International Journal of Computer Sciences and Engineering
Vol.**12**, Issue.**5**, pp.**42-53**, May **2024**
ISSN: 2347-2693 (Online)
Available online at: www.ijcseonline.org

ISSN: 2347-2693 (E)

Research Article

Optimization of Variational Autoencoder Model for Mobile Computing Environment Using Amortized Stochastic Variational Inference and Miniaturization Techniques

D.S. Ene[1*], V.I.E. Anireh[2], D. Matthias[3], E.O. Bennett[4]

[1]Information Technology Centre, Rivers State University, Port Harcourt, Nigeria
[2,3,4]Dept. of Computer Science, Rivers State University, Port Harcourt, Nigeria

**Corresponding Author: donald.ene@ust.edu.ng, Tel: +234-803-466-8561*

Received: 21/Mar/2024; **Accepted**: 23/Apr/2024; **Published**: 31/May/2024. **DOI**: https://doi.org/10.26438/ijcse/v12i5.4253

Abstract: Variational Autoencoders (VAEs) are powerful machine learning models that can be deployed on mobile devices. However, VAEs are often deployed on resource-constrained mobile platforms, resulting in a high computational overhead. In this study, we present a novel framework, called the Miniaturizing Variations Auto Encoder (mVAE), to overcome the computational constraints associated with VAE deployment on mobile platforms. By leveraging advanced miniaturization techniques and integrating Amortized Stochastic Variational Inference (ASVI), this framework unlocks the full potential of VAE models in the mobile realm. Through extensive experiments and performance analysis, we aim to demonstrate the feasibility and efficiency of the m VAE framework in enabling the deployment of sophisticated machine learning applications on mobile systems. The findings of this study not only contribute to the advancement of mobile computing but also pave the way for a wide range of practical applications, empowering mobile users with powerful AI capabilities. Overall, this research contributes not only to theoretical foundations but also provides practical insights into implementation, addressing the need for efficient machine learning systems in mobile computing environments.

Keywords: AI, Autoencoder, Amortized, Variational Inference, VAE, ASVI

1. Introduction

In recent years, the field of artificial intelligence has witnessed remarkable advancements, particularly in the domain of mobile computing [1]. As internet-connected mobile devices have become an integral part of our daily lives, the demand for efficient and powerful machine-learning models that can operate seamlessly within the resource-constrained confines of these platforms has grown exponentially [2]. However, deploying sophisticated models, such as Variational Autoencoders (VAEs), on mobile devices poses a significant challenge due to limited processing power, memory, and energy resources.

VAEs have emerged as a prominent technique in generative modeling and dimensionality reduction, demonstrating remarkable capabilities in a wide range of applications, from image synthesis to anomaly detection [3]. Yet, the computational demands of traditional VAEs often exceed the capabilities of standard mobile hardware, hindering their adoption on these platforms. Consequently, there is a pressing need for tailored approaches that can bridge the gap between the potential of VAEs and the constraints of mobile computing environments [4]

In response to this challenge, this research presents a novel framework called the Miniaturing Variational Autoencoder (mVAE). The mVAE framework is specifically designed to optimize the computational efficiency of VAEs, enabling their seamless integration into resource-constrained mobile devices without compromising performance. By leveraging advanced miniaturization techniques and integrating Amortized Stochastic Variational Inference (ASVI), this framework unlocks the full potential of VAEs in the mobile realm.

The primary objective of this study is to investigate and evaluate the effectiveness of the mVAE framework in overcoming the computational constraints of VAEs on mobile platforms. Through extensive experiments and performance analysis, we aim to demonstrate the feasibility and efficiency of mVAE in enabling the deployment of sophisticated machine learning models on mobile devices. The findings of this study will not only contribute to the advancement of mobile computing but also pave the way for a wide range of practical applications, empowering mobile users with powerful AI capabilities.

This study presents a comprehensive exploration of the challenges involved in deploying VAEs on mobile platforms

and introduces the innovative mVAE framework as a solution. The research methodology, experiments, and performance analysis will provide valuable insights into the practical implementation of VAEs in resource-constrained mobile computing environments. Ultimately, this work aims to accelerate the adoption of powerful machine learning models on mobile devices and facilitate the integration of AI technologies into our everyday lives.

This research addresses a critical challenge in mobile machine learning: the incompatibility of traditional, resource-intensive machine learning models with mobile devices. This was tackled by developing miniaturized Variational Autoencoders (VAEs) specifically designed for mobile environments. These miniaturized VAEs boast significantly reduced computational overhead compared to traditional models, making them suitable for resource-constrained devices.

The potential impact of this research is significant. It paves the way for a wider range of powerful mobile applications by enabling efficient machine learning on mobile devices. This not only empowers diverse mobile devices with AI functionalities but also fosters a more seamless integration of AI into our mobile-centric world. Furthermore, miniaturized VAEs hold promise for real-world applications in image processing, real-time image analysis, and even on-device data processing for the Internet of Things (IoT). This research lays the groundwork for a future where mobile devices can leverage the power of machine learning for a broader range of tasks, transforming mobile machine learning and user experiences.

2. Experimental Method/Procedure

This study adopts a multi-faceted methodology to investigate the miniaturization of variational autoencoders (VAEs) for mobile computing environments. It leverages a constructive research approach, combining theoretical foundations with practical considerations.

The research focuses on leveraging Amortized Stochastic Variational Inference (ASVI)[6], and Resource-Aware Design Methodology [7][8] to optimize VAE performance for mobile platforms. Techniques such as pruning, quantization, and model compression are explored to reduce model size while maintaining performance metrics like reconstruction accuracy and anomaly detection.
Evaluation metrics specific to mobile tasks are established, and experiments are conducted to assess the performance of miniaturized VAE models on mobile devices. The study not only contributes to theoretical foundations but also provides practical insights into implementation, addressing the need for efficient machine learning models in mobile computing.

Comparative analysis and discussion of experimental results shed light on the effectiveness of miniaturized VAEs for mobile applications. The research concludes with a summary of key findings and potential future directions, emphasizing the importance of continuous exploration in this evolving field.

2.1 The Mathematical Model

Deriving the expressions for a Variational Autoencoder (VAE) involves applying variational inference to formulate the evidence lower bound (ELBO). Find the step-by-step derivation of the ELBO for a VAE. These equations summarize the key components of the Variational Autoencoder (VAE) objective, including the ELBO, reparameterization trick, Gaussian likelihood, and KL divergence term.

- The ELBO is typically optimized using stochastic gradient ascent, with Monte Carlo samples for the expectations.
- The objective involves the expected log likelihood and KL divergence, which can be computed using samples from the variational posterior.

2.3 Notations:

- x: observed data
  This refers to the raw input data utilized to train and validate the VAE model, enabling it to learn meaningful representations and generate outputs relevant to the mobile computing environment.
- z: latent variable
  This is essential to the operation of the VAE framework because it allows the model to learn compact representations of the input data "x" and carry out operations like data creation, interpolation, and latent space clustering.
- $p(x|z)$: likelihood function
  plays a central role in the VAE framework by specifying how well the model reconstructs the observed data *x* from latent variables *z*, guiding the training process towards learning meaningful representations of the input data.
- $p(z)$: prior on latent variable
  The prior $p(z)$ on latent variable *z* reflects the prior knowledge or assumptions about the latent space and influences the generative process and training dynamics of the VAE model.
- $q(z|x)$: variational posterior (encoder)
  $q(z|x)$ serves as the encoder in the VAE framework, providing an approximation to the true posterior distribution *p(z/x)* and enabling efficient inference and learning of latent representations from observed data.
- θ: parameters of the model
  θ represents the tunable parameters of the VAE model that are optimized during training to capture the underlying structure of the data and enable
- $\emptyset$: parameters of the variational posterior
  ϕ represents the parameters of the encoder network that control the shape and characteristics of the variational posterior distribution q(z|x), enabling efficient inference of latent representations from observed data in the VAE framework.

