Vincent Wahid Ajoubi (student ID 17054)

IT-UNIVERSITETET I KØBENHAVN

DATA SCIENCE

EXPLORING MACHINE LEARNING METHODS FOR CHARACTERIZING MUSIC CONTENT

VINCENT WAHID AJOUBI (VIAJ@ITU.DK)
PROJECT SUPERVISOR: SAMI BRANDT

APRIL 2, 2024

IT-University of Copenhagen
April 2, 2024

Abstract

With the exponential growth of audio content, the demand for efficient and accurate audio content classification methods has become increasingly crucial. In this thesis, we conduct a comprehensive review of the latest literature, focusing on their application to audio content classification tasks. Our primary objective is to identify and understand the newest and most effective approaches that can be applied to the classification of audio content. We explore various machine/deep learning techniques. Through review of recent research papers, we analyze the methodologies, architectures, and experimental results of these strategies, aiming to understand their strengths, weaknesses, and potential applications in audio classification. Furthermore, we investigate the challenges and opportunities presented by utilizing these techniques in the context of audio content classification tasks. We aim to provide a comprehensive understanding of the state-of-the-art strategies for audio content classification, ultimately informing the development of more efficient and accurate classification systems in various application such as music instrument classification.

Contents

List of Tables

1 | Introduction

Musical, an art form as ancient as human civilization itself, has held a profound influence on societies across the globe. From the rhythmic beats of tribal drums echoing through prehistoric gatherings to the complex symphonies performed in grand concert halls, music has been a constant companion, shaping cultures, emotions, and identities. Its study is not merely an examination of melodies and harmonies but a journey into the depths of human expression, psychology, and culture.

Music has been extensively explored, delving into its diverse dimensions, methodologies, and applications. From Automatic music generation[1] to Automatic lyric transcription[2]. From Music Genre Classification[3] to Music instrument classification[4].

Central to these vast exploration of music, is the research of state-of-the-art to revolutionize the production, distribution and consumption of music on a global scale. Through the examination of latest works in this domain, this thesis seeks to unravel the complexities of novel approaches.

Furthermore the motivation of this thesis is a testament to the transformative potential of Machine/Deep learning methods in the context of music analysis, offering a comprehensive overview of its methodological approaches, and its practical applications.

Structure

We start with *Related Works* in chapter 2 reviewing literatures related to the problems, and how researchers expanded their knowledge to identify and evaluate the new methods with respect to past approaches to the problem. We will be reviewing the

successes and limitations of their proposed solution which provides important understanding related to state-of-the-art for general classification algorithms, and how they take advantage of previous successes, and most importantly, potentially improving the solution or the technique in general, when applied to the field of music classification.

In chapter 3 *Background* we provide more details on the current methods that was previously identified by researchers. The main intention for this chapter is to demonstrate and define the key elements of the methods which researchers claim the state-of-the-art. More specific about what and how elements are structured.

In chapter 4 *Method* we will be explaining on how we approach the identified methods and how we utilize the suggested algorithms and features to tackle the issue. Overall this chapter represents our chosen methods that we explored in previous chapters. Following, in chapter 5 *Experiment*, we report the experimented methods and the result of our procedure and findings.

In chapter 6, *Analysis*, we provide an in-depth examination of the performance of the models, their efficacy in audio classification tasks. We will be discussing the benefits and limitations of our work and the modeling we faced during this thesis. and finally we conclude this long journey in *Conclusion* section.

2 | Related works

This chapter will review the literatures introducing different Machine/Deep learning techniques that can be employed to classify and categorize music contents into genres, styles and instruments. In essence, this chapter will be a review of works related to the problem that this thesis aims to go through, and how methods from different branches find themselves in the field of computer vision, more specific in the field of audio content classification. We will be identifying and elaborating chronologically on the past/present methods to approach to the problem and how different state-of-the-art claiming models surpasses previous state-of-the-art.

The field of audio analysis within the domain of machine learning and deep learning is experiencing rapid growth. This growth is evidenced by the increasing number of research publications, workshops, conferences and even competitions dedicated to the intersection of music and machine/deep learning, reflecting a promising interest in leveraging computational approaches to unravel the complexities of music content. Moreover, collaborations between researchers, musicians, and industry professionals are fostering interdisciplinary innovation, driving forward the boundaries of what is achievable in understanding and harnessing the richness of musical expression through computational means.

ILSVRC stands for ImageNet Large Scale Visual Recognition Challenge has held its place in this rapid growth. The challenge was an annual competition between 2010-2017. The competition served as a benchmark for evaluating the performance of algorithms and models in tasks such as object detection, localization and classification. The competition played a crucial role in driving advancements in deep learning specifically *Convolution Neural Network* [CNN] from Computer Vision side. One of these architectures known as VGG-16[5] which achieved state-of-the-art and became the top performing model in the ImageNet challenge in 2014. Their architecture design was an extension of the original design for image classification by Krizhevsky et al.[6], Deep ConvNet. Although it has been surpassed by newer architectures in terms of state-of-the-art, however it serves as a baseline for benchmarking and comparison in the field of computer vision. Additionally, it becomes a popular choice for educational purposes and as a starting point for developing custom CNN architectures.

While VGG16 itself is primarily used in the domain of computer vision, its influence extended to the domain of music and audio analysis through the principles and techniques it popularized in deep learning. Some instance examples among variety of applications in audio classification such as music genre classification[7], music recommendation system[8], speech recognition[9], the applications go on.

But the field of audio classification research has moved far from models based on

hand-crafted features[10] to end-to-end models that directly map audio spectrograms to corresponding labels[11, 12]. Specifically, CNN [12, 13] which aside VGG-16 model have been widely used to learn representations from raw spectrograms for end-to-end modeling, as the inductive biases inherent to CNNs such as spatial locality and translation equivariance are believed to be helpful. In order to better capture long-range global context, a recent trend is to add a self-attention mechanism on top of the CNN. Such CNN-attention hybrid models have achieved state-of-the-art results for many audio classification tasks such as audio event classification [14, 15], speech command recognition[16], and emotion recognition [17]. However, motivated by the success of purely attention-based models in the vision domain [18, 19, 20], it is reasonable to ask whether a CNN is still essential for audio classification.

In 2017 Vaswani et al. proposes [21] "Transformers" for machine translation, and have since become the state of the art method in many Natural Language Processing (NLP) tasks. The dominant approach is to pre-train on a large text corpus and then fine-tune on a task-specific datasets. However the Transformers was limited to text-based applications. On the other hand in computer vision CNNs architectures remains dominant and it was the primary approach. Inspired by NLP successes, multiple works such as [22] carion et al. tries combining CNN-like architectures with self-attention Transformers for object detection. Later in the same year in 2020 [18] adosovitskiy et al. comes with the idea of directly using standard Transformers which was introduced by [21] for images, known as *ViT*, with the fewest possible modifications. To do so, they split an image into patches and provide the sequence of linear embeddings of these patches as an input to a Transformer. Image patches are treated the same way as tokens (words) in a NLP application task. They train the model on image classification in supervised fashion.

Having the new trend, Transformers, in mind, in 2018 devlin et al.[23] introduces *BERT* which the architecture model is based on Transformers proposes the idea of Masked Language Model (MLM). The goal is that the input sentences are masked by randomly replacing some words with a special [MASK] token, and the model is

then trained to predict the masked words based on the surrounding context. These methods enable training of generalizable NLP models containing over one hundred billion parameter. They train the model on a large text corpus in a self-supervised fashion. The pre-training process involves exposing the model to a vast amount of text data, and the model learns to generate contextualized representations of words. After pre-training, the BERT model can be fine-tuned on downstream tasks with labeled data, such as text classification.

Inspired by BERT[23], Facebook AI Research (FAIR) publishes in 2021, [24] "Masked Autoencoders Are Scalable Vision Learners". The paper presents an innovative architecture called Masked Autoencoders (MAEs) designed for self-supervised training of models. The primary focus of the training methodology is on architectures resembling vision transformers (ViT)[18]. MAE randomly drop out 75% of the image, and feed the image to an encoder which is a standard Transformer introduced by ViT[18], and recreate the image by the decoder. They claim their approach makes pre-training time 3X faster with better accuracy for various downstream task such as classification.

This paper has become the inspiration for various classification task which is based on Computer Vision, such as [25, 26, 27] as well as [28]. These very recent papers extend MAE with audio-visual multi-modality, and [28] only audio, claiming the state-of-the-art on either audio or video downstream classification task.

All these similar extensions inspired by MAE[24] and ViT[18] with some degree differences follows one same approach. Following ViT[18] and MAE[24] they model a Transformers based AutoEncoder to re-generate an image which was dropped by 75%, then attaching the pretrained encoder to a linear classification head for classification purpose. This has become the new trend that researchers try to claim the state of the art at the moment in any classification field.

In this thesis we will be diving deep in the concept of Transformers based method[21] which took the attention of [18] to apply such method for the field of computer vision and opened the gate to other researchers such as [29] who implement the clos-

est same method to classify audio spectrogram with the main focus of dual-modality Transformers [25, 26, 27] as well as audio only[28], aside the new feature Masked AutoEncoder[24]. This is the main approach of this thesis which will be explained thoroughly in later sections.

3 | Background

In this section we will be elaborating on the motivation behind Transformers that researchers have claimed the state-of-the-art by first providing a brief background to Transformers, and then introduce the structure of the model Transformers, that has been vastly studied by researchers for the purpose state-of-the-art. We put the focus on explaining the concepts and features that will be later used in the development of the proposed system. The discussed concepts are including Self-Supervised Transformers, Supervised Transformers ViT, Masked AutoEncoders.

We will end this chapter by elaborating on how machine/deep learning models effectively extract meaningful features from audio data.

3.1 Transformers

Transformers was first proposed for the application of NLP by [21], which is dominant for the field and its application for computer vision was limited to be used in conjunction with CNN like ResNet. However [18] adosovitskiy et al. comes with the motivation of pure Transformers for the applications of computer vision.

The term *Standard Transformer, ViT[Vision Transformer],Standard ViT architecture* or *Standard architecture* has been used by many papers [24, 25, 26, 27, 29] which they all refer to the completely Transformer architecture introduced by adosovitskiy et al.[18]. Through this report, for Standard Transformer architecture we refer to adosovitskiy et al. [18] and Original Transformer refers to Vaswani et al.[21]. Down here we explain all parts of the Transformer architecture. Image below 3.1 shows a peek of the main architecture.

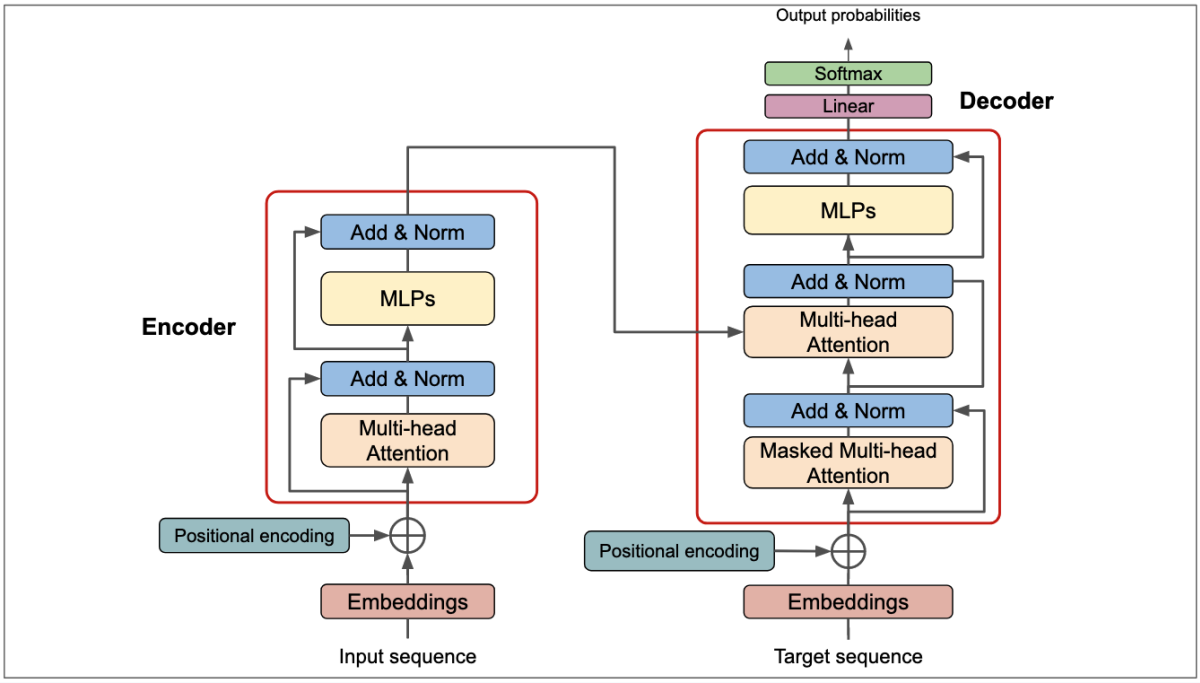

Figure 3.1: Original Transformer architecture

3.1.1 Input

The [21] Vaswani et al. Transformers maps a $1D$ sequence of words/tokens $(x_1, ..., x_n)$ and convert them into embeddings to capture semantic relationships between tokens before feeding it to the Transformer. However in the context of computer vision, an image is a grid of pixels, a two-dimensional representation of visual information, where each pixel has a specific position and contains information about the color and intensity of the image at that point. In other words matrix of pixel values. To handle a $2D$ image ViT reshape the image $\mathbf{x} \in \mathbb{R}^{H \times W \times C}$ into a sequence of flattened $2D$ patches of $\mathbf{x}_p \in \mathbb{R}^{N \times \left(P^2 \cdot C\right)}$

Transformers works well for NLP, however in case of memory and computing requirement it is limited in computer vision. Therefore [18] adosovitskiy et al. scale down an image of $x = [H, W, C]$ size to [224, 224, 3], split the image into non-overlapping fixed-size patches of 16x16, flattened to form a single vector, representing a token of word in case of NLP.

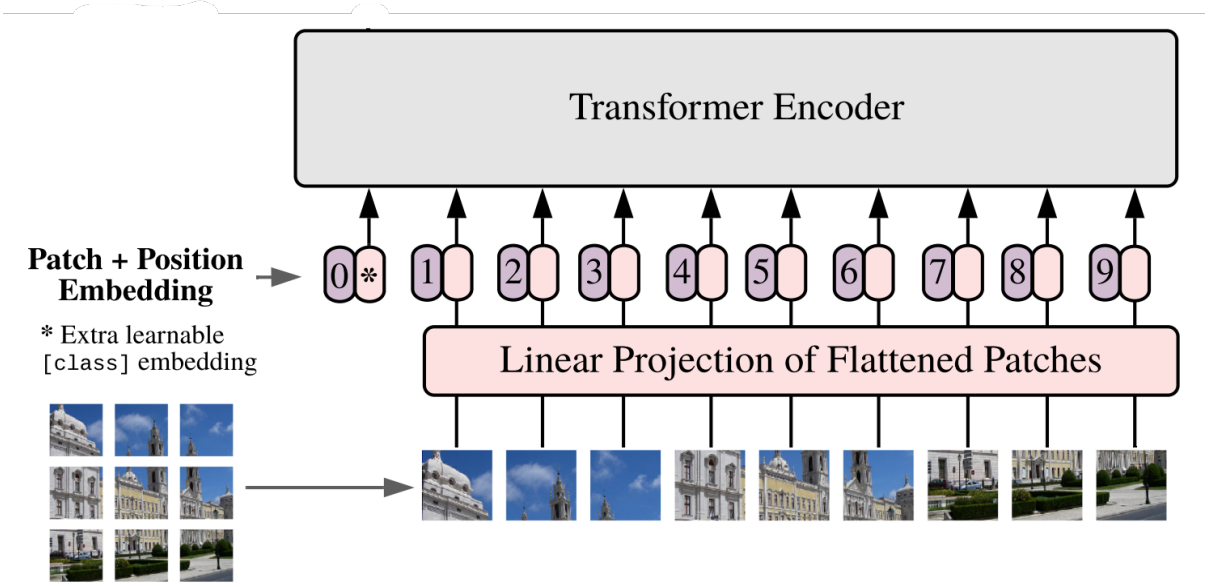

Figure 3.2: An image is divided into patches, creating a sequence $[P, P, C]$

Recurrent Neural Network(RNN) models, in the field of NLP, typically factor computation along the symbol positions of the input and output sequences, it needs to

capture dependencies of a sequence h_{t-1} in order to compute sequence h_t where h can be considered a word for time t. One of the main reasons Transformers outperformed sequence modeling like RNN families was that the whole sequence is fed to Transformers, which takes $O(1)$ rather than $O(n)$. On the other hand [18] uses this idea of sequence input like a text which consist of words to image by patchifying an image into portions and consider each patch as a sequence of tokens same as a sentence of words. With the core mechanism of Transformers namely *Attention* Transformers can have access to whole image with respect to the current patch, however CNN is limited due to its *Kernel* nature.

