JOHANNES KEPLER
UNIVERSITY LINZ

Author
Fabian Paischer

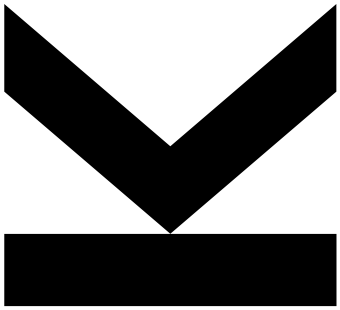
Submission
Institute of Computational Perception

Thesis Supervisor
Gerhard Widmer, Univ.-Prof., Dr.

Assistant Thesis Supervisor
Hamid Eghbal-Zadeh, M.Sc.

Month Year
June, 2018

Improving Generalization of Deep Convolutional Neural Networks for Acoustic Scene Classification



Bachelor's Thesis

to confer the academic degree of

Bachelor of Science

in the Bachelor's Program

Bioinformatics

JOHANNES KEPLER UNIVERSITY LINZ
Altenberger Str. 69
4040 Linz, Austria
www.jku.at
DVR 0093696

Contents

1 Affidavit

I hereby declare that I have worked on my bachelor's thesis independently and used only the sources listed in the bibliography. I hereby declare that, in accordance with Article 47b of Act No. 111/1998 in the valid wording, I agree with the publication of my bachelor / master / dissertation thesis, in full / in shortened form resulting from deletion of indicated parts to be kept in the Faculty of Science archive, in electronic form in publicly accessible part of the STAG database operated by the University of South Bohemia in Ceské Budejovice accessible through its web pages. Further, I agree to the electronic publication of the comments of my supervisor and thesis opponents and the record of the proceedings and results of the thesis defence in accordance with aforementioned Act No. 111/1998. I also agree to the comparison of the text of my thesis with the Theses.cz thesis database operated by the National Registry of University Theses and a plagiarism detection system.

Linz, June 2018

Abstract

In recent years deep learning has become one of the most popular machine learning techniques for a vast variety of complex problems. An example for such a task is to mirror the human auditory system to classify audio recordings according to the location they were recorded in. This work focuses mainly on the Acoustic Scene Classification task proposed by the IEEE DCASE Challenge. The dataset for Acoustic Scene Classification consists of recordings from distinct recording locations. The aim of the challenge is to classify an unseen test set of recordings. In the challenge of 2016 the training and test set did not differ significantly. In the challenge of 2017, however, the test set originated from a different distribution, implying a strong need for generalization. In the course of this work, the initial implementation consisting of a Deep Convolutional Neural Network for the DCASE 2016 challenge submission (done in Lasagne) was re-implemented in Keras. An extension of the ADAM optimizer (AMSGrad) was investigated for improvement in generalization. Other submissions to the DCASE 2017 challenge suggest that different types of spectrograms might be key for better generalization. Therefore experiments utilizing different kinds of spectrograms were conducted. Furthermore, different interpolation algorithms were used for data augmentation, with some of them yielding significant improvements in classification accuracy and generalization. For different spectrogram dimensions, slight adjustments in the network architecture also resulted in a performance gain. To better understand what different models "see" and what they focus on, their filters, and activations were visualized and compared for differences. Finally the adjustments which led to better generalization on the dataset of the DCASE 2016 challenge were tested on the dataset of the DCASE 2017 challenge, leading to an improvement over all submissions to the DCASE 2017 challenge from the Institute of Computational Perception.

2 Introduction

Humans are effortlessly able to immediately distinguish between different kinds of audio recordings from the environment and classify them according to the location where they have been recorded, such as beach or park. The structure of such recordings is rather complex with the occurrence of different sound events or changes in the pitch. Such sound events may be people talking, cars passing by, or the sound of glass breaking.

To realize a Machine Learning approach for Acoustic Scene Classification we are forced to exploit our knowledge on how recordings of different locations are produced and structured for being able to design algorithms that can learn from audio and produce reasonable results. A possibility of extracting information out of audio recordings is to create audio spectrograms. These spectrograms can be utilized as input for various Machine Learning algorithms.