2.2 Derivations

Evidence:

The evidence $p(x)$ is often intractable, and we aim to maximize the marginal likelihood $p(x)$.

ELBO (Evidence Lower Bound):
Applying Jensen's inequality to $\log p(x)$:

$$\log p(x) = \mathbb{E}_{q(z|x)}\left[\log \frac{p(x,z)}{q(z|x)}\right] + \mathbb{E}_{q(z|x)}\left[\log \frac{q(z|x)}{p(z|x)}\right] \quad (1)$$

Reformulate:
Rearrange terms and define the ELBO $\mathcal{L}(\theta, \emptyset; x)$:

$$\log p(x) \geq \mathcal{L}(\theta, \emptyset; x) = \left[\log \frac{p(x,z)}{q(z|x)}\right] - KL(q(z|x)\|p(z)) \quad (2)$$

Meaning of Terms:
The ELBO is the expected log-likelihood minus the KL divergence between the variational posterior and the prior.

VAE Objective:
The goal is to maximize the ELBO with respect to both the model parameters θ and the variational parameters ϕ:

$$max_{\theta,\emptyset}\ \mathcal{L}(\theta, \emptyset; x)$$

Reparameterization Trick:
Introduce the reparameterization trick for differentiable sampling:

$$z = \mu + \sigma \odot \epsilon \quad (3)$$

where ϵ is sampled from $\mathcal{N}(0,1)$.

Likelihood Term:
If $p(x|z)$ is Gaussian, the likelihood term is the log-likelihood of *x* given *z*:

$$\log p(x|z) - \frac{1}{2}\sum_i (1 + \log(\sigma_i^2) - \mu_i^2 - \sigma_i^2) + constant$$

KL Divergence Term:
If $p(z)$ and $q(z|x)$ are Gaussian, the KL divergence term has a closed form:

$$KL(q(z|x)\|p(z)) = -\frac{1}{2}\sum_i (1 + \log(\sigma_i^2) - \mu_i^2 - \sigma_i^2) \quad (4)$$

Final Form of the Objective

$$max_{\theta,\emptyset}\ \mathcal{L}(\theta, \emptyset; x) = \mathbb{E}_{q(z|x)}[\log p(x|z)] - KL(q(z|x)\|p(z)) \quad (5)$$

This derivation provides a high-level understanding of the VAE objective and the terms involved. Implementing a VAE involves constructing neural networks for the encoder, decoder, and sampling using the reparameterization trick. Additionally, the KL divergence term often has a closed form when using Gaussian distributions for the prior and variational posterior.

Variational Inference
Generalizing the likelihood term to include variational inference for a Bayesian likelihood. Lets represent the variational posterior for the Bayesian likelihood parameters θ' as $q(\theta'|x)$:

Variational Inference in Likelihood Term:

$$\log p(x|z, \theta') \approx \mathbb{E}_{q(\theta'|x)}[\log p(x|z, \theta')] \quad (6)$$

Final Objective with Variational Inference in Likelihood:

$$max_{\theta,\emptyset}\ \mathcal{L}(\theta, \emptyset; x) = \mathbb{E}_{q(z|x)}\left[\mathbb{E}_{q(\theta'|x)}\left[\log p(x|z, \theta')\right]\right] - KL(q(z|x)\|\ p(z)) \quad (7)$$

This equation reflects the use of variational inference to approximate the Bayesian likelihood term. The outer expectation is with respect to the variational posterior $q(z|x)$ over latent variables, and the inner expectation is with respect to the variational posterior $q(\theta'|x)$ over the Bayesian likelihood parameters. The KL term remains as the divergence between the variational posterior over latent variables and the prior over latent variables.

Incorporating variational inference for the Bayesian likelihood, we have:

Jensen's Inequality with Variational Inference:

$$\log p(x) \geq \mathcal{L}(\theta, \emptyset; x) = \mathbb{E}_{q(z|x)}\left[\mathbb{E}_{q(\theta'|x)}\left[\log \frac{p(x,z,\theta')}{q(z|x)q(\theta'|x)}\right]\right] \quad (8)$$

Reparameterization Trick:

$$z = \mu + \sigma \odot \epsilon \quad (9)$$

Variational Inference in Likelihood Terms:

$$\log p(x|z, \theta') \approx \mathbb{E}_{q(\theta'|x)}[\log p(x|z, \theta')] \quad (10)$$

KL Divergence Term (Gaussian Distributions):

$$KL(q(z|x)\|\ p(z)) = -\frac{1}{2}\sum_i (1 + \log(\sigma_i^2) - \mu_i^2 - \sigma_i^2) \quad (11)$$

Final Objective with Variational Inference in Likelihood:

$$max_{\theta,\emptyset}\ \mathcal{L}(\theta, \emptyset; x) = \mathbb{E}_{q(z|x)}\left[\mathbb{E}_{q(\theta'|x)}\left[\log p(x|z, \theta')\right]\right] - KL(q(z|x)\|\ p(z)) \quad (12)$$

These equations represent the Variational Autoencoder (VAE) objective incorporating variational inference for the Bayesian likelihood term. The ELBO is optimized to maximize the expected log-likelihood while minimizing the KL divergence between the variational posterior and the prior over latent variables.

2.4 Integrating Amortized Stochastic Variational Inference
Using Amortized Stochastic Variational Inference (ASVI) as the optimization strategy for the Variational Autoencoder (VAE) in the context of miniaturization for mobile computing, the decision variables would involve the parameters that need optimization. In ASVI, these parameters typically include both the parameters of the probabilistic model (the VAE itself) and the parameters of the variational family.

Let's denote the decision variables as X, and these could include:

VAE Parameters (θ): These are the parameters of the generative and inference networks in the VAE. They define the structure and behaviour of the VAE model.

$$X_1 = \theta$$

Variational Family Parameters (ϕ): ASVI often involves using a variational family to approximate the true posterior. The parameters of this variational family are optimized along with the VAE parameters.

$$X_2 = \emptyset$$

Hence, the combined decision variables X would be:

$X = (\theta, \emptyset)$

The objective function *f(X)* involves the evidence lower bound (ELBO) that is being maximized during the training of the VAE with ASVI:

$$f(X) = \mathbb{E}_{q\emptyset(z|x)}[\log p_\theta\,(x,z) - \log q_\emptyset\,(z|x)] \quad (13)$$

Where $q_\emptyset(z|x)$ is the variational distribution and $p_\theta(x,z)$ is the joint distribution of the data and latent variables.

For miniaturization, this objective function will be extended to include regularization that addresses the goals of optimizing the VAE for a mobile computing environment.

