5-6-2021 11:00 AM

Contrastive Learning of Auditory Representations

Haider Al-Tahan, *The University of Western Ontario*

Supervisor: Mohsenzadeh, Yalda, *The University of Western Ontario*

A thesis submitted in partial fulfillment of the requirements for the Master of Science degree in Computer Science

Abstract

Learning rich visual representations using contrastive self-supervised learning has been extremely successful. However, it is still a major question whether we could use a similar approach to learn more efficient auditory and audio-visual representations. In this thesis, we expand on prior self-supervised methods to learn better auditory and audio-visual representations. We introduce various data augmentations suitable for auditory and audio-visual data and evaluate their impact on predictive performance, and demonstrate that training with both supervised and contrastive losses simultaneously improves the learned representations compared to self-supervised pre-training followed by supervised fine-tuning. We illustrate that by combining all these methods and with substantially less labeled data, our framework achieves significant improvement on prediction performance compared to the supervised approach. Moreover, compared to the self-supervised approach, our framework converges faster with significantly better representations.

Keywords: Self-supervised Learning, Contrastive Learning, Supervised Learning

Summary for Lay Audience

Audio recognition is a fundamental challenge to the goal of automated perception. Although humans are proficient at perceiving and understanding sounds, making computers do the same poses a challenge due to the wide range of variations in auditory acoustics. Currently, there are overabundance amount of unlabelled (or not annotated) auditory data. Moreover, data are available in a wide range of formats, from various sources, and often they are not stored in a format that is ready to feed into a machine learning pipeline, hence, the process of curating and annotating a dataset is expensive and time-consuming. Therefore, advances in this area are being held back by the lack of adequate unsupervised learning algorithms for auditory data to utilize these data. This thesis directly addresses this shortcoming by developing an unsupervised approach of training algorithms to carry down tasks such as classification, detection, etc. The main goal of this thesis is to investigate the various components that have direct effect on the performance of these algorithms.

Acknowledgements

Navigating academia is a challenging task, especially during a pandemic. Without the enormous support I received throughout this endeavor I would not have been able to complete my master's degree. For this reason, I dedicate this section of my thesis to all the individuals that motivated and helped me succeed:

I would like to express my most sincere gratitude to my supervisor Dr. Yalda Mohsenzadeh for her valuable guidance and advice. Dr. Mohsenzadeh not only taught me various skills on how to address academic research questions but she provided me with many opportunities to learn, grow, and pursue my interests. It was an absolute honour to work with such a person.

With how hard the pandemic made things and being far from home, I feel fortunate to say that I never felt homesick nor lonely. This is solely because of the Heil family. To everyone in the Heil family, I extend my deepest gratitude as I always felt welcomed and at home among a family that cared about me. I will be forever grateful!

The completion of my dissertation would not have been possible without the relentless support and constructive advice of Dalton Heil. Dalton has always expressed profound belief in my abilities even when I did not. Even if my dissertation accounted for nothing, I would still be extremely appreciative of such an opportunity as it has introduced me to Dalton.

I cannot begin to express my appreciation to Ehsan Tousi, who always was the voice of reason in various parts of my manuscripts. Furthermore, Ehsan's unwavering advice, guidance, and patience taught me essential tools to navigate academia.

I'd also like to extend my deepest gratitude to Sohrab Salimian, George Tomou, Sharmini Atputharaj, and Gaelle Nsamba Luabeya. Their unparalleled insight, advice, and support inspired me to pursue academia.

I would like to thank Dr. Dan Lizotte, Dr. Boyu Wang, and Dr. Vijay Parsa for being on my thesis committee, their encouraging comments and fruitful review of my thesis gave me the ambition to pursue questions related to my thesis further.

Contents

List of Figures

List of Tables

List of Abbreviations

CLAR	Contrastive Learning of Auditory Representations
CNN	Convolutional Neural Network
BN	Batch Normalization
GAN	Generative Adversarial Network
AE	Auto-Encoder
NT-Xent	Normalized Temperature-scaled Cross Entropy Loss
MLP	Multi-layered Perceptron
ReLU	Rectified Linear Units

Chapter 1

Introduction

Major advances of the internet in the past decades have introduced an era of overwhelming amount of unlabelled digital data. The challenge in deep learning has been increasingly gravitating towards finding ways to utilize such untapped data. Although humans are proficient at perceiving and understanding the world around them, making computers do the same poses a challenge due to the wide range of variations in characteristics (e.g., appearance, acoustics). Roughly inspired by how the human brain navigates the world, recent deep learning approaches align with some of the basic principles of human perception. The product of such principle can be clearly seen in infants with their rapid ability to learn and generalize categorizations of various entities [13]. At a high level, this unique ability can be argued to be the result of the parent correcting the child's inaccurate articulation about entities within his/her environment (Supervised Learning). Simultaneously, it is also the ability of the child to discriminate entities by finding dissimilarities between their intrinsic characteristics e.g. appearance, acoustics (Unsupervised Learning). For instance, the child might intrinsically find that different toys are not the same based on appearance or even the sound they produce without knowing the name of the object. At the low-level, decades of cognitive neuroscience research has demonstrated that the brain accomplishes these complicated tasks through a cascade of hierarchical processes that nourish high-level representations of the environment [14, 15, 16, 17, 18].

With the power of deep learning and the abundance of uncurated digital data, there has been growing interest in developing better machines that can perceive the world around us using sound. Although humans are proficient at perceiving and understanding sounds, making algorithms perform the same task poses a challenge due to the wide range of variations in auditory features. For instance, sounds can be generated from interactions with objects (e.g., dropping a glass, closing a door), certain activities (e.g., dish washing, cooking), instruments (e.g., violin, drum), and can even be environmental (e.g., wind, rain). Further, some can be repetitive over a duration, while others are momentary. In many real-world scenarios these sounds occur together, making audio understanding exponentially more difficult. Achieving automated auditory perception requires the learning of effective representations. Often prior work derive effective representations through discriminative approaches [19, 20, 21, 22]. That is, similar to supervised learning, the model learns the mapping between the input signal to the class label. The underlying assumption with such approach is that the latent representations carry effective representations for the designed tasks. One fundamental problem with such learned representations is the potential limitation to generalizability. First, those rep-

resentations are only limited to availability of expensive and time consuming labeled data. Secondly, representations are skewed towards one particular domain (e.g. speech, music, etc ...). Therefore, in both cases, major fine-tuning to the targeted training data would be required. Alternatively, recent self-supervised approaches using contrastive learning in the latent space have been shown to learn efficient representations that achieves state-of-the-art performance in images [12, 23, 24, 25, 26, 27] and videos [23]. However, it is still a major question on how we can achieve similar landmark on auditory data.

1.1 Motivation

Developing an efficient system that can perceive sound can aid various applications of sound understanding ranging from surveillance [28] and music classification [29, 30] to audio generation [31, 32] and deep-fake detection [33]. However in-order to achieve a notable feat in sound understanding, we need to learning efficient data representations. Data representation is a crucial component contributing to the success of machine learning algorithms. Hence, much of the effort in deploying effective machine learning algorithms are spent on designing preprocessing pipelines that result in efficient representations (features). Learning efficient data representations implies that the critical transformations involved in a machine learning algorithm can transform or extract high-level task-relevant information from low-level sensory data (e.g. images/audio). In deep learning, task relevant information or representations are formed during optimization by a composition of multiple non-linear transformations of the input data with the goal of yielding efficient representations for tasks like classification, recognition etc. Therefore, those learned representations can change depending on - but not limited to - the model's architecture, hyper-parameters, model's initialization, training data, and/or optimization method [34].

As a simple example, consider the problem of optimizing a system to classify a set of shapes (e.g. circle, square, triangle etc...). Based solely on the training data, the system might converge to representations that distinguish between various types of shapes by counting the number of edges. Although those representations could aid in the classification of a simple sub-set of shapes, the system would drastically fail when asked to distinguish between a square and rectangle. This simple example demonstrates that indeed representations are important component of effective machine learning algorithms in terms of performance and generalizability. Furthermore efficient representations contribute to interpretability and explainability of machine learning algorithm. Over the past decade, a great amount of work has been dedicated towards visualizing representations of models optimized on natural images [35, 36, 37, 38, 3]. For natural images, representations are far more complex than just edges, although edges are part of those representations. Figure 1.1 shows visualized representations of VGG19 [1] architecture trained on ImageNet dataset [2] consisting of ~2 million natural images. In deep convolutional neural networks (CNN) [39], early layers representations capture lower level local information within an image such as edges and textures. The further the layer is from the input image, higher level global information are captured such as patterns and parts of an object.

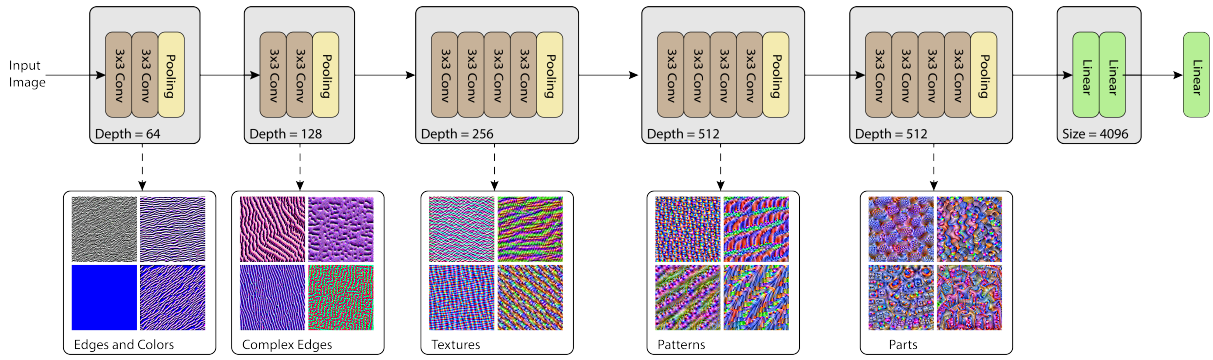

Figure 1.1: Illustration of visualized representations of VGG19 [1] architecture trained on ImageNet dataset [2] consisting of 2 million natural images. Each image below the convolutional block are representations visualized using optimization methods [3].

1.1.1 What makes a representation good?

Given the importance of good representations, knowing what makes a representation good is also important. Especially given that the machine learning research community is moving towards large-scale deep models that are more difficult to train and tune [1, 5, 40, 41, 42]. Therefore, developing such intuition can provide us with a road-map towards how we could improve on such complex systems. In this section, we will point out some relevant concepts to understanding good representations expressed in [34, 43]. Ideas discussed in this section provide some landmark to the frameworks introduced in Chapter 4.

Multiple Levels of Abstractions

When dealing with low-level sensory data, we are often presented with higher-level of variations that are not necessarily helpful for a target task. However, if we disentangle the factors of variation, it would allow for relatively easier generalization. Good low level representations should lead to more abstract representations because more abstract concepts can often be constructed in terms of less abstract ones (Figure 1.1); with more abstract concepts are generally invariant to local changes of the input. For instance, assume that we are given MNIST [44]; a handwritten digits dataset with 60,000 training samples of 28×28 images. The depth of a network is defined as the length of the path from the input layer to the output/last layer of the network. Hence, a relatively shallow feedforward neural network would have less degree of freedom over the construction of low-to-mid level concepts such as digits' minor rotations that generally does not affect the meaning of the digit but is a result of a typical handwriting alteration. Because the neural network is shallow, it is likely to degrade in performance due to fewer levels of abstractions in the representations. Alternatively, unlike shallow networks, deep networks can progressively develop and expand rich representations at higher layers. In addition to promoting progressive levels of abstractive representations, deep networks also enable feature re-use. The feature re-use property denotes that the number of ways that a network can re-use different representations can grow exponentially with its depth. The idea of feature re-use is re-enforced in modern CNNs by increasing the width of a network (i.e. the number of parameters/filters within a layer), showing consistent improvement in the learned represen-

tations across networks of different depth [45]. Thereby, by increasing the width of a network, we allow it to capture more ways to re-use representations.

Learning Criteria

If the network's architecture influences the level of abstraction the representations can reach, then the learning criteria could be said to influence the *what* representations are learned. In the past decades, tasks such as classification and recognition were widely adopted as learning objectives to learn efficient representations. However, in the case of representation learning, our true objective is fairly distanced from the down-stream objective, even though there are commonality in the representations. In representation learning, we are interested in learning representations that are not limited to one task, instead shared representations that can be utilized by multiple problems. While, in the case of classification, we want to minimize the number of incorrect classifications on the training dataset. Hence, one of the major challenges of representation learning is establishing a clear target objective for training. A good representation is one that disentangles the underlying factors of variation, but how do we translate that into appropriate training criteria? Luckily, recent advances in the area of unsupervised learning have given us an intuition into what such learning criteria can manifest into. We will discuss this in more detail in Section 2.

Shared Representations

Shared representation across tasks is another desired factor for good representations. A good representation allows for the re-use of features across tasks and domains. This is often seen when we train neural networks on general purpose tasks and aim to transfer the learned knowledge to a more specific task (Transfer Learning/Domain Adaptation). The general purpose tasks could produce representations that can be useful for another task with a small dataset. For example, we pre-train a neural network on ImageNet for object recognition then fine-tune the model on MNIST dataset to perform handwritten digit recognition; or the model representations could be generalized by the learning method, that is, we pre-train a neural network using unsupervised learning to learn general representations then fine-tune the model to down-stream applications such as classification. Transfer learning is one of the advantages behind deep learning, especially for low-level representations (i.e. edges and textures) that could be shared irrespective of the task [46, 47, 48]. A prominent example is [49] where pre-training on ImageNet shows significant boost in predictive performance even on sound classification tasks.

Priors for Representation Learning

In addition to shared representations, representation learning provides a convenient expression of general prior about the world around us. For instance, when dealing with temporal (e.g. sound), spatial (e.g. images), or spatio-temporal (e.g. video) data, one desired factor for good representations is that consecutive or spatially nearby observations tend to be associated with similar representations. Moreover, generally temporal or spatial changes in the data should produce slowly moving/changing representations [50, 51, 52]. Recent unsupervised learning

methods for visual representations capitalize on such spatial coherence by minimizing the difference between two different views of an image [12, 53, 54, 55]. Similarly, recent work with videos has explored similar spatio-temporal coherence [23, 52].

Additional prior that has been proven to be useful in recent unsupervised learning methods is simplicity of factor dependencies; good high-level representations are related to each other through linear dependencies. That is, high-level representations of categorical data should embody categorical information that are observable through linear predictors. Indeed, in the subsequent chapters, we will use a linear predictor on top of the frozen learned representations to evaluate the learned representations of unsupervised methods [56, 25, 24].

1.2 Contributions

The focus of this thesis is to leverage unsupervised learning using the contrastive learning framework to improve representation learning of audio data with deep neural networks. Furthermore, we investigate those approaches to learn superior auditory representations. We extend on SimCLR [12], a self-supervised framework for contrastive learning of visual representations. We show that similar framework could be adopted for learning effective auditory representations. Moreover, with a simple modification, we are able to reduce the training time and improve recognition performance. In order to accomplish this, we introduced major components that are important to nourish the learning of representations. We:

- Demonstrate the success of contrastive learning in learning efficient auditory representations.
- Investigate six data augmentation operations and show their effect on auditory classification task both with raw audio and extracted time-frequency audio features.
- Show that training with time-frequency audio features substantially improves the quality of the learned representations in contrastive learning compared to raw audio signals.
- Use our proposed framework CLAR (Contrastive Learning of Auditory Representations) to show when utilizing supervised and contrastive learning simultaneously while training, not only improves the learned representations but also speeds up the training.