Parallelism

Parallelism refers to the ability to perform computations concurrently across different parts of the input data, which this feature by nature is in Transformers, however CNN needs to exploit this through their convolutional and pooling operations. This is another reason Transformers can outperform CNN like architectures.

3.1.2 Positional Embedding

Unlike convolutional neural networks (CNNs), transformers doesn't inherently preserve the sequential order of input data, and as such, positional information needs to be explicitly encoded. Positional embeddings or encoding should provide the model with information about the position or order of tokens in the input sequence. It is typically added to the input embeddings of each token. The idea is to create a representation that combines the semantic information of the token with its position in the sequence. It's an additive types of embeddings, meaning it is added to each token. The dimensionality of positional embeddings is arbitrary but is typically the same as the dimensionality of the image token. Commonly used positional encoding is based on sine and cosine functions. The formula [1][21] for positional encoding is as follows for each position *pos* and each dimension i of the positional embedding and d is the dimension of the position embedding which is equal to dimension of the image patched

token. The authors state "We chose this function because we hypothesized it would allow the model to easily learn to attend by relative positions".

$$PE_{pos,2i} = \sin\left(\frac{pos}{10000^{\frac{2i}{d}}}\right) \quad PE_{pos,2i+1} = \cos\left(\frac{pos}{10000^{\frac{2i}{d}}}\right) \tag{1}$$

The key intuition to positional embedding is to create a pattern in the embedding with respect to the position of each patch. The formulation calculates as $\sin(\frac{pos}{a^i})$ approaches 0, $\cos(\frac{pos}{a^i})$ approaches 1. The created pattern allows the model to preserves the length constraint [30].

3.1.3 Class Token

A feature where Original Transformer applied to the architecture is [CLS] token which is randomly initialized values with the same dimension of input token and prepend aside the input token. Since randomly initialized, by itself does not carry any information. In the training process, particularly during fine-tuning on downstream tasks, in the backpropagation the class token starts accumulating meaningful features from the input token. For a classification task which Transformers are usually connected to a MLP head which uses the last layer's class token vector.

3.1.4 Distillation Token

DeiT which stands for *Distilled Data-efficient Image Transformer*. As the name suggest a DeiT model uses the technique *Knowledge Distillation* to transfer knowledge from the larger model, *ViT*, to the smaller one. By term they are called *teacher* and *student* models. It relies on a *distillation token* ensuring that the student learns from the teacher through attention.

During training, the student model learns to mimic not just the final predictions of the teacher model, but also the distribution of probabilities assigned to different classes. This allows the student model to learn from the rich information encoded in

the teacher's predictions, even for cases where the teacher model might not be perfectly confident in its predictions. This is called *Soft Distillation* which is typically achieved by using a loss function that compares the output probability distributions of the teacher and student models.

The student model also tries to directly replicate the exact predictions of the teacher model, aiming to match its output as closely as possible, instead of providing the probabilities assigned by the teacher model, the actual hard labels (class indices) predicted by the teacher model are used as targets for the student model. This is called *Hard Distillation*. While this approach may lead to faster convergence and simpler training dynamics compared to soft distillation, it might also result in the student model being overly reliant on the teacher's predictions and less capable of generalizing to unseen data.

In summary, DeiT represents a novel approach to image classification that combines the strengths of the Transformer architecture with the efficiency of knowledge distillation, making it particularly suitable for scenarios where computational resources or training data are limited. It also is publicly available with vast variety of examples which makes it easy to use.

3.1.5 Modality Embedding

Audio and video are commonly reserved in different modalities. Audio spectrograms represent a visual representation of the frequencies present in an audio signal over time. They are a way of visually displaying the spectrum of frequencies of a sound wave as it changes over time. On the other hand a frame corresponds to an individual image which represent spatial content of one moment where time is not included. Therefore the nature of the features are completely different and Transformers should learn a way to distinguish between these two.

This is an additive embedding to add modality embedding to each token of different modals. Modality embeddings are vectors of same dimension as audio/visual

tokens which are randomly initialized values and by itself does not have any meaning, however during backpropagation the model learns to optimize the modality vectors based on audio as well as video.

Modality embedding is not a feature that is related to Transformers. The intuition behind this feature lies in the idea of leveraging audio and video to enhance model performance.

3.1.6 Architecture

Transformers consist of two micro blocks, namely *Encoder* and *Decoder*. As you can see in the image 3.3 the architecture at a high level. Original Transformer uses 6 encoder/6 decoder layers, but Standard Transformer uses 12/24/32 encoder/decoder layers stacked on the top of each other.

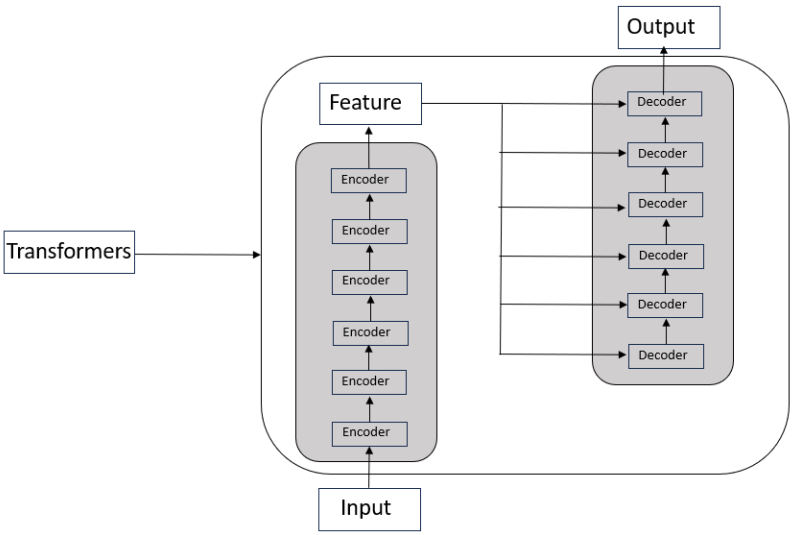

Figure 3.3: Original Transformer Encoder/Decoder layers at high level

Architecture: Encoder block

Inside Transformer's encoder layer has mainly 3 different sublayers, namely *Self Attention*, also known as *Scaled dot-product Attention*, *Normalization* and *MLP*. The fundamental sublayer is called *Self Attention* or in case of multiple *multi-attention heads*. The

following image [3.4] shows a high level look of encoder architecture.

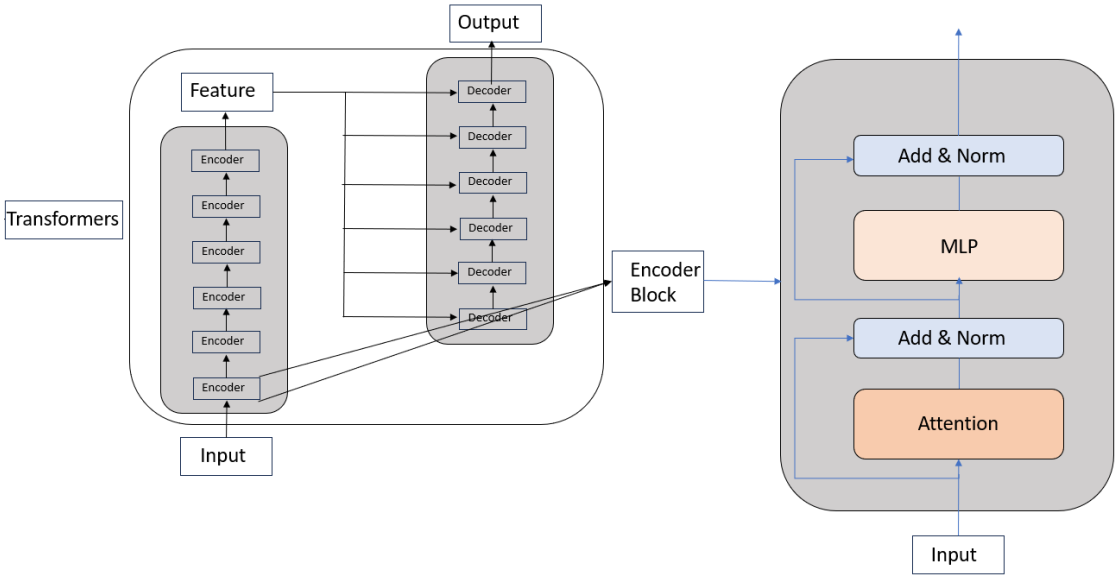

Figure 3.4: Encoder Transformers at high level. This Encoder relates to both Original[21] as well as Standard Transformer[18]

Architecture: Decoder block

The Transformer *Decoder* is exceptionally important. It all depends to its application. For example for a *Classification* task such as [18, 28, 29] they drop the decoder part completely, however classification tasks such as [25, 26, 27] do use the Decoder. This will be discussed in later chapters. The decoder layer is pretty much the same as Encoder except additional multi-head attention that operated over the output of the encoder. The goal of the decoder is to fuse encoder output with the target sequence and to make predictions. Decoder is also typically repeated the same times as encoder.

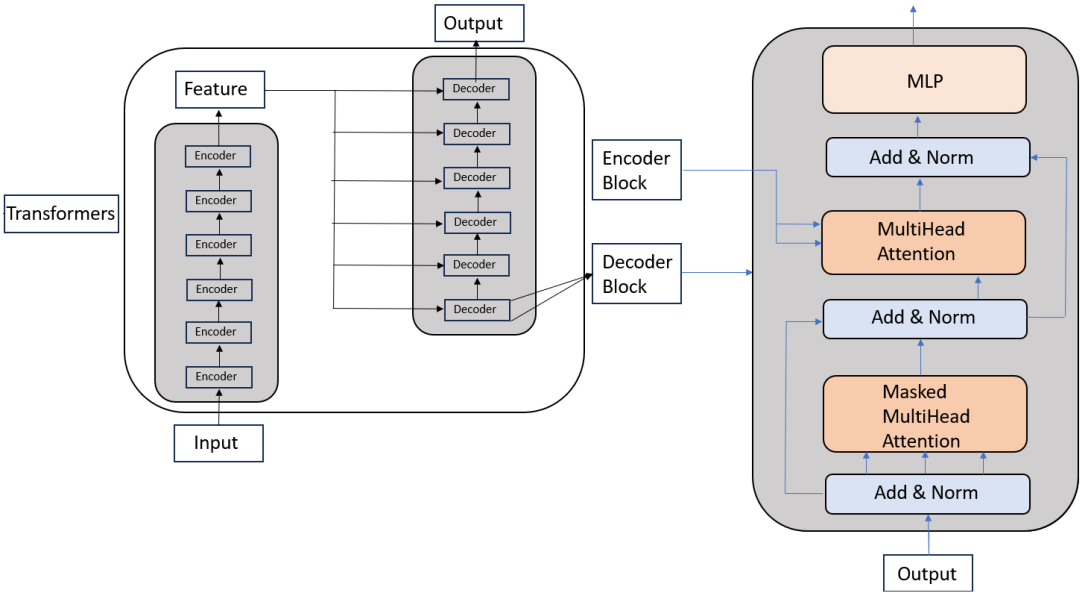

Figure 3.5: Decoder Transformers at high level. This Decoder relates only to the Original Transformer from Vaswani et al.[21]

3.1.7 Self-Attention & MultiHead Attention

Self-Attention, or *multi-head attention*(in case multiple) also known as Scaled dot-product Attention[21] is the primary core of the transformer architecture. The attention mechanism allows the model to focus on different parts of the input sequence when making predictions for a particular element in the output sequence. Essentially, self-attention empowers the Transformers to selectively focus on crucial portions of input data, emphasizing meaningful information while downplaying the significance of other parts. Transformer models typically have multiple attention heads, and each attention head learns different aspects of the relationships in the data. The Attention mechanism uses *key, query* and *value* concept. The key/query/value concept is analogous to retrieval systems.

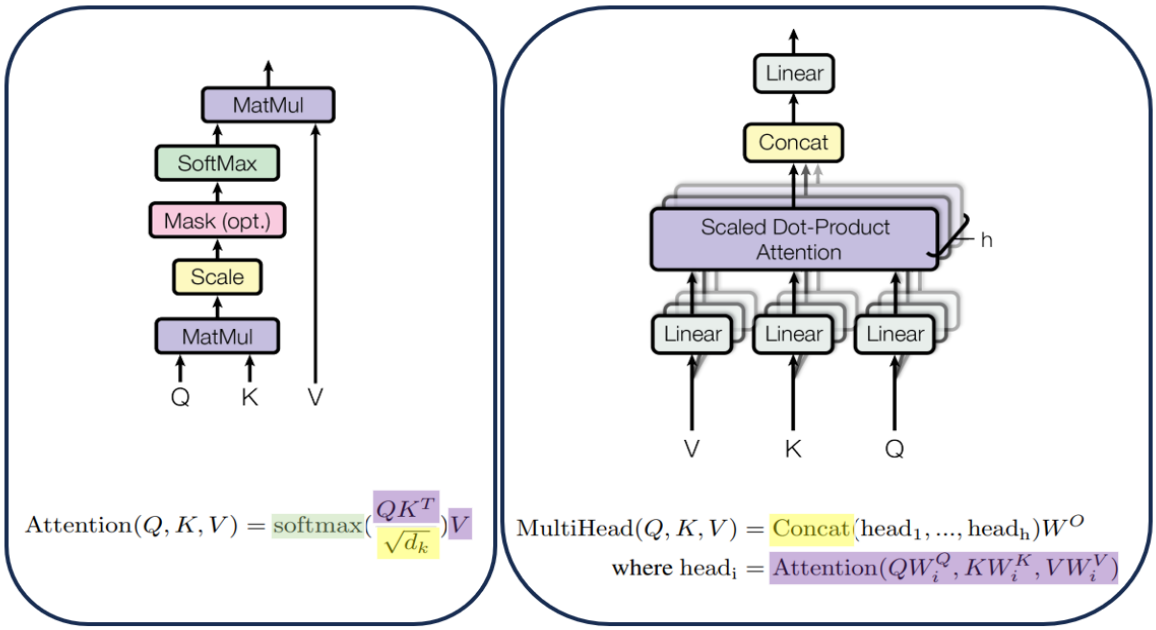

Figure 3.6: Self-Attention and Multi Head-Attention; Color coded. Graph and formula taken from Vaswani et al. [21]

Looking at the image 3.6, left, Self-attention, the first *Matrix Multiplication* from bottom, Q and K is the dot product of two 1D matrices which shows similarity measure, same as *Cosine Similarty*, where higher value of dot product of these two vectors shows how much they relate to each other. Each element of the Query vector will be multiplied with all elements of Key vector. Vaswani et al.[21] call this *attention weight* of values. The next step is to scale and normalize the values by dividing $\sqrt{d_k}$ which has same dimensionality as $K/Q/V$. Then using *SoftMax* we make sure of negative values and squeeze the matrix between 1 and 0, a probability distribution. *Mask* is optional, we do not focus on the section. The output of the self-attention will be passed to the *Add & Norm*, *MLP*, next encoder layers and eventually fed to *Decoder* as shown in image [3.4] where each will be explained in their sections.

The *MultiHead Attention* in image [3.6], right, shows the parallel computation of *Self-Attention*. In the formula, we have h number of *heads* which is the output of the *Scaled dot-product Attention* and W^O is a learned linear transformation.

Query, Key and Value

The original patched image vectors are multiplied by learnable parameters to obtain *Q, K, V* vectors as shown in image[3.7] where the *Key* vector and *Value* vectors are identical. The reason is to relate two different sequences of image to one another.