Deep Learning is a branch of machine learning which has proven its effectiveness on different sources of data in recent years and is yet to be fully explored. However it is applicable to a wide range of complex problems. One such problem is proposed by the IEEE Dcase Challenge. DCASE is shorthand for Detection and Classification of Acoustic Scenes and Events.

The DCASE Community [1] is a community of researchers from various groups focused on tackling challenging audio-related problems. They already organized three challenges, which took place in 2013, 2016, and 2017, respectively. The last DCASE challange consisted of four tasks, involving Acoustic Scene Classification, detection of rare sound events, sound event detection in real life audio, and large-scale weakly supervised sound event detection for smart cars.

The dataset for the Acoustic Scene Classification task consists of 15 predefined classes ({*beach*, *bus*, *cafe/restaurant*, *car*, *city center*, *forest path*, *grocery store*, *home*, *library*, *metro station*, *office*, *park*, *residential area*, *train*, *tram*}), characterizing the location in which the audio samples have been recorded. In the DCASE 2016 challenge, the 3-5 minute recordings were split into 30 second frames, whereas in the DCASE 2017 challenge they were split into 10 second frames. At the beginning of every challenge, a development dataset [2, 4] including a cross-validation setup was published on which the models should be trained and evaluated. Finally an evaluation dataset [3, 5] was published on which the final predictions had to be made and submitted. In 2016 the development and the evaluation dataset were rather similar. In the 2017 challenge the datasets were differently distributed, making the prediction process on the evaluation dataset much more difficult.

In 2016 and 2017 the Institute of Computational Perception of the Johannes Kepler University participated in the Acoustic Scene Classification task. The submission for the DCASE 2016 challenge [6] consisted in a late fusion approach combining binaural i-vectors and deep convolutional neural networks, and got ranked at first place. The same method was used to prepare a submission for the DCASE 2017 challenge [7], however failing to achieve the success of the DCASE 2016 challenge. The reason for that was that the submitted models apparently overfitted on the training data and were not able to generalize into the unseen test set, which had a different distribution than the training and validation

set. Although this problem appeared in the majority of the deep learning based methods participating in the challenge, a similar approach [8] managed to generalize much better on the evaluation dataset using the exact architecture designed by CP-JKU.

The evaluation datasets of both years are publicly available, therefore in this thesis, we will investigate the difference in generalization of the submitted models. Also differences and improvements in generalization for different data preprocessing steps and recently developed optimization algorithms will be studied in the conducted experiments. Finally we will get an insight of which models are able to better generalize on the unseen test set.

3 Audio Signal Processing

The audio recordings have to be preprocessed in a way that machine learning algorithms are actually able to learn from the data. We aim to extract abstract information out of audio recordings by computing different types of spectrograms. Even a short recording would yield a lot of data points which are not independent from each other. To get a compact representation of the oscillations over time, sinusoids can be used as prototypical oscillations. The time domain of an audio excerpt is multiplied by all sine and cosine functions which have a complete number of oscillations over the recording. Summing up these products over time yield spectral bins [9]. Computing spectrograms with differing number of spectral bins leads to different sizes in the frequency domain of the final spectrogram.

3.1 Raw Spectrograms

Usually the spectrograms are generated by a discrete fourier transform (DFT) or a short term fourier transform (STFT). The DFT applies the fourier transformation on a time dependent signal consisting of complex values. The original input is transformed into a frequency domain representation over time. Usually applying the DFT on audio recordings results in three dimensional surfaces, where the three dimensions consist of time, frequency and amplitude. We can use these three dimensional surfaces as 2D images of frequency over time with the amplitude being depicted in different colors on a dB scale. The STFT is a variant of the DFT using a sliding window over the time domain to extract the final spectrogram. For the creation of audio spectrograms mainly the python package madmom [10] was used.

