FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

# ASSIGNMENT OF BACHELOR'S THESIS

| | |
|---|---|
| **Title:** | Self-supervised model for efficient sound recognition trained on aggregated data |
| **Student:** | Vojtěch Houska |
| **Supervisor:** | Mgr. Alexander Kovalenko, Ph.D. |
| **Study Programme:** | Informatics |
| **Study Branch:** | Knowledge Engineering |
| **Department:** | Department of Applied Mathematics |
| **Validity:** | Until the end of winter semester 2021/22 |

## Instructions

The task is to create an efficient model based on neural network for state-of-the-art computer audition (eg. music instrument recognition). The model is intended to learn from aggregated data.

1) Analyze the latest state-of-the-art approaches for sound classification using deep neural networks, including signal preprocessing, neural network type and architecture.

2) Define general and specific obstacles, constraints and recommendations in the field of sound recognition.

3) Compare and suggest:
- various sound preprocessing techniques;
- various neural network types;
- various neural network architecture;
- manually vs. automatically extracted features;

4) Implement and test proposed applications

## References

Will be provided by the supervisor.

Ing. Karel Klouda, Ph.D.
Head of Department

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
Dean

Prague February 20, 2020

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Bachelor's thesis

# Self-supervised model for efficient sound recognition trained on aggregated data

***Vojtěch Houska***

Department of Applied Mathematics
Supervisor: Mgr. Alexander Kovalenko, Ph.D.

February 14, 2021

# Acknowledgements

I would like to thank my supervisor Mgr. Alexandr Kovalenko, Ph.D. for his great patience and advice provided throughout this work. Additional thanks belongs to my family for their support during my studies.

# Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as school work under the provisions of Article 60(1) of the Act.

In Prague on February 14, 2021 ……

Czech Technical University in Prague
Faculty of Information Technology


### Citation of this thesis

Houska, Vojtěch. *Self-supervised model for efficient sound recognition trained on aggregated data*. Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2021.

# Abstract

The thesis summarizes state-of-the-art approaches in deep learning. It discusses application of self-supervised autoencoders and pre-processing techniques used in sound recognition. YouTube platform served as a source of weakly-labeled data to train such models. Latent space properties of proposed autoencoders were compared and tested using K-means clustering. Implementation of Adversarially Constrained Autoencoder Interpolation failed to outperform randomly initialized autoencoder. The reasons are further discussed and several recommendations for future research are proposed.

**Keywords** Weakly-labeled data, Sound classification, Deep learning, Autoencoders, Imbalanced data, Self-supervised learning, K-means, Adversarially Constrained Autoencoder Interpolation

# Abstrakt

Tato práce shrnuje nejmodernější metody využívané v hlubokém učení. Probírá použití autoenkodérů a metody předzpracování v oblasti rozpoznávání zvuku. Jako zdroj slabě anotovaných dat pro učení těchto modelů byla použita platforma YouTube. Práce porovnala vlastnosti latentních prostorů navrhovaných autoenkoderů, které byly testovány pomocí shlukování K-means. Použitá metoda regularizovaného autoenkodéru nepřekonala náhodně inicializovaný autoenkodér. V závěru práce jsou rozebrány příčiny a byla navrhnuta další doporučení pro pozdější výzkum.

**Klíčová slova** slabě anotovaná data, rozpoznávání zvuku, hluboké učení, autoenkodéry, nevyvážená data, učení bez učitele, k-means, Adversarially Constrained Autoencoder Interpolation

# Contents

# List of Figures

# Introduction

## Motivation

In recent years there has been tremendous progress in machine learning, especially in field of deep learning, where more and more accurate models are created. But most of them are supervised since it is easier for a model to extract appropriate features of classes from clean data. With growing demand on classification models, not just on their accuracy but other properties of classifier, there is bigger need for larger, more balanced and robust datasets, which are expensive to create. In order to bypass labeling, model can be trained on data that is partially labeled (this technique is called weak labeling) or trained on few labeled data and bigger portion of unlabeled data (this technique is called semi-supervised learning). Other approaches are self-supervision, where model is supervised by input data, and unsupervised learning, where model does not use any labels at all and should assign appropriate classes based solely on feature extraction.

Why audio? because sounds are weakly labeled by nature. It is hard to have clean sound with only one class or without noise. And I believe, that it is much easier to mimic how ear works in deep learning models then it is with eye, because we can move them in our eye sockets or change focus of our sight. Hearing on other hand is quite stationary and there are no active parts in our ears that could give better hearing resolution, filter noise etc. Goal of this work is to investigate latest approaches in computer audiation and propose model that learns to classify instruments contained in sound, using videos aggregated from YouTube. Videos will be selected based on appropriate search query, which is very cheap but very noisy dataset.

## Brief History of Neural Networks

In 1957 Psychologist Frank Rosenblat proposed biologically inspired computational system Perceptron that can be considered as first neural net consisting of two layers [1, 2, 3].

Single unit of a Perceptron can be described as function:

$$f(x) = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} w_i x_i + b > 0 \\ 0 & otherwise \end{cases} \tag{1}$$

where $w_i$ is weight of $i$-th input pixel $x_i$ and $b$ is bias or shift along $x$ axis.

In 1960 Professor of electrical engineering Bernard Widrow and his graduate student Ted Hoff proposed perceptron called ADALINE (Adaptive Linear Neuron) which adapts in order to filter echos on a phone line. It was first application of neural network in commercial use and it is still used today [4, 2, 3].

Soon in 1969 cognitive scientist Marvin Minsky and mathematician Seymour Papert published book *"Perceptrons: An Introduction to Computational Geometry"* that points out shortcomings of the Perceptron [5, 2, 3].

Minsky and Papert's book has caused fall of Perceptrons popularity and fundings in the field of artificial neural networks [2, 3].

After multiple rediscoveries of backpropagation in 80s, neural networks gained back attention [2].

In 1986 Michael I. Jordan published his *"Attractor dynamics and parallelism in a connectionist sequential machine"* [6]. For first time introduced recurrence in neural networks. But first mention of name recurrent neural networks was in 1989 when Barak A. Pearlmutter published *"Learning state space trajectories in recurrent neural networks"* [7].

In 1991 Sepp Hochreiter in his diploma thesis described vanishing gradient occuring in recurrent neural networks, phenomenon that prevents learning of deeper neural networks and proposed recurrent neural network Long Short-term Memory (LSTM), that mitigates effects of vanishing gradient. Hochreiter and his colleague Jürgen Schmidhuber introduced final results of LSTM in 1997 [8].

In 1994 Yoshua Bengio et al. published paper that focuses solely about vanishing gradient in neural networks [9].

In 1998 Yann Lecun in his paper *"Gradient-Based Learning Applied to Document Recognition"* proposed model called Convolutional Neural Networks CNNs, that used convolutional kernels for feature extraction [10]. Model was intended to classify sequences of numbers from images and it was continuation of work done by Kunihiko Fukushima using backpropagation [11].

Chapter 1

# Deep Learning

## 1.1 Introduction

As mentioned earlier, artificial neuron is based on a model of biological neuron found in almost every living creature on our planet. Despite its oversimplification (accurate models that are used for prediction of its behaviour are far more complicated [12]) there are some obvious similarities that deserve simple description.

Neuron dendrites act as an input of electrical signal (action potential), received from nerves or other neurons through gaps or electrical links with different strength, called synapses. Stronger the synapse easier it is for signal to be passed. Action potential coming from dendrites are summed and propagated to neurons soma which might produce excitation, electrical signals known as spikes. Spikes are further distributed along axon to terminal which act as neuron‘s output that is received again by other neurons. Neurons with thousands of synapses were found [13].

Neurons that way pass information from senses to more complex structures of our brain forming distributed hierarchical structure, where each neuron is a single unit communicating with each other exhibiting complex behaviour [14].

Same as its biological brother, artificial neuron has sum of weighted inputs (dendrites) from other neurons or initial input followed by activation function (excitation of the soma) where its output is passed to other neurons of a network.

## 1.2 Basics of Deep Learning

### 1.2.1 Feedforward Neural Network

Neural networks are organized in layers of neruons. Simplest form of neural network is called Feedforward Neural Network (FFA) also known as Multi-

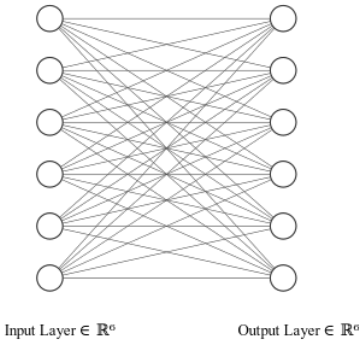



Figure 1.1: Single layer of Feedforward Neural Network with 6 neurons. M = 6 outputs of previous layer and N = 6 neurons

Layer Perceptron (MLP) [15]. Single layer of MLP is organized of N neurons connected to all M outputs of previous layer as is shown in 1.1

Therefore, output of the layer is vector of $M$ values computed as weighted sum of input vector passed through activation function. It is more convenient to think about it as linear operator followed by pointwise non-linear operation written as $f(W * x + b)$ where $W \in R^{N \times M}$ is weight matrix with weights of neurons written in rows, $x \in R^M$ is input vector, $b$ is bias and $f$ is pointwise non-linear function. Layers of Feedforward Neural Network are usually stacked on top of each other therefore it can be written as a composition of functions: $f_d(f_{d-1}...f_2(f_1(x))...))$ where $d$ is depth of the neural network and $f_i$ is parametric function described above. That is why it is called deep learning [16].

### 1.2.2 Used Activation Functions

#### Sigmoid Function

Sigmoid is a function that smoothly transition between 0 and 1 [17]. Can be described as

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{1.1}$$

and its derivative as

$$\frac{d}{dx}\sigma(x) = \sigma * (1 - \sigma) \tag{1.2}$$

### Hyperbolic Tangent

Hyperbolic tangent is function that smoothly transition between -1 and 1 [18]. It can be described as

$$\tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1} \tag{1.3}$$

and its derivative as

$$\frac{d}{dx}\tanh(x) = 1 - \tanh^2(x) \tag{1.4}$$

### Rectified Linear Unit

Rectified Linear unit ReLU also known as positive part [19]. It can be described as

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & otherwise \end{cases} \tag{1.5}$$

ReLU does not have derivative in 0 but its set to 1 or 0 as convention.

$$f(x) = \begin{cases} 1 & \text{if } x >= 0 \\ 0 & otherwise \end{cases} \tag{1.6}$$

### LeakyReLU

LeakyReLU is extension of ReLU to prevent so called dead neuron in ReLU by defining negative part as multiple of small constant therefore derivative in negative part is not 0 [20]. LeakyReLU can be described as

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ 0.01x & otherwise \end{cases} \tag{1.7}$$

and its derivative as

$$f(x) = \begin{cases} 1 & \text{if } x >= 0 \\ 0.01 & otherwise \end{cases} \tag{1.8}$$

### Softmax

Softmax is generalization of Sigmoid function on a vector. It softly distribute its input into output that sums to 1 [21]. $i$–th element of output vector can be described as

$$\text{softmax}(x)_i = \frac{e^{x_i}}{\sum_j^M e^{x_j}} \tag{1.9}$$

where $M$ is width of input vector $i \in M$ and $x \in \mathbb{R}^M$.

### 1.2.3 Backpropagation

Goal of a Feedforward Neural Network is to approximate some function $f*$ [15]. The way to learn multi layer neural network to do that is through backpropagation.

Backpropagation is an algorithm that computes gradients of loss function also know as objective function or cost function [22], w.r.t. neural networks parameters. Loss function measures performance of an network by computing a loss sometimes called error. This allows utilization of optimization algorithms to minimize the loss during training by updating the parameters.

Backpropagation needs activation function to be differentiable or continuous and differentiable almost everywhere and that loss to be a scalar value [23].

We can utilize chain rule to compute corresponding gradients since layers of neural networks can be viewed as composition of functions [24].

### 1.2.4 Optimization

Neural network is learned in so called training loop. We iterate through training data and compute loss. There are three ways to iterate through the data. We iterate through every single datapoint and compute loss. Weights of neural network, that we train, are updated by negative gradients of loss w.r.t. weights of the neural network. This is called gradient Descent.

