Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мультимедійних систем факультету інформатики

EXPLORATION OF MULTIMODAL APPROACHES IN IMAGE-TO-AUDIO SYNTHESIS

**Текстова частина до курсової роботи
за спеціальністю «Комп'ютерні науки» 122**

Керівник кваліфікаційної роботи

старший викладач Кузьменко Д.О.

_____

(підпис)

«____» _____ 2024 р.

Виконав студент

Беймук В.О.

«____» _____ 2024 р.

Київ 2024

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра мультимедійних систем факультету інформатики

ЗАТВЕРДЖУЮ
зав. кафедри мультимедійних систем,
доцент, к.ф-м.н.
Жежерун О.П.
_____
(підпис)
«_____» _____ 2024 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсову роботу

студента 3-го курсу факультету інформатики Беймука Володимира Олеговича

ТЕМА: Exploration of multimodal approaches in image-to-audio synthesis

Зміст ТЧ до курсової роботи:

Індивідуальне завдання

Календарний план

Анотація

Зміст

Вступ

Попередні відомості

Експерименти

Результати

Висновки

Список літератури

Дата видачі «___» _____ 2024 р. Керівник Кузьменко Д.О.

_____
(підпис)

Завдання отримав _____
(підпис)

Тема: Exploration of multimodal approaches in image-to-audio synthesis

Календарний план виконання роботи:

№	Назва етапу курсової роботи	Термін виконання етапу	Примітка
1.	Визначення теми кваліфікаційної роботи.	Жовтень 2023	
2.	Отримання завдання на кваліфікаційну роботу.	Жовтень 2023	
3.	Огляд літератури.	Листопад – Грудень 2024	
4.	Тестування наявних рішень.	Січень 2024	
5.	Вибір технологій для модифікації рішень.	Січень 2024	
6.	Експерименти з модифікацією рішень.	Січень 2024 – Травень 2024	
7.	Написання текстової частини.	Травень 2024	
8.	Захист кваліфікаційної роботи.	Травень 2024	

Студент _____

Керівник _____ «___» _____ 2024 р.

АНОТАЦІЯ

Це дослідження зосереджено на вивченні різних підходів до генерації аудіо з зображень, розгляді їхньої поведінки та вдосконалюючи їхні можливості за допомогою перевірки окремих гіпотез. Воно включає аналіз та порівняння сучасних моделей, які використовуються в цій галузі. Значна увага приділяється моделям SpecVQGAN та IM2WAV, які демонструють новітні методи з використанням глибинного навчання для синтезу точного та послідовного аудіопотоку. Також розглядаються техніки регуляризації, і аналіз їхнього впливу на якість моделей. Важливість дослідження зумовлена потенційним застосуванням у таких областях, як підтримка людей із вадами зору, віртуальна реальність, освітні інструменти та виробництво звукових ефектів у фільмах чи відеоіграх.

Ключові слова: генерація аудіо з зображень, глибоке навчання, генеративні моделі, трансфореми, техніки регуляризації.

TABLE OF CONTENTS

1 INTRODUCTION

Generative models have found much of use in the field of Artificial Intelligence. They allow the generation of high-quality images [1-10] and videos [11-15], as well as music [16-20] and audios [21-23]. The architectures used for these models can be categorized into several methodologies, such as Generative Adversarial Networks (GANs) [24], Variational Autoencoders (VAEs) [25] and autoregressive models like Transformers [26].

Previous works explore different problems in the sound generation domain such as calculating the correspondence between audio and video [27-28], localization and separation of audio sources in videos [29-31], and video-to-audio or image-to-audio generation [32-37].

Our work focuses on exploring different approaches to image-to-audio generation, investigating their behavior, and improving their capabilities by testing certain hypotheses which we describe later in this section, and seeing if they can be applied on image-to-audio domain.

Sound generation is a complex task in machine learning, given the challenges that come with synthesizing accurate and coherent audio. Still, it is an important topic with a wide range of possible practical applications. In recent years, significant progress has been made in this area and there exist multiple approaches to this problem:

Ning et al. [32] use three key components. First, a Convolutional Neural Network (CNN) [46] with dilated convolutions is used for image feature extraction. Next, an autoencoder is used to encode audio features. Finally, a cross-modal mapping network to map visual features to audio features. To generate sound, the authors

use 1D convolution kernels with dilation to capture long-range phoneme relationships.

Owens et al. [33] were one of the first works to synthesize instrument sounds. They did this by using a type of Recurrent Neural Network (RNN) called LSTM [39] and a variant of CNN called AlexNet [40] to generate waveforms. Later their approach was adapted by Chen et al. [34] where the authors synthesized images from audio and vice versa in solo musical performances with two Generative Adversarial Networks (GANs) generating spectrograms. This approach was later improved by Tan et al. [35] with using integrated self-attention [26] mechanisms used in Transformers. Iashin et al. [36] replaced the two GANs with a single transformer-based language model trained to sample from a Vector Quantized Variational Autoencoder (VQ-VAE) codebook. Finally, Sheffer et al. [37] replaced a single transformer with two transformers to sample both low- and high-resolution audio tokens and used a Contrastive Language-Image Pre-training (CLIP) [38] model instead of traditional CNNs used before.

The SpecVQGAN model by Iashin et al. [36], and the IM2WAV system by Sheffer et al. [37] represent two state-of-the-art approaches. We will focus on comparing them further in the next sections.

This research focuses on exploring and comparing these different approaches, as well as examining a couple of well-documented observations that hold true for other areas of deep learning but have not been previously explored in this domain.

For instance, it is well known that complex transformer models are better at capturing difficult patterns and aligning their predictions with the target, although they often generalize poorer to unseen data and tend to overfit when trained on a limited dataset [41-42]. For this reason, simpler models may achieve better results when trained on such data. [41]. In our work, we test whether these observations hold in the relatively unexplored domain of image-to-audio generation.

Moreover, we investigate the impact of various regularization techniques on the performance of these models. Techniques such as dropout, weight decay, and using learning rate schedulers have shown to improve model performance across several domains [43-45]. We aim to find optimal strategies that can improve the performance of models in the image-to-audio domain.

We believe that our research can provide valuable insights and contribute to advancing the state of the art, considering that the image-to-audio domain gets less attention relative to other fields in machine learning.

From the practical point of view, the significance of this work is directly tied to the significance of the image-to-audio domain. The applications of this domain include aiding visually impaired people by converting visual data into audio cues, enhancing virtual reality with corresponding audio for visual elements, improving educational tools where images from textbooks or videos could be accompanied by appropriate sounds to facilitate learning, and simplifying sound effects production in movies or video games. Furthermore, the research in image-to-audio field can give more insight and deeper understanding of cross-modal translations and can stimulate research in this area.

We divide our work into multiple sections, each of which further explores the concepts in an easy-to-follow way.

- Section 2 covers multiple preliminaries which will help readers to better understand the problems we face and our contributions.
- Section 3 describes the experiments we conducted, which includes the comparison of state-of-the-art approaches, the setup, the dataset, the resources we had, the flow of experiments and our thought processes throughout it.

- Section 4 describes the results as well as our reflections on the hypotheses we made in the introduction. We evaluate the efficacy of the regularization techniques and their impact on the performance.
- Section 5 draws the conclusions and future work. We summarize the key findings of our research, discuss limitations of our experiments, and suggest areas for future research that could further contribute to this domain.

With this structure in mind, we want to make our research comprehensible and accessible to as many people as possible. We hope our findings will contribute to the research of image-to-audio domain.

2 PRELIMINARIES

2.1 Convolutional Neural Networks (CNNs)

CNNs [46] are integral to many image processing tasks. They are a variation of Feedforward Neural Network (FNN) and use kernels to extract features from data. For this reason, they are very useful in image classification and feature extraction domain, where they found, arguably, the most use. An example of how CNNs work can be found in Figure 1.

