Unsupervised Radio Scene Analysis Using Variational Autoencoder

Hao Chen and Seung-Jun Kim
Department of Computer Science & Electrical Engineering
University of Maryland, Baltimore County, Baltimore, MD, USA
E-mail: {chenhao1,sjkim}@umbc.edu

***Abstract*—An unsupervised learning-based radio frequency (RF) scene analysis method is proposed in a variational autoencoder (VAE) framework. The method can process multiantenna RF signals that contain multiple concurrent transmissions in the time-frequency-space domains to extract individual component signals' latent codes and channel state information. The algorithm can be trained using mixture data sets without labels or clean component data, and does not require iterative optimization to estimate relevant parameters in each scene. To improve the efficiency of training and operation of the deep neural network (DNN), a novel successive estimation architecture is devised, which can be viewed as employing an attention mechanism in the spatial dimension. Numerical experiments verify the excellent performance and robustness of the proposed method compared to existing benchmarks.**

***Index Terms*—Deep learning, radio frequency spectrum, radio scene analysis, unsupervised learning, variational autoencoder.**

I. Introduction

Analyzing the radio frequency (RF) spectrum activities is an important prerequisite for efficient spectrum sharing, interference control, and security assurance in wireless networks. In cognitive radio (CR) systems, estimating spectrum occupancy in the temporal, spectral, and spatial domains plays a crucial role for opportunistic spectrum access. Understanding the dynamics of spectrum usage allows the system to accurately assess the risk of causing/receiving harmful interference and aggressively interweave secondary transmissions across spectrum holes.

Spectrum sensing algorithms in the CR literature determine the presence or absence of transmissions often relying on the signal strength or power spectrum measurements, known pilot signals, and cyclostationarity or higher-order statistical features [1]. These methods may thus suffer from detrimental fading effects in the wireless channel or require significant domain knowledge on the signal characteristics. Furthermore, they cannot effectively cope with complex radio scenes such as the ones containing multiple concurrent transmissions.

There are only a limited number of methods developed for analyzing RF scenes laden with multiple transmissions. Blind source separation techniques were employed to isolate constituent transmissions in [2], [3]. A tensor decomposition approach was taken to cluster the power spectrum over time and perform a fourth-order spectral analysis in [4]. Knowledge of the transmission protocol was exploited to extract robust detection features for mixed signals in [5]. Dictionary learning algorithms were tailored for identifying constituent signals in mixtures in [6]. However, in these works, the unique time-frequency (TF) patterns of the component signals were not fully exploited in a machine learning framework.

Recently, deep learning approaches have been actively pursued for analyzing mixture signals. Variational autoencoders (VAEs) were trained to capture the TF patterns of the component signals, which were then used to facilitate source separation through iterative optimization [7]. Such supervised learning approaches require both target mixture and clean source data sets, which may be expensive to construct. To mitigate this, weakly supervised and unsupervised methods have been developed in the VAE framework [8]–[10]. In [8], the labels of the sources that are present in the mixture are incorporated to learn the TF patterns from mixture data sets without source-only data sets. In [9], a VAE that produces the latent codes of all component signals simultaneously was employed for source separation, which was trained in an unsupervised fashion using only mixture data sets. These algorithms dealt with single-channel input signals. In [10], multichannel mixture RF signals were used to train a deep neural network (DNN) to capture TF patterns, while an expectation-maximization (EM) method was derived to estimate vector channel parameters as well.

In this work, our focus is on complex RF scene analysis using unsupervised learning based on multiantenna mixture signal data. Rather than employing iterative algorithms to estimate the channel and other parameters as in [7], [10], which can significantly increase the computational complexity of the method in both training and operational phases, our goal is to design an algorithm entirely based on DNNs in the VAE framework. However, a black-box deep learning approach might require a lot of training data to achieve good performance. To improve efficiency, a specialized successive estimation architecture is devised, inspired by array signal processing insights as well as attention mechanisms shown to markedly boost the performance in many deep learning applications [11].

The rest of the paper is organized as follows. The signal model and the problem statement are presented in Sec. II. The parameter estimation algorithm and the overall architecture are described in Sec. III. Results from numerical tests are presented in Sec. IV. Conclusions and future research directions are provided in Sec. V.

II. Signal Model and Problem Statement

Let $\mathbf{x}(t) \in \mathbb{C}^M, t = 1, \ldots, T$, be downconverted sample vectors of an RF signal received using M antennas. The signal contains transmissions from J transmitters. For simplicity, it is assumed that the transmitters have single antennas, the channel is frequency-nonselective, and J is known. Upon denoting the signal transmitted at time t from the j-th transmitter as $s_j(t)$, the channel between the j-th transmitter and the receiver as $\mathbf{h}_j \in \mathbb{C}^M$, and the receiver noise vector as $\mathbf{b}(t) \in \mathbb{C}^M$, $\mathbf{x}(t)$ can be expressed as

$$\mathbf{x}(t) = \sum_{j=1}^{J} \mathbf{h}_j s_j(t) + \mathbf{b}(t), \ t = 1, \ldots, T. \tag{1}$$

Let us define $\mathbf{H} := [\mathbf{h}_1, \ldots, \mathbf{h}_J]$ and $\mathbf{s}(t) := [s_1(t), \ldots, s_J(t)]^\top$, where $^\top$ denotes transposition. Then, (1) can be rewritten as

$$\mathbf{x}(t) = \mathbf{H}\mathbf{s}(t) + \mathbf{b}(t), \ t = 1, \ldots, T. \tag{2}$$

It is postulated that for each j, transmission $s_j(t)$ belongs to a distinct type with unique features, such as Wi-Fi or Bluetooth transmissions. These patterns are to be captured automatically using machine learning. However, instead of using a set of training waveforms for *each* signal type, a training set of *mixture* waveforms is utilized. In other words, the algorithm must learn from samples collected in a densely interfered RF environment in an unsupervised fashion, without relying on separate labels or clean signal data sets. Among others, this requires estimating the transmitter channels $\{\mathbf{h}_j\}$ as well.