Second way is that instead of iterating through every single datapoint we split data into mini-batches, which is collection of datapoints. Average of losses of the mini-batch is computed and weights are updated by gradients of this average. This is called mini-batch gradient descent [25].

Third way is same as previous but average loss is computed through whole dataset. This is called batch gradient descent.

#### Gradient Descent

Gradient Descent is simplest form of optimization of neural network. It can be described as:

$$w = w - lr\frac{\partial E}{\partial w} \tag{1.10}$$

where $w$ is updated weight, $lr$ is magnitude of update called learning rate and $E$ is loss computed by loss function.

Gradient descent is not used since the negative of gradient indicates the direction of „steepest descent“ orthogonal to the level set at the current point but minimum is usually not in that direction [26].

#### Stochastic gradient descent

Stochastic gradient descent (SGD) is a generalization of a gradient descent by taking average of an mini-batch [27].

**Optimization algorithm with Adaptive Learning Rates**

There are also optimization algorithms that adapt learning rates during training to converge to a minimum smoothly. These algorithms include ADAM, RMSProp, AdaGrad etc. [28]

## 1.3 Convolutional Nerual Networks

Convolutional Neural Network (CNN) is type of nerual network that utilize discrete conovlutional operation [29, 30]. Convolutional Neural Network exploit property of natural data which is compositionality (world is hierarchical)[30]. If the $N$ neurons are fully-connected to its previous layer of size $M$, size of weight matrix of such layer is $N \times M$ which can be large based on the size of input. By stating two hypothesis about structure of data the need of fully-connected neurons disappear. Data have to have strong local correlations between values and features can appear anywhere in the image [29, 31].

One dimensional discrete convolution is defined as:

$$y_i = \sum_j w_j x_i - j \tag{1.11}$$

but in practice is used cross correlation instead defined as:

$$y_i = \sum_j w_j x_i + j \tag{1.12}$$

Which is same as convolution but kernel is not flipped relative to the input to preserve commutativity of convolution [32]. [deep learning,332]

Two dimensional cross-correlation is defined as

$$y_{ij} = \sum_{kl} w_{kl} x_{i+k,j+l} \tag{1.13}$$

vector or matrix W is called convolutional kernel which in Convolutional Nerual Network act as weights. Since the kernel is only of limited size it is considered as sparse connection between previous layer and neurons of the layer [32].

Instead of convolving by „one step“ we can change the definition of convolution and add stride [29].

Thanks to CNNs parameter sharing in form of sliding kernels, CNNs converge faster, generalize better, they are not constrained to input size, connection sparsity reduces amount of computation and are highly parallelizable because kernels are independent [32, 29, 30].

In Feedforward Neural Network are alternations of matrix multiplication and application of activation function. In convolutional neural network there is alternation of convolution, activation function and pooling which is optional.

### Pooling

Pooling layer is an layer that reduces dimensionality of convolved input and makes the representation approximately invariant to small translations of the input by using aggregation function [33] . It takes aggregate of small area of an input using sliding window as was done in convolution but with stride equal to its size so the input is not overlapping.

There are several types of pooling. Max pool takes maximum of an input area, Average pool takes an average etc.

## 1.4 Recurrent Neural Networks

Recurrent Neural Network (RNN) is neural network which its current state depends on its previous state. Its original form can be described by diagram 1.2

In order to train this type of neural network you have to perform back-propagation through time see fig. 1.3.

Recurrent neural networks described above do not work very well due to the problem of vanishing or exploding gradient. Every iteration of backpropagation in time gradients are multiplied by the the weight matrix $W$ and if the weights are small gradient becomes smaller. Therefore gradients will be exponentially smaller in time. On the other hand if the weights of weight matrix $W$ are high then the gradients get exponentially bigger in time.

Due to vanishing or exploding gradient Recurrent neural networks have issue to learn things from a distant past. This problem is called long-term

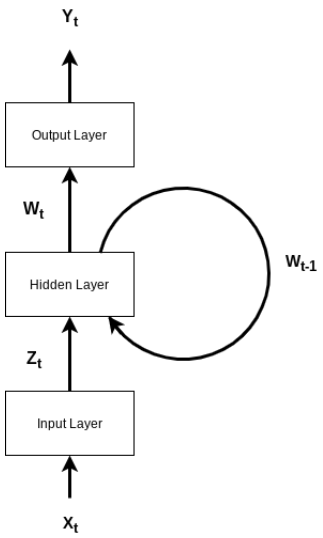



Figure 1.2: Recurrent neural network diagram with three layers, recurrent loop from previous timestamp t-1

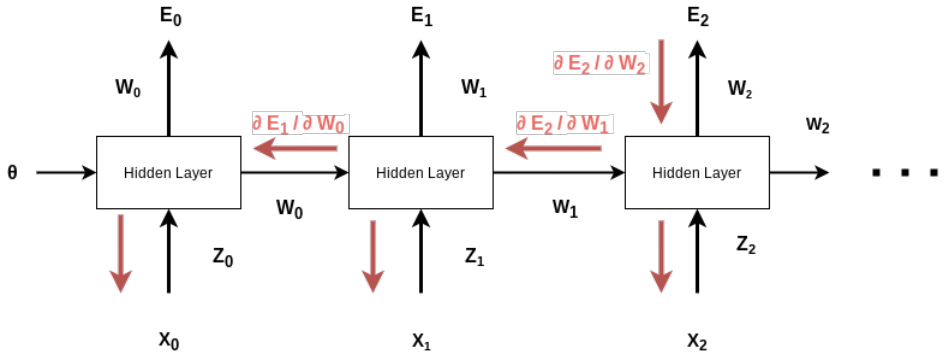



Figure 1.3: Backpropagation through time: In order to compute gradients of $E_2$ network has to first compute gradients of previous time stamps

dependency problem and was first described by Yoshua Bengio et al. in [9].

There are several ways to mitigate the effect of vanishing gradients in RNNs like like gradient clipping to avoid exploding gradient or vanishing gradient regularization for reducing vanishing gradient [34].

### 1.4.1 Long Short Term Memory

Long Short Term Memory (LSTM) is a type of Recurrent Neural Network. It is also used for sequential data but unlike RNNs it introduces gates to control LSTM cells state to mitigate vanishing or exploding gradient and therefore improve learning of long-term dependencies [35]. Single cell of LSTM can bee seen in 1.4

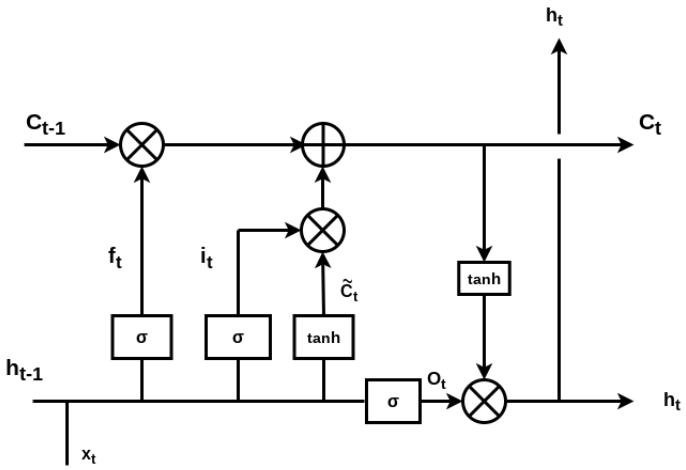



Figure 1.4: Single cell of LSTM with three gates $f, i, o$ which controls its state.

It can be described as:

$$
\begin{aligned}
f_t &= \sigma(W_f x_t + U_f h_{t-1} + b_f) \\
i_t &= \sigma(W_i x_t + U_i h_{t-1} + b_i) \\
o_t &= \sigma(W_o x_t + U_o h_{t-1} + b_o) \\
\tilde{c}_t &= \tanh(W_c x_t + U_c h_{t-1} + b_c) \\
c_t &= f_t * c_{t-1} + i_t * \tilde{c}_t \\
h_t &= o_t * \tanh(c_t)
\end{aligned}
$$

$c$ represents cells state, which is controlled by three gates. $f$ forget gate, $i$ input gate and $o$ output gate. Forget gate controls whether cells state should be reset. If $f = 0$ previous cell state is forgotten. Input gate controls if the cells state should be updated. If $i = 1$ then the cells state is updated by value of $\tilde{c}$ which lies between 1 and -1. Output gate controls visibility of the cells output. If the $o = 0$ than an output of the cell is completely hidden.

### 1.4.2 Gated Recurrent Unit

Gated Recurrent Unit (GRU) is a type of Recurrent Neural Network that like LSTM uses gates to control learning of long-term dependencies. GRU is simpler than LSTM and therefore computationally less demanding [36]. Single cell of GRU can be seen in 1.5

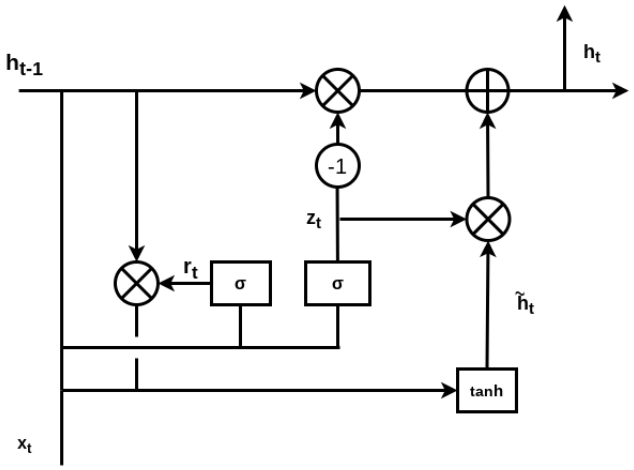



Figure 1.5: Single cell of GRU. It utilizes gates to control cells state as it is in the case of LSTM but GRU has fewer parameters.

GRU can be described as:

$$
\begin{aligned}
z_t &= \sigma(W_z x_t + U_z h_{t-1} + b_z) \\
r_t &= \sigma(W_r x_t + U_r h_{t-1} + b_r) \\
\hat{h}_t &= \tanh(W_h x_t + U_h(r_t * h_{t-1}) + b_h) \\
h_t &= (1 - z_t) * h_{t-1} + z_t * \hat{h}_t
\end{aligned}
$$

$x_t$ is an input vector, $h_t$ is an output vector, $\hat{h}_t$ is an candidate activation vector, $z_t$ is an update gate, $r_t$ is reset gate vector and $W, U, b$ are corresponding parameters of the cell.

$z_t$ is used to compute linear combination of two inputs ht-1 and $\hat{h}_t$ if $z_t = 1$ then $\hat{h}_t$ and if $z_t = 0$ then $h_{t-1}$ is used. In other words if $z_t = 0$ then copy previous state and ignore the input if $z_t = 1$ state of the cell is forgotten and you output $hath_t$. Reset gate $r_t$ resets previous state and if $r_t = 0$ previous state $h_{t-1}$ is not considered in the output $\hat{h}_t$.

## 1.5 Convolutional Recurrent Neural Networks

Convolutional recurrent neural network CRNN is combination of previously mentioned neural network types namely Convolutional Neural Network and Recurrent Neural Networks. CRNN are used in sequential data where state depends on previous states to describe its meaning. So they are used in music genre classification speech recognition etc. [37].

## 1.6 Residual Neural Network

Residual Neural Networks utilize skip connections in their architecture. It helps to train deeper neural networks since deeper neural networks are more difficult to optimize and easily fall to local minimum[38].

Residual connections can be seen in diagram in fig. 1.6.

## 1.7 Attention Mechanism

Attention Mechanism is a mechanism that control importance of different regions in the input. Attention Mechanism was first introduced for machine translation by Bahdanu et al. [39] in 2015. It was an application of attention on bidirectional recurrent neural networks .

In paper *"Attention Is All You Need"* [40] has been shown that there is no need for recurrent neural networks and only the self attention was used for machine translation. This model was trained without recurrence, in other words there was no backpropagation, which reduced computation needed to train the model.