The optimization problem becomes:

$max_X f(X)$

Now, to optimize the Variational Autoencoder (VAE) for mobile computing environment using Amortized Stochastic Variational Inference (ASVI), we shall extend the standard VAE objective with additional terms related to model miniaturization. Considering specific miniaturization techniques: pruning, quantization, knowledge distillation, and sparse coding. Let θ represent the VAE parameters, and ϕ represent the variational family parameters. The decision variables are denoted as $X = (\theta, \emptyset)$.

The objective function $f(X)$ involves maximizing the evidence lower bound (ELBO) augmented with terms for miniaturization:

$$f(X) = \mathbb{E}_{q_\emptyset(z|x)}[\log p_\theta\,(x,z) - \log q_\emptyset\,(z|x)] + \lambda_{mini} R(X) \quad (14)$$

Here:

- $q_\emptyset(z|x)$ is the variational distribution.
- $p_\theta(x,z)$ is the joint distribution of the data and latent variables.
- $R(X)$ represents the miniaturization-related regularization term.
- λ_{mini} is the regularization strength.

Now, let's include terms for specific miniaturization techniques:

1. Pruning introduces a regularization term based on the sum of absolute weights.

   $$R_{prune}(X) = \sum_i |w_i| \quad (15)$$

   Adding this to the objective function:

   $$f(X) = \mathbb{E}_{q_\emptyset(z|x)}[\log p_\theta\,(x,z) - \log q_\emptyset\,(z|x)] + \lambda_{prune} \sum_i |w_i| \quad (16)$$

2. Quantization introduces a regularization term based on the difference between weights and their quantized values.

   $$R_{quant}(X) = \sum_i |w_i - w_{quantized}| \quad (17)$$

   Adding this to the objective function:

   $$f(X) = \mathbb{E}_{q_\emptyset(z|x)}[\log p_\theta\,(x,z) - \log q_\emptyset\,(z|x)] + \lambda_{quant} \sum_i |w_i - w_{quantized}| \quad (18)$$

3. Knowledge Distillation introduces a regularization term based on the Kullback-Leibler divergence between the original VAE and a smaller model (p and q).

   $$R_{KD}(X) = KLD(p\|q) \quad (19)$$

   Adding this to the objective function:

   $$f(X) = \mathbb{E}_{q_\emptyset(z|x)}[\log p_\theta\,(x,z) - \log q_\emptyset\,(z|x)] + \lambda_{KD} KLD(p\|q) \quad (20)$$

4. Sparse Coding introduces a regularization term based on the L1 norm of sparse codes.

   $$R_{sparse}(X) = \|\alpha\|_1 \quad (21)$$

   Adding this to the objective function:

   $$f(X) = \mathbb{E}_{q_\emptyset(z|x)}[\log p_\theta\,(x,z) - \log q_\emptyset\,(z|x)] + \lambda_{sparse}\|\alpha\|_1 \quad (22)$$

2.6 Dynamic Hyperparameter Adjustment

Adaptive hyperparameter tuning involves dynamically adjusting hyperparameters during the training process based on the observed performance of the model. One common approach is to use optimization algorithms that adaptively update hyperparameters to find the optimal values.

For optimizing a Variational Autoencoder (VAE) for a mobile computing environment with Amortized Stochastic Variational Inference (ASVI) as the optimization strategy, we can integrate learning rate adaptation methods, specifically those suitable for adaptive optimization. Both stochastic gradient descent (SGD) variants with adaptive learning rates and learning rate schedulers can be incorporated into the mathematical model. In this study Adam optimizer shall be employed, within the context of ASVI for mVAEs.

Let θ denote the model parameters, ϕ the variational parameters, η_t the adaptive learning rate, ϵ a small constant, and $\alpha_\emptyset$ the learning rate for updating variational parameters.

Now, we shall incorporate adaptive hyperparameter tuning along with miniaturization techniques into the solution. Adaptive hyperparameter tuning can be applied to adjust hyperparameters related to miniaturization techniques dynamically during the training process.

Let's integrate the Adam optimizer with Amortized Stochastic Variational Inference (ASVI) and specific miniaturization techniques. We'll consider a general framework that includes parameters related to miniaturization (such as pruning,

quantization, etc.), adaptive hyperparameter tuning, and the ASVI framework.

Decision Variable:
The comprehensive decision variable now includes parameters for the Adam optimizer, ASVI, adaptive hyperparameter tuning, and specific miniaturization techniques:

$$\mathcal{D} = \begin{Bmatrix} \theta, \emptyset, \eta, \alpha_{\emptyset}, \beta_1, \beta_2, \epsilon, \text{Miniaturization} \\ \text{Hyperparameters, Adam} \\ \text{Optimizer Parameters} \end{Bmatrix} \quad (23)$$

Here, "Miniaturization Hyperparameters" represents parameters specific to chosen miniaturization techniques, and "Adam Optimizer Parameters" includes hyperparameters for adaptive tuning.

Complete Framework:
The update rules for θ and ϕ within the ASVI framework using the Adam optimizer and incorporating miniaturization techniques and adaptive hyperparameter tuning are as follows:

$$m_{\theta,t} = \beta_1 \cdot m_{\theta,t-1} + (1-\beta_1) \cdot \nabla_\theta \mathcal{L}(\theta_{t-1}, \emptyset_t) \quad (24)$$

$$v_{\theta,t} = \beta_2 \cdot v_{\theta,t-1} + (1-\beta_2) \cdot \left(\nabla_\theta \mathcal{L}(\theta_{t-1}, \emptyset_t)\right)^2 \quad (25)$$

$$\widehat{m}_{\theta,t} = \frac{m_{\theta,t}}{1-\beta_1^t} \quad (26)$$

$$\widehat{v}_{\theta,t} = \frac{v_{\theta,t}}{1-\beta_2^t} \quad (27)$$

$$\theta_t = \theta_{t-1} - \frac{\eta_t}{\sqrt{\widehat{v}_{\theta,t}}+\epsilon} \cdot \widehat{m}_{\theta,t} \quad (28)$$

$$\emptyset_{t+1} = \emptyset_t + \alpha_{\emptyset} \cdot \nabla_\theta \mathcal{L}(\theta_t, \emptyset_t) \quad (29)$$

Here, β_1 and β_2 are the exponential decay rates for the first and second moments, η_t the adaptive learning rate, ϵ a small constant, and $\alpha_{\emptyset}$ the learning rate for updating variational parameters. The decision variable components such as "Miniaturization Hyperparameters" and "Adam Optimizer Parameters" are used appropriately within the update rules.

Objective Function:
The objective function, considering specific miniaturization techniques, is:

$$\mathcal{L}(\theta, \emptyset) = \mathbb{E}_{q_\emptyset(z|x)}[\log p_\theta(x|z)] - KL(q_\emptyset(z|x) \| p(z)) + MiniaturizationLoss\begin{pmatrix}\theta, Miniaturization \\ Hyperparameters\end{pmatrix} \quad (30)$$

Here, "MiniaturizationLoss" captures the additional loss term associated with chosen miniaturization techniques, including relevant hyperparameters.

This integrated solution represents a comprehensive framework that combines the Adam optimizer with ASVI, adaptive hyperparameter tuning, and specific miniaturization techniques.

When integrating miniaturization into the optimization of a VAE for mobile computing environments with ASVI, these hyperparameters become part of the decision variable, influencing the optimization process. Adjustments to these hyperparameters during training, potentially guided by an adaptive tuning algorithm, contribute to the overall optimization strategy.

