WORCESTER POLYTECHNIC INSTITUTE

MASTER'S THESIS

Multi-modal Deep Learning

Author:
Si LIU

Advisor:
Dr. Randy PAFFENROTH

Reader:
Dr. Xiaozhong LIU

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

May 2022

Advisor	Date
Reader	Date

Abstract

Si LIU

Multi-modal Deep Learning

In many problems in machine learning, the same underlying phenomena can be observed using different types of data. We call such problems "multi-modal" since they allow multiple different views of the same problem. However, many current deep learning techniques are not designed for such multi-modal data, and here we study how to utilize more than one dataset to improve the performance of deep learning models. In particular, we demonstrate how deep neural networks performance can be improved using our proposed multi-modal learning compared with single-modal learning. Additionally, we explore effective and efficient ways to combine different data modalities. This research demonstrates our techniques on many multi-modal problems, including autoencoders trained with the MNIST dataset and deep neural network classifiers trained with real-world chemistry data. We propose the no-harm factor, a method to ensure that adding another modality doesn't harm the performance of models in the presence of a small amount of multi-modal data. The no-harm factor is easy to apply and is practical, especially in chemical analysis tasks where limited labeled data are available.

Acknowledgements

Thank you to my advisor Dr. Randy Paffenroth and reader Dr. Xiaozhong Liu.

Contents

List of Figures

List of Tables

Chapter 1

Executive Summary

Multi-modal deep learning is a field of building deep neural network models that can process information from multiple sources. Each source relates to one modality or mode. In our research, we use deep neural network models to process information represented by two modalities (Figure 1.1). First, we synthesize two modalities using MNIST data to better understand the algorithms of multi-modal learning. Second, we use two real-world datasets to conduct chemical analysis using multi-modal techniques.

Chemical analysis is important in many aspects of human lives, from environmental impact to drug discovery. However, it is very challenging, because there is a wide range of machine learning tasks but a small amount of data that are highly heterogeneous and expensive to gather and label. We use these multi-modal experiments to improve chemical machine learning tasks that are usually single-modal by utilizing the multi-modal characteristic of chemical data. We also explore deep neural networks' designs and training methods that can effectively and efficiently combine multiple modalities.

The main contribution of this paper is the no-harm factor, a method to make sure that multi-modal learning should not be worse than single-modal learning with either of the modalities. Multi-modal neural networks gain complexity when new layers are needed for an additional modality. Preventing performance drop is especially practical when there are insufficient data points because the complexity gained from multi-modal architecture might aggravate the overfitting problem.

A typical multi-modal neural network architecture is to combine the embeddings of the modalities later in the network. The no-harm factor λ is a learnable parameter that controls the importance of one embedding c_1 over another embedding c_2. As shown in Figure 1.2, the embeddings are the second last layers of Neural Network 1 and 2 (their last layers are predicting layers, which are not shown in this figure). When λ equals 0 or 1, this whole model is the same as either of the neural networks. The benefits of the no-harm factor are:

- It is easy to apply because we do not need to explicitly explore how and where to combine the embeddings.
- It learns the importance of embeddings without introducing too many learnable parameters, thus limiting the complexity gain.
- It simplifies the design of the multi-modal neural networks.

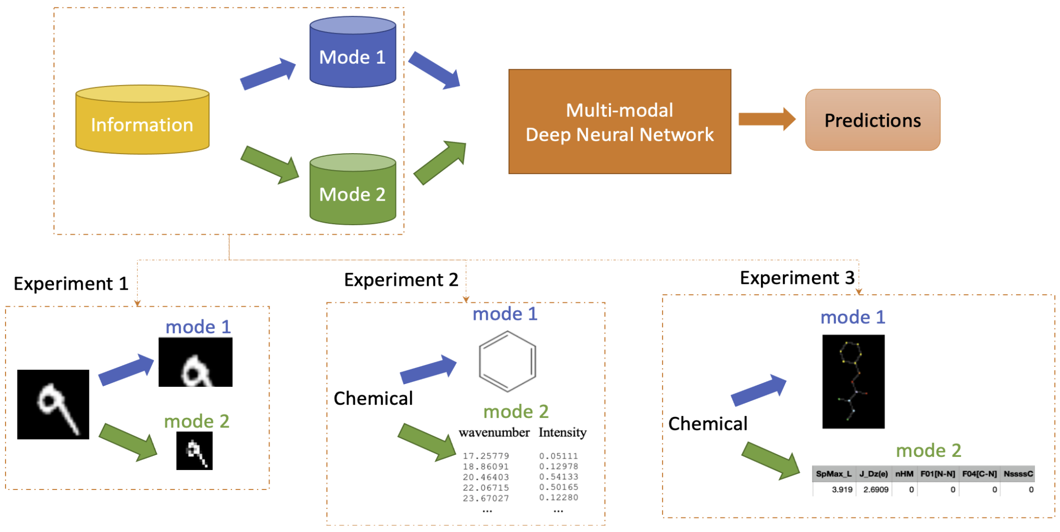


FIGURE 1.1: Overview of this research. The goal of the multi-modal deep learning is to create deep neural networks that can process information using two modalities of the information. In our research, we had three major experiments, and each experiment had different information represented by two modalities (mode 1 and mode 2). In Experiment 1, using the MNIST dataset, we synthesized two modalities: partial images (mode 1) and smaller images (mode 2). We used an encoder-decoder neural network to reconstruct original information/modality using synthesized modality. In Experiment 2, the dataset comprises chemical data represented by their molecular images (mode 1) and Raman spectrum data (mode 2). Raman spectra are considered the fingerprints of chemicals [21]. As far as we know, chemical Raman spectroscopy, an important real-world measurement technique, has never been used in multi-modal deep learning in chemical analysis. The task in Experiment 2 was to predict the number of double bonds, i.e., the number of two parallel lines in molecular images. In the third experiment, the first mode consists of chemical data represented by their molecular images (mode 1) with chemical characteristics [17], and the second mode consists of 41 descriptors. The chemical characteristics and descriptors were both selected by domain experts. The prediction task in Experiment 3 was to predict if a chemical was biodegradable in the natural environment.

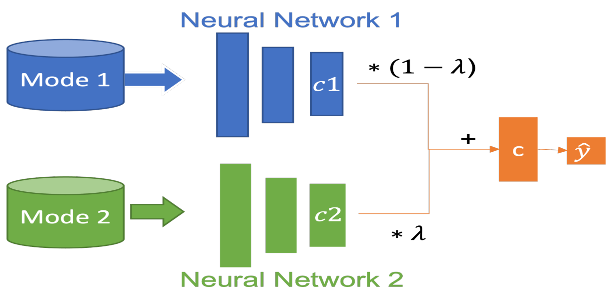
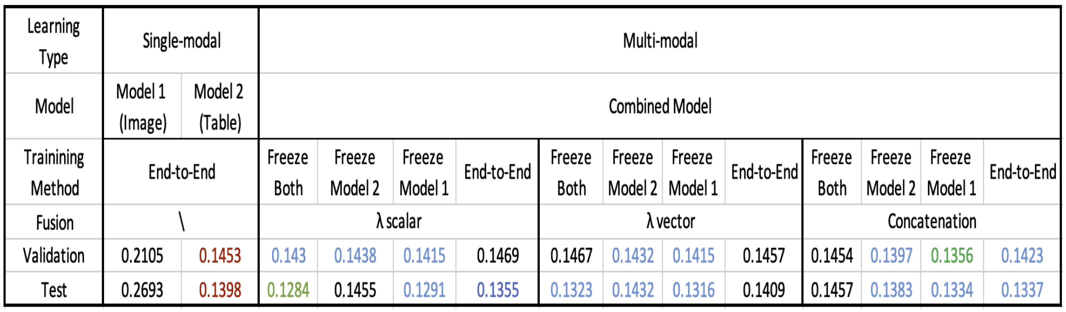


FIGURE 1.2: Implementation of no-harm factor λ in multi-modal architecture. λ controls the importance of one embedding c_1 over another embedding c_2. When λ is a scalar, λ is shared by all neurons in embedding c_1. When λ is a vector, each neuron in embedding c_1 has a specific λ_i value. c_2 gets $1-\lambda$ in the same way.

Table 1.1 shows the results of multi-modal learning with the no-harm factor compared to single-modal learning. Model 1 and Model 2 are the two single-learning models trained with single modalities, and Model 2 is the baseline because it has better performance than Model 1. Red values are the test and validation metrics of the baseline single-modal model. Blue values are lower than red values, which means these models outperform the baseline model when trained via specific training methods and the no-harm factor. Green values are the best results in test or validation time. This table shows that if the model is trained properly, multi-model learning with the no-harm factor can improve the performance of single-modal learning. With the no-harm scalar, the best test result is 8.2% better than that of the baseline when both of the pretrained models were frozen. Even though the no-harm factor can't guarantee better performance all the time, with very few learnable parameter added to the model, no-harm factor provides a good estimation of how much an additional modality can improve model performance when its embedding is combined with current embedding in linear way. Training methods in this table will be explained in Section 3.



Learning Type	Single-modal		Multi-modal											
Model	Model 1 (Image)	Model 2 (Table)	Combined Model											
Trainining Method	End-to-End		Freeze Both	Freeze Model 2	Freeze Model 1	End-to-End	Freeze Both	Freeze Model 2	Freeze Model 1	End-to-End	Freeze Both	Freeze Model 2	Freeze Model 1	End-to-End
Fusion	\		λ scalar				λ vector				Concatenation			
Validation	0.2105	0.1453	0.143	0.1438	0.1415	0.1469	0.1467	0.1432	0.1415	0.1457	0.1454	0.1397	0.1356	0.1423
Test	0.2693	0.1398	0.1284	0.1455	0.1291	0.1355	0.1323	0.1432	0.1316	0.1409	0.1457	0.1383	0.1334	0.1337

TABLE 1.1: Results of multi-modal learning with no-harm factor. Green values are the test and validation metrics of the baseline model. Blue values are lower than green values, which means these models outperform the baseline single-modal model when trained via specific training methods and the no-harm factor.

Chapter 2

Introduction

2.1 Overview

This world is multi-modal. It combines what we see, smell, touch, hear, etc. Our senses create different representations of the world. Multi-modal deep learning is a field of building deep neural network models that can process information from multiple sources. Each source relates to one modality or mode. We can have image, audio, and text datasets to represent the same data entities. Each representation is a modality or mode. On the contrary, single-modal data has one modality. In this era of big data, data are usually multi-modal because it is the natural form for information flow in the real world. They are usually collected from different resources. For example, the exponential growth of multi-modal content on the Internet leads to a massive volume of unstructured data such as text, audio, image, and video contributed by Web users through social media, blogs, online communities [53]; the advances in the medical device industry have allowed the physicians to collect a vast amount of multi-modal data such as x-ray imaging, magnetic resonance imaging (MRI), and recordings of the electrical activity of organs, etc. [22]. Big data technology has increased the availability and accessibility of multi-modal data, and multi-modal learning is vital in big data research to embrace the variety of information. Big data contain abundant information that is beneficial but challenging to traditional multi-modal machine learning tasks ranging from the Internet of Things, vehicular networks to social networks [14]. Thus, multi-modal deep learning is a timely topic.

Multi-modal deep learning aims to learn deep neural networks that process and relate information from multiple modalities [3]. Due to the superior performance in multiple domains, there has been an explosion of interest in multi-modal deep learning tasks such as analyzing audio-visual scenes [45], matching images and sentences [39], combining medical imaging and electronic health records [29], interpreting user activity and context captured using multi-sensor data [48], and recognizing objects using RGB and depth image data pair[13]. This research uses two modalities to train multi-modal neural networks that can process information from image-image or image-table modalities. Our main task is comparing single-modal and multi-modal learning; thus, two modalities are enough for us to understand multi-modal algorithms and explore how to train these algorithms efficiently and effectively. The topics specific to a higher number of modalities are out of the scope of this research.

In multi-modal tasks, limited resources for data collection affect the quality and availability of modalities. We usually spend fewer resources collecting lower-quality data than higher-quality data. For example, it is cheaper to take blurry or partial images than high-resolution images, or to record audio with huge background noise on the street than clear audios in professional studios. Different modes require different resources to be collected. When resources are limited, it is important to determine which modes are most valuable and should be concentrated on to achieve the best performance. For example, when designing multiple sensors for an autonomous vehicle, multiple sensors are needed to fully perceive the surroundings. Researchers have been looking for low-cost combinations in terms of computation, amount of resource needed, or price. In this paper [35], researchers combined RGB image and LiDAR laser sensor [11] data for a low-cost navigation system. In this research [8], lower resolution and cheaper sensors were used to detect road conditions. It is practical to explore cost-effective ways to combine multiple modalities, especially when only lower quality data is available because of limited resources. One of our experiments is to understand why and how two low-quality modalities combined can achieve comparable or better performance than a higher-quality modality alone.

Another common problem, especially with machine learning in the chemical field, is that only a limited amount of labeled data is available despite a tremendous number of unlabeled data. This problem leads to challenges such as overfitting and unstable model performance. Some researchers in the chemistry domain have utilized more than one dataset to compensate for the lack of labeled data to some extent. For example, researchers have combined molecular images and chemical property data to improve the prediction of biodegradation using 1055 data samples [18]. However, multi-modal learning is still not extensively studied in chemical analysis like in other domains such as computer vision and speech recognition. Chemical analysis is important in many aspects of human lives, from environmental impact to drug discovery. However, it is very challenging, one of the main reasons is that there are a wide range of machine learning tasks but a small amount of data that is highly heterogeneous and expensive to gather and label. Unlike collecting data such as online images or videos, obtaining precise and accurate chemical data is expensive because it typically requires special device and expert supervision [67]. We have conducted two chemical analysis experiments, both of which were to predict a chemical property using imagery and tabular modalities of limited labeled samples. We use these experiments to utilize the multi-modal characteristics of chemical data to improve chemical machine learning tasks that are usually single-modal.

Motivated to explore these two real-world applications of multi-modal learning, we have carried out this research to explore practical multi-modal deep learning methods. First, we synthesize low-cost modalities using MNIST data [70] to better understand the cost-effectiveness and algorithms of multi-modal learning. Second, we use two real-world multi-modal datasets to improve the performance of single-modal learning with a small amount of labeled data. We explore several aspects of effective and efficient multi-modal deep learning, such as training methods, fusion techniques, and representation learning.

2.2 Multi-modal Deep Learning

Multi-modal deep learning is a large field with active literature.The following references provide overviews of multi-modal deep learning and their challenges including representation, translation, alignment, fusion and co-learning [3], [50], [59]. There are dedicated reviews on some of these challenges, such as how to fuse learned multi-modal representations from various deep learning architectures [14]; key issues of deep learning techniques in representation learning that narrow heterogeneity gap among different modalities [22]; challenges of co-learning, which addresses the problem such as one or more modalities are missing or noisy, lacking labeled data, having unreliable labels, or are scarce in training or testing or both [49]. Comprehensive reviews on multi-modal alignment are missing but this topic can be found in specific areas such as matching features of multiple modalities in computer vision [4], aligning common pixels in images [33], and aligning music information [54]. The domain specified surveys on multi-modal translation cover topics such as audio-visual speech synthesis that maps data between audio modality and visual modality [42], cross-modal retrieval that takes one type of data as the query to retrieve relevant data of another type [65]. Despite its outstanding performance, multi-modal deep learning's social acceptance and usability are limited by the complex, opaque and black-box nature, leading to commentary on explainability in multi-modal deep neural networks [34].

Of particular interest for us in multi-modal deep learning are representation learning and fusion – two multi-modal deep learning topics that have been discussed together often. For example, in this survey, researchers discussed learning multi-modal representation, fusing multi-modal signals at various levels, and their applications [74]. In one of the first multi-modal deep learning applications, learning features over multiple modalities and combining them via different techniques were both key concepts [44]. Last but not least, representation learning and fusion were also important in decomposing single-modal representation into multi-modal dynamics using recurrent neural fusion architecture [40]. Facing unique challenges in multi-modality tasks, researchers have proposed specific techniques such as how to train deep learning models to focus on information from more reliable modalities while reducing the emphasis on the less reliable modalities [38], to learn the joint distribution of modalities in different modality domains [30], and to balance the trade-off between inter-modal fusion and intra-modal processing [64].

In the domain of deep learning for chemistry problems, there is a significant body of work where the researchers have taken advantage of the diverse representations of chemicals so that a collective performance could exceed the individual performance of each neural network trained with one representation. For example, domain experts have used text, images, and numerical features to predict toxicity levels of chemical compounds [36]. In drug repurposing, researchers utilized chemical structures and deferential gene expression data to discover medicines remarkably similar to the proposed medications [28]. In drug-drug interaction events, researchers have combined various drug features, including chemical substructures, targets, enzymes and pathways, to learn cross-modality representations of drug-drug pairs and predict their interactions [12].