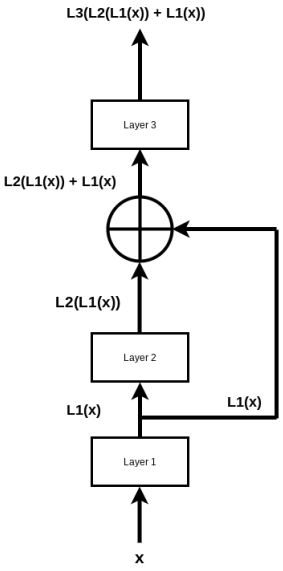
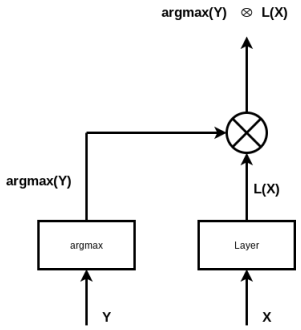



Figure 1.6: This diagram shows skipped connections present in Residual Network. It is worth to mention that connection can skip more then one layer as shown here.

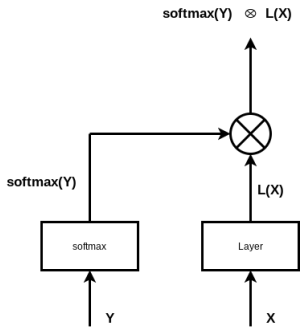
### 1.7.1 Basic Types of Attention Mechanism

#### Hard attention



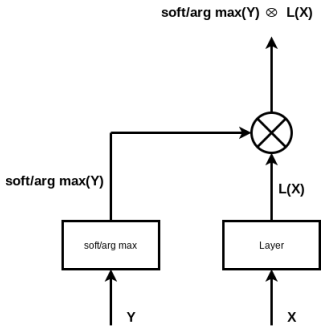

Selects only the most important region

### Soft attention



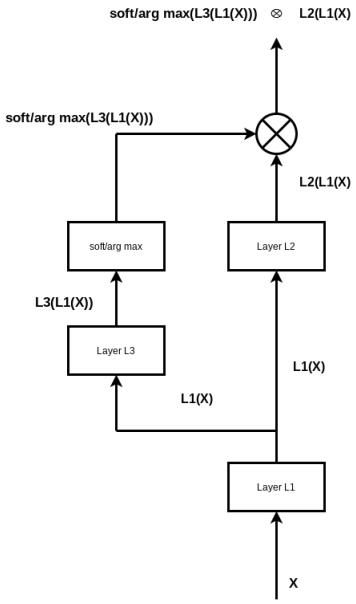

Soft Attention unlike Hard can select multiple important regions

### Cross attention




Input X and Y are two different inputs to a neural network.

**Self Attention**




Input to the attention branch is output of previous layer like in residual networks.

## 1.8 Autoencoder

Autoencoder is nerual network which is trying to reconstruct input on its output with minimal information loss, therefore it is considered as self-supervised model. Simplest architecture consists of encoder that is suppose to encode input into latent code and decoder that tries to replicate input from the latent code [41]. Diagram of an autoencoder can be see in fig. **??**.

This type of autoencoder is usually used for dimensionality reduction and unlike Principal Component Analysis (PCA) that can extract only linear features of a data, autoencoder can learn non-linear features but only if latent code is at least second hidden layer. However, if autoencoder consist only of input, hidden and output layer (commonly referred as shallow autoencoder)

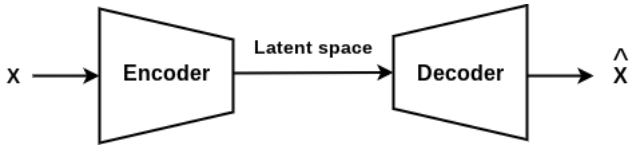



Figure 1.11: Autoencoder consists of Encoder and Decoder. Encoder maps input data X to a latent space. Decoder then tries to reconstruct X with minimal reconstruction loss.

then latent code can only describe basic representation as it is case with PCA. It was described by Marvin Minsky and Seymour Papert in their publication „Perceptrons: An Introduction to Computational Geometry“ [5] and experimentaly observed in work about sound reconstruction using autoencoders [42]

By introducing bottleneck autoencoder tries to code input into latent code features that are most important for reconstruction. Without bottleneck autoencoder would not be encouraged to do so as it would only copy input to output.

Latent space (space of all possible latent codes) of this simple autoencoder architecture is not continuous because nothing encourages it to do so which makes it difficult to explore. For example triangle inequality does not have to hold and therefore makes most of the clustering technique unusable. If we wanted to generate new datapoints by randomly initializing latent code and decode it with autoencoders decoder, we might get noise instead of realistic looking datapoint eg. Image that correspond to a used dataset for training, therefore various autoencoder architectures were proposed like Variational Autoencoders, Adversiarlly Constrained Autoencoder Interpolation etc. that address this issue [43, 44, 45].

### 1.8.1 Variational Autoencoder

Variational autoencder (VAE) is type of autoencoder that uses regularization to enforce desired properties of latent space mainly to generate realistic looking data by reconstructing certain latent code. Unlike autoencoder that maps input to single point in latent space Variational Autoencoder encodes it as normal distribution using mean and variance [45, 46] see fig. 1.12.

Unlike autoencoder that is deterministic for given input $x$. VAE is not deterministic for given input $x$. It is because output of VAE is reconstructed from latent code that is chosen randomly based on mean and variance. To be able to use backpropagation through sampled vector, this vector is calculated using parameterization trick $z = \mu(z) + \epsilon * \sqrt{\sigma(z)}$ where $\epsilon$ is random number from standard normal distribution [45].

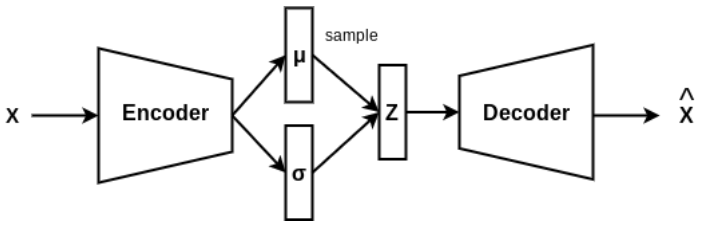



Figure 1.12: Latent variable Z is sampled from mean $\sigma$ and variance $\mu$. Decoder then tries to reconstruct it with minimal loss and Kullback-Leibler divergence enforces structure in latent space.

Loss of VAE is calculated as sum of reconstruction loss and discrete Kullback-Liebler divergence of latent code $z$ and standard normal distribution [45].

Reconstruction loss pushes points $z$ away from each other because closer they are bigger the probability of different inputs $x$ to be reconstructed as same output which will give higher reconstruction loss. Kullback-Liebler divergence tries latent code defined by $\mu$ and $\sigma$ to be as similar to standard normal distribution as possible which will try to push points towards center 0.

Thanks to the properties of its latent space Variational Autoencoder can be used for clustering [46].

### 1.8.2 Adversarially Constrained Autoencoder Interpolation

Adversarially Constrained Autoencoder Interpolation (ACAI) is method that improves latent space interpolation properties by using regularization procedure. It is based on idea that interpolation of two datapoints should appear realistic therefore it incorporates critic that guesses interpolation coefficient $\alpha$ which should always be 0 for realistically looking data [43, 44].

ACAI consists of encoder decoder and critic. During training encoder produces two latent codes $z_1$ and $z_2$ of two different inputs $x$. These codes are then interpolated and reconstructed by decoder. Critic then guesses interpolation based on reconstructed $x$ [43]. Whole procedure is described in diagram 1.13

Loss of the critic is described as

$$\mathcal{L}_d = ||d_w(\hat{x}_\alpha) - \alpha||^2 + ||d_w(\gamma x + (1-\gamma) g_\phi(f_\theta(x))||^2 \tag{1.14}$$

where $d_w$ denotes critic with parameters $w$, $\hat{x}_\alpha$ is reconstruction of interpolated latent codes $\alpha \in [0; 0.5]$ is interpolation coefficient, $\gamma$ is hyperparametr

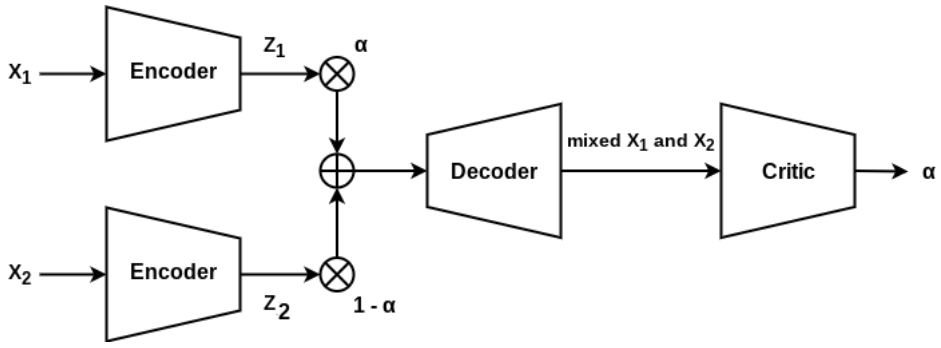



Figure 1.13: ACAI: Two latent codes produced by same encoder are interpolated together. Decoder tries to reconstruct realistically looking output of the interpolation. Critic acts as discriminator that tries to predict interoplation coefficient $\alpha$

for network stabilization, $g_\phi$ referes to decoder with parameters $\phi$, $f_\theta$ is encoder with parameters $\theta$ and $x$ is uninterpolated input [43].

First term of the loss is just an error of critics guess. Second term serves as a "clue" when should the critic start to take interpolations reconstructed by the autoencoder "seriously". It is a way to train the critic to recognize interpolations when the reconstruction of decoders are not good. It interpolates reconstructions with real datapoints by hyperparameter $\gamma$. This term is just to stabilize the model during training. Choice of hyperparameter $\gamma$ does not have any effect on reconstruction quality of interpolated codes [43].

Architecture of the Critic is similar to encoders architecture and $\alpha$ is computed as mean of the output of last layer [43].

Loss of autoencoder is describes as

$$\mathcal{L}_{f,g} = ||x - g_\phi(f_\theta(x))||^2 + \lambda||d_w(\hat{x}_\alpha)||^2 \tag{1.15}$$

as proposed in [43].

First term is just reconstruction loss. Second term is regularization term that forces autoencoder to output reconstructed data to appear realistic. Higher the output of a critic higher is the loss of the autoencoder [43].

Unlike VAE that directly optimizes latent space using Kullback-Lieber divergence, ACAI optimizes latent space to interpolate using adversary, namely critic, indirectly [43].

Thanks to its properties, Latent space of an autoencoder that uses ACAI method can be used for clustering despite that it was intended as an generative model as is shown in [44].

## 1.9 Obstacles and Advantages in Sound Classification in Comparison to Image Recognition

Since many techniques of deep learning used for sound classification comes from image classification these two domains will be compared.

In contrast to image recognition main obstacle in sound classification is overlapping of sounds that corresponds to different sources in frequency domain and amplitudes of same frequencies in certain time.

If classification is based on raw unprocessed signals then we deal with superposition of signals (signals with different sources are stacked on each other). If however, we project sound into frequency domain then overlapping of amplitudes applies only when sounds of different sources share common frequencies.

But still, main obstacle is to recognize which frequencies correspond to which sound source in given time frame. Therefore segmentation of the spec-

trogram has to be done in order to distinguish frequencies corresponding to different sources.

Due to hierarchical or semantic nature of symbolic representation of certain domains [44], like speech, bird songs or genre of music, sound classification can become a difficult task. Because in order to recognize spoken words, species of a bird or the music genre we start to deal with long sequences of a time series.

This does not necessarily have to apply to instrument classification. Identifying sounds does not have to depend on bigger structure when it comes to classifying non-semantic data like instruments. We should be able, based on few milliseconds time frame, to classify what instrument is present in the sound, thanks to instruments timbre. However higher frequencies decay faster then lower frequencies in atmosphere and in materials that instruments are made of, so timbre changes over time [47].

Sound has generally lower dimensionality then images, even that human ear can hear about 20000 different frequencies. ImageNet dataset [48] has size of $482 \times 418$ pixels in average. However dimensionality greatly depends on type of data that classification uses since sound signals can have different sampling rates and different pre-processings like mel-spectrogram can have different dimensionality based on their parameters. For example 4 seconds long sound signal sampled by 44.1kHz will have 176400 samples.