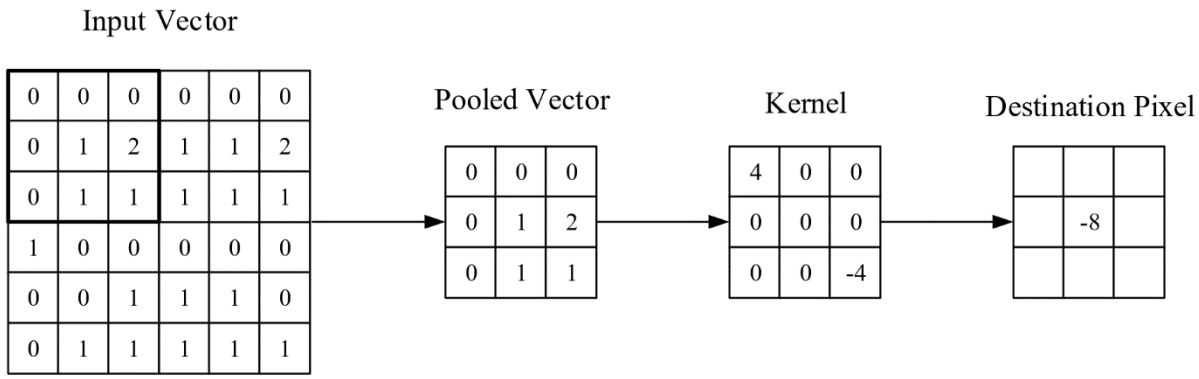

Figure 1. An example of a procedure of a 2D CNN [47].

Models like ResNet [48] and EfficientNet [49], which we will be using in our work, are two variants of CNNs trained on an ImageNet [51] dataset and widely used for image classification and feature extraction. An overview of the architecture of ResNet50 is provided in Figure 2.

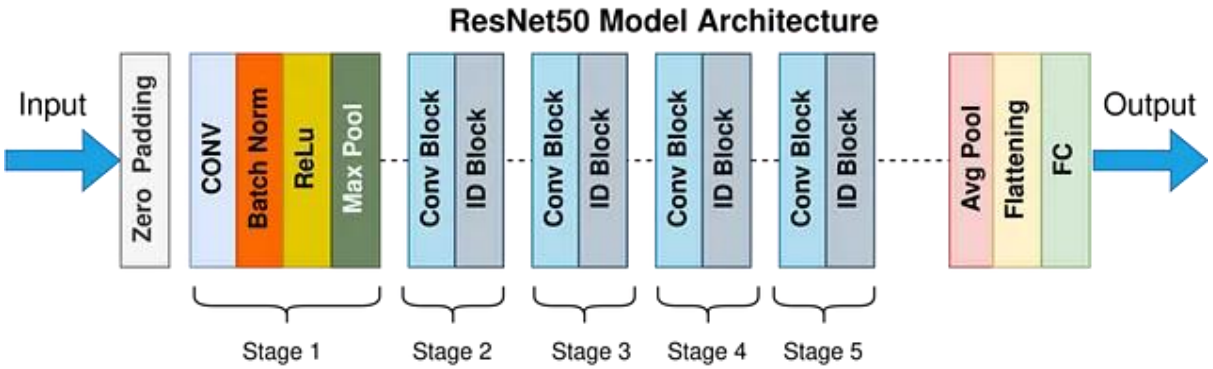

Figure 2. Architecture of ResNet50 model [52]

EfficientNet uses a more complex but still similar architecture inspired by ResNet. It is more complex than ResNet due to its approach to scaling and optimizing network dimensions. A comparison of ResNet, EfficientNet and other types of models can be seen in Figure 3.

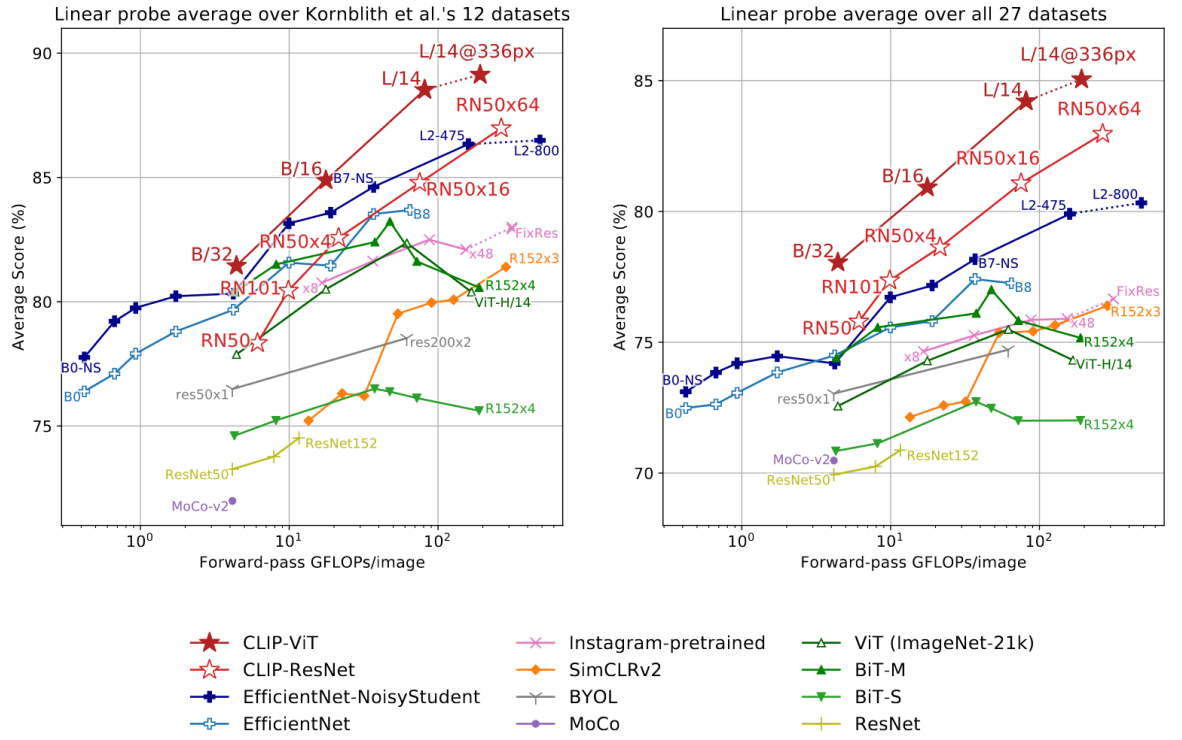

Figure 3. Comparison between ResNet, EfficientNet, CLIP, and other architectures based on the average accuracy scores across multiple datasets plotted against the number of floating point operations (FLOPs) taken for a single forward-pass of an image for that architecture [38].

In this work we are using two variants of ResNet called ResNet18 and ResNet50. They differ in size of the model used. The variant of EfficientNet architecture that we use is called EfficientNetV2 [50]. Specifically, we will be using two models called EfficientNetV2-S and EfficientNetV2-L that also differ in size. EfficnetNetV2 is a refined version of EfficientNet that optimizes training speed and parameter efficiency. The comparison between EfficientNet and EfficientNetV2 architectures can be seen in Figure 4.