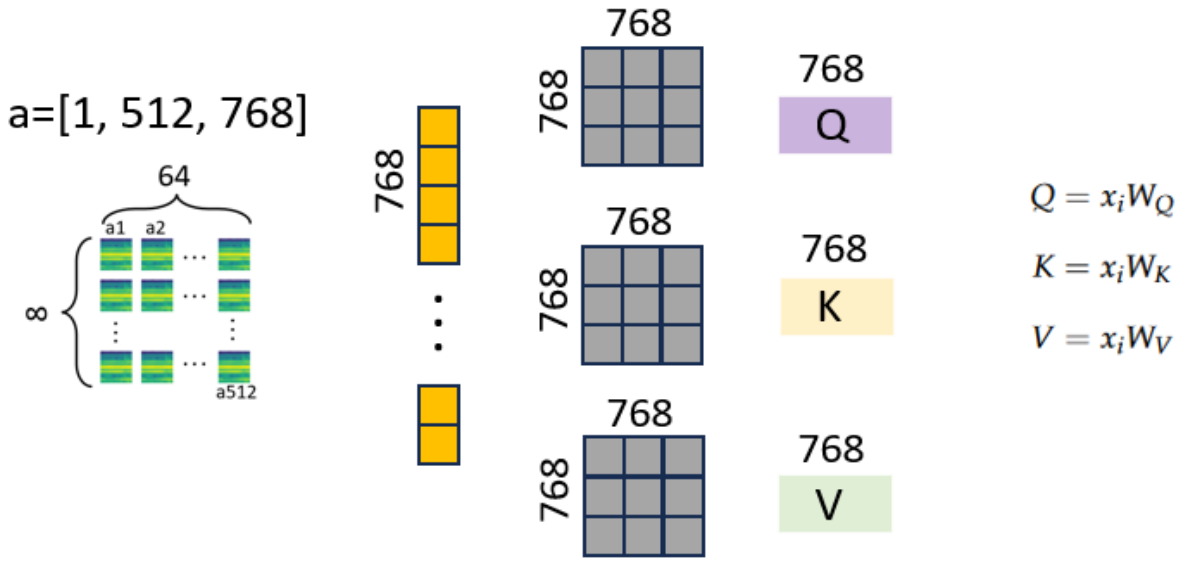

Figure 3.7: Randomly initialized weight matrices

Attention head in Encoder/Decoder

In the image[3.8] below we see the key difference between attention head process in Encoder[upper] and Decoder[lower]. As we showed previously in image[3.5], two feature vectors which are the output of the Encoder will be combined with one feature vector that comes from lower attention head of the Decoder.

- Q = The output feature vector from Decoder's masked attention
- K = The output feature vector from Encoder
- V = The output feature vector from Encoder

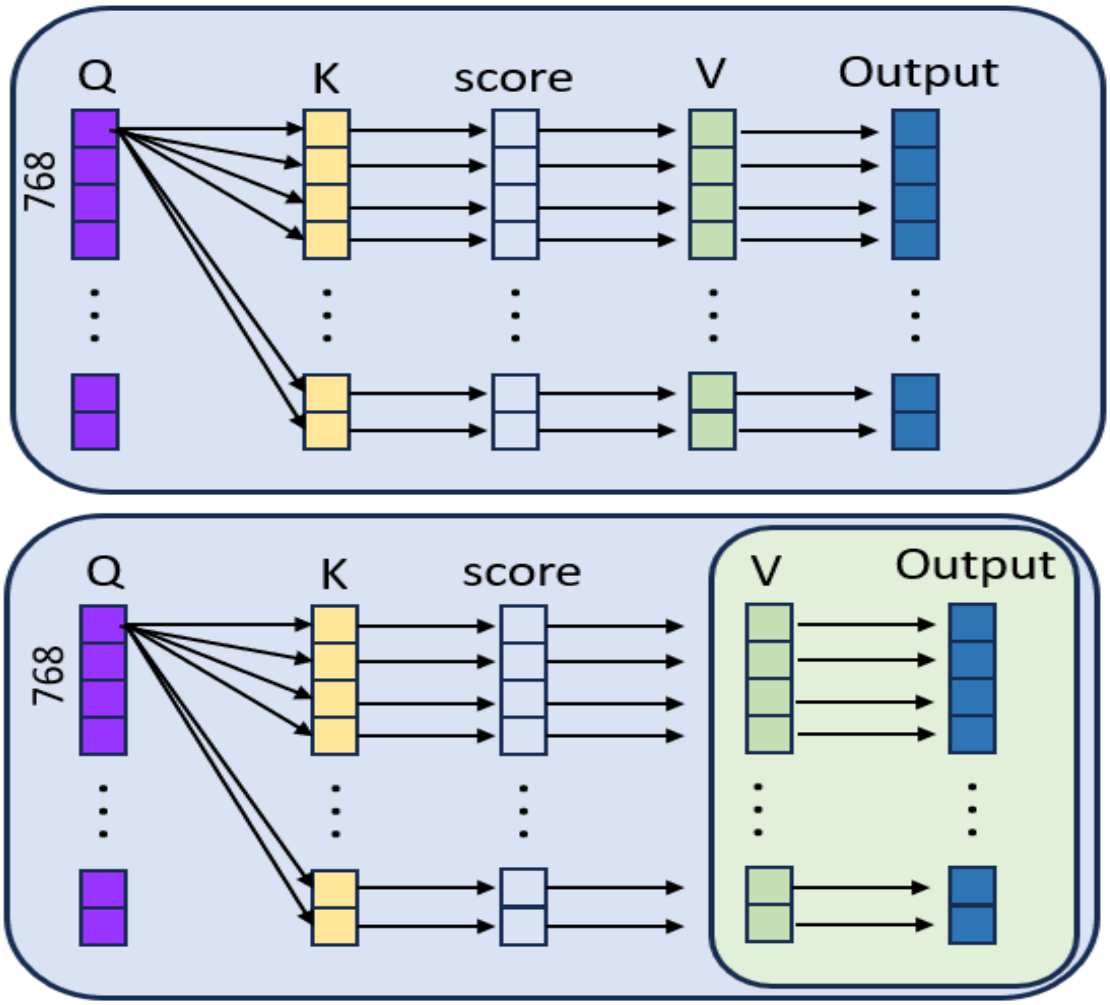

Figure 3.8: Difference between Encoder attention head and Decoder attention head.

3.1.8 Add & Norm

Add: Residual Connection

The *Residual Connection* which is also known as *Skip Connection*, is indicated by *add* after attention head allows the original input to be added directly to the output of attention which both have dimension d.

Norm: Layer Normalization

Layer Normalization in attention heads contributes to the stability, efficiency, and generalization capabilities of the model during training. This process reduces the *internal covariate shift*, which can occur during training when the distribution of activations in a layer changes, by normalizing the sum of activation function output to have a mean *mu* 0 and std *sigma* 1. For example the output of the activation function can get large gradient steps and hence unstable training. The formulation works as follows:

$$\mu^l = \frac{1}{H}\sum_{h=1}^{H} x_i^l$$

$$\sigma^l = \sqrt{\frac{1}{H}\sum_{h=1}^{H}\left(x_i^l - \mu^l\right)^2}$$

Given input sample of x_i of dimension H, we calculate the mean *mu* of dimension x_i^l, we subtract it from input sample x_i across all dimension l, square it to mitigate negative values, divide by $\frac{1}{H}$ and finally take the square root.

$$\hat{x}_{i,l} = \frac{x_{i,l} - \mu_i}{\sigma_i^2 + \epsilon}$$

Then we normalize each input sample such that the elements in the sample have zero mean and 1 unit variance. ϵ is for numerical stability in case the denominator becomes zero by chance.

$$y_i = \gamma\hat{x}_i + \beta \equiv \mathrm{LN}_{\gamma,\beta}\left(x_i\right)$$

Finally, there is a scaling and shifting step. γ and β are learnable parameters.

Adosovitskiy et al.[18] reports *Layer Normalization* is applied before each block and *Residual connections* are after the blocks.

3.1.9 MLP

Multi-Layer Perceptron(MLP) is a type of FeedForward Neural Network, which means a backpropagation is involved. It consist of multiple layers of nodes or neurons. Connection between each neurons in adjacent layer is associated with a value by term weight which represents the strength of the connection. Except the input layer the layers can include an activation function which introduce non-linearity to the summation of weights and inputs, allowing the network to learn complex patterns in data input. Additionally output of activation function can be associated with a value by term bias for network improvement. By term the intermediate layers are also called hidden layers. In the figure below [3.9] we see a simple example of the structure where x_i is data input vector and each x is associated with a w weight vector depending on number of neurons in hidden layer. $af(xw)$ stands activation function for non-linearity where the output is associated with a b bias value.

The w_{ij} are indeed the parameters that changes over time in backpropagation with the help of Optimization function and Loss function which we will be discussing them in later chapters.

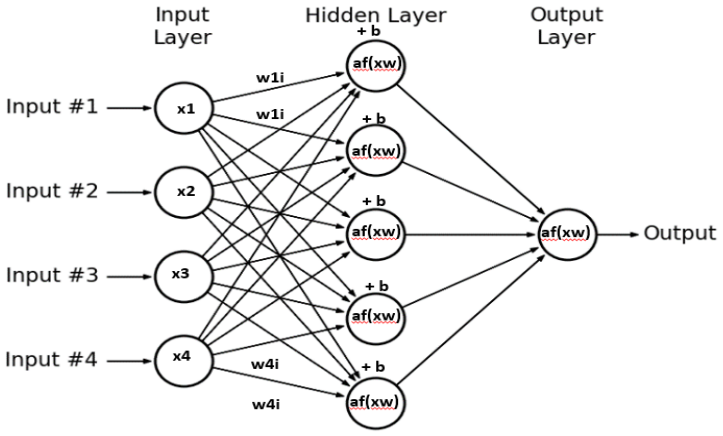

Figure 3.9: MLP structure example

In Transformers MLP by default is applied in each Encoder/Decoder blocks. It composed of two hidden layers with a Gaussian Error Linear Unit activation function in-between the hidden layers. The main intuition behind a non-linearity function after

attention block is to scale down to dimensionality back to original input dimension for either next encoder or decoder layer.

3.1.10 Activation function[GELU]

One of the reason an activation function is used is often to incorporate a thresholds, which enable neurons to activate selectively based on the input data. This selective activation helps the network focus on relevant features and ignore irrelevant ones. There many activation function which one of them is GELU.

The Gaussian Error Linear Unit (GELU) activation function is a non-linear activation function that has gained popularity in deep learning for its smoothness, especially in transformer architectures by [21]. It is designed to address some of the limitations of other activation functions. The function can be described as follows:

$$\text{GELU}(x) = x \cdot \Phi(x) = x \cdot \frac{1}{2}\left(1 + \text{erf}\left(\frac{x}{\sqrt{2}}\right)\right)$$

where x is the input, $\Phi(x)$ is the cumulative distribution function of the standard Gaussian distribution. $erf(x)$ is the error function, which returns the cumulative distribution function of a standard Gaussian distribution. The error function is another activation function where GELU follows to bring smoothness to the outcome. $\frac{x}{\sqrt{2}}$ is scaling factor is used to ensure that the input to the error function has a mean of zero and a standard deviation of one as well as $\frac{1}{2}$ which ensures that the output of the activation function is in the range of $[0, 1]$. This is a common practice in neural network activation functions.

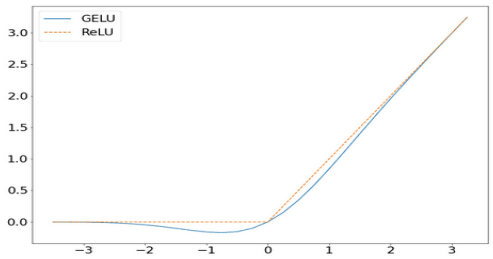

Figure 3.10: GELU vs. RELU activation function. Random image from internet.

The threshold that RELU activation function brings is shows in the above figure [3.10]. Any input value in x-axis which is smaller than 0 will get value of 0 otherwise based on y-axis. The higher the value the higher the importance of the neuron in the network. GELU brings smoothness to this calculation where RELU has a non-smooth point zero. This smoothness can contribute to better gradient flow during training and, in some cases, lead to faster convergence.

3.1.11 MAE:

Earlier we mentioned *Masked Auto Encoder*[MAE][24] from Facebook AI Research. This new interesting method has taken the focus of many researchers. The idea of MAE is very simple and effective. The regular preprocessing of image as explained before, then *Mask* or better to say remove/drop 75% of the image, feed to a *self-supervised* Transformer model and the output will be as original as depicted in the image below. The main idea is to reconstruct the missing pixels using an asymmetric Encoder/Decoder Transformers, where Encoder operates only on the remaining 25% visible of the image, and a "lightweight" Decoder that reconstruct the original image missing parts from the *Latent space* and the *Masked token*.

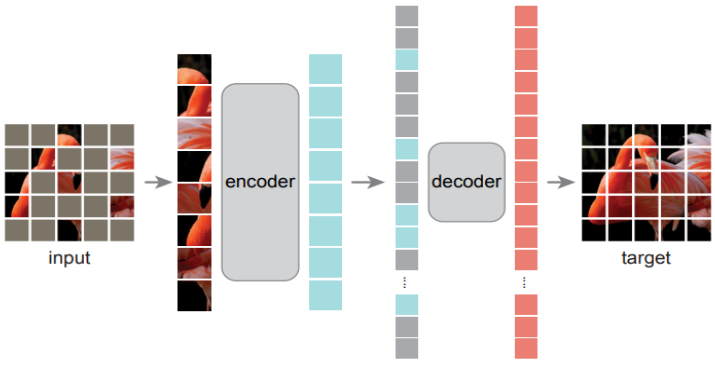

Figure 3.11: Masked AutoEncoder at high level. Image taken from the original paper, *Masked Autoencoders Are Scalable Vision Learners*[24]

Following ViT[18], they preprocess an image with same instruction regular non-overlapping patches and *randomly* sample a subset of patches without replacement following a uniform distribution. The random shuffled-index images are fed to a stan-

dard Transformer Encoder of *N* attention head blocks. This allows the Encoder to train on a very large dataset using a fraction of compute and memory. Then the encoded visible patches are combined with *Masked token* which is a shared, learned vector that indicates the presence of a missing patches to be predicted. Then these patches are fed to a "light" standard Transformer Decoder to regenerate the image patches to it's original pixels.

The main reason that this method has been studied in the thesis project is that the method has become one of the latest trends in machine/deep learning classification method as [25, 26, 27, 28] have used the method in their papers. Although the impact accuracy and State-of-the-art of the method is not significant at all, yet the computation and memory usage has been reported by the original paper[24] to be optimized by **2.8** up to **4.8** depending on the size of Encoder/Decoder.

3.2 Feature Extraction

To facilitate the training of any statistical or machine/deep learning models , the very initial step involves extracting valuable features from an audio signal. This necessitates audio feature extraction, a crucial stage in audio signal processing, a specialized subfield of signal processing dedicated to the manipulation and processing of audio signals. This process encompasses the removal of unwanted noise, the harmonization of time-frequency ranges achieved through the conversion of digital and analog signals of audio, and a concentration on computational methods designed for the modification of sound.

Mel Filterbanks

In audio processing, particularly in the analysis of music, the mel scale is used to approximate the human ear's sensitivity to different frequencies. The mel scale is divided into a set of overlapping triangular filters, commonly known as mel filterbanks. Each triangular filter in the mel filterbank is often referred to as a "mel band". mel

bands can vary according to the deep learning algorithms. These filters are applied to the spectrum of an audio signal to divide it into a set of frequency bands that are more perceptually relevant than a linear scale. Mel filterbanks are a set of filters applied to the frequency spectrum of an audio signal, while a mel spectrogram is a visual representation derived from the application of these filters. The mel spectrogram provides a time-varying representation of the energy in different mel frequency bands, making it a useful tool in various audio processing applications. Mel filterbanks are often used as a feature extraction technique. It helps in capturing the distribution of energy across different frequency bands, giving more weight to frequencies that are more perceptually important.

Mel Spectrograms

A mel spectrogram is derived from the mel filterbanks. It is a representation of the short-term power spectrum of a sound signal as it varies with time. It is obtained by applying the mel filterbanks to the magnitude spectrum of the signal and then taking the logarithm of the energy in each filter. The mel spectrogram provides a way to visualize the energy distribution in different frequency bands over time. In speech processing, mel spectrograms are used for feature extraction in tasks such as emotion detection. In music analysis, they play a crucial role in tasks like genre classification, instrument recognition, and audio classification. By capturing the distribution of energy across perceptually relevant frequency bands over time, mel spectrograms provide a valuable representation for understanding and processing audio signals in both speech and musical contexts.

4 | Method

We experimented two different methods to classify audio in this project. One model which we call *AST* after [29]*Audio Spectrogram Transformers*, is following Strandard Transformers ViT[18]. The second model which is the main focus of this thesis we call it *DualMAE* after Gong et al. [27] which uses the newest methods and framework to claim the state-of-the-art. A handful of similar papers are also studied aside the main paper, which follows the same main method and framework with slight difference.

Since we are working on two models with different structures, we will be explaining the structure of these two models.

4.1 Dataset

VGGSound is large-scale dataset of audio and video introduced and publicated by Chen et al. [31]. The dataset contains 200k audio-visual clips from Youtube. Each has 10 seconds of 300 different classes such as *home, people, animals, sport and music*. The dataset provides a csv file containing *YouTube ID, start seconds, label, train/test split*. Filtering all unnecessary non-related to music remains *57* classes of *46203* video as shown in image 6.8

Train/Test: We arrange 20% of total of each class for *test* purpose and the rest is for training. Result for train/test is 36984 and 9219 respectively.