The patterns of different signal types can be extracted in the TF domain. Let $\mathbf{x}(n, f) \in \mathbb{C}^M$, $\mathbf{s}(n, f) \in \mathbb{C}^J$, and $\mathbf{b}(n, f) \in \mathbb{C}^M$, with time (window) index $n = 1, \ldots, N$ and frequency bin index $f = 1, \ldots, F$, be the short-time Fourier transforms (STFTs) of $\mathbf{x}(t)$, $\mathbf{s}(t)$, and $\mathbf{b}(t)$, respectively. To formulate an inference problem, probabilistic assumptions are made on the signals. Specifically, it is assumed that $\mathbf{s}(n, f)$ is complex Gaussian with mean $\mathbf{0}$ and covariance

$$\mathbf{R}_{\mathbf{s}}(n, f) := \begin{bmatrix} v_1(n, f) & 0 & \cdots & 0 \\ & & \ddots & \\ 0 & \cdots & 0 & v_J(n, f) \end{bmatrix}. \tag{3}$$

Also, it is assumed that $s_j(n, f)$ is statistically independent of $s_{j'}(n', f')$ unless $j = j'$, $n = n'$, and $f = f'$, conditioned on $\{v_j(n, f)\}$ [12]. Then, it is straightforward to verify that $\mathbf{x}(n, f)$ is independent across n and f, and has a complex Gaussian distribution with mean $\mathbf{0}$ and covariance

$$\mathbf{R}_{\mathbf{x}}(n, f) := \mathbf{H}\mathbf{R}_{\mathbf{s}}(n, f)\mathbf{H}^{\mathcal{H}} + \mathbf{R}_{\mathbf{b}} \tag{4}$$

where $\cdot^{\mathcal{H}}$ denotes Hermitian transpose and $\mathbf{R}_{\mathbf{b}}$ the covariance matrix of $\mathbf{b}(n, f)$ (assumed stationary over n and f).

To capture the TF patterns in the j-th transmitted signal, matrix $\mathbf{V}_j \in \mathbb{R}_+^{N \times F}$, whose (n, f)-entry is $v_j(n, f)$, is modeled via a DNN $f_{\boldsymbol{\theta}}$ with parameter vector $\boldsymbol{\theta}$ as

$$\mathbf{V}_j = f_{\boldsymbol{\theta}}(\mathbf{z}_j). \tag{5}$$

Here, $\mathbf{z}_j \in \mathbb{R}^K$ is a latent variable input to the neural network, with a much lower dimension than that of $\mathbf{V}_j$, which is NF. Variable $\mathbf{z}_j$ not only encodes the differences in the signal types, but also any variabilities in the patterns within each signal type. Upon defining $\mathbf{Z} := \{\mathbf{z}_j\}_{j=1}^J$ and $\boldsymbol{\zeta} := (\mathbf{Z}, \mathbf{H}, \mathbf{R}_{\mathbf{b}})$, the conditional distribution for $\mathbf{X}$ given all the latents $\boldsymbol{\zeta}$ is then given by

$$p_{\boldsymbol{\theta}}(\mathbf{X}|\boldsymbol{\zeta}) = \prod_{n,f} \mathcal{N}(\mathbf{x}(n, f); \mathbf{0}, \mathbf{R}_{\mathbf{x}}(n, f)). \tag{6}$$

We adopt the prior for $\boldsymbol{\zeta}$ given by

$$p(\boldsymbol{\zeta}) = \prod_{j=1}^{J} p(\mathbf{z}_j) p(\mathbf{H}) p(\mathbf{R}_{\mathbf{b}}) \tag{7}$$

where $p(\mathbf{z}_j)$ is assumed to be Gaussian with mean $\mathbf{0}$ and covariance $\mathbf{I}$, and $p(\mathbf{H})$ and $p(\mathbf{R}_{\mathbf{b}})$ are assumed to be uniform (constant). Based on the generative model for $\mathbf{X}$ in (6)–(7), our goal is to train an algorithm in an unsupervised fashion, which can infer $\boldsymbol{\zeta}$ for a given $\mathbf{X}$.

III. Variational Learning

A. Evidence Lower-Bound

Directly estimating $\boldsymbol{\zeta}$ from $\mathbf{X}$ is intractable since computing the posterior $p_{\boldsymbol{\theta}}(\boldsymbol{\zeta}|\mathbf{X})$ involves challenging multidimensional integration. A practical approach is to employ a VAE framework [13]. Let $\mathcal{D}(\cdot||\cdot)$ represent the Kullback-Leibler (KL) divergence. Introducing an approximate posterior distribution $q_{\boldsymbol{\phi}}(\boldsymbol{\zeta}|\mathbf{X})$, one can verify that

$$\log p_{\boldsymbol{\theta}}(\mathbf{X}) = \mathcal{D}(q_{\boldsymbol{\phi}}(\boldsymbol{\zeta}|\mathbf{X})||p_{\boldsymbol{\theta}}(\boldsymbol{\zeta}|\mathbf{X})) + \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}) \tag{8}$$

with

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}) := \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{\zeta}|\mathbf{X})}\{\log p_{\boldsymbol{\theta}}(\mathbf{X}|\boldsymbol{\zeta})\} - \mathcal{D}\left(q_{\boldsymbol{\phi}}(\boldsymbol{\zeta}|\mathbf{X})||p_{\boldsymbol{\theta}}(\boldsymbol{\zeta})\right). \tag{9}$$

Thus, given $\boldsymbol{\theta}$, one can minimize the KL divergence between the true and the approximate posteriors by maximizing $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi})$ w.r.t. $\boldsymbol{\phi}$. Furthermore, since KL divergence is non-negative, the log-likelihood $\log p_{\boldsymbol{\theta}}(\mathbf{X})$ is lower-bounded by $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi})$, which is called the evidence lower-bound (ELBO). Therefore, one can improve the likelihood by maximizing $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi})$ w.r.t. $\boldsymbol{\theta}$. In summary, variational learning amounts to maximizing the ELBO w.r.t. both $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$.

