Multiple Hypothesis Dropout: Estimating the Parameters of Multi-Modal Output Distributions

David D. Nguyen[1,2,3], **David Liebowitz**[1,4], **Salil S. Kanhere**[1,3], **Surya Nepal**[2,3]
[1]UNSW Sydney
[2]CSIRO Data61
[3]Cybersecurity CRC
[4]Penten
david.nguyen@data61.csiro.au, david.liebowitz@penten.com, salil.kanhere@unsw.edu.au, surya.nepal@data61.csiro.au,

Abstract

In many real-world applications, from robotics to pedestrian trajectory prediction, there is a need to predict multiple real-valued outputs to represent several potential scenarios. Current deep learning techniques to address multiple-output problems are based on two main methodologies: (1) mixture density networks, which suffer from poor stability at high dimensions, or (2) multiple choice learning (MCL), an approach that uses M single-output functions, each only producing a point estimate hypothesis. This paper presents a Mixture of Multiple-Output functions (MoM) approach using a novel variant of dropout, Multiple Hypothesis Dropout. Unlike traditional MCL-based approaches, each multiple-output function not only estimates the mean but also the variance for its hypothesis. This is achieved through a novel stochastic winner-take-all loss which allows each multiple-output function to estimate variance through the spread of its subnetwork predictions. Experiments on supervised learning problems illustrate that our approach outperforms existing solutions for reconstructing multimodal output distributions. Additional studies on unsupervised learning problems show that estimating the parameters of latent posterior distributions within a discrete autoencoder significantly improves codebook efficiency, sample quality, precision and recall.

Introduction

Multiple-output prediction is the task of generating a variety of real-valued outputs given the same input, a powerful tool in scenarios that require a comprehensive understanding of multiple potential possibilities. This approach has demonstrated an extensive ability to capture diversity, creativity and uncertainty across many domains including generative modelling (Graves 2013; Ha and Eck 2017; Nguyen, Nepal, and Kanhere 2021), computer vision (Guzman-Rivera et al. 2014; Firman et al. 2018), pedestrian trajectory prediction (Makansi et al. 2019), spectral analysis (Bishop 1994), robotic movement (Zhou, Gao, and Asfour 2020) and reinforcement learning (Ha and Schmidhuber 2018).

Multiple-output prediction is a generalization of traditional single-output prediction. In the supervised learning context, a single-output function is given a dataset of input-output pairs $\{(\mathbf{x}_n, \mathbf{y}_n) \mid n \in \{1, \dots, N\}, \mathbf{x}_n \in \mathcal{X}, \mathbf{y}_n \in \mathcal{Y}\}$ (Guzman-Rivera, Batra, and Kohli 2012). The objective of the **single-output function** $f^{\boldsymbol{\omega}} : \mathcal{X} \mapsto \mathcal{Y}$ is to learn a mapping from a single input in input-space to single output in output-space using a set of parameters $\boldsymbol{\omega}$. It seeks to minimizes a loss function $\mathcal{L} : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}^+$ that computes the distance between its predictions $\hat{\mathbf{y}}_n$ and targets $\mathbf{y}_n$.

A **multiple-output function** $\mathcal{F} : \mathcal{X} \mapsto \mathcal{Y}^M$ can learn a mapping from input space to an M-tuple of outputs $\hat{\mathbf{Y}}_n = \{\hat{\mathbf{y}}_n^1, \dots, \hat{\mathbf{y}}_n^M | \hat{\mathbf{y}}_n^M \in \mathcal{Y}\}$. Multiple-output functions based on neural networks (NNs) can be categorized into two main types: those that produce multiple point estimates and those that produce multi-modal distributions. An effective class of algorithms that fall into the first group is based on multiple-choice learning (MCL) (Guzman-Rivera, Batra, and Kohli 2012; Guzman-Rivera et al. 2014; Lee et al. 2016, 2017; Firman et al. 2018). This class of algorithms construct an ensemble of M single-output functions that each produce single point estimates and are trained using a winner-take-all (WTA) loss function. MCL techniques are well-known for being stable and have been demonstrated on high-dimensional datasets such as images (Guzman-Rivera et al. 2014; Firman et al. 2018). However, a common criticism of MCL algorithms is that it requires the practitioner to arbitrarily choose the hyper-parameter M, which also assumes an equal number of predictors and targets for any given input. These models are also challenging to scale as increasing the number of single-output functions in the ensemble can be expensive from a computational and parameterization perspective.

The second approach to multiple-output prediction represents different modes of $\mathbf{y}_n$ with the same input value $\mathbf{x}_n$ as separate distributions. This is advantageous over point-estimate approaches because it provides an estimation of uncertainty through the covariance. Mixture density networks (MDN) (Bishop 1994) use the outputs of a neural network (NN) to predict the parameters of a mixture of Gaussians. A part of the outputs is used to predict mixture coefficients, while the remainder is used to parameterize each individual mixture component. MDNs at higher dimensions, however, are difficult to optimize with stochastic gradient descent due to numerical instability and mode collapse and require special initialization schemes (Rupprecht et al. 2017; Cui et al. 2019; Makansi et al. 2019; Zhou, Gao, and Asfour 2020; Prokudin, Gehler, and Nowozin 2018).

In this paper, we present a novel Mixture of Multiple-Output functions (MoM) that can learn multi-modal output distributions by combining the stability of MCL algorithms with the scalability of MDN parameter estimation. The parameters (modes and variances) are estimated using a mixture of NNs trained with a new Stochastic WTA loss function and a variant of dropout called **Multiple Hypothesis (MH) dropout**.

Dropout (known as binary dropout), introduced in (Hinton et al. 2012), is a training technique in which units of a NN are randomly "dropped" or set to zero during training, with dropout probability as a hyperparameter. This leads to the formation of *thinned subnetworks* during training, where each thinned subnetwork consists of a set of *undropped* weights derived from a base NN. We can think of the subnetworks that appear during training as elements of ensembles that share parameters.