In Section 2.5.1, we will talk about autoencoders that we use in our first experiment and explain the importance of embeddings heavily used in our research. In Section 2.5.3 and 2.5.2, we will talk about 2 classic neural network types that handle multiple modalities – image and tabular data – in our datasets. In section 2.5.4, we will talk about multi-modal representation learning. The final representation is where multiple representations are usually summed up, averaged, or concatenated. The combination, or fusion, methods will be discussed in Section 2.5.5.

In Section 2.5 of this chapter, we will talk more about the background of related topics in terms of deep learning and multi-modal learning. Then in Chapter 3, we will talk about each experiment in detail, explaining how we did it and showing the results. On top of the standard training methods for traditional single-modal deep neural networks, there are unique options to think about when training multi-modal deep learning models, and we will mention the training options we use in Section 3.3.1. We did these experiments sequentially, and the results of each one led to the next experiment. In Chapter 4, we will show the results of these experiments. In Chapter 5, we will conclude the research and discuss the results. Finally, in Appendix A, we will give more details on all the architectures of our deep learning models.

2.3 Justification for the Experiments

We do not always have the ideal modality of observed entities in real-life machine learning problems, but there are other accessible modalities for the same task. For example, sometimes a company can not afford high-cost sensors that render high-resolution images. However, the budget can still cover more and cheaper sensors that generate low-resolution or partial images. We want to know if the multi-modal substitute is as useful as the ideal modality despite the incomplete information in the substitute. One way to examine the potential of the substitute is to use an autoencoder [20] to reconstruct the original modality with full information using the substitute with partial information. If the encoder learns enough useful information in the hidden layer, the decoder doesn't need all the information from a original image to reconstruct a original image. More details about encoding and hidden layer will be discussed in Section 2.5.1. In our first experiment, the baseline model used the original modality to reconstruct the original modality. The only difference among the autoencoders was the input layer size dependent on the size of combined modalities. We then compared the losses of baseline autoencoder and multi-modal autoencoders. If multi-modal models achieved comparable performance to the baseline, we could combine less ideal modalities to substitute for the ideal modality.

Based on the results of the first experiment, we got the intuition that multi-modal learning could improve the performance of single-modal learning, especially when the performance of single-modal learning is really bad. However, in real-world problems, we do not always have as many data as in the MNIST dataset, so it's practical to check if multi-modal learning can handle the lack of data. We used experiment 2 to examine whether multi-modal learning could improve single-modal learning when either modality had sufficient real-world data entries. In the second experiment we combined Raman spectra with chemical molecular image to predict a chemical property: double bonds. As far as we are aware, chemical Raman spectrum from Raman spectroscopy, an important real-world measurement technique, has never been used in multi-modal deep learning in chemical analysis. If adding Raman modality to molecular modality could improve model performance, we would consider collecting more Raman data and using this combination to work on other chemical tasks.

In our last experiments, to test our models on a public dataset, we found a research paper that proposed the first effective multi-modal CNN-MLP neural network for chemistry property prediction [18]. We also used image-tabular modalities and CNN-MLP architectures in the second experiment, making our models comparable given the same dataset. While both of our experiments aimed to address the lack of labeled data in chemistry, our work explored more multi-modal techniques so that we may improve their models.

2.4 Contribution

First, we applied multi-modal learning to a unique combination of Raman spectrograph and molecular images. As far as we are aware, chemical Raman spectrograph, an important real-world measurement technique, has never been used in multi-modal deep learning in chemical analysis.

Second, we proposed the no-harm factor, a method to make sure that multi-modal learning is at least as good as single-modal learning with either of the modalities. Preventing performance drop is especially practical when there are insufficient data because the complexity gained from multi-modal architecture might aggravate overfitting problem. A typical multi-modal neural network architecture is to combine the embeddings of the modalities later in the network. A no-harm factor λ is a learnable parameter that controls the importance of one embedding c_1 over another embedding c_2. As shown in Figure 2.1, the embeddings are the second last layers of Neural Network 1 and 2 (their last layers are predicting layers, which are not shown in this figure). When λ equals 0 or 1, this whole model is the same as either single-modal neural network.

The benefits of the no-harm factor are as follows, and we will talk about these benefits in detail in Section 3.2.3:

- It is easy to apply because we do not need to explicitly explore how and where to combine the embeddings.
- It learns the importance of embeddings without introducing too many learnable parameters, thus limiting the complexity gain.
- It simplifies the design of the multi-modal neural networks.

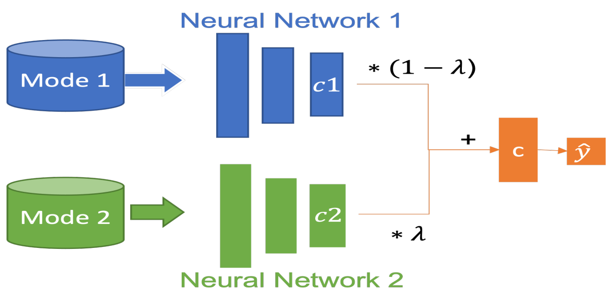


FIGURE 2.1: Implementation of no-harm factor λ in multi-modal architecture. λ controls the importance of one embedding c_1 over another embedding c_2. When λ is a scalar, λ is shared by all neurons in embedding c_1. When λ is a vector, each neuron in embedding c_1 has a specific λ_i value. The same goes to c_2.

2.5 Background

2.5.1 Autoencoder

An autoencoder is a type of neural network that learns a representation of input data in an unsupervised manner [20]. In particular, an autoencoder is a pair of functions called an encode and a decoder, which can be defined as functions ϕ and ψ. For example, in the MNIST problem, the encoder maps the high dimensional input data $\mathcal{X} \in \mathbb{R}^{784}$ into a low dimensional space $\mathcal{H} \in \mathbb{R}^{15}$ and the decoder maps from the low dimension space $\mathcal{H}$ back to the high dimensional space $\hat{\mathcal{X}} \in \mathbb{R}^{784}$ (Figure 2.3). These two functions are composed to have the mapping from the high dimensional space back to the high dimensional space be as close to the identity function as possible. Of course, the exact values for the first two dimensions are dependent on the size of input modalities.

$$\phi : \mathcal{X} \to \mathcal{H}$$
$$\psi : \mathcal{H} \to \hat{\mathcal{X}}$$

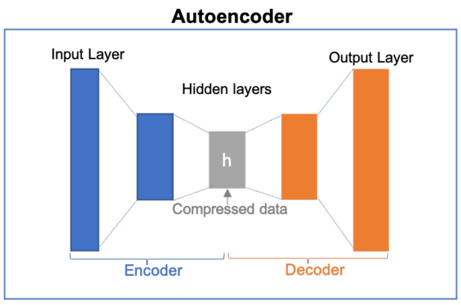


FIGURE 2.2: An example of an autoencoder. The encoder "compresses" input data into hidden space, and the decoder mirrors the encoder to reconstruct the input data using compressed representation. The goal of autoencoder is to learn a lower dimension representation that captures most important parts of the data in an unsupervised manner.

$$\phi, \psi = \underset{\phi,\psi}{\arg\min} ||X - (\psi \circ \phi)X||^2$$

To foreshadow the key results of this thesis, we observe that autoencoders are only the most basic architecture that we have studied. As we will show in Section 3, the architectures, input modalities, and training methods all play essential roles in maximizing model performance using multi-modal data.

Encoders, Embeddings, and Representations

The architecture of autoencoders is non-trivial since the high dimensional is being "compressed" into the low dimensional space, and this compression can only be accomplished if the encoder "understands" something about the structure of the data. Our goal is to test what exactly the encoder is doing to effectively compress the data. The low dimensional space $\mathcal{H}$ is usually called embedding of the input. Data $\mathcal{X}$ can have multiple embeddings $\mathcal{H}$ as the outputs of multiple encoders functions ϕ, and each embedding is a representation of this data mapped into a lower dimension. In the rest of this paper, we will call a neural network an encoder if it compresses inputs into lower-dimensional space and a lower dimension vector an embedding. If we use these embeddings for downstream tasks, they will be called representations because the downstream tasks only need a representation of the full input features.

2.5.2 Multilayer Perceptron - MLP

A multilayer perceptron is a collection of fully connected layers where each neuron of a layer is connected to all neurons in the next layer. A fully connected layer is a function from $\mathbf{R}^m \rightarrow \mathbf{R}^n$. Each linear layer is usually followed by a nonlinear function called activation function σ. The i-th layer $y_i \in \mathbb{R}^m$ is computed as follows where w_i is a learnable weight assigned to each neuron, and b is the bias:

$$y_i = \sigma(w_i x_1 + \ldots + w_m x_m) + b$$

Without activation functions, a neural network is just a linear function whose complexity is limited. No matter how many layers the neural network has, hidden layers have no effect because the output is a linear combination of the input [27]. Two activation functions are used in this research: sigmoid and ReLU [37]. The sigmoid function is a squashing function as it squashes input values within range (0,1), which

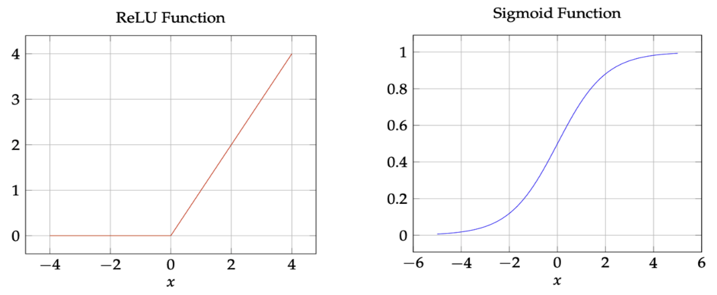


FIGURE 2.3: Activation functions we have used in this research. We use ReLu function (left) to transform input values so that they were not negative. We use sigmoid function (right) to squash input values into range (0,1).

is especially useful when we have to predict the probability as an output. It is denoted by:

$$y = 1/(1 + exp(-x))$$

ReLU stands for rectified linear unit, a function that outputs 0 if the input is equal or smaller than 0, and outputs the input directly if the input is positive. It is denoted by:

$$y = max(0, x)$$

One major advantage of ReLU is sparsity. Sparsity arises when x is smaller than 0; thus using ReLU gives rise to truly sparse representations. Some of the reasons that make parse representations appealing are: it can disentangle information as small input changes will not modify the following non-zero features; it creates variable-size representation because a variable number of zero neurons allows a model to control the effective dimensionality of the representation; it makes the representations more likely to be linearly separable in high-dimensional space [15].

2.5.3 Convolutional Neural Network - CNN

Convolutional Neural Networks(CNN) are specialized to learn useful features in higher dimension data such as images [1]. CNNs are consist of convolution, pooling, and fully connected layers. As shown in Figure 2.4, the convolutional layers utilize 2D kernels, matrices of weights, to capture 2D features such as lines or corners. Pooling layers usually follow convolutional layers to "summarize" the learned features. An output of convolution or pooling is also called a feature map.

The advantage of CNN over an MLP is a smaller number of parameters, and this advantage has prompted many achievements in complex tasks that are too challenging or inefficient for MLPs [1]. For example, in Figure 2.5 to learn a 2x2 feature map over a 3x3 image, a convolutional layer only needs 4 weights of a kernel. In contrast, a fully connected layer needs a sparse 9x4 kernel matrix, whose first dimension is the total number of pixels in the image and whose second dimension is the total number of convolutions.

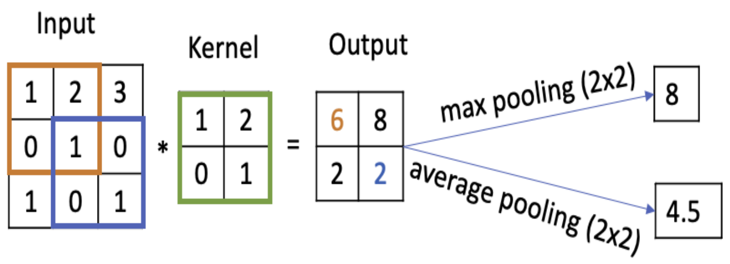


FIGURE 2.4: An example of convolution and pooling. In this example, the stride is 1x1, which means the kernel, or filter, slides on the input image one pixel at a time from left to right and from up to down. Each convolution output a pixel in the output image, so an output of a 2x2 image is the result of 4 convolutions. Max pooling is to output the max value over a window, which is the 2x2 output in this example. Average pooling is to output the mean value over a window.

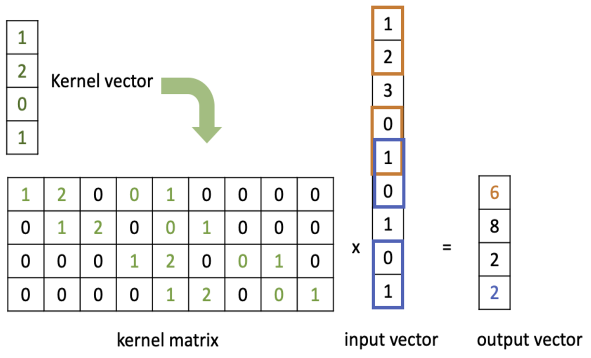


FIGURE 2.5: Number of weights in one convolution: CNN vs. MLP. To learn a 2x2 feature map over a 3x3 image, a convolutional layer only needs a 2x2 kernel (Figure 2.4), while a fully connected layer needs a sparse 9x4 kernel matrix, whose first dimension is the total number of pixels in the image and whose second dimension is the total number of convolutions.

2.5.4 Multi-modal Representation Learning

Although it is easy for human beings to perceive the world through comprehensive information from multiple sensory organs [43], how to endow machines with analogous cognitive capabilities brings up many challenges, one of which is the heterogeneity gap in multi-modal data [22]. In addition, multi-modal representations should capture the complementary information and avoid redundancy among modalities [3]. In this paper, we use representation and embedding interchangeably as the representations are embedded in the hidden layers of neural networks. The models learn to map higher-dimensional inputs to lower-dimensional embeddings to extract essential features of the modalities for downstream tasks. Based on what we understand from the papers mentioned above, there are two main categories of representation learning: joint representation and coordinated representation (Figure 2.6).

Joint representation projects uni-modal representations into a multi-modal space through operations such as concatenation [18] or tensor fusion [72]. Coordinated representation synchronizes uni-modal representation learning with restrains. The

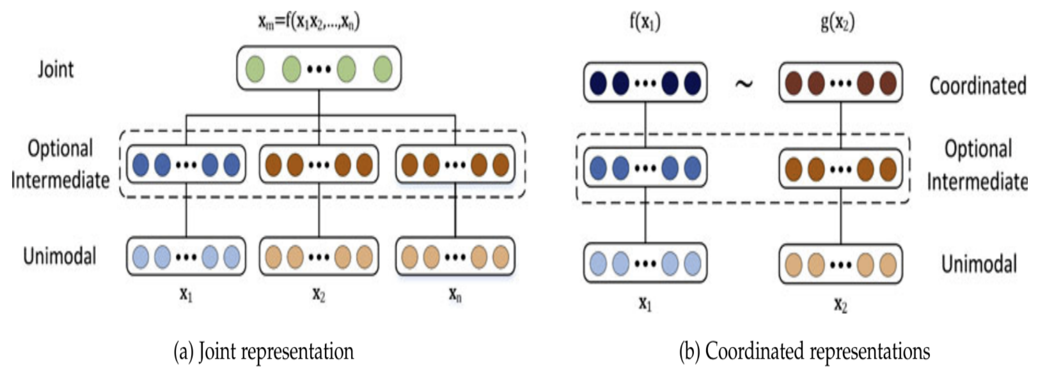


FIGURE 2.6: Image from [3]. Two general types of multi-modal representation: joint representation and coordinated representations. Joint representation projects uni-modal representations into a multi-modal space. Coordinated representation keeps uni-modal representations and synchronizes representation learning with restrains such as reinforcing similarity, correlation, or clustering.

restrains can enforce high similarity [68], high correlation [25], or clustering [69] among multiple representations. With these restraints, each uni-modal representation can capture intra-modality and inter-modality information by adjusting to representations from other modalities.

Due to this thesis's scope, we only experimented with a limited amount of representation learning techniques. We used join representation first because it allowed us to train end-to-end models. We mostly concatenated or summed the penultimate layer of uni-modal neural networks as the data representation. Then we worked on coordinated representation to improve the correlation between modalities, using a self-supervised learning technique called Barlow Twins [73].

Barlow Twins

As we mentioned in the last section, multi-modal representations should avoid redundancy among modalities [3]. When we pretrained single-modal models or trained multi-modal models end-to-end on the same task, our only goal was to reduce the prediction loss. We didn't measure how redundant the representations were to each other or enforce any representation qualities such as similarity or correlation between them. Therefore we used Barlow Twins, a similarity enforcing and redundancy reducing representation learning technique [73]. Barlow Twins is not a multi-modal representation learning method, but it uses data augmentation to randomly distort batches of samples to learn representations invariant under these distortions. In multi-modal learning, we see two modalities as nature's distortions of the same objects. Barlow Twins can take two modalities as two augmented views and increase correlations and avoid redundancy between embeddings of these views (Figure 2.7). Its objective function measures the cross-correlation matrix between the embeddings of two distorted versions of a batch of samples and tries to make this matrix close to the identity. As Barlow Twins is also relatively new, conceptually simple, and easy to implement, we decided to use this technique in our multi-modal experiments.