3.2 Constant Q Transform

The Constant-Q Transform is related to the Fourier Transform. It uses an additional parameter, the "quality factor", which is calculated by frequency over bandwidth. In general this transformation method is very well suited for music processing, because it reduces the frequency resolution for higher frequency bins while increasing the resolution for lower frequencies.

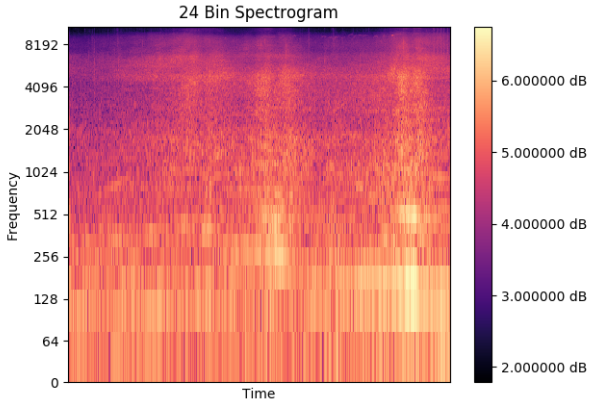


Figure 1: 24 Bin Spectrogram

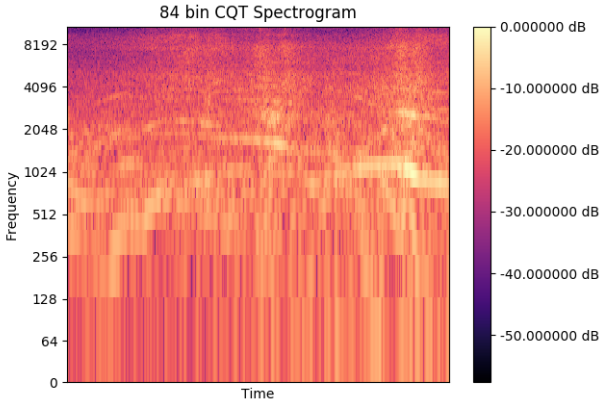


Figure 2: 84 Bin CQT Spectrogram

Figure 1 shows a sample spectrogram using 24 frequency bins for creation. Figure 2 shows a CQT spectrogram which was created using 84 frequency bins. By taking a look at the scaling of the dB colorbar we can observe the difference in frequency resolution between the spectrograms. This shows that the frequency resolution of the CQT spectrogram is higher for lower frequencies. The CQT spectrograms were extracted using the python package Librosa [11].

4 Interpolation Algorithms

As mentioned above the submission of [8] managed to generalize much better on the unseen test set of the DCASE 2017 challenge than the one from the Institute of Computational Perception [7]. A possible reason for this outcome could be the resizing of the spectrograms as they reported in the paper. According to [8], the CQT spectrograms were extracted using 84 bins resulting in a frequency domain of size 84 while the CP-JKU submission did only utilize raw spectrograms. For fitting the data into the input of the deep convolutional neural network an upsampling step might have been performed. Usually interpolation algorithms are used for images, but can also be applied to spectrograms. Hence, we investigate various interpolation algorithms as explained in the following sections.

4.1 Nearest Neighbour Interpolation

For interpolating a discrete pixel in an image, the nearest neighbor interpolation selects the value of the nearest neighbor and fills the area around the pixel with the exact same value. In figure 3 the principle of the nearest neighbor algorithm is depicted.