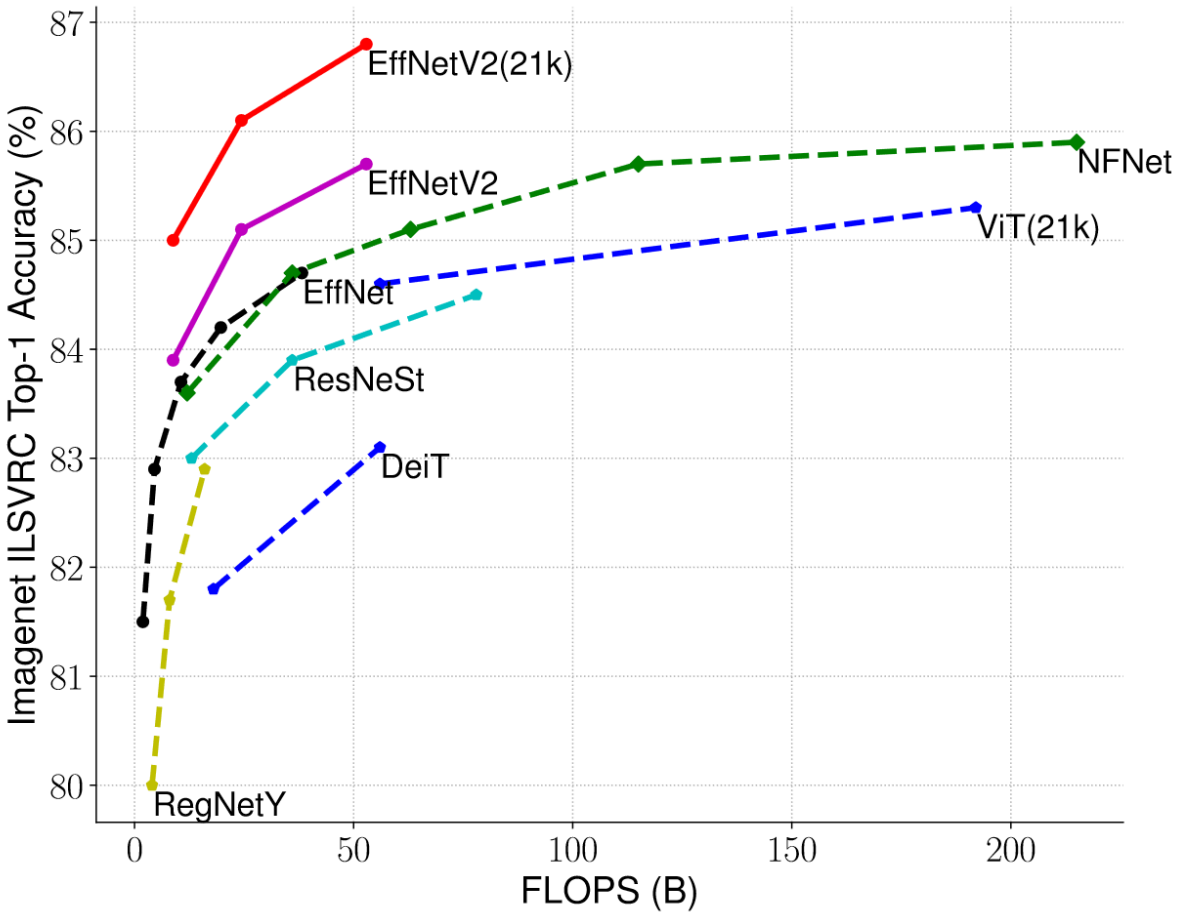

Figure 4. Accuracy of different architectures including EfficientNet and EfficientNetV2 on an ImageNet dataset plotted against FLOPs per forward-pass for an image.

2.2 Generative Pre-trained Transformer (GPT)

Transformer [26] is an architecture used in machine learning that has revolutionized machine learning in the past few years. It allows models to consider entire sequences of data at once by using attention mechanisms [26].

GPT [53] is a language model that uses the transformer architecture for predictive token generation. It was originally used for text generation but can be trained to

predict different kinds of tokens, including spectrograms [34, 36] and audio [37]. The architecture of a transformer presented in [26] can be seen in Figure 5.

In this work we use the version of GPT called GPT-2 developed by OpenAI [54].

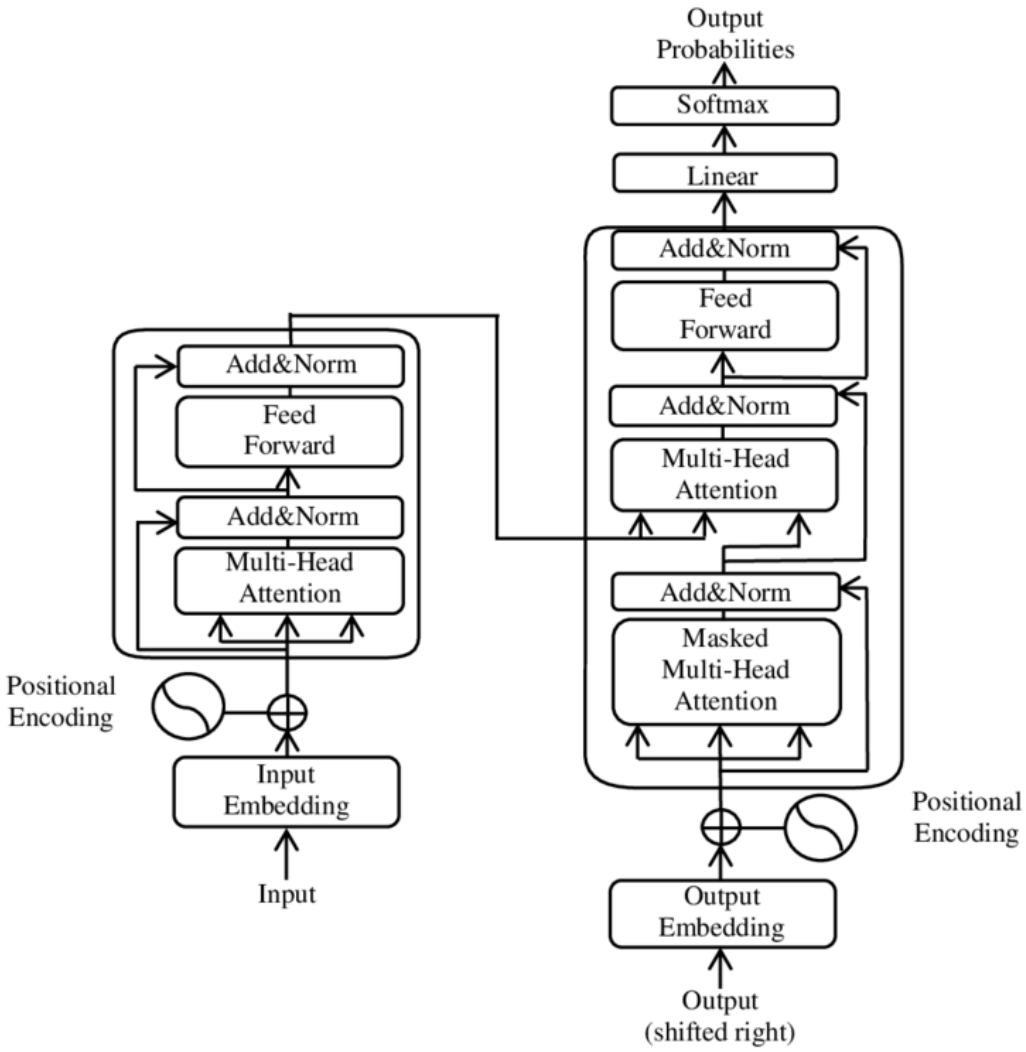

Figure 5. Architecture of a transformer [26].

2.3 Vision Transformers (ViTs)

ViTs adapt the transformer architecture described earlier to tasks in computer vision field such as image classifications, object detection, segmentation, action recognition etc. [55]. Figure 6 shows how Vision Transformers are structured.