1.3 Thesis Outline

While several works used contrastive learning on image data but this is yet to be studied in audio domain. In the following chapters, we trained two family of deep residual neural networks [5], one that is trained on raw audio signals while the other utilizes time-frequency audio features. Furthermore, we introduce composition of augmentations that are most efficient in learning auditory representations. Previous work on auditory data augmentations have shown effective methods that improve supervised classifications applied both on raw audio signal [57, 58, 59] and time-frequency audio features [60]. However, auditory data augmentations that nourish effective representations with contrastive learning is yet to be investigated. In this work, we also investigate the effect of the augmentations on learning representations

from both raw audio signal and time-frequency audio features in contrastive learning framework. Lastly, we introduce a novel semi-supervised approach that incorporate both supervised and self-supervised frameworks during training without fine-tuning which not only foster more efficient representations but also speed the training process.

Chapter 2

Background

2.0.1 Learning Algorithms

Machine learning algorithms can be broadly characterized by the kind of data they are allowed to use during their training process: **Supervised**, **Unsupervised**, and **Semi-Supervised**. In the following section, we will briefly describe each of them as they play an important part in this thesis.

Supervised Learning

Supervised Learning is the most popular learning approach as the conceptual idea behind it is simple and effective when presented with moderate to large amount of labelled data. As the name insinuates, supervised learning relies on a *teacher* to penalize inaccurate output from the model. Hence, the goal of a supervised learning approach is to learn the mapping from an input $\boldsymbol{x}$ to a corresponding output $\boldsymbol{y}$. Given a set of $\boldsymbol{N}$ training samples of inputs and outputs $\{(x_1, y_1), ..., (x_N,\ y_N)\}$, we seek to approximate a function ($f_\theta : X \rightarrow Y$) that maximizes the similarity between the groundtruth value y_i and the predicted output by the function $f_\theta(x_i)$. In this process, θ are the set of parameter(s) that cumulatively contribute to the outcome of the function. During training, we utilize gradient descent to systematically adjust the parameters (θ) to minimize a cost function. The cost function (J) is commonly used to assess the performance of a given function, the lower the value of the cost function, the less error a given function is performing on the training data:

$$J(\theta) = \sum_{i=1}^{m} L(f_\theta(x_i), y_i) \tag{2.1}$$

where L is a loss function that takes as inputs the predicted value $f_\theta(x_i)$ corresponding to the groundtruth value y_i and outputs how different they are. Depending on the data and task, $\boldsymbol{X}$ and $\boldsymbol{Y}$ can be of various forms. For instance, $\boldsymbol{X}$ can be image, audio, financial data, etc and if the task is classification, then $\boldsymbol{Y}$ would be a categorical variable from a finite set, i.e. $y_i \in \{1, ..., C\}$ where C is the number of classes. Alternatively, if the task is regression, then $\boldsymbol{Y}$ is a real value.

The advantages to this approach as stated before comes from its simplicity. That is, by minimizing the cost function ($J(\theta)$) of matching the predicted and true values of $\boldsymbol{Y}$, we are

able to obtain representations that are efficient to the task at hand. However, such approach suffers from various problems:

- **Performance degradation with more optimization**, in the process of minimizing the cost function, often large models that deals with complex data such as images would learn efficient representations to a certain point but the error would not reach an absolute zero. Hence, by training the model longer, the model would start to learn the noise of the training data, resulting in less efficient representations (i.e. over-fitting) [12, 61].
- **Dependence on labelled data**, given that we require to have $\boldsymbol{Y}$, we are limited by the number of samples within our dataset that are labelled. In applications were data are expensive or hard to annotate, this generates a generalizability problem that can be catastrophic in deployment (e.g. Healthcare [62], Self-Driving Cars [63]).

Unsupervised Learning

In machine learning, unsupervised learning is often attributed to algorithms that discover hidden patterns or data groupings without the need for annotated data. Generally, these algorithms involves clustering (e.g. K-Means), association (e.g. Association Rules), or dimensionality reduction (e.g. Principal Component Analysis). However, in deep learning, one use case for using unsupervised learning is that we want to learn the entire probability distribution that generated a dataset, whether explicitly (e.g. Variational Autoencoders), or implicitly (e.g. Generative Adversarial Networks). The benefit of this process is that it allows us to learn from unlabeled examples useful representations that can be then used for other down-stream tasks (i.e., in supervised learning). In Section 2.1.1, we describe in details some of the generative ways that unsupervised learning is used in representation learning which are closely related to this thesis.

More recently, there has been growing interest in using unsupervised learning to train models through discriminative approaches, called Self-Supervised Learning. Self-supervised learning is a type of unsupervised learning that uses the inherent structure of the data to learn meaningful representations. This is often done by automating the process of generating some kind of supervisory signal to solve an indirect task. For instance, we can train a model to predict the relative position of two randomly cropped patches from an unlabelled image [64]. Hence, the model would need to derive representations that contain the underlying structure of natural images (e.g. eyes are often close to the mouth). Similar to the generative unsupervised models, representations learned from this training can later be used for other down-stream tasks.

Semi-Supervised Learning

Following supervised and unsupervised learning, semi-supervised learning aims at integrating both by using labelled and unlabelled data for training. Depending on the context, semi-supervised learning can have different applied implications. One type of semi-supervised learning that considered a common practice in deep learning is to use small labeled data available to fine-tune parameters of a model - sometimes trained on a completely different task with unlabeled data- to improve performance. In self-supervised learning this approach is adopted by performing self-supervised pre-training on unlabeled data followed by supervised fine-tuning

on labeled examples [12, 65, 66, 67]. Although fine-tuning can be beneficial in big models (i.e. > 100 million parameters), fine-tuning on smaller models may result in *catastrophic forgetting* [68, 69, 70]. Catastrophic forgetting is a situation where in learning new tasks, the model may not use the shared parameters from the old task and "forget" them in the process. This phenomenon could partially explain why unsupervised learning benefits more from bigger models after fine-tuning [12]. That is, bigger models have the capacity to maintain representations from the unsupervised learning stage even after fine-tuning. In later chapters, we will present a different approach that abolish the two stage training by simultaneously training with supervised and unsupervised objective functions.

2.0.2 Neural Network Architectures

Perceptron

Loosely inspired by neuronal functions in the brain, neural networks are a family of machine learning models that aims to recognize underlying relationships in a set of data. These relationships are derived through a collection of units called nodes/neurons. A "neuron" in a neural network is a mathematical function that shares connections (through weights w_i) with neurons from the proceeding layer (x_i). Each neuron computes the sum over the product between the set of inputs (x_i) and their corresponding weights (w_i), followed by a nonlinear function, applied to the result. A neural network that consists of just an input and output layers is outline in Figure 2.1, called a *Perceptron*.

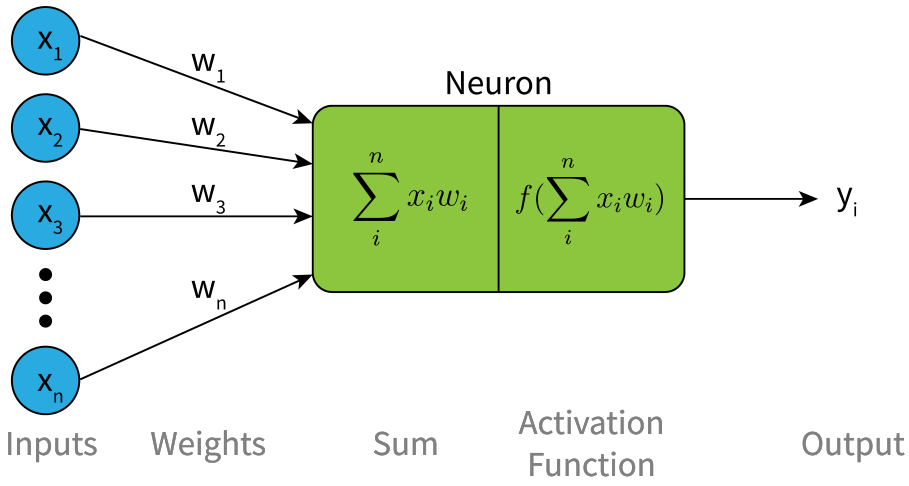

Figure 2.1: Perceptron Architecture.

Multi-layer Perceptron

Perceptrons are limited to the approximation of linear functions, where the output is similar to multiple linear regressions. By stacking perceptrons vertically, we can build a layer of perceptrons, also called the width of the model. If we interconnect those layers of perceptrons (i.e. hidden layers), we progress towards a multi-layered perceptron (MLP), also called deep

feedforward neural network, that can approximate a non-linear function and solve much more complex problems (Figure 2.2). The feedforward name originate from the notion that each layer's output becomes the input of the proceeding layers and no output is ever passed back to the current (self loop connection) or previous (feedback connection) layers. The learning algorithm updates hidden layers' weights until the neural network's margin of error is minimal. During training, the samples specify what the output layer must be for a given input, with no restraint on the hidden layers. Through the feedforward progression, the weights learn representations given the output of the previous layer and induce them into the neural network's further layers.

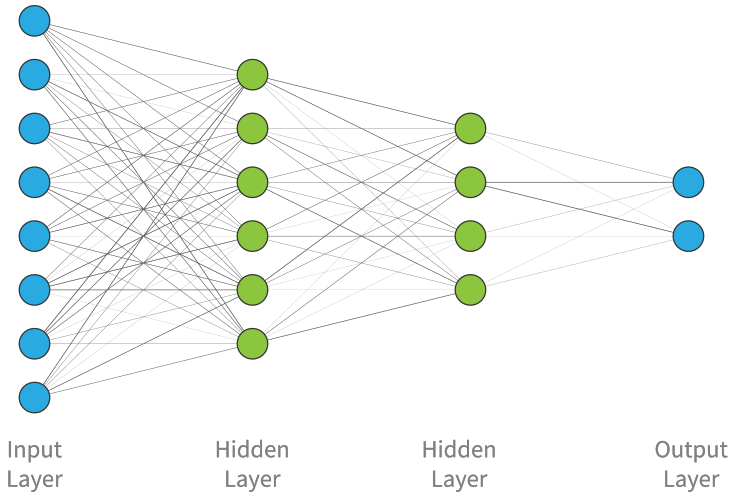

Figure 2.2: Multilayer perceptrons Architecture. Also called deep feedforward network or feedforward neural network.

MLP are computationally efficient when the size of input space is relatively low. However, as the input size increases, MLP becomes less viable option for such problems. For instance, we can consider the scenario were we are given an image dataset, which consist of colored images in $\mathbb{R}^{3m\times n}$ and would like to use a model to predict the classes of these images. We can flatten the 2-dimensional image to be a vector in $\mathbb{R}^{3mn}$ and feed it through an MLP. Where each pixel is a feature, however, this will lead to a drastic increase in the number of parameters as a function of m and n and would make the model harder to train. Furthermore, when dealing with 2-dimensional - or n-dimensional - data, we recognize that pixels that are spatially closer to each other share spatial coherence. This spatial representations would not be captured through a simple feedforward neural network. These drawbacks are few reasons why current state-of-the-art architectures predominantly rely on Convolutional Neural Networks [39] (CNN) to extract feature maps from high-dimensional data (e.g. image, audio, video).

2.0.3 Activation Functions

In prior section, we have introduced activation functions but we did not discuss the different types of activation functions and why some are used more heavily used than others. Given a hidden unit in a neural network, initially it computes a weighted sum of its input, were the value of the output can be anything ranging from $-\infty$ to ∞. This property makes it hard to articulate

to proceeding hidden unit that a certain hidden unit 'activated' or not. Hence, one purpose of activation functions is to non-linearly bound the the outputs to relay important information about the state of a hidden unit. Activation functions need to be non-linear as a neural network with many layers and comprised of only linear activation functions is just a linear function of the input of first layer. The design of activation functions does not yet have many definitive guiding theoretical principles. Although, there are practices that researchers often default to. In this section, we will go over some of those activation functions:

1. **Sigmoid Function**: is traditionally a very popular activation function for neural networks. The input to the function is transformed into a value between [0, 1], where inputs that are much greater than a certain value are transformed very closely to one, while, inputs that are much less a certain value are transformed very closely to zero. Unlike a linear function, the output of this activation function is bounded, hence, we would not have exploding activations on either side. On the other hand, the values on either sides tend to respond very less to changes in the input, hence the gradient at that region is going to be small (i.e. vanishing gradient).

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.2}$$

2. **Hyperbolic Tangent**: is very similar to the sigmoid function, in fact, $tanh(x) = 2\sigma(2x) - 1$. One distinction is that *tanh* gradient is stronger in magnitude, with the bounded range of the activations are between $[-1, 1]$. However, this also mean that similar to the sigmoid function, *tanh* is still susceptible to vanishing gradient problem.

$$tanh(x) = \frac{2}{1 + e^{-2x}} - 1 \tag{2.3}$$

3. **Rectified Linear Units (ReLUs)**: simply output the input if the input is positive, otherwise, it outputs 0. This means that the function is linear for values greater than zero. While the non-linear property of ReLU are in the negative values were negative values are always transformed to zero. These characteristics makes ReLU an attractive activation function for various reasons: (1) *computational cost*: ReLU is less computationally expensive than tanh and sigmoid functions because it does not require the use of an exponential calculation. (2) *Sparsity*: unlike tanh and sigmoid functions, ReLU is capable of outputting a true zero value. Meaning that some hidden units in the network are not activated and thereby making the activations sparse and efficient. This is a desirable property as it can accelerate learning, simplify the model, and overcome vanishing gradients problem.

$$ReLU(x) = \max\{0, x\} \tag{2.4}$$

   One major drawback of using ReLU is that they cannot learn on examples for which the input activations are less than equal to zero. For activations in the negative region of ReLU, the gradient will be zero, hence, the weights will not be adjusted based on that input. This means that some neurons can potentially get stuck in a state were variations

in the input always results in zero, often called *dying ReLU*. Various adjustments were proposed to ReLU to metigate this problem. For instance, absolute value rectification [71], leaky ReLU [72], or PReLU [5].

Convolutional Neural Networks (CNNs)

CNNs excel at efficiently processing data that has a known 2-D spatial structure. The name of CNNs originates from the network reliance on a mathematical operation called *convolutions*. LeNet-5 [39] were one of the early implementations of a CNN architecture, trained on MNIST dataset. The use of the convolutional operations typically occurs in the early layers where the input space is still of high-dimensions, however, as the input space is progressively reduced, there is a MLP that generally perform the final task e.g. classification. In this section, we will discuss operations that makes a CNN:

1. **Convolutional Layers** (Figure 2.3): Convolutional Layers consists of three components: input, kernel, and feature maps (output). The kernel is often smaller in size than the input but contain the same dimension. The kernel slides over the input, performing an element-wise multiplication with the part of the input it is currently on, and then summing up the results into a single output. The kernel repeats this process for every location it slides over, converting the input matrix of features into another matrix of features. The output features are the weighted sums of the input features located roughly in the same location of the output pixel on the input layer. We can mathematically define this operation as the following:

$$(f * g)(i) = \sum_{m}^{M} g(m) \cdot f(i - m) \tag{2.5}$$

where f is a 1D input vector, g is a 1D kernel vector, N is the length of f, and M is the length of g. We perform this operation for every $i \in N$. This operation can be generalized to 2D by convolving over more than one axis at a time:

$$(f * g)(i, j) = \sum_{m}^{M} \sum_{n}^{N} g(m, n) \cdot f(i - m, j - n) \tag{2.6}$$

The amount of movement the kernel steps per computation is determined by the value of *stride* (default = 1).

2. **Pooling Layers**(Figure 2.4): As mentioned previously, machine learning algorithms need to extract high-level task relevant information from high-dimensional low-level sensory data (e.g. images/audio). We extract those features through convolutions, however, as we progress to higher levels of representations, we need to reduce unnecessary spatial dependency of features. This is done through the pooling layers. The pooling layer is a down-sampling operation, which reduces the spatial size of generated feature maps. There are various types of down-sampling operations, however, the most widely adopted in literature is maximum pooling. Max pooling calculates the maximum, or largest, value in each patch of each feature map. Hence, the results are pooled feature maps that highlight the most dominant feature in the patch. Alternatively, average pooling involves calculating the average for each patch of the feature map.