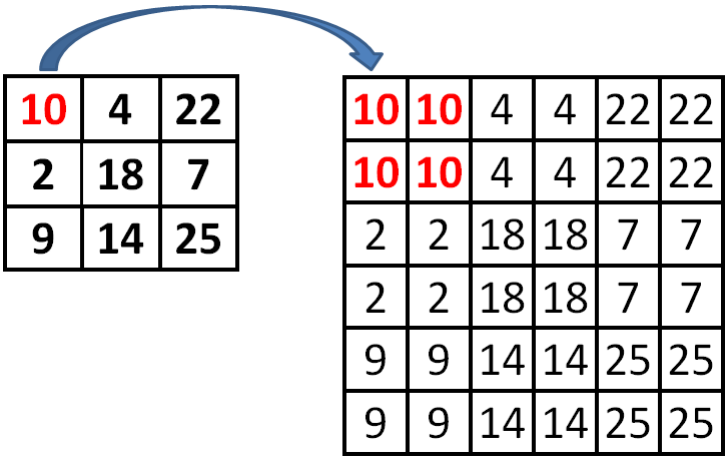


Figure 3: Nearest Neighbor Interpolation

4.2 Bilinear Interpolation

The bilinear interpolation is the two dimensional variant of linear interpolation. Linear interpolation fits a line to existing data points to infer new data points. In the bilinear interpolation this is done in 2D space, therefore a new data point is dependent on close data points in x and y direction.

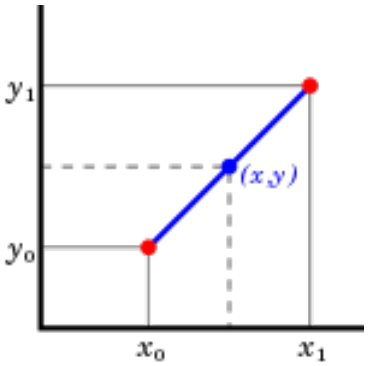


Figure 4: Linear Interpolation

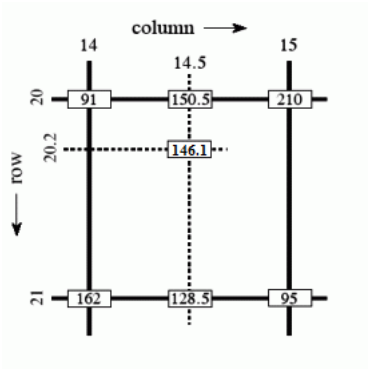


Figure 5: Bilinear Interpolation

Figure 4 depicts a simple linear interpolation, where the red points are given and the blue point is inferred by the linear interpolant. Figure 5 shows the same task in 2D space. Both figures were taken from Wikipedia.

4.3 Cubic Interpolation

Rather than fitting a linear curve onto the pixels of an image, the cubic interpolation fits a third degree polynomial across a set of data points. In figure 6 the fitting of a cubic interpolant is depicted.

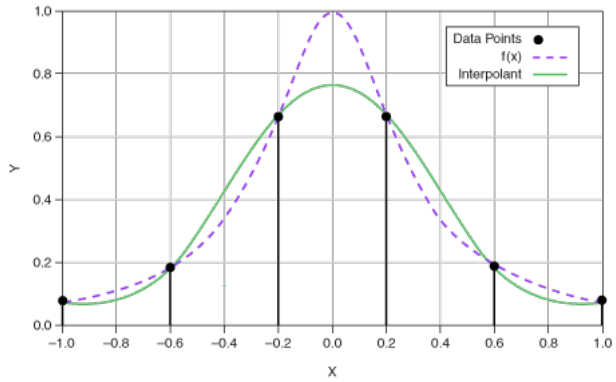


Figure 6: Cubic Interpolation

4.4 Bicubic Interpolation

The bicubic interpolation is a variant of the cubic interpolation and was designed for two dimensional data. In figure 7 the application of bicubic interpolation can be seen.