During inference, binary dropout networks obtain a *single prediction* by turning off the dropout operation, scaling the weights by dropout probability and using the entire network. This has been demonstrated to be equivalent to taking the geometric average of all subnetwork predictions in the ensemble (Baldi and Sadowski 2013). Further studies have demonstrated that prediction uncertainty can be modelled by computing the variance of predictions from thinned subnetworks during inference (Gal 2016; Amini et al. 2018), more commonly known as **Monte Carlo (MC) dropout**.

The work of (Ilg et al. 2018) discussed utilizing MC dropout for an ensemble of NNs to estimate the uncertainty of each prediction; however, we show later in the paper that this technique does not generalize to multi-output settings. Our MH Dropout extends the abilities of MC dropout to multiple-output prediction scenarios by producing accurate *variance estimates* using a novel loss function. The approach is aligned with the functionalities of MDNs while sidestepping their inherent numerical instability issues due to the use of a stable loss function.

Building upon this idea, we design the latent posterior distribution of a vector-quantization variational autoencoder (VQVAE) (Hinton and Zemel 1993) as a multi-modal Gaussian distribution by estimating its parameters using MH Dropout. The conventional unsupervised VQVAE employs a codebook of latent embeddings to learn rich representations that are used to reconstruct the input distribution accurately. This technique underpins many recent state-of-the-art image synthesis models (Rombach et al. 2022; Esser, Rombach, and Ommer 2021; Ramesh et al. 2021).

However, a weakness of the VQVAE is that the latent embeddings can only represent the *modes* of clusters in representational space. As far as we know, this framework cannot represent the spread or variance of each cluster. Due to this drawback, a common strategy is to scale the model's latent representational capacity by *saturating the continuous posterior*. This is achieved through either (1) increasing the codebook size or (2) the number of tokens per input. This scaling strategy has led to works that compromise on computational resources to learn longer sequences of tokens and lead to slower sampling during generation time. Instead, we propose an extension called **MH-MQVAE**, that learns both the modes and variances of the latent posterior distribution using MH Dropout networks, resulting in improved codebook efficiency and representational capacity. Through extensive experiments, we demonstrate that this improves generation quality, precision and recall across various datasets and existing VQ architectures.

To summarize, our contributions are as follows:

1. We introduce the Multiple Hypothesis Dropout, a novel variant of dropout that converts a single-output function into a multi-output function using the subnetworks derived from a base neural network.
2. We found that combining Winner-Takes-All loss with stochastic hypothesis sampling allows MH Dropout networks to stably learn the statistical variability of targets in multi-output scenarios.
3. We describe a Mixture of Multiple-Output Functions (MoM), composed of MH Dropout networks to address multi-modal output distributions in supervised learning settings. We show this architecture can learn the parameters of the components of a Gaussian mixture.
4. We propose a novel MH-VQVAE that employs MH Dropout networks to estimate the variance of clusters in embedding representational space. We show this approach significantly improves codebook efficiency and generation quality.

Preliminaries

Multiple-Output Prediction

The idea of generating diverse possible hypotheses for a downstream expert was presented in Multiple Choice Learning by (Guzman-Rivera, Batra, and Kohli 2012). In a supervised learning setting, a multiple-output function is a mapping for single input $\mathbf{x}_n$ from input space $\mathcal{X}$ to an M-tuple of outputs $\hat{\mathbf{Y}}_n = \{\hat{\mathbf{y}}_n^1, \ldots, \hat{\mathbf{y}}_n^M | \hat{\mathbf{y}}_n^M \in \mathcal{Y}\}$: $\mathcal{F} : \mathcal{X} \mapsto \mathcal{Y}^M$, where $\mathcal{F}$ is composed of M single-output functions $\mathcal{F} = \{f^m\}_{m=1}^M$ parameterized by M separate sets of weights $\{\omega^m\}_{m=1}^M$. The set of functions (or predictors) produces a set of M hypotheses given an input sample $\mathbf{x}_n$:

$$\mathcal{F}(\mathbf{x}_n) = \{f^1(\mathbf{x}_n), \ldots, f^M(\mathbf{x}_n)\} = \{\hat{\mathbf{y}}_n^1, \ldots, \hat{\mathbf{y}}_n^M\} \quad (1)$$

The ensemble is trained using a multiple hypotheses prediction algorithm and (vanilla) **winner-takes-all (WTA) loss**. During each iteration, only the "winning" function from the ensemble is chosen for back-propagation. The "winning" function is the one with its hypothesis closest to the target according to some distance function, such as L_2-norm. Gradients for the other predictors are eliminated by multiplying their respective losses by zero. The vanilla WTA loss function is summarized as:

$$\mathcal{L}_{wta}(\mathbf{x}_n, \mathbf{y}_n) = \sum_{m=1}^{M} w_m ||\mathbf{y}_n - f^m(\mathbf{x}_n)||_2^2 \quad (2)$$

$$\text{where } w_j = \begin{cases} 1 & \text{if } j = \arg\min_m ||\mathbf{y}_n - f^m(\mathbf{x}_n)||_2^2; \\ 0 & \text{otherwise.} \end{cases} \quad (3)$$

An issue with this loss function is that poorly initialized predictors can generate hypotheses in regions far from the outputs, which can be ignored throughout training. Determining which predictor has been trained or ignored at inference time can also be challenging.

To address this, the work of (Nguyen, Nepal, and Kanhere 2021) employed a mixture coefficient layer g to learn a probability distribution over the number of functions in the ensemble:

$$\boldsymbol{\phi} = \textit{softmax}(g(\mathbf{x}_n)) \tag{4}$$

where $\boldsymbol{\phi}$ is a vector of length M that sums to one. During training, the coefficients of "winning" predictors are maximized using a modified loss function:

$$\mathcal{L}(\mathbf{x}_n, \mathbf{y}_n) = \sum_{m=1}^{M} -log(\phi_m) w_m ||\mathbf{y}_n - f^m(\mathbf{x}_n)||_2^2 \tag{5}$$

During inference, functions are sampled according to $m \sim \mathcal{M}(\boldsymbol{\phi})$, a multinomial distribution parameterized by $\boldsymbol{\phi}$. This paper builds upon these ideas and introduces multi-output function-based subnetworks generated using the dropout mechanism. We will briefly review this concept next.

