6-29-2024

Hierarchical Quantized Autoencoders: Using Hierarchical Models for Data Compression Across Multiple Domains

Armani Lorenzo Rodriguez
Rowan University

HIERARCHICAL QUANTIZED AUTOENCODERS: USING HIERARCHICAL MODELS FOR DATA COMPRESSION ACROSS MULTIPLE DOMAINS

by

Armani Lorenzo Rodriguez

A Thesis

Submitted to the
Department of Computer Science
College of Science and Mathematics
In partial fulfillment of the requirement
For the degree of
Master of Science in Computer Science
at
Rowan University
May 2, 2024

Thesis Chair: Silvija Kokalj-Filipovic, Ph.D., Professor, Department of Computer Science

Committee Members:
Anthony Breitzman, Ph.D., Professor, Department of Computer Science
Patrick McKee, MS, Senior Lecturer, Department of Computer Science

Dedication

This thesis is dedicated to my wonderful mother, Elizabeth. Her love and support is what kept me going on this journey.

Acknowledgment

I would like to express my gratitude and appreciation to Professor Silvija Kokalj-Filipovic for her unwavering guidance throughout this project. Working with her I have learned an incredible amount of technical and professional skills that I will take with me throughout my professional career.

Abstract

Armani Rodriguez
HIERARCHICAL QUANTIZED AUTOENCODERS: USING HIERARCHICAL MODELS FOR DATA COMPRESSION ACORSS MULTIPLE DOMAINS
2023-2024
Silvija Kokalj-Filipovic, Ph.D.
Master of Science in Computer Science

In the era of vast data processing and transmission, sending data over a channel for downstream operations is a very common occurrence. The bandwidth of this data channel acts as a limiting factor in this operation, capping the amount of data that can be sent over a time period. Therefore, in addition to pursuing advancements in networking technology, there exists a need for more efficient means of data compression. Learned compression is the application of machine learning models to the data compression problem, and in this study, we leverage the ability of neural networks to learn the underlying structure of the training data to perform more informed compression, achieving a greater compression ratio than algorithmic data compression. Specifically, we analyze the efficacy of a model known as the Hierarchical Quantized Autoencoder (HQA) for lossy data compression across various datasets. This model adds a novel hierarchical architecture to the quantized auto encoder, the current standard for learned data compression, which not only allows for a higher compression ratio but adds flexibility to the model as the desired compression ratio can be set at inference time. We evaluate the performance of this model across different image datasets and propose a new model of the same structure as HQA but with a modified architecture suited to compress radio frequency (RF) data. We find that using hierarchical models, we can achieve a high compression ratio with minimal sacrifice to the performance of the downstream task.

Table of Contents

Table Of Contents (Continued)

List Of Figures

Figure Page

List Of Tables

Figure Page

Chapter 1

Introduction

Machine learning refers to the development of computer algorithms that can learn from data, rather than following explicit instructions. Very often we come across tasks that are trivial to us humans due to our innate intuition but are overwhelmingly difficult to complete from an explicit algorithmic approach. Examples of these problems include image classification, audio to text transcription, and language translation. These problems all deal with types of data catered for human perception such as images, sound, and text. For a computer to effectively use and process said datatypes, it must be able to deeply understand the underlying structure and patterns within the data. The neural network, machine learning model motivated by the biology of the brain, accomplishes this through being trained on an ideally large set of data known as the training set. Through this training process, the neural network will learn the underlying patterns in the data and can apply what it has learned to novel data unseen in the training set.

1.1 Problem Statement

The major recent developments in machine learning combined with the effectiveness of machine learning models being positively correlated with the amount of training data has made data more valuable than ever before. The sheer amount of data being collected has brought about the popularity of distributed systems where data is collected and then transported elsewhere over a data channel for downstream tasks. Whether this data channel be a wireless network channel or a physical cable, it will have a bandwidth, or a maximum transfer rate. Thus, there are two directions one can go in improving the flow of data, that being increasing the bandwidth, or decreasing the size of

the data. In this research, we focus in the latter while considering the security implications by studying our model's robustness to adversarial attacks [1].

1.2 Data Compression and Learned Compression

Data compression is the process of finding a new representation of data such that it can be represented in a fewer number of bits. Data compression can be lossless, in which zero information is lost and the original data can be reconstructed from the compressed representation with zero distortion. Conversely, lossy compression allows for minimal distortion to achieve a greater compression ratio than lossless compression.

Learned compression refers to the use of machine learning models to achieve lossy data compression. The motivation behind learned compression is to leverage the ability of neural networks to learn the underlying patterns behind data to achieve a compression codec that is more data informed, and thus in theory more effective than algorithmic compression.

Rate distortion theory, which describes the mathematical foundations of lossy compression, describes a distortion function. This distortion function $d(x, \hat{x})$ accepts as input the original datapoint x and the datapoint reconstructed from the compressed representation $\hat{x}$. The output of $d(x, \hat{x})$ is a measure of distortion or difference between the original and reconstructed datapoints. For lossless compression, the output of d is always zero since lossless compression has zero distortion by definition. For lossy compression, our goal is to minimize d while maximizing the compression ratio which represents the ratio between the original size in bits and the compressed size in bits. We will use a distortion function, specifically the squared-error distortion function, as our

criterion for training. However, we will not use distortion as our criterion for model selection, instead we focus on the performance of our downstream operation.

1.3 The Classification Problem

Classification is a recurring problem in data science, in which we seek to label input data as belonging to a certain class. For example, a bank may want to classify whether a transaction is fraudulent or not given its properties. All our input data in this research will have some set of classes associated with it, and our downstream operation will be classifying the data. Thus, we will measure how well our compression model performs based on how accurately the reconstructed data can be classified compared to the original.

1.4 Adversarial Attacks

Deep classifiers are subject to a vulnerability known as an adversarial attack. Such attacks are orchestrated by crafting an adversarial example, or a piece of data specifically crafted to perturb the model and produce an inaccurate output. The process of creating these adversarial inputs is taking a normal input datapoint and adding a certain amount of adversarial noise scaled by a small value ε such that there is a minimal perceptible difference between the original and adversarial datapoints [2].

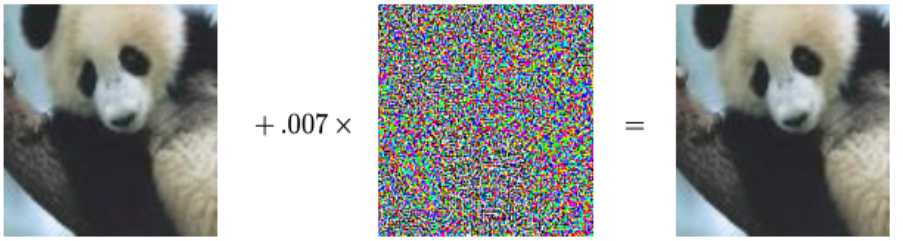
The fast gradient sign method (FGSM), a vastly popular method for creating adversarial noise, works by calculating the gradient of the loss function $J(\theta, x, y)$ with respect to each input feature to create noise which maximizes the loss. The formula for creating an adversarial example from an arbitrary input x is given by Equation 1.

$$x_{adv} = x + \varepsilon \times \text{sign}\left(\nabla_x J(\theta, x, y)\right) \quad (1)$$

We see that generating an adversarial example requires the attacker to have knowledge of the model parameters θ, the loss function J, and the true label of x denoted y. For now, this leaves defense against adversarial example attacks as an information security problem while machine learning researchers study ways of constructing deep classifiers that are robust to said attacks.