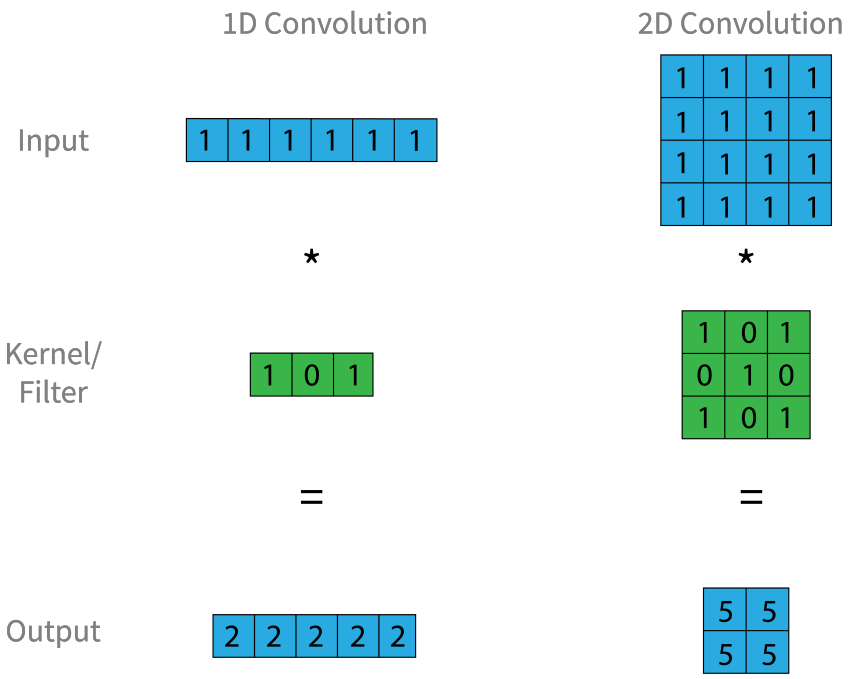

Figure 2.3: Computation of a convolutional layer. On the left, example of 1D convolution operation. On the right, example of 2D convolution operation.

There are other types of pooling methods that are used on the output of the last convolutional layer, called *global pooling*. Instead of down sampling patches of the input feature map, global pooling down samples the entire feature map to a single value. Global pooling is used on the output of the last convolutional layer as to aggressively summarize the presence of a feature and allow us to transition from feature maps to an output that can be used for predictions.

3. **Fully Connected Layers**: The last type of layers in a CNN is typically an MLP or fully connected layers. The input to the fully connected layer is the flattened or globally pooled output of the last convolutional layer. The output of the last convolutional layer is high-level features in low-dimensional space, adding a fully-connected layer is a cheap way of learning non-linear combinations of these features. Moreover, for a classification problem, this allows us to set the size of last fully connected layer to match the number of classes, where each node would output the probability that the input belongs to.

Figure 2.5 shows a basic CNN architecture given a sample image belonging to the frog class from CIFAR10 [4] dataset. The CIFAR-10 dataset consists of 60,000 32x32 coloured images in 10 classes, with 6,000 images per class. Each convolutional block in Figure 2.5 consists of a convolution layer, max pooling layer, and ReLU activation function. Although for CIFAR10 such architecture might suffice, progressively over the past decade, CNNs were built with deeper architectures to tackle more challenging datasets. One of those prominent datasets in Computer Vision is ImageNet [2]. ImageNet (or ILSVRC2012) consists of large hand-labeled 1.2 million natural images of 1000 categories. This evolution began with AlexNet [73]

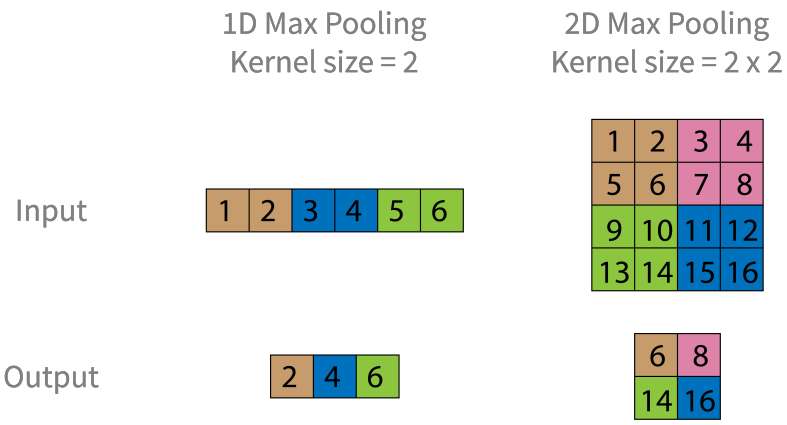

Figure 2.4: Computation of a max pooling layer. On the left, example of 1D max pooling operation. On the right, example of 2D max pooling operation.

breaking the state-of-the-art on ImageNet with 5 convolutional layers. Soon after, GoogleNet [74] and VGG [75] had 19 and 22 layers, respectively.

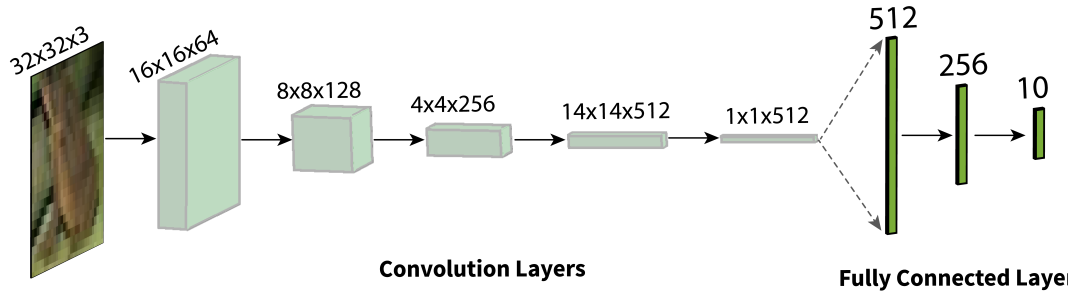

Figure 2.5: Example CNN. Each convolutional block consists of a convolution layer, max pooling layer, and ReLU activation function. The image is a sample belonging to the frog class from CIFAR10 [4] dataset.

ResNet Architecture

Although, increasing networks' depth improved performance on ImageNet, it also increased the difficulty to train those networks due to *vanishing gradient*. Vanishing gradient refers to the problem that the gradient in early layers approaches zero. For shallow network with only a few layers, this does not result in a big problem. However, when more layers are used, it can cause the gradient to be too small for training to work effectively. Early solutions to this problem introduced an auxiliary loss in a middle layer to re-flow the gradient [74]. However, a move direct solution to the problem was introduced in the ResNet Architecture [5].

ResNet is named after the residual blocks used within its architecture (Figure 2.7 and Figure 2.6). Figure 2.6 shows the residual connection directly adds the value at the beginning of the block, to the end of the block:

$$y = \mathcal{F}(x) + x \tag{2.7}$$

where x is the input, $\mathcal{F}$ is the convolutions operating on the input. By adding the identity value of x through the residual connection, the input would skip an activation function that could vanish the gradient. Thereby, stacking layers should not degrade the network's performance, because we could simply stack identity mappings that prevent the gradient approaching zero. This also means that deeper model should not produce a training error higher than its shallower counterparts, as letting the stacked layers fit a residual mapping is easier than letting them directly fit the desired under-laying mapping. Typical residual blocks do not change the spatial resolution of the input, however, as we discussed previously, we need to down-sample our representations. Hence, following n number of residual blocks without down-sample capability, we follow it up with one that down-sample the input x through both streams (2.6). Figure 2.7 shows an example architecture of ResNet18 and ResNet34, the difference between them is the number of residual connections without down-sampling capability. Following the initial ResNet architecture, the architecture over the past few years has been studied heavily with multiple variations [76, 77, 78]. This architecture is introduced because it will be used extensively in the next chapters.

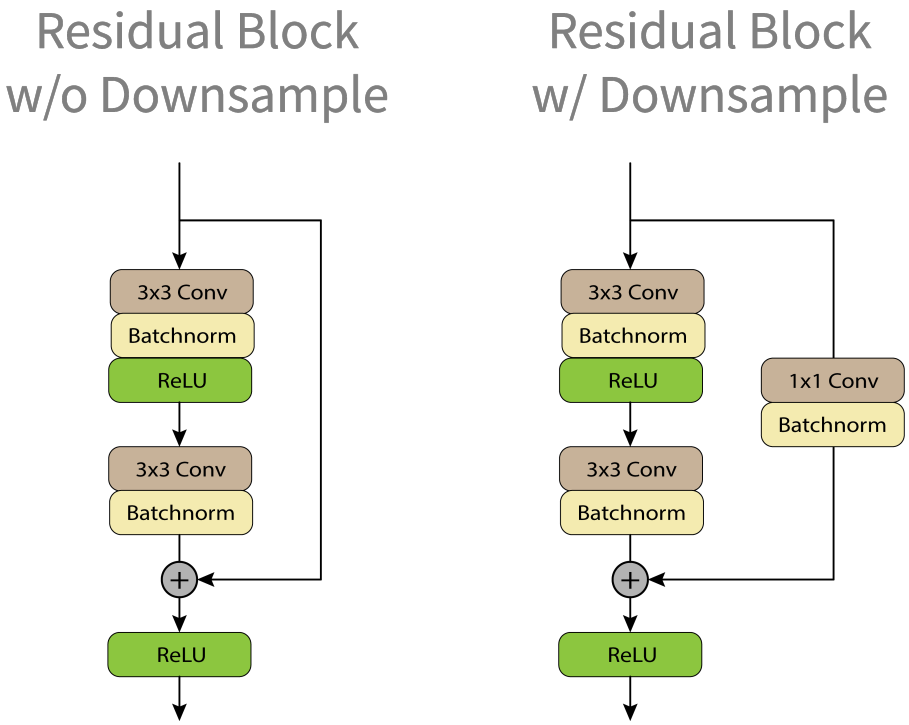

Figure 2.6: Residual blocks architecture [5].

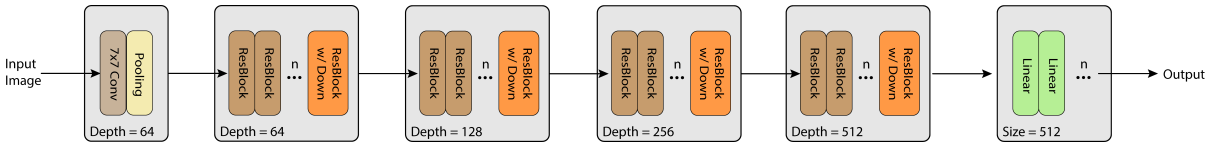

Figure 2.7: ResNet18/34 Architecture [5].

2.0.4 Batch Normalization

In statistics, normalization is the process of adjusting values measured on different scales to a common scale, and aligning a given data distribution to a normal distribution (i.e. mean of 0 and standard deviation of 1). Often in deep learning, we normalize the input data to the model by subtracting the data by the mean:

$$\begin{aligned} \mu &= \frac{1}{m}\sum_{i}^{m} x_i \\ x &:= x - \mu \end{aligned} \tag{2.8}$$

and normalizing the variance:

$$\begin{aligned} \sigma^2 &= \frac{1}{m}\sum_{i}^{m}(x_i - \mu)^2 \\ x &:= \frac{x}{\sigma^2} \end{aligned} \tag{2.9}$$

This is beneficial because different features might have contrasting ranges in value, hence, it can hinder the performance or training speed and can result in the model not learning at all. Batch normalization [79] reinforces this idea by performing such operation not only on the input data, but also in the intermediate layers of the network (Figure 2.6). Generally, in some way, each layer in a deep neural network aims to 'fit' the input data to some representations. However, the flow of data through the layers often create drift in the distributions. Thereby, making it harder for the proceeding layers to learn meaningful representations, as they would need to account for the drifts. Another problem that batch normalization aims to solve is *exploding and vanishing gradients*. Some activation functions might be pushed to their saturated areas, which could result in making some representations non-changing.

Batch normalization (Algorithm 1) solves those problems by normalizing each intermediate layer's inputs with the dataset mean and standard deviation over a given mini-batch. One distinction between input data normalization and batch normalization is that the statistics (mean and variance) applied over a mini-batch rather than on the entire dataset.

2.1 Related Works

2.1.1 Representation Learning

Learning effective visual representations without human supervision is a long-standing problem. Most mainstream approaches fall into one of two classes: generative or discriminative. In this section, we will describe both approaches.

Generative Models for Representation Learning

Auto-Encoders (Figure 2.8) are neural networks with two sub-networks [80]: (i) one that maps a given data sample into lower dimension representation (encoder), and (ii) the other sub-net-

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x_i} \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x_i} + \beta \qquad \text{// scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch [79].

work receives those representations and reconstructs them back into the given data sample (decoder). These family of neural networks are not designed to be generative instead they were aimed to compress high dimensional data into a smaller representation given no labels (i.e. unsupervised representational learning). When network is given a data input x, the encoder encodes it into a latent representation z and the decoder uses that latent representation to reconstruct the data input $\overline{x}$. Often to train such network the cost function is defined as the mean squared error between the input x and the reconstructed $\overline{x}$ data sample:

$$f(x, \overline{x}) = \|\overline{x} - x\|^2 \tag{2.10}$$

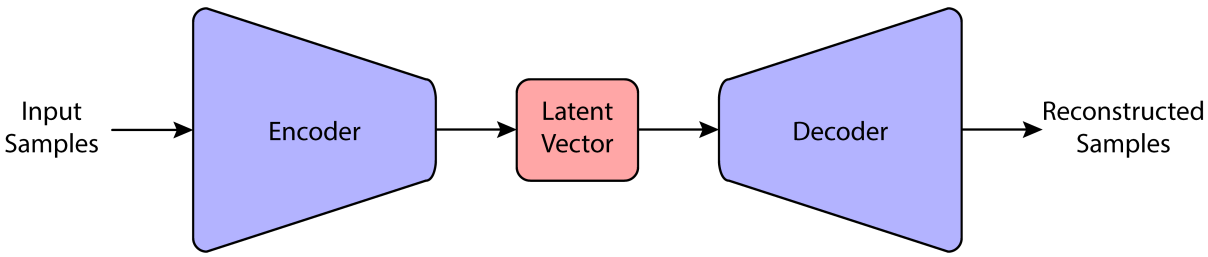

Figure 2.8: Auto-Encoders Architecture. composed of two sub-networks i.e. the encoder and the decoder.

Although, Auto-Encoders are generally successful at reconstructing data with high quality, often because of the high degree of freedom over the latent code, the training objective leads to a severe over-fitting in the latent space. That is, a small subset of the latent space which is identified by the encoder will yield meaningful content once decoded. However, if a random latent code is fed into the decoder, with high probability it will reproduce a meaningless content. Various methods have introduced measures to improve Auto-Encoders. Denoising Auto-Encoders [81] was one of the early proposed improvements to reduce over-fitting by partially corrupting the input by either adding noise or masking content and training the model to recover the

original uncorrupted input. More recent methods not only improved on the robustness but also reinforced the generative properties of Auto-Encoders [82, 83, 84, 85, 86, 87, 88].