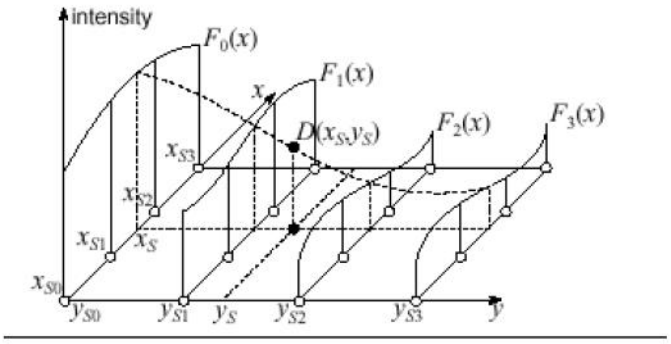


Figure 7: Bicubic Interpolation

4.5 Lanczos

The most recently developed method for interpolation is the Lanczos interpolation [12]. It makes use of the so-called Lanczos kernel, which is a cardinal sine function. This kernel is windowed over a second and longer cardinal sine function and the sum of the windows is evaluated at the newly created points. Figure 8 depicts a discrete signal (black dots) onto which the kernel is fitted. The red and green sine function show the cardinal sine for the creation of novel data points.

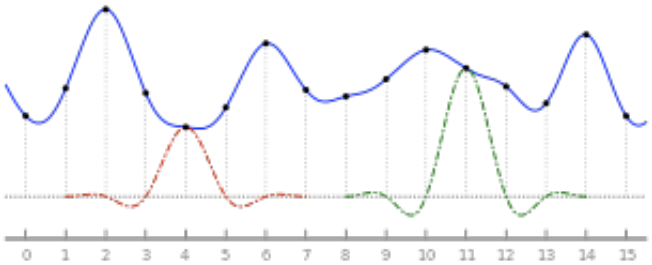


Figure 8: Lanczos Interpolation, source: Wikipedia

5 Machine Learning

As mentioned in [9], the process of machine learning can be split up into two general parts:

- Formulize a complex problem such that its solution can be represented as a mathematical function.
- adjusting parameters of this function by looking at the data.

Furthermore there are two key concepts for machine learning:

- Optimization
- Regularization

In recent years a variety of novel optimization algorithms and regularization techniques have been developed. The development of multi-layer perceptrons [13] opened the path to deep learning, which became more and more popular over the last few years. One type of deep learning are Convolutional Neural Networks [14].

5.1 Convolutional Neural Networks

Although designed for image recognition and classification, Convolutional Neural Networks have also proven their effectiveness when it comes to Acoustic Scene Classification. A convolutional neural network consists of convolutional layers. Figure 9 depicts such a layer. These layers contain filters with a particular kernel size that are slid over the input image to extract information. An activation function is applied to the extracted information leading to the final output. Usually rectified linear units (ReLUs) are used as activation function for convolutional layers.

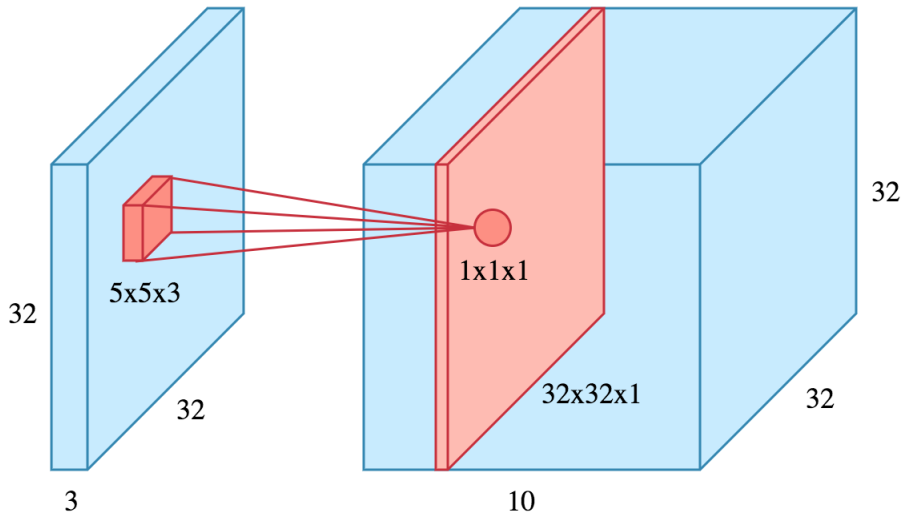


Figure 9: A (1x1) filter is applied onto an RGB image of dimension (32x32x3). The kernel slides over the image and extracts (1x1x1) pixels. After applying the activation function the original dimension of (32x32) is regained. In this example 10 filters are applied to the image, leading to a total output of 10 activations. Therefore the overall output is of dimension (32x32x10). Figure was taken from: Convolutional Neural Network, May 29 2018, `https://brilliant.org/wiki/convolutional-neural-network/`

With an increasing filter size, the resulting activations decrease in size. Figure 10 shows the effect of bigger kernel sizes.