Figure 1

An Example of Creating Adversarial Data Using FGSM. [2]



1.5 Significance

The use of a deep compression model can make a significant difference in the speed in which data is transferred. In our paper we focus on compressing two different data types, that being image data and radio frequency (RF) data. Both data types are extremely diverse and compression of these datatypes for faster transfer has many use cases.

1.5.1 Image Data

Most people are familiar with image data and interact with digital images daily. Images, especially high-resolution images, are rather expensive to transfer over a data link since every single pixel must be represented. A learned compression codec with a

high compression ratio could improve loading speeds of webpages, reducing the load on the server and reducing the cost of operating a website serving image data which receives many requests per second. Additionally, machine learning powered applications where images are sent from a user's device to an external server for processing before the result is then pushed back to the device are becoming exceedingly popular. These applications often suffer from poor performance when the user does not have a fast and stable internet connection and are often expensive to run due to the networking capabilities needed to support such a large stream of input data. This compression scheme could save precious bandwidth reducing operations costs and improving user experience.

1.5.2 Radio Frequency (RF) data

Waveform data is an extremely versatile datatype as any piece of data can be interpreted as a signal. A prominent example of the vast transmission of RF data is next generation cellular concepts such as Artificial Intelligence Radio Access Networks (AI-RAN) [3] which uses AI for real time analysis of traffic data and network load. Waveform data can also represent audio signals, underground seismic activity, or brain waves (EEG-signals) and other biological signals. Research into learned compression regarding time series data such as radio frequency data is quite scarce due to most learned compression research being focused on image data.

RF data is usually found at the lowest layers of network models and is the physical representation of information being transmitted over network channels. Not only is compressing RF data extremely versatile as any data can be converted to RF data, but compressing information at the physical network layer can help optimize every wireless channel faster and more effectively.

A major breakthrough in the generation of new vision content (images and video, even movies) has been recently achieved by so called diffusion models. The most successful diffusion models use vector quantized variational autoencoders as the core of their diffusion process primarily because it lowers the computational complexity and allows better resolution. For those models to be leveraged in the waveform domain, it is important to explore the utilization of vector quantized variational autoencoders in that domain.

Chapter 2

Prior Work

This chapter provides an overview of the prior work in the field of learned compression as well as prior research on hierarchical autoencoders.

2.1 Autoencoder Based Approach

We will discuss the specifics of the theory of autoencoders in Chapter 3. Autoencoders use an unsupervised learning approach to learn an efficient representation of the input data in a lower dimensional latent space. Variants of autoencoders such as the variational auto encoder, which maps input data to a probability distribution, have found success with both generation of deepfakes and learned compression. The vector quantized variational autoencoder (VQ-VAE), another generative autoencoder variant, has also been successful in the field of learned compression [3].

2.2 Transform-Based Approach

Transform based learned compression involves transforming the input data into an intermediary domain that is easier and more efficient to compress. Machine learning models such as deep neural networks can learn these transforms from the data which leads to data informed compression. Examples of the transform-based approach include learned image compression using convolutional neural networks and learned audio compression using time-frequency transforms. Perhaps the most well-known form of transform based compression is JPEG, where images are treated as signals and the fast Fourier transform algorithm is used to compress images in this codec.

2.3 Attention Based Approach

Attention based models, such as the transformer model introduced by Vaswani et. al. [4], have impressive sequential prediction power. Thus, we can harness this power to achieve learned compression by keeping only the most relevant parts of the data, while letting the attention-based model fill in the blanks.

2.4 Generative Adversarial Network Approach

Generative adversarial networks use two models, a generator and a discriminator, to achieve the generation of deepfakes. There is a recurring theme of repurposing generative models for learned compression. Generative models usually learn an efficient underlying representation of the input data while having the training objective of creating high quality deepfakes that are indistinguishable from original data. This ability is one we can often leverage for data compression.

2.5 HQA

The HQA model for compressing images was created and studied by Williams et. al. [5] for the purposes of generation and data compression. Their paper studied HQA from an information theory perspective where rate distortion metrics were used to measure the effectiveness of HQA as a lossy compression codec.

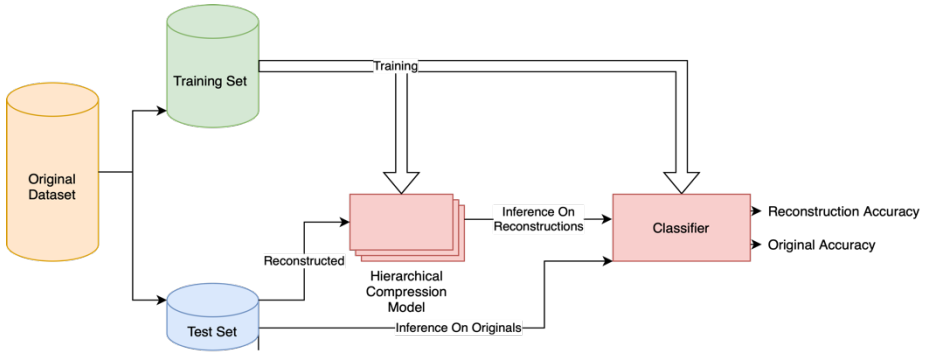
Chapter 3

Methodology

We seek to measure the efficacy of our model in a variety of domains; thus, we will perform our experimentation on three different datasets. Each dataset will require modifications to the models pertinent to the complexity and topology of the data; however, the general process remains constant. We first split our dataset into a training set and a test set. We then train our hierarchical learned compression model and our deep classifier on our training set. Our experiment begins by notating the accuracy score achieved by the deep classifier on the test set. We compare with the accuracy score achieved by the deep classifier on the test set transformed by being compressed and then reconstructed by the compression model. For our image datasets, we will compare HQA against JPEG, a widely used lossy compression scheme for image data.

Figure 2

An Illustration of Our Compression Experiment.