Observation of sound is much more difficult then observation of single images which makes evaluation of certain aspect difficult. For example if we have generative model that tries to generate sound it is much easier to just look at image it produced then to listen to sound. It takes more time and it is difficult to compare then to other generated sounds. However that might be said for all time series data.

Chapter 2

# Audio Pre-processing

## 2.1 Discrete Fourier Transform

Discrete Fourier Transform (DFT) commonly referred as Fast Fourier Transform (FFT) (thanks to widespread usage of fast algorithms that has computational complexity $\mathcal{O}(n * log(n))$ instead of $\mathcal{O}(n^2)$) is transformation from time series to frequency domain. It is common technique to get frequency components that are present in a signal [49].

In context of audio signal pre-processing it can be described as

$$\hat{f}_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi n \frac{k s_r}{N}} \tag{2.1}$$

Where $\hat{f}_k$ is k-$th$ fourier coefficient, $N$ is length of examined signal, $x_n$ is n-$th$ amplitude value $n \in \{0..N-1\}$ and $s_r$ is sampling frequency of the audio signal. Fourier coefficients $\hat{f}_k$ are then complex numbers corresponding to center frequency $f = \frac{k s_r}{N}$ in Hz for $k \in \{0..\lfloor \frac{N}{2} \rfloor + 1\}$. Absolute value of the complex number gives magnitude of that frequency and its angle gives phase shift of the frequency. The magnitude is usually used as an input feature for machine learning.

It is worth to note that range of frequencies that can be obtained is bound from 0 to $\frac{s_r}{2}$ due to Nyquist Criterion. So for $k > \lfloor \frac{N}{2} \rfloor + 1$ corresponding center frequency $f$ of k-$th$ Fourier coefficient can be calculated as $f = N - \frac{k s_r}{N}$.

## 2.2 Short-time Fourier transform

Short-time Fourier transform (STFT) or (STFFT) is a popular technique used to pre-process audio signal. STFT transforms from Amplitude domain to frequency domain like DFT. But unlike DFT it is using floating windows that each performs DFT. Because DFT on whole signal would not describe local occurrences and changes of frequency in time [49].

Spectrogram is visual representation of frequencies in the time and is usually produced by STFT. Deep learning models from image classification can be used for audio thanks to its image-like representation.

## 2.3 Mel-spectrogram

Mel-spectrogram, maps frequencies produced by STFT to Mel-scale using Mel-filter banks, which is scale that tries to mimic human perception of pitch intervals [50].

It describes that difference in lower frequencies is more distinguishable than difference in higher frequencies. It is worth to note that it is based on empirical evidences and the interval perception might be subjective but it give us estimate on how to linearize frequencies based on human perception. This property of Mel-spectrogram makes it suitable pre-processing tool for Convolutional Neural Networks since they require features to have same size anywhere in the input data [51]. CNNs can be tolerant to changes in location and rotations [52].

Transformation from linear frequency scale to Mel-scale can be described as

$$\text{Mel}(f) = 1125 \ln(1 + \frac{f}{700})$$

Mel-spectrogram is obtained by mapping spectrogram produced by STFT onto Mel-scale, using triangular overlapping windows also called Mel-filter banks.

## 2.4 Decibel Scale

Because ear does perceive intensity of sound logarithmically. Magnitudes of frequencies of spectrograms are transferred to decibel scale. Lower frequencies

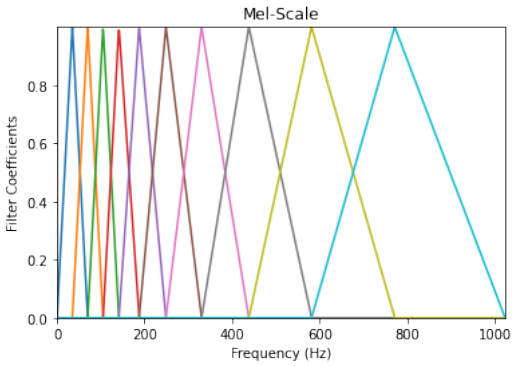



Figure 2.1: Triangular Mel-filter banks spaced according to Mel-scale. Only 10 bins are displayed for illustrative purposes.

are less louder then higher frequencies with same amplitude due to difference of their intensities.

## 2.5 Mel-frequency Cepstral Coefficients

Mel-frequency cepstrum coefficients (MFCC) takes the discrete cosine transform of individual frequencies in time of previously mentioned Mel-spectrogram in decibel scale.

Purpose of MFCC is to represent power spectrum in short time frame which is crucial for certain sound events, like speech.

Discrete cosine transformation compresses the Mel-spectrogram which results in lower dimensionality of input data.

## 2.6 Constant-Q transform

Musical instruments that produce tones with low degree of inharmonicity, eg. flute, which means that frequency of overtones are exact multiple of fundamental frequency. Frequencies present in music are strictly given by musical scale. Therefore overtones of such instruments will be present at frequencies that are multiple of this scale [53, 54, 55].

Constant-Q transform sets Fourier transform basis to match these frequencies. Constant-Q transform is a method to transform signal from time domain to frequency domain as does discrete Fourier transform but instead of setting centre frequencies linearly, like it was the case of discrete fourier transform, constant-Q transform selects them logarithmically according to musical scale [53, 54, 55].

Center frequencies can be calculated as

$$f_\mu = f_{min} 2^{\frac{\mu}{b}} \tag{2.2}$$

*where $f_min$ is chosen as the first center frequency, $b$ defines the number of bins per octave interval, and $\mu$ represents the number of semitone intervals beyond $f_{min}$* [53, 54, 55].

## 2.7 Other Preprocessing Techniques

### Gammatone filter banks

Gamma tone filter is like Mel-filter an approximation of human perception but unlike Mel-filters it is based on model of human auditory system [56].

### Learned Filter Banks

In contrast to mel-filter banks it is possible to learn filter banks within a neural network as was shown in [57].

## 2.8 No Pre-processing of Audio Signal

It is possible to train deep learning models on raw audio signal. If model is trained on pre-processed audio signal with e.g. Mel-spectrogram model learns only the features presented in Mel-spectrogram that should not be issue in classification. However if the model is suppose to generate sound then it will learn pre-processed representation. In case of autoencoders it will learn to reconstruct the Mel-spectrogram. But sound by pre-processing information is lost in the process which reduces quality of reconstructed audio signal.

Models like WaveNet uses 1D dilated convolution layers which are convolutional layers with sparser kernels (kernels with blank spaces) but with larger receptive fields [58].

There are also classification models that uses raw audio signal. They usually utilize 1D convolution layers [59, 60].

## 2.9 Data Augmentation

Data augmentation is process to apply slight alterations to the data in dataset to increase its size and variation. It can be applied to datasets where is only limited amount of datapoints for particular class which is usually case in sound classification.

Basic types of data augmentation for sound includes pitch shift which lowers or raises pitch of an audio sample, time stretching which slows or speeds up audio sample and adding noise to audio samples [61].

Chapter 3

# Analysis and Design

## 3.1 Dataset

Main goal of this work is to train model on aggregated data from YouTube based on search results of specific query. Model is supposed to train to classify musical instruments. This type of data is easily obtained but there might be problems due to copyright-protected material that videos include. Since content of the videos is not going to be reused but used solely for training it should not be an issue.

The quality of obtained data is another issue with the dataset. Recordings are very noisy in most cases so the sound does not contain only desired instrument but other instruments as well. This is not surprising in case of music where multiple instruments co-occure. Different instruments have different level of noise surrounding it since some instrument stand out more in records or they are used as solo instruments more then others.

Also, there is no guarantee that desired instrument is present in the recording or that there are duplicated recordings in the data which makes balancing the dataset very difficult.

Dataset used for testing should be labelled, should not contain too much noise and all the labels have to be correct. Two datasets that fulfil this criteria have been found. Dataset Nsynth containing 305979 of 4 seconds, 16kHz recordings of musical instruments [62] and IRMAS [63] which contains musical audio excerpts with annotations of predominant instruments present in the audio [63].

There are also datasets like OpenMIC-2018 [64] and Audio Set [65]. But these datasets are annotated YouTube videos which might be in conflict with obtained videos from search results. And it is also more difficult to obtain them.

## 3.2 Pre-processing

It is possible to train deep learning model on an unprocessed audio signal but it would be more computationally demanding due to higher dimensionality. It might be more convenient to train the model on carefully selected hand crafted features like spectrogram produced by STFT, Mel-spectrogram or constant-Q transform (all of them in decibel scale). These types of pre-processing techniques has disadvantage in comparison to raw signals. For example, if we consider generative models that are suppose to reconstruct generated signals so that they can be observed and examined. Information loss created by them might harm overall quality of the sound, as shown in the experiments. On the other hand it should not be an issue for classification but it makes evaluation of some models like autoencoders more difficult.

Spectrograms produced by different methods like STFT, Mel-filter banks or constant-Q transform has their own advantages.

Spectrogram produced by STFT has lower information loss then Mel-spectrogram making reconstruction of the sound back to waveform the easier. Due to the fact that people perceive pitch distance logarithmically some models like CNN cannot be used or at least the requirement of CNN that features have to appear anywhere in the input. The frequency domain features would have different shape with respect to different frequency if we do not linearize.

Mel-spectrogram linearize the frequency as it is perceived by humans. Which makes it possible for CNN based models to be used since sounds of slightly shifted pitch will have similar features. However, if we want to linearize musical features then it might be better to use musical scale instead which is also logarithmic in frequency but it differs from Mel-scale. Other issue is that resolution of Mel-spectrogram relies on resolution of short time Fourier transform. Increasing bin count of Mel-filter banks might not produce better spectrogram resolution. In addition to the linearization it also reduces dimensionality based on number of specified bins.

Constant-Q transform addresses this problem that it linearizes frequency domain during application of Fourier transform. It centres analysed frequencies around frequencies that correspond to musical scale. Problem with using such scale is that it emphasize on frequencies that are placed on musical scale and therefore predominates instruments with low inharmonicity. On the other hand drums like most percussion instruments have very high degree of inharmonicity. Guitars that usually have low degree inharmonicity have high degree of inharmonicity when distortion is applied what is common in rock music genre. Therefore this pre-processing technique might be biased towards instruments with low degree of inharmonicity discriminating those that have higher degree inharmonicity or instruments that are not properly tuned. Other issue is that constant-Q transform is more computational demanding then previously mentioned pre-processing techniques.

## 3.3 Different Classification and Supervision Types

### 3.3.1 Supervised Classification Models

Due to the noise level and imbalance that might occur inside the dataset. Models which depend on strongly labelled data might be insufficient or would underperform. Even if their performance would not be as good as they were trained on strongly labelled data, information about the noise might be lost and its recognition can be difficult.

### 3.3.2 Anomaly Detection Models

There are two categories of anomaly detection. Supervised which tries to classify anomalous data and data that are considered „normal“. This can be seen as supervised binary classification as discussed in previous paragraph. The other category can be seen as one class classification also known as unsupervised anomaly detection. Deep learning models that are used to solve this problem extract features from data of single class and then try to predict anomalies based on the extracted features. However they sometimes rely on the idea that anomalies are rare and their structure is not that complicated, which does not hold in this case.

### 3.3.3 Unsupervised Models

Unsupervised models try identify interesting datapoints based solely on underlying structure describing the data. In unsupervised learning, there is no apparent straightforward cost function that can directly distinguish individual classes [44]. Deep learning therefore uses cost function that measures reconstruction or similarity score [66] to extract important features. These features are then usually clustered or other algorithm is used that decides appropriate class. One example is SimCLR that uses contrastive learning that relies on data augmentation [66]. Online Deep Clustering for Unsupervised Representation Learning proposed by [67] uses pseudo labels produced by K-Means algorithm. Dynamic Autoencoder (DynAE) uses pseudo labels produced by K-Means clusters in latent space of ACAI.

Conventional clustering is typically performed on fixed features [67]. Deep neural networks can help capture hierarchical structure of features [44]. Utilization of latent space in regularized autoencoders, like it was used in DynAE, can help recognize noise in the data unlike in case of previously mentioned methods.