Dropout Networks

Here we describe dropout (Hinton et al. 2012) applied to a neural network f with L layers. Let $l \in [L]$ denote the index of the layers. Each of layer is parameterized by a weight matrix $\mathbf{W}^{(l)}$ and bias vector $\mathbf{b}^{(l)}$. Let $\mathbf{y}^{(l)}$ be the output vector of layer l, where $\mathbf{y}^{(L)} = \hat{\mathbf{y}}$. The output of layer l of a dropout network can be expressed as:

$$\mathbf{y}^{(l)} = \sigma(\mathbf{W}^{(l)}(\boldsymbol{\psi}^{(l)} \odot \mathbf{y}^{(l-1)}) + \mathbf{b}^{(l)}) \text{ with } \mathbf{y}^{(0)} = \mathbf{x} \tag{6}$$

where σ is an element-wise non-linear activation function such as sigmoid, $\sigma(x) = 1/(1+e^{-x})$, $\boldsymbol{\psi}^{(l)}$ is the dropout vector for layer l and $\odot$ is the Hadamard product (element-wise multiplication). The dropout vector $\boldsymbol{\psi}^{(l)}$ is a vector of the same shape as $\mathbf{y}^{(l-1)}$ where each element is a 0-1 Bernoulli gating variable sampled according to $P(\psi_i^{(l)} = 1) = p_i^{(l)}$. Here $\psi_i^{(l)}$ is the i^{th} element of dropout vector $\boldsymbol{\psi}^{(l)}$.

We can view a dropout network as representing an implicit ensemble of 2^D possible subnetworks, where D represents the total weights. This ensemble can be generated over all possible realizations of Bernoulli gating variables. In the following section, we leverage subnetworks to build a multiple-output function.

Multiple Hypothesis Dropout

Multiple Hypothesis Dropout (MH Dropout) can be thought of as converting a single-output neural network into an *accurate* multiple-output function (MH Dropout Network) trained with a stochastic WTA loss function. In the following sections, we describe this algorithm, its loss function and analyse its ability to capture variance using a toy dataset.

MH Dropout Networks

Let $\mathcal{F}^{\boldsymbol{\omega}}$ be a neural network parameterized by $\boldsymbol{\omega}$ with D total weights and suppose that it is trained with MH Dropout. The *set of all $M = 2^D$ possible subnetworks* of this base network can be generated over all M possible realizations of Bernoulli gating variables. Thus, a MH Dropout Network and its subnetworks is: $\mathcal{F}^{\boldsymbol{\omega}} = \{f^{\boldsymbol{\omega},m}\}_{m=1}^{M}$

where all subnetworks are parameterized using a shared set of weights $\boldsymbol{\omega}$. Here $f^{\boldsymbol{\omega},m}$ is the m^{th} subnetwork of the base network and the index m represents a specific realization of the Bernoulli gating variables.

Given an input vector $\mathbf{x}_n$, the set of output vectors (or hypotheses) of all subnetworks can be expressed as:

$$\mathcal{F}(\mathbf{x}_n; \boldsymbol{\omega}) = \{f^1(\mathbf{x}_n; \boldsymbol{\omega}), ..., f^M(\mathbf{x}_n; \boldsymbol{\omega})\} = \{\hat{\mathbf{y}}_n^1, \ldots, \hat{\mathbf{y}}_n^M\} \tag{7}$$

where $\hat{\mathbf{y}}_n^m$ denotes the hypothesis of the m^{th} subnetwork.

Variance Estimates. During inference, we can compute variance estimates using the set of subnetworks with a similar methodology to the work by (Gal 2016). First, we define the expected outputs across all subnetworks $\mathbb{E}[\mathcal{F}]$ and then compute the predictive variance $Var[\mathcal{F}]$.

To begin, we take T stochastic samples of Bernoulli gating variables, which gives T subnetworks $\{f^{\boldsymbol{\omega},1}, \ldots, f^{\boldsymbol{\omega},T}\}$, where $T < M$. The expected output can be estimated by taking the average of these subnetwork outputs given the same input: $\mathbb{E}[\mathcal{F}(\mathbf{x}_n; \boldsymbol{\omega})] \approx \frac{1}{T}\sum_{t=1}^{T} f^{\boldsymbol{\omega},t}(\mathbf{x}_n; \boldsymbol{\omega})$,where $t \in \{1, \ldots, T\}$. Thus, the predictive variance is:

$$Var[\mathcal{F}(\mathbf{x}_n; \boldsymbol{\omega})] = \mathbb{E}[(\mathcal{F}(\mathbf{x}_n; \boldsymbol{\omega}) - \mathbb{E}[\mathcal{F}(\mathbf{x}_n; \boldsymbol{\omega})])^2] \tag{8}$$

Loss Function. Vanilla WTA loss requires the hypotheses of *all predictors in the ensemble* to be compared against each target during each training iteration. With MH Dropout, however, it can be impractical to compute all possible subnetwork hypotheses when the number of weights is large. Our proposal is to only consider a random subset of all subnetwork hypotheses during winner-takes-all training, referred to as **Stochastic Winner-Take-All (SWTA)**:

$$\mathcal{L}_{swta}(\mathbf{x}_n, \mathbf{y}_n) = \sum_{t=1}^{T} w_t ||\mathbf{y}_n - f^t(\mathbf{x}_n; \boldsymbol{\omega})||_2^2 \tag{9}$$

This equation is nearly identical to vanilla WTA (Eq. 2), with the main modification being the replacement of M (number of total predictors) with T (random subset size). The critical difference between Stochastic WTA and its predecessor, vanilla WTA, is that it requires predictors to *share parameters* which allow poor subnetwork predictions in far regions from targets to move even when they are not the "winning" subnetwork. This is visualized in (Nguyen et al. 2023).