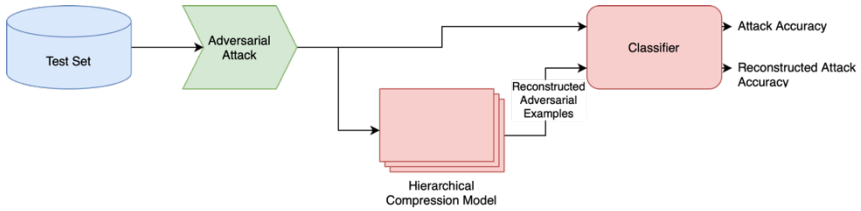


We then test our hypothesis regarding the resiliency of our proposed compression scheme to an adversarial example attack attempted on the downstream classification task

by transforming the test set to adversarial examples before comparing the accuracies of the classifier on the original attacked images with the reconstructed attacked images.

Figure 3

The Process of Testing Our Models' Resiliency to Adversarial Attacks.



3.1 The Model Architecture

The model architecture we will be using for compression is known as a hierarchical quantized autoencoder (HQA). To get a total comprehension of this model, we must first discuss our preliminary model known as HAE and the motivation behind vector quantization.

3.1.1 Autoencoders and the Hierarchical Auto Encoder (HAE)

An autoencoder is an unsupervised machine learning model used to learn efficient encodings of the input data while minimizing distortion. An autoencoder has two components, that being the encoder and the decoder. The encoder takes as input the original data x and outputs a representation of x as vectors in a lower dimensional latent space. We call this representation z, and it is effectively the compressed representation of x. The decoder takes as input z and outputs $\hat{x}$ which denotes the lossy reconstruction of x. Any criterion can be imposed on x and $\hat{x}$ depending on the use case of the autoencoder. For our use case of compression, we seek to minimize the distortion

between x and $\hat{x}$. We call this distortion measure the reconstruction loss, and we use the mean square error, shown in Equation 2, for reconstruction loss.

$$\mathcal{L}_{rec} = \frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{x_i})^2 \quad (2)$$

The hierarchical autoencoder model is made up of N encoder and decoder pairs denoted $(E_1, D_1), (E_2, D_2), \dots, (E_N, D_N)$. The n'th layer of this model is comprised of the composition of the encoders $E_1 \circ E_2 \circ \dots \circ E_n$ followed by the composition of the decoders $D_n \circ D_{n-1} \circ \dots \circ D_1$. Each layer is trained separately, and a model of n layers is trained using the following procedure:

1. Create $(E_1, D_1), (E_2, D_2), \dots, (E_n, D_N)$.
2. Train layer 1 consisting of E_1 and D_1.
3. To train layer $l \in \{2, \dots, \mathrm{n}\}$, freeze all $(E_k, D_k), 1 \leq k < l$ before the first epoch.

Freezing the encoder and decoder modules means that the gradient of the loss function with respect to their parameters are ignored, therefore the weights of the encoders and decoders are not affected by the training of later layers and only the deepest layer's modules are trained. This maintains the integrity of each layer allowing the user to select which layer to use at inference time as opposed to having no choice but to compress using the deepest layer.

This simple autoencoder does not effectively compress data enough and is not the focus of this research, but it does share the same structure and encoder and decoder components with HQA, thus we use HAE for quicker hyperparameter tuning and transfer learning.

3.1.2 Vector Quantization and the Vector Quantized Variational Auto Encoder

The Vector-Quantized Variational Auto Encoder is a model introduced by Oord, Vinyals, and Kavukcuoglu [3]. It was famous for its use in OpenAI's generative text to image model DALL-E. The model is an auto encoder which performs vector quantization on the latent vectors. Vector quantization (VQ) achieves a much greater level of compression by using persistent storage, such that the latent vectors $Z_e = \{z_e^1, z_e^2, \dots\}$ can be each represented by a set of integer indices rather than by a set of vectors. Apart from VQ quantizing Z_e into an array Z_q of integers, VQ-VAE also enhances the autoencoder by applying variational inference to learn the posterior distribution $P(Z_q | x)$, which defines VQ-VAE as a generative model, allowing it to generate new samples by sampling the prior over the latent space. However, in this research we will instead leverage this functionality for compression rather than generation of deepfakes.

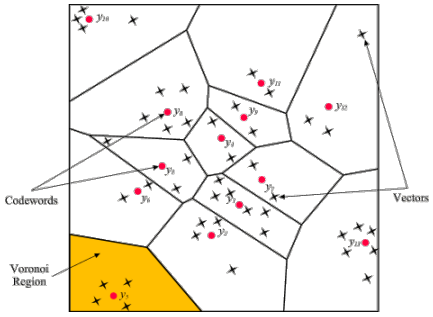
3.1.2.1 Vector Quantization Vector quantization is a process of discretizing a space through use of a vector quantization codebook denoted Q. This codebook has a size which denotes the number of vectors or codewords it contains. The process of quantizing an input z_e is simply choosing a codeword from the codebook based on that input. Oord. et. al. [3] propose Equation 3 which simply chooses the nearest neighbor of the input in the codebook based on Euclidean distance.

$$z_q = \min_{q \in Q} \| z_e - q \|^2 \tag{3}$$

Note that the "slices" of Z_e denoted z_e must have the same dimensionality as the codebook vectors q This quantization scheme effectively partitions our latent space into regions, where each region corresponds to all the vectors which quantize to a certain codeword. This is illustrated in Figure 4.

Figure 4

An Illustration of Vector Quantization in Two Dimensions.



The vectors in the codebook are trainable parameters in the VQ-VAE model, and thus the model must learn an optimal codebook during training time on top of optimizing the reconstruction error. Using a deterministic posterior, our loss function for VQ-VAE becomes Equation 4.

$$\mathcal{L}_{VQ-VAE} = \mathcal{L}_{REC} + ||\text{sg}[z_e] - z_q||^2 + \beta||z_e - \text{sg}[z_q]||^2 \quad (4)$$

Where sg[·] denotes the stop gradient function, and β is a hyperparameter which regularizes the amount in which the encoder should be trained relative to the codebook. Because of the transfer learning and due to the nature of the model, it is evident training the codebook vectors to minimize the distance from the codewords to the encoder output is more important than training the encoder to minimize the distance from the encoder's output to the codewords. Thus, we choose β as a small value between 0 and 1.