## 3.4 Proposed Methods

Audio signal will be pre-processed using STFT spectrogram or Mel-spectrogram with amplitude in decibel scale to reduce complexity of the models used to

extract features.

Datapoints will be represented as single sample of spectrogram. Using multiple samples from a given time frame would increase complexity of a model and it adds additional feature which is shape of a tone in time. Model then might give importance to changes of pitch in time rather then timbre of a tone that should describe an instrument.

Autoencoder was chosen to learn appropriate features from the data. Classification and recognition of conflicted data (datapoints that does not correspond to its search query) will be performed on their latent space. Therefore the latent space needs to be regularized to enforce smaller distance between latent codes of similar datapoints. So ACAI or VAE are good candidates.

After Autoencoder is trained on the data, clustering of latent codes of every datapoint from dataset will be performed to identify classes based on its respective query. For example if some cluster has majority of datapoints found by query piano, then the region of that cluster coresponds to piano. K-means was chosen as clustering method due to its scaling properties and known methods to identify number of clusters.

Properties of regularized autoencoder will be compared to same architecture of autoencoder but without regularization.

If latent space will have appropriate properties, vector arithmetic on latent codes might be performed based on their corresponding clusters to identify potential conflicts in the data.

Chapter 4

# Realisation

## 4.1 Used Technology

### Programming Languages

All programs where written using Python programming language and bash. Jupyter Notebook was used as environment performed tests and training of deep learning models. Bash was used usually for moving the data, for applying chains of scripts written in python that pre-processed the data or for applying programs to change format of the data.

### FFmpeg

FFmpeg is the leading multimedia framework, for decoding, encoding, streaming, filtering etc. videos, audio and other multimedia [68]. In this work it was used for converting downloaded videos to wav format.

### Youtube-dl

Youtube-dl is a command-line program to download videos from YouTube.com [69]. It was used to download youtube videos based on list provided by Youtube Data API

### Used Python Libraries

#### YouTube Data API

YouTube Data API is API allowing to upload videos, manage playlists and subscriptions, update channel settings etc. [70]. The API also allows to search for videos matching specific query which was used in this work to create dataset of instruments.

### Librosa

Librosa is python library for music and audio analysis [71]. All the pre-processing procedures like obtaining Mel-spectrogram, applying STFT or just loading the wav files were done by the Librosa.

### PyTorch

PyTorch is an open source machine learning framework mainly used for deep learning [72]. PyTorch allows to define deep learning models and train them using PyTorch tools. It has automatic differentiation engine that computes gradients automatically. Optimization algorithm like ADAM. Dataloader that allow to load data using parallelization.

Deep learning models proposed in this work were all modeled and trained using pytorch.

### Additional Libraries

NumPy was used mainly for storing pre-processed data in memory maps. Previously mentioned libraries Librosa and PyTorch work with NumPy arrays. Pandas for keeping track of information about acquired data. Scikit-learn for the implementation of K-Means.

## 4.2 Procedure

List of phases of the whole procedure:

1. Obtain video list based on appropriate search query.
2. Download videos based on the list
3. Convert obtained videos to monophonic wav files with sampling rate of 16kHz.
4. Create an file to collect info about audio files.
5. Create memory maps of pre-procesed sounds.
6. Train models on pre-processed sounds.
7. Translate datapoints into latent code.
8. Examine latent space using K-Means clustering.

### 4.2.1 Acquiring the Data

YouTube Data API v3 was used to return list of videos by given search query.

These queries included names of instrument classes present in IRMAS dataset [63]. Included instrument classes are: cello, clarinet, flute, acoustic guitar, electric guitar, organ, piano, saxophone, trumpet, violin, and human singing voice. However in my case electric and acoustic guitar was considered as one class same as is in NSynth dataset [62]. So search query 'electric guitar' or 'acoustic guitar' were not included but instead 'guitar' was used.

Results of search queries were then observed if they contain relevant data. In some cases like in query 'organ' resulted list did not give appropriate search results but if word 'music' was added at the end search results become more relevant. Therefore quality of the data can be enhanced by careful choice of search query.

Obtained video lists were downloaded with youtube-dl library [69] in m4a or webm format that was later converted using FFmpeg tools into monophonic wav audio files sampled by 16kHz.

Csv file was created to describe expected instruments and sizes of corresponding spectrograms.

### 4.2.2 Data Pre-processing

Every single recording was pre-processed into spectrograms using Short-time Fourier transform with FFT floating window wide 1024 samples and step size 512 mapped onto decibel scale. Output of short time Fourier transform is then array $\frac{l}{512} + 1 \times 513$ where l is number of samples. If the time series has 64000 samples, which is the case for sound excerpts from NSynth dataset, then resulting array would have size $126 \times 513$. Every array of an music recording by instrument was then concatenated into single memory map resulting in 10 memory-mapped files. Sum of all the file sizes were about 300GB.

Every frequency vector produced by STFT was then randomly shuffled into 10 files of same size. Therefore every file contained random selection of vectors from every instrument. This shuffling process was intended for unsupervised learning of ACAI model preventing similar vectors to be next to each other in the file. Reason to have multiple files was simply due to limited amount of fast storage on the computer where the files were transferred during training. PyTorch dataloader can randomly selects items from dataset but they were shuffled to keep distribution same across all files.

During training of ACAI model elements of each vector were scaled to values between 0 and 1 from scale -80dB to 0dB.

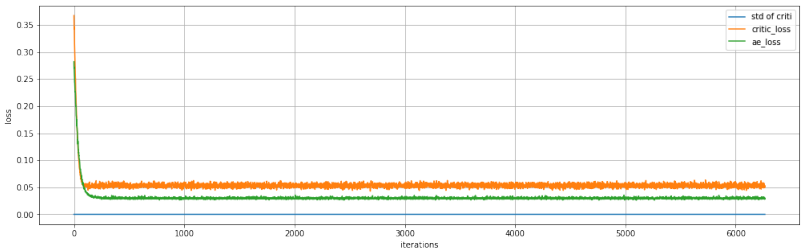



Figure 4.1: Example of training progress using Sigmoid as activation function in hidden layers. Standard deviation of predicted $\alpha$ is low indicating low stability of model.

### 4.2.3 Training on Data Aggregated From YouTube

First architecture of ACAI was trained on the pre-processed sound. Encoder E(·) comprised FC(Input, 256, ReLU), FC(256, 128, ReLU), FC(128, 64, ReLU), FC(64, 32, ReLU) and FC(32, 16, Sigmoid). The decoder network D(·) incorporated FC(16, 32, ReLU), FC(32, 64, ReLU), FC(64, 128, ReLU), FC(128, 256, ReLU) and FC(256, Output, Sigmoid) where FC($a, b, f$) represents a fully-connected layer with input neurons $a$, an output layer $b$, and activation function $f$. Critics architecture was same as of encoder E(·) and $\alpha$ was calculated as mean of its output.

### 4.2.4 Effects of Different Activation Functions on Stability

#### Sigmoid Function

If Sigmoid function is used instead of ReLU for hidden layers activation function. Autoencoder and Critic always converge. In some cases critic is not able to produce different guesses of $\alpha$. Its standard deviation per mini-batch can be seen in fig. 4.1.

#### Hyperbolic Tangent

Experiments and results using hyperbolic tangents were similar to the results when using Sigmoid function critic has fallen to local minimum and was not predicting variety of $\alpha$s.

#### ReLU and LeakyReLU

Stability of the model improved by using ReLU or LeakyReLU which both has shown similar results see fig 4.2.

Because Sigmoid and Hyperbolic tangent did not show promising results during training LeakyReLU was used in further experiments with except to latent space and output of autoencoder.

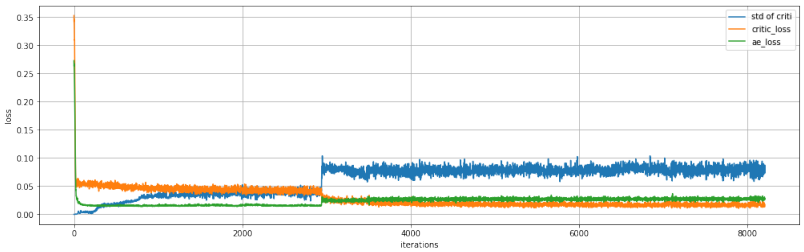



Figure 4.2: Example of training progress using ReLU as activation function in hidden layers.

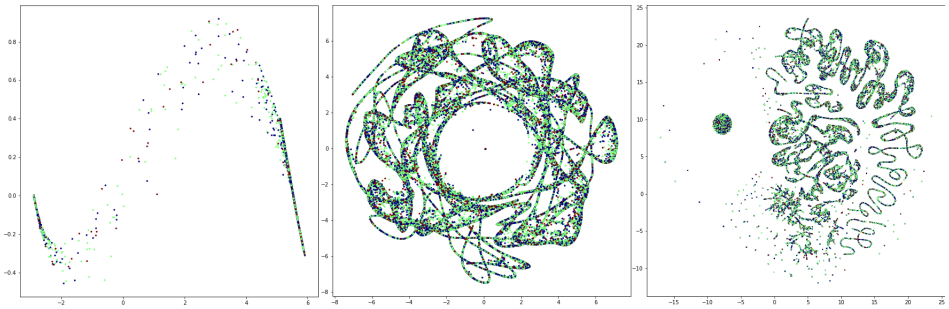



Figure 4.3: Latent space of first proposed model visualised in PCA, TSNE and UMAP. Individual frequency vectors are mapped into latent space and displayed in 2D. Instrument classes are distinguished by different color.

### 4.2.5 Testing the Model

In the end dataset IRMAS was not used, due to weakly labeled properties of IRMAS [63]. Labels correspond to predominant instruments in entire audio excerpt which contains recordings of multiple tones of instrument and in most cases multiple instruments are present. Therefore it is not known which time frames correspond to which instrument. Subset of dataset NSynth whose datapoints contain only single tone of single instrument, was used instead to visualise and test latent space with corresponding labels. The Subset included classes of instruments from guitar, human singing voice referred to as 'vocal', organ and flute.

Despite good reconstruction and prediction losses, latent space of this architecture behaved unexpectedly as is shown in its visualisations 4.3.

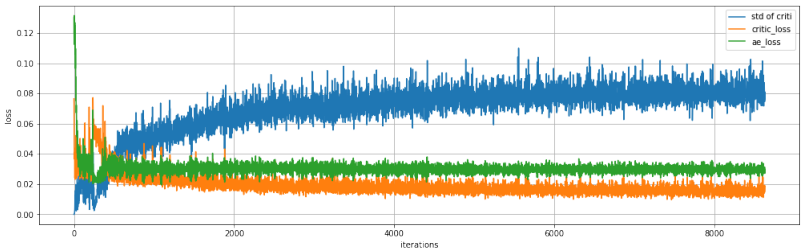



Figure 4.4: Training progress example of architecture used in [44]. Standard deviation of predicted $\alpha$ around 0.10 which indicates variety of guesses.

## 4.3 Second test

Architecture from [44] was used instead of previously proposed architecture. Encoder E(·) comprised FC(Input, 500, ReLU), FC(500, 500, ReLU), FC(500, 2000, ReLU), FC(2000, 10, none). The decoder network D(·) incorporated FC(10, 2000, ReLU), FC(2000, 500, ReLU), FC(500, 500, ReLU) and FC(500, Output, none) where FC($a, b, f$) represents a fully-connected layer with input neurons $a$, an output layer $b$, and activation function $f$. Critics architecture was same as of encoder E(·) and $\alpha$ was calculated as mean of its output.

To test whether usage of Mel-spectrogram have any impact on the properties of latent space, additional memory-mapped training file was created.

Similarly to previous method of creating memory-mapped Mel-spectrogram of every sound recording was created. However this time including only guitar, vocal, flute and organ instruments. Size of FFT floating window was set to 2048 with step size of 512 which resulted in same length of spectrogram as with STFT method. Number of Mel-filter bank bins was set to 512. Dimensions of the Mel-spectrogram were then $\frac{l}{512} + 1 \times 512$ where $l$ is number of samples. Resulted Mel-spectrogram was then scaled to decibel scale. Mel-spectrograms were then concatenated and shuffled to single file.