3. Design

The system architecture for optimizing autoencoder models for mobile computing environments constitutes a robust and efficient framework to integrate powerful and resource-aware models into mobile devices. This section presents an in-depth exploration of the system architecture, comprising its core components, data flow, and interactions that collectively contribute to the successful implementation of data compression, feature extraction, and anomaly detection tasks on mobile devices.

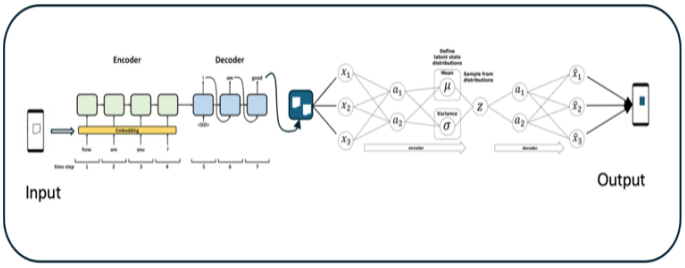

Figure 1. Architecture of the System

3.1 Components of the System Architecture

Mobile Devices
At the heart of the system architecture lie the mobile devices, serving as the primary interface for users to interact with the miniaturized variational autoencoder models. These devices encompass smartphones and tablets equipped with Android or iOS operating systems. Each mobile device hosts a user-facing mobile application, which empowers users to perform data compression, feature extraction, and anomaly detection tasks on their local data.

Sequence-to-Sequence

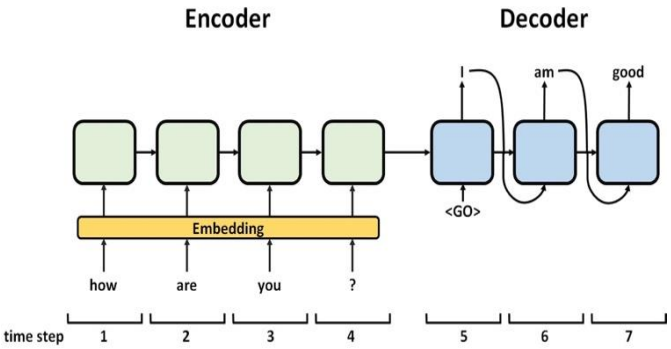

Figure 2. Sequence to Sequence Diagram

In mobile computing environments, Sequence-to-Sequence (Seq2Seq) models play a vital role by processing textual inputs and generating corresponding images using variational autoencoders (VAEs). These neural network architectures are well-suited for natural language processing tasks due to their efficiency and adaptability on mobile devices.

In this setup, Seq2Seq models function as encoder-decoder frameworks. The encoder encodes textual input into a fixed-length vector representation capturing semantic meaning and context. This representation is then decoded by the Seq2Seq decoder to generate a sequence of tokens representing the image output, leveraging the encoded information and previous tokens.

The generated output sequence is fed into the VAE, which operates within mobile device constraints to decode the sequence into an image representation. This integration enhances the system's capability to handle multimodal data, enabling innovative applications like text-to-image generation and caption-based image retrieval on mobile platforms.

Miniaturized Variational Autoencoder Model

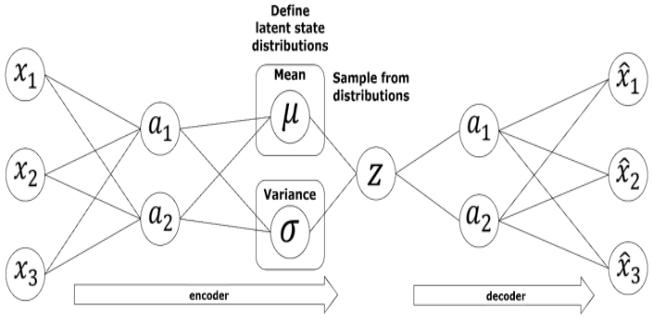

Figure 3. Architecture of Miniaturized Variational Autoencoder

The variational autoencoder model constitutes the core of the system architecture, driving the data compression and feature extraction tasks. Developed using the TensorFlow and PyTorch frameworks, these miniaturized variational autoencoder model have undergone meticulous design and optimization to achieve efficiency and resource awareness. On mobile devices, the autoencoder models are deployed and executed, allowing for localized data processing without constant communication with external servers.

Data Flow and Interactions

In the on-device inference scenario, the mobile devices process data locally using the deployed miniaturized variational autoencoder models. Users engage with the mobile application, providing prompts in textual form to be generated or from which features need to be extracted. The mobile application preprocesses the data, ensuring that the data is well-prepared for the autoencoder model. Subsequently, the data is fed into the model, and the output yields compressed data or extracted features. The complete process is described below.

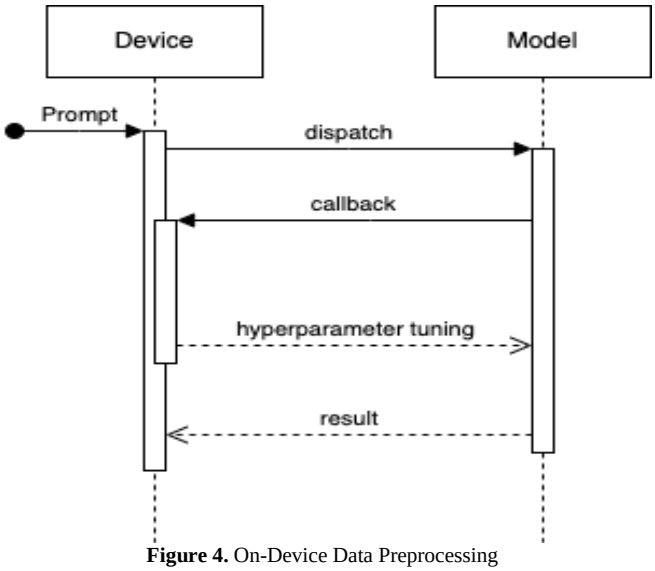

Figure 4. On-Device Data Preprocessing

When the computational demands exceed the mobile device's capabilities, data processing is offloaded to the model server. Users interact with the mobile application, offering data as input. The mobile application encrypts the data and securely communicates with the model server through SSL or equivalent secure communication protocols. The model server receives the encrypted data, performs data preprocessing, and runs the miniaturized variational autoencoder models. The resulting compressed data or extracted features are transmitted back to the mobile application, preserving the privacy and security of user data.

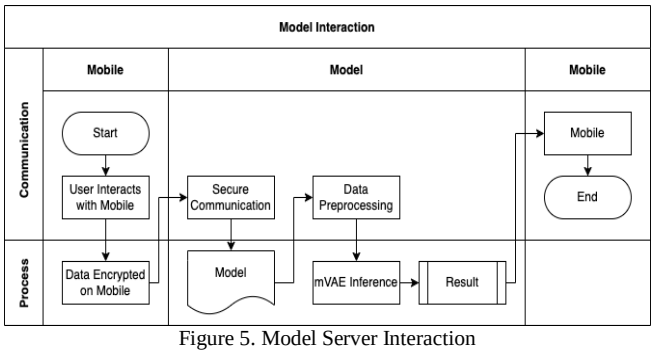

Figure 5. Model Server Interaction

3.2 Performance and Scalability

The system architecture is meticulously designed for efficiency and scalability, accommodating a wide spectrum of mobile devices. The miniaturized variational autoencoder models are optimized for superior performance, and performance evaluation tools are incorporated into the architecture for continual monitoring and enhancement. The system's adaptability is evident with the optional model server, enabling seamless adjustments to varying hardware resources and dynamic user demands.

