## Abstract

Following the success of deep neural networks in image recognition competitions in recent years, this thesis tries to apply such models in the field of music information retrieval, more specifically musical instrument classification. While identifying musical instruments in audio signals may seem intuitive for a human being, things get rather complicated in the digital domain. Deep learning provides researchers with tools in order to overcome such limitations. A convolutional neural network (CNN) is used to categorize an input signal into one of seven classes. The data set consists of short audio recordings of musical instruments typically found in symphony orchestras. To provide the CNN with the appropriate data, the signals are transferred into the digital domain using log mel-spectrograms.

The proposed network uses the convolutional base of a state-of-the-art computer vision model called Inception-ResNet-v2, which is based on the original Inception architecture developed by Szegedy et al. at Google. Inception-ResNet-v2 introduces residual connections to the initial network, further improving training speed. While features are extracted using the pre-trained convolutional base, a separately added classification network consisting of fully connected layers is trained on the new data set. Several configurations of this classifier get tested and compared. The loss score is being evaluated using a softmax activation function with cross entropy loss. While stochastic gradient descent is utilized for weight updating, some configurations implement nesterov's accelerated gradient in order to boost the optimization process.

Throughout this text, the reader is being introduced to basic methods and techniques used in deep learning and music information retrieval. State-of-the-art concepts in both computer vision and music information retrieval are being introduced. It is shown how machines are able to derive patterns from data, further elaborating on the details of the actual learning process. Additional literature is suggested for gathering further information about specific topics.

# Contents

# 1 Introduction

The continuous development of deep learning models in recent years has made an impact on a vast variety of research areas, including the field of music information retrieval (MIR). While sophisticated architectures like the convolutional neural network (CNN) were initially designed to perform computer vision tasks [28, 54, 19], they have paved the way for significant advancements in audio-related fields such as speech recognition [2, 49, 58], speech enhancement [42] and audio event classification [52]. Furthermore, Hershey et al. [20] have underpinned the applicability of image recognition models in audio classification problems.

These achievements can be attributed to the the introduction of better optimization techniques [69], larger datasets [28] and taking usage of parallel computing on graphics processing units (GPUs) [66]. The advantage of using GPUs as processing units stems from the introduction of the CUDA [1] programming interface introduced by NVIDIA in 2007, which makes the parallelization of matrix operations in deep neural networks computationally affordable. With the rising success of deep learning, specialized hardware such as Google's tensor processing unit (TPU) [2] has been developed with machine learning tasks in mind. Research [25] has confirmed the superior performance of TPUs compared to competitive technologies in the field. This improvement of GPUs in single user machines and the availability of TPU clusters through cloud services gives smaller research teams and interested individuals the ability to follow their scientific pursuits. Thus, making machine learning more accessible to a broader audience, while further improving the speed at which the field advances.

This bachelor thesis aims to introduce the reader to the field of deep learning in MIR, more specifically the task of automatic audio classification. Several related publications will be discussed, in order to find ideal strategies for the task at hand. In Section 3.1, the main concepts of machine learning will be demonstrated in a digestible manner, while shedding light on the underlying learning process. The overall theme of this text tiptoes around the question, if state-of-the-art computer vision models are capable of classifying musical instruments based on their respective timbre. It will be shown, that deep neural networks that have been pre-trained on large data sets can be adapted to a vastly different visual experience. Throughout Section 5 the actual learning algorithm [59] will be presented in detail. The experimental setups of related publications will be examined to find the best configuration for the proposed classification task. Section 6 evaluates the suitability of such computer vision algorithms for musical instrument classification. The data set consists of single-note recordings of various musical instruments, which are transformed into an appropriate visual representation in order to apply image recognition algorithms.

As CNNs are extensively used in speech recognition for modeling spatial and temporal correlations of visual representations [49, 2, 58], one may deduce that

[1]developer.nvidia.com/about-cuda
[2]cloud.google.com/tpu

such networks are capable of solving similar problems such as audio classification. Hence, a convolutional neural network (Section 3.2.2) will be used for the classification task. It is crucial to think about the appropriate visual representation for such learning algorithms. This will be conducted in Section 4.1, while further elaborating on the preprocessing stage.

## 2 Related Work

Traditional machine learning models like support vector machines (SVMs) have proven to be valuable approaches for MIR tasks [24]. However, as data sets grow in size, deep learning presents itself as a promising set of methods for tackling highly complex tasks such as audio classification. [38] provides a vast collection of music analysis methods, while also adapting borrowed concepts from speech recognition to the unique properties of music signals. Publications such as [48] propose more modern approaches for specialized tasks such as timbre analysis.

A huge chunk of recent works has been dedicated to solving audio-related problems with deep neural networks, especially CNNs [48, 46, 20, 2, 62]. There are several publications regarding chord recognition using DNNs, including [27] who solely rely on examining chroma vectors and [22] in which a CNN is used to classify different minor and major chords. CNNs are also used in [23] for environmental sound classification, while [62] uses the network for acoustic event detection.

There is only a sparse body of works [44, 67, 63, 35] regarding musical instrument recognition and classification using deep neural networks, making it a necessity to incorporate solutions employed by researchers in related fields such as speech recognition. [44] investigates the use of multiresolution recurrence plots (MRPs) in the context of CNNs. [67] compare performances of CNNs and recurrent neural networks (RNNs) on log mel-spectrograms. In [35], the authors utilize a CNN to identify musical instruments in polyphonic music using mel-frequency cepstral coefficients (MFCCs). [63] solely relies on the analysis of the frequency domain using Fast Fourier Transform (FFT). The resulting frequency spectrum is then partitioned into 50 different bands, each being computed using the average of their respective frequency range. It should be mentioned that the used dataset here is very unbalanced. While there are only 23 samples labeled as banjo, the violin class comprises a total of 366 samples. Other research [41] has shown that such imbalances can effect the performance of a neural network, especially when trying to optimize accuracy.

Only a single audio-related publication [52] was found that uses Inception-v4 [59] for it's classification step. This highlights the need for further experiments with Inception networks [60, 61, 59], especially the ones including residual connections [19].

# 3 Prerequisite Knowledge

This chapter should assist the reader in gaining a basic understanding of machine learning and its underlying algorithms. If one is already familiar with the fundamentals of this field, he or she may skip to Section 4. In order to fully understand the mathematical technicalities behind the technologies discussed throughout this text, it is advised to use [45] as supplementary material. [17, 15] should be consulted if one wants to delve deeper into the topics covered in this text.

## 3.1 Machine Learning

Essentially, machine learning can be seen as a process, in which a computer is able to learn from data. The acquired knowledge can then be applied on specific problems. A properly adjusted model should thereby yield a better performance based on the experience it gained throughout the learning process. Two main approaches in machine learning are supervised and unsupervised learning. Problems encountered by learning algorithms are typically assigned to either one of those categories, depending on what kind of experience the models gather from the data at hand.[17] In order to streamline the information covered in this chapter with the model presented in this thesis, only the supervised learning approach will be covered.