4.1.1 Preprocessing

Audio: We extract and sample the 10 seconds audio independently for each video. The first preprocessing of the audio waveform is to convert to a sequence of 128-dimensional log Mel filterbank (fbank) features computed with a 25ms Hanning window every 10ms. The result is a 1024 representing *time* and 128 representing for *frequency* spectrogram. As the regular rule of patching input data mentioned by [24, 25, 26, 27, 29], we split the whole spectrogram to a $16x16$ patches, resulting to $audio = [a^1, a^2, ...a^{512}]$ as the input to the model.

Video: For videos, we uniformly sample 10 RGB frames from each 10 seconds videos, center cropping and then resize each frames to a $224x224$. The next step is to normalize the images which we use *ImageNet's* mean and standard deviation. Since [24, 25, 27, 28, 29] where all are published very recently, use *ImageNet* initial weight as well is this project. Patchifying a $16x16$ as audio, resulting to $vidoe = [v^1, v^2, ...v^{196}]$ as it is represented in figures [4.1, 4.2].

Patchification: Patchification is global. Meaning we use the same method for both model structures represented in this project. We first patchify the audio spectrogram vector of size $a_x = [x, 1024, 128]$ where $x \in X^n$ where n is the total audio/video. 1024

represents *time* and *128* frequency, into a sequence of $N16x16$ patches with no overlap in time and frequency. Patchification gives a sequence of $a_x = [x, 512, 768]$. The same patch size is used for video $N16x16$ where one frame of size [x, 3, 224, 224] where 3 represents channel and 224 for H/W. The result is $v_x = [x, 196, 768]$.

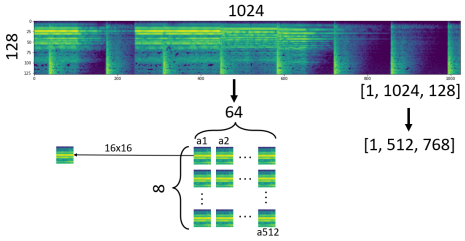

Figure 4.1: Audio patchification

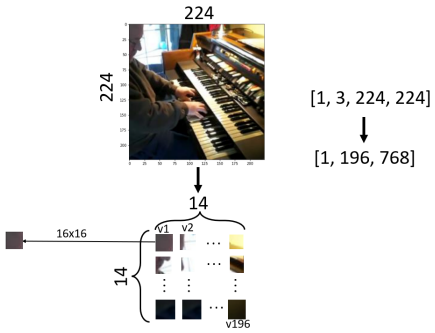

Figure 4.2: Video patchification

Design & Implementation

We first store the extracted audio wav format of all videos, 10 uniformly re-sampled frames of all videos, aside a json file with information of [*Video_id, wav/vid location, label*] on ITU's cluster.

Our DataLoader loads and preprocess 1 wav file and 1 random visual frame content. This is very important since we had a big issue realizing it. The preprocessing of audio includes loading the audio using *torchaudio* library, subtract the *mean* of wav file from the file, and convert the wav file using *torchaudio.compliance.kaldi.fbank* to a [x, 1024, 128] representing [batch, time_frame_num, frequency_bins] spectrogram. For videos, 1 random image is loaded using *PIL* libary, and then resized, center cropped and normalized using *torchvision.transforms* to a vector of size [x, 3, 224, 224] representing [batch, channel, H, W]. Labels are label encoded with [0,1] such that represents the index of the class from the list that is read from json file. This gives us a vector of [x, 57] representing [batch, num_class]. In total, our dataloader's output is [spectrogram, image, label_indexed, label string format].

4.2 Common Architecture

4.2.1 Sinusoidal Position Embedding

Following the Sinusoidal Position Embedding formula from [21] we create our own function. The formula is as bellow:

$$PE_{pos,2i} = \sin\left(\frac{pos}{10000^{\frac{2i}{d}}}\right) \quad PE_{pos,2i+1} = \cos\left(\frac{pos}{10000^{\frac{2i}{d}}}\right)$$

Where the function takes two inputs, max sequence length which is equivalent to the total number of patches [512, 196] for audio and video respectively and embedding dimension which is 768, and outputs the encoded values of size [x, total_num_patch, emb] where x is total number of inputs.

4.2.2 Optimizer

For the choice of optimizer we first decided to collect information from some papers which we have focus on in this thesis. We exclude papers that are not based on Transformers or does not mention in their paper.

Paper	Optimizer	Momentum	Weight Decay	Base LR
Vaswani et al.[21]	Adam	$\beta_1 = 0.9, \beta_2 = 0.98$	0.5	-
huang et al.[22]	AdamW	$\beta_1 = 0.9, \beta_2 = 0.95$	1e-5	1-e4
he et al.[24]	AdamW	$\beta_1 = 0.9, \beta_2 = 0.999$	0.05	1e-3
asodovitskiy et al.[18]	Adam	β_1 = -,β_2 = -	0.1 - 0.3	-
huang et al.[25]	AdamW	$\beta_1 = 0.9, \beta_2 = 0.95$	1e-5	2-e4
mgeorgescu et al.[26]	Adam	$\beta_1 = 0.95, \beta_2 = 0.999$	0	1e-4
gong et al.[27]	Adam	$\beta_1 = 0.9, \beta_2 = 0.95$	0.5	3e-4
chen et al.[28]	AdamW	$\beta_1 = 0.9, \beta_2 = 0.98$	0.01	-

Table 4.1: Different papers; Choice of optimizer. The '-' value represents empty information and/or many different values.

Looking at these papers, they all optimize their model using Adam optimizer with different parameters. Therefore we directly choose our optimizer the same.

Adam short for Adaptive Moment Estimation is a type of *Stochastic Gradient* optimizer to update network weights iterative based on loss value.

$$w_t = w_{t-1} - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

Where w_t is the current weight which is considered to be updated based on previous weight at time $t-1$. η is learning rate, and ϵ is for negating zero division. the optimization is as follows:

1. $t, m_t, v_t = 0$
2. Compute gradient g_t on the current minibatch with respect to w_{t-1}
3. Update biased first moment estimate (1)

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1)\, g_t \tag{1}$$

4. Update biased second moment estimate (2)

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2)\, g_t^2 \tag{2}$$

5. Correct bias in first moment

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{3}$$

6. Correct bias in second moment

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{4}$$

7. Update parameter using (1)

In the above equation β_1 and β_2 are parameters that control the exponential decay rates of the moment estimates.

Design & Implementation

We use *torch.optim.Adam(lr=1-e4, weight_decay=5e-7, betas=(0.95, 0.999))*. The implementation is a standard pytorch training/test procedure.

4.2.3 Init-weight pretrained

Weight initialization of the model can impact the convergence speed and final performance of the model. It is often applied to the parameters of the self-attention mechanism, MLP, and other components. By ensuring a balanced initialization of weights. In this thesis we initialize models with two different weights and compare the result of these two init-weights with no weight initialization at *Result* section.

ImageNet 1K pretrained weight which is based on ImageNet DataSet. It is one of the largest and most widely used datasets in the field of computer vision, where the dataset itself is not publicly available. However the models that are trained on ImageNet 1k is available in multiple sources, as we download the trained model from HuggingFace. ImageNet 1k refers to a subset of ImageNet dataset that contains 1,000 categories. These categories cover a wide range of objects, animals, scenes, and other visual concepts. Each category contains a variable number of images up to 1.3M images. Furthermore we notice that weight initialization Transformers models using this specific has become the standard starting point as [25, 26, 27, 28, 29] initialize their models with this pretrained weight.

AudioSet is perhaps the largest dataset available similar to VGGSound. By AudioSet pretrained weights we mean the weights of a similar model that is trained on AudioSet. We download the weights which is provided by Gong et al. [27] from their github repository. AudioSet is a large-scale dataset developed by Google that consists of manually annotated audio events extracted from YouTube videos. It contains over 2 million 10-second sound clips from various sources, covering a wide range of audio categories such as musical instruments, human sounds, animal sounds, environmental sounds, and more.

Xavier Uniform Xavier uniform weight initialization is a popular method used to initialize the weights of Transformers. Xavier uniform initialization sets the weights of a block layer such that the variance of the inputs and outputs to each layer are approximately equal. This helps in preventing vanishing or exploding gradients during training, which can impede learning.

We use *torch.nn.init.xavier_uniform_()* to initialize the model with no pretraining.

4.2.4 Loss function

The choice of loss function for classification can be vast, but a very simple and widely used choice can be *Cross Entropy*. Cross Entropy simply calculates the distance between predicted value and true value of the class. The model updates its parameter using the optimizer based on the distance value which is the output value of the loss function, until the loss value has been minimized. The process of loss calculation can be simplified as follows:

$$CE(t, p) = -\sum_{x} t(x) \log p(x)$$

where $t(x)$ is the true probability distribution, and $p(x)$ is the predicted probability distribution, and the cross-entropy loss is calculated by summing over all events x and taking the negative logarithm of the predicted probability $p(x)$ weighted by the true probability $t(x)$

4.3 Model Architecture[AST]

We adopt the architecture of AST model from Gong et al.[29] which is closest architecture as Standard Transformers[18]. The model does not have the decoder. It is a DeiT based model which is pretrained and finetund on ImageNet 1k data and achieved an accuracy of 83.04 [32]. The model takes only audio spectrograms as input, and outputs a probability distribution of the class. The architecture is the same as *ViT*[18],

except the distillation token [distil] which has learnable weights aside [CLS] token. We patchify the audio input the same regular patchification of size $16x16$ with no overlap in frequency and time. $a_n = [x, 1024, 128]$ as audio spectrograms to patches of $p_x = [x, 512, 768]$ where x, n are number of patches and number of audio inputs respectively. Similar to standard Transformers ViT, it has a depth layer of 12 and width of $\mathbb{R}^{768}$ which is equivalent to input vectors. Figure [4.3] shows a high level structure look of the model AST.

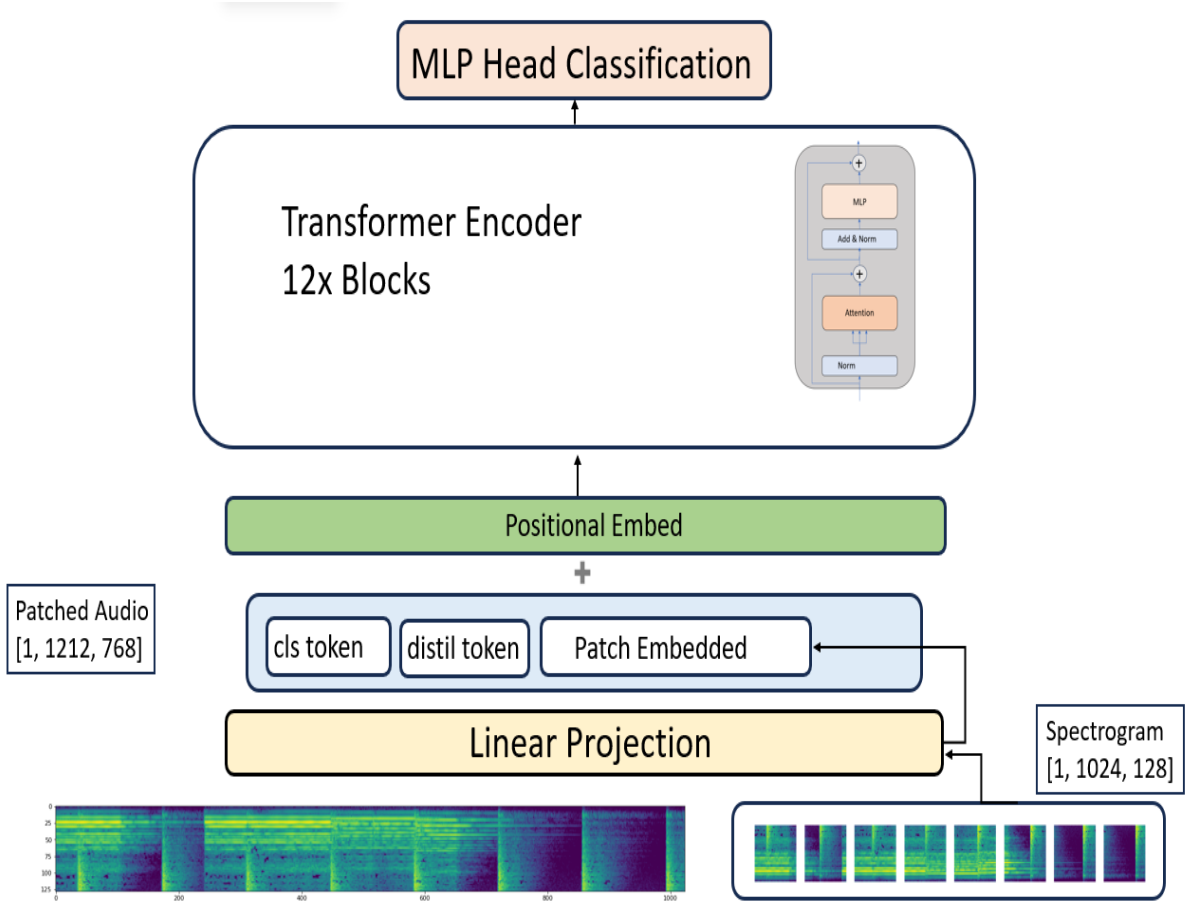

Figure 4.3: AST Main Structure

Following, we add *sinusoidal position embedding* to patches to allow the model to capture the spatial structure of the 2D audio spectrogram. This will be the preprocessing for audio spectrograms as input to the model. Figure 4.4 shows the process.

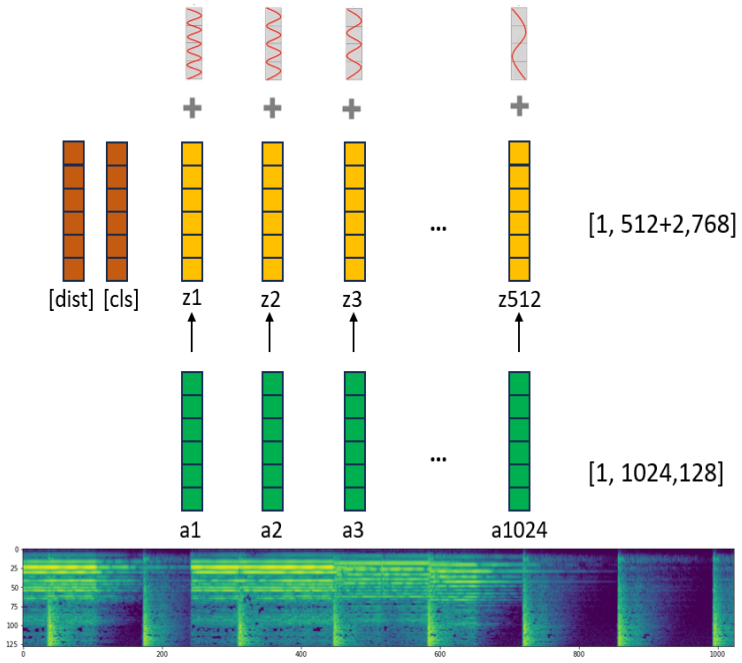

Figure 4.4: AST input

Design & Implementation

Patchification method is as follows: *nn.Conv2d(1, 1024, kernel_size=16, stride=16)*. The model can be downloaded using: *timm.create_model(model, pretrained=True)*, and the model name is *'vit_deit_base_distilled_patch16_224'*.

4.4 Model Architecture[DualMAE]

Recently researchers focus to classify contents such as audio contents, based on *Self Supervised* methods. We notice a new trend that claim State-of-the-Art. In general, the top accurate content classifications has three things in common, *Self-Supervised*[25, 26, 27, 28], *DualModality*[25, 26, 27] and *Masked Modeling*[25, 26, 27, 28]. Therefore we decided to study the recent trends based on these recent papers with the focus on[27]. This method claims the novel approach for audio classification for the time being, and the main focus of the researchers based on PapersWithCode. Unlike ViT[18] and AST[29], this method uses a decoder as well, and the result of the self-supervised method indicates how well it will perform on downstream task such as classifications.

This method includes dual-modality which we use audio and video to enhance model performance, based on the idea that human can perceive better musical instruments/genres etc. both from vision and listening. Even though acoustic and visual modalities have different properties, yet humans are able to seamlessly connect and integrate them to perceive.

The method includes the *Masked modeling* which previously explained thoroughly in previous chapter. Combining all these methods we assemble the architecture where has brought the novelty for audio classification.

4.4.1 Self-Supervised

The model starts with a self-supervised model architecture, where the Transformer Decoder is used as well. In the image[4.5] below we can see a high level look at the architecture we use in this thesis.