In our case, let the true posterior of $\boldsymbol{\zeta}$ be approximated by a probabilistic encoder model given by

$$q_{\boldsymbol{\phi}}(\boldsymbol{\zeta}|\mathbf{X}) = q_{\boldsymbol{\phi}}(\mathbf{Z}|\mathbf{H}, \mathbf{R}_{\mathbf{b}}, \mathbf{X}) q_{\boldsymbol{\phi}}(\mathbf{H}|\mathbf{X}) q_{\boldsymbol{\phi}}(\mathbf{R}_{\mathbf{b}}|\mathbf{X}) \tag{10}$$

$$= \prod_{j=1}^{J} q_{\boldsymbol{\phi}}(\mathbf{z}_j|\mathbf{H}, \mathbf{R}_{\mathbf{b}}, \mathbf{X}) \cdot q_{\boldsymbol{\phi}}(\mathbf{H}|\mathbf{X}) q_{\boldsymbol{\phi}}(\mathbf{R}_{\mathbf{b}}|\mathbf{X}). \tag{11}$$

For simplicity, point estimates are used for $q_{\boldsymbol{\phi}}(\mathbf{H}|\mathbf{X})$ and $q_{\boldsymbol{\phi}}(\mathbf{R}_{\mathbf{b}}|\mathbf{X})$, i.e.,

$$q_{\boldsymbol{\phi}}(\mathbf{H}|\mathbf{X}) = \delta(\mathbf{H} - g_{\boldsymbol{\phi}}^H(\mathbf{X})) \tag{12}$$

$$q_{\boldsymbol{\phi}}(\mathbf{R}_{\mathbf{b}}|\mathbf{X}) = \delta(\mathbf{R}_{\mathbf{b}} - g_{\boldsymbol{\phi}}^{R_b}(\mathbf{X})) \tag{13}$$

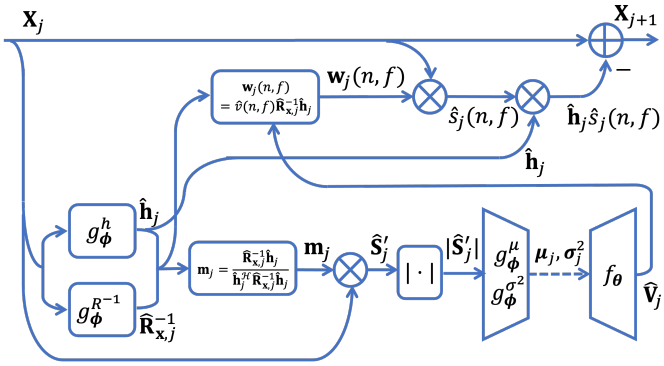

Fig. 1. Successive estimation architecture.

where δ is the Dirac delta function, and $g_{\boldsymbol{\phi}}^{H}$ and $g_{\boldsymbol{\phi}}^{R_b}$ are DNN mappings parameterized by $\boldsymbol{\phi}$. Then, the KL divergence term $\mathcal{D}\left(q_{\boldsymbol{\phi}}(\boldsymbol{\zeta}|\mathbf{X})||p_{\boldsymbol{\theta}}(\boldsymbol{\zeta})\right)$ in (9) can be simplified to

$$\sum_{j=1}^{J}\mathcal{D}(q_{\boldsymbol{\phi}}(\mathbf{z}_j|\mathbf{H}=g_{\boldsymbol{\phi}}^{H}(\mathbf{X}),\mathbf{R}_{\mathbf{b}}=g_{\boldsymbol{\phi}}^{R_b}(\mathbf{X}),\mathbf{X})||p_{\boldsymbol{\theta}}(\mathbf{z}_j))+C \tag{14}$$

where C represents a constant that does not depend on $\boldsymbol{\theta}$ or $\boldsymbol{\phi}$. The detail on the derivation of (14) is relegated to Appendix.

We model $q_{\boldsymbol{\phi}}(\mathbf{z}_j|\mathbf{H},\mathbf{R}_{\mathbf{b}},\mathbf{X})$ as Gaussian with mean $\boldsymbol{\mu}_j$ and covariance $\mathrm{diag}\{\boldsymbol{\sigma}_j^2\}$, where $\mathrm{diag}\{\boldsymbol{\sigma}_j^2\}$ is a diagonal matrix with the entries of $\boldsymbol{\sigma}_j^2$ on the diagonal. Upon defining $\mathbf{U}:=\{\boldsymbol{\mu}_j\}_j$ and $\boldsymbol{\Sigma}:=\{\boldsymbol{\sigma}_j^2\}_j$, $\mathbf{U}$ and $\boldsymbol{\Sigma}$ are obtained from DNNs as

$$\mathbf{U}=g_{\boldsymbol{\phi}}^{U}(\mathbf{H},\mathbf{R}_{\mathbf{b}},\mathbf{X}) \tag{15}$$

$$\boldsymbol{\Sigma}=g_{\boldsymbol{\phi}}^{\Sigma}(\mathbf{H},\mathbf{R}_{\mathbf{b}},\mathbf{X}). \tag{16}$$

Then, (14) can be computed in a closed form as [13]

$$\sum_{j=1}^{J}\left[\frac{1}{2}\sum_{k=1}^{K}\left(-1-\log(\sigma_{j,k}^2)+\mu_{j,k}^2+\sigma_{j,k}^2\right)\right]+C \tag{17}$$

where $\sigma_{j,k}^2$ and $\mu_{j,k}$ are the k-th entries of $\boldsymbol{\sigma}_j^2$ and $\boldsymbol{\mu}_j$, respectively.

B. Successive Estimation Architecture

DNNs are employed for mappings $g_{\boldsymbol{\phi}}^{H}$, $g_{\boldsymbol{\phi}}^{U}$ and $g_{\boldsymbol{\phi}}^{\Sigma}$. (For simplicity, a pre-specified constant function is used for $g_{\boldsymbol{\phi}}^{R_b}$ in this work.) In principle, one could build a single large DNN that processes input $\mathbf{X}$ and produces $\mathbf{H}$, $\mathbf{U}$, and $\boldsymbol{\Sigma}$. However, such a black box architecture might require a lot of data to train. To improve efficiency, we adopt a specialized successive estimation architecture shown in Fig. 1, inspired by array signal processing methods and attention mechanisms in the deep learning literature [11].

First, instead of estimating $\{\mathbf{h}_j\}$ for all j in one shot, a neural network $g_{\boldsymbol{\phi}}^{h}$ is used to map the j-th stage input $\mathbf{X}_j$ to a single component's channel vector $\hat{\mathbf{h}}_j$; i.e., $\hat{\mathbf{h}}_j:=g_{\boldsymbol{\phi}}^{h}(\mathbf{X}_j)$. In the first stage with $j=1$, $\mathbf{X}_1$ is set to $\mathbf{X}$. Also trained is a neural network $g_{\boldsymbol{\phi}}^{R^{-1}}$ to obtain an estimate of the inverse of the signal covariance matrix. That is, $\hat{\mathbf{R}}_{\mathbf{x},j}^{-1}:=g_{\boldsymbol{\phi}}^{R^{-1}}(\mathbf{X}_j)$ [14].