There is only one change needed to make Barlow Twins multi-modal. Instead of using the same encoder, we use different encoders to create embeddings because the modalities were heterogeneous (Figure 2.8). The encoders are the same single-modal

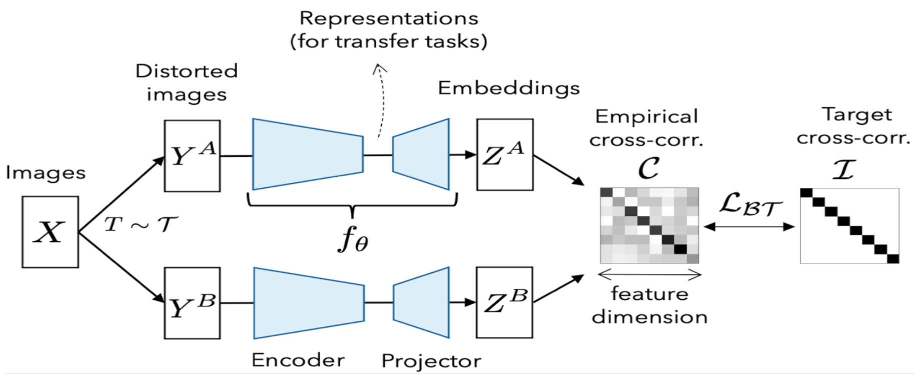


FIGURE 2.7: Figure 1 in [73]. Barlow Twins original implementation. Distorted batches of input samples share the same encoder and projector as function f_θ, which outputs embeddings that are as correlated as possible while avoiding redundancy between the components of these vectors. The closer the cross-correlation matrix is to the identity matrix, the smaller the loss value.

neural networks (MLP for tabular data and CNN for image data) without prediction layers, and the embeddings are the second-last layers of these networks.

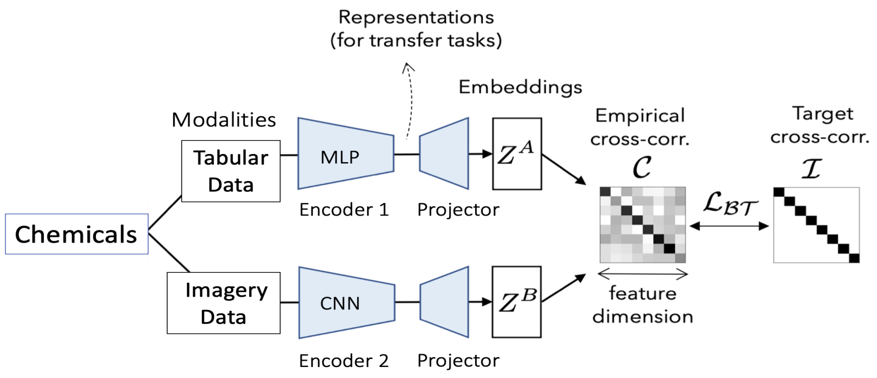


FIGURE 2.8: Multi-modal Barlow Twins implementation. Instead of using the same encoder, we use different encoders to create embeddings because the modalities are heterogeneous. The encoders are the same single-modal neural networks, and the embeddings are the second last layers of these networks.

Barlow twin's loss function was designed to increase correlation and reduce redundancy of the embeddings. This loss function measures how close the cross-correlation matrix is to its identity matrix. The loss function contains 2 terms: invariance term and redundancy reduction term. The invariance term is the sum of squared differences between diagonal elements of the cross-correlation matrix and the identity matrix. The invariance term makes the embeddings invariant to the heterogeneity of modalities. The redundancy reduction term is the sum of the squared value of off-diagonal elements of the cross-correlation matrix. This term decorrelates the different components of the embeddings. Lambda term λ decides the trade-off between correlation increasing and redundancy reducing.

Barlow Twins distinguishes itself from other methods by its innovative loss function $\mathcal{L}_{\mathcal{BT}}$:

$$\mathcal{L}_{\mathcal{BT}} \triangleq \underbrace{\sum_i (1 - \mathcal{C}_{ii})^2}_{\text{invariance term}} + \underbrace{\lambda \sum_i \sum_{j \neq i} \mathcal{C}_{ij}^2}_{\text{redundancy reduction term}}$$

The cross-correlation matrix is computed between the embeddings along the batch dimension. Below is the equation, where b indexes batch samples and i, j index the vector element of the embeddings. $\mathcal{C}_{ij}$ is a square matrix whose dimensions are the sizes of embedding z^A and z^B , and with values comprised between -1 (i.e., perfect anti-correlation) and 1 (i.e., perfect correlation).

$$\mathcal{C}_{ij} \triangleq \frac{\sum_b z^A_{b,i} z^B_{b,j}}{\sqrt{\sum_b (z^A_{b,i})^2} \sqrt{\sum_b (z^B_{b,j})^2}}$$

2.5.5 Multi-modal Fusion Techniques

Multi-modal fusion is to integrate information from multiple modalities for classification or regression. Dating to 33 years ago [71], it is one of the most researched topics of multi-modal machine learning [24], [55], [5], [46], [61], [14], [75], [9]. Encouraged by the growing availability of multi-modal data, researchers have explored multi-modal fusion to capture complementary information not observed in a single modality. The multi-modal fusion approaches are generally categorized based on three aspects: how to fuse, when to fuse, and what model is used to fuse [3], [2], [64].

As regards how to fuse, there are aggregation-based fusion, alignment-based fusion, and a mixture of them [64]. Aggregation-based fusion uses a particular operation (e.g., averaging, concatenation) to combine multiple neural networks into one network (A in Figure 2.9). Alignment-based fusion uses an additional term that regularizes certain properties between modalities (e.g., embedding distance [56], [63], pixel similarity [10]) (B in Figure 2.9). This regularization term is added to the loss function to make embeddings adaptive to other modalities. The regularization loss can be added to prediction loss for full model gradient descent, and it can be used alone for coordinated representation learning.

As regards when to fuse, multi-modal fusion can be specified as early, late, and hybrid fusion [3]. Early fusion learns to exploit the interactions between low-level features. The early fusion occurs at the input level so that one network is needed for multiple modalities. Late fusion combines uni-modal decisions, allowing more flexible single modal training while ignoring low-level interactions between modalities. Hybrid fusion exploits the advantages of both methods mentioned above. We start with early fusion and late fusion in our experiments because they are easier to apply. Our late fusion layers are not the decision layers but the second last layers of uni-modal neural networks.

Regarding the models used for fusion, the model-based fusion categories [3] are multi-modal kernel learning [6], graphical models [23], and neural networks [32], [3], [56]. Because our topic is multi-modal deep learning and the related methods within this scope, we only explore the neural network fusion.

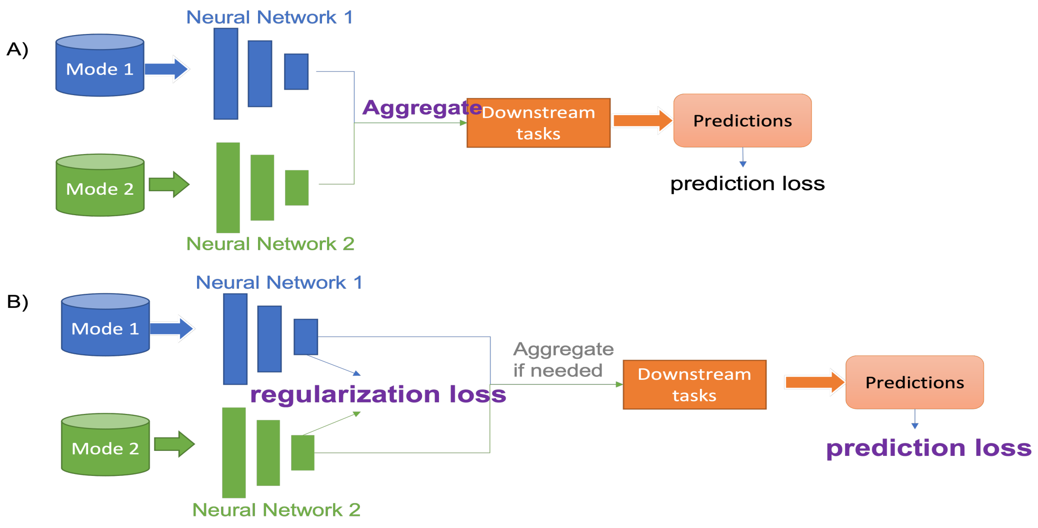


FIGURE 2.9: Two general types of multi-modal fusion: Aggregation-based fusion and Alignment-based fusion. Aggregation-based fusion (A) aggregates (e.g., averaging, concatenation) representations learned in both networks to get a joint representation. Alignment-based fusion (B) introduces an additional loss term to make embeddings adaptive to each other, and the adaptive embedding layers are not limited to one layer. The representations learned can be aggregated for the same downstream tasks or stay separated for different tasks. We can add regularization loss to prediction loss or use it alone for coordinated representation learning.

Controlling Factors during Fusion

We consider our proposed no-harm factor *lambda* as a controlling factor that "controls" the weight of one modality to another. There are other such control factors in the literature. For example, in [62], a controlling factor was calculated using error metrics so that the individual model with higher error should be given less weightage as compared to another individual model at the fusion layer. However, the error metrics were not computed in training time and researchers needed to use the right combination of metrics to correctly indicate which modality was more important. On the contrary, the no-harm factor is a simple learnable parameter like other weights between neural network layers. In [52], researchers used fusion gates to control the degree to which modalities contribute to the final prediction. The fusion gates were made of highway layers [57] that defined how much of a modality embedding should be transformed or passed directly to the next layer. Compared to the no-harm factor, this method requires more hyperparameter tuning, such as the number of gates and size of linear layers inside of the gates, while the no-harm factor does not introduce any new hyper-parameter. Therefore, even though methods that control the weights of modalities are not new, we consider the no-harm factor a contribution because it is straightforward, easy to apply, and practical in our unique problems.

Chapter 3

Methodology

3.1 Datasets and Prepossessing

3.1.1 MNIST Dataset and Its Synthesized Datasets

To better understand multi-modal learning with a sufficient amount of data, we have created synthetic multi-modal data using the MNIST dataset [70]. MNIST is a dataset of 70000, 28x28 pixel grey-scale images of handwriting digits from 0 to 9. 60,000 images are in training and validation sets, and 10,000 images are used as a testing set (Table 3.1). The first modality is the original modality missing some bottom rows, and the second modality was the blurry and smaller version of the original modality. We cut the original images by rows from the bottom up and interpolated the original images into small and blurry images to create different data modes(Figure 3.1). In this case, the synthesized modalities do not have the complete information of the original modality. To make the synthesized modalities have various amount of partial information, we cut the bottom rows by 5, 10, 15 and 20, and interpolated the images by 1/2, 1/4 and 1/7. So when a modality missing 20 bottom rows was combined with the modality scaled down by 1/7, the information combined should be less than a modality missing 5 bottom rows was combined with the modality scaled down by 1/2.

Data	Total
Training/Validation	60,000
Testing	10,000

TABLE 3.1: MNIST dataset. Pre-split train-test dataset from PyTorch. Because this is for an unsupervised task, no label is needed.

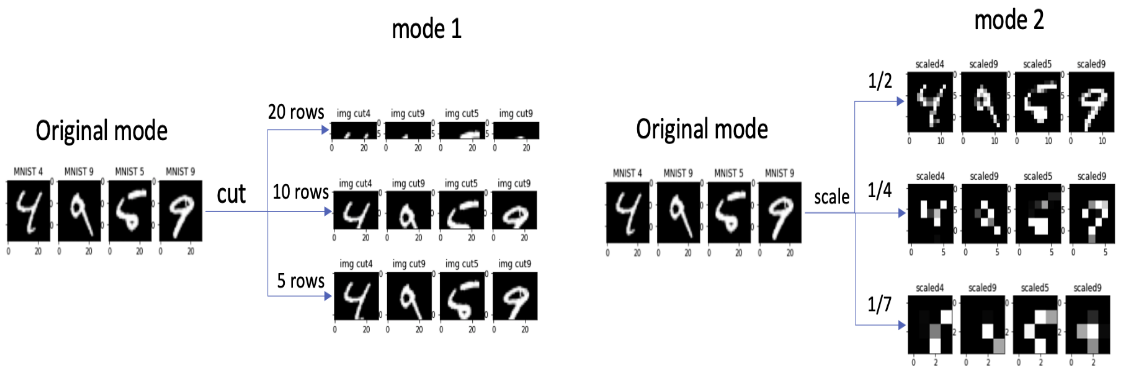


FIGURE 3.1: Examples of synthesized modalities of the MNIST data. Mode 1 is the partial versions(e.g., remaining top 8 rows if we cut bottom 20 rows) of the original data. Mode 2 is the blurry and small version(e.g, 1/2 size) of the original data. The purpose of the synthetic data was to extract partial information from the original data.

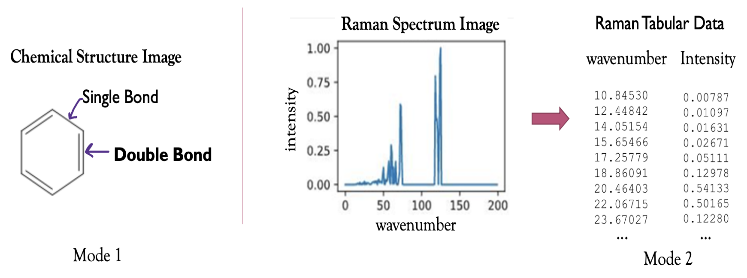
3.1.2 Chemical Structure and Raman Spectrum Datasets

We have chemical structure images and Raman tabular data of 72 chemicals. Because the double bond labels are highly imbalanced, we created new labels based on double bond ranges (Table 3.2). Each chemical has two modes (Figure 3.2). On the left side of the figure, a chemical in mode 1 is a structure image of 80x80 pixels. The single-line edges are single bonds, and the double-line edges are double bonds. In chemistry, a double bond is a covalent bond between two atoms sharing two pairs of electrons as opposed to one pair in a single bond [58].

On the right side, each Raman table of mode 2 is the tabular version of a Raman spectrum, which is considered the "fingerprint" of a chemical [21]. (Figure 3.3). Each Raman tabular sample has two columns — Intensity and Wavelength – that represent the intensity of the scattered photons (y-axis) for each energy (frequency/wavenumber/Raman shift) of light (x-axis). To convert 2D Raman data into 1D vectors to train an MLP, we used the min and max wavelength values in training set to define fixed windows of 10 wavelength units, then averaged the intensity in these windows, so each Raman sample became a vector of length 332 (Figure 3.4).

	0 (Double Bonds<6)	1 (Double Bonds=6)	2 (Double Bonds>6)	Total
Data	24	23	25	72

TABLE 3.2: Chemical structure and Raman spectrum datasets. double bonds labels are highly imbalanced in this small dataset, so we created a new label that categorized double bonds into 3 balanced classes: 0 as double bonds is smaller than 6, 1 as double bonds is equal to 6, 2 as double bonds is bigger than 6. The test set was created after the training/validation dataset, and the labels are imbalanced.



wavenumber	Intensity
10.84530	0.00787
12.44842	0.01097
14.05154	0.01631
15.65466	0.02671
17.25779	0.05111
18.86091	0.12978
20.46403	0.54133
22.06715	0.50165
23.67027	0.12280
...	...

FIGURE 3.2: An example of a chemical with two modes. Mode 1 is the black and white structure image of 80x80 pixels. Mode 2 is the Raman tabular data. The structure image shows that this chemical has 3 double bonds(two-line edges), so the label of this sample is 0 because the double bond number < 6.