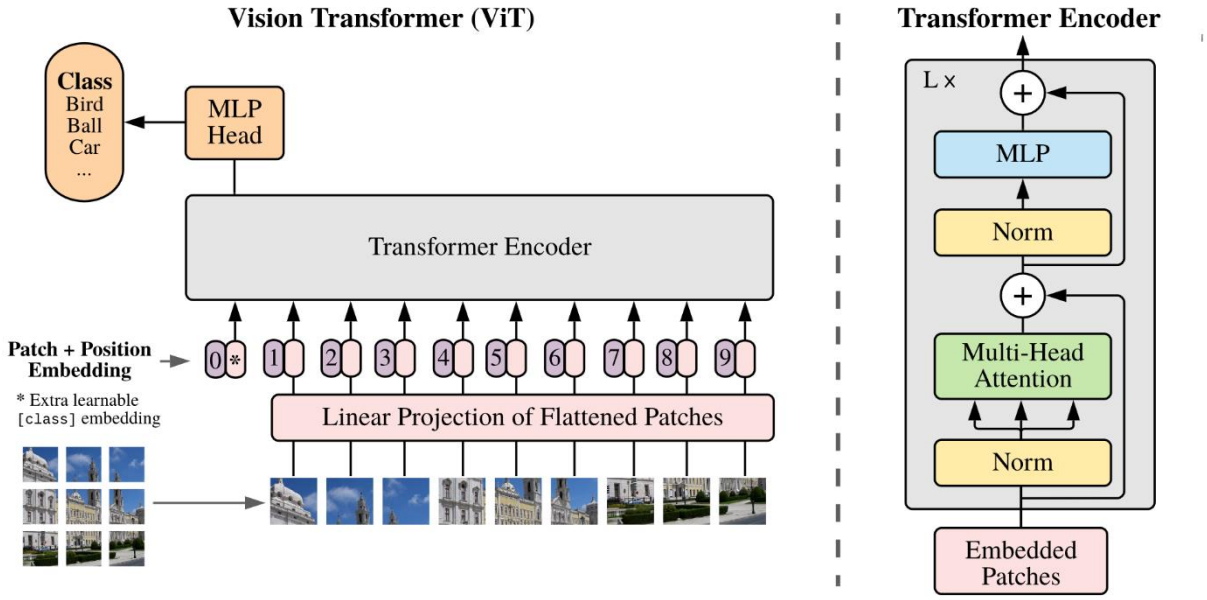

Figure 6. The structure of a Vision Transformer and the transformer encoder [55].

2.4 Contrastive Language-Image Pretraining (CLIP)

CLIP is a new approach proposed by OpenAI [38] that integrates visual perception of images with natural language understanding, by pairing images with textual descriptions during training, which helps it to better understand images and learn more visual concepts. It is an improvement over traditional CNN architecture which typically requires a fixed set of labeled images and learns from pixel-level data alone without the context provided by language. The approach used in CLIP is shown in Figure 7. In this work we will use the variant of CLIP based on Vision Transformer architecture, called CLIP-ViT-B-32 where

"B" stands for the size of the size model (Base) and 32 stands for the size of the patches of the image that is used to divide images (i.e. the images will be processed in 32x32 patches of pixels). The comparison between different variants of CLIP, EfficientNet and other models is shown in Figure 3.

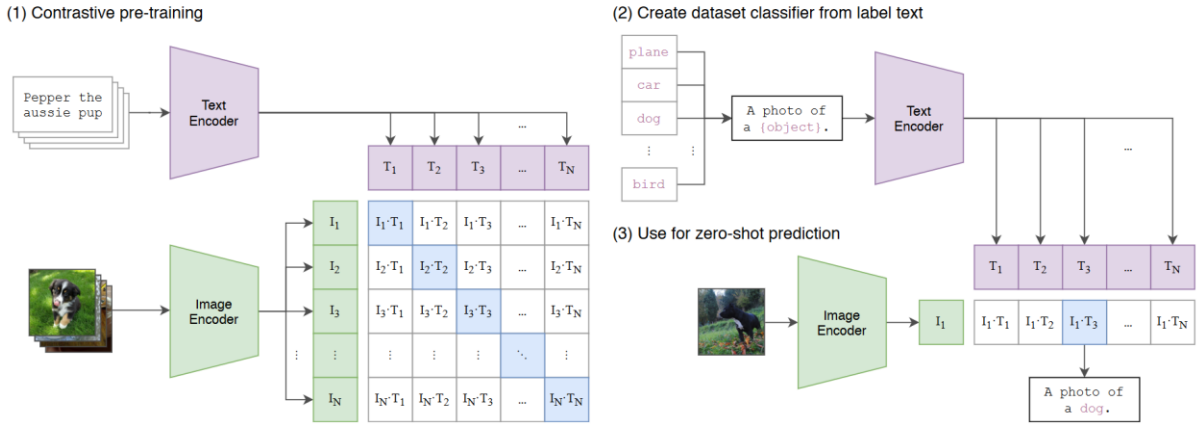

Figure 7. The summary of the approach used by CLIP [38]

2.5 Generative Adversarial Networks (GANs)

GANs, first presented by Goodfellow et al. [24] are implemented by a system of two neural networks, a generator and a discriminator, that are contesting with each other in a zero-sum game framework, a game theory concept where the total available reward is fixed, meaning that any gain by one player must come at a loss to another. GANs are often used in machine learning to generate new data instances that mimic real-world data [1,3,11-12].

2.6 Autoencoders and Variational Autoencoders (VAEs)

An autoencoder [56] is a type of neural network that has an encoder and a decoder, used to encode data to the compressed latent representation and reconstruct it back, while minimizing the difference between the input and its reconstruction. Variational Autoencoders (VAEs) [25] extend this concept by converting data to

a distribution over all possible values. This helps them to better generalize and be able to generate new data.

2.7 Vector Quantized Variational Autoencoders (VQ-VAEs)

VQ-VAEs introduced by Oord et al. [57] improve traditional VAEs by addressing a common problem in VAEs known as blurry outputs. They do this by adding a quantization mechanism that transforms a continuous latent representation into a discrete space. This way, each input is mapped to the nearest vector in a learned discrete set of embedding vectors (known as a codebook). Each vector in a codebook acts as a cluster center for similar types of data features. This allows the encoder to capture more sharp and distinct features compared to the smoother approach of VAEs. This approach incentivizes the model to learn more stable and distinctive features and thus results in better reconstructions.

3 EXPERIMENTS

We decided to first compare two existing state-of-the-art models for audio generation from video frames: SpecVQGAN developed by Sheffer et al. [37] and IM2WAV by Iashin et al. [36].

Both models use a class-agnostic approach and VQ-VAE for creating codebooks. However, they have different approaches. SpecVQGAN uses a variation of Autoencoder known as Vector Quantized Generative Adversarial Network (VQGAN) which is based on GANs and adds perceptual and adversarial losses to the traditional VQ-VAE losses. IM2WAV uses a hierarchical VQ-VAE instead.

Both models also use a similar transformer-based architecture for sampling but do it in a different way. SpecVQGAN uses a single transformer (a variant of GPT-2 called minGPT [58]) to sample spectrogram features which are then converted

to audio using a vocoder, while IM2WAV uses two transformers to directly sample low- and high-resolution audio tokens. The approach to sample spectrograms is more traditional, as often results in less noise.

In addition, SpecVQGAN conditions its single transformer on a sequence of features extracted using a variant of CNN called ResNet50, while IM2WAV uses an average of features extracted from CLIP. The authors claim that this allows its model to capture more semantic information because of the multi-modal approach used in CLIP.

While the benchmarks provided by authors of IM2WAV [37] suggest that their model outperforms SpecVQGAN, when testing these models in real scenarios we often noticed that SpecVQGAN produces cleaner and less noisy audio than IM2WAV. We suppose that this improvement is due to using spectrogram tokens instead of pure audio tokens. However, the other observation was that IM2WAV was better at recognizing certain objects and subtle details in the picture and reflecting them in the generated sound. We think that this improvement is due to using a more modern feature extractor (CLIP) instead of an older ResNet50.

We found that SpecVQGAN consistently outperforms IM2WAV, and considering this factor, as well as the clarity of the code, detailed documentation, and more conventional approach, we decided to focus our experiments on this model. We also assumed it had more space for improvement, since it relies on an older ResNet50 feature extractor, compared to the more recent CLIP model.

Dataset: Our primary dataset is The Visually Aligned Sound (VAS) dataset, proposed by Chen et al. [34], and was also used by Iashin et al. when training their SpecVQGAN model. However, the most advanced version of SpecVQGAN relies on a larger VGGSound dataset [59]. We opted for a smaller dataset in the initial planning of our experiments because of the resource limitations. This also

allowed us to train and test models faster than we could have with the larger dataset.