Then, spatial filtering is done via a minimum-variance distortion-less response (MVDR) beamformer given by

$$\mathbf{m}_j:=\frac{\hat{\mathbf{R}}_{\mathbf{x},j}^{-1}\hat{\mathbf{h}}_j}{\hat{\mathbf{h}}_j^{\mathcal{H}}\hat{\mathbf{R}}_{\mathbf{x},j}^{-1}\hat{\mathbf{h}}_j} \tag{18}$$

to obtain a rough estimate of the j-th component signal as

$$\hat{s}'_j(n,f):=\mathbf{m}_j^{\mathcal{H}}\mathbf{x}_j(n,f),\ \forall n,f. \tag{19}$$

One may view this as a spatial attention mechanism, by which the dimension of the variational encoder input is significantly reduced. Attention mechanisms have proved instrumental in various deep learning applications [15], [16].

Subsequently, collect the magnitudes $|\hat{s}'_j(n,f)|$ into an $N\times F$ matrix $|\hat{\mathbf{S}}'_j|$. It is then assumed that $\boldsymbol{\mu}_j$ and $\boldsymbol{\sigma}_j^2$ can be obtained from neural networks as

$$\boldsymbol{\mu}_j=g_{\boldsymbol{\phi}}^{\mu}(|\hat{\mathbf{S}}'_j|) \tag{20}$$

$$\boldsymbol{\sigma}_j^2=g_{\boldsymbol{\phi}}^{\sigma^2}(|\hat{\mathbf{S}}'_j|). \tag{21}$$

Now, one can take a sample from $\mathcal{N}(\boldsymbol{\mu}_j,\mathrm{diag}\{\boldsymbol{\sigma}_j^2\})$, which is denoted as $\hat{\mathbf{z}}_j$. Then, $\mathbf{V}_j$ can be estimated from the decoder as $\hat{\mathbf{V}}_j=f_{\boldsymbol{\theta}}(\hat{\mathbf{z}}_j)$.

Once the variance estimates $\{\hat{v}_j(n,f)\}_{n,f}:=\hat{\mathbf{V}}_j$ are obtained, one can get a cleaner estimate of $s_j(n,f)$ by using this information. Specifically, for each n and f,

$$\mathbf{w}_j(n,f):=\hat{v}_j(n,f)\hat{\mathbf{R}}_{\mathbf{x},j}^{-1}\hat{\mathbf{h}}_j \tag{22}$$

$$\hat{s}_j(n,f):=\mathbf{w}_j(n,f)^{\mathcal{H}}\mathbf{x}_j(n,f). \tag{23}$$

Unlike the space-only filtering in (19), note here that filtering is done over space, time, and frequency domains. In fact, if we had a signal covariance estimate $\hat{\mathbf{R}}_{\mathbf{x},j}(n,f)$ available at each TF bin (n,f), we could have used it in (22) in the place of $\hat{\mathbf{R}}_{\mathbf{x},j}$, which would have corresponded to multichannel Wiener filtering. Since we do not actually have access to $\{\hat{\mathbf{R}}_{\mathbf{x},j}(n,f)\}$, simply $\hat{\mathbf{R}}_{\mathbf{x},j}$ is used, which turns out to be quite effective according to our numerical tests.

Now the estimated component signal is subtracted from $\mathbf{x}_j(n,f)$ to yield the stage-$(j+1)$ input $\mathbf{X}_{j+1}=\{\mathbf{x}_{j+1}(n,f)\}$.

$$\mathbf{x}_{j+1}(n,f):=\mathbf{x}_j(n,f)-\hat{\mathbf{h}}_j\hat{s}_j(n,f),\ \forall n,f. \tag{24}$$

The architecture generates $\hat{\mathbf{H}}:=\{\hat{\mathbf{h}}_j\}_j$, $\mathbf{U}$ and $\boldsymbol{\Sigma}$ by repeating this process over $j=1,2,\ldots,J$.

C. DNN Training

The DNN training can be done using I samples of RF scene STFTs $\{\mathbf{X}[i]\}_{i=1}^{I}$. Let $\mathcal{I}_\ell$ be the set of sample indices belonging to the ℓ-th mini-batch of data. The overall algorithm is listed in Table I, where input η is a positive step size parameter, and $\sigma_b^2>0$ is an estimate of background noise level used to set $\hat{\mathbf{R}}_b=\sigma_b^2\mathbf{I}$. The update for DNN parameters $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ is done through the stochastic gradient ascent based on the ELBO in (9), where the expectation is evaluated approximately using a sample $\hat{\mathbf{Z}}[i]:=\{\hat{\mathbf{z}}_j[i]\}_{j=1}^{J}$ generated in lines 9–10 of Table I. The gradient of the ELBO in line 15 can thus be computed using the reparameterization trick [13].

TABLE I
DNN TRAINING ALGORITHM.

Input: $\{\mathbf{X}[i]\}_{i=1}^{I}$, J, η, σ_b^2, and $\{\mathcal{I}_\ell\}_\ell$
Output: $\boldsymbol{\phi}^*$ and $\boldsymbol{\theta}^*$