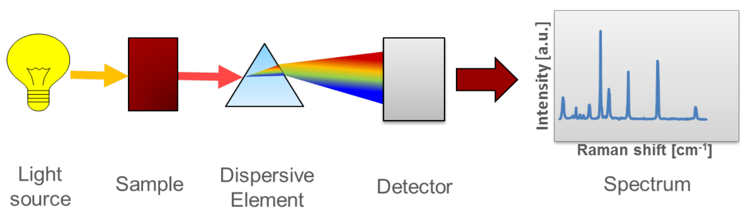
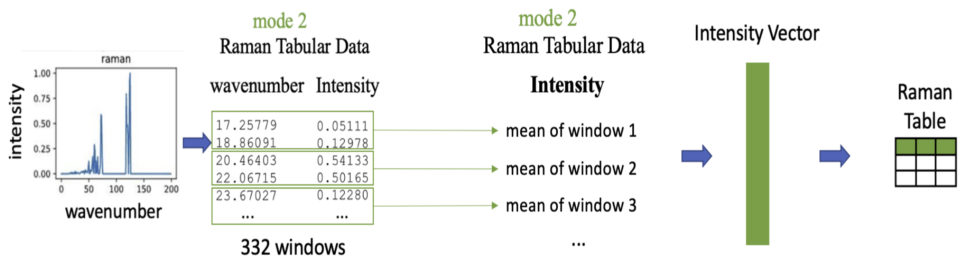


FIGURE 3.3: Figure 2 in [51]. Schematic of the measurement principle in a Raman spectroscopy. A laser light source hits the sample and the light interacts with the molecules and is scattered in all directions. While almost all the the light just pass thought the sample without interaction, a small percentage of this light is Raman scattered. Each Raman scattered light waves has a different wavelength compared to the laser light source because the sample absorbs a certain amount of energy. And such differences in wavelength are recorded along with the intensities of the Raman scattered light waves, leading to the final spectrum.



wavenumber	Intensity
17.25779	0.05111
18.86091	0.12978
20.46403	0.54133
22.06715	0.50165
23.67027	0.12280
...	...

FIGURE 3.4: Convert a Raman spectrum into a 1D vector. The Raman tabular data we have are 2D data that was converted from the Raman spectrum. We averaged the intensity of Raman data in fixed windows in the size of 10 wavenumbers. These windows started from the minimum and ended on the maximum wavenumber value among all training set Raman samples. Each Raman spectrum was eventually converted into an intensity vector of 332 features. Each vector makes up one row of the Raman dataset as a table.

3.1.3 Public Biodegradation Dataset

We used the same QSAR biodegradation dataset [47] that was used to predict biodegradability by Goh et al. [18]. The 41 selected chemical descriptors, referred to as Ballabio-40, were obtained from Mansouri et al. [41]. Following the steps in the paper, we mixed and split training, validation, and test datasets while keeping their original sizes. There are 1725 data points in total. Each chemical is identified by a SMILES string, 41 selected chemical descriptors, and a label as either non-biodegradable (NRB) or biodegradable (RB). SMILES stands for Simplified Molecular Input Line Entry Specification, a notation that describes chemical structures [66]. We created the image modality of this dataset using the same preparation mentioned in [18] based on two previous papers [16] and [17]. Each image of 80x80 pixels has four channels that capture 5 different chemical properties (Figure 3.5).

Data	RB (Biodegradable)	NRB (non-biodegradable)	Total
Training/Validation	356	699	1055
Testing	191	479	670

TABLE 3.3: Biodegradability classification dataset. "RB" means biodegradable, and "NRB" means non-biodegradable. The original full dataset has 1725 data points with fixed training, validation and test sizes. We kept these sizes after shuffling and splitting the full dataset.

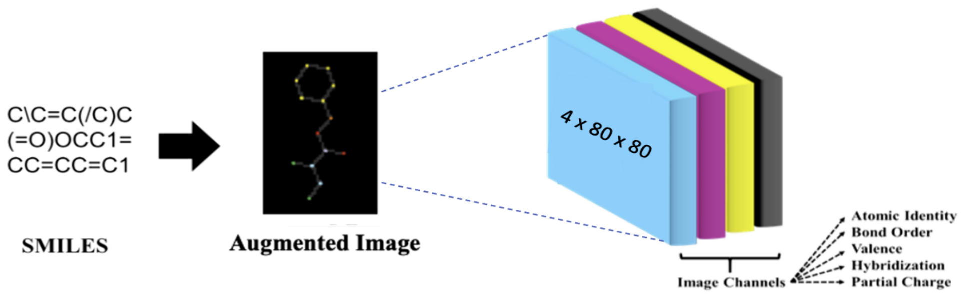


FIGURE 3.5: Convert a SMILES string to an augmented image. Picture in the right is from [17]. Each SMILE string was converted into an augmented image of 80x80 pixels and 4 channels. Each channel contains one or two chemical properties such as Atomic Identity or Valence. The augmented images have more chemical information than black and white single-channel molecular images in the last experiment using Raman spectrum.

3.2 Architecture and Optimization

3.2.1 Reconstructing MNIST Modality Using Encoder-decoders

The architecture of the autoencoder is shown in Figure 3.6. The training batch size was 128. Layer sizes were fixed to [input_size, 128, 81, 128, 784] to control the architecture in our experiments, despite the various input sizes because of modality changes. More architecture details are in Appendix A. Strictly speaking, this model is not an autoencoder but a combination of an encoder and a decoder because it reconstructed the original images instead of the input images. We used Root Mean Square Error (RMSE) as the loss function and Adam optimizer with a default learning rate of 0.001. Each autoencoder was trained for 100 epochs with an early stopping criterion of 0.0005 for 10 epochs.

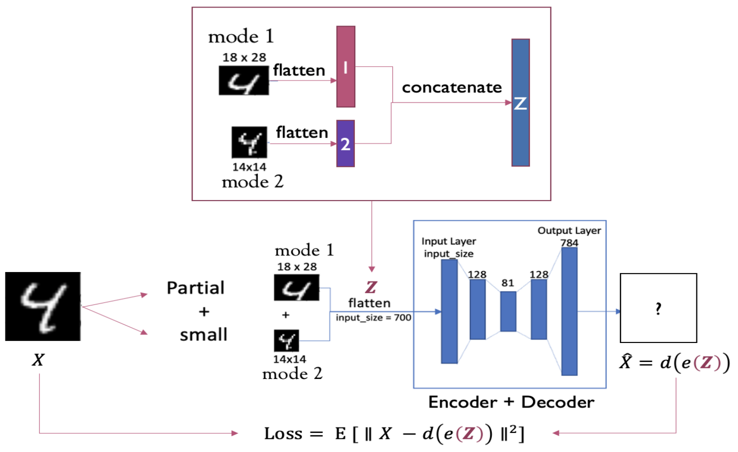


FIGURE 3.6: The architecture of the encoder + decoder neural network in Experiment 1. Input images were flattened and then concatenated into vector z, which was the input of the neural network. The input layer size depended on the size of z, but the rest of the layers were fixed as [128, 81, 128, 784]. Evaluated by mean square error loss, the encoder + decoder network learned to output $\hat{X}$, a reconstruction of original image X.

3.2.2 Predicting Double Bond of Chemicals Using MLP and CNN

The architecture of the combined model was a combination of MLP and CNN models (Figure 3.7). Model A was a convolutional neural network including convolutional, pooling, and fully-connected layers. Model B was a feedforward MLP that took Raman tabular data. We will not discuss the model in greater detail because we did not focus on optimizing the model performance. In addition, our goal was to compare multi-modal learning with single-modal learning, so the architectures of the combined model should not add too much complexity to the single-modal models. More architecture details are in Appendix A. We used CrossEntropyLoss as the loss function. For the uni-modal model trained with the Raman table, we used stochastic gradient descent (SDG) optimizer with a learning rate of 0.5. The CNN model was optimized by SGD with a learning rate of 0.01. The third modal, a multi-modal MLP, used SGD with a learning rate of 0.5. The training batch size was 32.

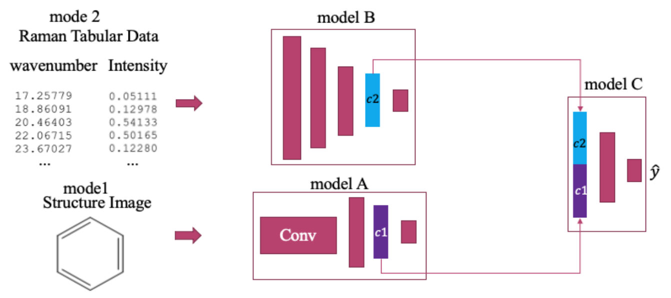


FIGURE 3.7: The architecture of the multi-modal deep neural network to predict double bonds. Model A and B are the single-modal neural networks for modality 1 and 2. Mode 1 is single channel structure image data of 80x80 pixels, and mode 2 is the Raman tabular data. Raman tabular data were converted into vectors of intensity as the intensity values were aggregated (mean or sum) over fixed windows of wavelengths.

3.2.3 Predicting Biodegradation of Chemicals Using MLP and CNN

Benchmarking using the original dataset was not our goal. Because we randomly mixed and split the datasets, the data samples in our training, validation, and test datasets should be different from theirs. In this case, it is not convincing to compare the performance of our models with the performance they reported, so we used the architecture of their best model (Figure 3.8) as a baseline for our validation and test sets. Of all the training methods explored in the baseline experiments, pretraining two models separately was not mentioned. For modality fusion, the baseline model used concatenation, leaving other fusion methods such as averaging or tensor fusion [72] to be explored.

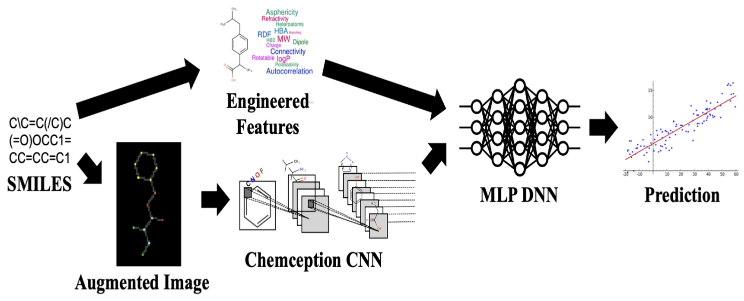


FIGURE 3.8: Figure 1 in [19]. Baseline model to predict biodegradation. In the sequential setup, the Chemception CNN model was first trained on the chemical images to predict biodegradation. After the training was completed, the CNN weights were frozen. Then the penultimate layer output was concatenated with the chemical descriptors. Finally, an MLP network used the combined vector to classify the chemicals.

The CNN block is called Chemception [16] as shown in Figure 3.9 and Figure 3.10. Chemception is a general-purpose neural network in chemical research that processes molecular images. The MLP block is a standard feed-forward neural network that uses molecular descriptors as the input data. In the baseline model, the descriptors were concatenated with the penultimate layer from Chemception. Then the joined vector was fed into an MLP classifier. However, to improve model performance, Goh et al. eventually used a more comprehensive set of 1400 descriptors and pre-trained the Chemception block on numerous chemical rules [19] and fine-tuned it using the current dataset. Since our goal is not to benchmark, we have focused on improving the model using our multi-modal techniques without pretraining on other datasets or changing the descriptor modality.

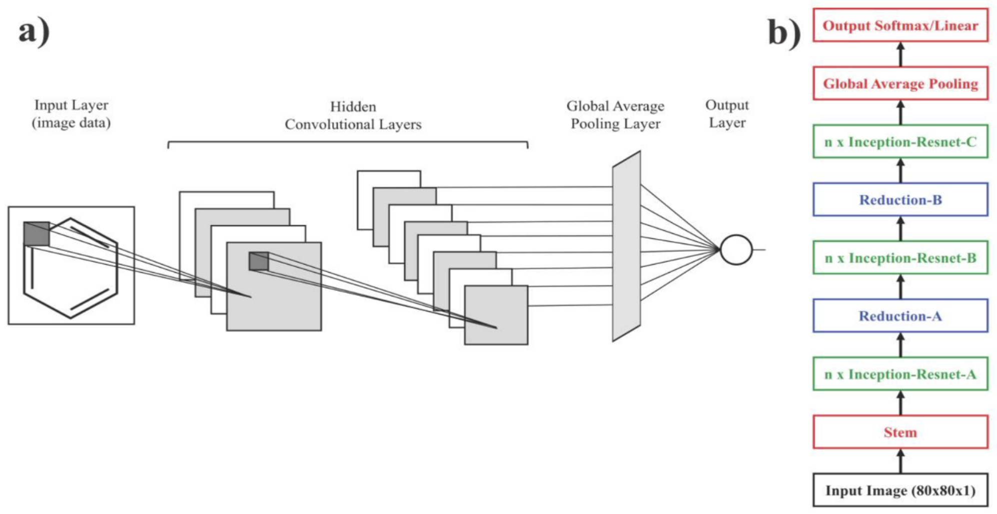


FIGURE 3.9: Figure 3 in [16]. Building blocks of the baseline model to predict biodegradation. a) Typical convolutional neural network and b) High-level architecture of Chemception. Each block of Chemception contains some convulsion and/or pooling layers and serves unique purposes. Stem, Reduction-A, and Reduction-B blocks increase channel numbers from 4 to 16 to 64 to 126 and decrease the input sizes(width, height) from (80,80) to (39, 39) to (19, 19) to (9, 9). Inception-Resnet-A, Inception-Resnet-B, Inception-Resnet-C use combinations of different convolutions layers for feature learning without changing input dimensions. See more details in Appendix A.

The metric is called classification error rate (Er) [19] , which is a function of sensitivity (Sn) and specificity (Sp) calculated based on true negative(TN), true positive(TP), false negative(FN), and false positive(FP):

$$Er = 1 - (Sp + Sn)/2$$

$$Sp = TN/(TN + FP)$$

$$Sn = TP/(TP + FN)$$

Because our dataset is imbalanced – we have 356 labeled as positive and 699 as negative – this metric can capture how well the model learns to predict imbalance labels. We used binary cross entropy loss (BCELoss) as the loss function. We used the random search for hyper-parameter tuning. For the uni-modal MLP trained with descriptors, we used Adam optimizer with a learning rate of 0.00006, batch size 64, 300 epochs, and a dropout rate of 0.58 in each layer. For Chemception CNN, we used the Adam optimizer with a learning rate of 0.0001, batch size 32, and 150 epochs. The third modal, a leaner layer, had the same hyper-parameters mentioned for the MLP.

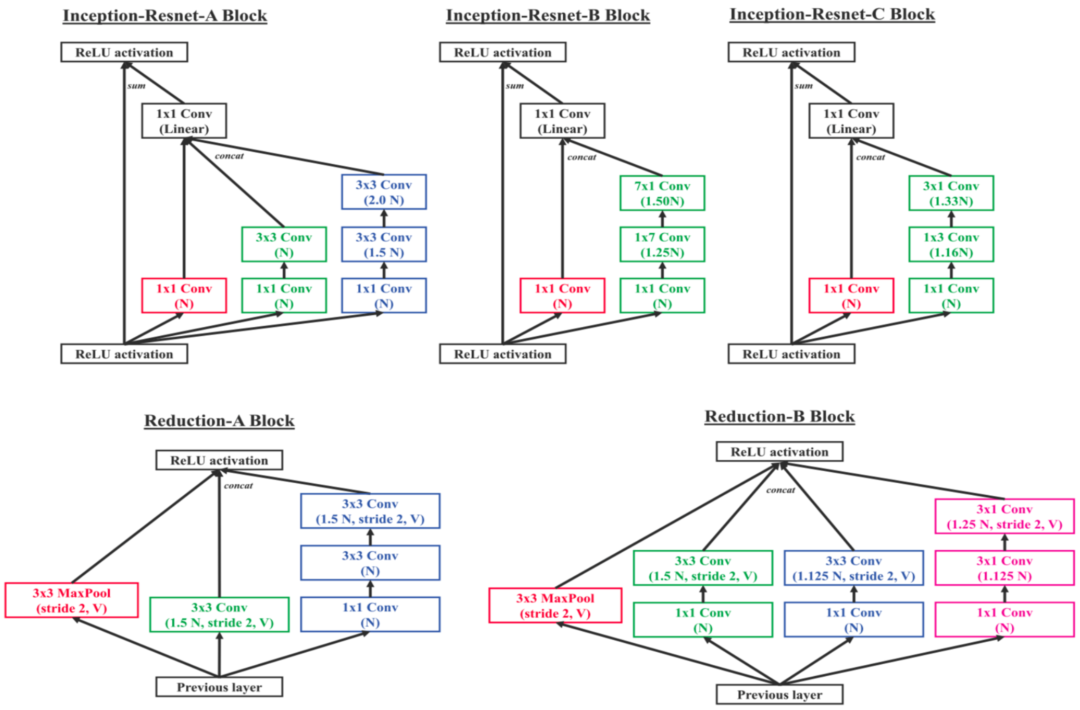


FIGURE 3.10: Figure 4 in [16]. Details of the inception/reduction blocks in Chemception. The Inception-ResNet blocks combine the Inception models and Residual learning. The Inception aspect of these blocks is using different convolution kernels on the same input. The Residual learning aspect of these blocks is adding the input of blocks to the output of convolution layers. Reduction blocks reduce input sizes(width, height) while increasing input channels. See more details in Appendix A.