Estimating the Variance of Multiple Points

To quantitatively assess a MH Dropout network's ability to capture the variance of multiple targets, we describe a toy multi-point dataset and metric. Our toy multi-point dataset contains N different outputs given the same input:

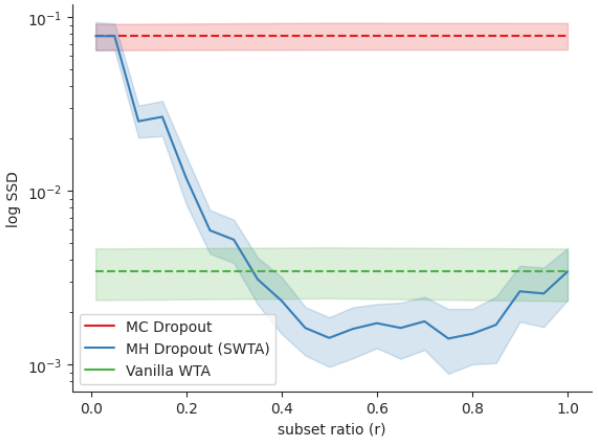

Figure 1: SSD vs subset ratio curves for FFNs trained with 3 techniques. The line represents the average SDD value, while the shadow represents the 95% confidence interval.

$\{(\mathbf{x}_1, \mathbf{y}_n) \, | \mathbf{x}_1 \in \mathcal{X}, \mathbf{y}_n \in \mathcal{Y}\}$ as proposed by (Makansi et al. 2019). The inputs are sampled from $\mathcal{N}(0, 1)$ and outputs from a uniform distribution in the range [0,1].

We define a simple metric, Standard Deviation Distance (SDD), that measures the distance between the standard deviation of two sets of vectors: predicted hypotheses $\hat{\mathbf{Y}}_k$ and targets $\mathbf{Y}_k$: $SDD = \frac{1}{K} \sum_{k=1}^{K} ||sd(\hat{\mathbf{Y}}_k) - sd(\mathbf{Y}_k)||_2$ where sd represent the standard deviation operation and k represents the experiment trial.

For this experiment, we employ a three-layer feed-forward network (FFN) with four hidden units. Multiple copies of the network are initialized with the same weights and trained with a different MH dropout subset size T to represent different levels of stochasticity. Let the **subset ratio** r be the subset size T over the total subnetwork size 2^D. Thus as the subset ratio approaches zero, it begins to approximate binary dropout, and as it approaches one, it approximates Vanilla WTA. We use two baselines with the same network to demonstrate this: one with MC dropout during inference and another trained with vanilla WTA. This experiment is repeated across $K = 30$ trials and results are presented in Fig. 1.

In Fig. 1, notice that models with MC dropout and Stochastic WTA at a lower subset ratio under 0.4 result in higher SSD values. This implies that these models are increasingly unable to capture the spread of targets. Stochastic WTA with subset ratios between 0.5 and 0.7 provide a lower SDD value; however, the SSD increases as the ratio rises above 0.8.

These results suggests that the model has an optimal range for noise to learn the spread of a target distribution, making it a valuable tool for a range of problems. This finding aligns with the work of Gal et al. (Gal 2016), which found that the *predictive variance* of all subnetwork predictions corresponds to the statistical variability in the targets. The difference is that while MC dropout can only provide variance estimates for single-output problems, MH Dropout generalizes this ability to situations with *multiple outputs*.

Mixture of Multiple-Output Functions

Model

In this section, we propose a model that combines multiple MH Dropout Networks to learn the mean and variance of multi-modal output distributions within a supervised learning setting. This model is structured in two main hierarchies described below. Here, the superscript $\boldsymbol{\omega}$ is omitted for ease of reading.

Primary Hierarchy. At the top level, the primary hierarchy consists of MH Dropout Networks represented as $\boldsymbol{\mathcal{F}} = \{\mathcal{F}^m\}_{m=1}^{M}$, with separate sets of weights for each network $\{\boldsymbol{\omega}\}_{m=1}^{M}$. Additionally, a mixture coefficient layer is employed, learning a discrete distribution $\boldsymbol{\phi}$ over the M networks.

Within each network there is an encoder function $q^m : \mathcal{X} \mapsto \mathcal{Z}$ which transforms the input to latent vector $\mathbf{z}$, subsequently dividing it into two equal sized vectors $(\mathbf{e}, \mathbf{e}')$ The latter vector is passed to the secondary hierarchy.

Secondary Hierarchy. The secondary hierarchy comprises of subnetworks derived from each of the primary hierarchy's MH Dropout Networks. For each network $\mathcal{F}^m$, a random subset of T subnetworks is:

$$\mathcal{F}^m = \{f^{m,1}, \ldots, f^{m,T}\} \text{ with } \mathcal{F}^m \in \boldsymbol{\mathcal{F}} \tag{10}$$

During training, the vector $\mathbf{e}'$ is passed to one of these randomly selected subnetwork and combined with $\mathbf{e}$ to form a hypothesis:

$$\hat{\mathbf{y}}_n^{m,t} = \mathbf{e} + f^{m,t}(\mathbf{e}') \tag{11}$$

Loss Function. The supervised learning training objective combines Stochastic WTA and a modified version of Eq. 5 into a single loss function. An issue with Eq. 5 is that the log-coefficient error can dominate the distance error resulting in sub-optimal behaviour. To address this, we simply scale the log-coefficient error and sum the terms:

$$\mathcal{L}(\mathbf{x}_n, \mathbf{y}_n) = -\sum_{m=1}^{M} \sum_{t=1}^{T} w_m v_t (\log(\phi_m)\lambda - ||\mathbf{y}_n - \hat{\mathbf{y}}_n^{m,t}||_2^2) \tag{12}$$

$$\text{where } w_j = \begin{cases} 1 & \text{if } j = \arg\min_m ||\mathbf{y}_n - \hat{\mathbf{y}}_n^{m,t}||_2^2 \\ 0 & \text{otherwise.} \end{cases} \tag{13}$$

$$\text{and } v_j = \begin{cases} 1 & \text{if } j = \arg\min_t ||\mathbf{y}_n - \hat{\mathbf{y}}_n^{m,t}||_2^2; \\ 0 & \text{otherwise.} \end{cases} \tag{14}$$

where λ is the hyper-parameter used to scale the log-coefficient term. The two gating variables, w and v, only allow the gradients of the "winning" primary hierarchy and its respective "winning" subnetwork to pass to the optimizer during back-propagation.