Initialize $\boldsymbol{\phi}^{[0]}, \boldsymbol{\theta}^{[0]}$ randomly and set $\hat{\mathbf{R}}_b = \sigma_b^2 \mathbf{I}$
For $\ell = 0, 1, \ldots$ /* the ℓ-th mini-batch */
  For $i \in \mathcal{I}_\ell$
    Set $\mathbf{X}_1 \leftarrow \mathbf{X}[i]$
    For $j = 1, 2, \ldots, J$ /* the j-th component */
      Compute $\hat{\mathbf{h}}_j[i] = g^h_{\boldsymbol{\phi}^{[\ell]}}(\mathbf{X}_j)$ and $\hat{\mathbf{R}}^{-1}_{\mathbf{x},j} = g^{R^{-1}}_{\boldsymbol{\phi}^{[\ell]}}(\mathbf{X}_j)$
      Compute $\hat{\mathbf{S}}'_j := \{\hat{s}'_j(n,f)\}_{n,f}$ from (18)–(19)
      Compute $\boldsymbol{\mu}_j[i] = g^{\mu}_{\boldsymbol{\phi}^{[\ell]}}(|\hat{\mathbf{S}}'_j|)$ and $\boldsymbol{\sigma}^2_j[i] = g^{\sigma^2}_{\boldsymbol{\phi}^{[\ell]}}(|\hat{\mathbf{S}}'_j|)$
      Sample $\boldsymbol{\epsilon}_j[i] \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
      Set $\hat{\mathbf{z}}_j[i] = (\mathrm{diag}\{\boldsymbol{\sigma}^2_j[i]\})^{1/2}\boldsymbol{\epsilon}_j[i] + \boldsymbol{\mu}_j[i]$
      Compute $\hat{\mathbf{V}}_j[i] = f_{\boldsymbol{\theta}^{[\ell]}}(\hat{\mathbf{z}}_j[i])$
      Compute $\{\hat{s}_j(n,f)\}_{n,f}$ from (22)–(23)
      Update $\mathbf{X}_{j+1}[i] := \{\mathbf{x}_{j+1}[i](n,f)\}_{n,f}$ from (24)
    Next j
    Let $\mathcal{L}[i] = p_{\boldsymbol{\theta}}(\mathbf{X}[i]|\hat{\mathbf{Z}}[i], \hat{\mathbf{H}}[i], \hat{\mathbf{R}}_b)$
      $-\sum_{j=1}^{J}\left[\frac{1}{2}\sum_{k=1}^{K}(-1 - \log(\sigma^2_{j,k}[i]) + \mu^2_{j,k}[i] + \sigma^2_{j,k}[i])\right]$
  Next i
  Update $\boldsymbol{\phi}^{[\ell+1]} = \boldsymbol{\phi}^{[\ell]} + \frac{\eta}{|\mathcal{I}_\ell|}\sum_{i\in\mathcal{I}_\ell}\frac{\partial \mathcal{L}[i]}{\partial \boldsymbol{\phi}}$
  Update $\boldsymbol{\theta}^{[\ell+1]} = \boldsymbol{\theta}^{[\ell]} + \frac{\eta}{|\mathcal{I}_\ell|}\sum_{i\in\mathcal{I}_\ell}\frac{\partial \mathcal{L}[i]}{\partial \boldsymbol{\theta}}$
Next ℓ
Set $\boldsymbol{\phi}^* = \boldsymbol{\phi}^{[\infty]}$ and $\boldsymbol{\theta}^* = \boldsymbol{\theta}^{[\infty]}$

IV. NUMERICAL TESTS

A. Experiment Setup

The proposed algorithm was tested on a RF signal data set collected in the 2.4 GHz band using a software-defined radio. The signal bandwidth is 40 MHz. There are 6 signal types in total, namely Wi-Fi with high occupancy (which we term "Wi-Fi1"), Wi-Fi with low occupancy (Wi-Fi2), Bluetooth (BT), Bluetooth Low Energy (BLE), and two types of proprietary frequency-hopping spread spectrum (FHSS) signals (FHSS1 and FHSS2) [6]. The signals were collected in a RF shield box using a single receive antenna. The multi-antenna mixture signals were then synthesized for our experiment by multiplying to the individual signals steering vectors corresponding to a uniform linear array with random incident angles (A minimum angle spacing of 10° was maintained) and adding them up. The component signals were of equal power. For each sample, a snapshot of duration 100 ms was taken, which was processed by STFT with $N = 64$ and $F = 66$. Unless otherwise stated, training was done using $I = 18,000$ samples, and validation and testing using $1,000$ samples each.

B. DNN Implementation

The DNN implementation for g^h_ϕ and $g^{R^{-1}}_\phi$ is shown in Fig. 2. The vertical lines correspond to the input/output vectors. The arrows with different colors indicate fully-connected layers with various nonlinearities. Group normalization was used in each layer except the final layers [17]. The size of the transformation matrix is written on top of each arrow. The input to the DNN is based on a signal covariance matrix estimate $\hat{\mathbf{R}}_{\mathbf{x},j} := \frac{1}{NF}\sum_{n,f}\mathbf{x}_j(n,f)\mathbf{x}_j(n,f)^{\mathcal{H}}$. The lower-triangular elements of $\hat{\mathbf{R}}_{\mathbf{x},j}$ are taken and their real and imaginary parts are stacked in a real vector of dimension $\alpha := M^2$, which is the input to the DNN. The output from the lower branch of the DNN in Fig. 2 is the angle of arrival, a scalar, which is used to construct the steering vector $\hat{\mathbf{h}}_j$ based on the antenna array geometry. From the upper branch, an α-dimensional vector is output to construct $\hat{\mathbf{R}}^{-1}_{\mathbf{x},j}$. The implementation of the encoders g^μ_ϕ and $g^{\sigma^2}_\phi$, and the decoder $f_{\boldsymbol{\theta}}$ is shown in Fig. 3. The vertical lines are tensors of the indicated dimensions. The encoder input $|\hat{\mathbf{S}}'_j|$ is an $N \times F$ real matrix. The dimension K of $\mathbf{z}_j$ is chosen to be 32. The the decoder output is squared and clipped by a small threshold to ensure that it is nonnegative and strictly larger than 0.