No-harm factor

We designed a no-harm factor λ as a fusion technique. The goal of the no-harm factor λ was to guarantee that the performance of the combined model was at least as good as the best single modal. When λ was a scalar, it controlled the importance of the embeddings c. When λ was a vector, each element in the vector controlled the importance of the corresponding neuron in the embeddings. We assigned λ to one embedding and 1 - λ to the other embedding as we only used two modalities. If lambda was learned to be 1 or 0, one of the embeddings would be ignored, which meant adding the ignored modality did not help the performance. There was a trade-off between the complexity of the model and performance protection when choosing between concatenation (Figure 3.11, right) and a no-harm factor (Figure 3.11, left). The concatenation fusion method could learn intra-embedding and inter-embedding relationships, while the no-harm factor only captured simpler inter-embedding relationships. The intra-embedding relationship within an embedding means that the neurons of this embedding are connected with other neurons. The inter-embedding relationship between two embeddings means that the neurons in one embedding are connected with the neurons in the other embedding.

$$c = (1 - \lambda) * c_1 + \lambda * c_2$$

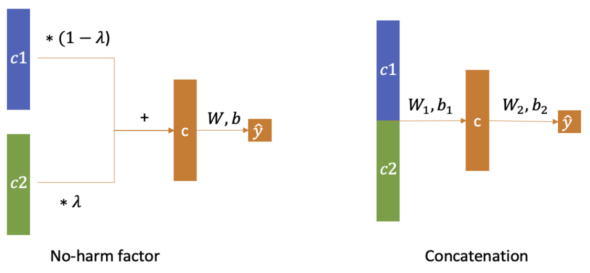


FIGURE 3.11: Implementation of no-harm factor λ on embedding layers and following prediction layer. Compared to embedding concatenation, the no-harm factor learns to control embedding importance and reduces learnable parameters during training. First, λ controls the importance of one embedding c_1 over another embedding c_2. When λ is a scalar, λ is shared by all neurons in embedding c_1. When λ is a vector, each neuron in embedding c_1 has a specific λ_i value. The same goes to c_2. Second, when using a no-harm factor, the learnable parameters connecting these layers are λ, W, b, with W, b as the weight matrix and bias term that connect embedding c and the prediction layer. When using concatenation, the learnable parameters are W_1, b_1, W_2, b_2. W_1, b_1 are the weight matrix and bias term that connect concatenated embeddings c_1, c_2 and embedding c. W_2, b_2 are the weight matrix and bias term that connect embedding c and the prediction layer.

We combined the no-harm factor with different training methods to see how well the model could learn the no-harm factor and other weights in different model sections. For example, if we trained the model end-to-end, all model weights could update along with the no-harm factor during backpropagation. Moreover, if we trained the model with both MLP and Chemception pre-trained and frozen, only the weights of the last layer could adjust with the no-harm factor. The combined model would learn differently under different dynamics between the number of learnable model weights and the no-harm factor.

There are 3 main benefits of the no-harm factor. First, it is easy to apply. Because it fixes the way we combine the embeddings, we only need to consider combining embeddings of the same length. Second, it learns the importance of embeddings without introducing too many learnable parameters, thus limiting the complexity gain. The excessive complexity of the model is important when the data size is small. As shown in Figure 3.11, when a no-harm factor is a scalar, it only adds one learnable parameter to the overall model; when it is a vector, the number of learnable parameters added equals the number of neurons in one embedding: N. In the meantime, concatenation of embeddings adds learnable parameters W_1 and b_1, whose number of elements is $N^2 + 1$. Third, This architecture design ensures that if the learned λ is 0 or 1, the combined model is the same as the best single model, thus causing no harm by adding a modality that's not useful. To achieve this purpose, the single-model networks should have the same layers after their embeddings, and the layers after combined embedding should be the same. This method also simplifies the design of a multi-modal neural network.

One example of the no-harm factor architecture is shown in Figure 3.12. The layers combined with the no-harm factor should have the same length. The following layers after the combination should be the same as the layers after the embeddings in a best single-model neural network. In this case, if the no-harm factor is learned to be 1, the architecture of the combined neural network is the same as the best single-model neural network because the other modality is ignored because of 0 weight at the fusion layer. This design ensures that if the inferior modality is not useful, the multi-modal output is as close to the best single-modal output as possible.

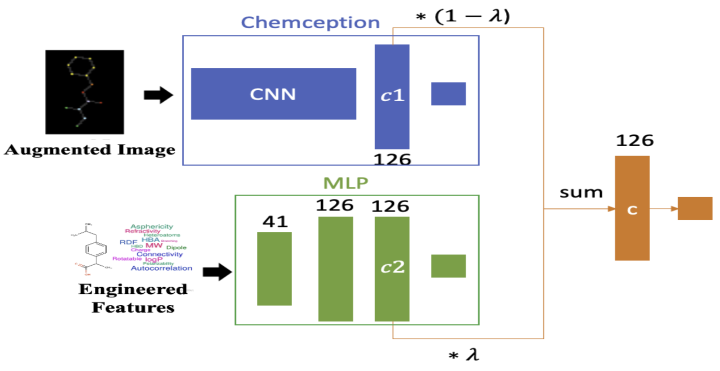


FIGURE 3.12: The architecture of the combined model with no-harm factor λ. The layers combined with the no-harm factor should have the same length. The following layers after the combination should be the same as the layers after the embeddings in the best single-model neural network. In this case, if the no-harm factor is learned to be 1, the architecture of the combined neural network is the same as the best single-model neural network because the other modality is ignored because of 0 weight at the fusion layer. This design ensures that if the inferior modality is not useful at all, the multi-modal output is as close to the best single-modal output as possible.

3.3 Training and Implementation

3.3.1 Multi-modal Training Method

We combined modalities by concatenating the last hidden layers of two models. We had two options to train the combined modal (Figure 3.13). The first training option was to train a combined model end-to-end. The second training option was split into two phases. In the first phase, we pretrained one single-modal network (model A or model B) or both single-modal networks (model A and model B) for the same classification task and froze the pretrained model(s). In the 2nd phase, we combined the second-last layers of the first two models as the input of the third network (model C). Then we train the third network with the individual models pretrained. Models A and B needed very different learning rates when we pretrained them. Thus, it was easier for us to set up separate pretraining than combined training and monitor both networks' training losses. The downside of this separated training method is that models A and B learned representations of their modalities in individual tasks, missing the chance of catching inter-modality information. We first used option II because our goal was to compare multi-modal and single-modal learning. If the easier choice could deliver a better result, we would decide later if we want to optimize the combined model or not.

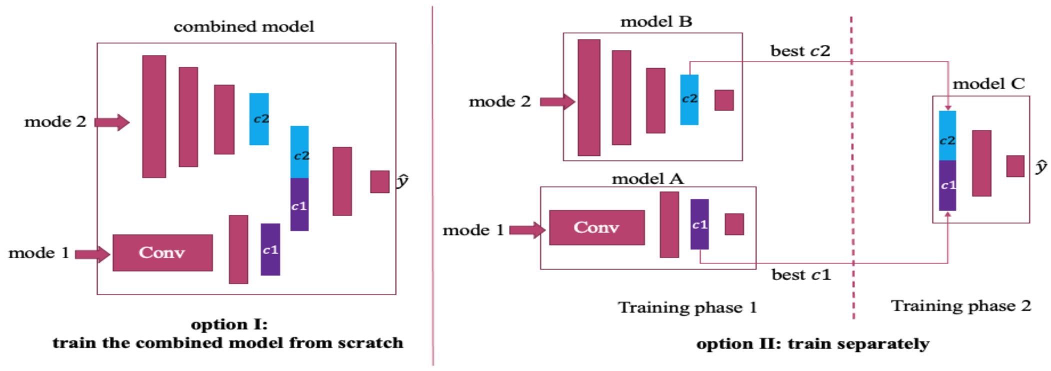


FIGURE 3.13: 2 examples of training options for multi-modal neural networks. Option I allowed us to train combined model end-to-end, Option II was to train two domain-specific models as model A and B, then froze the trained models, so model C got the best hidden layers $c1$ and $c2$ as representations, or "compressed" information,of mode 1 and mode 2.

3.3.2 Implementation Algorithms

Multimodal-learning with Synthesized MNIST Data

To combine the two modes, we flattened both images and then concatenated them as a single input for the encoder-decoder network. Like in a normal autoencoder, the goal of this encoder was to "understand" something about the structure of the input data so the decoder could map the compressed information back to original images. First, we trained the autoencoders using the single mode of partial or full images. Second, we introduced the scaled image to the partial images to check if a second mode could improve the performance of a single-modal autoencoder. If there was an improvement, we wanted to know if multi-modal learning with compromised modalities could perform better than single-model learning with complete information.

We used 5-fold cross-validation on the training set. The best models were chosen based on the best validation loss. Then we tested these 5 learned models on the test set. The training process is shown in Algorithm 1.

Algorithm 1 PyTorch-style pseudocode for multi-modal learning with synthetic MNIST data

```
1:  $mode_1$ = cut images by n rows
2:  $mode_2$ = interpolate images by a scale factor
3:  for i in range(5) do
4:      split train and validation datasets from $mode_1$ and $mode_2$ combined
5:      Make data loader with datasets
6:      model = encoder_decoder()
7:      for $m_1$, $m_2$ in loader do
8:          $c$ = cat($m_1$.flatten(), $m_2$.flatten())
9:          $\hat{y}$ = model($c$)
10:         loss.backward()
11:         optimizer.step()
12:     end for
13: end for
```

Multimodal-learning with Limited Chemical Data

The deep learning task was to predict the number of double bonds of each chemical. We only have 72 data samples. We bootstrapped (with replacement) 100 times on the 72 samples to make training data. We used the out-of-bag samples as validation data. We trained and validated all models 100 times and took the average accuracy to evaluate the performance. If the results showed a significant increase from single-modal to multi-modal learning, we wanted to know that the increase was not because of different model architectures. In the multi-modal setting, we added controlled experiments by replacing either mode with random noise. If the combined model with random noise and normal modality could outperform the single model with either normal modality, we would want to remove the effect of the architecture change. The implementation is in Algorithm Algorithm 2.

Algorithm 2 PyTorch-style pseudocode for multi-modal learning with limited chemical data

```
for $b = 1, 2, \ldots, 100$ do
    Bootstrap with replacement from training and validation sets combined
    Pretrain and freeze classifier $model_1$ with $mode_1$, and classifier $model_2$ with $mode_2$
    $model_1 = f_\theta(m_1)$
    $model_2 = h_\phi(m_2)$

    for $m_1$, $m_2$ in loader do
        # calculate embeddings
        $c_1$ = $model_1$(input = $m_1$, embedding_layer = -2)
        $c_2$ = $model_2$(input = $m_2$, embedding_layer = -2)

        # concatenate embeddings
        $c$ = cat($c_1$, $c_2$)

        # use MLP network as $model_3$ to predict
        $\hat{y} = model_3(c)$

        # optimize $model_3$
        loss.backward()
        optimizer.step()
    end for
end for
```

Coordinated Representation using Barlow Twins

We wanted to know if Barlow Twins could improve model performance by increasing correlation and reducing redundancy between modality representations. Before we combined the learned representations from Barlow Twins, we needed to examine the quality of the embeddings by visualizing the cross-correlation matrix. If the cross-correlation matrix is far from the identity matrix, we doubt that the representations learned could be useful in downstream multi-modal learning tasks. The implementation is shown in Algorithm 3.

Algorithm 3 PyTorch-style pseudocode for multi-modal Barlow Twins

```
# $f_\theta$: encoder network for $mode_1$
# $f_\phi$: encoder network2 for $mode_2$
# lambda: weight on the off-diagonal terms
# N: batch size
# D: dimensionality of the embeddings
# mm: matrix-matrix multiplication
# eye: identity matrix
# off_diag: off diagonal elements of a matrix

create multi-modal data loader using $mode_1$, $mode_2$
for $y_a, y_b$ in loader: do
    # $y_a$ is a batch from $mode_1$, $y_b$ is a batch from $mode_2$
    # Compute representations
    $z_a = f_\theta(y_a)$
    $z_b = f_\phi(y_b)$

    # normalize representations along batch dimension
    $z_a$_norm = ($z_a$ - $z_a$.mean(0)) / $z_a$.std(0)
    $z_b$_norm = ($z_b$ - $z_b$.mean(0)) / $z_b$.std(0)

    # cross-correlation matrix
    $c$ = mm($z_a$_norm, $z_b$_norm) / (N - 1)

    # loss
    $c$_diff = ($c$ - eye(D)).pow(2)
    off_diag($c$_diff).mul_(lambda)
    loss = $c$_diff .sum()

    # optimize step
    loss.backward()
    optimizer.step()
end for
```

Multimodal-learning with Public Chemical Data

Algorithm 4 shows the process of pre-training single-modal models and then learning no-harm factor and prediction. The algorithm shown is just one of the many training examples.

Algorithm 4 PyTorch-style pseudocode for multi-modal learning with biodegradability data

```
Pretrain and freeze classifier $model_1$ with $mode_1$, and classifier $model_2$ with $mode_2$
$model_1 = f_\theta(m_1)$
$model_2 = h_\phi(m_2)$
for $m_1, m_2$ in loader do
    # calculate embeddings
    $c_1$ = $model_1$(input = $m_1$, embedding_layer = -2)
    $c_2$ = $model_2$(input = $m_2$, embedding_layer = -2)

    # combine embeddings using no-harm factor
    $c = c_1 * \lambda + c_2 * (1 - \lambda)$

    # use last linear layer to predict
    $\hat{y}$ = sigmoid(liner_layer($c$))

    # optimize last linear layer and lambda
    loss.backward()
    optimizer.step()
end for
```

Chapter 4

Results

4.1 Multi-modal Learning with Incomplete Information

We combine multiple modalities – with missing information to some extent – to reconstruct the original modality with full information. The baseline model of this experiment is an autoencoder that uses original MNIST images and outputs images that are as similar to the inputs as possible. We tracked the training, validation, and test losses to ensure the models did not overfit or underfit and the results were not random. During training, the baseline model improved steadily and did not overfit because the training and validation lines are smooth and close in the loss plot (Figure 4.1, left image). Also, the test loss is very close to the best validation loss. The same learning pattern goes to a multi-modal encoder-decoder, as shown in the right-side plot of Figure 4.1.

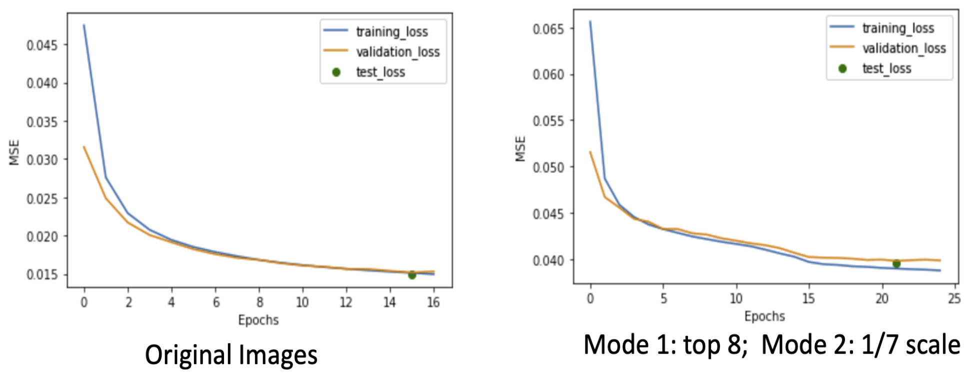


FIGURE 4.1: Sample loss plots of MNIST data reconstruction. The left plot shows the learned losses from single-modal learning, i.e., reconstructing original images using the original images. The right plot shows the learned losses from multi-modal learning, i.e., reconstructing original images using the top 8 rows (mode 1) and scaled by 1/7 (mode 2). We used early stopping, so the total epochs in these plots vary. To test the model, we used the best model saved when the validation loss was the lowest, so the test loss (green dot) on the plot is located at the epoch where the best model was saved.

The baseline test loss was 0.0123 (Table 4.1). In single-modal learning using partial or scaled images only, i.e., the second column or sixth row, none of the models achieved comparable performance to baseline. However, the more rows or smaller the scale, the lower the loss because the modalities contain more information or the images to be reconstructed. Moreover, as we introduced the second modalities, the performance increased. The loss was as good as the baseline when we combined partial images with the top 23 rows and scaled images by 1/2. To illustrate the

	No scaled img	+Scale by 1/7	+Scale by 1/4	+Scale by 1/2
Top 23	0.0148	0.0144	0.0143	**0.0123**
Top 18	0.0214	0.0191	0.0173	0.0133
Top 13	0.0331	0.0272	0.0208	0.0145
Top 8	0.0595	0.0402	0.0267	0.0142
No partial img	\	0.0473	0.0266	0.0142
Baseline	**0.0123**			

TABLE 4.1: Table of results of Experiment 1. Table values are MSE loss of predictions on the test set. The baseline model has the lowest loss: 0.0123. The single-modal learning models, i.e., column 2 and row 6, did not outperform the baseline model. However, as the second modality was combined with the single modality, the model's performance improved. When the top 23 rows of images were combined with images scaled by 1/2, the test loss was equal to the baseline.

effectiveness of adding the second modalities, we use Figure 4.2 and Figure 4.3 as 2 image views of Table 4.1.