Inference. During inference, the model generates predictions by sampling from a parameterized multimodal Gaussian distribution. This sampling process uses the means from the M encoders and sample variance of the outputs from the secondary hierarchy of T random subnetworks:

$$\hat{\mathbf{y}}_n^m \sim \mathcal{N}(\mathbf{e}; Var[\mathcal{F}^m(\mathbf{e}')]) \tag{15}$$

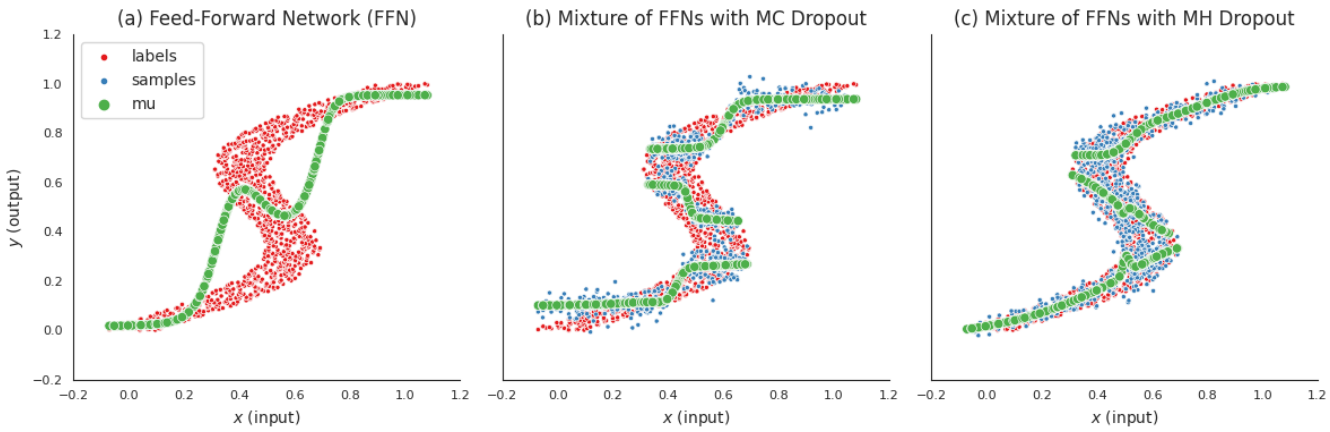

Figure 2: Reconstructions by three models trained on the inverse sine wave, a classic multi-modal output dataset introduced by (Bishop 1994). Our proposed mixture of MH Dropout networks (MoM) accurately learns the center and spread of the sine wave.

The Inverse Sine Wave Problem.

We demonstrate our proposed MoM's ability to estimate the mean and variance of multi-modal distributions using the classical inverse sine wave problem. This problem was initially proposed by Bishop in his work on mixture density models (Bishop 1994) (See Section 5). The inverse sine wave function can be defined as: $x = y + 0.3 \sin 2\pi y + \epsilon$, where x is an *output* variable, y is the *input* variable, and ϵ is a random variable drawn from a uniform distribution over the range $(-0.1, 0.1)$. To create a dataset for this problem, we generate 1000 evenly spaced inputs y in the range $(0, 1)$. Each y is passed through the function to output x.

Here we compare three models: (a) Feed-forward network (FFN) (b) Mixture of FFNs trained with MC Dropout and (c) Mixture of FFNs with MH Dropout (MoM). Each FFN contains 6 hidden units, 6 layers, tanh activation functions. The mixture models employ three FFNs trained with a dropout rate of 0.5 and employ mixture coefficient layers. The mixture model of MH Dropout networks is trained using a subset ratio of 0.1.

As seen in Fig. 2, the feed-forward network learns an average of the labels (green line) because it is a single-output function. The mixture model with MC dropout is unable to provide a good fit either, struggling to learn the correct mean or variance using the three components. It is clear that the MoM model fits the tri-modal structure of the sine wave. It accurately predicts the center (green) and the spread of the sine wave, reconstructed with the samples (blue) drawn from a parameterized Gaussian (Eq. 15).

Application to Generative Models

In this section, we turn to unsupervised learning, specifically the autoencoder framework where the goal of the function $f^{\boldsymbol{\omega}} : \mathcal{X} \mapsto \mathcal{X}$ is to produce a reconstruction $\hat{\mathbf{x}}$ of the input $\mathbf{x}$, such that it minimizes a loss function, $\mathcal{L}(\mathbf{x}, \hat{\mathbf{x}})$. This can be thought of as a communication problem, where a sender wishes to communicate a dataset $\mathcal{D} = (\mathbf{x}_n\}_{n=1}^N$ to a receiver as efficiently as possible. Among various approaches to this problem, one particularly effective method is to map each input to a latent posterior distribution and then sample from this distribution during inference, introduced as Variational Autoencoders (VAEs) (Kingma and Welling 2013).

The vector quantization variational autoencoder (VQ-VAE) implements the continuous posterior distribution as a mixture of embeddings from a codebook. We can consider each embedding as centers of clusters in representational space. Below, we propose an extension to VQVAE by learning both the centers and variances of these clusters using separate codebooks and MH Dropout. This approach promises to scale representational capacity more efficiently by simply learning the parameters that represent a full posterior distribution, instead of every point in the distribution.

Background: Vector quantization The VQ framework is compromised of three key components: an encoder q, decoder p and a latent codebook containing a set of K embeddings $\mathbf{e}_k \in \mathbb{R}^d$ where $k \in \{1, \ldots, K\}$.

The **encoder** is a function $q : \mathcal{X} \mapsto \mathcal{Y} \subseteq \mathbb{R}^d$ that maps each input $\mathbf{x}_n \in \mathcal{X}$ from pixel space to an encoded latent vector $\mathbf{y}_n \in \mathcal{Y}$ in latent space such that $q(\mathbf{x}_n) = \mathbf{y}_n$.

The **codebook** replaces the encoded vector with the indices of the nearest embeddings, typically based on L_2 norm: $z_n = \arg\min_k ||\mathbf{y}_n - \mathbf{e}_k||_2$ where z_n is the specific index (or token) of the codebook. A lookup operation is performed against the codebook to obtain an embedding $\mathbf{e}_* = \hat{\mathbf{y}}_n \in \mathcal{Y}$. The symbol $*$ denotes the specific index chosen in the codebook according to the token z_n.