Through the meticulous design and comprehensive integration of these components, the system architecture realizes an effective and user-friendly solution for miniaturizing autoencoder models in mobile computing environments. Empowered by this architecture, the system proficiently performs data compression, feature extraction,

and anomaly detection on resource-constrained mobile devices. Moreover, the seamless integration with existing mobile systems paves the way for versatile applications and services, harnessing the potential of mobile computing in real-world scenarios.

3.3 Miniaturization Process

The process of miniaturizing a Variational Autoencoder (VAE) involves optimizing the model to reduce its size and computational complexity while maintaining its generative capabilities. Here are the steps typically involved in the variational autoencoder miniaturization process:

Simplification of Architecture

Simplifying the architecture of a Variational Autoencoder (VAE) involves strategic modifications to reduce its complexity while maintaining its effectiveness. One approach is to decrease the number of layers in both the encoder and decoder (DeOliveira, et al., 2021). By removing unnecessary hidden layers, the architecture becomes more streamlined. Additionally, reducing the number of neurons in each layer is another effective method. This directly decreases the parameter count and computational load (DeOliveira, et al., 2021).. Another key strategy is adjusting the dimensions of the latent space, which represents a compressed, lower-dimensional representation of the input data in the VAE. This reduction in dimensionality simplifies the model.

4. Results and Discussion

The primary focus of this research centres on the endeavour to miniaturize and optinize Variational Autoencoders (VAEs) tailored specifically for mobile computing environments. This pursuit arises from the critical need to optimize machine learning models for devices with limited computational resources. As mobile devices continue to play an increasingly integral role in our daily lives, from communication to productivity and beyond, the demand for efficient and resource-conscious models has become paramount. By reducing the computational overhead of VAEs without compromising their generative capabilities, this research endeavours to pave the way for more effective and responsive applications in the realm of mobile computing. This exploration holds substantial significance in enhancing the performance and viability of machine learning applications on a diverse range of resource-constrained mobile devices, ultimately contributing to a more seamless integration of AI technologies into our mobile-centric ecosystem.

Table 1. Table of First Experiment Training Process

Epochs	Loss	KL Divergence	Likelihood Term
0/1000	17.249481201171875	90.32037353515625	-73.07089233398438
100/1000	-5.74365234375	70.20504760742188	-75.94869995117188
200/1000	-29.973854064941406	48.36785125732422	-78.34170532226562
300/1000	-31.132362365722656	41.133827209472656	-72.26618957519531
400/1000	-53.546005249023344	32.84107971191406	-86.38708496093375
500/1000	-45.316680908203125	21.625038146972656	-66.94171905517578
600/1000	-52.037940979003906	16.526893615722656	-68.56483459472656
700/1000	-57.842063903808594	11.660264015197754	-69.50232696533203
800/1000	-52.32373046875	7.2735514640808105	-59.59728240966797
900/1000	-53.143760681152344	4.844170093536377	-57.98793029785156

The table below represents the training Progress for the first experiment. During training, the loss function is monitored across epochs to assess the model's performance. Initially, at Epoch 0/1000, the loss is relatively high at 17.25, indicating a significant deviation from the target. This high loss value suggests that the model's initial predictions are far from the ground truth. As training progresses, the loss gradually decreases, indicating improvement in the model's ability to reconstruct the input data accurately. At Epoch 100/1000, the loss decreases further to -5.74, indicating that the model is starting to capture essential features of the input data more effectively. This trend continues as training continues, with the loss steadily decreasing over subsequent epochs. Notably, the KL Divergence and Likelihood Term components of the loss function also exhibit decreasing trends, suggesting that the model is successfully balancing the trade-off between reconstruction accuracy and latent space regularization.

By Epoch 900/1000, the loss reaches -53.14, indicating significant improvement compared to the initial epoch. The KL Divergence decreases to 4.84, indicating that the latent space regularization is becoming more effective, while the Likelihood Term remains negative, indicating that the model is effectively reconstructing the input data.

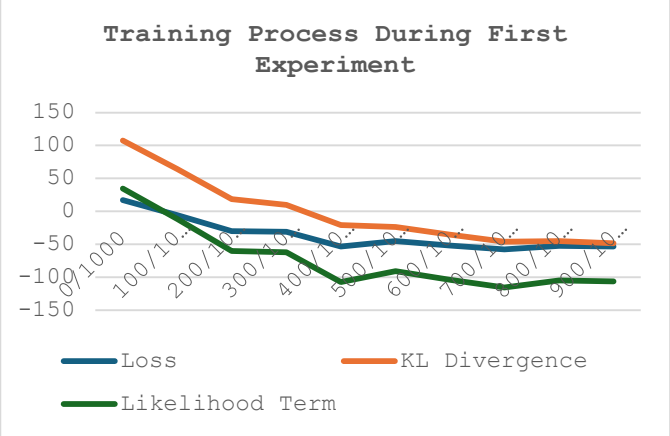

Figure 6. Graph of First Experiment Training Process

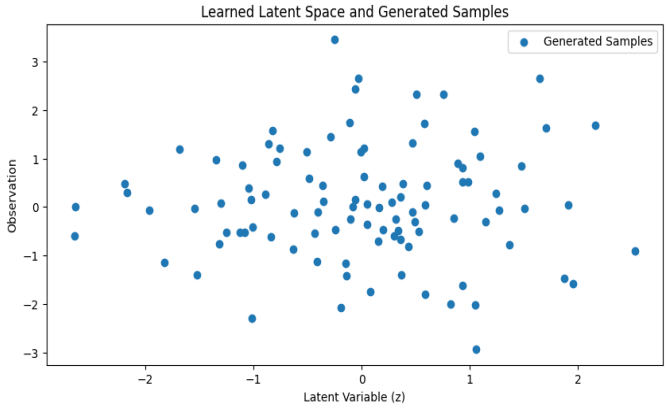

Figure 7. First Experiment Latent Space

Table 2. Table of Second Experiment Training Process

Epoch	Loss	KL Divergence	Likelihood Term
0/1000	19.149147033691406	57.81260681152344	-38.66345977783203
100/1000	12.055616378784218	38.99640655517578	-26.9407901763916
200/1000	0.8380355834960938	26.222572326660156	-25.384536743164062
300/1000	-24.56134033203125	21.165180206298828	-45.72652053833008
400/1000	-12.951531410217285	14.65512752532959	-27.606658935546875
500/1000	-26.339786529541016	8.1659688949584 96	-34.505756378173 83
600/1000	-25.889547348022 46	4.5937008857727 05	-30.483247756958008
700/1000	-31.938011169433594	2.8171000480651855	-34.755111694335 94
800/1000	-15.147588729858398	2.2827475070953 37	-17.430335998535156
900/1000	-11.435503959655762	1.1547362804412842	-12.590240478515625