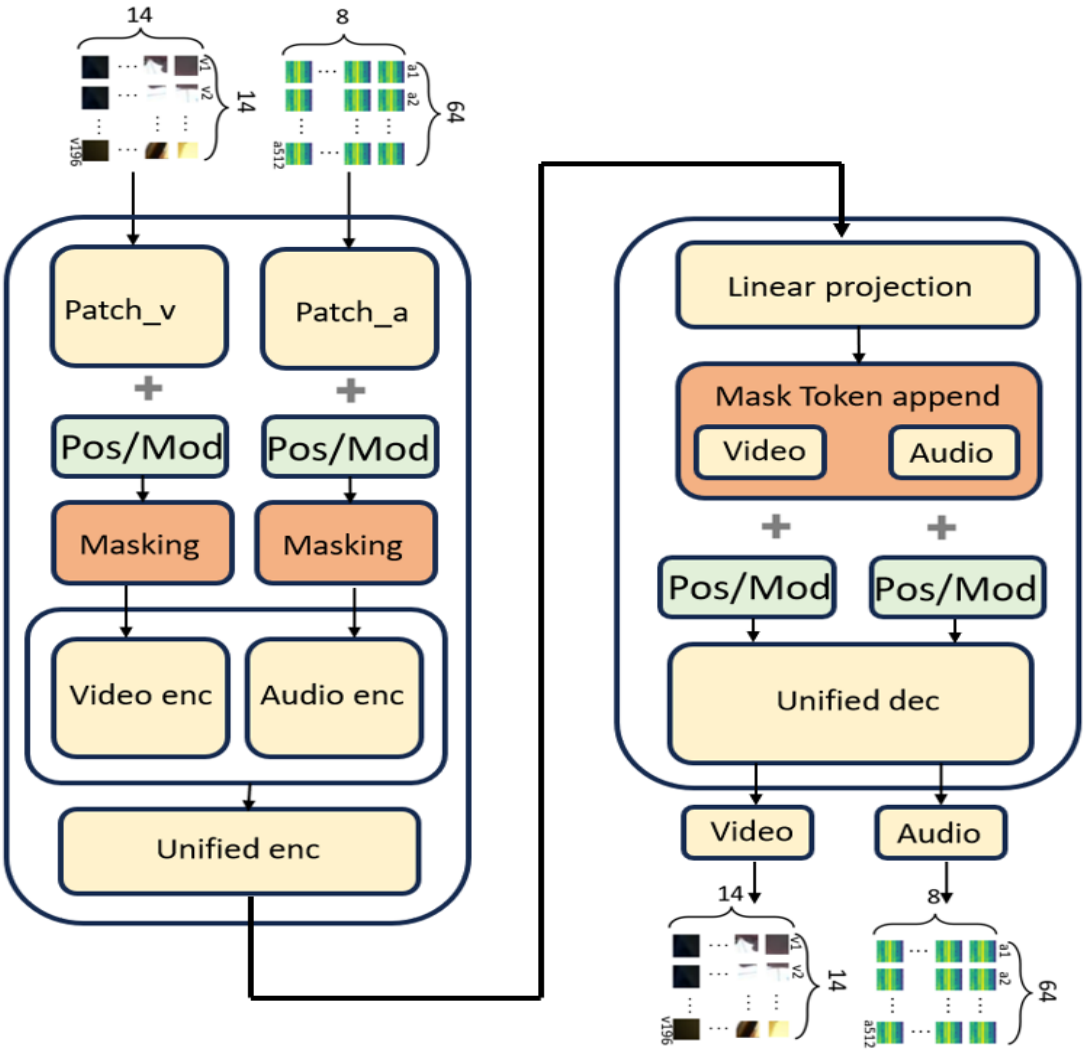

Figure 4.5: A high level look of Dual Modality Masked AutoEncoder. Left side represents all process til input vectors are out of Encoder. Right side represents from post processing of output of Encoder to Decoder til regenerated data inputs are out.

Modality

After we preprocessed audio/video, and patchify the input to get a_i, v_i for each sample i and project them into $\mathbb{R}^{768}$, and add *Sinusoidal Positional Embedding*, we also need to specify modality, since audio differs from video.

Modality embedding is simply a learnable tensor of zeros. We use *nn.Parameter(torch.zeros(1, 1, embed_dim))*.

Masking

As we mentioned/studied *Masked AutoEncoder*[24], we also have applied this feature to the model. The original paper mentions three different types of masking input feature, [random, block, grid] masking with different ratio where 75% gives the best result. We have followed same idea. After preprocessing, projection and adding Positional/modality embedding we randomly drop 75% of patches and shuffle the indices.

$$a_i^{unmasked} = Mask_{0.75}(Proj(a_i) + P_a + M_a)$$
$$v_i^{unmasked} = Mask_{0.75}(Proj(v_i) + P_v + M_v)$$

In the image below [4.6] we see a human interpretable example of video image which is patched, masked and shuffled.

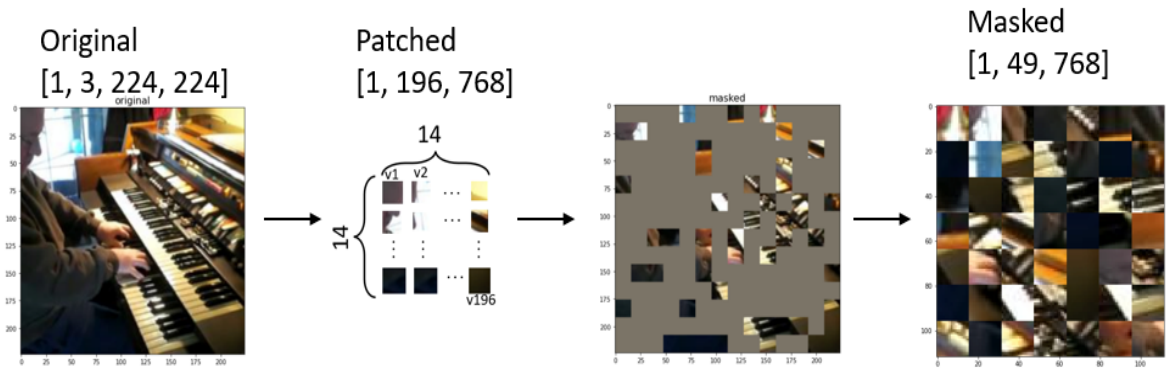

Figure 4.6: Input data example of video

Encoder

We then feed the input $a_i^{unmasked}, v_i^{unmasked}$ to independent audio and video Transformers, $E_a(\cdot), E_v(\cdot)$. These two separate, similar standard Transformers have a depth layer of 12 and width of $\mathbb{R}^{768}$ which is equivalent to input vectors. The output of Transformer blocks are then concatenated and added to a unified Transformers with same parameter as independent Encoders parameter. Image[4.7] below depicts the Transformers Encoder architecture.

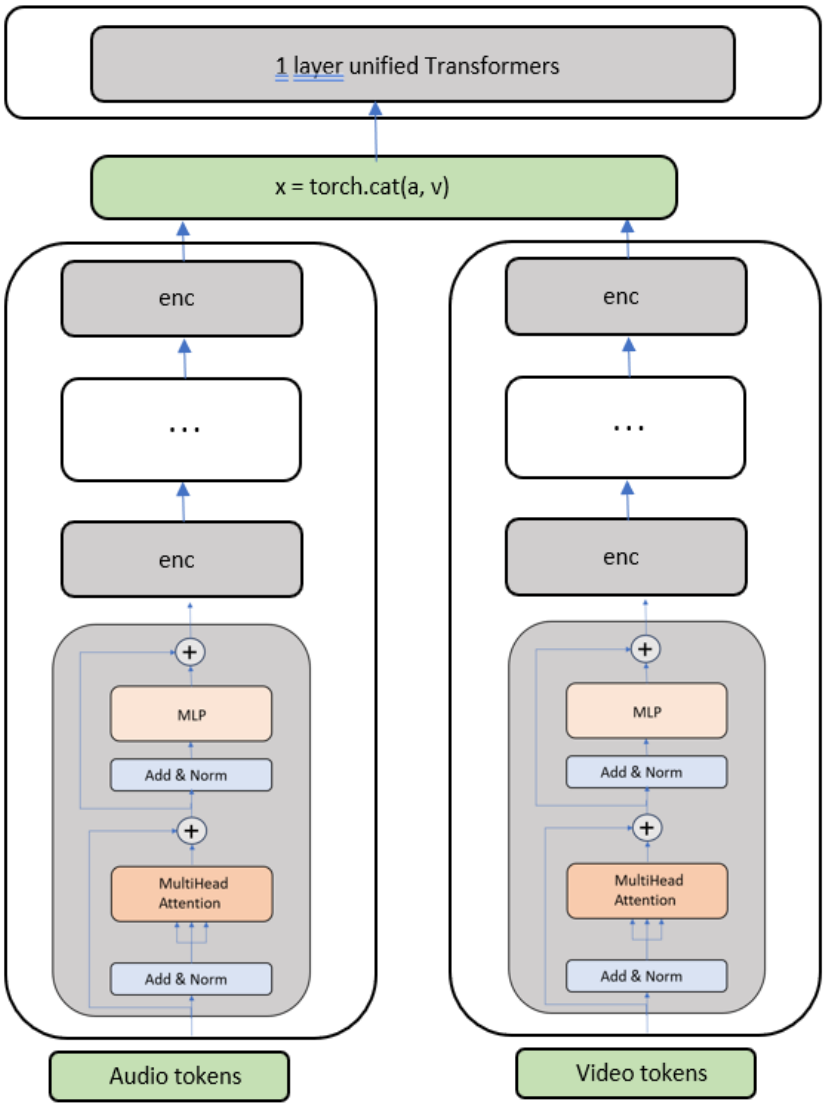

Figure 4.7: Transformer Encoder

Decoder

The latent space of the unified encoder block which includes masked audio/visual feature representation is the input to *Decoder*. We now need to regenerate the audio/visual representation which is now in the form of latent space. The main purpose of the decoder is to set the learnable masked tokens back to its original patch. But we still need to keep track of the position and modality. Therefore the whole process repeats again. So the whole process is to separate audio from video and append the learnable masked tokens. Add position and modality embedding and re-concatenated in order to be fed to a unified 8 blocks of Transformer decoder. Down in the image

[4.8] below we show the whole process.

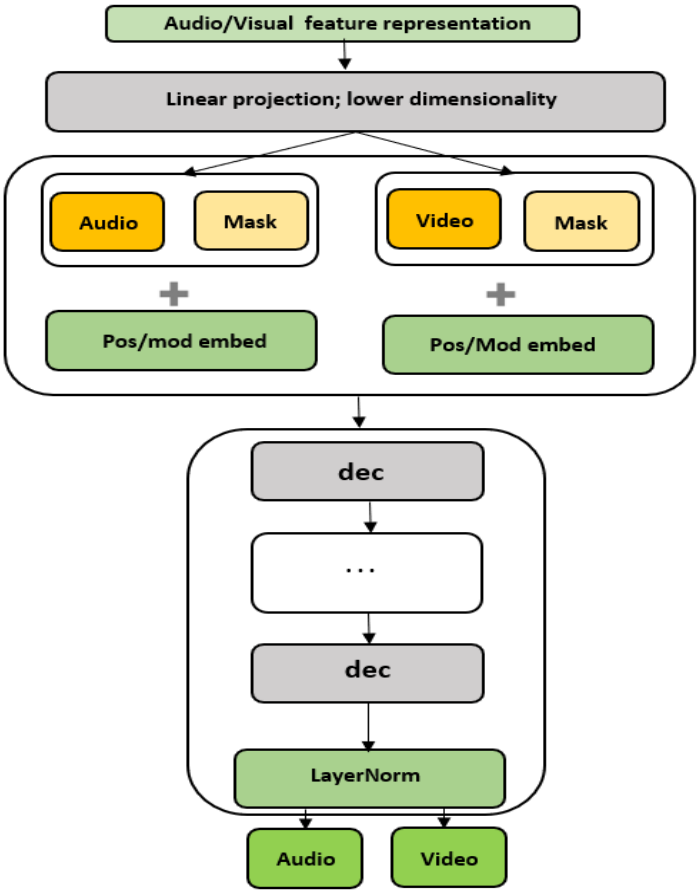

Figure 4.8: Transformer Decoder

4.4.2 Supervised

Since the purpose of the model is to classify the audio instruments, we follow the same idea as Standard Transformers[18] and many other classification paper which was introduced throughout this report. All previously works on classification modeling using Transformers are attaching the Encoder to either a linear classification head or fully connected layer[25, 26, 27, 28, 29] or add a [CLS] token aside input vector[18, 23, 24]. Previously in AST model we presented how [CLS] token works. In this model we attach a linear classification head to classify audio.

Image below[4.9] shows a brief overlook on how the model is structured. We create the same model with same depth/width in attention layer, and remove the Mask Modeling feature. The training procedure is same as before. The added section con-

tains a regular normalization layer and the output is fed to a 1 hidden layer MLP where the input is the average pooled of output vector from Encoder, and the output is the probability distribution of classes.

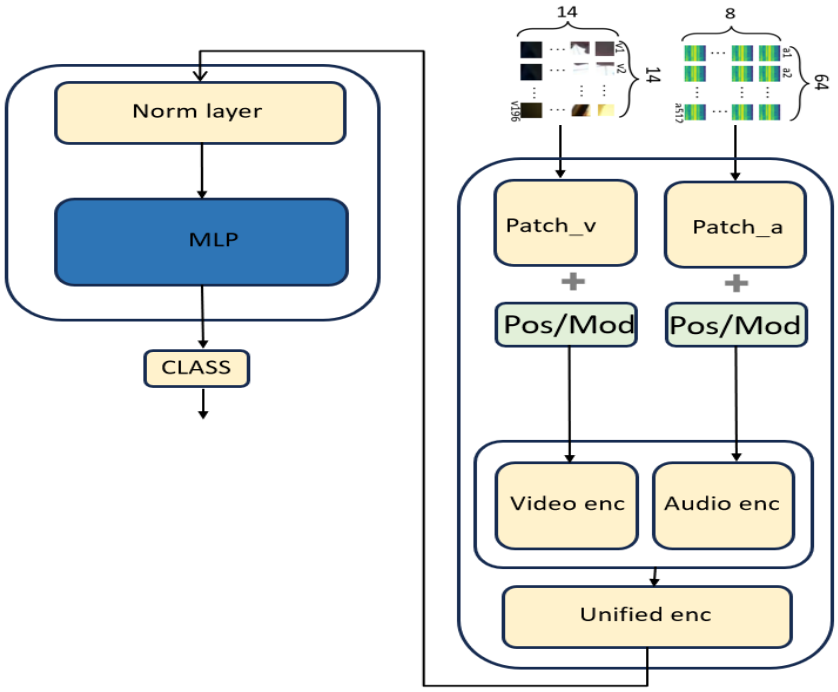

Figure 4.9: A high level look of Dual Modality classification task. Right hand side shows the same Encoder as Self-Supervised, however *Mask Modeling* feature is removed. Left hand side shows a simple linear classification method which outputs the probability distribution of the class.

5 | Experiment

In the previous chapter we described the process of building the architecture of our two models. In this chapter, the results on the data set attained with the final models are described.

Furthermore in the *Result* section we will be demonstrating the result of these two models and expand the result with multiple evaluation metrics for classification and benchmark for assessing the performance of the models in characterizing music content. At the end we briefly discuss how we can ensure that these models predictions align with human perception and judgment.

Setup

The experiment is conducted on two machines provided by ITU.

1. *cn3*, with 2x Intel(R) Xeon(R) Gold 6136 CPU, Cores= 48, GPU= 2x Tesla V100 PCIe 32 GiB,

2. *cn4*, with 2x Intel(R) Xeon(R) Gold 6136 CPU, Cores= 48, GPU= 4x Tesla V100 PCIe 32 GiB, using python 3.7 and timm==0.4.5

Before we start with DualMAE, we should notify about an obstacle we faced with this specific model.

As we previously mentioned this model has two stages. First the model is trained in a self-supervised model. The re-generated audio/visual of this stage demonstrates how well it will perform for the second stage where the model encoder is attached to a linear classification head. Due to our shortage on data this is not possible for us to conduct. In total we have $\sim$ 46k input data. Due to hunger nature of Transformer models, this amount by itself is a very small dataset. This is the main reason we decided to use AudioSet pre-trained weight. This will be elaborated in next chapter *Analysis*. If we halve the dataset into two separate, we are remained with 23k for each stage and this amount of data is not possible based on our experiment. Therefore we train the self-supervised and supervised completely separate.

After many attempts of trial and error we decide with model parameter we describe in the table below [5.1].