Generative Adversarial Networks (GANs) over the past few years have been the most popular generative model framework achieving state-of-the-art across various domains of data [89, 90, 91, 92, 6, 93]. GANs provide an implicit alternative to modeling the density function, which allows us to synthesize samples from $p(x)$. The GAN framework is a min-max adversarial game between two distinct neural networks: (i) The generator (G), aims at generating synthetic data by learning the distribution of the real data and (ii) the discriminator (D), aims at distinguishing the generator's fake data from real data (Figure 2.9). The generator uses a function $G(z)$ that maps latent vector z from the prior $p(z)$ (a Gaussian distribution) to the data space $p(x)$. $G(z)$ is trained to maximally confuse the discriminator into believing that samples it generates come from the training data distribution. While, the discriminator tries to accurately predict whether the given samples are from the training data (real) or generated from $G(z)$ (fake). The solution to this game can be expressed as following [94]:

$$\min_G \max_D \Big[\mathbb{E}_{x\sim p_{data}}[log\ D(x)] + \mathbb{E}_{z\sim p_z}[log\ (1 - D(G(z)))] \Big] \tag{2.11}$$

where the training procedure for D is to maximise the probability of assigning correct labels for training examples from the data and examples from G. The training procedure for G is to maximise the probability of D making a mistake, or in other words to minimise $log\ (1 - D(G(z)))$, where z is the prior input noise variables and x is the data. Figure 2.10 demonstrate the capability of GANs at generating synthetic images.

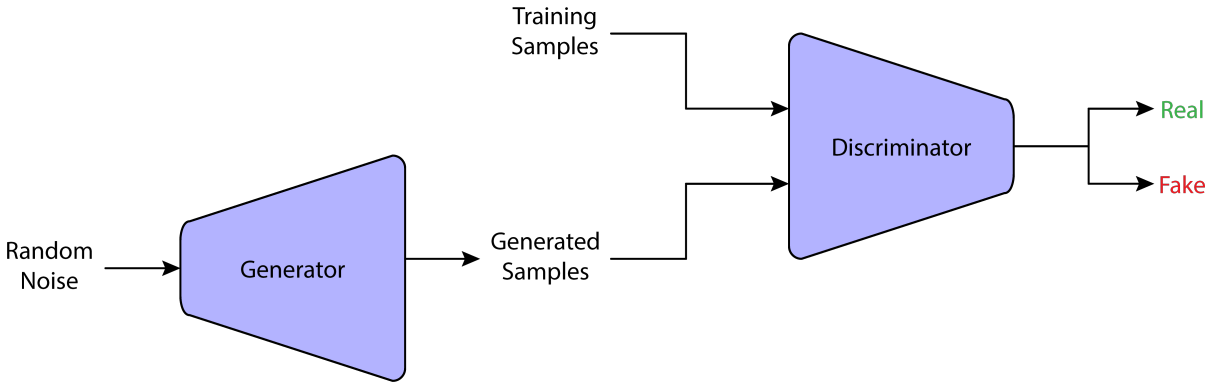

Figure 2.9: GAN Architecture.

In both generative frameworks, the subsequent discriminative/encoding sub-network is extracted and utilized as pre-trained model. Hence the major disadvantages of using such approach to representation learning is that pixel-level generation is computationally expensive to train and generating images of high-fidelity may not be necessary for representation learning.

2.1.2 Discriminative Models for Representation Learning

As described earlier self-supervised learning is a type of unsupervised learning that uses the inherent structure of the data to learn meaningful representations. Generative models can be

Figure 2.10: Examples of randomly generated images using BigGAN [6] a GAN variant.

thought as self-supervised, but with different objectives. Discriminative self-supervised learning exploit priors about the data and treat them as labels. The self-supervised task guides the training through a supervised loss function, however, we do not directly use the loss for final performance evaluation. Rather we investigate the learned intermediate representations at extracting meaningful information from our data. In this section, we will introduce some of the self-supervised methods.

Augmentations

Data Augmentations has been used extensively in Computer Vision literature to reduce overfitting to the training data, especially during supervised training. The augmented data often provide a more comprehensive range of possible data points, thus minimizing the distance between the training and testing sets. The underlying assumption is that small distortion on an image should not modify the semantic meaning of an image. For instance, if we have an image of a dog and we rotate the image over the horizontal axis, we would still expect it to be an image of a dog. [7] utilized such property to create surrogate classes using data augmentations to perform self-supervised learning, called Exemplar-CNN. Exemplar-CNN was one of the early methods that were able to out-perform prior unsupervised methods for learning image representations. The framework crops 32×32 patches from images and apply a set of augmentations randomly with varying degree (Figure 2.11). Exemplar-CNN utilized in total six augmentations:

1. **Translation**: vertical and horizontal translation by a distance.
2. **Scaling**: multiplication of the patch scale by a factor.
3. **Rotation**: rotation of the image by an angle.

4. **Contrast 1**: multiply the projection of each patch pixel onto the principal components of the set of all pixels by a factor.

5. **Contrast 2**: raise saturation and value of all pixels to a power.

6. **Color**: add a value to the hue of all pixels in the patch.

The training objective is to classify these patches according to their augmented surrogate classes, where surrogate classes can be created arbitrarily. For instance, the magnitude of the parameters for each augmentation has a finite space of values between the boundaries of parameters. Hence, if all of the augmentation have 5 parameters, we can create 5^6 surrogate classes.

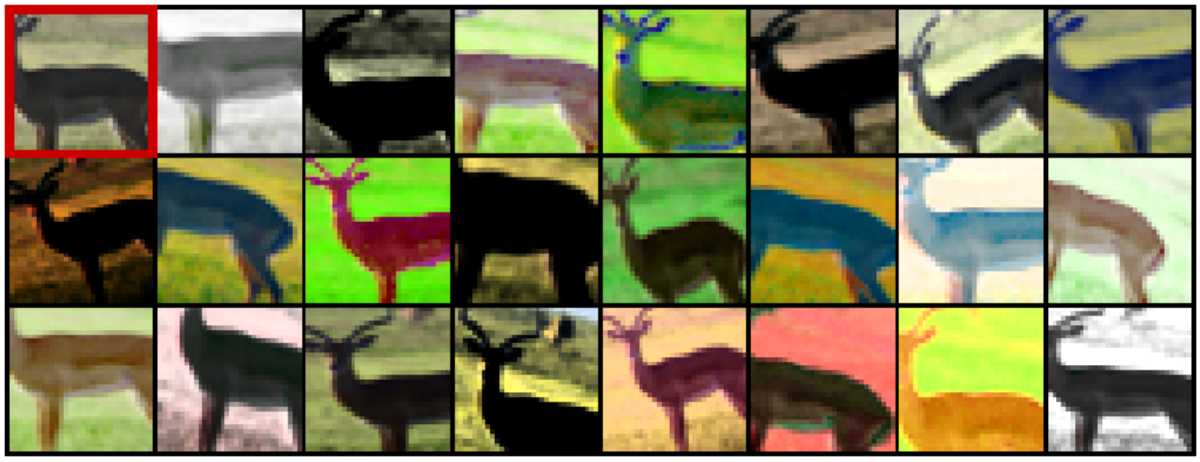

Figure 2.11: Random augmentations applied to the original patch (top left corner) utilized in [7].

More recently, the authors of [8] demonstrated a more simple approach for utilizing augmentations to learn visual representations. Compared to previous framework, the work in [8] introduces a more cheap process, where each input image is rotated by a multiple of 90°at random, corresponding to [0°, 90°, 180°, 270°]. As such, the training objective is to predict the degree of rotation that has been applied to the input image (thus it is considered a 4-class classification problem; Figure 2.12). The intuition behind using these image rotations (as the only form of augmentation) relates to the idea that in order to effectively predict the degree of rotations, a model has to first learn to recognize slight differences between classes of objects as well as their semantic parts in images. That is, a model must necessarily learn to localize objects in the image, recognize their orientation and object type, and then relate the object orientation with the dominant orientation that each type of object tends to be depicted within the data distribution.

Patches

While the previously introduced self-supervised methods utilized augmentations to derive the objective function, the proposed method in [9] employs the spatial relationship between patches

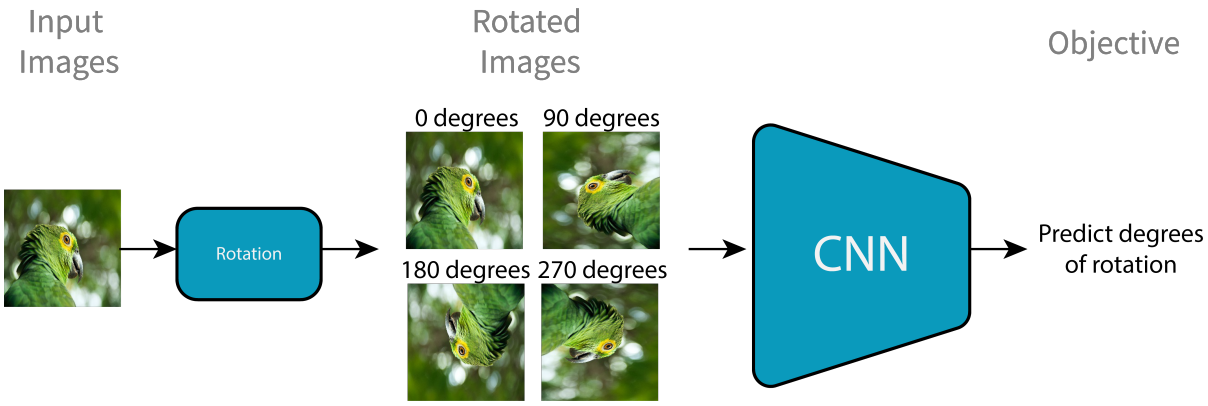

Figure 2.12: Demonstrates the framework proposed by [8].

of an image to derive a training objective. The training objective in [9] is to accurately predict the relative position between two random patches from an image. Similar to the rotation method, to accomplish this task effectively, a model would need to learn to recognize differences between classes of objects as well as their semantic parts in images. For instance, if one of the patches was a cat ear and the second patch was the eye of the cat. The model would need to first recognize that those patches are part of a cat and that they should be relatively close to each other.

The method starts by randomly sampling the first patch from the original image, followed by sampling a 3x3 grid of patches with the first patch located in the middle of the grid. By using the first patch as the point of reference, and randomly selecting one of the 8 neighboring as the second patch, the model would predict which one of the 8 neighboring locations the second patch is selected from (Figure 2.13). To retrain the model from capitalizing on low-level trivial signals, such as textures continuing between patches, chromatic aberration, and pixelation, additional noise is introduced. This is done by: (1) adding additional gaps between sampled patches, (2) randomly jittering each patch location by up to 7 pixels, (3) randomly down-sampling some patches to as little as 100 total pixels, and then up-sampling, and (4) randomly dropping 2 of the 3 color channels. Chromatic aberration is triggered by different focal lengths of lights at different wavelengths passing through a lens. Hence, such process can create small offsets between color channels where a model can exploit. Chromatic aberration serves as an excellent example to how models can exploit some underlying features and when utilizing self-supervised learning this can be severe.

While [9] demonstrated the process on 2 patches, [95] built on the work to generalize to all 9 patches, in a task called jigsaw puzzle. In jigsaw puzzle, the model processes each patch independently and outputs a probability vector per patch index out of a predefined set of permutations. To control the difficulty of jigsaw puzzles, the paper proposed to shuffle patches according to a predefined permutation set and configured the model to predict a probability vector over all the indices in the set.

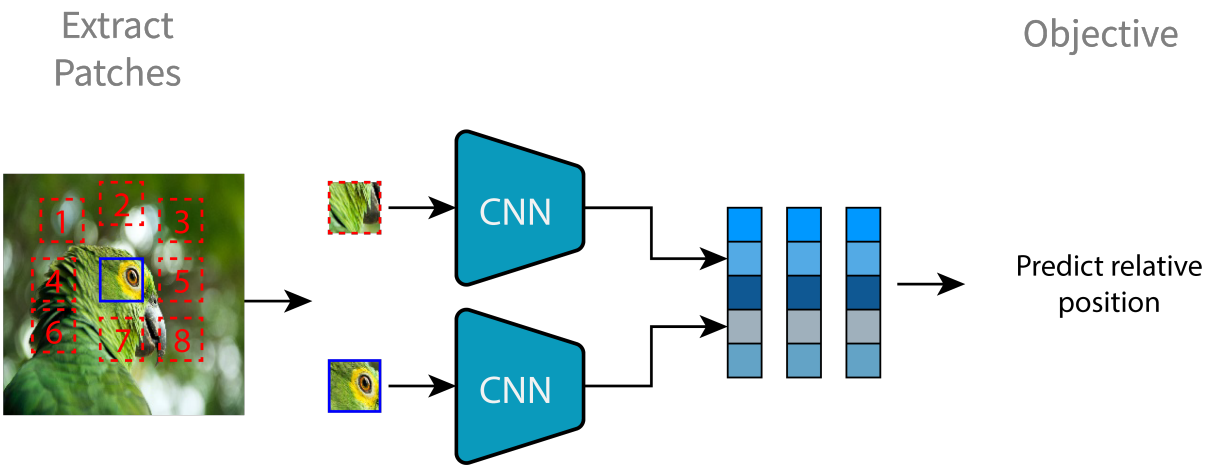

Figure 2.13: Demonstrates the framework proposed by [9].

Contrastive Learning

Previously introduced methods performed self-supervision by utilizing data driven objectives as a classification problem. This means that such approach although successful in one data domain, can become insufficient just by changing the data type (e.g. from images to audio). For instance, although we can use rotation to learn visual representations [8], we would not be able to utilize similar framework in the audio domain. More recent approaches to self-supervised learning focus on instance-level discrimination, where each sample/instance is treated as a distinct class of its own. Suppose we have a CNN (f) that project high-dimensional input to low-dimensional feature space v, such that $v_i = f(x_i)$. With such framework, we are interested in minimizing the distance of these low-dimensional features for visually similar images closer to each other, that is, $d(x, y) = \|f(x) - f(y)\|$. This can be formulated using the softmax function as the following:

$$P(i|v) = \frac{exp(w_i^T \cdot v)}{\sum_{j=1}^{n} exp(w_j^T \cdot v)} \tag{2.12}$$

where v is feature vector of a given sample x ($v = f(x)$), $P(i|v)$ is the probability of the sample being recognized as the i-th example, w_j is a weight vector for class j, and $w_j^T v$ measures how well v matches the j-th instance. Such approach is problematic in an instance-level discrimination, as the weight vector serves as a class prototype, thereby, preventing explicit comparisons between instances. The authors in [10] built on such work by proposing a non-parametric variant (called contrastive loss) of Eq. 2.12 that replaces $w_j^T \cdot v$ with $k \cdot v$, such that the probability $P(i|v)$ becomes:

$$P(i|v) = \frac{exp(k \cdot v/\tau)}{\sum_{j=1}^{n} exp(k \cdot v/\tau)} \tag{2.13}$$

where k is the feature vector from memory bank, and τ is a temperature parameter that controls the concentration level of the output distribution, proposed in [96]. The denominator of Eq. 2.13 require the feature vectors for all images, hence, rather than exhaustively computing these

representations, [10] utilize a feature memory bank for storing them as shown in Figure 2.14. During each learning iteration, the parameters are optimized based on the representations then stored at the corresponding instance entry in the memory bank. Hence, the memory bank will consist of the representations of all samples in the dataset. It is worth noting that prior to [10], [97] introduced earlier variant of the contrastive loss (or InfoNCE), which defined the prediction task by splitting samples into patches and measuring how well the model can classify representations of close patches amongst a set of unrelated samples.

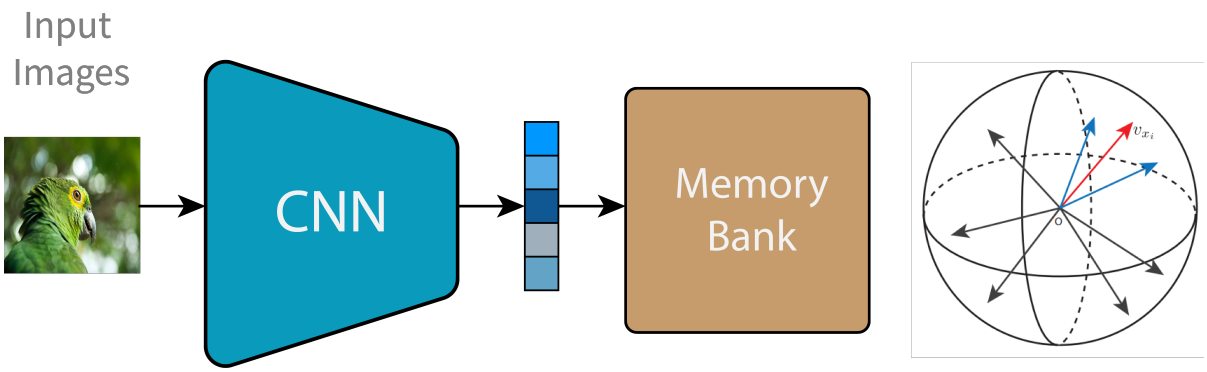

Figure 2.14: Demonstrates the framework proposed by [10]

.