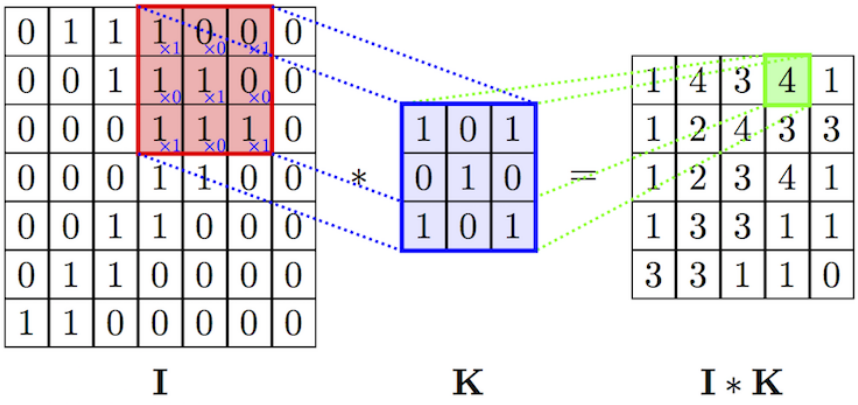


Figure 10: The effect of bigger kernel sizes

CNNs learn kernels that reflect specific aspects of the inputs such as edges. Usually pooling layers [15] are added between convolutional layers to further extract features that are relevant to the task at hand. This type of layer is a common practice for gaining invariant features. It aims to aggregate multiple features over a specific neighborhood. A maximum pooling layer for example extracts the highest value within a certain neighborhood, while an average pooling layer extracts the mean of the pixels within this area. For a more thorough comparison of the pooling techniques, see [16]. The following figure depicts the effect of a max pooling layer.

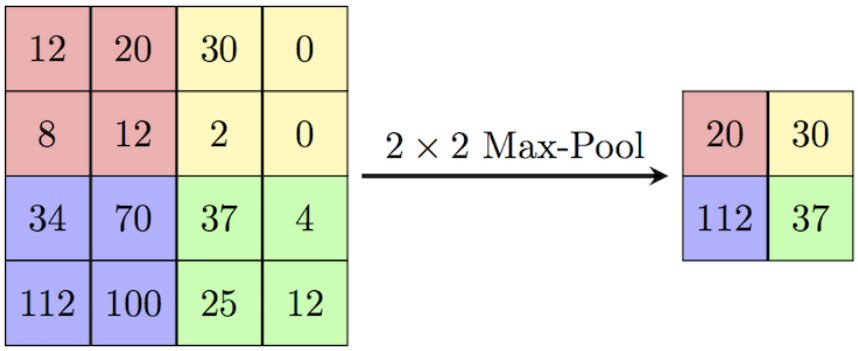


Figure 11: Max Pooling Layer

The last step before being able to classify the inputs is to feed the output of the final convolutional layer into dense layers and apply a softmax activation to get the probabilities for each class with respect to a certain input image. To transform the convolutional output in a way that it can be fed into the dense layers usually a Flatten layer or a Global Average Pooling layer is added. A flatten layer transforms each filter into a vector by simply concatenating all the rows of an activation. These vectors are then further concatenated to the total feature vector which can then be fed into the input of a dense layer. The Global Average Pooling layer computes the global average of each filter and concatenates the averages into a feature vector, such that the length of the resulting feature vector is equal to the number of the filters of the last convolutional layer. Therefore usually the number of filters of the last convolutional layer is designed to be equal to the

number of classes.