Resources: Initially, we started the feature extraction pipeline and the model training on a laptop GPU RTX 3060 with 6 GB of VRAM. Even though this allowed us to extract features and train the model on a minimal batch size, we quickly realized this was not enough for training, and especially for the evaluation part. For this reason, we moved to a desktop environment with RTX 3080 with 12 GB of VRAM and later to the local SLURM cluster. On this cluster, two nodes were primarily utilized: one with three A100 GPUs (each with 80 GB VRAM) and another with eight V100 GPUs (each with 40 GB VRAM).

Evaluation: We used the same metrics used by Iashin et al. to evaluate the fidelity and relevance of trained models. That is, we use Melception based Fréchet Inception Distance (FID) and KL-divergence (MKL) introduced by Iashin et al. [36].

3.1 Experiments Flow

3.1.1 Reproducing the initial results

At first, we tried reproducing the results of running the second step of training SpecVQGAN on a VAS dataset, which consisted of training a GPT-2 transformer, conditioned on video frame features to sample from the codebook. We used the pre-trained codebook from the same paper that was sampled on the same dataset (VAS).

3.1.2 Changes to the code

At this point we have transferred our entire training and validation pipeline to the local SLURM cluster and used SLURM jobs for both pipelines. Even though we had access to multiple nodes, to speed up the training we switched from using Distributed DataParallel (DDP) backend to DataParallel (DP). Even though this restricted us to only one node for training, we opted for it to avoid additional overhead and complexity caused by DDP, since the transformer model was not so big that it was required to scale up to more than one node. This allowed us to save resources and run more training pipelines in parallel.

The batch size was also increased to benefit from additional VRAM from the cluster GPUs. The learning rate was scaled accordingly to the batch size and the number of GPUs as a product of both:

$$Learning\ rate = Base\ learning\ rate * Batch\ Size * Number\ of\ GPUs * Number\ of\ Accumulated\ Grad\ Batches$$

Where the number of accumulated grad batches was kept as 1 to stay consistent with the experiments of Iashin et al. [36].

We also removed the early stopping callback due to its tendency to prematurely halt the training, complicating the debugging, and resulting in non-optimal model.

3.1.3 Changing the model backbones

Our next step was to replace the original ResNet50 feature extractor with EfficientNetV2 implementation from PyTorch. This choice was based on EfficientNetV2's more sophisticated yet conceptually similar architecture, which shows better performance on image classification tasks. In addition, the smallest version of EfficientNetV2 has approximately the same number of parameters as

ResNet50 (Figure 4), which makes for a fair comparison between two architectures and does not increase the time needed to extract features.

For feature extraction, we used the same pipeline as the authors of SpecVQGAN [36] which in turn was inspired by RegNet [34]. The pipeline uses weights that were pre-trained on ImageNet dataset and extracts RGB features from video frames. The size of the features depends on the architecture of the model, and in the case of EfficientNetV2, it is a vector of 1280 floating point values per frame.

We later did the same experiment with CLIP backbone (specifically CLIP-ViT-B-32), with pre-trained ImageNet weights. CLIP uses a different architecture than ResNet50 and EfficientNetV2, and shows really good performance on benchmarks, outperforming both (Figure 3). We were curious to see whether our transformer would benefit from this new approach. The size of the feature vector dimension for CLIP is now 512.

3.1.4 Addressing overfitting

During training, the model showed signs of overfitting early in the training (around 10^{th} – 15^{th} epoch). We thought that by preventing this overfit, we would give the transformer more time to train and therefore get better results. We tried using known regularization techniques, such as reducing the learning rate (by decreasing batch size), adding dropout and weight decay, and implementing learning rate schedulers (e.g. ReduceLROnPlateau). For finding the optimal base learning rate, we also used the automatic learning rate finding algorithms from PyTorch. We observed that applying all these regularization techniques has significantly improved the performance of the model. However, this alone did not solve the issue with overfitting completely. We tried using different learning rate schedulers such as Cosine Annealing, but were unable to get them to work due to

an error in PyTorch implementation. We decided to stick to ReduceLROnPlateau for its simplicity and popularity.

We also hypothesized that the overfitting could be related to the complexity of the transformer when using it on a small dataset such as VAS. To test this, we simplified the GPT-2 transformer by halving the number of layers (from 24 to 12), and the number of attention heads (from 16 to 8). First, we tested the model on ResNet50 features, then on CLIP features, where we also reduced the size of embedding dimension to match the size of the feature vector.

4 RESULTS

In total we tested the performance of the model for 18 different sets of parameters. We tried four different feature extractors, four different batch sizes, two values of dropout rate (0 and 0.3), two values of weight decay (0.01 and 0.1) and two variants of the transformer: one with 24 layers, 16 heads, 1024 embedding dimensions, and one with 12 layers, 8 heads and 512 embedding dimensions. Not all different sets of parameters were tested due to the long duration of the evaluation pipeline.

Features	**LR schedulers**	**Batch**	**GPUs**	**layers**	**heads**	**embed**	**dropout**	**decay**	**epochs**	**overfit?**	**KL**	**FID**
ResNet50	None	16	4	24	16	1024	0	0.01	?	yes	**6.3***	**25.1***
ResNet50	None	16	4	24	16	1024	0	0.01	10	yes	6.6	36.2
ResNet50	ReduceLROnPlateau	64	1	12	8	512	0.3	0.1	103	no	6.9	50.3
ResNet50	None	64	8	24	16	1024	0.3	0.01	10	yes	6.8	32.3
ResNet50	None	64	8	24	16	1024	0.3	0.1	7	yes	7.1	39.6
ResNet18	ReduceLROnPlateau	64	1	12	8	512	0.3	0.1	98	no	7.2	51.4
ResNet18	None	64	3	24	16	1024	0	0.01	13	yes	**6.8**	**34.7**
EfficientNetV2-S	None	32	3	24	16	1024	0	0.01	6	yes	**6.5**	47.7
EfficientNetV2-S	None	128	8	24	16	1024	0.3	0.01	8	yes	7.2	37.1
EfficientNetV2-S	None	128	8	24	16	1024	0.3	0.1	10	yes	**6.5**	**30.6**
EfficientNetV2-L	None	64	3	24	16	1024	0	0.01	16	yes	**7.9**	**34.0**
CLIP-ViT-B-32	None	32	3	24	16	1024	0	0.01	8	yes	6.4	43.1
CLIP-ViT-B-32	None	64	3	24	16	1024	0.3	0.01	10	yes	**5.7**	27.1
CLIP-ViT-B-32	None	64	8	24	16	1024	0.3	0.1	11	yes	6.1	25.1
CLIP-ViT-B-32	ReduceLROnPlateau	64	8	24	16	512	0.3	0.01	24	yes	5.9	**24.8**
CLIP-ViT-B-32	ReduceLROnPlateau	64	8	24	16	512	0.3	0.1	25	yes	6.0	25.9
CLIP-ViT-B-32	None	64	3	12	8	512	0	0.01	16	yes	5.9	34.4
CLIP-ViT-B-32	None	64	4	24	16	1024	0.3	0.1	11	yes	6.2	24.9

Table 1. The comparison between different sets of hyperparameters used when training the model and the results obtained. The models are grouped by the feature extractors used. * The first row represents the results listed on the creators GitHub page.

Our initial attempts at reproducing the results of training the SpecVQGAN transformer failed, despite using the same configuration as in the original paper, which can be seen in the first two rows of the table: the first row represents the baseline results we should that the authors claimed, the second row represents what we achieved. We noticed that others have raised similar issues on the author's GitHub repository when training this exact model for VAS dataset we used, suggesting possible inaccuracies on the author's end.

Despite the differences, we continued our experiments on the same dataset and the same repository since the goal of our work was to test the underlying theories about transformers and the effects of different hyperparameters on its performance, which was feasible regardless of correctness of the implementation code provided by the authors.