The first view (Figure 4.2) shows the results of adding partial images to scaled images. Each line shows a decreasing trend when partial images with more and more top rows were added to scaled images. The second view (Figure 4.3) shows the results of adding scaled images to partial images. Each line shows a decreasing loss trend when each partial image was combined with a bigger and bigger down-scaled image. As shown in the bottom right of both figures, where all lines converge, when the partial images were combined with images scaled by 1/2, the losses were very close to the baseline.

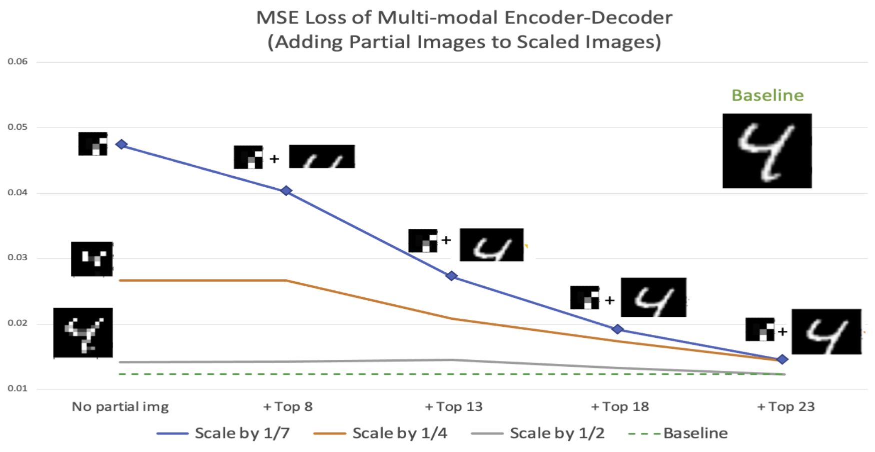


FIGURE 4.2: Loss values of modality combinations in MNIST data reconstruction (view 1). Each colored line represents a modality of smaller images scaled down by factors of 1/7, 1/4, or 1/2. Each column represents a modality of partial images, which are the top n rows of the original images. The first column is single-modal learning with only scaled images, and the baseline is the single-modal learning with original images (dotted green line).

The decreasing trend in both figures indicates that with more information added to the first modality, the encoder-decoder network could reconstruct the full images better. We assumed that the synthesized modalities could have compensated for

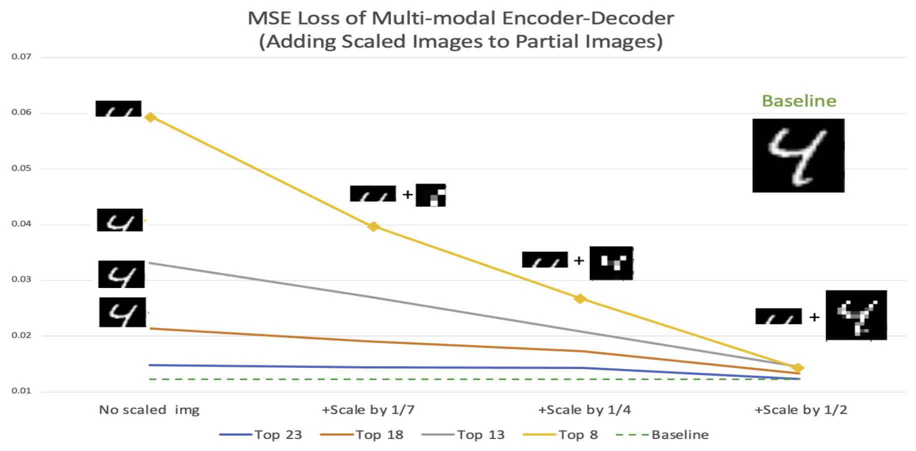


FIGURE 4.3: Loss values of modality combinations in MNIST data reconstruction (view 2). Each colored line represents a modality of partial images, which are the top n rows of the original images. Each column represents a modality of scaled images by factors of 1/7, 1/4, or 1/2. The first column is single-modal learning with only partial image, and the baseline is the single-modal learning with original images (dotted green line).

each other's missing information. To better understand the compensation, we drew some examples of the reconstructed images. Figure 4.4 shows the sample input and output of baseline and multi-modal learning with partial images combined with scaled images.

The images were sampled from the validation set. The output of the model using original images are not perfect, but the numbers are clear to human eyes. When the input is the top 8 rows, the reconstructions are a lot more blurry, and the third last image, number 9, was reconstructed as number 4. When the input is the top 8 rows and the scaled images by 1/4, the reconstructions are less blurry, and the third last number 9 was reconstructed as 9. Because the scaled images provided some useful information about the missing rows in the first modality, the model was more " certain " about the bottom rows to create more clear lines.

When the input is scaled images by 1/4, the output is close to the baseline when the digits are simple, such as the first 7 numbers. However, the last 3 outputs are wrong " guesses " of the original numbers where lines and circles are both present. It is worth noticing that the reconstructions of the third last image are both wrong numbers in single-modal learning but is correct in multi-modal learning. The information in the two modalities has compensated for each other when reconstructing full information.

However, the extra information from the second modality does not always help. For example, the last sample image, number 9, was reconstructed as a blurry 9 when the model only had the top 8 rows. However, it was reconstructed like a 7 when the model also considered the scaled images. The first and last scaled images look like 7 even though they are interpolations of original numbers 7 and 9. There is a limit to multi-modal learning when the two modalities cannot compensate well enough.

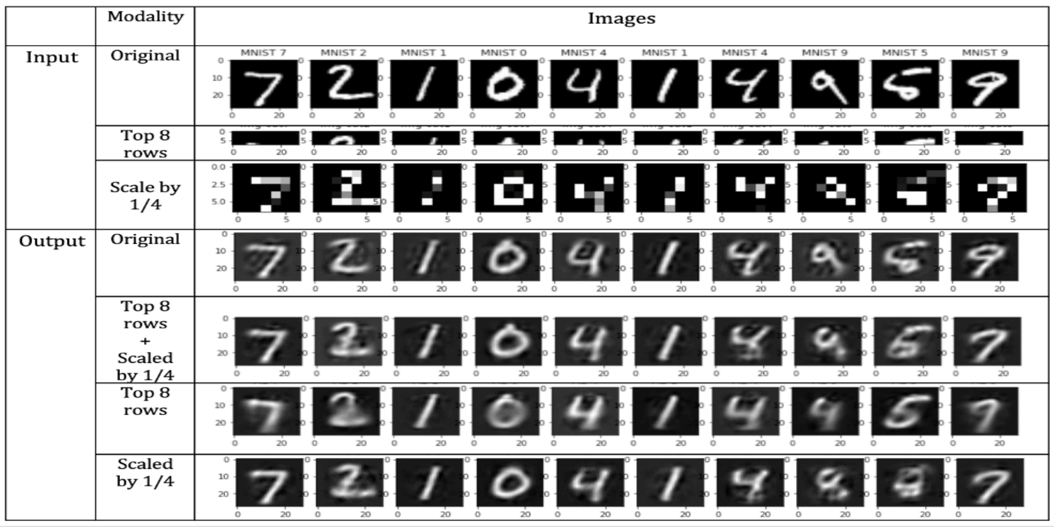

Modality
Images
Input
Original
Top 8 rows
Scale by 1/4
Output
Original
Top 8 rows + Scaled by 1/4
Top 8 rows
Scaled by 1/4
MNIST 7
MNIST 2
MNIST 1
MNIST 0
MNIST 4
MNIST 1
MNIST 4
MNIST 9
MNIST 5
MNIST 9

FIGURE 4.4: Sample input and output in MNIST data reconstruction. The baseline model used original images as input to output reconstructions of original images. In single-modal learning, only images with top 8 rows or images scaled by 1/4 were used as input, so the output was more blurry than baseline output. When we combined these two modalities, the output was less blurry because the modalities have compensating information. However, the compensation is still not enough for the multi-modal encoder-decoder to reconstruct images and baseline autoencoder.

4.2 Multi-modal Learning with Limited Data

Because the size of the dataset was small, a single validation split from the training dataset might not be a good representation of the data. So we bootstrapped 72 data samples with replacement 100 times, and we used the out-of-bag data as the validation dataset. Each model was trained from scratch for its bootstrap samples. We collected the best validation accuracy of each model in each bootstrap and made this plot (Figure 4.5) to see if multi-modal learning could help single-modal learning models improve. Sometimes the combined models were worse than the single models, and we assumed that this is because the learned representation of the second modality was not helpful in specific bootstraps.

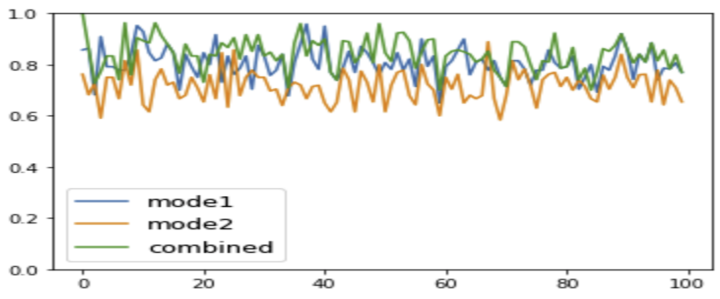


FIGURE 4.5: Bootstrap results of multi-modal and single-modal learning in double bonds prediction. Sometimes the combined models were worse than single models, but the mean accuracy of the combined model was better than either single model.

In a clearer view of how much multi-modal learning outperformed both single-modal learning models, we created a bar chart (Figure 4.6) to compare the mean accuracy among all bootstraps (Figure 4.5). In the single-modal learning with either mode, the accuracy was 71.55% with mode 2 and 80.49% with mode 1. The accuracy increased to 84.35% with both modes. We wanted to know if the performance improvement was not because of the change in architecture, so we used random noise as the second modality. When either of the two modes was random, multi-modal learning did not improve significantly compared to single-modal learning. The performance increased when mode2 was combined with the random noise as mode1. Based on the results, we recommend that in multi-modal learning, when the multi-modal architecture is different from single-modal architecture, it is worth checking how much the architecture change affected the performance. Especially when there are more parameters in multi-modal neural networks, we do not want to waste extra computation on other modalities that are not useful to improve the model.

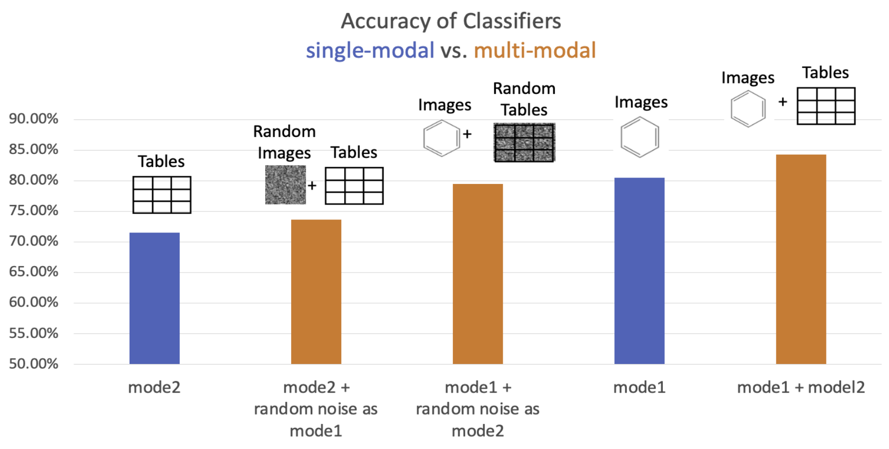


FIGURE 4.6: Results of multi-modal and single-modal learning in double bonds prediction. The accuracy was 84.35% when we combined mode 1 and mode 2. single-modal learning achieved an accuracy of 80.49% with mode 1, and 71.55% with mode 2. In the controlled Experiments, the accuracy was 79.49% when mode 1 was combined with random noise and 73.68% when mode 2 was combined with random noise. Multi-modal learning outperformed single-modal learning or controlled Experiments of multi-modal learning.

4.2.1 Representation Learning using Barlow Twins

Before using representations learned in Barlow Twins for the downstream multi-modal prediction task, we found that the cross-correlation matrices we got were not ideal because they were not close to the identity matrix of the same dimensions. For example, when the embedding size was 10, the beginning and best validation cross-correlation matrices and the identity matrix are shown in Figure 4.7. Ideally, the red color should be as dark as possible on the diagonal line, and the colors in the off-diagonal area should be as light as possible. Compared to the matrix at the beginning of the training, the best validation cross-correlation matrix showed lighter colors overall but did not show dark red on the diagonal line, meaning that the model failed to learn the correlation and did not reduce enough redundancy between embeddings. The worse result goes to larger embedding sizes, as shown in Figure 4.8.

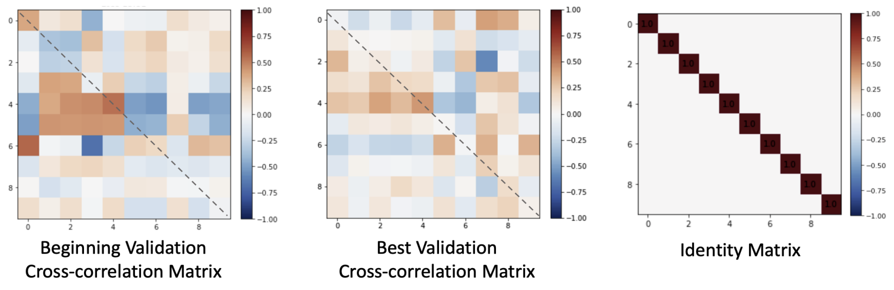


FIGURE 4.7: Barlow Twins cross-correlation matrix sample result 1. The cross-correlation matrices should be as close to the identity matrix as possible. Compared to the matrix in the beginning of the training, the best validation cross-correlation matrix shows lighter colors overall, but doesn't show dark red on the diagonal line, which means that the model failed to learn the correlation and didn't reduce enough redundancy between embeddings.

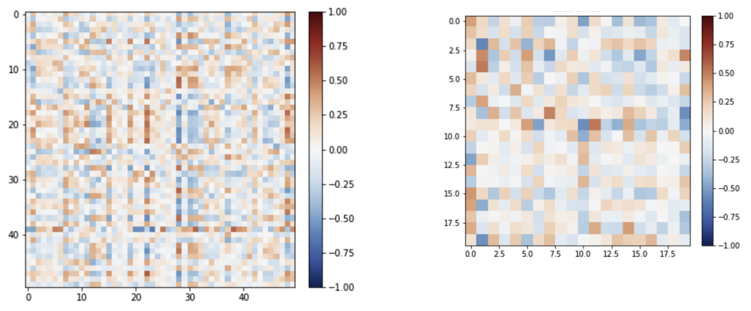


FIGURE 4.8: Barlow Twins cross-correlation matrix sample result 2. The embedding size is 50 on the left and 32 on the right. These two results are also not ideal as they are not close to their identity matrices.

Because the data size is too small, the learning process is not stable and as the embedding size increases, the model tends to overfit more.Because of the time limit and the scope of this research, we decided not to continue working on improving Barlow Twins.

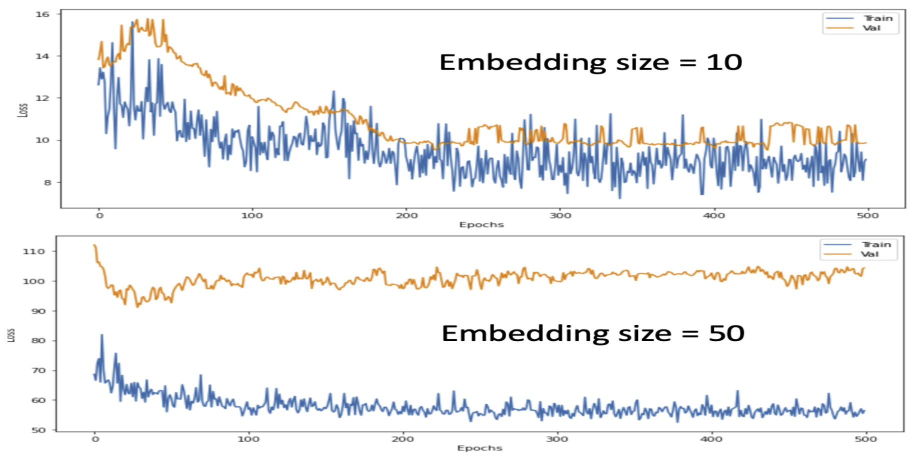


FIGURE 4.9: [Barlow Twins sample loss plots in training. The learning process is not stable and as the embedding size increases, the model tends to overfit more.

4.3 Multi-modal Learning with Public Datasets

In the beginning, contrary to the work of Goh et al. [18], our combined model did not perform better than the best single model (Fig 4.10). Unlike in Figure 3.11, the concatenation was connected to the prediction layer directly. After using the no-harm factor, we started to see improvement in multi-modal learning.