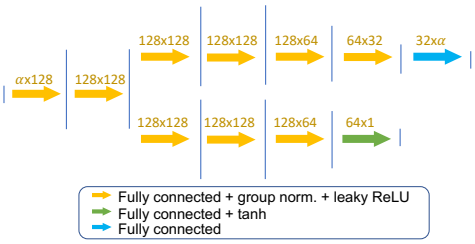

Fig. 2. DNNs for g^h_ϕ and $g^{R^{-1}}_\phi$

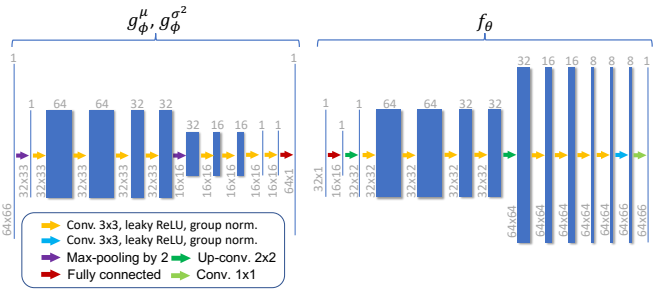

Fig. 3. DNNs for g^μ_ϕ, $g^{\sigma^2}_\phi$, and $f_{\boldsymbol{\theta}}$

C. Test with Mixtures of 3 Signal Types

First, we performed an experiment using mixtures of BLE, FHSS1, and Wi-Fi2 signals, as these types have quite distinct spectrogram patterns. The number of receive antennas is also set to $M = 3$. For comparison, the EM and the NEM methods derived in [10] were also tested. The EM method does not employ a neural network architecture. The NEM method incorporates a DNN but retains EM-based iterations in both training and testing phases, which incurs significant computational complexity. On the other hand, our proposed algorithm does not require iterations.

Fig. 4 shows the performance of the tested algorithms at various signal-to-noise power ratios (SNRs). Different SNR values were simulated by adding white Gaussian noise to the samples. The left panel in Fig. 4 depicts the correlation coefficient between the true channel vector $\mathbf{h}_j$ and the estimated vector $\hat{\mathbf{h}}_j$ corresponding to component signal j, averaged over j. The right panel shows the correlation coefficient between the true component STFT $\mathbf{S}_j := \{s_j(n,f)\}$ and the estimated one $\check{\mathbf{S}}_j := \{\check{s}_j(n,f)\}$, also averaged over j. The final estimates $\{\hat{\mathbf{S}}_j\}_j$ are computed through multichannel Wiener filtering as

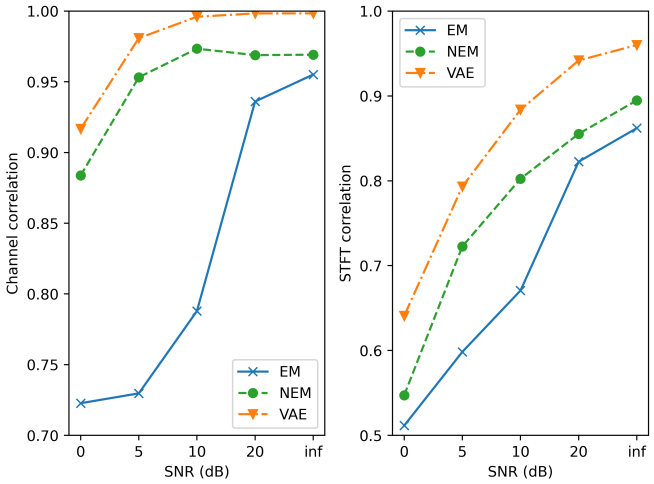

Fig. 4. Channel and STFT estimation performance for 3-type mixtures.

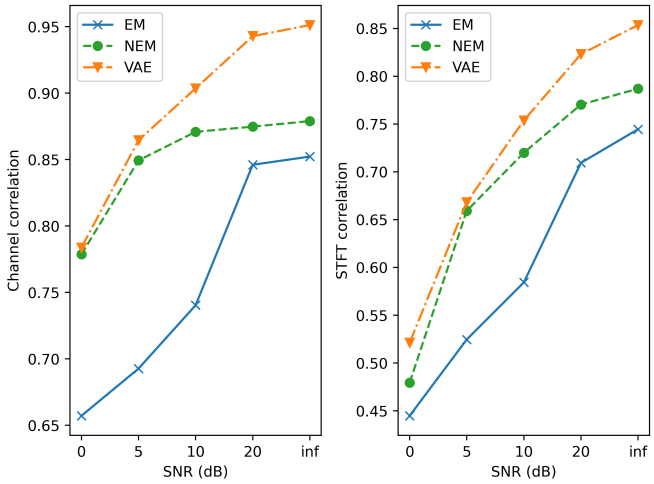

Fig. 6. Channel and STFT estimation performance for 6-type mixtures.

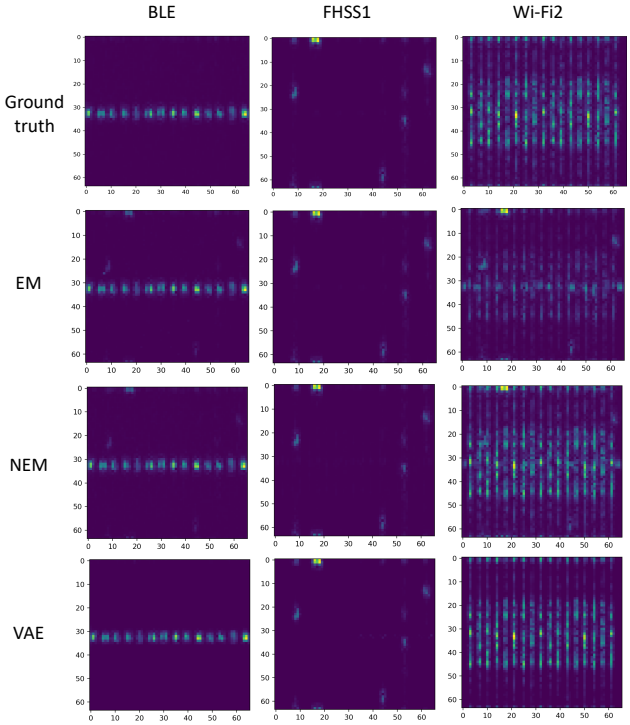

Fig. 5. True and estimated component spectrograms.

$\check{\mathbf{s}}(n,f) := \hat{\mathbf{R}}_{\mathbf{s}}(n,f)\hat{\mathbf{H}}^{\mathcal{H}}\hat{\mathbf{R}}_{\mathbf{x}}(n,f)^{-1}\mathbf{x}(n,f)$, where $\hat{\mathbf{R}}_{\mathbf{x}}(n,f)$ is computed from (4) with $\mathbf{H}$, $\mathbf{R}_s(n,f)$, and $\mathbf{R}_b$ replaced with $\hat{\mathbf{H}}$, $\hat{\mathbf{R}}_s(n,f) := \text{diag}\{\hat{v}_j(n,f)\}_j$, and $\hat{\mathbf{R}}_b$, respectively. It can be seen that the proposed VAE-based method outperforms the EM and the NEM algorithms in terms of both channel and STFT estimation. Both the VAE and the NEM algorithms are robust against low SNR conditions. However, the VAE-based algorithm achieves better performance at a much lower complexity than the NEM algorithm.