When dealing with a classification problem, the goal of the learning algorithm is to assign the input to a specific category. In fields like image classification, such categories may be nominal labels such as tree or car. Other instances may reveal themselves as regression problems, in which the model is told to return a numerical value. A machine learning model tries to find a relation between each example $x_i$ and its corresponding response $y_i$. Formally, such a relation can be written as

$$y = f(x) + \epsilon$$

where $f$ is an unknown function of the input variable $x$ and $\epsilon$ is a random error term. The learning algorithm tries to estimate $f$ using linear and non-linear approaches - the latter being deployed in this text.[17, 15]

Before utilizing such models in real world applications, they have first to be trained on a set of pre-collected data called the training set. Such data contains labels, which represent the desired output for each example. A training set therefore provides the model with the desired results. In order to measure the performance of a learning algorithm, one relies on metrics such as accuracy or error rate. This is done through measuring the difference between the current result $\hat{y}$ and the desired output $y$. In other words, the learning process can be described as the adjustment of the error metric. In deep learning literature, this error metric is often called loss value. As we are interested in the performance of the model on unseen data, performance is typically measured on a separate test set which is preemptively split from the initial training data.[17, 15] It should be mentioned that there exists a vast variety of loss functions [40], each affecting the learning dynamics of a network in a different way - log loss being widely adopted in classification tasks. If the test error is substantially higher than the training error, the model is potentially overfitting the data set. This is often the case when the training data is sparse, making the learning algorithm prone to finding patterns in sampling noise. A possible solution to this problem is to add a mechanism called dropout [55].

Once the model is sufficiently trained, it can be applied to real world problems. A key concept of this process is to transform the input data into a meaningful representation. In other words, a certain style of representation can make the task of finding an appropriate solution much easier. This idea can be further studied in Figure 1.

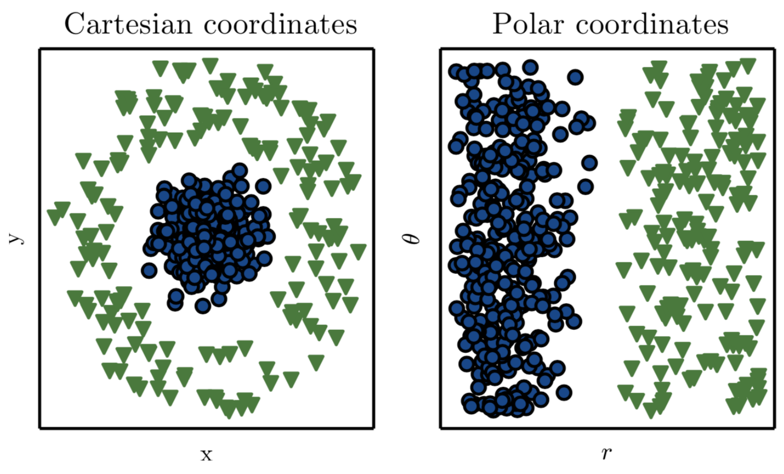


Figure 1: Two plots representing the same data set. When using a linear decision boundary, the plot on the right makes the task far easier to solve. Therefore, polar coordinates as style of representation is superior for this specific task.

## 3.2 Deep Learning

Deep learning can be seen as a subfield of machine learning that is able to produce more complicated mappings from input to target in a hierarchical fashion, expressing high-level representations through less complex representations. The primary structure found in deep learning models is the neural network [50], which can be thought of as an interconnected web of processing units. These units typically process simple linear or nonlinear computations and are organized in layers.

A single unit - also commonly referred to as perceptron [50] - first computes an affine transformation $z = w \cdot x + b$ on an arbitrary number of outputs from the lower layer, collectively a vector of $N$ inputs $x$. This is done by computing the dot product of the input vector with a vector of weights $w$, followed by adding a bias term $b$. The resulting scalar value is passed through a nonlinear activation function $g(z)$ such as a rectified linear unit (ReLU) [39]. ReLUs will be further discussed in Section 5.2 in the course of introducing the model's architecture. Figure 2 visualizes the structure of a single processing unit.[50, 17]

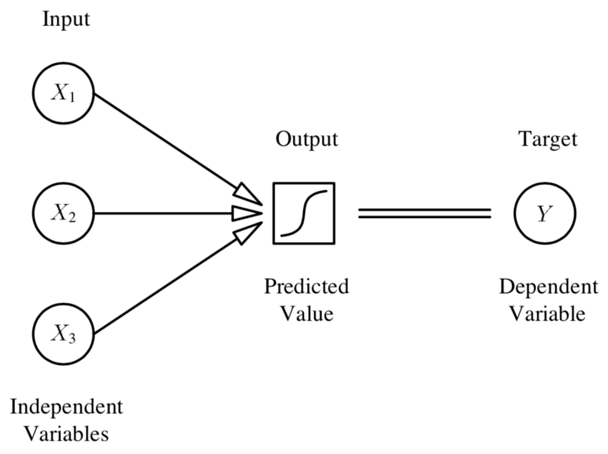


Figure 2: Single perceptron with three input units and a nonlinear activation function.

While such networks can be found in other machine learning approaches, the term deep learning is associated with the type of neural network that consists of multiple successive layers resulting in a particularly deep architecture. Modern deep learning models can contain tens or hundreds of such layers. Such additional layers - added between input and output layer - are called hidden layers, as their values are not visible for the observer. As mentioned before, deep neural networks are able to interpret data through hierarchical representation learning. Each hidden layer conducts a simple data transformation, whose function is approximated through an optimization process during learning. Together with the nonlinear property of the activation functions, the succession of operations from layer to layer allows the network to learn high-level patterns in the data set [6]. Figure 3 illustrates this feature hierarchy in a simplified manner.[9, 17]

### 3.2.1 Stochastic Gradient Descent (SGD)

As researchers are now able to work with deep networks and huge data sets, they are being confronted with the computational complexity of their models. With the growing number of parameters, one needs to reach for an appropriate optimization technique in order to effectively train a neural network. As mentioned before, each layer contains a set of weights and biases. Such parameters