The **decoder** is a function $p : \mathcal{Y} \mapsto \mathcal{X} \subseteq \mathbb{R}^D$ that maps the embeddings from latent space to a reconstruction of the input $\hat{\mathbf{x}}_n \in \mathcal{X}$ in pixel space such that $p(\hat{\mathbf{y}}_n) = \hat{\mathbf{x}}_n$.

The overall loss function is the sum of the reconstruction and commitment loss. The **reconstruction loss** trains the decoder to minimize the error between each input and the reconstruction: $\mathcal{L}_{rec}(\mathbf{x}_n, \hat{\mathbf{x}}_n) = \|\mathbf{x}_n - \hat{\mathbf{x}}_n\|_2^2$. This is possible by back-propagating through the codebook using the

straight-through gradient estimator (Bengio, Léonard, and Courville 2013) which passes the gradients directly from the decoder to the encoder with the following codebook loss:

$$\mathcal{L}_{cb}(\mathbf{y}_n, \hat{\mathbf{y}}_n) = \|sg[\mathbf{y}_n] - \hat{\mathbf{y}}_n\|_2^2 + \beta\|\mathbf{y}_n - sg[\hat{\mathbf{y}}_n]\|_2^2 \quad (16)$$

where sg refers to the stop-gradient operation. This is commonly referred to as the "commitment loss" proposed in (Van Den Oord, Vinyals et al. 2017).

In a secondary training stage, a probabilistic model learns a distribution over the observed tokens $\mathbf{z}$, more formally known as the **categorical posterior distribution**, using cross-entropy loss. During generation time, the tokens are sampled from the categorical posterior distribution to reconstruct a **continuous posterior distribution** using codebook embeddings.

Multiple Hypothesis VQVAE (MH-VQVAE)

Our proposed extension to VQVAE learns the variances of the multi-modal posterior distribution using MH Dropout. In contrast to the supervised learning setting, MH Dropout is applied in the latent representational space. To learn the variance, we also introduce a secondary branch composed of an encoder q' and a secondary latent codebook of K embeddings $\mathbf{e}'_k \in \mathbb{R}^d$ where $k = 1 \dots K$. This new structure can be thought of as a hierarchical VQVAE, however, in contrast to previous works, the secondary branch learns the variance of the latent mixture components.

The **secondary encoder** is a function $q' : \mathcal{X} \mapsto \mathcal{Y}' \subseteq \mathbb{R}^d$ that maps each input $\mathbf{x}_n \in \mathcal{X}$ from pixel space to a secondary encoded vector $\mathbf{y}'_n \in \mathcal{Y}'$ in secondary latent space such that $q'(\mathbf{x}_n) = \mathbf{y}'_n$.

The **secondary codebook** replaces the secondary encoded vector $\mathbf{y}'_n$ with the indices of the nearest embeddings, $z'_n = \arg\min_k ||\mathbf{y}'_i - \mathbf{e}'_k||_2$ where z'_n is the secondary codebook token. Similarly, a lookup operation is performed against the codebook to obtain an embedding $\mathbf{e}'_* \in \mathcal{Y}'$ that corresponds to the secondary codebook token.

Here, the **MH Dropout network** is a multiple-output function $\mathcal{F}^{\boldsymbol{\omega}} : \mathcal{Y}' \mapsto \mathcal{Y}^T$ that maps the secondary embedding $\mathbf{e}'_*$ to a T-tuple of hypotheses in primary latent space $\hat{\mathbf{Y}}_n = \{\hat{\mathbf{y}}^1_n, \dots, \hat{\mathbf{y}}^T_n | \hat{\mathbf{y}}^t_n \in \mathcal{Y}\}$ where:

$$\hat{\mathbf{y}}^t_n = \mathbf{e}_* + f^t(\mathbf{e}'_*) \quad (17)$$

Here $\mathcal{F}^{\boldsymbol{\omega}}$ is composed of T randomly sampled subnetworks $\{f^t\}_{t=1}^T$. Notice the similarity to Eq. 11.

During training, the MH Dropout network employs the Stochastic WTA loss function, thus only the hypothesis $\hat{\mathbf{y}}^t_n$ nearest to the primary encoded vector $\mathbf{y}_n$ is passed to the decoder. During inference, the model samples from a parameterized multi-modal Gaussian distribution using the primary embedding (as the mean) and the variance of the hypotheses (as the variance): $\hat{\mathbf{y}}_n \sim \mathcal{N}(e_*; Var[\mathcal{F}^{\boldsymbol{\omega}}(\mathbf{e}'_*)])$. In the following section, we conduct an extensive range of experiments that assess the performance of our proposed model.

Experiments

Here we focus on understanding the impact of adopting our approach by integrating it into two well-known VQ architectures: VQVAE-2 and VQGAN. We first discuss our experimental setup and then conduct a range of experiments.

Setup

Model Comparisons. We compare our method with hierarchical VQ as proposed in (Razavi, Van den Oord, and Vinyals 2019). **MH-VQVAE** and **MH-VQGAN** directly replaces the top-bottom hierarchy of *VQVAE-2* and *VQGAN* respectively. However all models utilize the same encoder and decoder as VQVAE-2 due to GPU memory constraints. The models differ in their use of PixelCNN and Transformer models for categorical posterior modeling, as proposed in previous work (Rombach et al. 2021).

Hyper-parameters. We study the effects of MH Dropouton performance by varying two hyper-parameters: the number of codebook entries (total of K split between primary and secondary codebooks) and tokens per image ($S + S'$). The number of hypotheses per pass was 64. Following existing practices (Esser, Rombach, and Ommer 2021), high down-sampling factors $F = (14, 16, 32)$ and compression rates above 38.2 bits per dimension are applied.

Datasets. Experiments utilize medium resolution image datasets: FashionMNIST 28×28, CelebA 64×64, and ImageNet 64×64 (Xiao, Rasul, and Vollgraf 2017; Liu et al. 2018; Deng et al. 2009). Token numbers vary by dataset, with primary tokens in the range 4–16 and a single secondary token per image.