One major drawback of using memory bank is that the representation of samples in the memory bank can be from past epochs, thus are less consistent with the encoder's current state. For instance, assume we are at epoch n and we compute the loss for the first mini-batch v. We will still be using feature vectors from epoch $n-1$ (i.e. k). The framework proposed in [11] (called Momentum Contrast or MoCo) enhances such approach by incorporating another encoder (called momentum encoder) which not only outperforms the memory bank method in regard to the learned representations but also is more memory-efficient and can be trained on billion-scale data. MoCo framework is motivated by the fact that these methods can be thought of as building dynamic dictionaries, where the keys in the dictionary are sampled from data and are represented by an encoder network. Hence, an encoded query should be similar to its matching key and dissimilar to others (Figure 2.15). This is accomplished by incorporating two components:

1. **Dictionary Queue**: rather than using a memory bank that will span the whole dataset, MoCo adopt a queue were samples in the dictionary are progressively replaced. Hence, the dictionary always represents a sampled subset of all data, without extra computation. Dictionary size can be much larger than the size of a mini-batch, and is seen as a hyper-parameter.

2. **Momentum Encoder**: [9] uses samples in the current mini-batch as the dictionary, so the keys are consistently encoded. However, such approach is limited as the dictionary size is coupled with the size of the mini-batch.. MoCo resolves this issue by adopting a momentum update to the key encoder. It can be denoted as:

$$\theta_k \leftarrow m\theta_k + (1-m)\theta_q \tag{2.14}$$

where $m \in [0, 1)$ is a momentum coefficient, θ_k are the parameters for the momentum encoder, and θ_q are the parameters for the query encoder. Only θ_q are updated by back-propagation, θ_k evolves smoothly through Eq. 2.14.

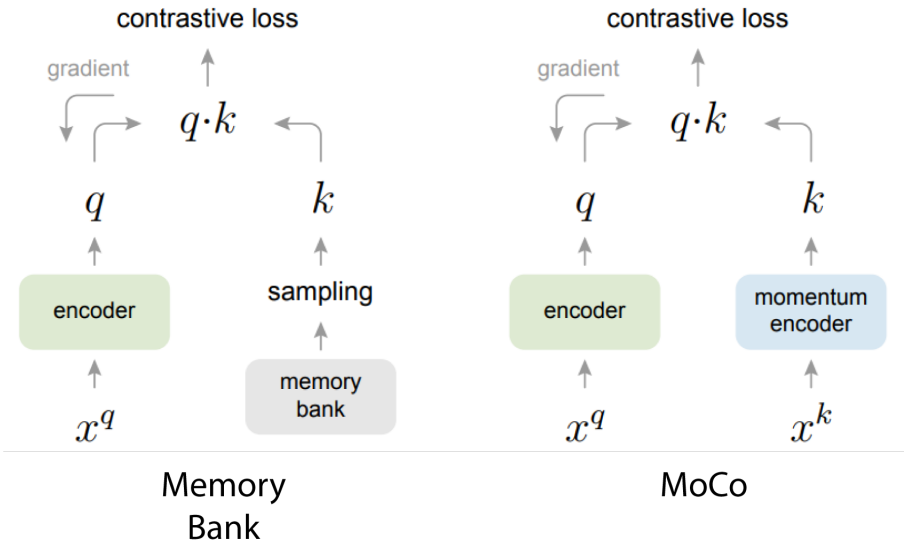

Figure 2.15: Conceptual comparison of [10] and [11]. On the left, key representations are sampled from a memory bank [10]. On the right, shows Moco encoding the new keys on-the-fly by a momentum-updated encoder, and maintains a queue of keys [11].

Following MoCo's work in learning efficient representations from unlabelled data, [12] not only introduced a novel method (SimCLR) but also investigated various components of how to improve performance that can be beneficial to other self-supervised methods. SimCLR has been shown to not only outperform previous self-supervised methods on ImageNet but also outperform supervised methods on some natural image classification datasets [12]. In essence, the framework aims at learning efficient representations by maximizing agreement between differently augmented views of the same data and maximizing difference across contrasting images via contrastive loss in the latent space (Figure 2.17). SimCLR consists of three major modules:

1. **Augmentations Module**: transforms any given data example randomly resulting in two augmented views of the same example, denoted $\hat{x}_i$ and $\hat{x}_j$, which are considered the positive pairs. SimCLR investigated several augmentations for images and their effect on the learned representations. One type of augmentations involves spatial transformations such as cropping and resizing, rotation [8], flipping, and cutout [98]. The other types of augmentations involve appearance transformations, such as color distortion, Gaussian blur, and sobel filtering (Figure 2.16). Through comprehensive search, [12] found that random cropping followed by resizing back to the input image size, random color distortion, and random Gaussian blur yield the most efficient representations.

2. **Encoder Module** - $f(\cdot)$: maps the high-dimensional input images to low-dimensional

feature space from augmented examples. Where $h_i = f(\hat{x}_i)$ and $h_j = f(\hat{x}_j)$, such that $h_{i,j} \in \mathbb{R}^d$.

3. **Projection Head Module** - $g(\cdot)$: maps encoder representations to representations space where contrastive loss is applied. It was shown that MLP with ReLU non-linearity is crucial in this method. Compared to using a linear counterpart or no projection head, the non-linearity significantly improve the quality of the representations obtained. That is, $z_i = g(h_i)$ and $z_j = g(h_j)$, such that $z_{i,j} \in \mathbb{R}^d$. The output dimensionality of z was shown to not change the performance significantly.

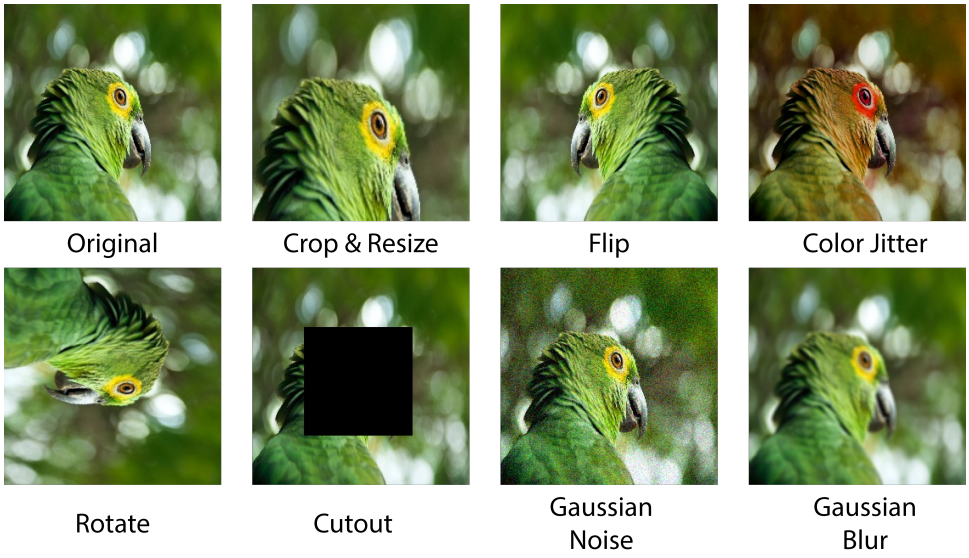

Figure 2.16: Illustrations of the studied data augmentation in [12]. Each augmentation can transform data with some internal parameters (e.g. rotation degree, noise level).

The loss for such objective is termed Normalized Temperature-scaled Cross Entropy Loss (NT-Xent):

$$l_{i,j} = \frac{\exp\left(\text{sim}\left(\mathbf{z}_i, \mathbf{z}_j\right)/\tau\right)}{\sum_{k=1}^{2N} \mathbf{1}_{[k\neq i]} \exp\left(\text{sim}\left(\mathbf{z}_i, \mathbf{z}_k\right)/\tau\right)} \tag{2.15}$$

where $\mathbf{1}_{[k\neq i]} \in 0, 1$ is an indicator function returning 1 iff $k \neq i$, τ denotes a temperature parameter (default is 0.5), and $\mathbf{z}$ denotes the encoded representations of a given augmented view. N is the number of training samples within a mini-batch, (i, j) are positive pairs of each sample. The loss is computed across all positive pairs, in a mini-batch. $\text{sim}(\mathbf{u}, \mathbf{v}) = \mathbf{u}^T\mathbf{v}/||\mathbf{u}||||\mathbf{v}||$ denotes the cosine similarity between two vectors $\mathbf{u}$ and $\mathbf{v}$.

The advantage of MoCo compared to SimCLR is that MoCo decouples the batch-size from the number of negative samples, but SimCLR requires a large batch-size in order to have enough negative samples. Hence, SimCLR requires large batch-sizes to reach competitive performance. With SimCLR proposed improvements such as the projection head and stronger data augmentation, MoCoV2 [99] was introduced with better performance and no dependency on large batch-sizes.

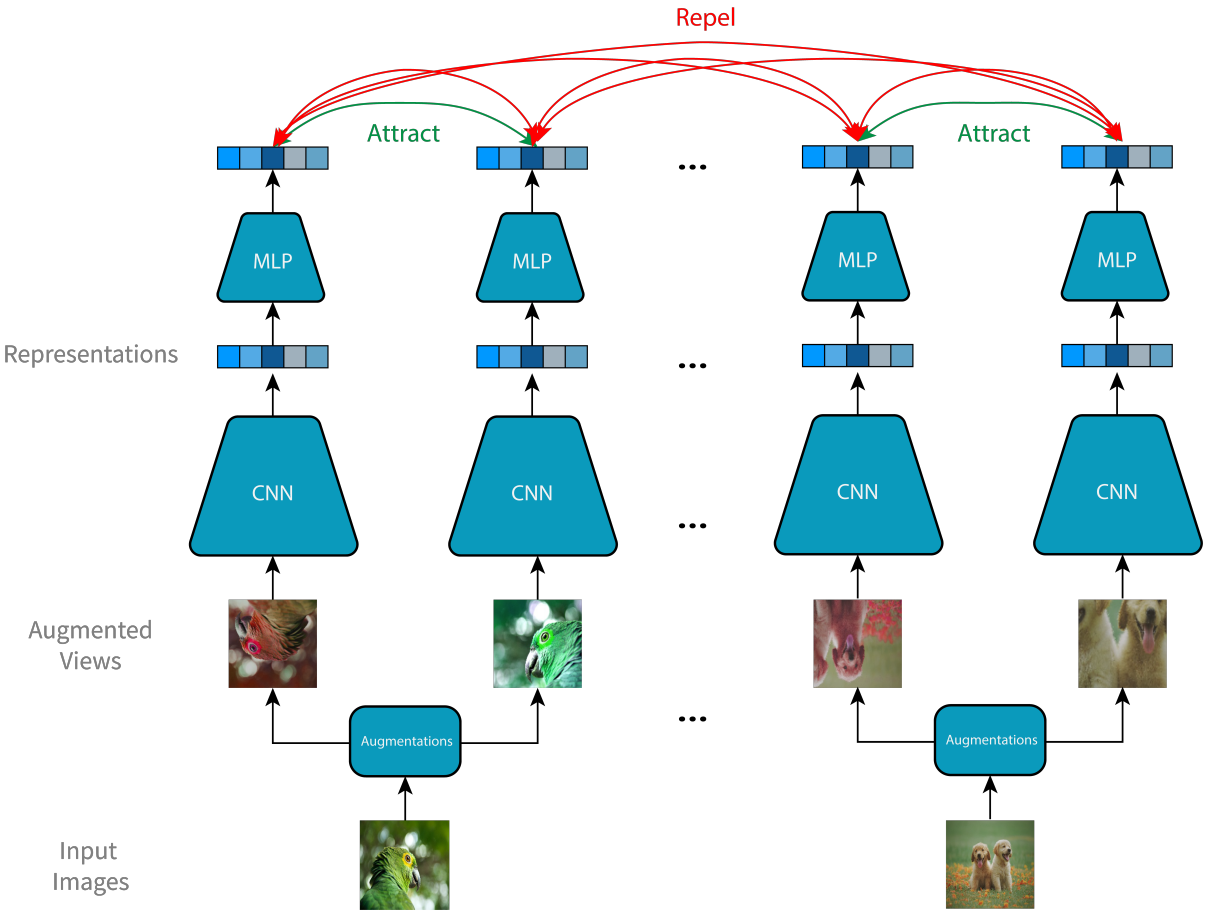

Figure 2.17: SimCLR framework proposed by [12].

Chapter 3

Methodology

3.1 Audio Pre-porcessing

We trained two families of models, one that takes as input raw audio signals while the other utilizes time-frequency audio features as input. For both models, we start by down-sampling (when applicable) all audio signals to 16 kHz, followed by signal padding (by zeros) or clipping the right side of the signal to ensure that all audio signals are of the same length. The target length of the audio signal is set based on the datasets' assigned audio length (Section 3.3). For models dependant on time-frequency features, we computed the short-time Fourier transform (STFT) magnitudes and phase angles of the input audio with 16 ms windows and 8 ms stride [100], further we projected the STFT to 128 frequency bins equally spaced on the Mel scale (Figure 3.1). Moreover, we computed the log-power of magnitude STFT and Mel spectrogram (Eq. 3.1) and stacked the three time-frequency features in the channel dimension, resulting in a matrix of size: $3 \times F \times T$. Where F is the number of the frequency bins and T is the number of frames in the spectrogram. This was done to ensure that we have comprehensive features that could capture multi-domain audio signals (e.g. speech, environmental, and music sounds) and would not require us to change the baseline ResNet 2D architecture. To compute the spectrograms, we utilized *nnAudio* a neural network based audio processing framework that leverages 1-D Convolutional Neural Network to transform time-domain audio signal to frequency-domain spectrograms on the GPU [101].

$$f(S) = 10log_{10}|S|^2 \tag{3.1}$$

where S is the mel-spectrogram or magnitude STFT.