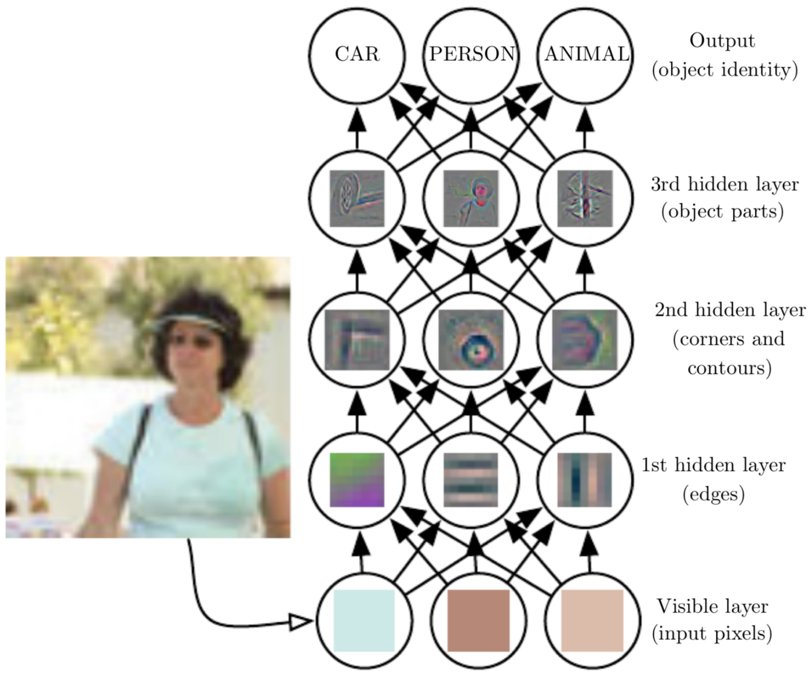


Figure 3: The input image is displayed as a matrix of $m$ x $n$ pixel values. Each unit of the input layer (visible layer) represents one pixel of the image. Following the feature hierarchy of the network, hidden layers are able to find patterns like edges and corners in their preceding layer. High-level layers manage to detect more complex structures such as object parts. A classifier at the end of the network then makes the final prediction by computing the corresponding label.

are trainable and essentially store the information learned about the input data. In many cases these weights are initialized randomly, setting a starting point for the learning process. Through a feedback signal the network then gradually adjusts the learnable parameters based on its learning experience during training. The final goal of this adjustment is the reduction of the current loss value. In order to do this process effectively, the gradient of the loss with regard to the model's parameters is evaluated. This is possible, because the functions found throughout the network's layers are differentiable. Incremental steps in the opposite direction of the gradient can be used to decrease the loss.[9]

In other words, one tries to find the best settings for the weights such that the global minimum of the function can be found. As illustrated in Figure 4, functions may comprise multiple local minima. This makes it harder for gradient descent optimization to find the lowest point, since reaching a local minimum often halts the minimization. Therefore in some cases, settling at a local minimum presents itself as the best possible outcome.[17]

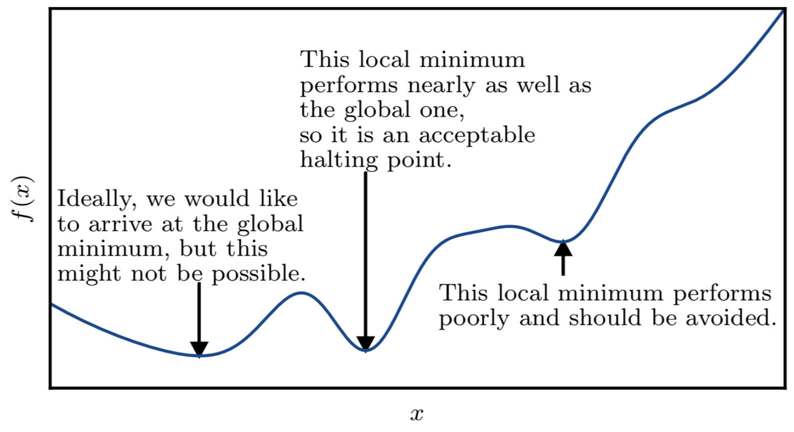


Figure 4: Global minimum and multiple local minima of a function.

Bottou [4] proposes stochastic gradient descent (SGD) as a promising extension of gradient descent optimization. For a single step, the initial gradient descent (GD) algorithm needs to compute the cost gradient for the whole data set. This can be denoted as

$$\nabla_\theta J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\nabla_\theta L(x^i, y^i, \theta)$$

where $\theta$ represents the optimal parameters for $m$ examples and $J$ is the cost function over the sum of $m$ loss scores $L$. Since deep learning often involves working with huge data sets, GD's computational cost $O(m)$ would reach an inefficient state. With SGD, one estimates the gradient for a subset of examples at a time. This subset is commonly referred to as mini-batch. The gradient of a mini-batch can be written as

$$g = \frac{1}{m'}\nabla_\theta \sum_{i=1}^{m'}\nabla_\theta L(x^i, y^i, \theta)$$

where $m'$ is the mini-batch size. Each of these estimates can be regarded as a gradual update to the optimization process. The computational cost of SGD isn't proportional to the training set size. Usually, the best weight values are found before the algorithm has processed every training example.[17] The changes to the weights are being deployed through a backward pass in the network using an algorithm called backpropagation [34], which involves applying the chain rule to its respective functions. The mechanism guiding the backpropagation algorithm is as follows:

$$x_{k+1} = x_k - \eta_k g_k$$

Here, $x_k$ is the current weight matrix, $\eta_k$ is the learning rate and $g_k$ represents the gradient of the respective function. SGD is being utilized in the Inception-ResNet-v2 model [59], which gets thoroughly discussed in Section 5.2.

### 3.2.2 Convolutional Neural Networks (CNNs)

Initially proposed by [30], convolutional neural networks have become an integral part of state-of-the-art image recognition architectures [28, 54, 60, 61, 59]. Research [21, 32] has shown that feedforward neural networks are able to predict any kind of measurable function. This gives CNNs the capability of being applied to a broad spectrum of tasks, including the field of MIR. This kind of deep neural network has become incredibly popular for classification tasks [49, 58] which strongly depend on local relationships. This is due to its ability to preserve spatial correlations and learn translational invariance without decreasing the network's performance. Learning translational invariance allows the network to recognize a learned pattern in multiple regions of the input, improving the model's generalization power [9].

CNNs employ special kinds of layers, namely convolutional and pooling layers. As the name suggests, convolutional layers perform an operation called convolution:

$$s(t) = (x * w)(t)$$

where $x$ is the input and $w$ is the kernel at a particular moment $t$. When working with image data, however, one is confronted with a multidimensional array of data points. Hence, convolution needs to be applied on several axis. For a two-dimensional input, the functions can be written as

$$S(i, j) = (I * K)(i, j) = \sum_m \sum_n I(m, n)K(i - m, j - n)$$

where $I$ is the two-dimensional input image, $K$ is a kernel with matching dimensions, while $i$ and $j$ are positional arguments regarding the array. The convolution operation produces a so-called feature map. Every unit of this feature map corresponds to its specific patch in the previous layer. When working with colored images, the input consists of 3 axis. Namely height, width and depth, the latter marking the number of RGB channels. A convolutional layer can apply multiple kernels - also referred to as filters - on the input, resulting in a higher number of output channels in the feature map. Such feature channels are unrelated to the RGB channels of the initial input image. Current state-of-the-art image recognition models [28, 19, 59] typically employ squared windows of size 3x3. These windows are sliding over the input map, stopping at every possible location. Figure 5 describes each step of the convolution in detail. In some cases, it may be appropriate to skip certain locations.[17, 9]