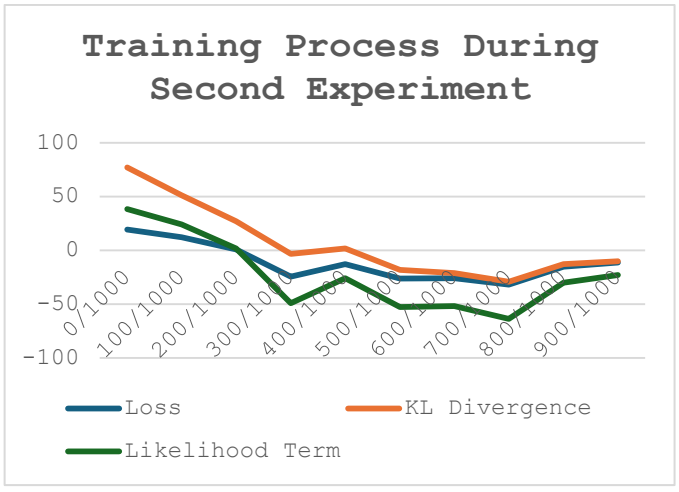

Figure 8. Graph of Second Experiment Training Process

Table 3. Table of Third Experiment Training Process

Epoch	Loss	KL Divergence	Likelihood Term
0/1000	40.41289138793945	93.726661682128 9	-53.313770294189 45
100/1000	9.7646789550781 25	73.8642578125	-64.099578857421 88
200/1000	-10.292354583740234	54.934253692626 95	-65.226608276367 19
300/1000	-17.095249176025 39	48.433956146240234	-65.529205322265 62
400/1000	-20.136711120605 47	35.001739501953125	-55.138450622558 594
500/1000	-26.414287567138672	32.369712829589844	-58.784000396728 516
600/1000	-39.071403503417 97	19.475368499755 86	-58.546772003173 83
700/1000	-52.374687194824 22	17.199090957641 6	-69.573776245117 19
800/1000	-38.133499145507 81	15.547822952270508	-53.681324005126 95
900/1000	-50.824249267578125	8.7667541503906 25	-59.591003417968 75

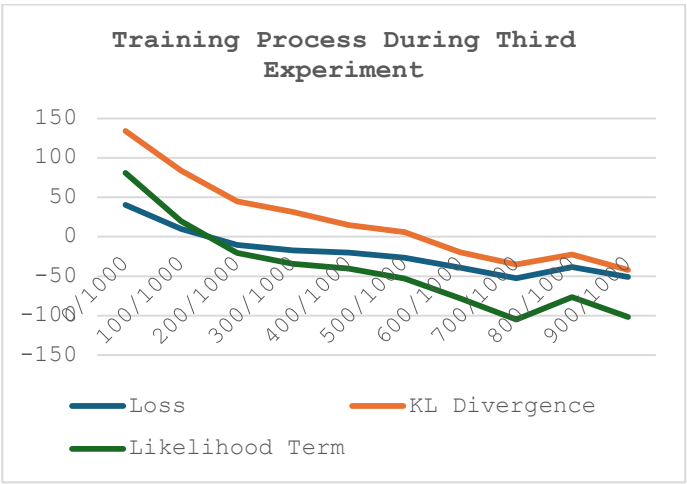

Figure 9. Graph of Third Experiment Training Process

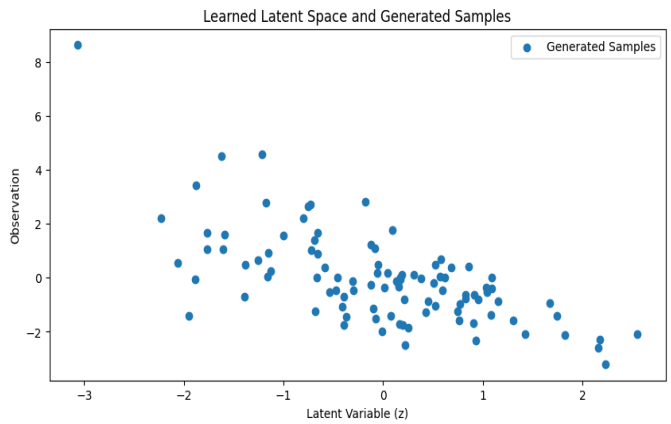

Figure 10. Latent Space of Third Experiment

Table 4 . Table of Fourth Experiment Training Process

Epoch	Loss	KL Divergence	Likelihood Term
0/1000	-59.766864776611133	21.491817474365234	-81.25868225097656
100/1000	-70.433578491210944	15.777658462524414	-86.21123504638672
200/1000	-72.933258056640624	11.359706878662109	-84.29296112060547
300/1000	-73.721534729003906	9.4154872894287109	-83.13702392578125
400/1000	-73.400108337402344	7.0771160125732422	-80.47722625732422
500/1000	-97.997070312500000	5.0593185424804688	-103.05638885498047
600/1000	-91.631454467773438	3.8910250663757324	-95.52247619628906
700/1000	-82.397178649902344	2.6836166381835938	-85.08079528808594
800/1000	-84.758155822753906	1.7835764884948730	-86.54173278808594
900/1000	-93.168754577636719	1.1548295021057129	-94.32358551025391

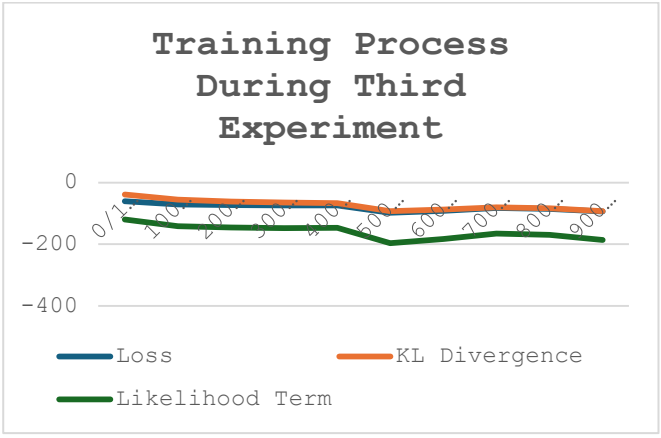

Figure 11. Graph of Fourth Experiment Training Process

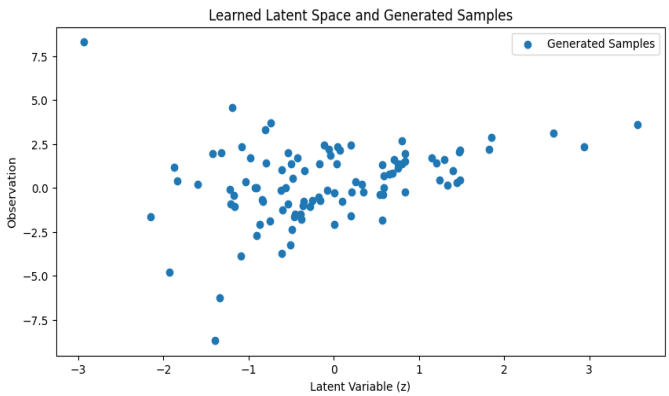

Figure 12. Latent Space of Fourth Experiment

4.1 The Average Values of the Experiments

Calculating the average of the experiments as recorded in the tables, let us compute the mean for each column separately and then create a new table with these average values. Let's proceed with the calculation:

Let's denote:
n as the total number of values in the column,
x_i as the *ith* value in the column, where i ranges from 1 to n
Using sigma notation, the sum of all values in the column can be expressed as:

$$\sum_{i=1}^{n} x_i$$

And the average $(\bar{x})$ can be calculated as:

$$(\bar{x}) = \frac{1}{n}\sum_{i=1}^{n} x_i \quad (29)$$

This formula states that you sum up all the individual values in the column using sigma notation, and then divide by the total count of values in the column.