4.1 Impact of different feature extractors

When using EfficientNetV2 and CLIP models we saw a decrease in performance in our first attempts. However, after some tuning these models started to outperform the original ResNet50 model on the same hyperparameters, as seen in Table 1. This aligns with the statement of the authors of SpecVQGAN that the performance of the model improves when using a feature extractor that performs better on benchmarks [36]. The CLIP model works exceptionally well and aligns with the hypothesis of authors of IM2WAV that multi-modal optimization of CLIP allows for an easier sampling of an additional modality (in this case, audio features).

We have confirmed our initial hypothesis that more complex models like ResNet50 (compared to simpler ResNet18) can improve MKL scores due to their ability to better capture unorthodox patterns and align predictions with the target. However, they are prone to overfitting and produce higher FID due to their

specificity, and poorer generalization to unseen data. The results can be seen in Table 1. As we can see, the model trained on ResNet18 features has MKL value of **6.8** compared to **6.6** of the equivalent model which uses ResNet50 features, while having lower FID score of **34.7** (ResNet18 features) against **36.2** (ResNet50 features)

Interestingly, this hypothesis does not hold for the EfficientNetV2 models: in Table 1 the model with EfficientNetV2-L features shows improvement on the FID metrics but decrease on MKL. We assume this is due to the non-straightforward architecture of EfficientNetV2 compared to relatively simpler ResNet50, Because of this complexity there is no direct scaling of specificity in its features, and the behavior of the model is harder to predict. Another reason for this is the fact that the larger model is able to capture a broader range of features that ultimately represent the ground truth better, even if the distribution of predicted features is not identical. We were curious to see the performance of EfficientNetV2-M model, which is in the middle between EfficientNetV2-S and EfficientNetV2-L in terms of the number of parameters (Figure 4), but due to the lack of resources we left this idea for the future work.

4.2 Impact of transformer size

Our experiments support the theory we made in the introduction that simplified models could yield a better result with limited data than the larger models. As an example, consider the two models that use the ResNet50 extractor in Table 1 that have the same hyperparameters except for the number of layers, attention heads, and embedding dimensions. This also holds for two models that use CLIP backbone. However, this is not the case when combining this method with regularization techniques since the model starts to underfit and results in worse performance as also seen in Table 1. We describe this notion in more detail in the next section.

4.3 Impact of overfitting

Although using regularization techniques prevented overfitting, and lead to smoother validation loss graphs, the model's overall performance dropped.

We disproved the assumption we made when running the experiments that the quick overfit of the model in the beginning of the training is inherently bad and should be avoided. While it is true that this allows the model to train for up to 100 epochs and consistently improves over time (Figure 9), compared to the more unorthodox loss plot seen in Figure 8. the performance of the model severely decreases as seen in Table 1. We suppose that at this point the model's capacity was not sufficient and it started to underfit.

It is also worth noting that even in the case of overfitting, the model checkpoint will be saved at the end of each epoch if its validation loss was the best among all the previous epochs. This means that we did not evaluate the performance of the version of the model, which was overfitting, but rather the last model before the overfitting started.

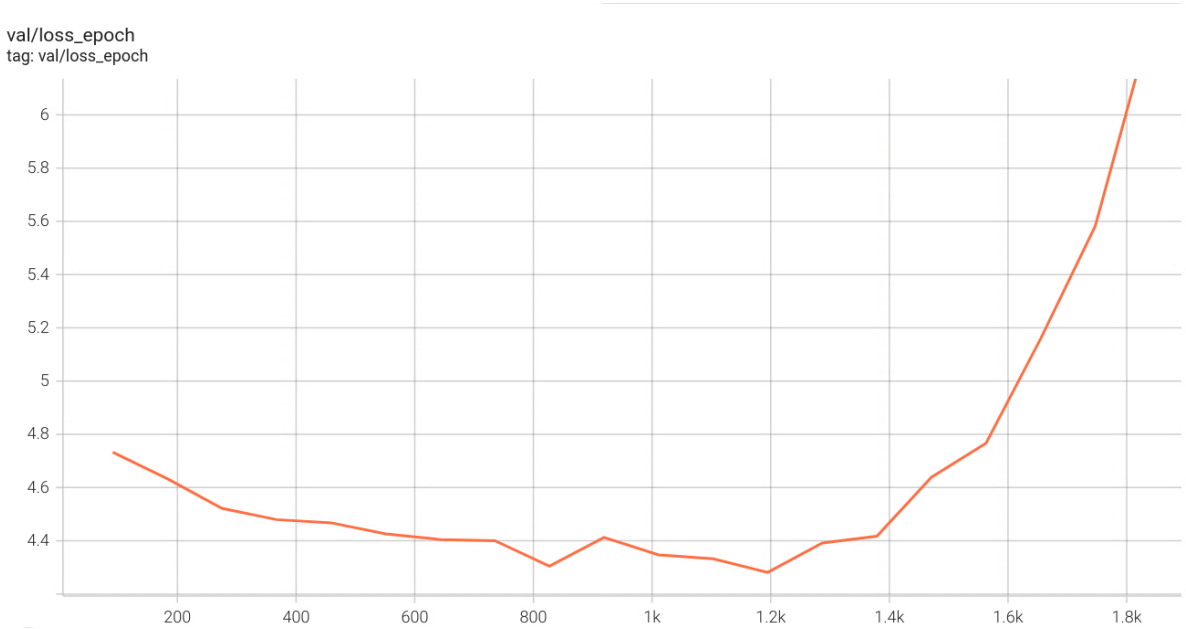

Figure 8. The plot of validation loss against the number of training steps taken when the model overfits. The curve starts improving in the beginning but starts overfitting around 1.5K step. Note that even though the model overfits, the best model checkpoint at ~1.2K steps) is saved and will be used for evaluation.

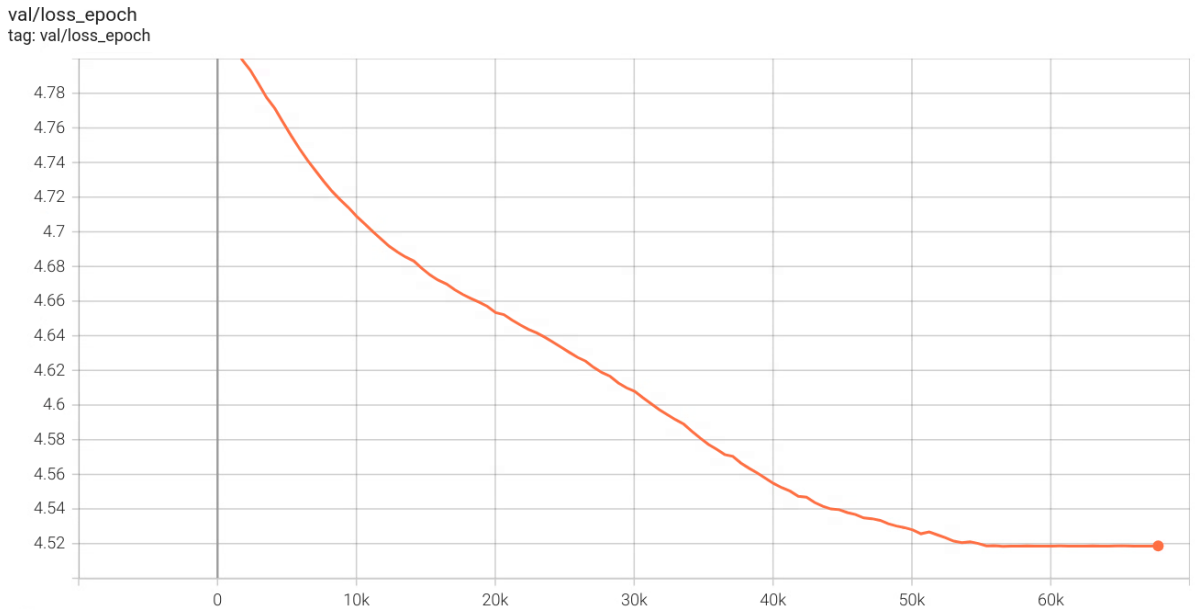

Figure 9. The plot of validation loss against the number of training steps taken when the model does not overfit. One can see that the curve pattern suggests very consistent improvements of the model throughout the training, compared to the case when the model overfits. Also note the difference in the number of training steps: in this case the model keeps converging for a longer time.