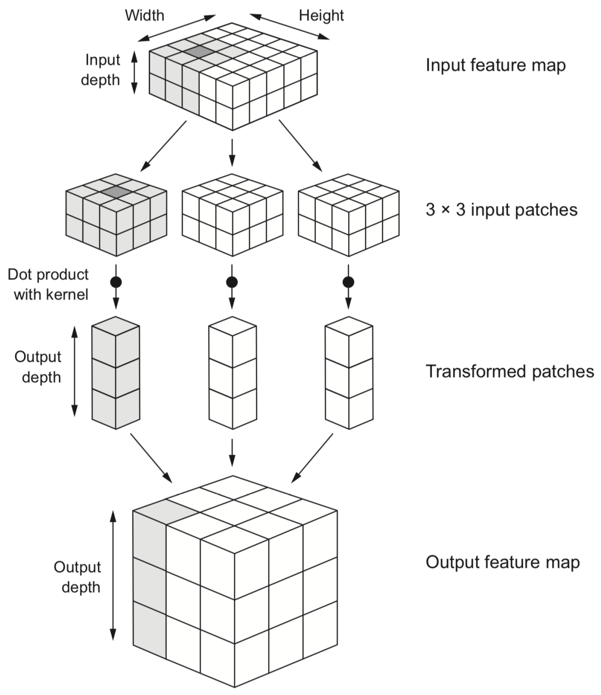


Figure 5: A predefined window gets applied to individual locations of the input feature map. The resulting patches build a dot product with the kernel, resulting in one-dimensional vectors on the depth-axis. All vectors generated through this process merge into the output feature map. In this case 3 filters were applied, hence, the output feature map has the dimensions 3x3x3.

While traditional deep neural networks often rely on fully connected layers where each unit is connected with all units of the previous layer, CNNs utilize sparse interactions between layers. By doing this, the network can detect useful patters in specific local regions of the input image. This significantly reduces memory usage, as fewer parameters have to be stored. A simplified visualization of this concept can be found in Figure 6.[17]

Another feature of convolutional layers is called parameter sharing. Here, each kernel owns a distinct set of weight which gets applied to every position in the input layer. In other words, a single kernel $W_{ij}$ is mapping all receptive fields of the $i^{th}$ input channel to the $j^{th}$ feature map. This further reduces the number of parameters that have to be stored. Therefore, only a single weight matrix per kernel has to be learned, instead of $m$ x $n$ parameters for $m$ inputs and $n$ outputs in fully connected layers.[17]

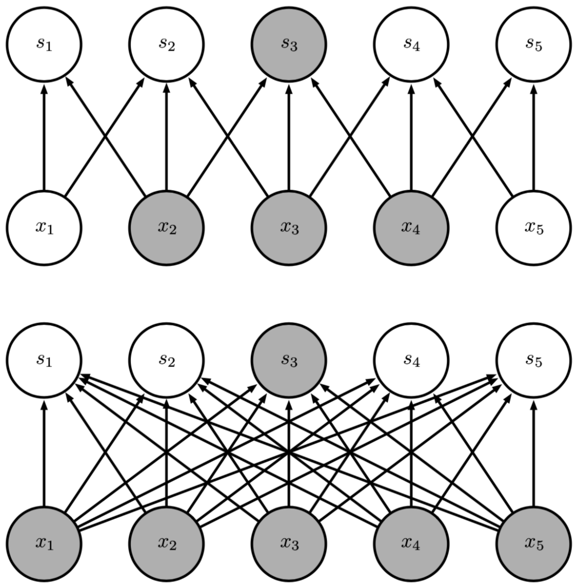


Figure 6: In the structure on the top, unit $s_3$ is only sparsely connected to the units in the previous layer. Therefore, only units $x_{2-4}$ are mapped to it. On the bottom, one can see the same structure as a fully connected version. Here, $s_3$ is affected by every unit of the input layer.

Pooling layers are usually applied after convolutional layers in order to introduce spatial invariance [51] to the feature map. This is done by downsampling the resolution of such maps, allowing the network to learn small shifts between learned features and unseen data. The Inception-ResNet-v2 [59] in Section 5.2 uses average pooling after the last Inception block and 2 instances of max pooling in the stem. Max pooling essentially applies windows to the input feature map, computing the maximum value of each individual patch. Average pooling on the other hand evaluates the average value of each window patch.[9]

## 4 Data Set

The data set used in this thesis consists of short, monophonic recordings of various musical instruments. The audio samples were provided by Vienna Symphonic Library [3], a sample library developer specialized in large-scale recordings of orchestra instruments. The instruments chosen are part of a typical symphony orchestra, including violin, cello, double-bass, flute, oboe, trumpet

[3]vsl.co.at/en

| | Violin | Cello | Double-Bass | Flute | Oboe | Trumpet | Tuba |
|---|---|---|---|---|---|---|---|
| Samples | 200 | 200 | 185 | 156 | 170 | 210 | 180 |
| Pitch | G3 - G#7 | C2 - C#6 | E1 - E4 | C4 - D7 | A#3 - G6 | F#3 - G#5 | F1 - F4 |
| Dynamic | pp - f | pp - f | pp - ff | p - mf | p - f | pp - f | p - ff |

Table 1: Further specifications regarding the data set.

and tuba. Special attention has been focused on creating a balanced data set, ensuring a fair sample distribution on the different instrument classes. This is done in order to prevent performance issues caused by unbalanced data [41]. The samples cover multiple octaves and dynamic levels of each instrument, introducing variation to the learning algorithm. All recordings are provided in 44.100 Hz and a bit depth of 16 bits. The recordings are very short and only consist of a single note each. The woodwind samples solely consist of staccato sounds. Furthermore, the samples were recorded in a dry space, meaning that there is no additional noise caused by the acoustic properties of the recording space. Table 1 exhibits the specifications of each class in detail.

## 4.1 Preprocessing

The following paragraphs elaborate on useful techniques in order to optimize data for the subsequent training stage. The condensed version of the preprocessing pipeline is presented in Section 6. To provide the CNN with an appropriate input, the samples have to be transformed into the visual domain. Before doing so, it may be beneficial to tweak the following audio-related features.

First and foremost, several researchers tend to reduce the sample rate in order to reduce running time. Music-related publications suggest to downsample the audio material to 22 kHz [18], 32 kHz [24], 7 kHz [22], 8-16 kHz [6] or 12 kHz [48, 5, 8]. Following the Nyquist sample theorem [29], samples with a bandwidth $W$ of e.g. 12 kHz would allow the network to recognize features up to 6 kHz. One needs $2W$ samples per second in order to recover a stable signal with $W$ Hz of bandwidth.