3.1.3 The Hierarchical Quantized Auto Encoder

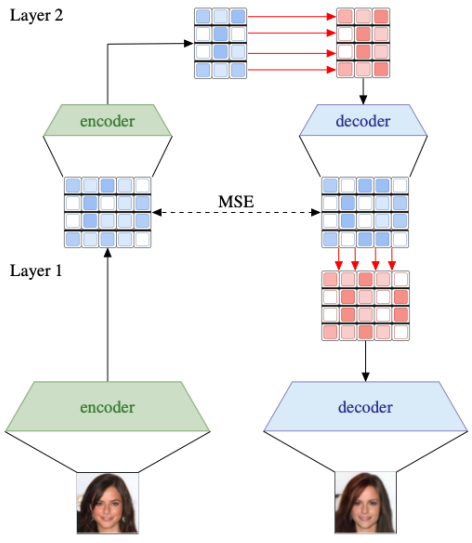
The hierarchical auto-encoder (HQA) inherits the hierarchical structure and training patterns of HAE but adds one quantization module Q for each layer. The forward propagation process for an HQA model consisting of two layers is as follows:

1. Let x denote the datapoint to be encoded.
2. $z_{e1} \leftarrow \mathrm{E}_1(\mathrm{x})$
3. $z_{e2} \leftarrow E_2(z_{e1})$
4. $z_{q2} \leftarrow Q_2(z_e 1)$
5. $\widehat{z_{e1}} \leftarrow D_2(z_{q2})$
6. $z_{q1} \leftarrow Q_1(\widehat{z_{e1}})$
7. $\hat{x} \leftarrow D_1(z_{q1})$

In this example, $\hat{x}$ denotes the reconstruction of x and z_{q2} is the compressed representation of x.

Figure 5

A Graphic Representation of the Forward Propagation of an Image Through a 2-layer HQA model. [1]



In fact, comparing the performance of HQA to VQ-VAE, another generative learned compression model that uses codebook learning is trivial, as a single layer HQA model is equivalent to the VQ-VAE model.

3.1.3.1 Stochastic Quantization For the vector quantization step in HQA we diverge from the discrete posterior imposed on the codebook of VQ-VAE and instead impose a stochastic posterior on our codebook introduced by Sønderby et al. [6] we achieve stochastic quantization. Our posterior is defined in Equation 5 as a sampling from a categorical probability distribution $Q(z_q = q|x)$ where the logits are defined by the Euclidean distance from the vector to be quantized to each codeword in the codebook.

$$Q(z_q = q|x) \propto e - \| z_e - q \|^2 \tag{5}$$

Vector quantization is then accomplished by sampling from $Q(z_q = q|x)$. In both cases, there exists a non-differentiability problem that we must account for during training. For deterministic quantization, we simply copy the gradients from z_q to z_e [3]. For stochastic quantization, we use the Gumbel-Softmax reparameterization trick and relaxation to achieve a differentiable sample [7]. The temperature of this softmax operation is a hyperparameter and best results are achieved by initially setting a high temperature allowing the model to explore during the initial phases of training, before slowly annealing the temperature as the model nears convergence.

3.1.3.1 Compression Ratio of HQA The size of the compressed representation of our data depends on the size of the data, the number of channels in the data, and the size of our codebook. The encoders and decoders consist of convolutional layers which each down sample and up sample by a factor of 2 in each convolution dimension. Thus, an input with c channels consisting of n values each being compressed by a model

containing a codebook with $|Q|$ codewords, an encoder and decoder using convolutional layers of dimensionality d_c, and l layers, we derive Equation 6 denoting the size of our compressed representation in bits:

$$S_c = \log_2(|Q|)\frac{n}{2^{ld_c}} \tag{6}$$

The size of our original data can be represented as such:

$$S_o = b \times c \times n \tag{7}$$

Where b represents the amount of bits in the datatype comprising the original data. Our compression ratio r is then given by $\frac{S_o}{S_c}$, giving us a general formula for our compression ratio in terms of the hyperparameters for our HQA model in Equation 8.

$$r = \frac{S_o}{S_c} = \frac{b}{\log_2|Q|}\, c2^{ld_c} \tag{8}$$

3.2 Datasets

In this research we use the MNIST handwritten digit dataset [8] and ImageNet100 as our image datasets and the modulations dataset from the torchsig library [8] as our RF dataset. The MNIST dataset consists of 70,000 28x28 black and white images of handwritten numeric digits from 0-9. ImageNet100 [10] is a subset of the well-known ImageNet [10] dataset, consisting of full color 224x224 images of various everyday objects and animals. As suggested by its name, ImageNet100 is a subset that only includes 100 of the 1000 classes. The modulations dataset is a synthetically created dataset consisting of communications signals modulations. We will only use a subset of the classes available in the modulation's dataset. Those being 4ask, 8pam, 16psk, 32qam_cross, 2fsk, and ofdm-256.

3.3 MNIST Methodology

Prior to training, we resize the dataset to 32x32 such that each image consist of 1024 pixels (bytes), and split the dataset into a train set and test set of 60,000 and 10,000 samples respectively. We initialize a 5 layer HQA model with a codebook size of 256 and encoder and decoder blocks that use 2D convolutions. This gives us a compression ratio of 4^l for layer l.

3.3.1 Training

This section delves into the details of training this model. Unless stated otherwise, this applies to all datasets and iterations of HQA.

3.3.1.1 Optimization We will use a variant of the well-known Adam stochastic optimization algorithm known as Rectified Adam (RAdam) introduced by Liu et al [8]. RAdam rectifies the high variance problems and the tendency of failing to properly generalize during the few initial steps suffered by Adam and other adaptive learning rate optimization algorithms. RAdam rectifies this by ignoring the momentum term for the initial steps of the algorithm and lowering the initial earning rate during the warmup step to achieve less variance and faster convergence.

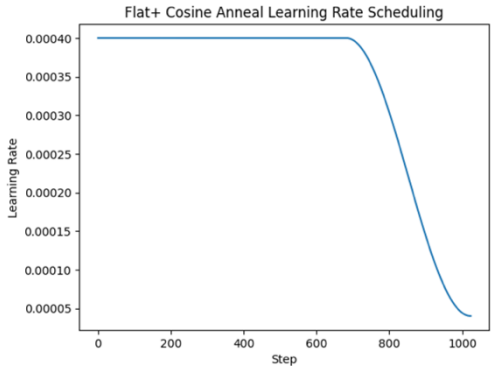
3.3.1.2 Learning Rate Scheduling It is often advantageous to anneal the learning rate η during training such that the trainable parameters are modified less drastically as the model nears convergence. We used Flat + Cosine Annealing to schedule our learning rate, choosing to keep our learning rate constant for the first T_0 steps of training before applying cosine annealing. Our learning rate at a step t is given by Equation 9.

$$\eta_t = \begin{cases} \eta_{max}, & t < T_0 \\ \eta_{min} + \frac{1}{2}(\eta_{max} - \eta_{min})\left(1 + \cos\left(\frac{t - T_0}{T_{max}}\pi\right)\right), & t \geq T_0 \end{cases} \quad (9)$$

Where η_{max} is our initial learning rate, η_{min} is a configurable minimum, and T_{max} is the total number of training steps we will perform. We choose $T_0 = \frac{2}{3}T_{max}$ such that the learning rate is annealed after two-thirds of the training is completed. We bound our learning rate by letting $\eta_{max} = 4 * 10^{-4}$ and $\eta_{min} = 4 * 10^{-5}$.

Figure 6

The Curve of Our Learning Rate Using Flat+ Cosine Annealing.