Parameter			
	Pre-training	Fine-tuning	AST
Adam Optimizer	weight decay=0.5 betas=(0.95, 0.999)	weight decay=0.5 betas=(0.95, 0.999)	weight decay=0.5 betas=(0.95, 0.999)
epochs	5	5	5
Batch size	12	12	12
Loss Function	-	CE	CE
Weight Averaging	False	True	True
Input Norm Mean	-5.081	-5.081	-5.081
Input Norm STD	4.485	4.485	4.485
Learning Rate	1e-4	1e-4	1e-4
Classification head LR	-	1e-3	1e-3

Table 5.1: Column wise the table shows two sections. The right hand side shows *AST* model parameters, and left *DualMAE* model parameters.

5.1 Result

To evaluate the performance of both these two models, we run all possible scenarios to observe which model will perform better based on loading the model with different init-weight.

We run two classification models with 5 epochs and 12 batches with standard audio/visual preprocessing, and training procedure that was explained earlier. The table[5.3] below shows the accuracy of 6 different scenarios with model weight initialization.

The result shows the winner of the scenarios where a dual modality Transformers have the highest score of 82.41% when the model is loaded with a pre-trained weight based on AudioSet. However the same model competes well with an accuracy of 81.34% with audio only modality, following *AST* model which is based on *DeiT* with an accuracy of 79.14%.

We should as well mention that the reason we write AudioSet + Imagenet is that the model trained on AudioSet dataset has been loaded with ImageNet 1k pretrained weight as we previously mentioned that we noticed that researchers start their starting point with ImageNet 1k pretrained weight.

	Init-weight	Acc
AST	ImageNet	69.68
AST	AudioSet + ImageNet	79.14
DualMAE(audio)	ImageNet	66.36
DualMAE(audio)	AudioSet + ImageNet	81.34
DualMAE(aud/vid)	No init weight	52.61
DualMAE(aud/vid)	ImageNet	68.58
DualMAE(aud/vid)	AudioSet + ImageNet	**82.41**

Table 5.2: Accuracy of different models with different init-weight

From table above we clear our mind that dual modality does affect the result to some degree, and we show in image below[6.9] how the model accuracy behaves during these 10 epochs with different model weight initialization.

DualMAE performs its best when we use AudioSet weight init, and its worst performance belongs to no weight init. However fine-tuning the model with AudioSet does not actually arises from starting point, which we believe this can be a sign of overfitting, where the model has already seen the input vector previously. Since the source of AudioSet and VGGSound is same which is Youtube, there's a high likelihood same video has been captured by both sources. This is our assumption for a somewhat linear accuracy line. We also observe a small fluctuations between epoch 5 to 9 in all three models which gives us better assumption to model overfitting. [25, 26, 27] reports an epoch number of [60, 50, 10] respectively on full VGGSound dataset, which we filtered and preprocessed our dataset based on [27]

Other Metrics

Confusion matrix which we see down below[5.1] is one of the best ways to align the model performance with human perception and judgment. We have to give a disclaimer that due to largeness of class numbers we had to cut a small portion of it. The table allows visualization of the performance of the DualMAE model by best accuracy. It displays the counts of true positive, true negative, false positive, and false negative predictions made by the model. Column-wise we have true reference and row-wise we have predicted. Diagonally we see the number of correctly predicted

class which class 0 has value of 572, but model predicted 1 of them as class 1 and 22 as class 2.

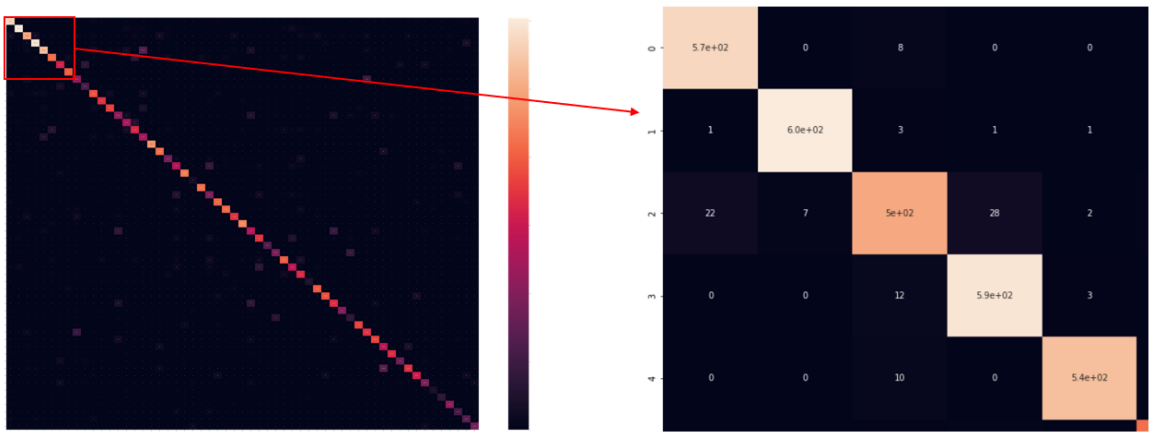

Figure 5.1: Confusion Matrix of DualMAE. The best result of the model has been shown here

Precision, recall and F1-score are also common evaluation metrics used in classification tasks to assess the performance of the model. They are typically calculated based on the confusion matrix generated by comparing the predicted labels with the true labels. We generate the table with the same best result of our DualMAE model.

Precision measures the proportion of true positive predictions among all positive predictions made by the model. Precision indicates how many of the instances predicted as positive are actually positive. A high precision value suggests that the model is good at avoiding false positives. A low precision means the model has a high number of False positive. A good example of model is not well to predict is, *washboard* where the number of occurrence in dataset was low. We had 199 train/test 'washboard' inputs in total. Therefore the model predicted correct only 32% of the time, in compare to 'bassoon' and 'harp' which we had in total of 1040, 1038 data points in total.

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive}$$

Recall indicates how many of the actual positive instances the model is able to correctly identify. A high recall value suggests that the model is good at capturing

positive instances.

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative}$$

Accuracy by itself is not a good indication of model performance. **F1-score** provides a balance between precision and recall and is particularly useful when the class distribution is imbalanced. It reaches its best value at 1 (perfect precision and recall) and worst at 0. A low score indicates class imbalance issue, where one class is much more frequent than the others. This can cause the model to favor the majority class and perform poorly on the minority class, leading to a low F1 score.

$$f1 - score = \frac{2 * Precision * Recall}{Precision + Recall}$$

and **Support** represents the number of occurrences of each class in the true labels. It indicates how many instances belong to each class in the dataset.

Class	Precision	Recall	f1-score	support
harpsichord	0.88	0.94	0.91	607
bassoon	0.91	0.86	0.89	702
piano	0.75	0.66	0.70	760
harp	0.90	0.87	0.88	683
bass_guitar	0.81	0.63	0.71	845
violin	0.76	0.69	0.72	616
congas	0.55	0.72	0.62	163
acoustic_guitar	0.77	0.72	0.75	535
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...
double_bass	0.68	0.73	0.70	355
cello	0.66	0.66	0.66	501
oboe	0.72	0.65	0.68	286
cornet	0.64	0.58	0.61	467
tabla	0.88	0.91	0.89	485
washboard	0.32	0.67	0.43	45
bagpipes	0.93	0.97	0.95	438
trumpet	0.49	0.58	0.53	332

Table 5.3: Evaluation metrics used in classification

We have been through a very long journey to describe the new methods studied

by researchers to achieve the state-of-the-art. We have represented the final outcome, and here we briefly discuss the quality and reliability of the outcomes.

One way as we previously showed we can use metrics such as confusion matrix, precision, recall, f1-score and eventually accuracy of the model. But how the model can capture and ensure its prediction align with human perception and judgement?

To ensure the reliability of the model outcome we believe that incorporating human feedback and validation is necessary. This involves collecting human annotations and judgments on a subset of data points to evaluate model performance and refine its predictions. This can provide an insights into cases where the model's output deviates from human perception, helping identify biases, errors for further refinement.

6 | Analysis

In this section we will be representing the analysis and the discussion section of our models.

Furthermore we will be elaborating on whether machine learning models can be trained to identify the musical instruments used in a track, contributing to the understanding of musical compositions and potentially facilitating the creation of instrument-specific contents.

AST: Supervised

To analyze the performance of our AST model, we randomly download a short video from Youtube. In the short clip 3 people plays 3 instrument, a cello, a violin and a piano. The same preprocessing we apply and cut 10 second of the video, and we extract the audio, since AST takes only audio. We feed the model the new not seen audio.

You can have access to the Google Colab notebook using this link.

Predicted	Accuracy
cello:	0.9993
violin:	0.9984
double_bass:	0.9099
oboe:	0.8731
erhu:	0.7874

Table 6.1: AST accuracy on random Youtube music video

6.0.1 DualMAE: Self-Supervised

We start our analysis with our *Self-Supervised* modeling first. Image 6.1, 6.3 shows the predicted image frame and audio spectrogram respectively. We compare our result figure [6.1] with the original paper [24] figure [6.2] as well. Despite the fact that our predicted image is far way more blurred in compare to the original paper's work, but we see a successful prediction on the masked area, where the masked area are fed as zero vectors to the encoder, however the masked learnable tokens are predicted correct. Each patches of the frames have been set correctly with right position. Our model has predicted the keyboard in the image way better than sides where the keyboard for example which has a texture of black and white better in compare to other areas.

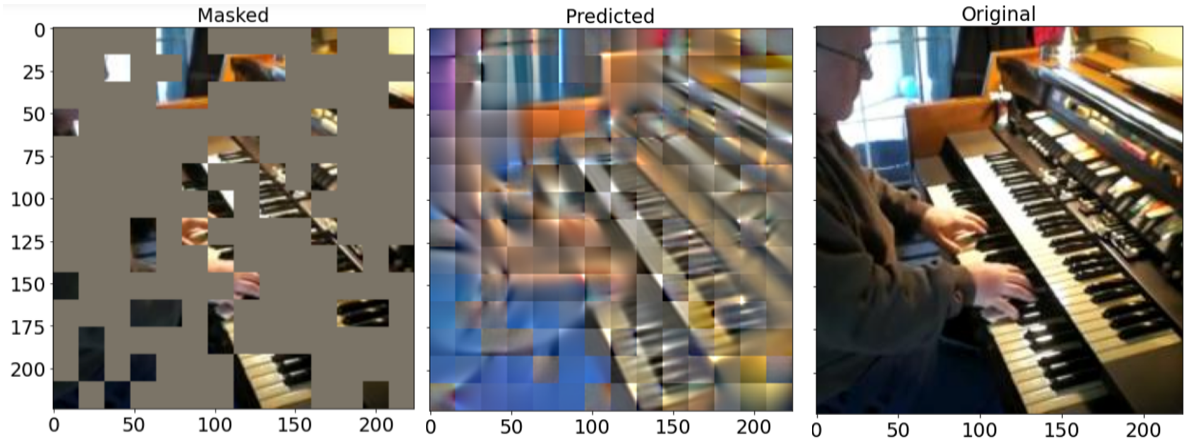

Figure 6.1: Predicted video frame by DualMAE Self-Supervised method

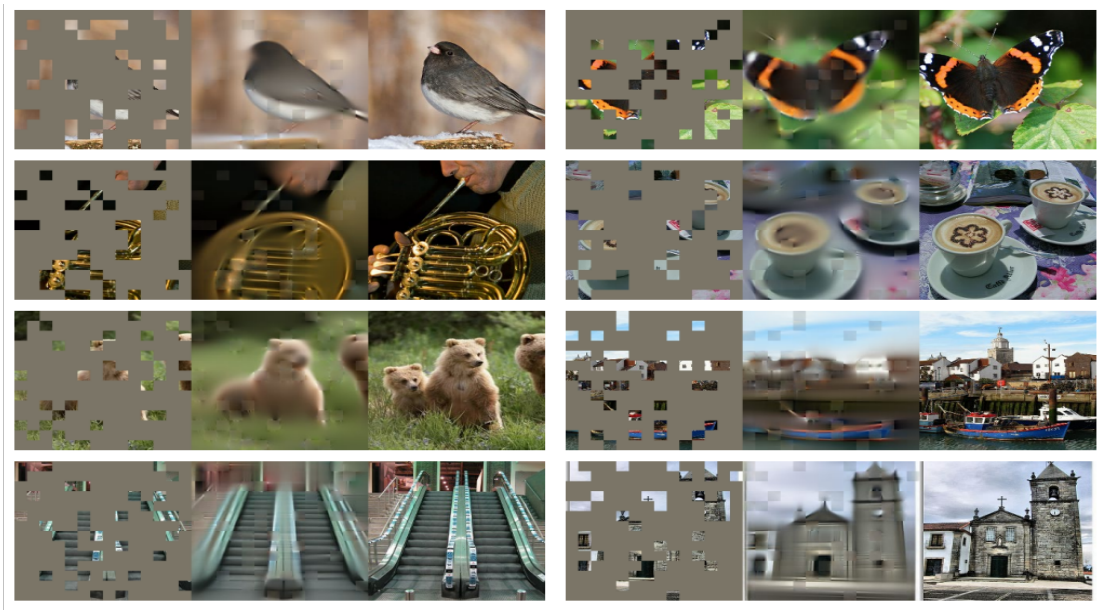

Figure 6.2: Image taken from he et al. [24]; "Masked Autoencoders Are Scalable Vision Learners ". The image shows different evaluated image frames from ImageNet1K. Left[masked], middle[predicted], right[original]

The image below[6.3] shows the original/predicted audio spectrograms. Looking at the images we notice the same behaviour as video frames, that the model successfully regenerate the frequency pattern in y-axis, and same in time, in x-axis. However the color density of the predicted spectrograms does not show a clear audio.

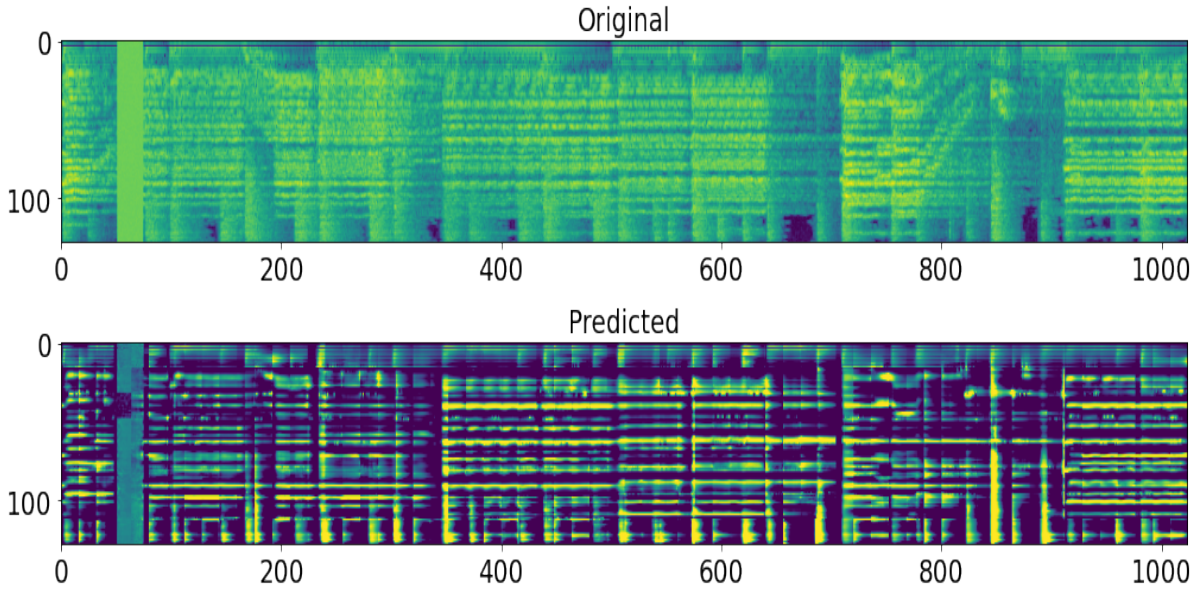

Figure 6.3: Predicted aduio Spectrograms by Self-Supervised method

6.0.2 DualMAE: Supervised

We analyze our DualMAE model by first looking at the loss behaviour over batches. The first figure [6.4] shows the model DualMAE loaded with AudioSet pretrained weight. The image shows loss value in y-axis, and averaged loss values of 12 sample batch in x-axis on our VGG dataset. Hyperparameters are set to the parameter table [5.1] as this setup results the best performance after our multiple trial. We see in the image that both train and test fluctuates, however in the last epoch [right] the fluctuation decreases. The learning curve reaches a stable or optimal state. Ideally, we want the learning curve to converge to a point where further training doesn't lead to overfitting. On the other hand the test loss keeps fluctuating the entire epochs where the maximum loss value reaches 4.0.