3.2 Training/Evaluation Protocol

To maintain consistency and allow for a fair comparison across supervised, self-supervised and our proposed method (CLAR), we adopted four components in our framework (see Figure 3.2):

1. **Data Augmentation** module generates two random views of each sample, which contains an augmented information of the original sample. Section 3.5 describes all the various augmentations that were used in our experiments and Section 4.2 provides the

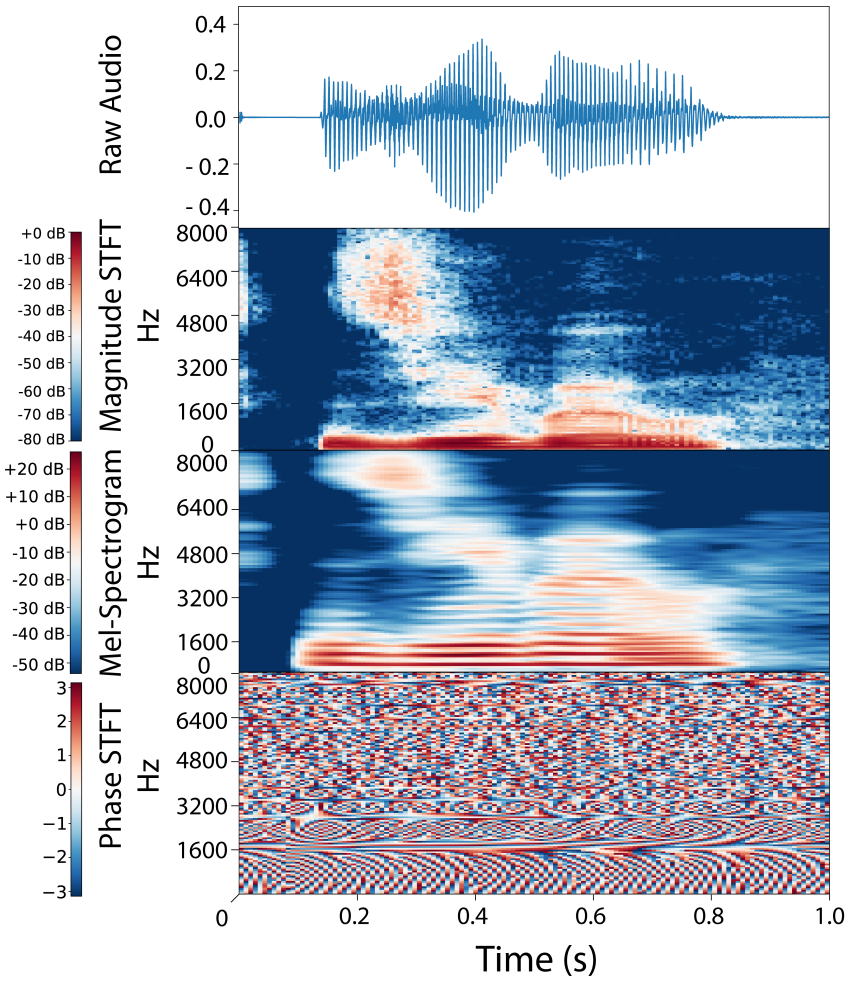

Figure 3.1: Raw audio (first) with the subsequent time-frequency features, specifically, STFT (short-time Fourier transform) magnitudes (second) and phase angles (fourth), and Mel-spectorgram (third). The raw audio and time-frequency features were utilized in training 1D and 2D versions of ResNet.

effect of these augmentations on auditory classification task both with raw audio and extracted time-frequency audio features.

2. **Encoder** maps the data samples into a representational vector. To vectorize our representations we performed adaptive average pooling on the output of the encoder. For our encoder, we trained 1D and 2D variants of standard ResNet18 [102] with SimCLR training protocol. For models trained on time-frequency audio features (i.e. spectrograms) as input, we utilized a typical ResNet18 [20] with random initialization. Alternatively, we switched all ResNet18 operations such as convolutions, max-pooling and batch normalization from 2D to 1D for models that takes raw audio signal as input. The output of both models is a 512 dimension vector.

3. **Projection head** maps the extracted encoder representations to a space where contrastive

and supervised loss is computed. Projection head consists of three fully connected layers with ReLU activation functions. We calculated the losses on the output of the projection head with fixed vector size of 128. In the supervised approach we replaced the final linear layer used for contrastive loss with a linear layer with the size of the class numbers to compute cross entropy loss. Lastly, in our proposed approach, in addition to the cross entropy loss computed on the last layer, we computed contrastive loss using the representations in the layer preceding the last layer.

4. **Evaluation head** was used to replace the projection head after training the encoder using different training methods. The evaluation head is a linear classifier trained on top of the frozen encoder to assess the learned representation quality by computing the test accuracy. When limiting labeled data to compare performance across supervised, self-supervised and our proposed approach (Section 4.4), we trained the evaluation head on the full labeled data. This approach is commonly adopted to evaluate the learned representations of self-supervised methods [12, 103, 24].

We trained all models with 1024 batch size, layer-wise adaptive rate Scaling (LARS) optimizer [104] with learning rate of 1.0, weight decay of 10^{-4}, linear warmup for the first 10 epochs, decay of the learning rate with the cosine decay schedule without restarts [105] and global batch normalization. For some datasets, we reduced the batch size to 512 to be able to fit the data in memory. We trained all our models on augmentations that we found to yield the best performance on test accuracy. All models were trained from random initialization with 4 NVIDIA v100 Tesla 32GB GPUs.

3.3 Datasets

In this work, we evaluated the performance of our proposed framework on three audio datasets from different domains (speech, music, and environmental sounds). All datasets have predefined train-validation-test splits by authors (except the ECS dataset), we used the test splits to compute analysis. The datasets are:

- **Speech Commands** (Speech): composed of 105,829 16kHz single-channel audios [106] from 2,618 speakers. Each audio file contains a one second recording of a single spoken English word from limited vocabulary. The dataset contains 35 labels (words) such as one-digit numbers, action oriented words, and arbitrarily short words. In addition to the full dataset, we derive a simpler version (~20k samples) with only the utterances of the one-digit numbers.

- **NSynth** (Music): contains 305,979 four seconds audio of musical notes, each with a unique pitch and musical instrument family [107]. For every musical note, the note was held for the first three seconds and allowed to decay for the final second. Similar to Speech Commands, we composed two variations of the same dataset, (1) we utilized musical instrument family as the class labels (11 classes) and (2) we used pitch as the class labels (128 classes). Both variations of NSynth dataset included the same amount of data, however, the number of classes varied.

- **Dataset for Environmental Sound Classification** (Environmental): consists of two variants ESC-10 and ESC-50 provided by the authors. Similar to previous datasets described, the two variants describe the number of classes. The ESC-50 dataset consists of 2,000 5-seconds environmental recordings equally distributed across 50 classes (40 clips per class). Classes such as animal, natural and water, non-speech human, interior and exterior sounds [108]. The ESC-10 is a subset of ESC-50 consisting of 400 recordings, making it the dataset with the least number of training data in our collection. Both datasets are divided into 5 folds by the authors. In this work, we utilized the first 4 folds for training and the last for testing.

3.4 CLAR Framework

Contrastive and supervised learning share the common goal of constructing representations that distinguish samples for different tasks. The supervised approach focuses on distinguishing samples from multiple classes without constraints on the latent representations. While, contrastive learning constructs such representations between paired views from samples with the constraint being that latent representations of negative views (from different samples) are maximized and positive views (from same samples) are minimized. These two frameworks have their own advantages and disadvantages. For instance, constrastive learning benefits from larger batch sizes and longer training [109, 12]. However, the supervised approach is simpler to optimize, hence, requires less training to achieve relative performance [12]. To improve the performance of self-supervised contrastive learning, the authors combined the shared representations from both self-supervised and supervised frameworks by performing self-supervised pre-training followed by supervised fine-tuning on labeled examples [12, 65, 66, 67]. However, this could result in problems such as *catastrophic forgetting*, especially in smaller networks [68, 69, 70] and makes the training more difficult as there are two stages that would need to be optimized. In this work, we abolish the fine-tuning step and integrate both contrastive and supervised learning frameworks simultaneously during training:

$$L = \mathcal{L}_{CL} + \mathcal{L}_{CE} \tag{3.2}$$

where $\mathcal{L}_{CL}$ is the contrastive loss and $\mathcal{L}_{CE}$ is the Categorical Cross-Entropy (CE) loss of the labeled samples. In CE loss, in cases where the labels for some of the samples within the mini-batch are missing, then the CE loss will be set to zero. Alternatively, contrastive loss is always applied as it is not dependent on the labels. A possible approach to guarantee labeled samples within a mini-batch is to use stratified sampling, this is especially important when the labeled data is substantially small portion of the whole dataset (e.g. 1% of the data is labeled). In our analysis, we utilized random sampling because (1) datasets utilized are not large, especially when compared with ImageNet and (2) we utilized very large batch size (1024). Using the projection head, we apply the $\mathcal{L}_{CE}$ loss on the last layer of the projection head and the $\mathcal{L}_{CL}$ loss on the layer preceding the last layer of the projection head (Figure 3.2).

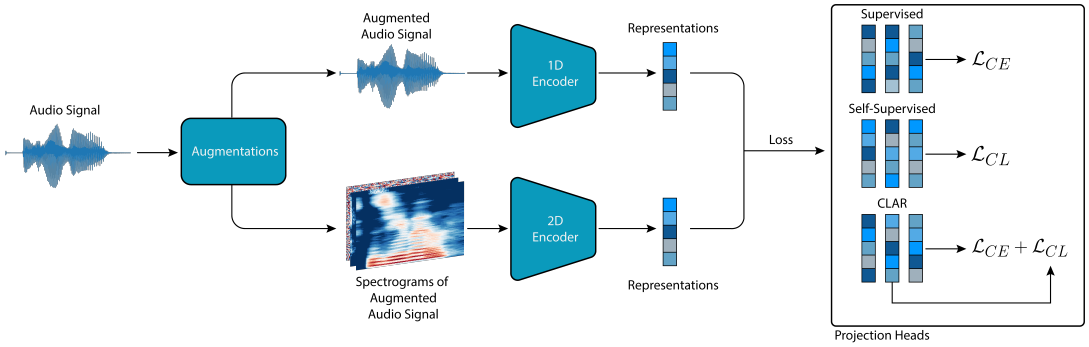

Figure 3.2: Training pipeline. We start by applying augmentations directly to the audio signal followed by either feeding the augmented audio signal or the spectrograms of the augmented audio signal to the subsequent encoder. We feed the representations from the encoder to the same projection head architecture across all approaches. However, we change the loss and the the layer representation employed in the loss computation.

3.5 Augmentations

To investigate the impact of various data augmentations suitable for learning auditory representations, we deployed six distinct augmentations (Figure 3.3). Each augmentation was applied directly to the audio signal. Augmentations that directly influence spectrograms were not included [60] to ensure that we could make direct comparison of augmentations performance both with the 1D and 2D models. In each iteration during training, each augmentation has a hyper-parameter which is randomly sampled from a uniform distribution that result in either a degree of data transformation or none at all. Introduced augmentations could be categorised as either frequency or temporal transformations:

1. Frequency Transformations

   (a) **Pitch Shift (PS)**: randomly raises or lowers the pitch of the audio signal [110]. Based on experimental observation, we found the range of pitch shifts that maintained the overall coherency of the input audio was in the range [-15, 15] semitones.

   (b) **Noise Injection**: mix the audio signal with random white, brown and pink noise. In our implementation, the intensity of the noise signal was randomly selected based on the strength of signal-to-noise ratio. We adopted two versions of this augmentation: (1) applied only white noise with varying degree of intensity (White Noise), (2) applied either white, brown, or pink depending on an additional random parameter sampled from uniform distribution (Mixed Noise).

2. Temporal Transformations

   (a) **Fade in/out (FD)**: gradually increases/decreases the intensity of the audio in the beginning/end of the audio signal. The degree of the fade was either linear, logarithmic or exponential (applied with uniform probability of 1/3). The size of the

fade for either side of the audio signal could at maximum reach half of the audio signal. The size of the fade was another random parameter picked for each sample.

(b) **Time Masking (TM)**: given an audio signal, in this transformation we randomly select a small segment of the full signal and set the signal values in that segment to normal noise or a constant value. In our implementation, we not only randomly selected the location of the masked segment but also we randomly selected the size of the segment. The size of the masked segment was set to maximally be $1/8$ of the input signal.

(c) **Time Shift (TS)**: randomly shifts the audio samples forwards or backwards. Samples that roll beyond the last position are re-introduced at the first position (rollover). The degree and direction of the shifts were randomly selected for each audio. The maximum degree that could be shifted was half of the audio signal, while, the minimum was when no shift applied to the signal.

(d) **Time Stretching (TST)**: slows down or speeds up the audio sample (while keeping the pitch unchanged). In this approach we transformed the signal by first computing the STFT of the signal, stretching it using a phase vocoder, and computing the inverse STFT to reconstruct the time domain signal [110]. Following those transformations, we down-sampled or cropped the signal to match the same number of samples as the input signal. When the *rate* of stretching was greater than 1, the signal was sped up. Otherwise when the *rate* of stretching was less than 1, then the signal was slowed down. The *rate* of time stretching was randomized for each audio with range values of [0.5, 1.5].

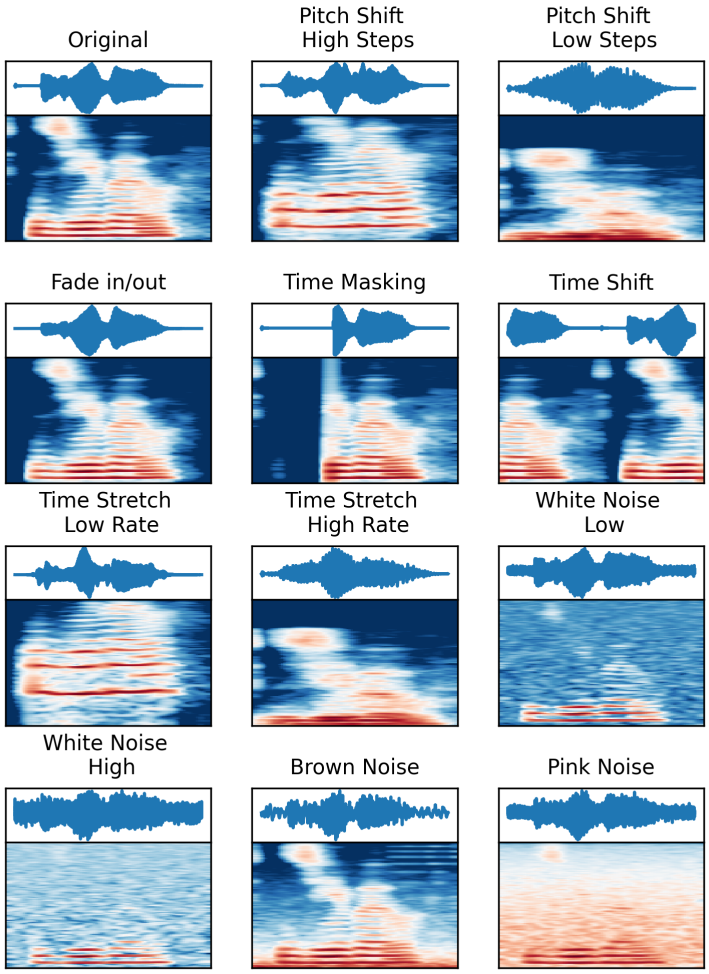

Figure 3.3: Raw audio with the subsequent mel-spectrogram for each audio augmentation utilized in the paper. Each augmentation have a varying degree of the shown transformation depending on random hyper parameters. Here an example of each augmentation is illustrated.

Chapter 4

Results

4.1 Overview

Although humans are proficient at perceiving and understanding sounds, making algorithms perform the same task poses a challenge due to the wide range of variations in auditory features. Applications of sound understanding range from surveillance [28] and music classification [29, 30] to audio generation [31, 32] and deep-fake detection [33].

Achieving automated auditory perception requires the learning of effective representations. As discussed in previous chapters, often prior work derive effective representations through discriminative approaches [19, 20, 21, 22]. In other words, similar to supervised learning, the model learns the mapping between the input signal to the class label. The underlying assumption with such approach is that the latent representations carry effective representations for the designed tasks. One fundamental problem with such learned representations is the potential limitation to generalizability. First, those representations are only limited to availability of expensive and time consuming labeled data. Secondly, representations are skewed towards one particular domain (e.g. speech, music, etc ...). Therefore, in both cases, major fine-tuning to the targeted training data would be required. Alternatively, recent self-supervised approaches using contrastive learning in the latent space have been shown to learn efficient representations that achieves state-of-the-art performance in images [12, 23, 24, 25, 26, 27] and videos [23]. However, it is still a major question on how we can achieve similar landmark on auditory data.

In this work, we build on SimCLR [12], a self-supervised framework for contrastive learning of visual representations. We show that similar framework could be adopted for learning effective auditory representations. Moreover, with a simple modification, we are able to reduce the training time and improve recognition performance.