3.3.1.3 Learning the Codebook The codebook is the most important aspect of the model to learn as it not only achieves compression but acts as the largest source of distortion in the model. The codebook is also difficult to train properly due to utilization problems. Similar to the mode collapse problem suffered by GANs, the VQ-VAE can suffer from suboptimal codebook utilization. This occurs when only a small fraction of the codebook is used and can happed due to bad initialization or improper training. The stochastic quantization helps alleviate this issue, as well as codebook resets. Codebook resets introduced by Williams et. al. [5] seek to increase the usage of underutilized

codewords. It works by accumulating the number of usages of each codeword over a certain number of batches. If the least used codeword q_l is used 3% or less than the most used codeword q_m, q_l is reset according to rule defined in Equation 10.

$$q_l \coloneqq q_m + \epsilon \tag{10}$$

Where $\epsilon \in \mathcal{N}(0,0.01)$. This ensures that no codewords that otherwise would be useful are underused due to bad initialization. We considered the number of steps between each reset as a hyperparameter and achieved the best results by periodically increasing this value.

3.3.2 MNIST Classifier

LeNet, a convolutional neural network proposed by LeCun et al. [9] in 1998, remains popular to this day due to its high effectiveness and ease to implement. LeNet utilizes two 2D convolutions for feature extraction followed by a multi-layer perceptron with a hidden layer consisting of 84 neurons for classification. We utilize the cross-entropy loss function which is standard for training a classification problem with more than two classes.

3.4 ImageNet100 Methodology

ImageNet100 being another image dataset, our procedure will be similar to that of MNIST. We must consider that the input shape is different (224x224x3) rather than (28x28). Thus we must modify our compression model to expect an input with three channels (red, green and blue) rather than one (grayscale), and output an image with three channels. ImageNet100 being a much more complex dataset with 100 different classes, we must use a more complex classifier. We choose to use ResNet-50, a 50-layer convolutional neural network for classification [14]. The final layer consists of a multi-

layer perceptron consisting of 1,000 neurons. We modify this such that the output consists of only 100 neurons and perform fine tuning on a ResNet-50 instance pretrained for ImageNet.

We also note that our compression ratios per layer are different than those on MNIST. ImageNet100 samples consist of 3 channels, and using our equation for r we get a compression ratio of 3×4^l for layer l.

3.5 Modulations Methodology

VQ-VAE and therefore HQA by extension is a model suited to work on images. Therefore, we needed to make some changes to the architecture to allow HQA to train on time series RF data. This RF data consists of complex numerical data sampled at a consistent time interval. Thus, for use in neural networks, we must convert our data to two channels consisting of the real and imaginary parts of the original data. Our first change to the model architecture is replacing the 2D convolutions to 1D convolutions. Because our data consists of both positive and negative values, we replace the sigmoid activation at the end of the outer decoder with hyperbolic tangent. We finally add another term to our loss function which represents the cosine similarity to preserve the phase, an important feature in digital phase modulations [9]. We also add a weight to the cosine loss as yet another hyperparameter to control how prevalent this term is in backpropagation. We call this novel modified model HQA-RF.

Similar to MNIST, we train a deep classifier on the original samples. The classifier we use is the pretrained EfficientNet4 model [15] provided by the torchsig library [9]. This model is a modified version of EfffficientNet4 tailored to classify RF

data. We compare the accuracy of the classifier on the original data, with the accuracies of the reconstructions over all four layers.

Chapter 4

Results

4.1 MNIST Results

Table 1 shows the accuracy scores of our LeNet-5 classifier trained on the original test-set data on the original and reconstructed data by layer.

Table 1

Accuracy of Original MNIST Samples Compared to the Reconstructed Samples.

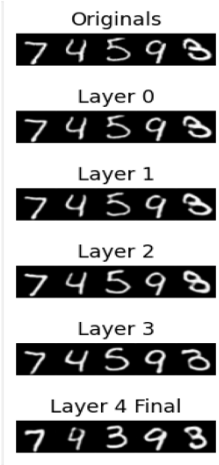
Dataset	LeNet-5 Accuracy	Compression Ratio
Original Test Set	0.9909	1
JPEG (Quality = 100)	0.9914	1.19
JPEG (Quality = 75)	0.9912	2.03
JPEG (Quality = 50)	0.9898	2.26
JPEG (Quality = 25)	0.9902	2.43
JPEG (Quality = 0)	0.9763	2.84
Layer 1 Reconstruction Test Set	0.9896	4
Layer 2 Reconstruction Test Set	0.9863	16
Layer 3 Reconstruction Test Set	0.951	64
Layer 4 Reconstruction Test Set	0.8533	256
Layer 5 Reconstruction Test Set	0.7164	1024

We can see from Table 1 that we can push the compression ratio to as much as 64 before the accuracy score falls below 90%. Compared to the average 10:1 compression ratio achieved by JPEG, HQA can push much further with minimal loss in both image quality and downstream task performance. Recall that the layer used can be chosen at inference time, so with only one model the user can choose how much accuracy they are willing to sacrifice in exchange for a higher compression ratio. We visualize the distortion between the original and reconstructed samples for each layer in Figure 7. We

note that the distortions present are not the result of seemingly random noise, but rather the behavior exhibited is samples morphing into other digits as the layers increases. This serves to illustrate how our model is data informed, as the expected distortions still take the form as valid digits rather than unstructured noise. In the cases where digits remain constant across the layers, we see the digits drawn in a slightly different style than the originals, again illustrating how well the model learns the underlying patterns and structure of the handwriting.

Figure 7

Comparison of Original and Reconstructions Across All 5 HQA Layers.



After creating adversarial examples based on the classifier using FGSM for various values of ε, we proceed with our experiment and perform classification on the various reconstructions of the attacked dataset. We performed the same experiment using various values for ε representing attacks of various intensities. In all cases, the reconstructions generated from HQA show a resiliency to adversarial noise generated using FGSM. We find that performing reconstruction on the attacked images using just

one layer of hierarchy before classification adds substantial resiliency to low intensity attacks, while more layers are necessary to maximally mitigate high intensity attacks.

Table 2

Accuracy of Original vs Reconstructed Images With Added Adversarial Noise.