### 4.3.1 Training

Second architecture was trained on spectrogram and Mel-spectrogram data. All model converged and critics guesses had around 0.10 standard deviation per mini-bath see fig. 4.4.

After model was trained on the YouTube data. Nsynth datapoints were then mapped into latent space. Every single NSynth recording was transferred into spectrogram or Mel-spectrogram scaled in decibels, accordingly to the models pre-processing and from these spectrograms vector with the highest energy was selected to be projected into latent space.

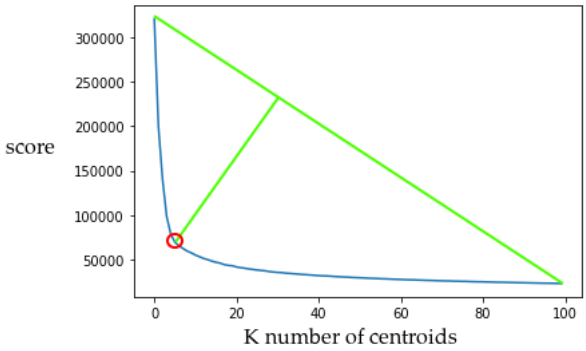



Figure 4.5: Visualisation of elbow method, red circle indicates selected $k$ which is furthest point from imaginary line connecting first and last $k$.

### 4.3.2 K-Means

Projected vectors were then clustered using k-means to see the properties of models.

#### Number of Centroids Estimation

Reason that number of centroids can not be set to be equal number of instruments is because it is assumed that there is going to be more clusters for each instrument in latent space.

Elbow method of squared distances to their respective centroids (in Scikit-learn referred as inertia) was selected to estimate number of centroids for k-means. Reason being is that all other methods like silhouette method did not scale well. Despite that it was not feasible to compute all k up to number of datapoints. Maximal k of 500 and 100 was selected. To test whether elbow method gives good estimate of optimal k, Rand Index RI, Normalized Mutual Information NMI and Purity was calculated along for every k based on instrument labels. Computational time of maximal k 500 was 8 hours.

#### Elbow Method Implementation

Implementation of elbow method consisted of drawing imaginary line between first point and last point of plotted inertia values. Scores of every k was calculated as distance to the imaginary line and selected k. k with maximal score was then selected as optimal number of centroids see fig 4.5.

However as was observed, selected $k$ by elbow method depends on maximal $k$. Larger maximal $k$ prolongs the imaginary line, changes its angle and hence

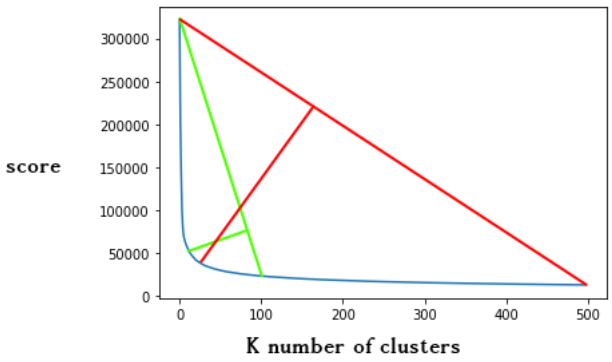



Figure 4.6: Effect of maximal $k$ on selection of optimal centroids. When maximal $k$ was selected as 100, 7 centroids was chosen as optimal and for maximal $k = 500$, 25 was selected instead.

changes the scores of $k$s. Larger is the maximal $k$ then Elbow Method selects larger number of centroids as optimal. Whole situation is pictured in fig 4.6

Elbow method presented here did not give good estimate of optimal number of centroids. Therefore to test the properties of latent space for every k was observed and Rand Index (RI), Normalized Mutual Information (NMI) and Purity was considered as evaluation. These metrics were then observed in contrast to selected $k$ by Elbow method based on maximal $k$ to evaluate its predictions. See figs. 4.7, 4.8 and 4.9 that consider $k$ predictions in ACAIs, Autoencoders and Randomly Initialized autoencoders latent spaces.

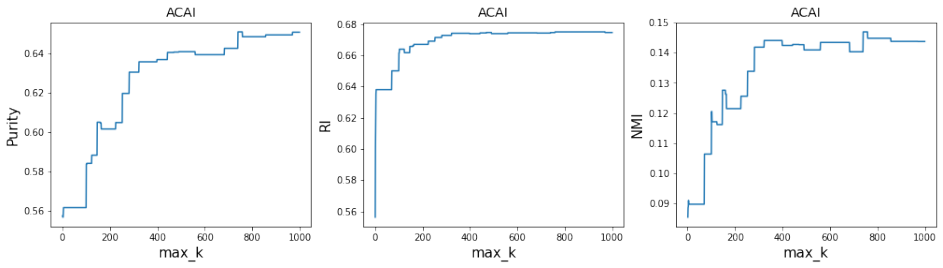



Figure 4.7: Elbow method predictions on ACAIs latent space. For maximal $k > 400$ Elbow methods gives good predictions about number of centroids in all metrics.

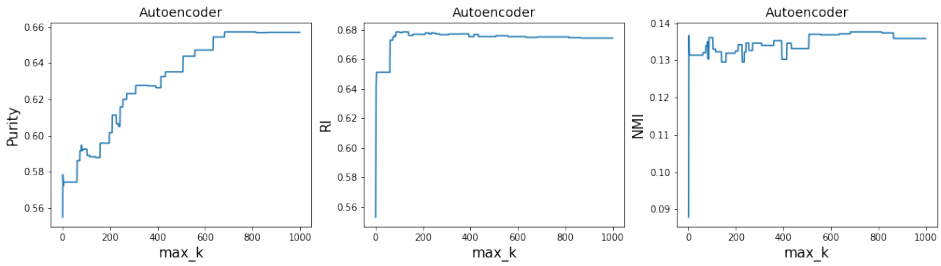



Figure 4.8: Elbow method predictions on Autoencoders latent space. Elbow methods gives good predictions almost immediately for RI and NMI. Purity grows with higher maximal $k$.

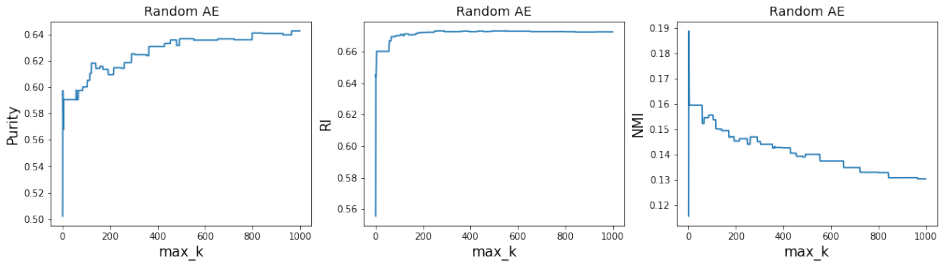



Figure 4.9: Elbow method predictions for Autoencoders latent space. Purity and RI grows rapidly as maximal $k$ rises. NMI on the other hand rapidly falls as maximal $k$ rises.

### 4.3.3 Test Results

ACAI and autoencoder with similar architectur but without regularization was trained on the spectrogram and the Mel-spectrogram both in decibel scale. Nsynth data were projected on their latent spaces and clustered using K-Means afterwards. Rand Index, Normalized Mutual Information and Purity of clusters were measured for every k up to maximal k. Selected visualisations of latent spaces can be seen in Appendix A. All measured results are summarized in fig 4.10. All the autoencoders showed similar results suggesting that regularization in ACAI did not have effect on clustering performance of latent space. Autoencoder and randomly initialized autoncoder even outperformed ACAI in NMI. However, additional tests have to be performed to assess consistency of the results which is difficult due to computational demand of whole procedure.

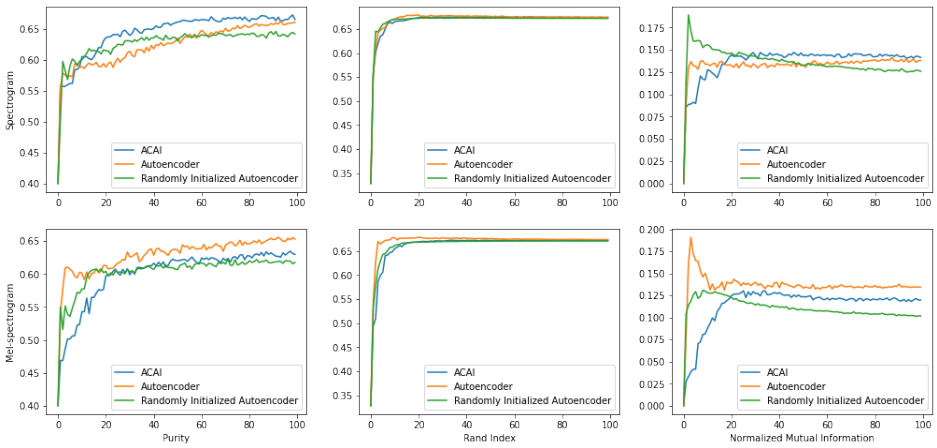



Figure 4.10: Comparison of ACAI, Autoencoder and Randomly Initialized Autoencoder. All Autoencoders have similar results, which indicates no improvment of regularization present in ACAI. There does not seem to be an improvement in using Mel-spectrogram over spectrogram in our case. Top row is depicting scores for spectrogram and bottom for Mel-spectrogram. Left column depicts Purity scores, middle column Rand Index and right column Normalized Mutual Information.

### 4.3.4 Evaluation of Results

As can be seen in the the figure 4.10 ACAI did not performed better then autoencoder which was as good as randomly initialized autoencoder. It is needed to make additional tests to assess what is the underlying reason. Visualisations can not be taken as an evaluation of ACAIs performance. And metrics of K-Means clusters does not give us much idea of the properties of its latent space.

There might be reasons why ACAI did not performed better then simple autoencoder.

The way datapoints are mixed in the ACAI is natural to sound. Multiple instruments are overlapped in a single time frame and the datapoints have different amount of volume which is not the case for images. Only case it might happen in images is when the objects would have different amount of transparency which happens rarely. This property of sound can result in multiple correct $\alpha$ and therefore can have negative effects on regularization of latent space. For example let us consider two single tones of instruments bass and guitar mixed together with $\alpha = 0.5$ then critic can correctly guess $\alpha = 0$ realistic (these instruments usually play together) or $\alpha = 0.5$ those tones are mixed. This might be partially solved by considering only time frames with highest volume using strongly labeled data. Different sound intensities might be an issue for prediction of $\alpha$. But additional research in this regard has to

be made.

However, clusters that emerged from this randomly initialized autoencoder achieved good scores in reaching 65% Rand Index and Purity. This suggests that frequency vectors produced by STFT or Mel-spectrogram showed that similar frequency vectors grouped into clusters. This might indicate that there are strong regularities among them. That might be used as a stepping stone for further development.

## 4.4 Additional Experiments - Effects of Pre-processing on Reconstructed Sound

In order to observe information loss of spectrogram produced by STFT and Mel-spectrogram. They were applied to single sound recording of an piano composition and then reconstructed back to sound signal with inverse procedures.

Audio pre-processed with STFT sounded like the original audio with added noise. But piano present in the audio was recognizable. Mel-spectrogram on the other hand has clearly changed pianos timbre. It sounded more like an bells rather than piano and the whole composition resembled background music of an carol.

However perception of reconstructed sound is subjective so it can not be used as measurement. Its purpose is just to point out that using pre-processing might not be well suited for generation of new sounds.

# Conclusion and Outlook

The review section of the presented thesis summarized the state-of-the-art approaches to deep learning. Emphasis was put on self-supervised autoencoders and their application in image and sound processing. Audio recognition techniques were compared to image classification. Commonly used pre-processing approaches were also discussed.

The practical part of the thesis concerned audio classification of a weakly-labeled sound excerpts. YouTube platform was used as a suitable source of such data. Self-supervised autoencoder with regularized latent space was chosen as an appropriate model. Adversarially Constrained Autoencoder Interpolation seemed to be a good implementation of such model. Multiple architectures were tested within this method. Initial architectures that utilized Sigmoid function in encoders latent layer did not have good latent space properties as evident from their visualizations. Therefore, architecture with linear embedding in latent layer was tested.