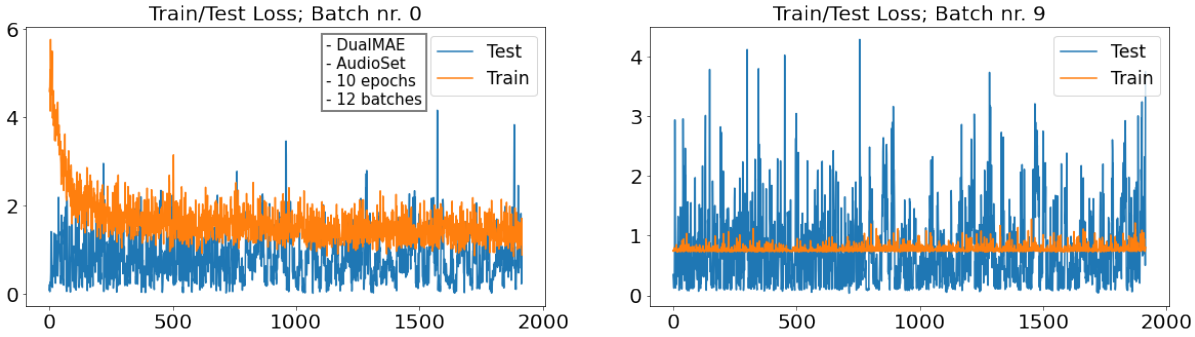

Figure 6.4: DualMAE loss procedure using AudioSet weight initializing

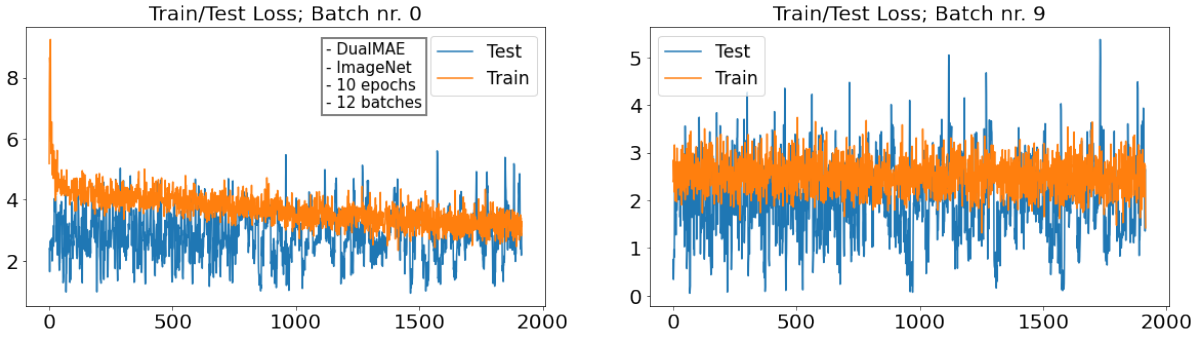

Figure 6.5: DualMAE loss procedure using ImageNet weight initializing

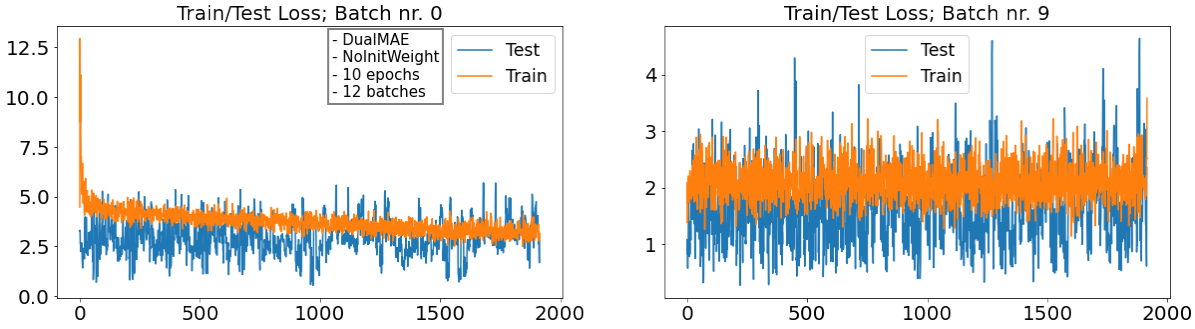

Figure 6.6: DualMAE loss procedure without weight initializing

We believe there are several reasons that causes fluctuations in training/test loss over epochs. The first one is batch size which determines how many samples it takes for the to model update parameters. In the next page we show in the image [6.7] how the number of batch size affect fluctuations. We are in fact limited to the batch size, as the machines provided by ITU, which is the best available machine we have access, runs out of memory if we precede training more than 12 samples in one batch while Gong et al. [27] reports the best result is batch size of 48 samples. In the image below we experimented how batch size can affect the training. We run 1 epoch over whole training dataset with different batch size of [1, 2, 6, 12, 18]. In the image[6.7]

below Each point represents the loss value of the mean of total number of batch. That means the average of batch x amount of audio/visual. We see the higher number of batch size, the lower value of loss function. We report that more than 12 samples a batch gives us constantly Out-of-Memory error constantly and leave us the choice of 12 samples a batch.

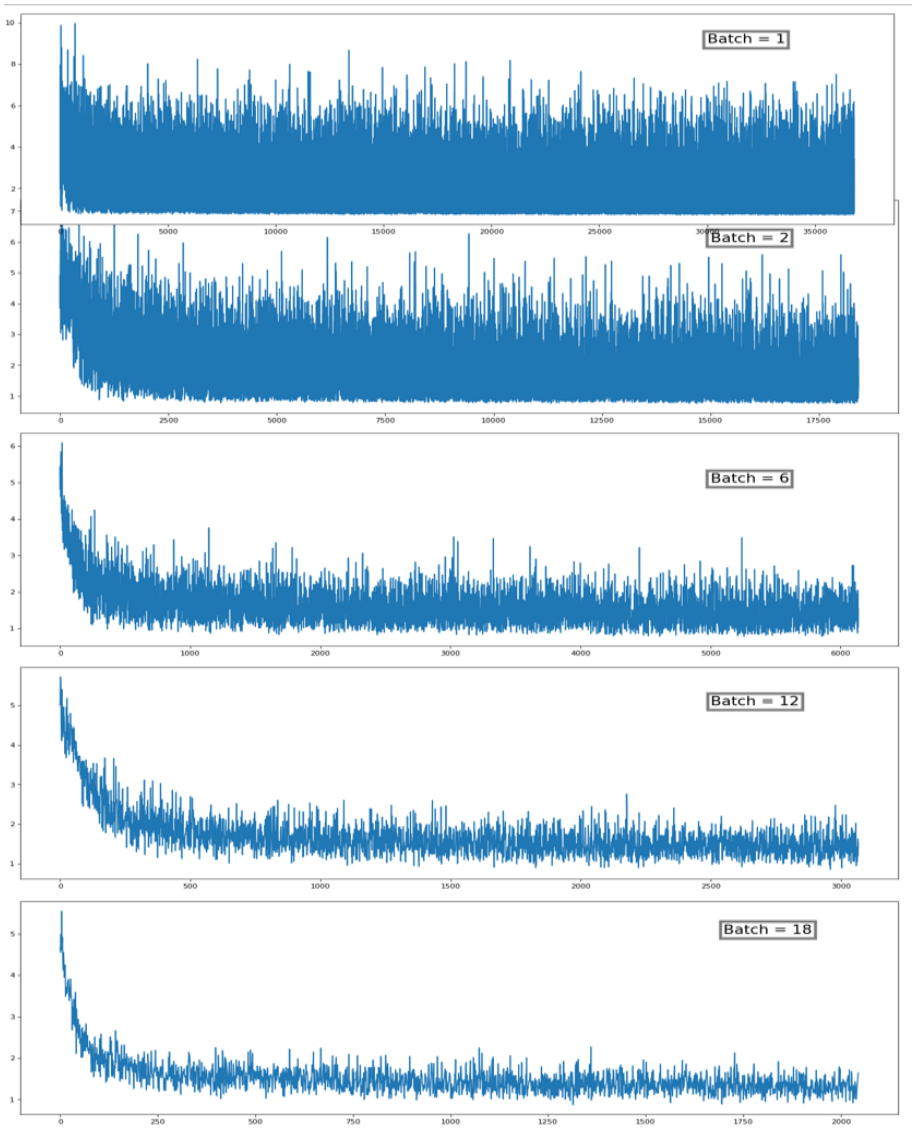

Figure 6.7: Choice of batch size; y shows loss value, and x shows the amount of dataset

Another reason we believe causes fluctuations and poor performance in general is the model is indeed a very large network. Total trainable parameter number is 164.546 million while our training data is $\sim$ 37k for training, and $\sim$ 9k for test. To put this into perspective the model tries to optimize weights to find local minimum in

164K dimension based on such small dataset.

We believe that the poor performance is also due to our instability to the number of data input for each class causes such fluctuation, resulting to model has not learned as much as it can from the training data and has not reached to its best performance. Thus no matter how many epochs we run. Therefore as we saw in table [5.3] the model has 0.91 f1-score on some classes like harpsichord which has 1015 data input in compare to washboard with f1-score 0.43 with 199 data input. In the image below we can see the distribution of classes.

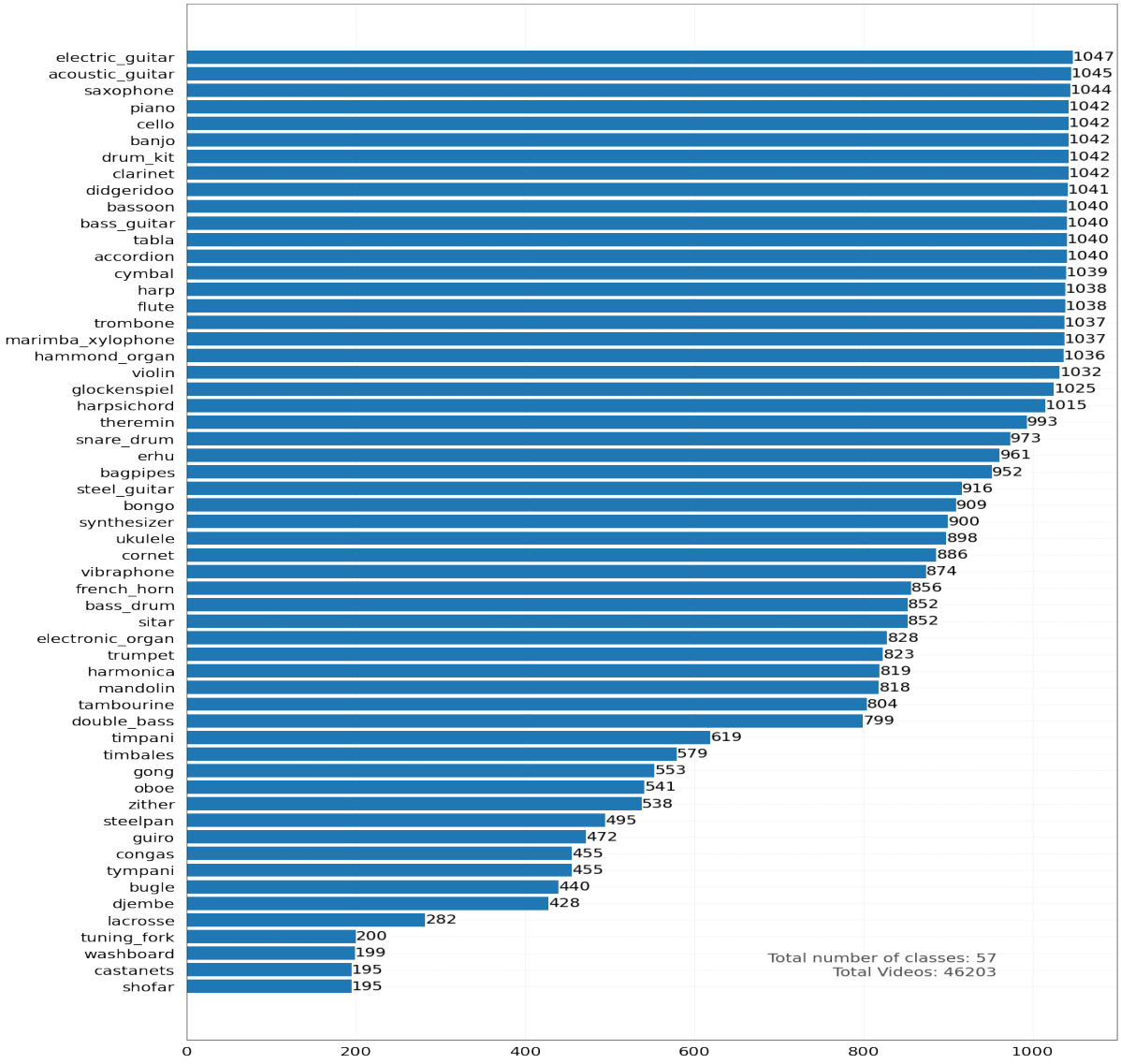

Figure 6.8: VGGSound Classes

It is worth mentioning that the focused papers relating to DualMAE [25, 26, 27,

28, 29] train their model on whole VGGSound dataset which is 200k while we use approximately 18.5% of the dataset which is musical instruments.

We report figure [6.5] when model is loaded with ImageNet 1k has faster convergence in compare to model with AudioSet. However the image on right hand side shows slight changes which we still consider that as overfitting. Surprising when the model has Xavier weight initialization has a very small and narrow loss for training and test, but still after 10 epochs the result turn back to overfitting as well.

Another reason that can contribute to our not satisfactory result is our raw data. We investigate and realize there are anomalies in our raw data. Since the data comes from YouTube some input data has an introduction with no sound and the video frame has text on it.

for further analysis we visualize the DualMAE accuracy for 10 epochs while the model has been loaded with different pretrained weights. Figure below [6.9] shows the result.

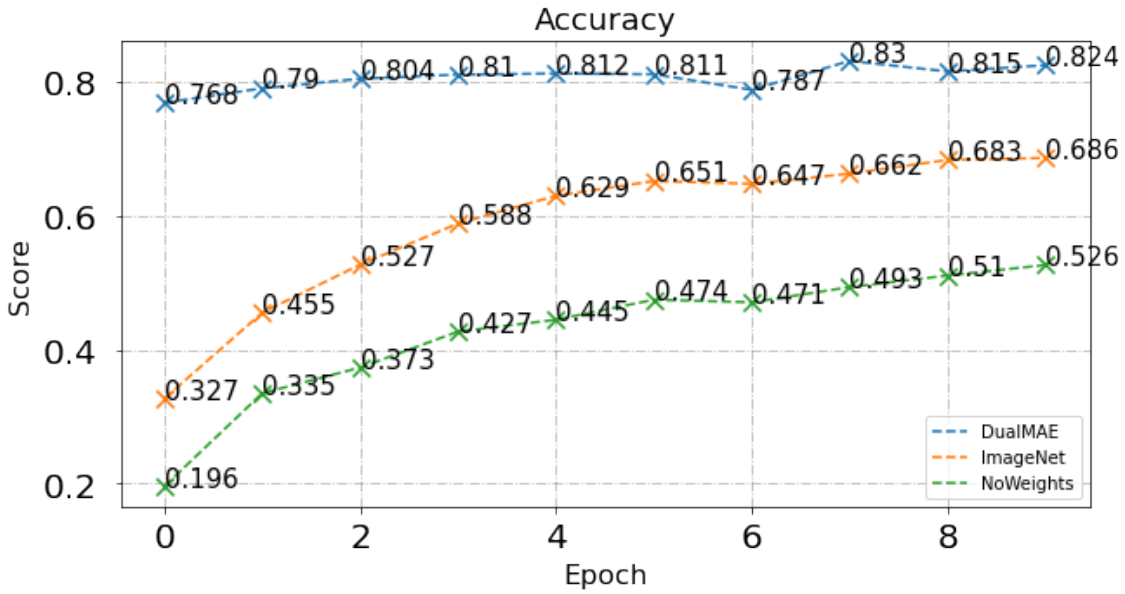

Figure 6.9: DualMAE accuracy based on weight initialization

We see that the learning curve when the model is not loaded with AudioSet pretrained weight starts indeed fast up until 5th epoch, but in the next epoch all three runs drop to some degree. We believe that this is a sign of overfitting and more than 5

epoch is not necessary. We also observe the learning curve for DualMAE when loaded with AudioSet pretrained weight[Blue dashed] seems to be more linear in compare to other two [Orange, Green dashed]. We put our assumption that since AudioSet and VGGSound has same source which is YouTube, there is likelihood that the model has seen very similar contents before. But we can not provide any certain answer to this, since neither of these two dataset provide any information on that.