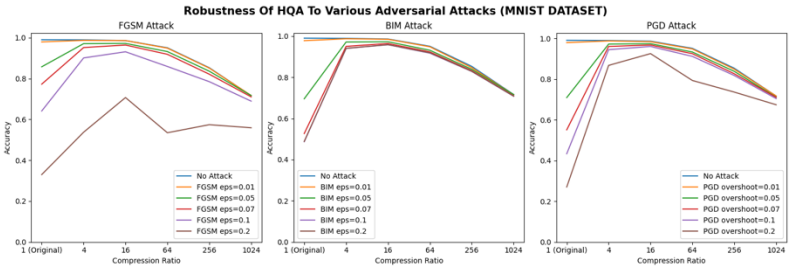
Dataset (FGSM)	**Accuracy** ($\epsilon = 0.1$)	**Accuracy** ($\epsilon = 0.05$)	**Accuracy** ($\epsilon = 0.07$)	**Accuracy** ($\epsilon = 0.1$)	**Accuracy** ($\epsilon = 0.2$)
Original Test Set	0.9797	0.8579	0.7728	0.6413	0.3298
Layer 1 Reconstructions	0.9873	0.9713	0.9521	0.9013	0.5373
Layer 2 Reconstructions	0.9862	0.9724	0.9645	0.9315	0.7072
Layer 3 Reconstructions	0.9492	0.9325	0.9191	0.8595	0.5347
Layer 4 Reconstructions	0.8514	0.8363	0.8217	0.7851	0.5742
Layer 5 Reconstructions	0.7173	0.717	0.7106	0.6896	0.5591

In addition to the above experiment which used FGSM, we ran a small experiment in which we repeated the above process using two additional adversarial example attacks. We saw that the resilience of our proposed pipeline is even more pronounced, with FGSM performing the best (causing the most perturbance) out of all attacks using our modified pipeline. The BIM attack not only performed worse than the FGSM attack on the original data, but HQA was also more effective at mitigating the BIM attack. While PGD performed better than FGSM on the original data, it too was combatted by HQA extremely effectively. The resilience is shown in Figure 8, as the accuracy curves of the attacked samples appear to converge closer and closer to the curve of the non-attacked samples as more layers of hierarchy are used for reconstruction.

Across all attacks of various magnitudes, we can see that applying our compression does yield a considerable improvement in accuracy.

Figure 8

Accuracy Curves Comparing the Accuracy Score of LeNet on Original Attacked Data and Reconstructed Attacked Data Using Various Attack Methods and Parameters.



We also hypothesize that the decline in accuracy for greater values of epsilon is mainly due to the noise perturbing the perceptual quality of the image rather than the noise artificially maximizing the loss function of the classifier. Figure 9 visually shows the perceptual perturbance caused by the addition of FGSM generated noise, and importantly is shows that the attack loses its imperceptibility as epsilon increases.

Figure 9

Visualization of FGSM Attacked MNIST Images With Varying Epsilon. Color Map Changed From Greyscale to Emphasize Noise. Predicted Class Notated Under Each Sample.

4.2 ImageNet100 Results

Upon modifying our HQA model to accommodate the ImageNet100 dataset, we proceeded with the same process. Unsurprisingly, due to the added complexity, size, and number of classes of the data, ImageNet100 performed worse than MNIST. However, visually the images were quite perceptible up to layer 3, which had a compression ratio of 192 and accuracy of 36%. In comparison, JPEG was able to achieve a maximum compression ratio of 59.33 while achieving 22% accuracy. The results of our experiment are displayed in Table 3.

Table 3

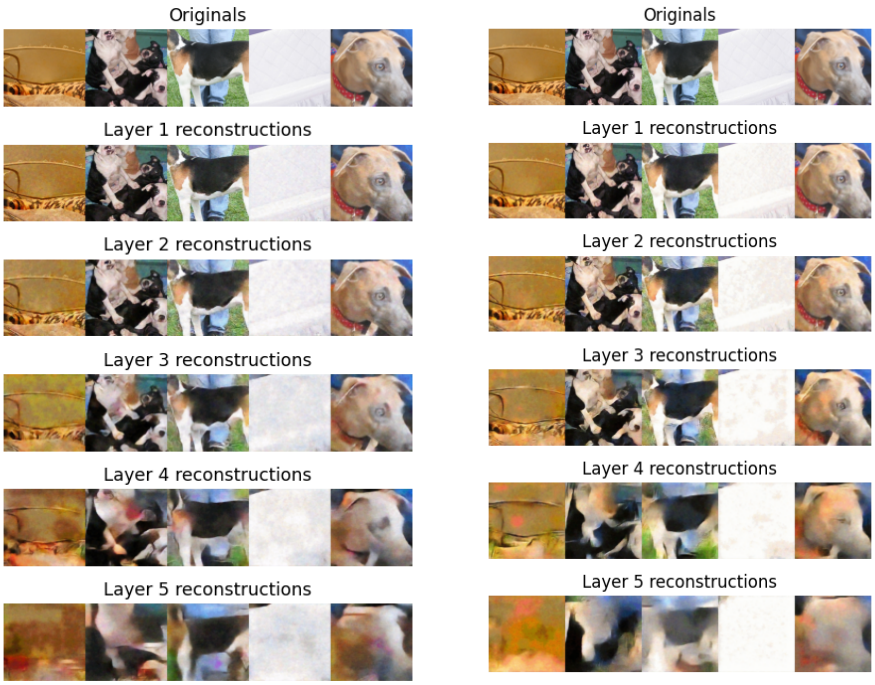
Accuracy Scores on Original ImageNet100 Samples Compared to the Reconstructed Samples.

Dataset	ResNet-50 Accuracy	Compression Ratio
Original Test Set	0.856	1
JPEG (Quality = 100)	0.857	1.43
JPEG (Quality = 75)	0.8526	9.22
JPEG (Quality = 50)	0.8492	14.02
JPEG (Quality = 25)	0.8242	21.21
JPEG (Quality = 0)	0.2224	59.33
Layer 1 Reconstruction Test Set	0.8254	12
Layer 2 Reconstruction Test Set	0.7264	48
Layer 3 Reconstruction Test Set	0.3592	192
Layer 4 Reconstruction Test Set	0.089	768
Layer 5 Reconstruction Test Set	0.0314	3072

Noticing the drop in accuracy score is much more drastic than with MNIST, we visualize the distortion in the reconstructed samples in Figure 10. As an attempt to increase the perceptual quality of the model, we trained an equivalent model using 512 codewords hypothesizing that a more complex dataset would require more codewords to accurately represent. This model ended up performing marginally better in the early layers but had poor performance in the later layers. Perceptually we can see the deeper layer reconstructions in the 256 codeword model are superior to those in the 512 codeword model. This could be due to the fact that a codebook of size 512 codewords is significantly harder to train and that each codeword contains less information, therefore leading to more difficulty creating an accurate reconstruction using the same amount.

Figure 10

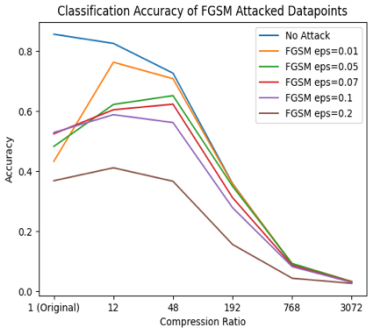
Reconstructions on 256 Codeword Model (left) vs 512 Codeword Model (right).