Since the goal of this thesis is to classify musical instruments, recognizing timbre patterns in the time-frequency domain is crucial. Pons et al. [48] further elaborates on this issue. It is advised to utilize magnitude spectrograms as input images, as they employ time, frequency and amplitude. Here, the researchers created a STFT using 512 points FFT. Huzaifah et al. [23] conducted a study, evaluating the best performing time-frequency representation for CNN classification of environmental audio recordings. Two data sets with multiple CNN configurations were tested. It was shown, that Mel-STFT spectrograms performed consistently better than baseline MFCC features. Linear-STFT and Constant-Q Transform (CQT) did well on some CNN configurations, especially their respective wide-band versions.

Many publications in the field of speaker and speech recognition [49] rely on MFCC features. Regarding musical instrument classification, Joder et al.

[24] use MFCCs with a SVM classifier. Moreover, derivates of the MFCCs are computed in order to produce time-conscious discriminating features. Sturm et al. [56] further elaborate on this topic, i.e. using multiple time-scales to compute MFCCs. The researchers show significantly better performance of such features over MFCCs computed from a single time-scale.

Papers specifically regarding musical instrument recognition rely on log mel-spectrograms [18, 67], spectrograms with multiresolution recurrence plots (MRPs) [44] and MFCCs or end-to-end [35]. [8] uses log mel-spectrograms with 96 mel-bins and 256 hop-size samples for music tagging, basing their decision on research [6, 13] showing their superior performance compared to STFT, MFCCs and linear-amplitude mel-spectrograms. [64] has affirmed these results in the DCASE 2013 challenge for acoustic scene classification, while also ranking 7th place in the DCASE 2016 challenge. Han et al. [18] rely on such representations in order to identify instruments in polyphonic music, using 128 mel-bins, 1024 sample windows and a hop-size of 256. [37] states that mel-scaled spectrograms are an ideal representation for timbre analysis.

Based on this collection of available information, one may conclude that there exists no established consensus regarding audio recognition nor musical instrument classification at this moment. Hence, samples will be downsampled to 12 kHz, making a reasonable trade-off between computational expenses and timbre representation. Log mel-spectrograms are chosen as input for the CNN based on the results proposed by [6, 13]. Further specifications are mentioned in Section 6.

## 5 Model Selection

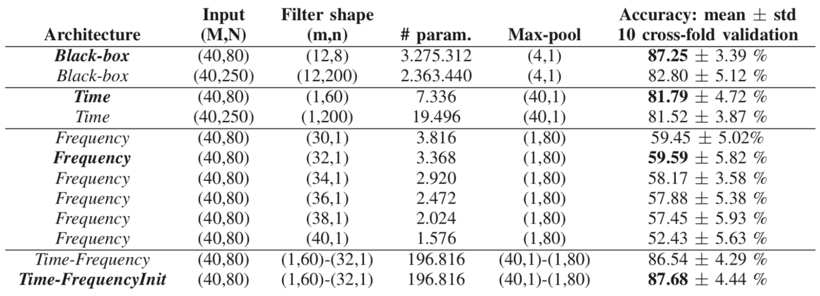
This section presents the selection process and architecture of the proposed machine learning model, which will be used to perform the task of musical instrument classification. There are several deep neural networks at our disposal, ranging from recurrent neural networks (RNNs) for sequential data [3] to convolutional neural networks for computer vision tasks [28]. The related work covered in Section 2 suggests using the latter network for audio classification tasks. The following subsections elaborate on further design choices posed by the modular design of deep neural networks.

### 5.1 Architecture

As shown in Section 2, CNNs not only excel in image recognition tasks, but also deliver useful results in MIR. Since the dataset used in this thesis gets transformed into a visual representation [23] throughout the preprocessing stage, it seems consequential to use CNNs for the subsequent feature extraction and classification step. Another adaptive quality of such networks is their modular design [40], making it possible to attach or detach an arbitrary number of layers or interchange components in order to evaluate different design choices.

It should be noted that the actual learning process of deep learning is not yet fully understood. This fundamental problem is subject of various publications in recent years. Progress has been made in the visualization of the aforementioned feature hierarchy in convolutional neural networks [53, 68], illustrating the filter response of the hidden layers. Dieleman et al. [12] affirm this hierarchy in audio-related tasks, showing that lower-level layers in their network pick up low-level features. Whereas higher-level convolutional filters in their model yield more complex musical concepts. Choi et al. [7] further enhanced this concept by producing audio signals from deconvolved spectrograms using inverse-transformation, hence, making it possible for researchers to study the audible results of specific hidden layers.

Pons et al. [46] propose a time-frequency architecture, which makes predictions on a musically motivated feature extraction process. This is done by combining results from two separate CNNs, which incorporate filter dimensions for either time-dependent features or frequency features. Thus, allowing the network to detect music-related patterns. However when looking at the provided results in Figure 7, the accuracy of black-box architectures [7, 43] are on par with the proposed time-frequency structure. Black-box means, that the network's components were chosen without a musically motivated design in mind. While Pons et al. show the viability of models based on musical features, it should be questioned whether this could be attributed to the adaptive nature of deep neural networks [36] instead of the music-related filter sizes. The time-frequency architecture manages to reduce the number of parameters from 3 million to around 200.000, decreasing the computational expenses.



| Architecture | Input (M,N) | Filter shape (m,n) | # param. | Max-pool | Accuracy: mean ± std 10 cross-fold validation |
|---|---|---|---|---|---|
| ***Black-box*** | (40,80) | (12,8) | 3.275.312 | (4,1) | **87.25** ± 3.39 % |
| *Black-box* | (40,250) | (12,200) | 2.363.440 | (4,1) | 82.80 ± 5.12 % |
| ***Time*** | (40,80) | (1,60) | 7.336 | (40,1) | **81.79** ± 4.72 % |
| *Time* | (40,250) | (1,200) | 19.496 | (40,1) | 81.52 ± 3.87 % |
| *Frequency* | (40,80) | (30,1) | 3.816 | (1,80) | 59.45 ± 5.02% |
| ***Frequency*** | (40,80) | (32,1) | 3.368 | (1,80) | **59.59** ± 5.82 % |
| *Frequency* | (40,80) | (34,1) | 2.920 | (1,80) | 58.17 ± 3.58 % |
| *Frequency* | (40,80) | (36,1) | 2.472 | (1,80) | 57.88 ± 5.38 % |
| *Frequency* | (40,80) | (38,1) | 2.024 | (1,80) | 57.45 ± 5.93 % |
| *Frequency* | (40,80) | (40,1) | 1.576 | (1,80) | 52.43 ± 5.63 % |
| *Time-Frequency* | (40,80) | (1,60)-(32,1) | 196.816 | (40,1)-(1,80) | 86.54 ± 4.29 % |
| ***Time-FrequencyInit*** | (40,80) | (1,60)-(32,1) | 196.816 | (40,1)-(1,80) | **87.68** ± 4.44 % |

Figure 7: Four networks studied by Pons et al. regarding the viability of musically dependent architectures.