4.4 Impact of regularizations

Four techniques for regularization were considered: dropout, L2 regularization using bigger weight decay, taking smaller learning rate (by increasing batch size) and using learning rate schedulers (ReduceLROnPlateau from PyTorch Lightning Python library. The results of its usage are shown on Figure 10)

We confirmed another assumption that we outlined in the introduction that different regularization techniques transfer also to the image-to-audio domain.

Adding dropout of **0.3** has shown to consistently improve the performance of the models, regardless of the feature extractor or any other parameters. We did not test the performance of any other dropout since we were already satisfied with the improvements, though we assume the value could be tuned further, which we leave for the future work.

Increasing weight decay from **0.01** to **0.1** also helps prevent overfit and increases the results most of the time, even though there are some exceptions as seen in Table 1.

The increase of batch size did not severely affect the accuracy of the model, and this allows scaling the batch size based on the amount of VRAM in a GPU.

Adding the learning rate schedulers allows to smooth the training curve towards the end of the training. This allows model to train for more epochs towards the end when it starts overfitting, thus improving performance (see Table 1)

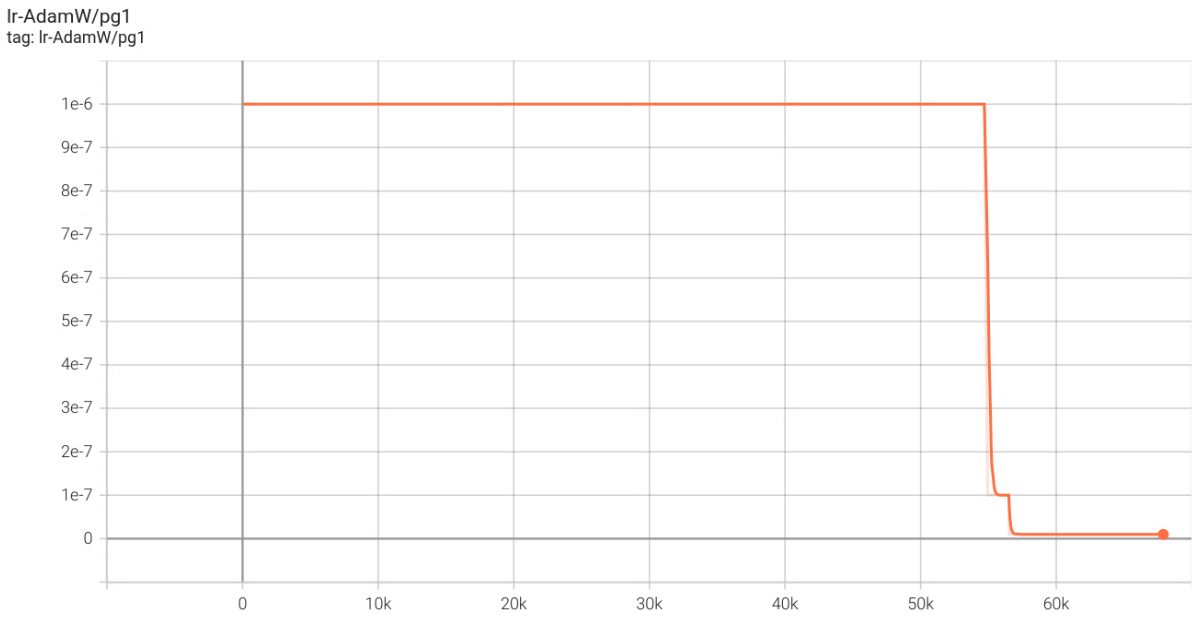

Figure 10. The plot of learning rate against the number of training steps taken when using ReduceLROnPlateau learning rate scheduler with patience 10. The learning rate drops in value when the model stops improving, which slows down the training and prevents overfit.

4.5 Limitations of the experiments and future work

We were unable to evaluate the performance of all configurations of parameters due to the amount of time it takes to sample and evaluate the model's performance. For example, we did not use other learning rate schedulers besides ReduceLROnPlateau and the future work could try using more of them to further finetune the model.

The base learning rate was the same for all models (1e-6), the same base learning rate that was used when training SpecVQGAN on the same dataset. The learning rate was automatically adjusted according to the number of batches and GPUs. As we had limited resources, we did not fully explore further experiments with changing the base learning rate.

To get better results, one might also try combining two approaches used by SpecVQGAN and IM2WAV. For example, one might try to improve IM2WAV model by sampling spectrogram tokens instead of pure audio tokens, or the other idea might be to use a more recent type of codebook like ViT-VQGAN [60] based on Vision Transformers or HQ-VAE [61] in both models.

5 CONCLUSIONS

In this work, we presented a comprehensive study of image-to-audio generation domain, which has a wide range of possible applications in our daily life, ranging from improving accessibility to improving entertainment experiences. We believe that our work has the potential to incentivize further research in the domain of sound generation and cross-modal translations in general.

We reviewed current state-of-the-art models and studied different approaches they used. We looked more closely at two state-of-the-art models, SpecVQGAN

[36] and IM2WAV [37], and applied them in real-world scenarios. Through our experiments, we observed notable differences in the audio output quality and semantic accuracy between these models, with SpecVQGAN generally providing clearer audio but worse results in object recognition, and IM2WAV excelling in the latter.

We have also tested the application of known hypotheses from various machine learning domains to the image-to-audio context by evaluating the changes in performance across different architectural conditions. Our tests included various feature extractor architectures such as ResNet, EfficientNetV2 and CLIP. We also explored the impact of different model configurations, including the number of layers, attention heads, and embedding dimensions, and the significance of various hyperparameter settings, such as the batch size, dropout rate and weight decay.

We confirmed our initial hypothesis that more complex models like ResNet50 (compared to simpler ResNet18) can improve KL scores due to their ability to better capture sophisticated patterns and align predictions with the target. However, these complex models often lead to worse generalization of new data and require a lot of data for training, which gives them lower FID scores. For this reason, simpler models might be used in cases where the dataset is limited. However, one might still prioritize for complex models if their goal is to minimize KL scores (when prioritizing accuracy over generalization).

We also confirmed that various regularization techniques could indeed be critical in fine-tuning these models, but they need to be used carefully and require delicate balance to maintain good results. For example, adding dropout to the model significantly improves the performance of the model regardless of the underlying feature extractor used, while the effects of weight decay vary. Simple models that have less capacity (like ResNet) might be less capable of managing the complexity introduced by higher regularization without weakening its ability to

learn from data. Thus, lower weight decay is likely a better balance for such architectures, while for models EfficientNetV2 and CLIP, a higher value of weight decay is a great counterbalance to their complexity and helps to maintain healthy regularization and prevent overfitting. We have also tested the use of learning rate schedulers and shown their effectiveness. Our results also illustrated a common trade-off in machine learning between model complexity and its tendency to overfit or underfit, where too little regularization makes the model overfit and too much regularization and model simplification results in underfitting.

In addition, we verified the claims of the creators of SpecVQGAN and IM2WAV. Iashin et al. assumed that model performance improves with backbones that have higher benchmarking scores [36], and Sheffer et al. [37] believed that using multi-modal feature extractors like CLIP makes the generation of additional modalities easier when compared with traditional CNN approach.

The future work could focus on exploring different learning rate schedulers, adjusting the base learning rate, and trying hybrid approaches that combine successful elements from SpecVQGAN, IM2WAV and potentially other models.

We believe that our findings provide a valuable insight into the challenges of fine-tuning of image-to-audio models. since we have not only proved that most common techniques from other fields of machine learning are applicable in this domain as well, but also opened other areas of future work. As this field evolves, it will be crucial to use more dynamic and adaptive models and tuning techniques capable of handling complex scenarios.

6 REFERENCES

[1] Choi et al., “StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation”, arXiv:1711.09020, Sep. 2018.