We can see the same resiliency to adversarial attacks that we saw in the experiment using the MNIST dataset. When compared to attacking the MNIST dataset, the attack appeared to be much more effective on the uncompressed data when using lower values for epsilon. We hypothesize this is due to the greater number of features as well as the greater number of classes, which increases the chance of an attack creating a disturbance large enough to cross the decision boundary of the true class. However, we see the same trend of achieving maximum accuracy when utilizing only one or two layers of hierarchy. Additionally, we notice that adding at least one layer of hierarchy helps improve accuracy in all cases. The improvement in accuracy achieved by using one layer of hierarchy over no compression is inversely proportional to the intensity of the attack.

Figure 11

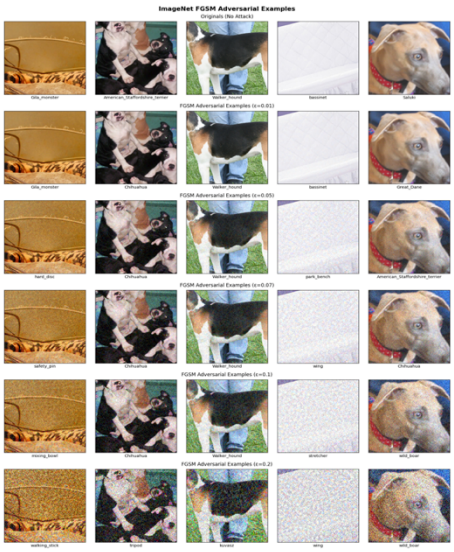
Accuracy Curves Comparing the Accuracy Score of FSGM Attacked ImageNet Samples at Each Layer.



As imperceptibility is important for the practical efficacy of adversarial attacks, we again visualize samples from our dataset that have been attacked at various magnitudes. We can see from Figure 12 the loss of imperceptibility as the noise can be easily detected past a threshold epsilon.

Figure 12

FGSM Attacked ImageNet100 Samples. Predicted Class Notated Below.



4.3 Modulations Results

We conclude the results chapter with our findings from our experimentation using the modulations dataset. Due to the large amount of experimentation and modifications, we will include a subsection dedicated to model selection in this section.

4.3.1 Model Selection

We used our accompanying HAE-RF model for transfer learning and hyperparameter tuning concerning the encoder and decoder. However, to learn the loss coefficients and codebook hyperparameters, we performed a large ablation study on our HQA-RF model. We varied our codebook initialization, KL divergence loss coefficient, commit loss coefficient, cosine similarity coefficient, the number of batch normalization layers. We ignored the loss functions and decided to perform this selection exclusively using the accuracies.

Table 4

Select Results from Model Selection With our Best Result Highlighted.

Codebook Size	Codebook Dimension	# Residual Blocks	KL Loss Coefficient	Commit Loss Coefficient	Cosine Similarity Coefficient	# Batch Normalization Layers	Layer 1 Accuracy	Layer 2 Accuracy
128	128	2	0.1	0.005	0.7	1	0.7898	0.6939
128	64	2	0.1	0.005	0.7	1	0.6895	0.5360
64	128	2	0.1	0.005	0.7	1	0.7926	0.3045
64	64	2	0.1	0.005	0.7	1	0.8012	0.6223
64	64	3	0.001	0.001	0.7	0	0.8333	0.82
256	256	2	0.1	0.005	0.7	1	0.8312	0.4181

We attempted codebook initialization using both normal and uniform distributions. We omitted this as a column since normal distribution performed better in 100% of our selection runs. We also found that transfer learning the encoder and decoder

from HAE-RF was an essential part of training, improving our accuracy as much as 10%. Table 5 shows the accuracies of EfficientNet on reconstructions from both HQA-RF and HAE-RF. Reminder that the HAE-RF is a preliminary model that does not achieve optimal compression. Results from HAE-RF are displayed only for establishing a theoretical limit.

Table 5

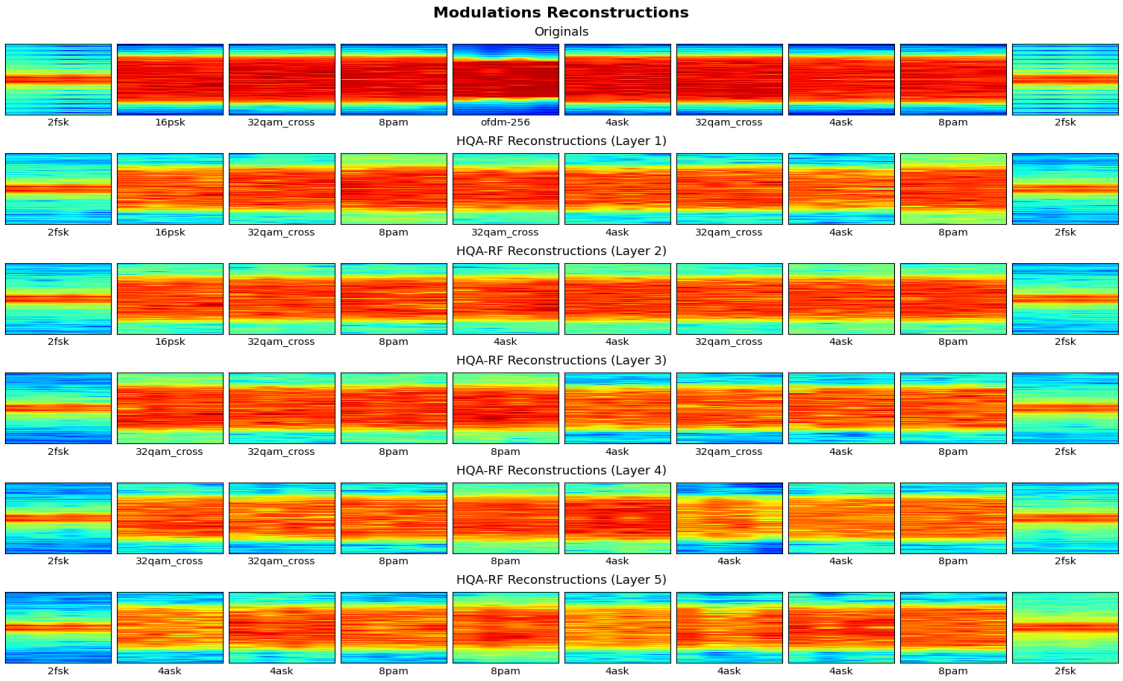
Comparing Accuracy of EfficientNet on HQA-RF and HAE-RF Reconstructions (Original Accuracy is 100%).

Layer #	HQA-RF	HAE-RF
1	0.83	1
2	0.82	1
3	0.6666	1
4	0.6516	1
5	0.535	0.87333

The modulations dataset being a much more complex dataset, we expect to see a drop in performance from image datasets with regards to classification accuracy. An important feature of IQ data is that it is heavily reliant on preserving the frequency and phase, with the "quadrature" or Q component having a phase offset of 90 degrees from the "in-phase" or I component. Thus, any noise added by the reconstructions will perturb the classifier. We can see this perturbance in the frequency map between the originals and reconstructions by visualizing them using spectrograms.