In Fig. 5, exemplary component spectrograms estimated from a mixture using the EM (the 2nd row), NEM (the 3rd row), and VAE (the last row) algorithms at the ∞ SNR are shown, together with the ground truth spectrograms (the first row). The average STFT correlation values for the shown instances are 0.89, 0.97 and 0.99, for the EM, NEM, and VAE methods, respectively. From the EM result, it can be seen that Wi-Fi2 is not very clearly estimated and FHSS1 is leaking into the BLE and Wi-Fi2 components. In the NEM result, the estimated Wi-Fi2 signal is much stronger, but traces of FHSS1 still exist in BLE and Wi-Fi2. The proposed VAE method recovers the component spectrograms cleanly, very close to the ground truth patterns.

D. Test with Mixtures of 6 *Signal Types*

The test was repeated with mixture signals containing 6 different types of RF signals. The number of antennas was set to $M = 6$. The channel and the STFT correlation performances are shown in Fig. 6 for varying SNRs. Since some signal types possess quite similar spectrograms (such as FHSS1/FHSS2, BT/BLE, and Wi-Fi1/Wi-Fi2), the problem becomes much harder. Thus, the correlation values are lower than those of the 3-type mixture case. Still, the VAE-based algorithm is seen to provide much improved performance over the EM and the NEM benchmarks.

Finally, we evaluated the performance of the algorithms when trained using a varying number of training samples. Fig. 7 shows the channel estimation performance of the VAE and the NEM algorithms with $J = M = 6$ at SNR $= \infty$. It can be seen that when the training set size is small (with less than $6,000$ samples), the performance of the NEM algorithm is actually better than the VAE. The EM algorithm does not need training, which is shown as a flat line. The NEM algorithm maintains robust performance even with as few as $1,000$ samples, when the VAE algorithm deteriorates significantly. Recall that in the NEM algorithm, the channel estimation is done by an iterative signal processing loop, whereas in the VAE-based algorithm, everything is done through a DNN architecture. Thus, it is reasonable that with limited training samples, the signal processing-aided algorithm can outperform methods entirely based on machine learning. However, as the number of samples increases, the VAE architecture catches up and eventually significantly outperforms the NEM.

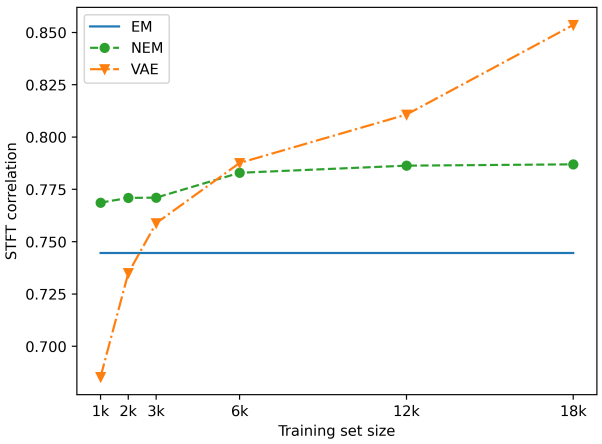

Fig. 7. Performance comparison with varying training set size.

V. Conclusions and Future Directions

A multiantenna-based RF scene analysis algorithm has been proposed in a deep VAE framework to automatically learn the unique TF patterns of various types of transmissions. The algorithm can be trained in an unsupervised fashion using signal sets that contain mixtures of interferers without relying on labels or clean signals of individual transmitters. The algorithm does not require iterative optimization to estimate relevant parameters in each scene, but rather exploits DNNs end-to-end, which renders the algorithm computationally very efficient both for training and operation. A novel successive estimation architecture was designed for the DNNs, which provides with a spatial attention mechanism, significantly reducing the training burden of the algorithm. Numerical tests verified that the proposed method can outperform benchmark algorithms in terms of channel and component signal estimation performances, as well as robustness to SNR conditions. Future research directions include extension of the method to more general frequency-selective channels and automatic determination of the number of interferers in the mixture.

Appendix: Proof of (14)

Let us denote $\hat{\mathbf{H}} := g_{\boldsymbol{\phi}}^{H}(\mathbf{X})$ and $\hat{\mathbf{R}}_{\mathbf{b}} := g_{\boldsymbol{\phi}}^{R_b}(\mathbf{X})$. Then,