With these unsolved questions in mind, this thesis will rely on state-of-the-art computer vision models, namely the Inception-ResNet-v2 [59] architecture. This decision could be affirmed by the notion that spatial dependencies in generic image recognition tasks are as important as in visual representations of audio material, hence, making such models applicable to both use cases. Furthermore, [20] suggests using state-of-the-art computer vision models such as ResNet-50 [19] and Inception-v3 [61] for audio classification tasks, as they outperform simple fully connected networks and previous image classification models.

## 5.2 Inception-ResNet-v2

In [59], Szegedy et al. investigate the viability of residual connections proposed by He et al. [19] in conjunction with their latest Inception model [61]. While He et al. argue that such components are essential for training of very deep networks, Szegedy et al. oppose this claim with their results. However, it is shown that the implementation of residual connections improves training speed, legitimizing their use in addition to the Inception architecture. Utilizing residual connections allows gradients of specific layers to circumvent non-linear activations of subsequent layers. It was shown [33] that this process can drastically improve the geometry of loss landscapes in very deep networks, simplifying the training process. When visualizing a loss function (Figure 8) that relies on skip connections, its topology appears to be much smoother compared to models without such components.

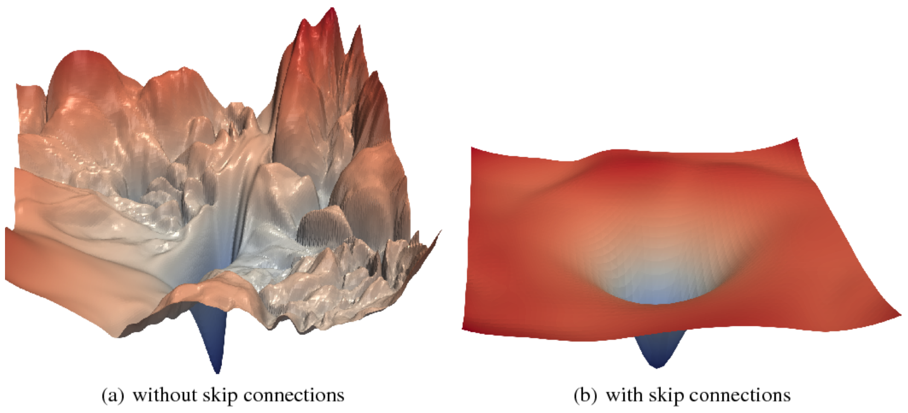


Figure 8: Visualization of a loss function with (right) and without (left) skip connections.

The newly proposed Inception-v4 [59] exhibits a simplified structure in order to dismiss the unnecessary complexity of previous versions. This model solely relies on a deep convolutional network without incorporating the aforementioned residual connections. However, the authors additionally introduced Inception-ResNet-v1 and Inception-ResNet-v2, which make use of the concept proposed by He et al. [19]. As pictured in Figure 9, Inception-ResNet-v1 yields similar results to Inception-v4 and Inception-ResNet-v2 when training for 60-100 epochs. While Inception-ResNet-v1 has a similar computational cost to Inception-v3, its top-5 and top-1 error is better in the mentioned epoch range.

Inception-ResNet-v2 and Inception-v4, however, excel their relatives by a significant margin when training for longer epochs. While Inception-ResNet-v2 and Inception-v4 match in their raw resource demands, Inception-ResNet-v2 exhibits a significantly faster step time and yields slightly better results in practice. Therefore Inception-ResNet-v2 is the model of choice for this thesis.

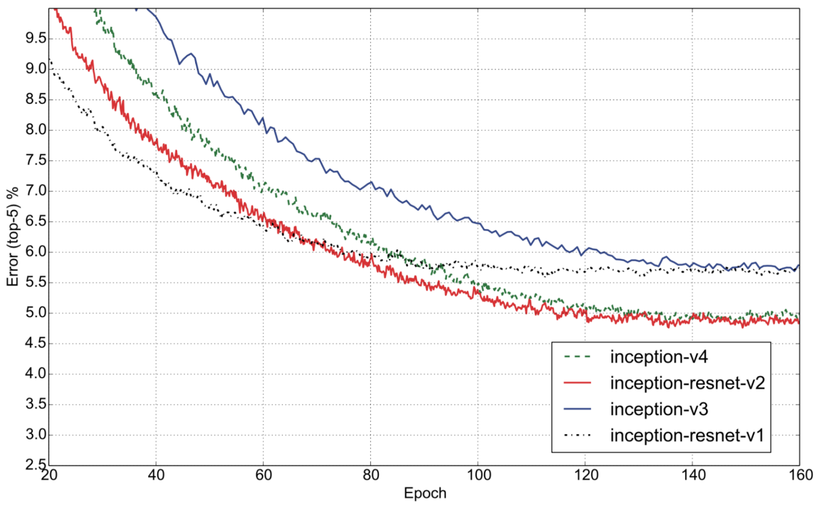


Figure 9: Comparison of several alterations of the Inception network with regard to the top-5 error metric.

Inception-ResNet-v2 utilizes ReLU activation functions throughout its network, commonly denoted as $g(z) = \max\{0, z\}$. The operation is examined in detail in Figure 10. ReLUs have been advocated as a reliable default component by various publications in the past [39, 16].

### 5.2.1 Residual Inception Block

While pure inception blocks in Inception-v4 rely on average pooling layers next to the convolution process, Inception-ResNet architectures interchange such components with the residual connections of [19]. This is done by utilizing additive merging of the convolution signal and the initial input value, which can be further studied in Figure 11. In order to add both signals together, a 1x1 filter-expansion layer has to be attached to the convolution block. Doing so scales up its dimensionality, now matching the depth of the original signal. It should be noted that aforementioned pooling layers are still present in the reduction blocks.

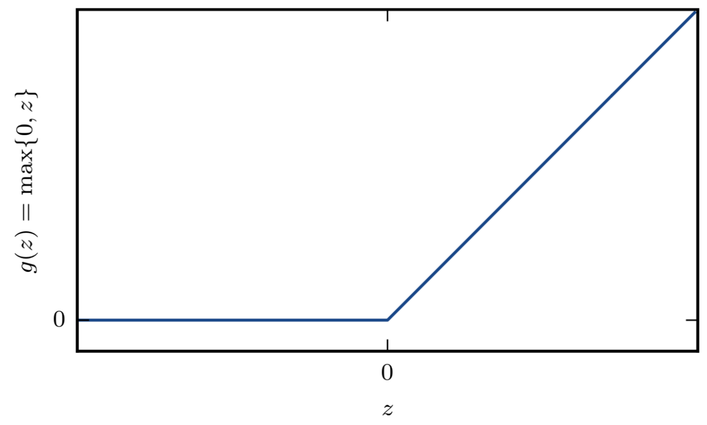


Figure 10: Graph of a rectified linear unit. ReLUs present themselves as efficient solutions for gradient-based optimization due to their partly linear nature. While positive values are processed in a linear fashion, negative inputs will return 0 as output.