6.0.3 Masking

We study other papers that uses *Masked Modeling* features.[25] reports a ratio of 80% for both audio/visual, [26] reports 50% for audio, and 90% for video, and at the end [27] reports a casual percentage as original work[24] which is 75% for both audio/visual. We load the model with our two different initial weights and set the masking ratio parameter to different values to observe loss values behaviour. All masking methods are set to default which is random drop of patched indices in an unstructured way. The table 6.2 shows the averaged loss values of the entire 1 last epoch. The value belongs to test dataset and the model is trained/tested for 5 epochs. Models are ran with the decided parameter from table [5.1]. We use both modality audio/visual to observe the averaged loss value of each data type, including the total loss value.

	Avg aud	Avg vid	Avg total
0.05%	3.65	1.84	5.57
0.25%	3.76	2.08	5.91
0.50%	4.12	2.74	6.93
0.75%	4.85	4.03	7.42
0.95%	5.06	4.73	7.89

	Avg aud	Avg vid	Avg total
0.05%	3.41	1.78	5.23
0.25%	3.53	2.05	5.61
0.50%	3.84	2.69	6.57
0.75%	4.48	3.95	8.48
0.95%	6.67	6.90	8.87

Table 6.2: Table on the left hand side has pre-weight of *ImageNet*, and the right hand side is set to *AudioSet*. In the table we have the mean of all loss values of pre-training method. Masked ratio are set to different values.

We do not observe an effective difference in loading the model with different initial weight, as the average audio, video and total loss values are very close. We saw earlier from loss value visualisation from figures [6.4,6.5, 6.6] that no matter what pretrained weight we use, as at the end the model's test loss were the same with same fluctua-

tion behaviour. However we see an increasing loss value if the mask ratio increases. This means the masked token in the *Self-supervised[Decoder]* should learn to predict its relative patch given x ratio unmasked. We trained and tested the model for 5 epochs to make sure the model has learnt to adjust weight vectors in back-propagation.

Furthermore we did apply techniques such as k-fold cross validation and dropout in MLP head and adding more layers with activation functions, but unfortunately they were not of some help. And we concluded that the main issue comes from our small dataset.

Computational report

In the image[6.10] below we observed that as the original paper stated, computational and time wise, the model is indeed faster if the mask ratio is higher. for 1 single full epoch training and evaluation if the mask ratio is set to 0.05%, it takes $\sim$ 25 minutes, while if the ratio is set to 0.95%, it takes $\sim$ 16 minutes.

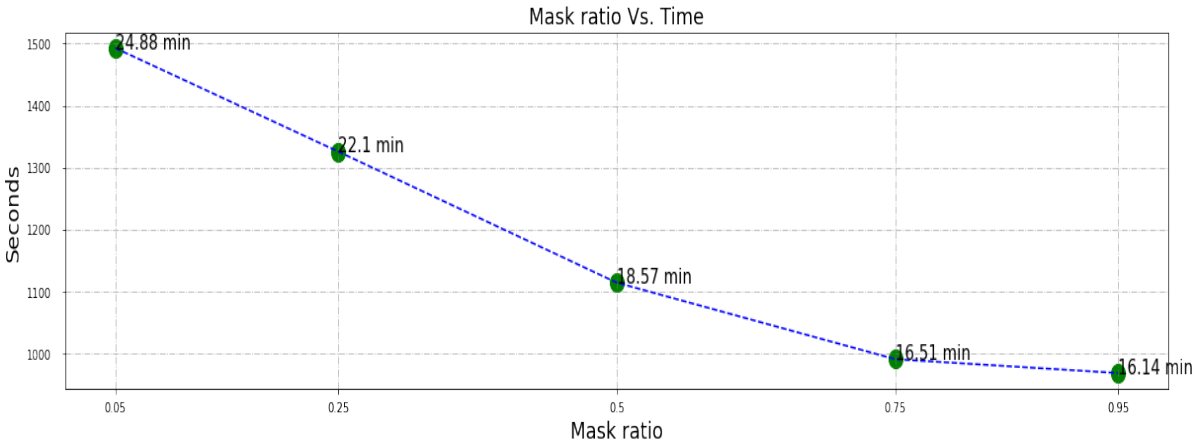

Figure 6.10: Mask ratio Vs. Time

6.1 Discussion

We believe that Transformers have emerged as leading models in computer vision, achieving state-of-the-art performance across many tasks such as audio content classification, however ViTs are thought to be more "data hungry" than brains based on pandey et al.[33]. This can as well be seen from Standard Transformers benchmark by

adosovitskiey et al.[18].

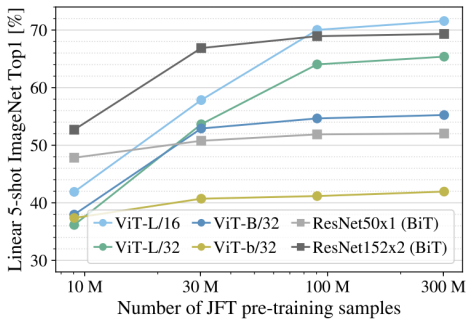

Figure 6.11: "ResNets perform better with smaller pre-training datasets" by [18]

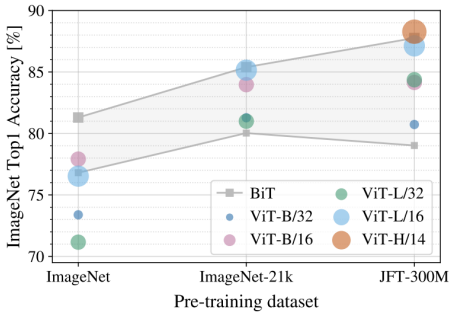

Figure 6.12: "ViT models perform worse than BiT ResNets (shaded area) when pre-trained on small datasets." by [18]

The figures shows two models variation of ViT[Transformers] and ResNet[CNN]. Adosovitskiey et al. report on figure [6.11, 6.12] that Transformers with different number of layers in Encoder performs worse if the model is trained on small dataset, but CNN models performs best. This result reinforces the intuition that the CNN inductive bias is useful for smaller datasets.

Diversity and subjectivity

The utilization of transformer architectures in conjunction with dual modality and masked modeling within this thesis has prompted another discussion on the efficacy and potential implications of such an approach for handling subjectivity and diversity in music classification.

diversity and subjectivity in music classification refer to certain characteristics and challenges associated with the task of categorizing or labeling musical pieces. Music is a highly diverse art form with variations in terms of genres, styles, instruments, tempo, mood, and cultural influences. This diversity makes it challenging to create a universally applicable set of categories or labels for music classification like classical, jazz, rock, pop, hip-hop, electronic, and many other genres exist, each with its own subgenres and variations. Additionally, within a single genre, there can be significant diversity. Subjectivity in music classification recognizes individual preferences, cul-

tural backgrounds, and personal interpretations can influence how people perceive and categorize music, because musical taste is subjective, and what one person considers a specific genre or style might be interpreted differently by someone else. This subjectivity can lead to ambiguity and challenges in defining clear criteria for classification. Two individuals might have differing opinions on whether a particular piece is best categorized as "indie rock" or "alternative rock". Cultural background and personal experiences can shape one's understanding of music genres. Now this makes it challenging for incorporating user feedback and preferences to create/label a globally music classification systems which is a part of classification in general. Therefore it's important to recognize that achieving a perfect, universally agreed-upon music classification system is inherently difficult due to the diversity and subjectivity inherent in musical expression.

Fully connected layer or [CLS] token?

We showed in this project two different methods of how a transformer model can be fine-tuned for classification task. Model AST follows the Standard Transformer by adosovitskiy et al.[18] with a [CLS] token aside input vector same as Bert[23],

$$[x_{class}, x_p^1, ..., x_p^N]$$

where x_p^i are image patches, and model DualMAE which uses a fully connected layer as [25, 26, 27, 28, 29]. Question arises that would there be any benefit on whether one method is better than other? Or is the extra class token important to predict the results? The answer is no. The Standard Transformer authors state that they only wanted to keep the work as close to Bert and Original Transformer. We conclude that there might me dis/advantages in the field of NLP, but it is not important for Computer Vision.

Code

Our code is inspired by many different github repositories of papers we mention during this thesis, however mostly we followed from Gong et al. github repository.

Data

We could not give ourselves the permission to download *AudioSet* dataset from YouTube, and their website does not provide a method to do so. Downloading content from YouTube without explicit permission from the content owner or without using the provided download options on the YouTube platform can violate YouTube's terms of service. Additionally, it may infringe on the copyright of the content creator, as videos on YouTube are typically protected by copyright law. YouTube's terms of service state that you should not download content unless a download button or link is clearly provided by YouTube. Moreover, downloading copyrighted content without permission may lead to legal consequences, as it infringes on the rights of the content creator. The only available dataset was VGGSound which contained audio instrument content which is in fact contents from YouTube, but a direct download link has been provided by VGGSound website. Therefore we were short on data.

Imbalanced DataSet

Another big issue we faced during this thesis was an imbalanced dataset, which can make the model perform well in favor of majorities. An easy way to understand the issue of imbalanced dataset is to have 100 testing examples. 82 of them are from class A and the other 18 are from class B. The model can achieve 82% accuracy simply by classifying A each time! Such is why the F1 score is so important in evaluating models on unbalanced datasets. An improvement can be balancing the dataset which in our case was not possible.

7 | Conclusions

To conclude, this thesis has explored the recent methods that claims the state-of-the-art for general classification, and we applied these methods to characterize the content of music, offering a comprehensive investigation into the use of Transformer models with the benefits of using dual modality by combining audio and video. Through experimentation and analysis, it has been demonstrated that leveraging transformers for dual modality tasks presents a promising avenue for advancing various fields such as characterizing music contents. We experimented and demonstrated a new approach, Masked Modeling which optimizes Transformer models for better accuracy and time consumption.

Considering the limitation of our data and processing machine that led to trouble connecting the Self-supervised method to the Supervised method, we demonstrated how the recent trends of state-of-the-art for music content classification takes place.

The finding of this research underscore the effectiveness of Transformer models in capturing complex interdependencies between audio and video while masking data inputs leading to enhanced performance of Transformers. By harnessing the power of attention mechanisms inherent in transformers, our experimented approach has showcased remarkable capabilities in understanding and synthesizing information from multiple modalities simultaneously.

In summary, we hope the insights from this study provide valuable implications for future research for the content of audio and musical analysis.

7.1 Future work

The results of the experiments in this thesis point it out to several directions and options for further experimentation with the network architecture:

This thesis suffered so much because of having small in-domain dataset, it could be best to build a CNN based model to compare both aside, but due to heaviness of structuring Transformers based models we did not manage to do so.

Out-domain dataset and investigate whether audio that is not related to music can enhance the performance of audio/music content classification. Since we used AudioSet dataset which has a big range of classes.

Contrastive learning is another technique that aims to learn representations of data by maximizing the distance between similar samples and minimizing it between dissimilar samples. This technique has also been studied by [25, 27]. This could be another reason to expand our work for further analysis.

There has been many slight changes in different papers either as been mentioned in this thesis or not. Even by the time of writing this report, there has been published even another paper which claims better performance in image classification based on Transformer models.

Bibliography

[1] A. Agostinelli, T. I. Denk, Z. Borsos, J. Engel, M. Verzetti, A. Caillon, Q. Huang, A. Jansen, A. Roberts, M. Tagliasacchi *et al.*, "Musiclm: Generating music from text," *arXiv preprint arXiv:2301.11325*, 2023.

[2] X. Gu, L. Ou, W. Zeng, J. Zhang, N. Wong, and Y. Wang, "Automatic lyric transcription and automatic music transcription from multimodal singing," *ACM Transactions on Multimedia Computing, Communications and Applications*.

[3] Z. Cataltepe, Y. Yaslan, and A. Sonmez, "Music genre classification using midi and audio features," *EURASIP Journal on Advances in Signal Processing*, vol. 2007, pp. 1–8, 2007.

[4] J. Marques and P. J. Moreno, "A study of musical instrument classification using gaussian mixture models and support vector machines," *Cambridge Research Laboratory Technical Report Series CRL*, vol. 4, p. 143, 1999.

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

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[7] A. Ghildiyal, K. Singh, and S. Sharma, "Music genre classification using machine learning," in *2020 4th international conference on electronics, communication and aerospace technology (ICECA)*. IEEE, 2020, pp. 1368–1372.

[8] E. A. Retta, R. Sutcliffe, E. Almekhlafi, Y. K. Enku, E. Alemu, T. D. Gemechu, M. A. Berwo, M. Mhamed, and J. Feng, "Kiñit classification in ethiopian chants, azmaris and modern music: A new dataset and cnn benchmark," *Plos one*, vol. 18, no. 4, p. e0284560, 2023.

[9] P. Beckmann, M. Kegler, H. Saltini, and M. Cernak, "Speech-vgg: A deep feature extractor for speech processing," *arXiv preprint arXiv:1910.09909*, 2019.

[10] B. W. Schuller, A. Batliner, S. Amiriparian, A. Barnhill, M. Gerczuk, A. Triantafyllopoulos, A. E. Baird, P. Tzirakis, C. Gagne, A. S. Cowen *et al.*, "The acm multimedia 2023 computational paralinguistics challenge: Emotion share & requests," in *Proceedings of the 31st ACM International Conference on Multimedia*, 2023, pp. 9635–9639.

[11] S. Dieleman and B. Schrauwen, "End-to-end learning for music audio," *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6964–6968, 2014. [Online]. Available: https://api.semanticscholar.org/CorpusID:13094557

[12] J. Lee, J. Park, K. L. Kim, and J. Nam, "Samplecnn: End-to-end deep convolutional neural networks using very small filters for music classification," *Applied Sciences*, vol. 8, p. 150, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:3286115

[13] Y. LeCun, Y. Bengio *et al.*, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, 1995.

[14] Q. Kong, Y. Cao, T. Iqbal, Y. Wang, W. Wang, and M. D. Plumbley, "Panns: Large-scale pretrained audio neural networks for audio pattern recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 2880–2894, 2020.

[15] Y. Gong, Y.-A. Chung, and J. Glass, "Psla: Improving audio tagging with pretraining, sampling, labeling, and aggregation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 3292–3306, 2021.

[16] O. Rybakov, N. Kononenko, N. Subrahmanya, M. Visontai, and S. Laurenzo, "Streaming keyword spotting on mobile devices," *arXiv preprint arXiv:2005.06720*, 2020.

[17] P. Li, Y. Song, I. V. McLoughlin, W. Guo, and L.-R. Dai, "An attention pooling based representation learning method for speech emotion recognition," 2018.

[18] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[19] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, "Training data-efficient image transformers & distillation through attention," in *International conference on machine learning*. PMLR, 2021, pp. 10 347–10 357.

[20] L. Yuan, Y. Chen, T. Wang, W. Yu, Y. Shi, Z.-H. Jiang, F. E. Tay, J. Feng, and S. Yan, "Tokens-to-token vit: Training vision transformers from scratch on imagenet," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 558–567.

[21] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[22] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *European conference on computer vision*. Springer, 2020, pp. 213–229.

[23] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[24] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 16 000–16 009.

[25] P.-Y. Huang, V. Sharma, H. Xu, C. Ryali, H. Fan, Y. Li, S.-W. Li, G. Ghosh, J. Malik, and C. Feichtenhofer, "Mavil: Masked audio-video learners," *arXiv preprint arXiv:2212.08071*, 2022.

[26] M.-I. Georgescu, E. Fonseca, R. T. Ionescu, M. Lucic, C. Schmid, and A. Arnab, "Audiovisual masked autoencoders," *arXiv preprint arXiv:2212.05922*, 2022.

[27] Y. Gong, A. Rouditchenko, A. H. Liu, D. Harwath, L. Karlinsky, H. Kuehne, and J. R. Glass, "Contrastive audio-visual masked autoencoder," in *The Eleventh International Conference on Learning Representations*, 2022.

[28] S. Chen, Y. Wu, C. Wang, S. Liu, D. Tompkins, Z. Chen, and F. Wei, "Beats: Audio pre-training with acoustic tokenizers," *arXiv preprint arXiv:2212.09058*, 2022.

[29] Y. Gong, Y.-A. Chung, and J. Glass, "Ast: Audio spectrogram transformer," *arXiv preprint arXiv:2104.01778*, 2021.

[30] S. Takase and N. Okazaki, "Positional encoding to control output sequence length," *arXiv preprint arXiv:1904.07418*, 2019.

[31] H. Chen, W. Xie, A. Vedaldi, and A. Zisserman, "Vggsound: A large-scale audio-visual dataset," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 721–725.

[32] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, "Training data-efficient image transformers & distillation through attention," in *International conference on machine learning*. PMLR, 2021, pp. 10 347–10 357.

[33] L. Pandey, S. Wood, and J. Wood, "Are vision transformers more data hungry than newborn visual systems?" *Advances in Neural Information Processing Systems*, vol. 36, 2024.