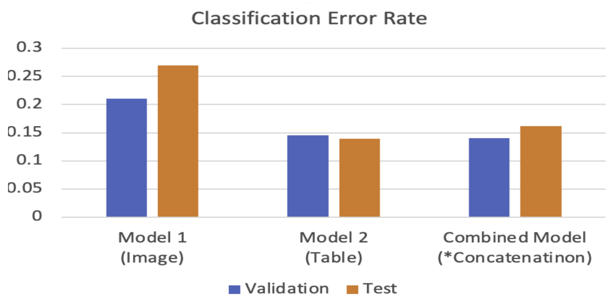


FIGURE 4.10: The results when combined model's MLP part was trained without the no-harm factor λ. The combination of modalities did not help the best single model improve and was worse than single-modal learning. Unlike in Figure 3.11, the concatenation was connected to the prediction layer directly.

Table 4.2 shows the results of multi-modal learning with the no-harm factor compared to single-modal learning. The effect of the no-harm factor varied based on the training methods, and the no-harm factor did not always prevent performance drop. Specifically, the model performed better on validation and testing sets when we froze both pretrained models or trained the best single model with the no-harm factor. The Concatenation fusion is the right figure in Figure 3.11, a fusion method used to compare with the no-harm factor because they have the same amount of layers. Model 1 and Model 2 are the two single-learning models trained with single modalities, and Model 2 is the baseline because it has better performance than Model 1. Red values are the test and validation metrics of the baseline single-modal model. Blue values are lower than red values, which means these models outperform the baseline model when trained via specific training methods and the no-harm factor.

Green values are the best results in test or validation time. This table shows that if the model is trained properly, multi-model learning with the no-harm factor can improve the performance of single-modal learning. With the no-harm scalar, the best test result is 8.2% better than the baseline when both pretrained models were frozen. Even though the no-harm factor cannot always guarantee better performance, with very few learnable parameters added to the model, it estimates how much an additional modality can improve model performance.

Learning Type	Single-modal		Multi-modal											
Model	Model 1 (Image)	Model 2 (Table)	Combined Model											
Trainining Method	End-to-End		Freeze Both	Freeze Model 2	Freeze Model 1	End-to-End	Freeze Both	Freeze Model 2	Freeze Model 1	End-to-End	Freeze Both	Freeze Model 2	Freeze Model 1	End-to-End
Fusion	\		λ scalar				λ vector				Concatenation			
Validation	0.2105	0.1453	0.143	0.1438	0.1415	0.1469	0.1467	0.1432	0.1415	0.1457	0.1454	0.1397	0.1356	0.1423
Test	0.2693	0.1398	0.1284	0.1455	0.1291	0.1355	0.1323	0.1432	0.1316	0.1409	0.1457	0.1383	0.1334	0.1337

TABLE 4.2: Results of multi-modal learning with no-harm factor. Green values are the test and validation metrics of the baseline model. Blue values are lower than green values, which means these models outperform the baseline single-modal model when trained via specific training methods and the no-harm factor.

To prove that the model can learn effective no-harm factor, we compared the model with a no-harm scalar λ and models with fixed no-harm scalar λ values: 0, 0.25, 0.5, 0.75, 1. In most training cases, the learnable no-harm scalar was better, despite a few cases. Values of fixed lambda marked in green are better than the learnable no-harm scalar. We think the reason is that the learned no-harm scalar values are mostly around 0.65, and models with λ in the range 0.5 to 0.75 tend to perform similarly. Overall, the learnable no-harm scalar λ is better, and it is worth the time saved from trying multiple fixed values.

Training	Freeze Both					
λ	**λ scalar**	λ = 0	λ = 0.25	λ = 0.5	λ = 0.75	λ = 1
Validation	0.143	0.2128	0.1589	0.144	0.148	0.1473
Test	0.1284	0.2643	0.1769	0.1368	0.1256	0.1295
Training	Freeze Model 2 & Train Model 1					
λ	**λ scalar**	λ = 0	λ = 0.25	λ = 0.5	λ = 0.75	λ = 1
Validation	0.1438	0.2104	0.1549	0.1438	0.1415	\
Test	0.1455	0.2672	0.1703	0.1412	0.1451	\
Training	Freeze Model 1 & Train Model 2					
λ	**λ scalar**	λ = 0	λ = 0.25	λ = 0.5	λ = 0.75	λ = 1
Validation	0.1415	\	0.1494	0.1436	0.1422	0.1423
Test	0.1291	\	0.1327	0.1291	0.1341	0.133
Training	End-to-End					
λ	**λ scalar**	λ = 0	λ = 0.25	λ = 0.5	λ = 0.75	λ = 1
Validation	0.1469	0.2162	0.1548	0.1476	0.1479	0.1532
Test	0.1355	0.2605	0.1552	0.148	0.1363	0.1412

TABLE 4.3: Results of fixed and learnable no-harm factors. When the no-harm factors (as a scalar) were fixed, the results were mostly worse than learnable no-harm factors(green). Only a few are better than the learnable no-harm factor. In general, the learnable no-harm scalar is better than the fixed ones.

Chapter 5

Conclusions and Discussions

5.1 Conclusions

We used three experiments to explore the effectiveness of multi-modal deep learning compared with single-modal learning. In our second experiment, the combination of Raman spectrum modality and molecule image modality in chemical analysis is unique in multi-modal deep learning. We have experimented and demonstrated several aspects of multi-modal learning compared to single-modal learning. First, multi-modal learning with incomplete information can achieve comparable performance as single-modal learning with complete information. Second, multi-modal learning can improve single-modal learning that cannot achieve ideal performance due to a lack of data. Third, multi-modal deep learning does not guarantee better performance than single-modal learning due to the complexity of training, model design, or lack of data. To mitigate this problem, we propose a no-harm factor to prevent performance drop after combining the second modality. In general, the multi-modal learning methods we used in these experiments are viable approaches in the physical science domain, such as chemistry, where the amount of labeled data is significantly smaller than those available in conventional deep learning research.

5.2 Discussion

First, when comparing the performances of multi-modal and single-modal learning, many variables were involved because of different model architectures and input types. It might be worth considering fixing the model design when comparing their performance. For example, by summing up two image inputs in the same shape from two modes, we fix the input shape of multi-modal and single-modal learning. We can train the same deep neural network to compare the results in this scenario.

Second, we noticed that when combining modalities from small datasets, the complexity of the modal induced by multiple modalities might dominate, weakening the model performance [31]. In this case, the no-harm factor is practical in designing multi-modal baseline architectures. While there are other ways to compensate for the lack of data, the no-harm factor is straightforward to implement. However, its simplicity limits the potential of the fusion layer. For now, between the embeddings of two modes, the interaction is not flexible as the ith neuron in embedding 1 can only be connected with the ith neuron in embedding 2. One direction is to explore no-harm options that encourage more intra-embedding or inter-embedding interactions. Intra-embedding interaction means that elements within an embedding can be connected to other elements in the same embedding. Inter-embedding interaction means that elements within an embedding can be connected to other elements in the same embedding, regardless of their indices.

Moreover, we can consider non-linear no-harm factor fusion such as:

$$c = e^{(1-\lambda)\ln(c_1)+\lambda\ln(c_2)}$$

Third, the results of the Barlow Twins were not ideal, and we did not focus on improving the model because of the scope of this research. One way to improve is to collect more unlabeled chemical data. Regarding self-supervised deep learning with multi-modal chemical data, several researchers have proposed similar solutions using other self-supervised methods such as SIMCLR [7]. However, at least for now, we have not found a published paper that experimented on Barlow Twins. One challenge of this task is that input molecules can have arbitrary sizes and components, highly variable connectivity, and many three-dimensional conformers [67]. It might be challenging to learn a large number of molecules in a fixed dimension. On the other hand, it can be much easier to collect one modality than the other. For example, collecting Raman spectra takes time and effort of domain experts, while molecular structures are highly available online. So large-scale self-supervised learning on both modalities can be difficult to apply to some modality combinations.

Last but not least, our fusion techniques are not the most advanced ones because when we were mostly dealing with small data sizes, we started to focus on the no-harm factor to ensure that the complexity added to the model would not aggravate the overfitting problem. We have explored a fusion technique called Tensor Fusion [72], which reserves single-modal features and multi-modal interactions in the combined embedding. However, the result was not better when we used the same hyper-parameters and appeared to overfit (Figure 5.1). One thing we like about no-harm factor is that it doesn't require extra hyper-parameter tuning. In future studies, if our goal is to improve a model as much as possible, especially in benchmarking, more advanced fusion techniques are needed.

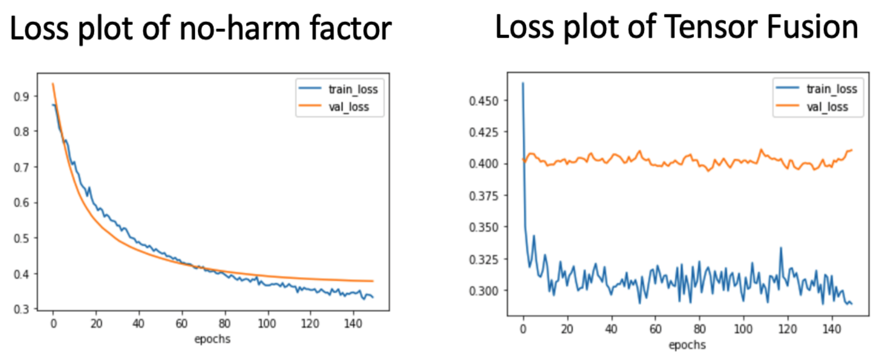


FIGURE 5.1: When sharing the same hyper-parameters with no-harm factor and trained and validated on the same datasets, the tensor fusion model appears to overfit a lot.

Appendix A

More Details in 3 Experiments

A.1 Experiment 1

A.1.1 Encoder-Decoder architecture

The encoder-Decoder was made of 7 fully connected layers including the input layer, each followed by a ReLU activation function. The layer sizes were [input_size, 250, 128, 81, 128, 250, 784], and the input_size was the sum of sizes of flattened modalities.

```
EncoderDecoder(
  (linear1): Linear(input_size, 250)
  (relu): ReLU()
  (linear2): Linear(250, 128)
  (relu): ReLU()
  (linear3): Linear(128, 81)
  (relu): ReLU()
  (linear4): Linear(81, 128)
  (relu): ReLU()
  (linear5): Linear(128, 250)
  (relu): ReLU()
  (linear6): Linear(250, 784)
  (relu): ReLU()
)
```

A.2 Experiment 2

A.2.1 Image Modality - CNN

```
CNN(
     (conv_1): Conv2d(1, 16, kernel_size=(5,5), padding=2),
     (bn_1): BatchNorm2d(16),
     (maxpool_1): MaxPool2d(kernel_size=(2,2)),
     (conv_2): Conv2d(16, 32, kernel_size=(5,5), stride=1, padding=2)
     (bn_2): BatchNorm2d(32)
     (maxpool_2): MaxPool2d(kernel_size=(2,2))
     ( flatten ): Flatten ()
     (linear_1): Linear(32 * 20 * 20, 126)
     (bn_3): BatchNorm1d(126)
     (linear_2): Linear(126, 3)
)
```

A.2.2 Architecture for Raman Table Modality - MLP

The architecture of model for tabular data was made of 4 fully connected feedforward layers in sizes: [325, 160, 80, 20], and the output size is 3 as there were 3 classes to predict. Each of first 3 layers were followed by sigmoid activation function.

```
MLP(
  (linear1): Linear(325, 160)
  (sigmoid): Sigmoid()
  (linear2): Linear(160, 80)
  (sigmoid): Sigmoid()
  (linear3): Linear(80, 20)
  (sigmoid): Sigmoid()
  (linear4): Linear(20, 3)
)
```

A.3 Experiment 3

A.3.1 Architecture for Image Modality - Chemception

Customized to chemical properties, Chemception [16] is based on Google's Inception-ResNet. Inception [60] networks provide multiple kernel sizes in each convolution step while avoiding the extra expense by adding 1x1 convolution layers before each kernel. When deeper networks are able to start converging, with the network depth increasing, accuracy gets saturated and then degrades rapidly. ResNet [26] networks addresses this degradation problems in very deep networks.

On top of the combination of these to networks, the chemical domain knowledge of Goh et al. led to the creation of Chemception that is well suited to use chemical properties in domain tasks.

The Chemception model is consist of a Stem section and 4 Inception-Resnet blocks: InceptionResnetA, ReductionA, InceptionResnetB, ReductionB, Inception-ResnetC. Each block either reduces the size of images(shape of 2D feature maps) while increasing the number of channels(number of feature maps), or keeps the input shape. Through all blocks and individual layers, the shape(batch_size, channel, width, height) of a batch of images changes in this order: (batch_size, 4, 80, 80) →Stem →(batch_size, 16, 39, 39) →InceptionResnetA →(batch_size, 16, 39, 39) →ReductionA →(batch_size, 64, 19, 19) →InceptionResnetB →(batch_size, 64, 19, 19) →ReductionB →(batch_size, 126, 9, 9) →InceptionResnetC →(batch_size, 126, 9, 9) →AvgPool2d →(batch_size, 126, 1, 1) →reshape →(batch_size, 126) →linear layer →(batch_size, 1). Below is the high level architecture of Chemception, and we will explain each block in detail afterward.

```
Chemception(
    (stem): Stem(self.base_filters, self.input_channels)
    (inception_blocks): (
        InceptionResnetA(num_filters=16, input_channels=16),
        ReductionA(num_filters=16, input_channels=16),
        InceptionResnetB(num_filters=16, input_channels=64),
        ReductionB(num_filters=16 input_channels=64),
        InceptionResnetC(num_filters=16, input_channels=126)
```

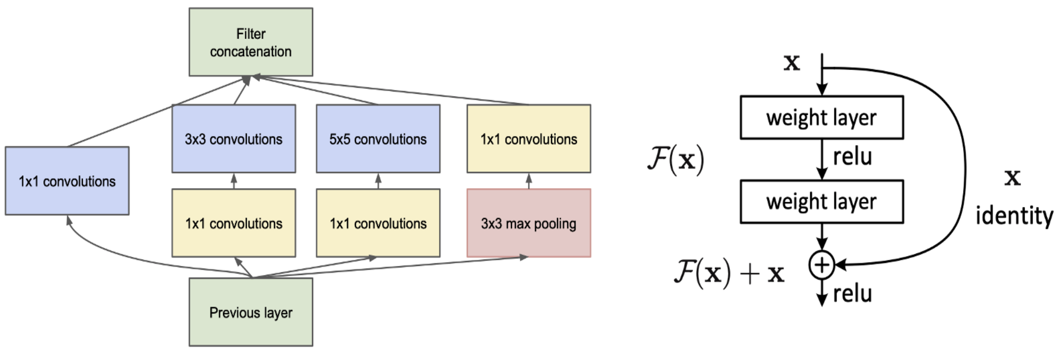


FIGURE A.1: Figure 2 in [60] and Figure 2 in [26]. Building blocks of inception module(left) and residual Learning(right). Inception networks provide multiple kernel sizes in each convolution step while avoiding the extra expense by adding 1x1 convolution layers before each kernel. ResNet networks are used to ease the training of neural networks that are very deep.

```
    )
    (avg_pooling): AvgPool2d(kernel_size=(9, 9), stride=(1, 1))
    (last_linear): Linear(126, n_classes)
```

Stem block used 16 4x4 kernels with stride 2x2 to reduce the size of the image from 80x80 to 39x39, while increasing the channels from 3 to 16. The convolution layer is followed by a ReLU activation. The output of Stem is passed to InceptionResnetA.

```
Stem(
   Conv2d(4, 16, kernel_size=(4, 4), stride=(2,2))
   ReLU()
)
```

InceptionResnetA uses 3 blocks of different sizes of kernels – conv_block1, conv_block2, conv_block3 – to transform the input. The feature maps after these pooling layers are concatenates over the channel dimension, leading to 16 channels of 39x39 feature maps, which are followed by a convolution layer in conv_concat_block. All convolution layers of InceptionResnetA are followed by a ReLU activation. The output of InceptionResnetA is passed to ReductionA block.

```
InceptionResnetA(
   (conv_block1): (
        Conv2d(16, 16, kernel_size=(1, 1), stride=(1,1), padding="same")
        ReLU()
    )
    (conv_block2): (
        Conv2d(16, 16, kernel_size=(1, 1), stride=(1,1), padding="same")
        ReLU()
        Conv2d(16, 16, kernel_size=(3, 3), stride=(1,1), padding="same")
        ReLU()
    )
    (conv_block3): (
        Conv2d(16, 16, kernel_size=(1, 1), stride=(1,1), padding="same")
        ReLU()
        Conv2d(16, 24, kernel_size=(3, 3), stride =(1,1), padding="same")
        ReLU()
        Conv2d(16, 32, kernel_size=(4, 4), stride =(1,1), padding="same")
        ReLU()
    )
    (conv_concat_block): (
        Conv2d(64, 16, kernel_size=(1, 1), stride =(1,1), padding="same")
    )
```