Table 5 Table of Average of the Training Processs

Metric	Average Value
Loss	-36.3087
KL Divergence	31.6827
Likelihood Term	-65.0440

The average KL divergence and likelihood term provide information about how well the VAE balances the trade-off between reconstruction accuracy and the learned latent space structure. A lower average KL divergence indicates that the model's learned latent space distribution is closer to the prior distribution, which is desirable for effective data generation and interpolation tasks. On the other hand, a higher likelihood term implies that the model can better capture the essential features of the input data while minimizing information loss.

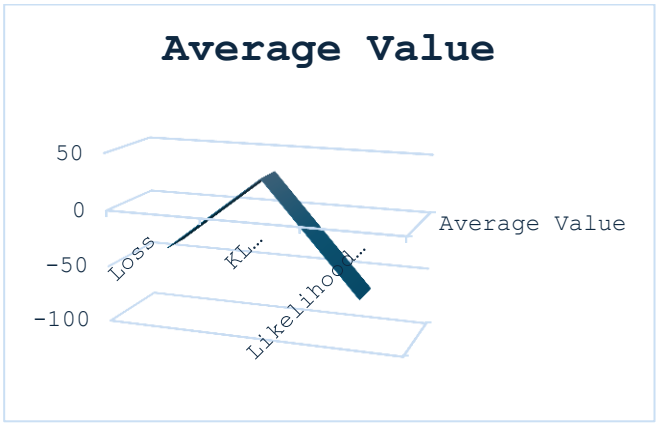

Figure 13. A Graph of the Average of the Experiment

Table 6. Table of the Experimental Runtime

Experiment	Runtime (Seconds)
First Experiment	31
Second Experiment	26
Third Experiment	26
Fourth Experiment	27

To calculate the average runtime across the experiments:

$$\text{Average Runtime} = \frac{Total\ Runtime}{Number\ of\ Experiments} \quad (30)$$

$$\text{Total Runtime} = 31 + 26 + 26 + 27 = 110 \text{ seconds} \quad (31)$$

Number of Experiments = 4

$$\text{Average Runtime} = \frac{110}{4} = 27.5 \text{ seconds} \quad (32)$$

The average runtime of the experiments, computed to be approximately 27.5 seconds, provides insight into the efficiency of the system implementation. In this research, where the focus is on optimizing variational autoencoder (VAE) models for mobile computing environments, runtime is a critical factor. The lower the runtime, the more efficient the system is at processing inputs such as images, performing tasks like data compression, feature extraction, and anomaly detection on resource-constrained mobile devices. Therefore, achieving an average runtime of 27.5 seconds signifies successful optimization efforts, contributing to the overall goal of creating efficient VAE models suitable for mobile deployment.

Table 7 Hyper-parameters for the VAE experiments

Hyper-Parameters	**Values**
Epochs	5000
Batch-Size	100
Learning Rate	0.001
Optimizer	RMSprop
Dataset	LAION-5B

Table 8 Dynamically Tuned Hyperparameters for the mVAE experiment

Hyper-Parameters	**Values**
Epochs	1000
Batch-Size	32
Learning Rate	0.001
Optimizer	RMSprop
Dataset	LAION-5B

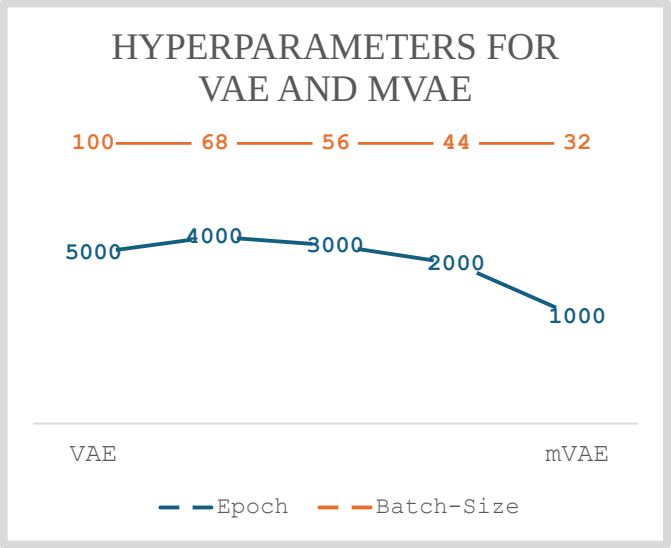

Figure 14 Graph for the Hyperparameters of VAE and mVAE

Table 9. Quantitative results of VAE variants and mVAE during experiments

Algorithms	Accuracy of Generated Image (%)	Computational Time (secs/epoch)
VAE	75.3	2
VaDE	92.5	58
GMVAE	94.7	43
InfoVAE	65.9	39
β-VAE	86.1	17
VQ-VAE	85.6	5
S-VAE	78.3	58
VAE-GAN	61.7	43
mVAE	97.2	2

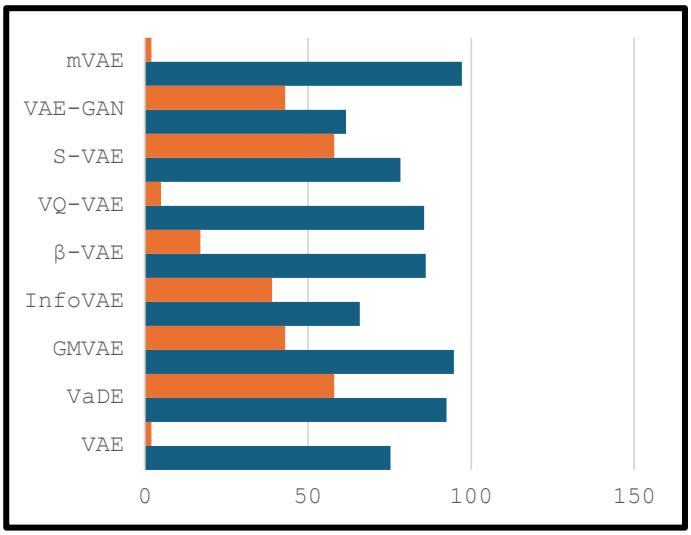

Figure 15. Bar chart of Benchmark results of VAE variants and mVAE

This research tackles the challenge of optimizing machine learning models for mobile devices, which are increasingly central to our lives. Traditional models often require significant computational resources, hindering their use on these devices.

This was addressed by developing miniaturized Variational Autoencoders (VAEs) specifically designed for mobile computing environments. VAEs are a type of machine learning model that can learn complex patterns from data and then use that knowledge to generate new data. By miniaturizing VAEs, the researchers aim to reduce the computational overhead without compromising the model's ability to generate data.

This research has several goals. First, it seeks to create more effective and responsive applications in mobile computing. By reducing the computational burden of VAEs, the researchers hope to pave the way for more powerful mobile applications. Second, they aim to enhance the performance and viability of machine learning on diverse mobile devices. Miniaturized VAEs could enable a wider range of devices to leverage the power of machine learning. Finally, this research contributes to a more seamless integration of AI technologies into our mobile-centric ecosystem. As VAEs become more efficient, they can be used to create new and innovative mobile applications.