4.2 Audio Data Augmentations for Contrastive Learning

Data augmentations have been widely adopted in audio [57, 58, 59, 60] and image [111, 112] domains. Moreover, recently it has been shown that some sequences of augmentations in image domain can offer a relatively better performance compared to other augmentations in contrastive learning [12]. In this section, we investigate the impact of different augmentations applied to the signal level on the quality of learned auditory representations.

To investigate the effect of individual auditory data augmentations and their sequential ordering, we perform comprehensive training of SimCLR framework on Speech Commands-10 dataset for 1000 epochs on all the proposed auditory augmentations (Section 3.5). Figure 4.1 shows top-1 test performance on both 1D and 2D variants of ResNet18. The diagonal line represents the performance of single augmentation, while other entries represent the performance of paired augmentations. Each row indicates the first augmentation and each column shows the second augmentation applied sequentially. The last column and row in each matrix represents the averaged predictive performance of a specific augmentation. The last column depicts the average when the augmentation was applied first, while the last row shows the average when the corresponding augmentation was applied second. The bottom right element represents the average of the whole matrix. Similar to [12], we found that multiple augmentations are required to learn efficient representations. In particular, with the 1D model, we observe that the composition of fade in/out, time stretching and pitch shifting significantly are the top-3 augmentations that improve the quality of representations. While on the 2D model, we observe that fade in/out, time masking and time shifting are premier in improving the quality of the auditory representations. On average the 1D variant of the model shows better performance (**1D**: 68.6 ± 0.82; **2D**: 67.0 ± 1.36), partially due time masking yielding the worse performance when applied as the first augmentation during the training of the 2D model. Moreover, the 2D variant of the model achieves the maximum recorded performance (89.3%).

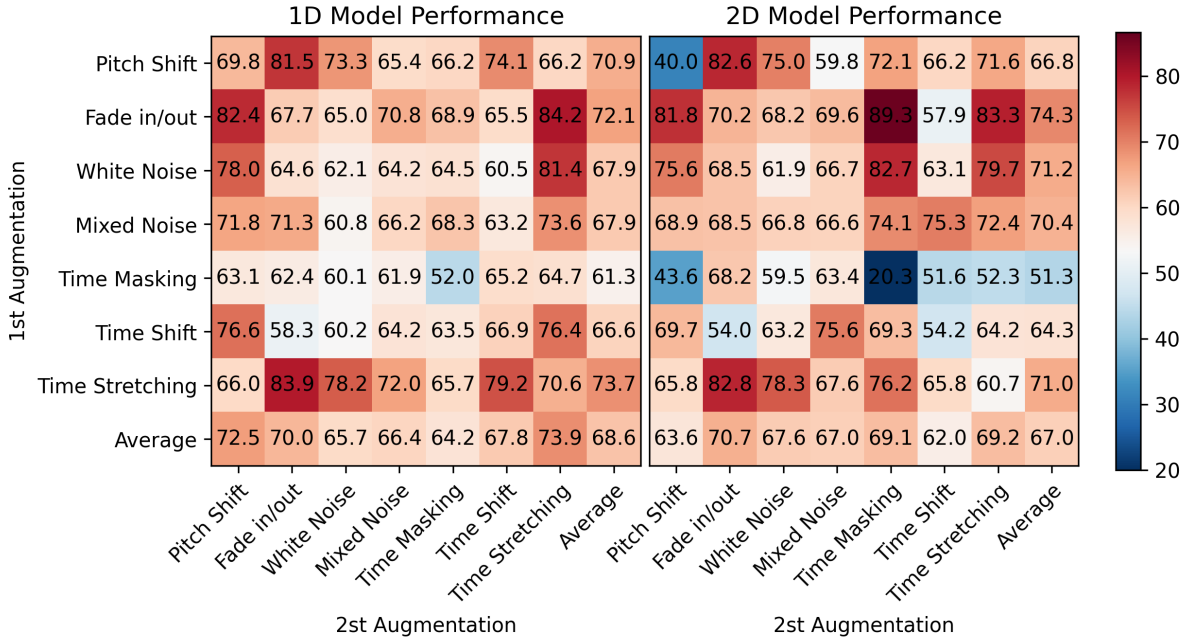

Figure 4.1: Top-1 test performance on Speech Commands-10 for 1D (left matrix) and 2D (right matrix) models trained for 1000 epochs. The diagonal line represents the performance of single augmentation, while other entries represent the performance of paired augmentations. Each row shows the first augmentation and column shows the second augmentation applied sequentially.

Table 4.1: Accuracy of ResNet18 trained on various datasets using different augmentations.

Models	Datasets					
1D	ECS-10	ECS-50	SC-10	SC-50	NSynth-11	NSynth-128
FD	12.5	3.3	67.7	16.1	25.1	52.2
FD + TST	52.1	2.9	84.2	24.2	34.6	70.3
FD + TST + PS	10.0	3.5	79.9	23.6	46.2	10.8
2D						
FD	46.2	19.5	70.2	13.7	31.1	52.8
FD + TM	**68.7**	33.7	**89.3**	**29.1**	**48.8**	**84.1**
FD + TM + TS	62.5	**40.4**	84.6	25.0	48.6	80.4

4.3 Raw Signal versus Time-Frequency Features

Time-frequency audio features have been used extensively in the literature, as algorithms trained on such features have consistently demonstrated better performance compared to algorithms trained on raw audio signals [21, 113, 22, 107]. In this section, we extend on this concept by investigating the efficiency of representations learned from raw signal and time-frequency features using contrastive learning.

Table 4.1 shows the accuracy of evaluation head attached to the frozen encoder trained with augmentations that yielded the best performance from section 4.2. We found that time-frequency features compared to raw audio signal consistently improve the learned representations. In particular, fade in/out and time masking using time-frequency features outperforms all other methods. We also found that increasing the number of augmentations does not necessary improve the learned representations. This was observed when predictive performance degraded after appending time-stretching augmentation to fade in/out and time masking.

4.4 CLAR versus Supervised & Self-Supervised

In this section, we utilize augmentations that yielded the best performance on SC-10 from section 4.3 to investigate the efficiency of CLAR when compared with supervised and self-supervised methods. We utilize SC-10 as the dataset of choice and investigate both the predictive performance of the evaluation head (Section 3.2 for more explanation) every 10 epochs for a maximum of 1000 epochs. This would not only shed light on the final performance but also the speed at which the methods reach such performance. Moreover, as CLAR is capable of semi-supervised training, we test its capability by training models on 100%, 20%, 10% and 1% labeled data. For the self-supervised simCLR method we use no labeled data, hence, the performance would be the same across the board.

Table 4.2: Accuracy of ResNet18 trained for 1000 epochs on Speech Command-10 dataset with incrementally less labels. During evaluation phase, we trained the evaluation head on all the labeled data with the frozen encoder.

Method	Type	Labeled Data Percentage			
		100%	20%	10%	1%
Cross-Entropy	Supervised	94.9	86.4	68.4	28.6
SupCon	Supervised	96.0	87.9	82.1	26.6
SimCLR	Self-supervised (Unsupervised)		84.8		
CLAR	Semi-Supervised	**96.1**	**90.5**	**89.1**	77.9

4.4.1 CLAR improves Learned Representations

Table 4.2 shows the top-1 accuracy of models trained using various methods while changing the percentage of labeled data. We found that the representations of the encoder trained with the CLAR method outperforms the representations learned using supervised and self-supervised methods when trained over the same number of epochs. When trained on 100% of the labeled data, CLAR achieves 96.1%, followed by SupCon [114] with 96.0% and Cross-Entropy with 94.9%. Furthermore, we show that this trend continues as we decrease the labeled data. That is, while the supervised methods loses 65% of the performance as we decrease the labeled data from 100% to 1%, CLAR decreases by only 19%. These results show that CLAR indeed combines both supervised and self-supervised methods to draw more efficient representations. Lastly, we found that when we decrease the amount of labeled data to only 1%, the performance of CLAR's learned representation degrades compared to the self-supervised method. This could be the result of overfitting more to the labeled data resulting in less efficient representations.

4.4.2 CLAR improves Speed of Convergence

Figure 4.2 shows the top-1 test performance of SC-10 dataset computed every 10-epochs by training an evaluation head attached to a frozen encoder over 1000 epochs. We found that the CLAR method not only improves the representations in the encoder but also improves the speed at which those representation are learned compared to the self-supervised approach. In particular, when we provide 100% of the labeled data, CLAR shows not only better predictive performance compared to supervised and self-supervised method but also show better training speed than self-supervised. As we decrease the amount of labeled data, this trend continues while the gap between the supervised method and CLAR test performance increases. These results suggest that both the self-supervised and supervised training tasks indeed share common representations that improves latent representations in the encoder. Moreover, the training algorithm optimize for the $\mathcal{L}_{CE}$ loss first followed by gradual slow optimization of the $\mathcal{L}_{CL}$

loss.

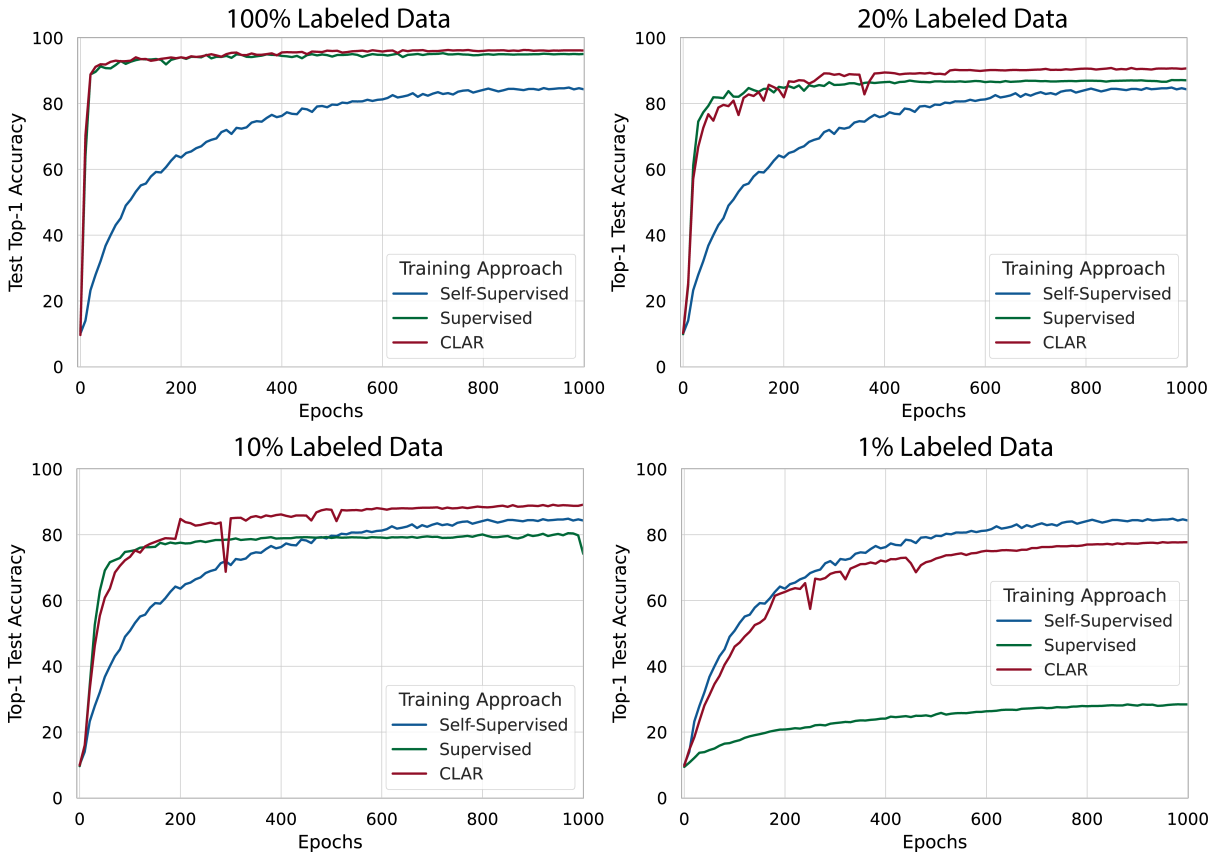

Figure 4.2: Top-1 test performance on Speech Commands-10 computed every 10-epochs by training an evaluation head attached to a frozen encoder over 1000 epochs. Each sub-figure represents the top-1 test performance on varying percentage of labeled data.

Chapter 5

Discussion & Conclusion

5.1 Conclusion

There has been growing interest in self-supervised learning to learn efficient representations that can be used for down-stream tasks. Recent methods has been successful in learning rich visual representations by leveraging the inherent structure of unlabeled images. However, it is still unclear whether we could use a similar self-supervised approach to learn superior auditory representations. In this thesis, we demonstrated the success of contrastive learning in earning efficient auditory representations. We performed extensive and comprehensive experiments on various design choices revealed the effectiveness of our proposed framework (CLAR) in terms of recognition performance as well as reduced training time in comparison with supervised and self-supervised methods. With our experiments, we (1) introduced and evaluated the impact of various auditory data augmentations on predictive performance, (2) showed that training with time-frequency audio features substantially improves the quality of the learned representations compared to raw signals, and (3) demonstrated that simultaneous training with both supervised and contrastive losses improves the learned representations compared to self-supervised and supervised training. Together, our results depicts a promising path towards automated audio understanding.

5.2 Limitations

The limitations of the thesis originates predominately from high demand of resources required to pre-trained these models:

1. **Batch & model size**: Both the batch-size and model-size are hyper-parameters that directly effect each other. This is due to hardware limitations in how much memory they have. As discussed in earlier chapters CLAR requires large batch-sizes due to its dependency on negative samples. In turn, in this thesis we were not able to experiment with bigger models. As we would not be able to compare the performance directly since the batch-size would not be controlled for. During the writing of this thesis, various recent frameworks were proposed to substantially alleviate such limitation, such as MoCov2 [99], SimSiam [55], BYOL [53], and Barlow Twins [115].

2. **Generalizability after fine-tuning**: While CLAR achieved top performance in-term of the quality of representations. It raises an interesting question: does achieving better performance with the linear classifier correlates with better generalizability to other datasets? That is, if we were to pre-train using self-supervised, and supervised methods on a speech dataset, would we observe similar margins in performance between these methods when fine-tuning on other dataset? Although this question might seem trivial, the answer is definitely not. As we can assume that the best achieving model on the speech dataset might've learned more than necessary for that one domain, hence can result in worse performance when fine-tuned on another dataset.

5.2.1 Applications

Learning abstract auditory representations using self-supervised learning has various downstream applications. As these representations can make the process of learning tasks for more specialized applications substantially easier. Applications of sound understanding can range from surveillance [28] and music classification [29, 30] to audio generation [31, 32] and deepfake detection [33]. Moreover, similar to how models for audio recognition benefit from pretraining on image dataset, we hypothesize that these audio models might be able to generalize to vast range of problems that are in the intersection of signal processing and deep learning such as medical tools [116, 117].

5.3 Future Research

In addition to resolving the limitations described earlier, the current work could substantially benefit from investigating the quality of representations from models that can encapsulate temporal dynamics such as Convolutional Recurrent Neural Network [29], or Transformers [118]. These dynamics might not improve the quality of auditory representations drastically but would considerably advance future research in more challenging data, such as videos. Videos are difficult to work with because they are both spatially and temporally rich, hence finding entangling representations would be greatly more challenging.

Bibliography

[1] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[2] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[3] C. Olah, A. Mordvintsev, and L. Schubert, "Feature visualization," *Distill*, 2017. https://distill.pub/2017/feature-visualization.

[4] A. Krizhevsky, V. Nair, and G. Hinton, "Cifar-10 (canadian institute for advanced research),"

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[6] A. Brock, J. Donahue, and K. Simonyan, "Large scale gan training for high fidelity natural image synthesis," *arXiv preprint arXiv:1809.11096*, 2018.

[7] A. Dosovitskiy, P. Fischer, J. T. Springenberg, M. Riedmiller, and T. Brox, "Discriminative unsupervised feature learning with exemplar convolutional neural networks," 2015.