Metrics. Sample quality is assessed using Fréchet Inception Distance (FID) (Heusel et al. 2017), where lower values indicate better similarity between real and generated samples. Two F-score numbers (Sajjadi et al. 2018) are also reported to quantify model precision ($F_{1/8}$) and recall (F_8), with higher values denoting better performance. Further metrics, such as MSE and perceptual loss (LPIPS) are reported in (Nguyen et al. 2023).

Results

Precision and Recall. We provide an empirical analysis demonstrating the improvement in the representational capacity of existing VQ models when utilizing our proposed MH-VQ framework instead. Our findings emphasizes the robustness of our MH-VQ framework at higher compression rates which can be attributable to its ability to learn a richer posterior distribution by utilizing variance estimates.

To demonstrate this, we conduct an experiment involving multiple instances of each model trained on each dataset. The models were subjected to the same hyper-parameters except for the total codebook entries, which measure the model's overall representational capacity.

First we compare the performance of VQVAE2 and MH-VQVAE on the FashionMNIST dataset using precision and recall. We train multiple copies of each model using different total codebook entries and plot their precision and recall on Figure 3. The scatter plot shows that the MH-VQVAE framework (cross) significantly improves precision and recall compared to similar-sized VQVAE2 (circle). Our findings show that MH-VQVAE can outperform VQVAE2 across both metrics with considerably fewer codes (32 vs 256).

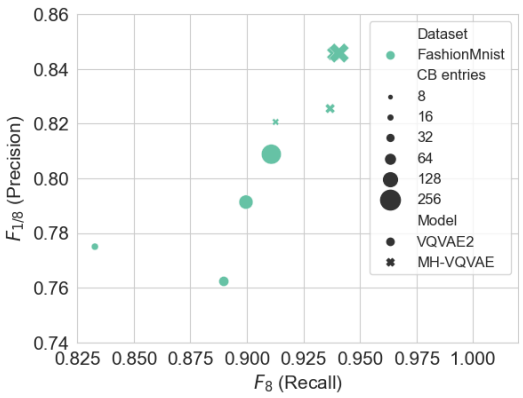

Figure 3: Recall and precision for multiple VQVAE2 (circle) and MH-VQVAE (cross) on the FashionMNIST dataset. This shows MH-VQVAE improves both scores and can outperform VQVAE2 with $1/4$ of the codebook size.

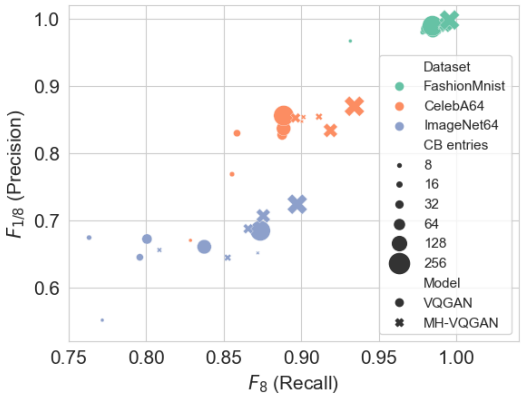

Figure 4: Recall and precision for VQGAN (circle) and MH-VQGAN (cross). These scores show that MH-VQ improves recall and precision across all datasets, reflecting its ability to learn a richer posterior distribution.

The same comparisons were applied to VQGAN and MH-VQGAN using all proposed datasets as shown in Fig. 4. These results also show that MH-VQ improves precision and recall across all VQ models. We notice that recall improves slightly more than precision, possibly connected to the observation that GANs already induce higher precision, as seen in (Sajjadi et al. 2018).

Sample Quality In Fig. 5, we report the FID scores, shown as bar charts, which provide a direct comparison between pairs of VQ and MH-VQ models. The steepness of the FID scores for both VQVAE-2 (dark red) and VQGAN (blue) signifies the standard VQ framework's dependence on increasing codebook entries to achieve good reconstruction quality. Conversely, the MH-VQ models demonstrate a more gradual decrease in FID scores relative to codebook size, attaining optimal performance with fewer entries.

Our findings show that the adoption of MH-VQ leads to a considerable reduction in codebook entries—up to 4 times smaller with VQVAE-2, 8 times smaller with VQGAN on the FashionMnist dataset, and an impressive 16 times on the CelebA64 dataset, comparing MH-VQGAN with a codebook size of 16 to VQGAN at 256. The improvements recorded over the Imagenet64 dataset were relatively lower at a 4-fold reduction, an outcome we attribute to the extended sequence length of 17.

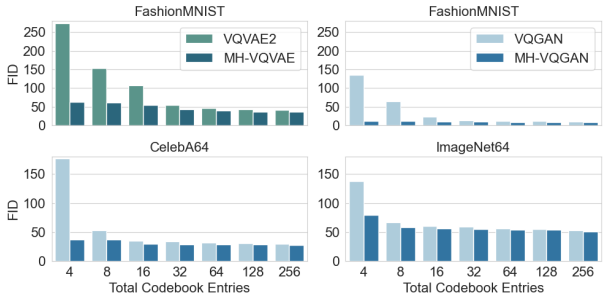

Figure 5: FID ↓ for validation samples. MH-VQ models outperform their counterparts at all codebook sizes and also scales to their performance limit with less codes.

Figure 6: Samples generated by MH-VQGAN using 4 codebook entries for FashionMNIST (left 6 columns) and 64 codebook entries for CelebA64 (middle 4 columns) and ImageNet64 (right 4 columns).

In Fig. 6, we present samples of MH-VQGAN across each dataset, visually demonstrating the realism and diversity in the generated outputs, achieved even with a minimalistic codebook design. This reinforces our argument for using the MH-VQ framework as an alternative to VQ for efficient latent representational learning.

Conclusion

This paper presents several novel concepts based on Multiple Hypothesis Dropout, a novel variant of dropout that creates multiple-output functions that stably and efficiently capture the statistical variability of targets. We build on this key component by introducing two similar architectures for two problems: Mixture of Multiple-Output functions (MoM) and MH-VQVAE. MoM can fit multi-modal distributions in output space while MH-VQVAE is designed for distributions in latent space. They both demonstrate improvements in quality, stability and scalability over existing approaches for these types of problems. We suspect these tools can generalize to other domains, such as robotics and reinforcement learning. We also believe that the predictive variance of MH Dropout networks has some correlation to uncertainty estimation but leave this connection for future work.