K-means clustering was performed to assess final performance of this model. This model did not show promising results in comparison to randomly initialized autoencoder. However, clusters that emerged from this randomly initialized autoencoder achieved good scores. This suggests that frequency vectors produced by pre-processing techniques presented in this work showed that similar sounds obviously grouped into clusters. This might indicate that other methods than those based on neural networks can be suitable for sound classification.

To train self-supervised model on this type of data different objective functions probably have to be applied. Rather than trying to mix two latent codes as implemented in Adversarially constrained autoencoder interpolation, it might be more suitable to find objective function that could evaluate quality of sound decomposition. Sound excerpt would be decomposed into multiple audio components which should appear realistic. Unrealistic sounds could be penalized by adversary. Another possibility would be to use distributional

similarity based representations as a measure of similarity between individual vectors produced by pre-processing. This idea is widely used in natural language processing.

# Bibliography


1. ROSENBLATT, Frank. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*. 1958, vol. 65, no. 6, pp. 386–408.
2. SAEED, Khalid; HOMENDA, Wladyslaw (eds.). *CISIM*. Vol. 9842, Computer Information Systems and Industrial Management - 15th IFIP TC8 International Conference, CISIM 2016, Vilnius, Lithuania, September 14-16, 2016, Proceedings. Springer, 2016. Lecture Notes in Computer Science. ISBN 978-3-319-45377-4. Available also from: `http://dblp.uni-trier.de/db/conf/cisim/cisim2016.html`.
3. NEHA YADAV Anupam Yadav, Manoj Kumar (auth.) An Introduction to Neural Network Methods for Differential Equations. In: 1st ed. Springer Netherlands, 2015, pp. 13–15. SpringerBriefs in Applied Sciences and Technology / SpringerBriefs in Computational Intelligence. ISBN 940179815X, ISBN 9789401798150.
4. WIDROW, Bernard; HOFF, Marcian E. Adaptive Switching Circuits. In: IRE, 1960, pp. 96–104.
5. MINSKY, M.; PAPERT, S. *Perceptrons*. Cambridge, MA: MIT Press, 1969.
6. JORDAN, Michael I. Attractor Dynamics and Parallelism in a Connectionist Sequential Machine. In: *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*. Hillsdale, NJ: Erlbaum, 1986, pp. 531–546.
7. PEARLMUTTER, Barak A. Learning State Space Trajectories in Recurrent Neural Networks. In: *Proceedings of the International Joint Conference on Neural Networks (Washington, DC)*. Piscataway, NJ: IEEE, 1989.

8. HOCHREITER, S. *Untersuchungen zu dynamischen neuronalen Netzen. Diploma thesis, Institut für Informatik, Lehrstuhl Prof. Brauer, Technische Universität München*. 1991.

9. BENGIO, Y.; SIMARD, P.; FRASCONI, P. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*. 1994, vol. 5, no. 2, pp. 157–166.

10. LECUN, Y.; BOTTOU, L.; BENGIO, Y.; HAFFNER, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*. 1998, vol. 86, no. 11, pp. 2278–2324. ISSN 0018-9219. Available from DOI: 10.1109/5.726791.

11. FUKUSHIMA, Kunihiko. Neocognitron: A Self-Organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position. *Biological Cybernetics*. 1980, vol. 36, pp. 193–202.

12. HODGKIN, A. L.; HUXLEY, A. F. A quantitative description of membrane current and its application to conduction and excitation in nerve. *Journal Physiology*. 1952, vol. 117, pp. 500–544.

13. VANDERAH, Todd W. Nolte's The human brain : an introduction to its functional anatomy. In: 7th edition / Todd W. Vanderah, Douglas J. Gould.. 2015, p. 20. ISBN 9781455728596.

14. KOUKOLÍK, František. Lidský mozek. In: Na Bělidle 34, 150 00 Praha 5: Galén, 2012, pp. 32–39. ISBN 978-80-7262-861-2.

15. GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. Deep learning. In: 2017, p. 168. ISBN 9780262035613 0262035618. Available also from: https://www.worldcat.org/title/deep-learning/oclc/985397543.

16. GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. Deep learning. In: 2017. ISBN 9780262035613 0262035618. Available also from: https://www.worldcat.org/title/deep-learning/oclc/985397543.

17. GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. Deep learning. In: 2017, pp. 67–68. ISBN 9780262035613 0262035618. Available also from: https://www.worldcat.org/title/deep-learning/oclc/985397543.

18. GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. Deep learning. In: 2017, p. 195. ISBN 9780262035613 0262035618. Available also from: https://www.worldcat.org/title/deep-learning/oclc/985397543.

19. GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. Deep learning. In: 2017, pp. 193–194. ISBN 9780262035613 0262035618. Available also from: https://www.worldcat.org/title/deep-learning/oclc/985397543.

20. MAAS, Andrew L.; HANNUN, Awni Y.; NG, Andrew Y. Rectifier nonlinearities improve neural network acoustic models. In: *in ICML Workshop on Deep Learning for Audio, Speech and Language Processing*. 2013.

21. GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. Deep learning. In: 2017, pp. 81, 184–187. ISBN 9780262035613 0262035618. Available also from: https://www.worldcat.org/title/deep-learning/oclc/985397543.

22. GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. Deep learning. In: 2017, p. 204. ISBN 9780262035613 0262035618. Available also from: https://www.worldcat.org/title/deep-learning/oclc/985397543.

23. GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. Deep learning. In: 2017, p. 326. ISBN 9780262035613 0262035618. Available also from: https://www.worldcat.org/title/deep-learning/oclc/985397543.

24. GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. Deep learning. In: 2017, pp. 205–207. ISBN 9780262035613 0262035618. Available also from: https://www.worldcat.org/title/deep-learning/oclc/985397543.

25. GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. Deep learning. In: 2017, p. 152. ISBN 9780262035613 0262035618. Available also from: https://www.worldcat.org/title/deep-learning/oclc/985397543.

26. GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. Deep learning. In: 2017, p. 85. ISBN 9780262035613 0262035618. Available also from: https://www.worldcat.org/title/deep-learning/oclc/985397543.

27. GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. Deep learning. In: 2017, pp. 151–153. ISBN 9780262035613 0262035618. Available also from: https://www.worldcat.org/title/deep-learning/oclc/985397543.

28. GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. Deep learning. In: 2017, p. 308. ISBN 9780262035613 0262035618. Available also from: https://www.worldcat.org/title/deep-learning/oclc/985397543.

29. ALBAWI, S.; MOHAMMED, T. A.; AL-ZAWI, S. Understanding of a convolutional neural network. In: *2017 International Conference on Engineering and Technology (ICET)*. 2017, pp. 1–6. Available from DOI: 10.1109/ICEngTechnol.2017.8308186.

30. LECUN, Yann; BENGIO, Yoshua. Convolutional Networks for Images, Speech and Time Series. In: *The Handbook of Brain Theory and Neural Networks*. Ed. by ARBIB, Michael A. The MIT Press, 1995, pp. 255–258.

31. GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. Deep learning. In: 2017, p. 203. ISBN 9780262035613 0262035618. Available also from: https://www.worldcat.org/title/deep-learning/oclc/985397543.

32. GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. Deep learning. In: 2017, p. 332. ISBN 9780262035613 0262035618. Available also from: https://www.worldcat.org/title/deep-learning/oclc/985397543.

33. GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. Deep learning. In: 2017, p. 342. ISBN 9780262035613 0262035618. Available also from: https://www.worldcat.org/title/deep-learning/oclc/985397543.

34. GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. Deep learning. In: 2017, p. 302. ISBN 9780262035613 0262035618. Available also from: https://www.worldcat.org/title/deep-learning/oclc/985397543.

35. GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. Deep learning. In: 2017, pp. 410–411. ISBN 9780262035613 0262035618. Available also from: https://www.worldcat.org/title/deep-learning/oclc/985397543.

36. GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. Deep learning. In: 2017, pp. 411–412. ISBN 9780262035613 0262035618. Available also from: https://www.worldcat.org/title/deep-learning/oclc/985397543.

37. SHI, Baoguang; BAI, Xiang; YAO, Cong. An End-to-End Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text Recognition. *CoRR*. 2015, vol. abs/1507.05717. Available also from: http://dblp.uni-trier.de/db/journals/corr/corr1507.html#ShiBY15.

38. HE, Kaiming; ZHANG, Xiangyu; REN, Shaoqing; SUN, Jian. *Deep Residual Learning for Image Recognition*. 2015. Available also from: http://arxiv.org/abs/1512.03385. cite arxiv:1512.03385Comment: Tech report.

39. BAHDANAU, Dzmitry; CHO, Kyunghyun; BENGIO, Yoshua. *Neural Machine Translation by Jointly Learning to Align and Translate*. 2014. Available also from: http://arxiv.org/abs/1409.0473.

40. VASWANI, Ashish; SHAZEER, Noam; PARMAR, Niki; USZKOREIT, Jakob; JONES, Llion; GOMEZ, Aidan N.; KAISER, Łukasz; POLOSUKHIN, Illia. Attention is All you Need. In: GUYON, I.; LUXBURG, U. V.; BENGIO, S.; WALLACH, H.; FERGUS, R.; VISHWANATHAN, S.; GARNETT, R. (eds.). *Advances in Neural Information Processing Systems 30*. Curran Associates, Inc., 2017, pp. 5998–6008. Available also from: `https://papers.nips.cc/paper/7181-attention-is-all-you-need`.

41. GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. Deep learning. In: 2017, pp. 357, 502–503. ISBN 9780262035613 0262035618. Available also from: `https://www.worldcat.org/title/deep-learning/oclc/985397543`.

42. ROCHE, Fanny; HUEBER, Thomas; LIMIER, Samuel; GIRIN, Laurent. *Autoencoders for music sound modeling: a comparison of linear, shallow, deep, recurrent and variational models*. 2019. Available from arXiv: `1806.04096 [eess.AS]`.

43. BERTHELOT, David; RAFFEL, Colin; ROY, Aurko; GOODFELLOW, Ian. *Understanding and Improving Interpolation in Autoencoders via an Adversarial Regularizer*. 2018. Available from eprint: `arXiv:1807.07543`.

44. MRABAH, Nairouz; KHAN, Naimul Mefraz; KSANTINI, Riadh; LACHIRI, Zied. *Deep Clustering with a Dynamic Autoencoder: From Reconstruction towards Centroids Construction*. 2019. Available from eprint: `arXiv:1901.07752`.

45. KINGMA, Diederik P; WELLING, Max. *Auto-Encoding Variational Bayes*. 2013. Available also from: `http://arxiv.org/abs/1312.6114`. cite arxiv:1312.6114.

46. PRASAD, Vignesh; DAS, Dipanjan; BHOWMICK, Brojeshwar. Variational Clustering: Leveraging Variational Autoencoders for Image Clustering. In: *2020 International Joint Conference on Neural Networks, IJCNN 2020, Glasgow, United Kingdom, July 19-24, 2020*. IEEE, 2020, pp. 1–10. Available from DOI: `10.1109/IJCNN48605.2020.9207523`.

47. BASS, H.E.; SUTHERLAND, L.C.; ZUCKERWAR, A.J.; BLACKSTOCK, D.T.; HESTER, D.M. *Atmospheric absorption of sound: Further developments*. Calhoun, 1995. Available also from: `https://calhoun.nps.edu/handle/10945/62134`.

48. DENG, J.; DONG, W.; SOCHER, R.; LI, L.-J.; LI, K.; FEI-FEI, L. ImageNet: A Large-Scale Hierarchical Image Database. In: *CVPR09*. 2009.

49. C. WEIHS D. Jannach, I. Vatolkin; RUDOLPH, G. *Music Data Analysis: Foundations and Applications*. Chapman  Hall/CRC, 2016. ISBN 978-1-4987-1956-8.

50. STEVENS, Stanley S.; VOLKMANN, John; NEWMAN, Edwin B. A Scale for the Measurement of the Psychological Magnitude Pitch. *The Journal of the Acoustical Society of America*. 1937, vol. 8, pp. 185–190. Available from DOI: 10.1121/1.1915893.