## 5.3 Filter Shape

As mentioned earlier, a convolutional layer learns local patterns in its input space by applying windows of pre-defined size on the data. Such windows are commonly referred to as filters or kernels. State-of-the-art models in the field of computer vision [28, 54, 59] tend to use square-shaped filters of size 3x3, 5x5. When deciding on a filter size, one has to make a trade-off between resource expenses and expressiveness. Due to their success, MIR researchers have adopted such configurations and integrated them into their own architectures [44, 18, 64].

[46] suggests to have a more conscious mindset when choosing the filter dimensions, as filters in image recognition models are chosen with spatial dependencies in mind. Audio-related tasks on the other hand often rely on analyzing spectrograms, which consist of a time and a frequency domain. [67] evaluated different filter shapes (5x5, 3x8) using the LeNet [31] architecture and found the 3x8 kernel to yield more relevant results. Hence, it may be beneficial to to use wider filter settings when analyzing temporal dependencies, while higher filters would be more adequate for timbral features. Such boundaries of course can vary depending on the audio material at hand.

[23] shows that 2D convolution performs better than 1D convolution on environmental audio samples. The publication further suggests to experiment with variable-sized filters. However, as shown in Section 5.1 musically motivated architectures tend to achieve similar results to traditional deep learning networks. Therefore using a sophisticated computer vision model like Inception-ResNet-v2 could be just as appropriate for the task at hand.

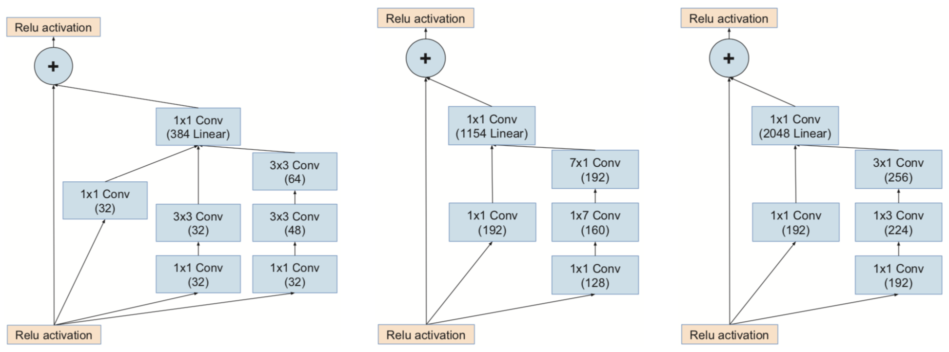


Figure 11: Inception-A (35x35), Inception-B (17x17) and Inception-C (8x8). Residual connections used to merge the separate signals. Notice how the 1x1 Conv layer increases the filter bank dimensionality.

## 6 Experiments

The entirety of the audio samples is down-mixed to mono and resampled to 12 kHz. The duration of the initial signals is reduced to 1 second in order to maximize frequency content in the spectrogram. A further notion behind this decision is that such a small time frame improves the discovery of timbre patterns by the learning algorithm. The log mel-spectrogram of each recording is computed using 128 mel-bins, 2048 samples for the window size and a hop-size of 1024. A natural logarithm is used to compress the magnitude of the resulting representation. Preprocessing until here is done using librosa [37] - an audio and music signal analysis library for python. A random sample of the obtained log mel-spectrograms can be seen in Figure 12.

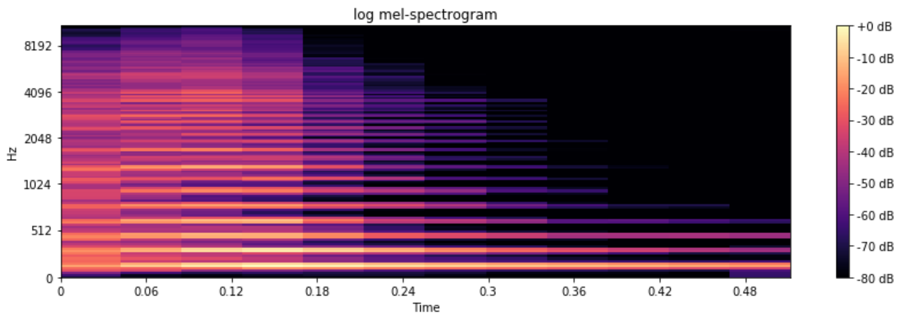


Figure 12: Log mel-spectrogram of a cello sample.

Inception-ResNet-v2 is being implemented using Keras [4], a high-level API for neural networks. Keras provides many state-of-the-art convolutional neural networks, thus, making it the ideal API for research that depends on fast experimentation. Furthermore, CPU and GPU processing is supported. TensorFlow [1] is used as backend, which allows expressing abstract computations (e.g. matrix multiplication) as operations in dataflow graphs. Such computations can then be executed on multiple kernels or devices.

The samples of each individual class are split, leaving the training set with 80 % of the initial data while 20 % are separated for testing. This training set is split again with the same ratio, now isolating 20 % of the training data as a validation set. This results in 829 images for training, 212 for validation and 260 for testing. Afterwards, all images are rescaled to size 150x150x3 using the ImageDataGenerator module in Keras.

The model is tested on the data set with 4 different settings, some including an additional layer in the classifier. Only the convolutional base of the Inception-ResNet-v2 architecture is used, since the fully connected part of initial model is trained on 1000 categories. A new fully connected classifier is added on top of this base structure. While there is an increase in adaptive gradient techniques such as ADAM [26] being employed in deep neural networks [14], recent research [65] has shown that such methods perform worse than SGDs in their ability to generalize. Hence, this text follows the instructions of [59], utilizing stochastic gradient descent with a decay of 0.9 and a learning rate of 0.045. During training, the decay parameter adjusts the learning rate after each batch update. In some setups, Nesterov's accelerated gradient (NAG) [57] is implemented as the momentum method in order to accelerate gradient descent. Additionally, every setup has a separate momentum parameter of 0.9.

The model is trained using transfer learning [10] with Inception-ResNet-v2 being pre-trained on the ImageNet [11] dataset, a database of 3.2 million annotated images. Transfer learning allows to reuse weight parameters that were already trained on a separate data set. Therefore, only the fully connected layers have to be trained on the new input. The convolutional base of Inception-ResNet-v2 is solely used for feature extraction, resulting in an output feature map of size 3x3x1536. These features are then passed through the newly attached classifier. For classification, a softmax classifier using cross entropy loss [40] $-\sum_j y^{(j)} \log \sigma(o)^{(j)}$ is utilized. As for the rest of the fully connected subnetwork, several settings are being examined. Each setup is trained for 50 epochs, evaluating accuracy and loss for the training, validation and test set. Computing is done on an internal quad-core CPU (2,3 GHz Intel Core i7).