$$\begin{aligned}
&\mathcal{D}\left(q_{\boldsymbol{\phi}}(\boldsymbol{\zeta}|\mathbf{X})||p_{\boldsymbol{\theta}}(\boldsymbol{\zeta})\right) \\
&= \mathcal{D}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{H},\mathbf{R}_{\mathbf{b}},\mathbf{X})\delta(\mathbf{H}-\hat{\mathbf{H}})\delta(\mathbf{R}_{\mathbf{b}}-\hat{\mathbf{R}}_{\mathbf{b}})||p_{\boldsymbol{\theta}}(\boldsymbol{\zeta})) \quad (25)\\
&= \iiint q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{H},\mathbf{R}_{\mathbf{b}},\mathbf{X})\delta(\mathbf{H}-\hat{\mathbf{H}})\delta(\mathbf{R}_{\mathbf{b}}-\hat{\mathbf{R}}_{\mathbf{b}}) \\
&\quad \cdot \log \frac{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{H},\mathbf{R}_{\mathbf{b}},\mathbf{X})\delta(\mathbf{H}-\hat{\mathbf{H}})\delta(\mathbf{R}_{\mathbf{b}}-\hat{\mathbf{R}}_{\mathbf{b}})}{p_{\boldsymbol{\theta}}(\mathbf{z},\mathbf{H},\mathbf{R}_{\mathbf{b}})} d\mathbf{z}\, d\mathbf{H}\, d\mathbf{R}_{\mathbf{b}} \quad (26)\\
&= \int q_{\boldsymbol{\phi}}(\mathbf{z}|\hat{\mathbf{H}},\hat{\mathbf{R}}_{\mathbf{b}},\mathbf{X}) \log \frac{q_{\boldsymbol{\phi}}(\mathbf{z}|\hat{\mathbf{H}},\hat{\mathbf{R}}_{\mathbf{b}},\mathbf{X})}{p_{\boldsymbol{\theta}}(\mathbf{z},\hat{\mathbf{H}},\hat{\mathbf{R}}_{\mathbf{b}})} d\mathbf{z} \quad (27)\\
&= \int \prod_j q_{\boldsymbol{\phi}}(\mathbf{z}_j|\hat{\mathbf{H}},\hat{\mathbf{R}}_{\mathbf{b}},\mathbf{X}) \Big\{ \sum_j \Big[\log q_{\boldsymbol{\phi}}(\mathbf{z}_j|\hat{\mathbf{H}},\hat{\mathbf{R}}_{\mathbf{b}},\mathbf{X}) \\
&\qquad - \log p_{\boldsymbol{\theta}}(\mathbf{z}_j)\Big] - \log p_{\boldsymbol{\theta}}(\hat{\mathbf{H}}) - \log p_{\boldsymbol{\theta}}(\hat{\mathbf{R}}_{\mathbf{b}}) \Big\} d\mathbf{z} \quad (28)\\
&= \sum_j \int q_{\boldsymbol{\phi}}(\mathbf{z}_j|\hat{\mathbf{H}},\hat{\mathbf{R}}_{\mathbf{b}},\mathbf{X}) \left[\log \frac{q_{\boldsymbol{\phi}}(\mathbf{z}_j|\hat{\mathbf{H}},\hat{\mathbf{R}}_{\mathbf{b}},\mathbf{X})}{p_{\boldsymbol{\theta}}(\mathbf{z}_j)} \right] d\mathbf{z} + C \quad (29)\\
&= \sum_j \mathcal{D}(q_{\boldsymbol{\phi}}(\mathbf{z}_j|\hat{\mathbf{H}},\hat{\mathbf{R}}_{\mathbf{b}},\mathbf{X})||p_{\boldsymbol{\theta}}(\mathbf{z}_j)) + C. \quad (30)
\end{aligned}$$

Here, (25) is due to (10)–(13), (26) is from the definition of the KL divergence, (27) is obtained through the sifting property of the delta function and $\int \delta(t)\log\delta(t)dt = 0$, (28) results by substituting (7) to (27), and the uniform distributions of $p_{\boldsymbol{\theta}}(\mathbf{H})$ and $p_{\boldsymbol{\theta}}(\mathbf{R}_{\mathbf{b}})$ are used to get (29).

References

[1] E. Axell, G. Leus, E. G. Larsson, and H. V. Poor, "Spectrum sensing for cognitive radio: State-of-the-art and recent advances," *IEEE Sig. Process. Mag.*, vol. 29, no. 3, pp. 101–116, May 2012.

[2] O. Duval, A. Punchihewa, F. Gagnon, C. Despins, and V. K. Bhargava, "Blind multi-sources detection and localization for cognitive radio," in *Proc. IEEE GLOBECOM*, New Orleans, LA, Dec. 2008, pp. 1–5.

[3] W. Guibène and D. Slock, "Signal separation and classification algorithm for cognitive radio networks," in *Proc. Int. Symp. Wireless Commun. Syst.*, Paris, France, Aug. 2012, pp. 301–304.

[4] G. Ivković, P. Spasojević, and I. Šeškar, "Single sensor radio scene analysis for packet based radio signals," in *Proc. Info. Theory Appl. Workshop (ITA)*, La Jolla, CA, Jan.-Feb. 2010, pp. 1–10.

[5] S. Grimaldi, A. Mahmood, and M. Gidlund, "Real-time interference identification via supervised learning: Embedding coexistence awareness in IoT devices," *IEEE Access*, vol. 7, pp. 835–850, Jan. 2019.

[6] H. Chen and S.-J. Kim, "Robust RF mixture signal recognition using discriminative dictionary learning," *IEEE Access*, vol. 9, pp. 141107–141120, Oct. 2021.

[7] S. Seki, H. Kameoka, and L. Li, "Investigation and comparison of optimization methods for variational autoencoder-based underdetermined multichannel source separation," in *Proc. IEEE Int. Conf. Acous. Speech Sig. Process.*, Singapore, May 2022, pp. 511–515.

[8] E. Karamatlı, A. T. Cemgil, and S. Kırbız, "Audio source separation using variational autoencoders and weak class supervision," *IEEE Sig. Process. Lett.*, vol. 26, no. 9, pp. 1349–1353, Sep. 2019.

[9] J. Neri, R. Badeau, and P. Depalle, "Unsupervised blind source separation with variational auto-encoders," in *Proc. Eur. Sig. Process. Conf.*, Dublin, Ireland, Aug. 2021, pp. 311–315.

[10] H. Chen and S.-J. Kim, "Unsupervised radio scene analysis using neural expectation maximization," in *Proc. IEEE MILCOM*, Dec. 2022.

[11] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. of the 31st Conf. Neural Info. Process. Syst.*, Long Beach, CA, Dec. 2017.

[12] C. Févotte and J.-F. Cardoso, "Maximum likelihood approach for blind audio source separation using time-frequency Gaussian source models," in *Proc. IEEE Workshop Appl. Sig. Process. Audio Acoust.*, New Paltz, NY, Oct. 2005, pp. 78–81.

[13] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[14] Z. Zhang, Y. Xu, M. Yu, S. Zhang, L. Chen, and D. Yu, "Adl-mvdr: All deep learning mvdr beamformer for target speech separation," in *Proc. IEEE Int. Conf. Acous. Speech Sig. Process.*, Aug. 2021, pp. 6089–6093.

[15] C. P. Burgess, L. Matthey, N. Watters, R. Kabra, I. Higgins, M. Botvinick, and A. Lerchner, "MONet: Unsupervised scene decomposition and representation," *arXiv preprint arXiv:1901.11390*, Jan. 2019.

[16] M. Engelcke, A. R. Kosiorek, O. P. Jones, and I. Posner, "Genesis: Generative scene inference and sampling with object-centric latent representations," *arXiv preprint arXiv:1907.13052*, Nov. 2020.

[17] Y. Wu and K. He, "Group normalization," in *Proc. of the Eur. Conf. Comput. Vis.*, Munich, Germany, Sep. 2018, pp. 3–19.