51. XU, Yichong; XIAO, Tianjun; ZHANG, Jiaxing; YANG, Kuiyuan; ZHANG, Zheng. Scale-Invariant Convolutional Neural Networks. *CoRR*. 2014, vol. abs/1411.6369. Available also from: http://dblp.uni-trier.de/db/journals/corr/corr1411.html.

52. KAYHAN, Osman Semih; GEMERT, Jan C. van. *On Translation Invariance in CNNs: Convolutional Layers can Exploit Absolute Spatial Location*. 2020. Available from arXiv: 2003.07064 [cs.CV].

53. SCHÖRKHUBER, Christian; KLAPURI, Anssi. Constant-Q transform toolbox for music processing. *Proc. 7th Sound and Music Computing Conf.* 2010.

54. CWITKOWITZ, Frank C. Jr. End-to-End Music Transcription Using Fine-Tuned Variable-Q Filterbanks. 2019. Available also from: https://scholarworks.rit.edu/theses/10143.

55. BROWN, Judith C. Calculation of a Constant Q Spectral Transform. *Journal of the Acoustical Society of America*. 1991, vol. 89, no. 1, pp. 425–434.

56. VENKITARAMAN, Arun; ADIGA, Aniruddha; SEELAMANTULA, Chandra Sekhar. Auditory-motivated Gammatone wavelet transform. *Signal Processing*. 2014, vol. 94, pp. 608–619. Available from DOI: 10.1016/j.sigpro.2013.07.029.

57. SAINATH, T. N.; KINGSBURY, B.; MOHAMED, A.; RAMABHADRAN, B. Learning filter banks within a deep neural network framework. In: *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*. 2013, pp. 297–302. Available from DOI: 10.1109/ASRU.2013.6707746.

58. OORD, Aaron van den; DIELEMAN, Sander; ZEN, Heiga; SIMONYAN, Karen; VINYALS, Oriol; GRAVES, Alex; KALCHBRENNER, Nal; SENIOR, Andrew; KAVUKCUOGLU, Koray. *WaveNet: A Generative Model for Raw Audio*. 2016. Available also from: http://arxiv.org/abs/1609.03499. arxiv:1609.03499.

59. LEE, Jongpil; KIM, Taejun; PARK, Jiyoung; NAM, Juhan. Raw Waveform-based Audio Classification Using Sample-level CNN Architectures. *CoRR*. 2017, vol. abs/1712.00866. Available also from: http://dblp.uni-trier.de/db/journals/corr/corr1712.html#abs-1712-00866.

60. CHEN, Jiaxu; HAO, Jing; CHEN, Kai; XIE, Di; YANG, Shicai; PU, Shiliang. An End-to-End Audio Classification System Based on Raw Waveforms and Mix-Training Strategy. In: KUBIN, Gernot; KACIC, Zdravko (eds.). *INTERSPEECH*. ISCA, 2019, pp. 3644–3648. Available also from: http://dblp.uni-trier.de/db/conf/interspeech/interspeech2019.html#ChenHCXYP19.

61. SALAMON, Justin; BELLO, Juan Pablo. Deep Convolutional Neural Networks and Data Augmentation for Environmental Sound Classification. *CoRR*. 2016, vol. abs/1608.04363. Available also from: http://dblp.uni-trier.de/db/journals/corr/corr1608.html#SalamonB16.

62. ENGEL, Jesse; RESNICK, Cinjon; ROBERTS, Adam; DIELEMAN, Sander; ECK, Douglas; SIMONYAN, Karen; NOROUZI, Mohammad. *Neural Audio Synthesis of Musical Notes with WaveNet Autoencoders*. 2017. Available from eprint: arXiv:1704.01279.

63. BOSCH, J.; JANER, J.; FUHRMANN, Ferdinand; HERRERA, Perfecto. A Comparison of Sound Segregation Techniques for Predominant Instrument Recognition in Musical Audio Signals. In: *13th International Society for Music Information Retrieval Conference (ISMIR 2012)*. Porto, Portugal, 2012, pp. 559–564. Available also from: http://mtg.upf.edu/system/files/publications/Bosch-ISMIR2012.pdf.

64. HUMPHREY, Eric J.; DURAND, Simon; MCFEE, Brian. OpenMIC-2018. 2018. Available from DOI: 10.5281/zenodo.1432913.

65. GEMMEKE, Jort F.; ELLIS, Daniel P. W.; FREEDMAN, Dylan; JANSEN, Aren; LAWRENCE, Wade; MOORE, R. Channing; PLAKAL, Manoj; RITTER, Marvin. Audio Set: An ontology and human-labeled dataset for audio events. In: *Proc. IEEE ICASSP 2017*. New Orleans, LA, 2017.

66. CHEN, Ting; KORNBLITH, Simon; NOROUZI, Mohammad; HINTON, Geoffrey. *A Simple Framework for Contrastive Learning of Visual Representations*. 2020. Available from arXiv: 2002.05709 [cs.LG].

67. ZHAN, Xiaohang; XIE, Jiahao; LIU, Ziwei; ONG, Yew Soon; LOY, Chen Change. *Online Deep Clustering for Unsupervised Representation Learning*. 2020. Available from arXiv: 2006.10645 [cs.CV].

68. [N.d.]. Available also from: http://ffmpeg.org/.

69. *youtube*. [N.d.]. Available also from: http://ytdl-org.github.io/youtube-dl.

70. *YouTube Data API | Google Developers*. Google, [n.d.]. Available also from: https://developers.google.com/youtube/v3.

71. *librosa*. [N.d.]. Available also from: https://librosa.org/doc/latest/index.html.

72. [N.d.]. Available also from: https://pytorch.org/.

Appendix A

# Latent Space Visualisations

This appendix shows visualised latent spaces of autoencoders used in finall tests. These visualisations are not good evaluation of performance of the latent spaces. However, they can provide hint when the latent spaces do not perform optimally as was case of the first model. Only most interesting images are chosen because visualisation produced by same techniques give similar results it is intended to be just illustrative.

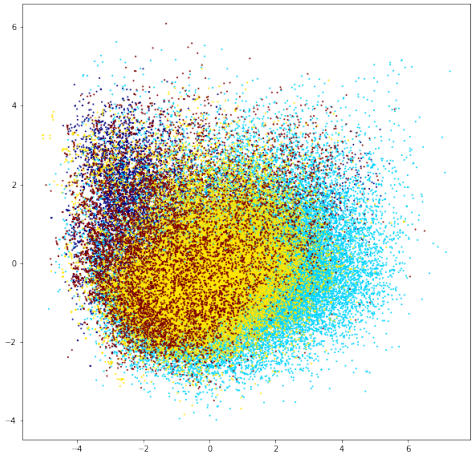



Figure A.1: PCA visualisation of latent space corresponding to Randomly Initialized Autoencoder. Subset of NSynth classes is used. Color differentiate datapoints belonging to different labels namely flute, guitar, vocal and organ.

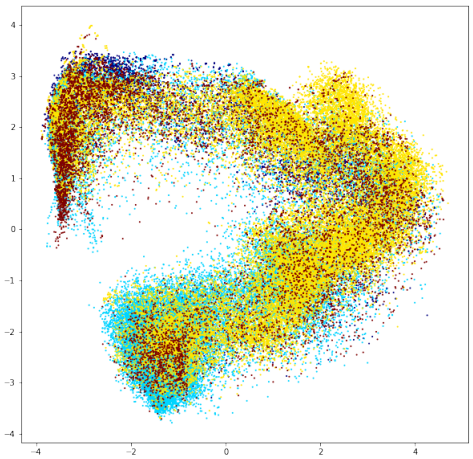



Figure A.2: Visualisation of same data as in previous picture but using ACAI instead.

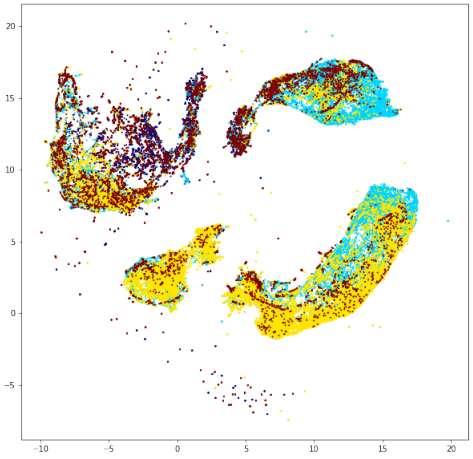



Figure A.3: UMAP visualisation of NSynth data mapped in ACAIs latent space.

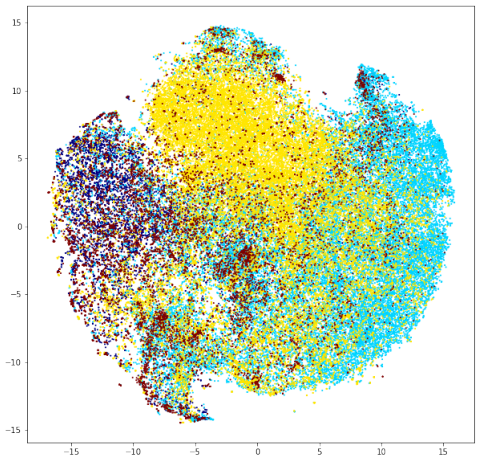



Figure A.4: t-SNE visualisation of Randomly Initialized Autoencoder on NSynth data

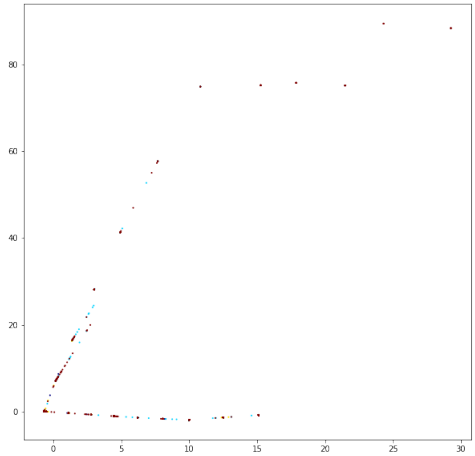



Figure A.5: PCA visualisation of ACAI when Sigmoid is used for latent vector. This picture could indicate that latent space might not be well suitable for clustering and other methods.

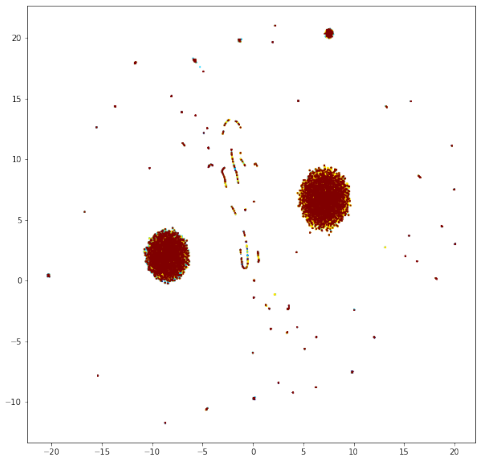



Figure A.6: UMAP visualisation of ACAI when Sigmoid is used for latent vector.

Appendix B

# Acronyms

**LSTM** Long Short-term Memory

**CNN** Convolutional Neural Network

**FFA** Feedforward Neural Network

**MLP** Multilayer Perceptron

**ReLU** Rectified Linear Unit

**SGD** Stochastic Gradient Descent

**RNN** Recurrent Neural Network

**GRU** Gated Recurrent Unit

**CRNN** Convolutional Recurrent Neural Network

**PCA** Principal Component Analysis

**VAE** Variational Autoencoder

**ACAI** Adversarially Constrained Autoencoder Interpolation

**DFT** Discrete Fourier Transform

**FFT** Fast Fourier Transform

**STFT** Short-time Fourier transform

**STFFT** Short-time Fast Fourier transform

**MFCC** Mel-frequency cepstral Coefficients

**DynAE** Dynamic Autoencoder

**RI** Rand Index

**NMI** Normalized Mutual Information

Appendix C

# Contents of enclosed DVD

- `readme.txt` ..... the file with DVD contents description
- `src` ..... directory of source codes
  - `preproc` ..... source codes of pre-processing programs
  - `models` ..... source codes of tested models
  - `thesis` ..... the directory of LaTeX source codes of the thesis
- `text` ..... the thesis text directory
  - `thesis.pdf` ..... the thesis text in PDF format
  - `thesis.ps` ..... the thesis text in PS format