.35% |
| Min | 40.00% | 43.00% |
| Max | 63.00% | 70.00% |

*Table 1.* Descriptives for Average Scores Across Judges

accuracy of 20 human judges recruited online. The ideal performance is 50 percent, indicating that the judges cannot distinguish between characters produced by the human and the model, and the worst-case performance is 100 percent.

Additionally, we asked follow up questions after displaying each set of images to probe if the machine's outputs could potentially surpass the quality of human generated characters. The question was phrased "Which set do you think represents a better job of creating four new characters?".

**Overall Results for Generative Tasks** The overall identification accuracy and preference scores averaged across 20 judges are revealed in Table 1 for 10 sets of questions. Furthermore, Figure 4 displays the breakdown of scores to reveal the subjective evaluations between each individual judge. These overall scores reveal promising results, as identification accuracy is close to random chance. Remarkably, preference for machine generated characters rank slightly higher than the human-generated characters, which could prompt for further exploration within AI-generated content.

## 5. Conclusion and Future Works

In this paper, we have presented a novel approach for addressing the challenging problem of one-shot learning using AGPs. AGPs leverage GMMs to build representative prototypes for each class by abstracting upon the subparts of the single available instances for each class. First, we proposed the AGP approach for generating new data from a single example per class. Second, we developed a generative pipeline employing VAEs to utilize the AGPs for generating *new* classes.

While our approach presents promising contributions to one-shot learning, there are limitations to be acknowledged. The extension to handling numerous classes and instances could pose computational challenges. The computational cost of utilizing individual GMMs for each class may become prohibitive in extensive datasets. Future research directions should aim to address these limitations and further refine the AGP framework for broader and more robust applicability in the field of few-shot learning. Nonetheless, we aim to present AGPs as a novel approach to propel a more compositionally informed method of learning with minimal data.

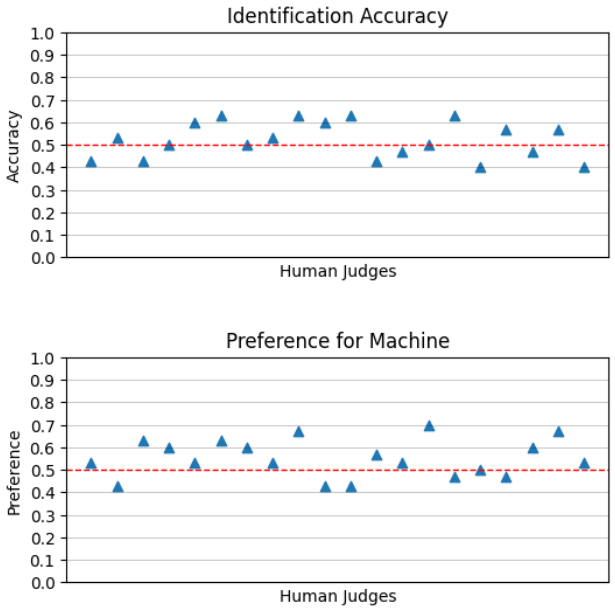


*Figure 4.* Each marker reveals the evaluation scores of a human judge, averaged across all 10 sets of comparisons. The ideal performance of 50 percent is indicated by the dashed red line.

## References

Kadam, S. and Vaidya, V. Review and analysis of zero, one and few shot learning approaches. In *Intelligent Systems Design and Applications: 18th International Conference on Intelligent Systems Design and Applications (ISDA 2018) held in Vellore, India, December 6-8, 2018, Volume 1*, pp. 100–112. Springer, 2020.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Kingma, D. P., Welling, M., et al. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019.

Lake, B. M., Salakhutdinov, R. R., and Tenenbaum, J. One-shot learning by inverting a compositional causal process. *Advances in neural information processing systems*, 26, 2013.

Lake, B. M., Salakhutdinov, R., and Tenenbaum, J. B. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

Lake, B. M., Salakhutdinov, R., and Tenenbaum, J. B. The omniglot challenge: a 3-year progress report. *Current Opinion in Behavioral Sciences*, 29:97–104, 2019.

Lee, T.-C., Kashyap, R. L., and Chu, C.-N. Building skeleton models via 3-d medial surface axis thinning algo-

rithms. *CVGIP: Graphical Models and Image Processing*, 56(6):462–478, 1994.

Liang, C., Wang, W., Miao, J., and Yang, Y. Gmmseg: Gaussian mixture based generative semantic segmentation models. *Advances in Neural Information Processing Systems*, 35:31360–31375, 2022.

McLachlan, G. J. and Basford, K. E. *Mixture models: Inference and applications to clustering*, volume 38. M. Dekker New York, 1988.

Reynolds, D. A. et al. Gaussian mixture models. *Encyclopedia of biometrics*, 741(659-663), 2009.

Wang, Y., Yao, Q., Kwok, J. T., and Ni, L. M. Generalizing from a few examples: A survey on few-shot learning. *ACM computing surveys (csur)*, 53(3):1–34, 2020.

Yu, D., Deng, L., Yu, D., and Deng, L. Gaussian mixture models. *Automatic Speech Recognition: A Deep Learning Approach*, pp. 13–21, 2015.

Zhang, T. A fast parallel algorithm for thinning digital patterns. *Commun. ACM*, 27(3):337–343, 1997.