Acknowledgements

The authors would like to acknowledge the support of the Commonwealth of Australia and the Cybersecurity Cooperative Research Centre. This project was supported with computing infrastructure resources from CSIRO IMT Scientific Computing and DUG Technology Cloud, as well as technical expertise from Ondrej Hlinka of CSIRO and Rowan Worth of DUG. We thank Anh Ta and Kristen Moore for their valuable feedback on the manuscript.

References

Amini, A.; Soleimany, A.; Karaman, S.; and Rus, D. 2018. Spatial uncertainty sampling for end-to-end control. *arXiv preprint arXiv:1805.04829*.

Baldi, P.; and Sadowski, P. J. 2013. Understanding dropout. *Advances in neural information processing systems*, 26: 2814–2822.

Bengio, Y.; Léonard, N.; and Courville, A. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*.

Bishop, C. M. 1994. Mixture density networks.

Cui, H.; Radosavljevic, V.; Chou, F.-C.; Lin, T.-H.; Nguyen, T.; Huang, T.-K.; Schneider, J.; and Djuric, N. 2019. Multimodal trajectory predictions for autonomous driving using deep convolutional networks. In *2019 International Conference on Robotics and Automation (ICRA)*, 2090–2096. IEEE.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, 248–255. Ieee.

Esser, P.; Rombach, R.; and Ommer, B. 2021. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 12873–12883.

Firman, M.; Campbell, N. D.; Agapito, L.; and Brostow, G. J. 2018. Diversenet: When one right answer is not enough. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 5598–5607.

Gal, Y. 2016. Uncertainty in deep learning. *University of Cambridge*, 1(3).

Graves, A. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Guzman-Rivera, A.; Batra, D.; and Kohli, P. 2012. Multiple Choice Learning: Learning to Produce Multiple Structured Outputs. In *NIPS*, volume 1, 3. Citeseer.

Guzman-Rivera, A.; Kohli, P.; Glocker, B.; Shotton, J.; Sharp, T.; Fitzgibbon, A.; and Izadi, S. 2014. Multi-output learning for camera relocalization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1114–1121.

Ha, D.; and Eck, D. 2017. A neural representation of sketch drawings. *arXiv preprint arXiv:1704.03477*.

Ha, D.; and Schmidhuber, J. 2018. World models. *arXiv preprint arXiv:1803.10122*.

Heusel, M.; Ramsauer, H.; Unterthiner, T.; Nessler, B.; and Hochreiter, S. 2017. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *arXiv preprint arXiv:1706.08500*.

Hinton, G. E.; Srivastava, N.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. R. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Hinton, G. E.; and Zemel, R. 1993. Autoencoders, minimum description length and Helmholtz free energy. *Advances in neural information processing systems*, 6.

Ilg, E.; Cicek, O.; Galesso, S.; Klein, A.; Makansi, O.; Hutter, F.; and Brox, T. 2018. Uncertainty estimates and multi-hypotheses networks for optical flow. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 652–667.

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

Lee, K.; Hwang, C.; Park, K.; and Shin, J. 2017. Confident multiple choice learning. In *International Conference on Machine Learning*, 2014–2023. PMLR.

Lee, S.; Purushwalkam, S.; Cogswell, M.; Ranjan, V.; Crandall, D.; and Batra, D. 2016. Stochastic multiple choice learning for training diverse deep ensembles. *arXiv preprint arXiv:1606.07839*.

Liu, Z.; Luo, P.; Wang, X.; and Tang, X. 2018. Large-scale celebfaces attributes (celeba) dataset. *Retrieved August*, 15(2018): 11.

Makansi, O.; Ilg, E.; Cicek, O.; and Brox, T. 2019. Overcoming limitations of mixture density networks: A sampling and fitting framework for multimodal future prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 7144–7153.

Nguyen, D. D.; Liebowitz, D.; Nepal, S.; and Kanhere, S. S. 2023. Multiple Hypothesis Dropout: Estimating the Parameters of Multi-Modal Output Distributions. arXiv:2312.11735.

Nguyen, D. D.; Nepal, S.; and Kanhere, S. S. 2021. Diverse Multimedia Layout Generation with Multi Choice Learning. In *Proceedings of the 29th ACM International Conference on Multimedia*, 218–226.

Prokudin, S.; Gehler, P.; and Nowozin, S. 2018. Deep directional statistics: Pose estimation with uncertainty quantification. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 534–551.

Ramesh, A.; Pavlov, M.; Goh, G.; Gray, S.; Voss, C.; Radford, A.; Chen, M.; and Sutskever, I. 2021. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, 8821–8831. PMLR.

Razavi, A.; Van den Oord, A.; and Vinyals, O. 2019. Generating diverse high-fidelity images with vq-vae-2. *Advances in neural information processing systems*, 32.

Rombach, R.; Blattmann, A.; Lorenz, D.; Esser, P.; and Ommer, B. 2021. High-Resolution Image Synthesis with Latent Diffusion Models. *arXiv preprint arXiv:2112.10752*.

Rombach, R.; Blattmann, A.; Lorenz, D.; Esser, P.; and Ommer, B. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10684–10695.

Rupprecht, C.; Laina, I.; DiPietro, R.; Baust, M.; Tombari, F.; Navab, N.; and Hager, G. D. 2017. Learning in an uncertain world: Representing ambiguity through multiple hypotheses. In *Proceedings of the IEEE International Conference on Computer Vision*, 3591–3600.

Sajjadi, M. S.; Bachem, O.; Lucic, M.; Bousquet, O.; and Gelly, S. 2018. Assessing generative models via precision and recall. *Advances in Neural Information Processing Systems*, 31.

Van Den Oord, A.; Vinyals, O.; et al. 2017. Neural discrete representation learning. *Advances in neural information processing systems*, 30.

Xiao, H.; Rasul, K.; and Vollgraf, R. 2017. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*.

Zhou, Y.; Gao, J.; and Asfour, T. 2020. Movement primitive learning and generalization: Using mixture density networks. *IEEE Robotics & Automation Magazine*, 27(2): 22–32.