ReductionA has a self-explanatory name because it reduces the size of the feature maps from 39x39 to 19x19, while increasing the channels from 16 to 64. It reduces the size of the feature maps in 2 ways: max pooling(pool_block1) and convolution(conv_block1, conv_block2). Like InceptionResnetA, ReductionA uses different sizes of kernels in sequentially in conv_block2. All pooling and convolution layers are followed by ReLU activation. The outputs of pool_block1, conv_block1, conv_block2 are concatenated over channel dimension so we get more channels. The output of the concatenation will be sent to InceptionResnetB.

```
ReductionA(
    (pool_block1): (
        MaxPool2d(kernel_size=(3, 3), stride=2)
        ReLU()
    )
    (conv_block1): (
        Conv2d(16, 24, kernel_size=(3, 3), stride =(2,2), padding="valid")
        ReLU()
    )
    (conv_block2): (
        Conv2d(24, 16, kernel_size=(1, 1), stride =(1,1), padding="same")
        ReLU()
        Conv2d(16, 16, kernel_size=(3, 3), stride =(1,1), padding="same")
        ReLU()
        Conv2d(16, 24, kernel_size=(3, 3), stride =(2,2), padding="valid")
        ReLU()
    )
```

InceptionResnetB uses 2 blocks – conv_block1, conv_block2 – to transform the output from ReductionA. In conv_concat_block, the feature maps after these pooling layers are concatenates over the channel dimension, which are followed by a convolution layer. All convolution layers of InceptionResnetB are followed by a ReLU activation. The output of InceptionResnetB is passed to ReductionB block.

```
InceptionResnetB(
    (conv_block1): (
        Conv2d(64, 16, kernel_size=(1, 1), stride =(1,1), padding="same")
    )
    (conv_block2): (
        Conv2d(64, 16, kernel_size=(1, 1), stride =(1,1), padding="same")
        ReLU()
        Conv2d(16, 20, kernel_size=(1, 7), stride =(1,1), padding="same")
        ReLU()
        Conv2d(20, 24, kernel_size=(7, 1), stride =(1,1), padding="same")
        ReLU()
    )
    (conv_concat_block): (
        Conv2d(40, 64, kernel_size=(1, 1), stride =(1,1), padding="same")
    )
```

ReductionB is like ReductionA. It reduces the size of the feature maps from 19x19 to 9x9, while increasing the channels from 64 to 126. It reduces the size of the feature maps in 2 ways: max pooling(pool_block1) and convolution(conv_block1, conv_block2, conv_block3). All pooling and convolution layers are followed by ReLU activation. The outputs of pool_block1, conv_block1, conv_block2, conv_block3 are concatenated over channel dimension so we get more channels. The output of the concatenation will be sent to InceptionResnetC.

```
ReductionB(
   (pool_block1): (
        MaxPool2d(kernel_size=(3, 3), stride=2)
    )
    (conv_block1): (
        Conv2d(64, 16, kernel_size=(1, 1), stride=1, padding="same")
        ReLU()
        Conv2d(16, 24, kernel_size=(3, 3), stride =(2,2), padding="same")
        ReLU()
    )
    (conv_block2): (
        Conv2d(64, 16, kernel_size=(1, 1), stride =(1,1), padding="same")
        ReLU()
        Conv2d(16, 18, kernel_size=(3, 3), stride =(2,2), padding="valid")
        ReLU()
    )

    (conv_block3): (
        Conv2d(64, 16, kernel_size=(1, 1), stride =(1,1), padding="same")
        ReLU()
        Conv2d(16, 18, kernel_size=(3, 1), stride =(1,1), padding="same")
        ReLU()
        Conv2d(18, 20, kernel_size=(3, 3), stride =(2,2), padding="valid")
        ReLU()
    )
```

InceptionResnetC has the same block structure as InceptionResnetB, except the kernel sizes and numbers customized to the output of ReductionB. In conv_concat_block, the feature maps after these pooling layers are concatenates over the channel dimension, which are followed by a convolution layer. All convolution layers of Inception-ResnetC are followed by a ReLU activation. The output of InceptionResnetB is passed to a global average pooling layer, which takes the mean value of each (9x9) feature map.

```
InceptionResnetC(
(conv_block1): (
      Conv2d(126, 16, kernel_size=(1, 1), stride =(1,1), padding="same")
      ReLU()
 )
 (conv_block2): (
      Conv2d(126, 16, kernel_size=(1, 1), stride =(1,1), padding="same")
      ReLU()
      Conv2d(16, 19, kernel_size=(1, 3), stride =(1,1), padding="same")
      ReLU()
      Conv2d(19, 21, kernel_size=(3, 1), stride =(1,1), padding="same")
      ReLU()
 )
 (conv_concat_block): (
      Conv2d(37, 126, kernel_size=(1, 1), stride =(1,1), padding="same")
 )
```

Bibliography

[1] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. "Understanding of a convolutional neural network". In: *2017 International Conference on Engineering and Technology (ICET)*. Ieee. 2017, pp. 1–6.

[2] Pradeep K Atrey et al. "Multimodal fusion for multimedia analysis: a survey". In: *Multimedia systems* 16.6 (2010), pp. 345–379.

[3] Tadas Baltrušaitis, Chaitanya Ahuja, and Louis-Philippe Morency. "Multimodal machine learning: A survey and taxonomy". In: *IEEE transactions on pattern analysis and machine intelligence* 41.2 (2018), pp. 423–443.

[4] Khaled Bayoudh et al. "A survey on deep multimodal learning for computer vision: advances, trends, applications, and datasets". In: *The Visual Computer* (2021), pp. 1–32.

[5] Samy Bengio et al. "Confidence measures for multimodal identity verification". In: *Information Fusion* 3.4 (2002), pp. 267–276.

[6] Serhat S Bucak, Rong Jin, and Anil K Jain. "Multiple kernel learning for visual object recognition: A review". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.7 (2013), pp. 1354–1369.

[7] Ting Chen et al. "A simple framework for contrastive learning of visual representations". In: *International conference on machine learning*. PMLR. 2020, pp. 1597–1607.

[8] Yulu Luke Chen et al. "Inexpensive multimodal sensor fusion system for autonomous data acquisition of road surface conditions". In: *IEEE Sensors Journal* 16.21 (2016), pp. 7731–7743.

[9] Bowen Cheng et al. "Multi-task Learning and Multimodal Fusion for Road Segmentation". In: *IEEE Access* (2022).

[10] Yanhua Cheng et al. "Locality-sensitive deconvolution networks with gated fusion for rgb-d indoor semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 3029–3037.

[11] RTH Collis. "Lidar". In: *Applied optics* 9.8 (1970), pp. 1782–1788.

[12] Yifan Deng et al. "A multimodal deep learning framework for predicting drug–drug interaction events". In: *Bioinformatics* 36.15 (2020), pp. 4316–4322.

[13] Andreas Eitel et al. "Multimodal deep learning for robust RGB-D object recognition". In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2015, pp. 681–687.

[14] Jing Gao et al. "A survey on deep learning for multimodal data fusion". In: *Neural Computation* 32.5 (2020), pp. 829–864.

[15] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. "Deep sparse rectifier neural networks". In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2011, pp. 315–323.

[16] Garrett B Goh et al. "Chemception: a deep neural network with minimal chemistry knowledge matches the performance of expert-developed QSAR/QSPR models". In: *arXiv preprint arXiv:1706.06689* (2017).

[17] Garrett B Goh et al. "How much chemistry does a deep neural network need to know to make accurate predictions?" In: *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE. 2018, pp. 1340–1349.

[18] Garrett B Goh et al. "Multimodal deep neural networks using both engineered and learned representations for biodegradability prediction". In: *arXiv preprint arXiv:1808.04456* (2018).

[19] Garrett B Goh et al. "Using rule-based labels for weak supervised learning: a ChemNet for transferable chemical property prediction". In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2018, pp. 302–310.

[20] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[21] PRGDJ Graves and D Gardiner. "Practical raman spectroscopy". In: *Springer* (1989).

[22] Wenzhong Guo, Jianwen Wang, and Shiping Wang. "Deep multimodal representation learning: A survey". In: *IEEE Access* 7 (2019), pp. 63373–63394.

[23] Mihai Gurban et al. "Dynamic modality weighting for multi-stream hmms inaudio-visual speech recognition". In: *Proceedings of the 10th international conference on Multimodal interfaces*. 2008, pp. 237–240.

[24] David L Hall and James Llinas. "An introduction to multisensor data fusion". In: *Proceedings of the IEEE* 85.1 (1997), pp. 6–23.

[25] David R Hardoon, Sandor Szedmak, and John Shawe-Taylor. "Canonical correlation analysis: An overview with application to learning methods". In: *Neural computation* 16.12 (2004), pp. 2639–2664.

[26] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[27] Kaiming He et al. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1026–1034.

[28] Seyed Aghil Hooshmand et al. "A multimodal deep learning-based drug repurposing approach for treatment of COVID-19". In: *Molecular diversity* (2020), pp. 1–14.

[29] Shih-Cheng Huang et al. "Fusion of medical imaging and electronic health records using deep learning: a systematic review and implementation guidelines". In: *NPJ digital medicine* 3.1 (2020), pp. 1–9.

[30] Xun Huang et al. "Multimodal unsupervised image-to-image translation". In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 172–189.

[31] Yu Huang et al. "What Makes Multimodal Learning Better than Single (Provably)". In: *arXiv preprint arXiv:2106.04538* (2021).

[32] Xu Jia et al. "Guiding the long-short term memory model for image caption generation". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 2407–2415.

[33] Xingyu Jiang et al. "A review of multimodal image matching: Methods and applications". In: *Information Fusion* 73 (2021), pp. 22–71.

[34] Gargi Joshi, Rahee Walambe, and Ketan Kotecha. "A review on explainability in multimodal deep neural nets". In: *IEEE Access* (2021).

[35] Ikhyun Kang et al. "Fusion drive: End-to-end multi modal sensor fusion for guided low-cost autonomous vehicle". In: *2020 17th International Conference on Ubiquitous Robots (UR)*. IEEE. 2020, pp. 421–428.

[36] Abdul Karim et al. "Toxicity prediction by multimodal deep learning". In: *Pacific Rim Knowledge Acquisition Workshop*. Springer. 2019, pp. 142–152.

[37] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *nature* 521.7553 (2015), pp. 436–444.

[38] Kuan Liu et al. "Learn to combine modalities in multimodal deep learning". In: *arXiv preprint arXiv:1805.11730* (2018).

[39] Lin Ma et al. "Multimodal convolutional neural networks for matching image and sentence". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 2623–2631.

[40] Sijie Mai, Haifeng Hu, and Songlong Xing. "A unimodal representation learning and recurrent decomposition fusion structure for utterance-level multimodal embedding learning". In: *IEEE Transactions on Multimedia* (2021).

[41] Kamel Mansouri et al. "Quantitative structure–activity relationship models for ready biodegradability of chemicals". In: *Journal of chemical information and modeling* 53.4 (2013), pp. 867–878.

[42] Wesley Mattheyses and Werner Verhelst. "Audiovisual speech synthesis: An overview of the state-of-the-art". In: *Speech Communication* 66 (2015), pp. 182–217.

[43] Harry McGurk and John MacDonald. "Hearing lips and seeing voices". In: *Nature* 264.5588 (1976), pp. 746–748.

[44] Jiquan Ngiam et al. "Multimodal deep learning". In: *ICML*. 2011.

[45] Andrew Owens and Alexei A Efros. "Audio-visual scene analysis with self-supervised multisensory features". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 631–648.

[46] Norman Poh and Samy Bengio. "How do correlation and variance of base-experts affect fusion in biometric authentication tasks?" In: *IEEE Transactions on Signal Processing* 53.11 (2005), pp. 4384–4396.

[47] *QSAR biodegradation Data Set*. URL: https://archive.ics.uci.edu/ml/datasets/QSAR+biodegradation.

[48] Valentin Radu et al. "Multimodal deep learning for activity and context recognition". In: *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 1.4 (2018), pp. 1–27.

[49] Anil Rahate et al. "Multimodal Co-learning: Challenges, Applications with Datasets, Recent Advances and Future Directions". In: *arXiv:2107.13782* (2021).

[50] Dhanesh Ramachandram and Graham W Taylor. "Deep multimodal learning: A survey on recent advances and trends". In: *IEEE signal processing magazine* 34.6 (2017), pp. 96–108.

[51] *Raman Spectrum*. URL: https://wiki.anton-paar.com/en/basics-of-raman-spectroscopy/raman-spectroscopy-applications/.

[52] Morteza Rohanian, Julian Hough, Matthew Purver, et al. "Detecting Depression with Word-Level Multimodal Fusion." In: *INTERSPEECH*. 2019, pp. 1443–1447.

[53] Nusrat J Shoumy et al. "Multimodal big data affective analytics: A comprehensive survey using text, audio, visual and physiological signals". In: *Journal of Network and Computer Applications* 149 (2020), p. 102447.

[54] Federico Simonetta, Stavros Ntalampiras, and Federico Avanzini. "Multimodal music information processing and retrieval: Survey and future challenges". In: *2019 International Workshop on Multilayer Music Representation and Processing (MMRP)*. IEEE. 2019, pp. 10–18.

[55] Cees GM Snoek, Marcel Worring, and Arnold WM Smeulders. "Early versus late fusion in semantic video analysis". In: *Proceedings of the 13th annual ACM international conference on Multimedia*. 2005, pp. 399–402.

[56] Sijie Song et al. "Modality compensation network: Cross-modal adaptation for action recognition". In: *IEEE Transactions on Image Processing* 29 (2020), pp. 3957–3969.

[57] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. "Highway networks". In: *arXiv preprint arXiv:1505.00387* (2015).

[58] Eric Stauffer, Julia A. Dolan, and Reta Newman. "CHAPTER 3 - Review of Basic Organic Chemistry". In: *Fire Debris Analysis*. Ed. by Eric Stauffer, Julia A. Dolan, and Reta Newman. Burlington: Academic Press, 2008, pp. 49–83. ISBN: 978-0-12-663971-1. DOI: https://doi.org/10.1016/B978-012663971-1.50007-5. URL: https://www.sciencedirect.com/science/article/pii/B9780126639711500075.

[59] Jabeen Summaira et al. "Recent Advances and Trends in Multimodal Deep Learning: A Review". In: *arXiv preprint arXiv:2105.11087* (2021).

[60] Christian Szegedy et al. "Going deeper with convolutions". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9.

[61] Valentin Vielzeuf et al. "Centralnet: a multilayer approach for multimodal fusion". In: *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*. 2018.

[62] Aarohi Vora, Chirag N Paunwala, and Mita Paunwala. "Improved weight assignment approach for multimodal fusion". In: *2014 International Conference on Circuits, Systems, Communication and Information Technology Applications (CSCITA)*. IEEE. 2014, pp. 70–74.

[63] Jinghua Wang et al. "Learning common and specific features for RGB-D semantic segmentation with deconvolutional networks". In: *European Conference on Computer Vision*. Springer. 2016, pp. 664–679.

[64] Yikai Wang et al. "Deep multimodal fusion by channel exchanging". In: *Advances in Neural Information Processing Systems* 33 (2020).

[65] Yunchao Wei et al. "Cross-modal retrieval with CNN visual features: A new baseline". In: *IEEE transactions on cybernetics* 47.2 (2016), pp. 449–460.

[66] David Weininger. "SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules". In: *Journal of chemical information and computer sciences* 28.1 (1988), pp. 31–36.

[67] Zhenqin Wu et al. "MoleculeNet: a benchmark for molecular machine learning". In: *Chemical science* 9.2 (2018), pp. 513–530.

[68] Ran Xu et al. "Jointly modeling deep video and compositional text to bridge vision and language in a unified framework". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 29. 1. 2015.

[69] Yan Yang and Hao Wang. "Multi-view clustering: A survey". In: *Big Data Mining and Analytics* 1.2 (2018), pp. 83–107.

[70] Christopher J.C. Burges Yann LeCun Corinna Cortes. *THE MNIST DATABASE of handwritten digits*. http://yann.lecun.com/exdb/mnist/.

[71] Ben P Yuhas, Moise H Goldstein, and Terrence J Sejnowski. "Integration of acoustic and visual speech signals using neural networks". In: *IEEE Communications Magazine* 27.11 (1989), pp. 65–71.

[72] Amir Zadeh et al. "Tensor fusion network for multimodal sentiment analysis". In: *arXiv preprint arXiv:1707.07250* (2017).

[73] Jure Zbontar et al. "Barlow twins: Self-supervised learning via redundancy reduction". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 12310–12320.

[74] Chao Zhang et al. "Multimodal intelligence: Representation learning, information fusion, and applications". In: *IEEE Journal of Selected Topics in Signal Processing* 14.3 (2020), pp. 478–493.

[75] Yifei Zhang et al. "Deep multimodal fusion for semantic image segmentation: A survey". In: *Image and Vision Computing* 105 (2021), p. 104042.