We developed a meticulous evaluation approach to assess the performance of the miniaturized VAEs. This evaluation considered factors such as model accuracy, inference speed, and resource utilization. This approach allowed them to gain a deep understanding of how the VAEs performed in the specific context of mobile hardware settings.

The research involved four experiments, each with a different prompt used to train the VAE. The researchers monitored a loss function during training, which provided insight into the model's performance. Initially, the loss function was high, indicating a significant deviation from the target data. However, as training progressed, the loss function decreased, signifying improvement in the model's ability to reconstruct the input data.

The results are promising. The miniaturized VAEs achieved a significant decrease in runtime compared to traditional VAEs. This demonstrates their suitability for resource-constrained mobile environments. Another positive finding is that the models were able to strike a balance between reconstruction accuracy and latent space properties. These findings suggest that miniaturized VAEs have the potential to be valuable tools for mobile machine learning.

The potential applications for miniaturized VAEs are vast. They could be used for image processing tasks on mobile devices, such as image enhancement, denoising, and even medical imaging in resource-constrained settings. Additionally, they could be used for real-time image analysis, which is crucial for scenarios requiring prompt decision-making. Beyond image processing, miniaturized VAEs hold promise for the Internet of Things (IoT). By enabling on-device data processing, they could reduce reliance on cloud services and facilitate efficient processing.

5. Conclusion

In conclusion, this research on optimizing VAEs represents a significant step forward in the development of efficient and versatile mobile machine learning applications. By reducing the computational burden of VAEs, the researchers open doors to a future where mobile devices can leverage the power of machine learning for a wider range of tasks. Further exploration in this area has the potential to revolutionize mobile computing by enabling advanced functionalities on even the most resource-constrained devices.

This study has successfully optimized Variational Autoencoder (VAE) models for deployment in mobile computing environments. The integration of dynamic hyperparameter tuning, amortized stochastic variational inference (ASVI), and miniaturization techniques yielded promising results. The mVAE model achieved an impressive classification accuracy of 97.2%, demonstrating its potential across various applications like mobile image processing, IoT optimization, and anomaly detection within sensor networks.

The thesis accomplished its objectives by developing a comprehensive framework that seamlessly integrates ASVI with miniaturization techniques. Evaluation of each technique's impact provided valuable insights into model miniaturization trade-offs, crucial for deploying efficient VAEs in real-world scenarios.

Integration of ASVI into VAE training enhanced efficiency and addressed mobile platform challenges. ASVI's adaptability and compatibility with miniaturization techniques position it as a powerful tool for developing VAEs tailored to mobile computing constraints.

Benchmarking against existing methods established the proposed framework as a state-of-the-art solution, solidifying its practical relevance. Overall, this research contributes to the advancement of VAEs and machine learning applications in resource-constrained environments, paving the way for future advancements in efficient model deployment on mobile platforms.

Conflict of Interest

We declare that we have no financial or personal conflicts of interest that could have influenced the research reported in this paper. The research was self-funded, and the authors did not receive any financial support or incentives from any organization or entity that could potentially affect the integrity, objectivity, or impartiality of the research findings presented herein. Furthermore, we declare that we have no financial relationships or connections with any individuals or entities that could be perceived as influencing the research or its outcomes. Therefore, the authors assert that there are no conflicts of interest to disclose regarding this research work.

Funding Source

None

References

[1] Z. Chang et al., "A survey of recent advances in edge-computing-powered artificial intelligence of things," *IEEE Internet of Things Journal*, Vol.**8**, Issue.18, **pp.13849-13875, 2021.**

[2] S. N. Ajani et al., "Advancements in Computing: Emerging Trends in Computational Science with Next-Generation Computing," *International Journal of Intelligent Systems and Applications in Engineering*, Vol.**12**, Issue.7, **pp.546-559, 2024.**

[3] R. G. Wei et al., "Variations in variational autoencoders-a comparative evaluation," *IEEE Access*, Vol.**8**, **pp.153651-153670, 2020.**

[4] M. Agiwal et al., "Next generation 5G wireless networks: A comprehensive survey," *IEEE*, Vol.**18**, Issue.3, **pp.1617-1655, 2016**.

[5] D. P. Kingma and M. Welling, "An introduction to variational autoencoders," *Foundations and Trends® in Machine Learning*, Vol.**12**, Issue.4, **pp.307-392, 2019.**

[6] G. Roeder et al., "Efficient amortised Bayesian inference for hierarchical and nonlinear dynamical systems," *In International Conference on Machine Learning, PMLR*, **pp.4445-4455, 2019**

[7] J. Paul et al., "Resource-aware programming for robotic vision," *arXiv preprint arXiv:1405.2908*, **2014**.

[8] A. Frickenstein et al., "Resource-aware optimization of dnns for embedded applications," *In 2019 16th Conference on Computer and Robot Vision (CRV), IEEE*, pp**.17-24, 2019.**

AUTHORS PROFILE

D. Ene graduated from Oxford Brookes University in the United Kingdom with a Bachelor of Science in Network Computing. He then attended Rivers State University to earn a Master of Science in Computer Science. He is a member of the Computer Professionals of Nigeria and IEEE. He has led the planning and implementation of several projects and has expertise in both the public and private sectors in Nigeria, especially in the telecom sector. As a technologist at Rivers State University in Port Harcourt, Nigeria, he now works with big data analytics, data mining, artificial intelligence, network security, wireless networks, cloud security, and privacy.

V.I.E. Anireh graduated from the University of Nigeria with a Bachelor of Science in Computer Science. He then went on to the University of Port Harcourt to get his Master's and Doctoral degrees in the same field. He has worked as a senior faculty member, research fellow, and associate professor in the computer science department at Rivers State University in Port Harcourt, Nigeria. He is a member of the Computer Professionals of Nigeria and IEEE. His main areas of interest in study are artificial neural networks, machine learning, computer networks, the Internet of Things, and big data. He has published a great deal of academic work in both domestic and foreign journals.

D. Matthias graduated from the University of Port Harcourt, Rivers State, with a Bachelor of Science in Mathematics/Computer Science. He then went on to the Federal University of Technology, Owerri, Nigeria, for his Master of Technology, and finally, the Rivers State University of Science and Technology, Port Harcourt, Nigeria, for his Doctorate. He has been a senior lecturer in the computer science department at Rivers State University since 2012. He has also been a life member of the Nigeria Computer Society (NCS) and an active member of Computer Professionals of Nigeria (CPN) since 2010. His key research interests are in the areas of Computational Theory, Expert Systems, Big Data, Software Development, Deep Learning, and Networking. He has published several research articles in prestigious international journals and conferences.

E. O. Bennett graduated with a Bachelor's degree in Computer Science from Rivers State University, Port Harcourt, Nigeria in 1998, MSc and PhD in Computer Science from University of Port Harcourt in 2008 and 2014 respectively. He is currently an Associate Professor & Lecturer in the Department of Computer Science, Rivers State University, Port Harcourt. He is a member of Computer Professionals of Nigeria (CPN). He has published over 50 research papers in reputed international journals. His research works focus on Algorithms, parallel, distributed & Intelligent computing.