Figure 13

Spectrograms of Original and Reconstructed Data by Layer. Predicted class notated below.



With regards to the adversarial attack resiliency, we were unable to find an adversarial attack method that is compatible with IQ signals. Traditional adversarial attack methods would perturb the phase relationship of the signal, thus making it impossible to be received and interpreted as a wireless signal. In order to evaluate the robustness of HQA-RF against adversarial attacks, we must apply a sophisticated adversarial perturbation that maintains the integrity of the phase information. However, this is an ongoing research field and out of the score of this thesis.

.

Chapter 5

Conclusions and Further Research

5.1 Conclusions

We conclude that our HQA model can effectively compress image data, being able to achieve a larger compression ratio than JPEG and in both cases achieving a higher accuracy than JPEG at higher compression ratios. We also confirm our hypothesis that our proposed data pipeline of compression, transmission, and classification adds resiliency to adversarial example attacks targeting the deep classifier for our image datasets.

We also proposed a novel variant of HQA known as HQA-RF which is effective at compressing RF data. While it is effective at compressing our synthetically generated datapoints, it reveals a weakness to reconstructing noisy data. Further research is required to confirm the robustness of HQA-RF to adversarial attacks both general and targeted towards IQ data.

5.2 Further Research

There are multiple ways our research can be extended. I will propose both changes to our models to create higher fidelity reconstructions as well as changes to our training and experimentation that should be explored.

5.2.1 VQ-GAN

VQ-GAN is a model introduced by Esser et. al. [10] which modifies the VQ-VAE structure in two different ways. It adds a GAN component to the decoder to the decoder to increase the fidelity of the images. It also uses a transformer to learn the prior distribution over the codebook, such that it can choose codewords to generate new

images. As stated in chapter 2, we can potentially leave out some codewords and use the transformer to generate the missing codewords given the ones we leave in. Using this, we can potentially accomplish greater data compression by combining two different approaches for learned compression.

5.2.2 Increasing the Robustness of HQA-RF to Noise

By adding synthetic noise to our dataset prior to training, we could potentially reduce the sensitivity of HQA-RF to noisy data. Not only would our model serve as a proof of concept for a denoising solution, but we could perform further experimentation to explore our hypothesis on whether HQA-RF adds resiliency to adversarial attacks.

5.2.3 Additional Datasets

In another project, I have recently found success with using a VQ-VAE model to generate audio. I accomplished this by transforming the audio to a spectrogram representation and using a transformer to learn the prior over the codewords. However, I have not yet found success reconstructing audio using the hierarchical model, and more research can be done into exploring this domain using 1D convolutions.

I have also recently discovered EEG datatypes which are time series samples of brain waves. Augmented neural technology such as Neuralink is already being tested on human subjects, and while the ethical ramifications of such technology must be considered, there is no doubt that the compression of EEG data for downstream processing has many legitimate medical and research-based use cases.

References

[1] M. Williams, S. Kokalj-Filipovic and A. Rodriguez, "Analysis of Lossy Generative Data Compression for Robust Remote Deep Inference," 28 June 2023. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/3586209.3591404. [Accessed 29 April 2024].

[2] I. J. Goodfellow, J. Shlens and C. Szegedy, "Explaining and Harnessing Adversarial Examples," 20 March 2015. [Online]. Available: https://arxiv.org/abs/1412.6572. [Accessed 29 April 2024].

[3] "AI-RAN Alliance," AI-RAN Alliance, [Online]. Available: https://ai-ran.org. [Accessed 29 May 2024].

[4] A. v. d. Oord, O. Vinyals and K. Kavukcuoglu, "Neural Discrete Representation Learning," 30 May 2018. [Online]. Available: https://arxiv.org/abs/1711.00937. [Accessed 29 April 2024].

[5] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser and I. Polosukhin, "Attention Is All You Need," 2 Augus 2023. [Online]. Available: https://arxiv.org/abs/1706.03762. [Accessed 29 April 2024].

[6] W. Williams, S. Ringer, T. Ash, J. Hughes, D. MacLeod and J. Dougherty, "Hierarchical Quantized Autoencoders," 16 October 2020. [Online]. Available: https://doi.org/10.48550/arXiv.2002.08111. [Accessed 28 April 2024].

[7] C. K. Sønderby, B. Poole and A. Mnih, "Continuous Relaxation Training of Discrete Latent Variable Image Models," 2017. [Online]. Available: http: //bayesiandeeplearning.org/2017/papers/54.pdf.

[8] E. Jang, S. Gu and B. Poole, "Categorical Reparameterization with Gumbel-Softma," 5 August 2017. [Online]. Available: https://arxiv.org/abs/1611.01144. [Accessed 29 April 2024].

[9] L. Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine,* vol. 29, no. 6, pp. 141-142, 2012.

[10] M. G. G. V. P. V. B. C. S. K.-F. C. L. R. D. M. Luke Boegner, "Large Scale Radio Frequency Signal Classification," 20 July 2022. [Online]. Available: https://arxiv.org/pdf/2207.09918. [Accessed 29 May 2024].

[11] C.-H. Yeh and Y. Chen, "{IN100pytorch}: PyTorch Implementation: Training ResNets on ImageNet-100," 2022. [Online]. Available:

https://github.com/danielchyeh/ImageNet-100-Pytorch?tab=readme-ov-file. [Accessed 29 May 2024].

[12] W. D. R. S. L.-J. L. K. L. L. F.-F. Jia Deng, "ImageNet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, Miami, FL, USA, 2009.

[13] L. Liu, H. Jiang, P. He, W. Chen, X. Liu, J. Gao and J. Han, "On the Variance of the Adaptive Learning Rate and Beyond," 26 October 2021. [Online]. Available: https://arxiv.org/abs/1908.03265. [Accessed 29 April 2024].

[14] L. B. Y. B. P. H. Yann LeCun, November 1998. [Online]. Available: http://vision.stanford.edu/cs598_spring07/papers/Lecun98.pdf. [Accessed 29 04 2024].

[15] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," [Online]. Available: https://arxiv.org/abs/1512.03385.

[16] A. Rodriguez, Y. Kaasaragadda and S. Kokalj-Filipovic, "Deep-Learned Compression for Radio-Frequency Signal Classification," 2024 March 5. [Online]. Available: https://arxiv.org/abs/2403.03150. [Accessed 29 April 2024].

[17] Q. V. L. Mingxing Tan, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," 03 June 2019. [Online]. Available: http://arxiv.org/abs/1905.11946. [Accessed 29 April 2024].

[18] P. Esser, R. Rombach and B. Ommer, "Taming Transformers for High-Resolution Image Synthesis," 23 December 2021. [Online]. Available: https://arxiv.org/abs/2012.09841. [Accessed 29 April 2024].