[2] Esser et al., “Taming Transformers for High-Resolution Image Synthesis”, arXiv:2012.09841, Jun. 2021.

[3] Lee et al., “MaskGAN: Towards Diverse and Interactive Facial Image Manipulation”, arXiv:1907.11922, Apr. 2021.

[4] Saharia et al., “Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding”, arXiv:2205.11487, May 2022.

[5] Ramesh et al., “Zero-Shot Text-to-Image Generation”, arXiv:2102.12092, Feb. 2021.

[6] Nichol et al., “GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models”, arXiv:2112.1074, Mar. 2022.

[7] Reed et al., “Generative Adversarial Text to Image Synthesis”, arXiv:1605.05396, Jun. 2016.

[8] Rombach et al., “High-Resolution Image Synthesis with Latent Diffusion Models”, arXiv:2112.10752, Apr. 2022.

[9] Zhang et al., “Self-Attention Generative Adversarial Networks”, arXiv:1805.08318, Jun. 2019.

[10] Ho et al., “Denoising Diffusion Probabilistic Models”, arXiv:2006.11239, Dec. 2020.

[11] Tulyakov et al., “MoCoGAN: Decomposing Motion and Content for Video Generation”, arXiv:1707.04993, Dec. 2017.

[12] Chen et al., “Mocycle-GAN: Unpaired Video-to-Video Translation”, arXiv:1908.09514, Aug. 2019.

[13] Li et al., “Video Generation From Text”, arXiv:1710.00421, Oct. 2017.

[14] Lee et al., “Stochastic Adversarial Video Prediction”, arXiv:1804.01523, Apr. 2018.

[15] Chan et al., “Everybody Dance Now”, arXiv:1808.07371, Aug. 2018.

[16] Copet et al., “Simple and Controllable Music Generation”, arXiv:2306.05284, Jan. 2024.

[17] Huang et al., “Noise2Music: Text-conditioned Music Generation with Diffusion Models”, arXiv:2302.03917, Feb. 2023.

[18] Schneider et al., “Moûsai: Text-to-Music Generation with Long-Context Latent Diffusion”, arXiv:2301.11757, Jan. 2023.

[19] Garcia et al., “VampNet: Music Generation via Masked Acoustic Token Modeling”, arXiv:2307.04686, Jul. 2023.

[20] Lam et al., “Efficient Neural Music Generation”, arXiv:2305.15719, May 2023.

[21] Liu et al., “AudioLDM: Text-to-Audio Generation with Latent Diffusion Models”, arXiv:2301.12503, Jan. 2023.

[22] Huang et al., “Make-An-Audio: Text-To-Audio Generation with Prompt-Enhanced Diffusion Models”, arXiv:2301.12661, Jan. 2023.

[23] Borsos et al., “SoundStorm: Efficient Parallel Audio Generation”, arXiv:2305.09636, May 2023.

[24] Goodfellow et al., “Generative Adversarial Nets”, arXiv:1406.2661, Jun. 2014.

[25] Kingma et al., “Auto-Encoding Variational Bayes”, arXiv:1312.6114, Dec. 2013.

[26] Vaswani et al., “Attention Is All You Need”, arXiv:1706.03762, Jun. 2017.

[27] Algur et al., “Correlation analysis of audio and video contents: A metadata based approach”, 10.1109/ICATCCT.2015.7456908, Oct. 2015.

[28] Yu et al., “Deep Cross-Modal Correlation Learning for Audio and Lyrics in Music Retrieval”, arXiv:1711.08976, Nov. 2017.

[29] Mo et al., “A Unified Audio-Visual Learning Framework for Localization, Separation, and Recognition”, arXiv:2305.19458, May 2023.

[30] Majumder et al., “Active Audio-Visual Separation of Dynamic Sound Sources”, arXiv:2202.00850, Jul. 2022.

[31] Qian et al., “Multiple Sound Sources Localization from Coarse to Fine”, arXiv:2007.06355, Jul. 2020.

[32] Ning et al. “Audio Description from Image by Modal Translation Network”, arXiv:2103.10018, Mar. 2021.

[33] Owens et al., “Visually Indicated Sounds”, arXiv:1512.08512, Dec. 2015.

[34] Chen et al., “Generating Visually Aligned Sound from Videos”, arXiv:2008.00820, Jul. 2020.

[35] Tan et al., “Spectrogram Analysis Via Self-Attention for Realizing Cross-Model Visual-Audio Generation”, DOI:10.1109/ICASSP40776.2020.9052918, May 2020.

[36] Iashin et al., “Taming Visually Guided Sound Generation”, arXiv:2110.08791, Oct. 2021.

[37] Sheffer et al., “I Hear Your True Colors: Image Guided Audio Generation”, arXiv:2211.03089, Nov. 2022.

[38] Radford et al., “Learning Transferable Visual Models From Natural Language Supervision”, arXiv:2103.00020, Feb. 2021.

[39] Hochreiter et al., “Long Short-Term Memory”, DOI:10.1162/neco.1997.9.8.1735, Nov. 1997.

[40] A. Krizhevsky, “One weird trick for parallelizing convolutional neural networks”, arXiv:1404.5997v2, Apr. 2014.

[41] J. Schmidt, “Testing for Overfitting”, arXiv:2305.05792v1, May 2023.

[42] Strashko et al., “Generalization and Overfitting in Matrix Product State Machine Learning Architectures”, arXiv:2208.04372v1, Aug. 2022.

[43] O' Neill et al., “Analysing Dropout and Compounding Errors in Neural Language Models”, arXiv:1811.00998v1, Nov. 2018.

[44] A. Lewkowycz, “How to decay your learning rate”, arXiv:2103.12682v1, Mar. 2021.

[45] Xu et al., “Empirical Study of Overfitting in Deep FNN Prediction Models for Breast Cancer Metastasis”, arXiv:2208.02150v1, Aug. 2022.

[46] Lecun et al., “Gradient-based learning applied to document recognition”, DOI:10.1109/5.726791, Nov. 1998.

[47] O'Shea et al., “An Introduction to Convolutional Neural Networks”, arXiv:1511.08458, Dec. 2015.

[48] He et al., “Deep Residual Learning for Image Recognition”, arXiv:1512.03385, Dec. 2015.

[49] Tan et al., “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks”, arXiv:1905.11946v5, Sep. 2020.

[50] Tan et al., “EfficientNetV2: Smaller Models and Faster Training”, arXiv:2104.00298, Jun. 2021.

[51] Deng et al., “ImageNet: A large-scale hierarchical image database”, DOI:10.1109/CVPR.2009.5206848, Jun. 2009.

[52] Nitish Kundu. “Exploring ResNet50: An In-Depth Look at the Model Architecture and Code Implementation”, Medium, Jan. 23, 2023. [Online]. Available: https://medium.com/@nitishkundu1993/exploring-resnet50-an-in-depth-look-at-the-model-architecture-and-code-implementation-d8d8fa67e46f

[53] Radford et al., “Improving Language Understanding by Generative Pre-Training”, 2018. [Online]. Available: https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf

[54] Radform et al., “Language models are unsupervised multitask learners”, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:16002553

[55] Dosovitskiy et al., “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”, arXiv:2010.11929, Jun. 2021.

[56] U. Michelucci, “An Introduction to Autoencoders”, arXiv:2201.03898v1, Jan. 2022.

[57] Oord et al., “Neural Discrete Representation Learning”, arXiv:1711.00937, Nov. 2017.

[58] A. Karpathy, (re-implementation of [53]), 2020. [Online]. Available: https://github.com/karpathy/minGPT

[59] Chen et al., “VGGSound: A Large-scale Audio-Visual Dataset”, arXiv:2004.14368, Sep. 2020.

[60] Yu et al., “Vector-quantized Image Modeling with Improved VQGAN”, arXiv:2110.04627, Jun. 2022.

[61] Takida et al., “HQ-VAE: Hierarchical Discrete Representation Learning with Variational Bayes”, arXiv:2401.00365, Mar. 2024.