[8] S. Gidaris, P. Singh, and N. Komodakis, "Unsupervised representation learning by predicting image rotations," 2018.

[9] C. Doersch, A. Gupta, and A. A. Efros, "Unsupervised visual representation learning by context prediction," 2016.

[10] Z. Wu, Y. Xiong, S. X. Yu, and D. Lin, "Unsupervised feature learning via non-parametric instance discrimination," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3733–3742, 2018.

[11] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," 2020.

[12] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," 2020.

[13] L. Zaadnoordijk, T. R. Besold, and R. Cusack, "The next big thing(s) in unsupervised machine learning: Five lessons from infant learning," 2020.

[14] T. Serre, A. Oliva, and T. Poggio, "A feedforward architecture accounts for rapid categorization," *Proceedings of the national academy of sciences*, vol. 104, no. 15, pp. 6424–6429, 2007.

[15] J. J. DiCarlo, D. Zoccolan, and N. C. Rust, "How does the brain solve visual object recognition?," *Neuron*, vol. 73, no. 3, pp. 415–434, 2012.

[16] D. L. Yamins, H. Hong, C. F. Cadieu, E. A. Solomon, D. Seibert, and J. J. DiCarlo, "Performance-optimized hierarchical models predict neural responses in higher visual cortex," *Proceedings of the national academy of sciences*, vol. 111, no. 23, pp. 8619–8624, 2014.

[17] R. Epstein and N. Kanwisher, "A cortical representation the local visual environment," *Nature*, vol. 392, pp. 598–601, apr 1998.

[18] R. Epstein, A. Harris, D. Stanley, and N. Kanwisher, "The parahippocampal place area: Recognition, navigation, or encoding?," *Neuron*, vol. 23, no. 1, pp. 115–125, 1999.

[19] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "Specaugment: A simple data augmentation method for automatic speech recognition," *arXiv preprint arXiv:1904.08779*, 2019.

[20] S. Hershey, S. Chaudhuri, D. P. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, *et al.*, "Cnn architectures for large-scale audio classification," in *2017 ieee international conference on acoustics, speech and signal processing (icassp)*, pp. 131–135, IEEE, 2017.

[21] Y. Tokozume and T. Harada, "Learning environmental sounds with end-to-end convolutional neural network," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2721–2725, IEEE, 2017.

[22] A. Guzhov, F. Raue, J. Hees, and A. Dengel, "Esresnet: Environmental sound classification based on visual domain models," *ArXiv*, vol. abs/2004.07301, 2020.

[23] R. Qian, T. Meng, B. Gong, M.-H. Yang, H. Wang, S. Belongie, and Y. Cui, "Spatiotemporal contrastive video representation learning," 2020.

[24] P. Bachman, R. D. Hjelm, and W. Buchwalter, "Learning representations by maximizing mutual information across views," in *Advances in Neural Information Processing Systems*, pp. 15535–15545, 2019.

[25] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.

[26] A. Dosovitskiy, J. T. Springenberg, M. Riedmiller, and T. Brox, "Discriminative unsupervised feature learning with convolutional neural networks," in *Advances in neural information processing systems*, pp. 766–774, 2014.

[27] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2, pp. 1735–1742, IEEE, 2006.

[28] R. Radhakrishnan, A. Divakaran, and A. Smaragdis, "Audio analysis for surveillance applications," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, 2005.*, pp. 158–161, IEEE, 2005.

[29] K. Choi, G. Fazekas, M. Sandler, and K. Cho, "Convolutional recurrent neural networks for music classification," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2392–2396, IEEE, 2017.

[30] K. M. Ibrahim, J. Royo-Letelier, E. V. Epure, G. Peeters, and G. Richard, "Audio-based auto-tagging with contextual tags for music," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 16–20, 2020.

[31] J. Engel, K. K. Agrawal, S. Chen, I. Gulrajani, C. Donahue, and A. Roberts, "GANSynth: Adversarial neural audio synthesis," in *International Conference on Learning Representations*, 2019.

[32] C. Donahue, J. McAuley, and M. Puckette, "Adversarial audio synthesis," 2019.

[33] T. Mittal, U. Bhattacharya, R. Chandra, A. Bera, and D. Manocha, "Emotions don't lie: A deepfake detection method using audio-visual affective cues," 2020.

[34] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[35] D. Erhan, Y. Bengio, A. Courville, and P. Vincent, "Visualizing higher-layer features of a deep network," *University of Montreal*, vol. 1341, no. 3, p. 1, 2009.

[36] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," *arXiv preprint arXiv:1312.6034*, 2013.

[37] A. Mahendran and A. Vedaldi, "Understanding deep image representations by inverting them," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5188–5196, 2015.

[38] A. Nguyen, A. Dosovitskiy, J. Yosinski, T. Brox, and J. Clune, "Synthesizing the preferred inputs for neurons in neural networks via deep generator networks," *Advances in neural information processing systems*, vol. 29, pp. 3387–3395, 2016.

[39] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[40] A. Krizhevsky, "One weird trick for parallelizing convolutional neural networks," 2014.

[41] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," 2014.

[42] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," 2018.

[43] Y. Bengio, Y. LeCun, *et al.*, "Scaling learning algorithms towards ai," *Large-scale kernel machines*, vol. 34, no. 5, pp. 1–41, 2007.

[44] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[45] S. Zagoruyko and N. Komodakis, "Wide residual networks," *arXiv preprint arXiv:1605.07146*, 2016.

[46] J.-T. Huang, J. Li, D. Yu, L. Deng, and Y. Gong, "Cross-language knowledge transfer using multilingual deep neural network with shared hidden layers," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 7304–7308, IEEE, 2013.

[47] H. Chang, J. Han, C. Zhong, A. M. Snijders, and J. H. Mao, "Unsupervised transfer learning via multi-scale convolutional sparse coding for biomedical applications," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 5, pp. 1182–1194, 2018.

[48] D. George, H. Shen, and E. Huerta, "Deep transfer learning: A new deep learning glitch classification method for advanced ligo," *arXiv preprint arXiv:1706.07446*, 2017.

[49] G. Gwardys and D. M. Grzywczak, "Deep image features in music information retrieval," *International Journal of Electronics and Telecommunications*, vol. 60, no. 4, pp. 321–326, 2014.

[50] S. Becker and G. E. Hinton, "Self-organizing neural network that discovers surfaces in random-dot stereograms," *Nature*, vol. 355, no. 6356, pp. 161–163, 1992.

[51] L. Wiskott and T. J. Sejnowski, "Slow feature analysis: Unsupervised learning of invariances," *Neural computation*, vol. 14, no. 4, pp. 715–770, 2002.

[52] H. Mobahi, R. Collobert, and J. Weston, "Deep learning from temporal coherence in video," in *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 737–744, 2009.

[53] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. A. Pires, Z. D. Guo, M. G. Azar, B. Piot, K. Kavukcuoglu, R. Munos, and M. Valko, "Bootstrap your own latent: A new approach to self-supervised learning," 2020.

[54] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin, "Unsupervised learning of visual features by contrasting cluster assignments," 2021.

[55] X. Chen and K. He, "Exploring simple siamese representation learning," 2020.

[56] R. Zhang, P. Isola, and A. A. Efros, "Colorful image colorization," in *European conference on computer vision*, pp. 649–666, Springer, 2016.

[57] B. McFee, E. J. Humphrey, and J. P. Bello, "A software framework for musical data augmentation.," in *ISMIR*, vol. 2015, pp. 248–254, 2015.

[58] J. Schlüter and T. Grill, "Exploring data augmentation for improved singing voice detection with neural networks.," in *ISMIR*, pp. 121–126, 2015.

[59] T. Ko, V. Peddinti, D. Povey, and S. Khudanpur, "Audio augmentation for speech recognition," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[60] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "Specaugment: A simple data augmentation method for automatic speech recognition," *Interspeech 2019*, Sep 2019.

[61] P. Goyal, P. Dollár, R. B. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He, "Accurate, large minibatch SGD: training imagenet in 1 hour," *CoRR*, vol. abs/1706.02677, 2017.

[62] J. Wiens and E. S. Shenoy, "Machine learning for healthcare: on the verge of a major shift in healthcare epidemiology," *Clinical Infectious Diseases*, vol. 66, no. 1, pp. 149–153, 2018.

[63] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[64] C. Doersch and A. Zisserman, "Multi-task self-supervised visual learning," *CoRR*, vol. abs/1708.07860, 2017.

[65] O. J. Hénaff, A. Srinivas, J. D. Fauw, A. Razavi, C. Doersch, S. M. A. Eslami, and A. van den Oord, "Data-efficient image recognition with contrastive predictive coding," 2019.

[66] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2020.

[67] R. Kiros, Y. Zhu, R. Salakhutdinov, R. S. Zemel, A. Torralba, R. Urtasun, and S. Fidler, "Skip-thought vectors," 2015.

[68] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," in *Psychology of learning and motivation*, vol. 24, pp. 109–165, Elsevier, 1989.

[69] I. J. Goodfellow, M. Mirza, D. Xiao, A. Courville, and Y. Bengio, "An empirical investigation of catastrophic forgetting in gradient-based neural networks," *arXiv preprint arXiv:1312.6211*, 2013.

[70] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 12, pp. 2935–2947, 2017.

[71] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun, "What is the best multi-stage architecture for object recognition?," in *2009 IEEE 12th International Conference on Computer Vision*, pp. 2146–2153, 2009.

[72] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. icml*, vol. 30, p. 3, Citeseer, 2013.

[73] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), vol. 25, Curran Associates, Inc., 2012.

[74] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," 2014.

[75] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.

[76] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," 2016.

[77] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," 2017.

[78] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," 2018.

[79] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, p. 448–456, JMLR.org, 2015.

[80] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[81] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, (New York, NY, USA), p. 1096–1103, Association for Computing Machinery, 2008.

[82] A. Makhzani and B. Frey, "k-sparse autoencoders," 2014.

[83] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio, "Contractive auto-encoders: Explicit invariance during feature extraction," in *Icml*, 2011.

[84] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," 2014.

[85] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "beta-vae: Learning basic visual concepts with a constrained variational framework," 2016.

[86] A. v. d. Oord, O. Vinyals, and K. Kavukcuoglu, "Neural discrete representation learning," *arXiv preprint arXiv:1711.00937*, 2017.

[87] A. Razavi, A. van den Oord, and O. Vinyals, "Generating diverse high-fidelity images with vq-vae-2," 2019.

[88] K. Gregor, G. Papamakarios, F. Besse, L. Buesing, and T. Weber, "Temporal difference variational auto-encoder," 2019.

[89] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4401–4410, 2019.

[90] J. Donahue, P. Krähenbühl, and T. Darrell, "Adversarial feature learning," *arXiv preprint arXiv:1605.09782*, 2016.

[91] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, "Analyzing and improving the image quality of stylegan," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8110–8119, 2020.

[92] M. Bińkowski, J. Donahue, S. Dieleman, A. Clark, E. Elsen, N. Casagrande, L. C. Cobo, and K. Simonyan, "High fidelity speech synthesis with adversarial networks," *arXiv preprint arXiv:1909.11646*, 2019.

[93] C. Donahue, J. McAuley, and M. Puckette, "Adversarial audio synthesis," *arXiv preprint arXiv:1802.04208*, 2018.

[94] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems 27* (Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, eds.), pp. 2672–2680, Curran Associates, Inc., 2014.

[95] M. Noroozi and P. Favaro, "Unsupervised learning of visual representations by solving jigsaw puzzles," 2017.

[96] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015.

[97] A. van den Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," 2019.

[98] T. DeVries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," 2017.

[99] X. Chen, H. Fan, R. Girshick, and K. He, "Improved baselines with momentum contrastive learning," 2020.

[100] J. Allen, "Short term spectral analysis, synthesis, and modification by discrete fourier transform," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 25, no. 3, pp. 235–238, 1977.

[101] K. W. Cheuk, H. Anderson, K. Agres, and D. Herremans, "nnaudio: An on-the-fly gpu audio to spectrogram conversion toolbox using 1d convolutional neural networks," *IEEE Access*, 2020.

[102] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[103] A. Kolesnikov, X. Zhai, and L. Beyer, "Revisiting self-supervised visual representation learning," 2019.

[104] Y. You, I. Gitman, and B. Ginsburg, "Large batch training of convolutional networks," *arXiv preprint arXiv:1708.03888*, 2017.

[105] I. Loshchilov and F. Hutter, "Sgdr: Stochastic gradient descent with warm restarts," *arXiv preprint arXiv:1608.03983*, 2016.

[106] P. Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," 2018.

[107] J. Engel, C. Resnick, A. Roberts, S. Dieleman, D. Eck, K. Simonyan, and M. Norouzi, "Neural audio synthesis of musical notes with wavenet autoencoders," 2017.

[108] K. J. Piczak, "ESC: Dataset for Environmental Sound Classification," 2015.

[109] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He, "Accurate, large minibatch sgd: Training imagenet in 1 hour," 2018.

[110] B. McFee, V. Lostanlen, A. Metsai, M. McVicar, S. Balke, C. Thomé, C. Raffel, F. Zalkow, A. Malek, Dana, K. Lee, O. Nieto, J. Mason, D. Ellis, E. Battenberg, S. Seyfarth, R. Yamamoto, K. Choi, viktorandreevichmorozov, J. Moore, R. Bittner, S. Hidaka, Z. Wei, nullmightybofo, D. Hereñú, F.-R. Stöter, P. Friesch, A. Weiss, M. Vollrath, and T. Kim, "librosa/librosa: 0.8.0," July 2020.

[111] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[112] O. J. Hénaff, A. Srinivas, J. De Fauw, A. Razavi, C. Doersch, S. Eslami, and A. v. d. Oord, "Data-efficient image recognition with contrastive predictive coding," *arXiv preprint arXiv:1905.09272*, 2019.

[113] Y. Tokozume, Y. Ushiku, and T. Harada, "Learning from between-class examples for deep sound recognition," *arXiv preprint arXiv:1711.10282*, 2017.

[114] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan, "Supervised contrastive learning," 2020.

[115] J. Zbontar, L. Jing, I. Misra, Y. LeCun, and S. Deny, "Barlow twins: Self-supervised learning via redundancy reduction," 2021.

[116] L. Brunese, F. Martinelli, F. Mercaldo, and A. Santone, "Deep learning for heart disease detection through cardiac sounds," *Procedia Computer Science*, vol. 176, pp. 2202–2211, 2020. Knowledge-Based and Intelligent Information Engineering Systems: Proceedings of the 24th International Conference KES2020.

[117] A. Raza, A. Mehmood, S. Ullah, M. Ahmad, G. S. Choi, and B.-W. On, "Heartbeat sound signal classification using deep learning," *Sensors*, vol. 19, no. 21, p. 4819, 2019.

[118] S.-w. Yang, A. T. Liu, and H.-y. Lee, "Understanding self-attention of self-supervised audio transformers," *arXiv preprint arXiv:2006.03265*, 2020.

Curriculum Vitae

Name:	Haider Al-Tahan
Education:	Honour B.Sc. in Computer Science and Psychology 2014-2019 York University Toronto, Ontario
Honours and Awards:	NSERC USRA 2019 Ontario Graduate Scholarship 2020 Graduate Student Innovation Scholars 2021
Related Work Experience:	Teaching Assistant The University of Western Ontario 2019 - Present

Publications:

1. Al-Tahan, H., Mohsenzadeh, Y. (2021), Reconstructing feedback representations in the ventral visual pathway with a generative adversarial autoencoder. *PLOS Computational Biology* 17(3): e1008775. https://doi.org/10.1371/journal.pcbi.1008775

2. Al-Tahan, H., Mohsenzadeh, Y. (2021), CLAR: Contrastive Learning of Auditory Representations. *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics, PMLR 130:2530-2538*. http://proceedings.mlr.press/v130/al-tahan21a/al-tahan21a.pdf