## 6.1 Results

Figure 13 compares the validation accuracies of all settings over 50 epochs. The results highlight the difficulty that is involved when trying to find the appropriate components for the task at hand. While setup 4 with both NAG

[4]github.com/keras-team/keras

and an additional fully connected layer (1024 processing units) outperforms its competitors by reaching a final validation accuracy of 86 %, just leaving out NAG drastically impacts the models learning process. Looking at setup 2, NAG seems to be the driving factor for the bad performance. Since the same setting without nesterov's accelerated gradient approaches an accuracy of over 80 %. Interestingly, the baseline setup (blue line) is on par with the setup consisting of an augmented classifier. Increasing the batch size from 20 to 30 resulted in a faster initial decrease in training loss, while validation accuracy stagnated quickly. By adding a second fully connected layer to the classifier, this problem was prevented from occurring. Batch sizes bigger than 30 didn't seem to affect performance.

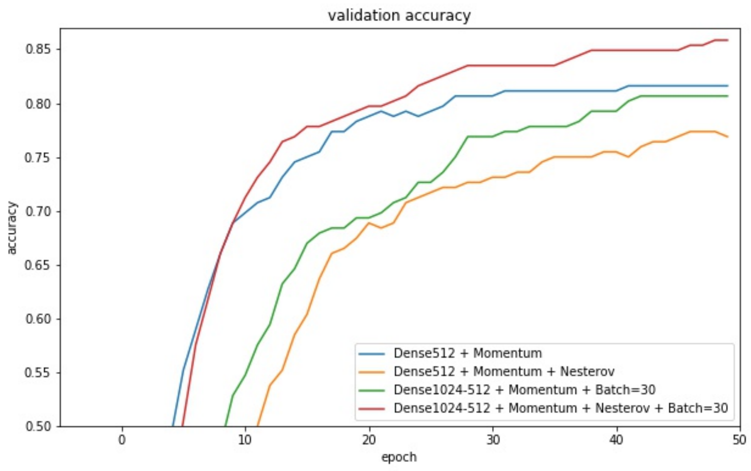


Figure 13: Validation accuracies of the different setups. The baseline setup consists of a single fully connected layer with 512 units and momentum of 0.9. Altered setups include NAG, an additional fully connected layer and/or an increased batch size.

Looking at Figure 14, similar observations can be made about the training accuracy. The most astonishing finding here is the performance of the baseline setup compared to the most expanded setup. Both configurations yield a training accuracy of about 89 %. However, Figure 13 shows the stagnating improvements of the baseline setup regarding validation accuracy at around 30 epochs, while setup 4 trends upwards. The final results on the test set can be seen in Table 2. Test loss and test accuracy approve of the findings in the previous Figures. Setup 1 and 4 exceed their counterparts, setup 4 yielding the highest test accuracy.

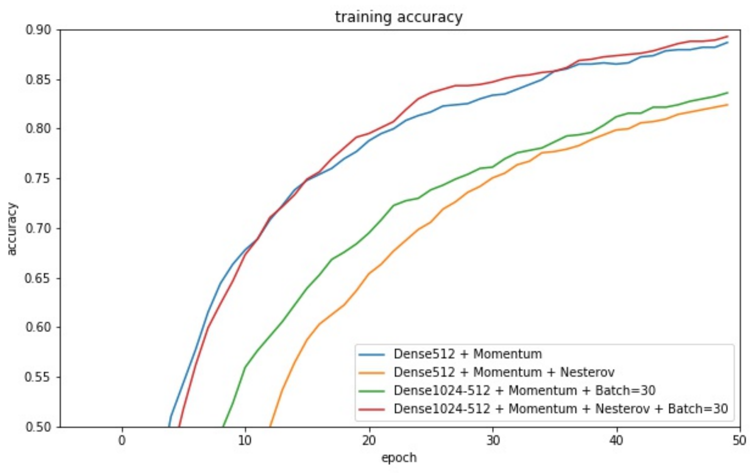


Figure 14: Training accuracies of all configurations.

| | setup 1 | setup 2 | setup 3 | setup 4 |
|---|---|---|---|---|
| settings | Dense 512 Momentum | Dense 512 Momentum NAG | Dense 1024-512 Momentum Batch = 30 | Dense 1024-512 Momentum NAG Batch = 30 |
| test loss | **0.69** | 0.89 | 0.83 | 0.75 |
| test acc. | 0.77 | 0.74 | 0.77 | **0.80** |

Table 2: Results for test loss and test accuracy.

### 6.2 Discussion

The results show that a pre-trained Inception-ResNet-v2 architecture is adaptable to recognizing musical instruments. Training for only 50 epochs already yields meaningful performance gains. Transfer learning presents itself as a viable tool for musical instrument classification, even though the new data set consisting of spectrograms vastly differs from images in the ImageNet data set. Setup 4 seems to include the highest performing settings, consisting of a classifier with two fully connected layers (1024, 512), the addition of NAG and an increased batch size.

Some configuration details however are still unclear, especially regarding long-term efficiency. Models have to be trained for more epochs (100-300) in order to see if SGD is able to mitigate small changes in the settings. Additionally, higher initial learning rates (0.1-0.01) have to be examined, as changing this parameter resulted in drastic performance fluctuations. More settings have to be evaluated to gain a better understanding about how each component influences performance. E.g. additional layers of various sizes should be added to the classifier, while also considering the application of a dropout mechanism [55] to circumvent overfitting the model. The homogeneous property of the used data set leaves open the question, if more variety in sound quality and recording spaces would improve performance.

# 7 Conclusion

Inspired by the adaptive nature of deep neural networks, this thesis presented a way of utilizing this property in the context of musical instrument classification. After evaluating related work and promising approaches, convolutional neural networks have been chosen as the fundamental architecture. Time was spent explaining the structure of such networks, while also elaborating on the underlying learning process. The proposed data set consisting of various recordings of musical instruments was transferred into the visual domain using log mel-spectrograms.

Inception-ResNet-v2 was presented as a viable solution for aforementioned classification tasks. After discussing other versions of the original Inception model, it is made clear why this specific alteration was selected. The final model incorporates the convolutional base of Inception-ResNet-v2 for feature extraction and a separate fully connected part for classification. The convolutional base was pre-trained on the ImageNet dataset, while the fully connected layers were initialized with random weights. The implementation of the proposed network was done using Keras with TensorFlow backend. Training on the data set was solely performed by the fully connected classifier, using a softmax activation layer with cross entropy loss for measuring performance. Stochastic gradient descent was applied to gradually minimize the loss score.

Several setups of the model are tested, providing insightful information about the usefulness of certain components and parameters. The different configurations were trained for a total of 50 epochs each, delivering presentable results. Both loss and accuracy metrics were evaluated for the training, validation and testing stage. Results were compared and discussed thoroughly, while further elaborating on possible enhancements and approaches in future research. The overall information gathered throughout the experiments legitimizes the use of state-of-the-art image recognition networks for musical instrument classification.

## 7.1 Future Work

While Inception-ResNet-v2 has been proven to be a useful solution, similar networks like Inception-v4 should be tested with comparable settings in order to evaluate the the performance boost gained by adding residual connections. In order to exhaust the performance threshold of the current results, more articulations and instruments have to be added to the data set. Training on similar