The main difference between those two layers is that the output of a flatten layer has to be fed into a fully connected layer, before applying a softmax activation for classification, whereas the softmax activation can be directly applied to the output of the global average pooling layer as the resulting feature vector is already of desired dimension. This point can be crucial in terms of generalization, as the additional dense layer adds a certain number of parameters and therefore increases the model complexity and the degree of overfitting. Therefore we avoid using any dense layers and replace them with global average pooling followed by a softmax.

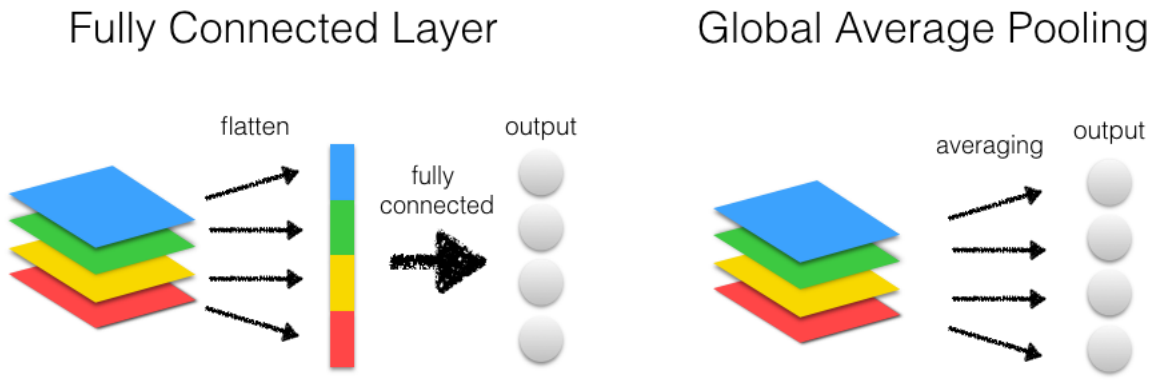


Figure 12: Difference Flatten Layer and Global Average Pooling Layer

Figure 13 depicts an example architecture of a DCNN. The most popular architectures of DCNNs are VGG-Style neural networks [17], AlexNet [18], GoogleNet [19], and ResNet [20].

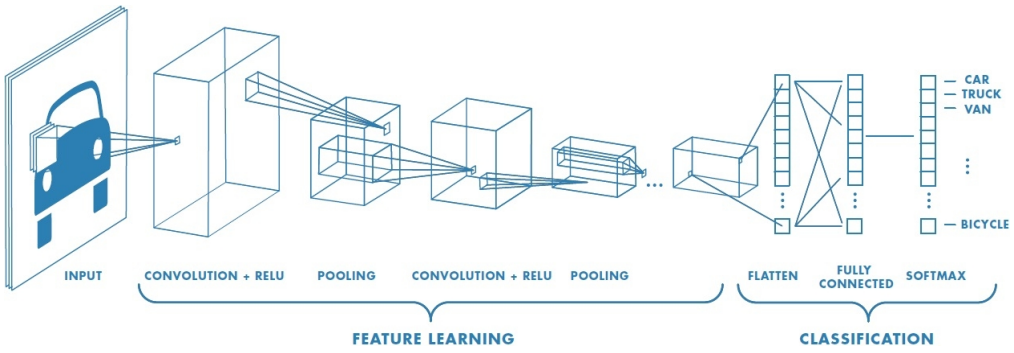


Figure 13: Architecture of a DCNN, taken from: [21]

5.2 Optimization and AMSGrad

After designing the network architecture, it is crucial to select an efficient training method and an appropriate optimizer. The most used optimizers for neural networks are SGD (Stochastic Gradient Descent [22]) and ADAM [23], which is an adaptive version of SGD.

The purpose of optimizers is to update the weights of the model by the gradients calculated based on the loss, and converge to a minimum in the loss function.

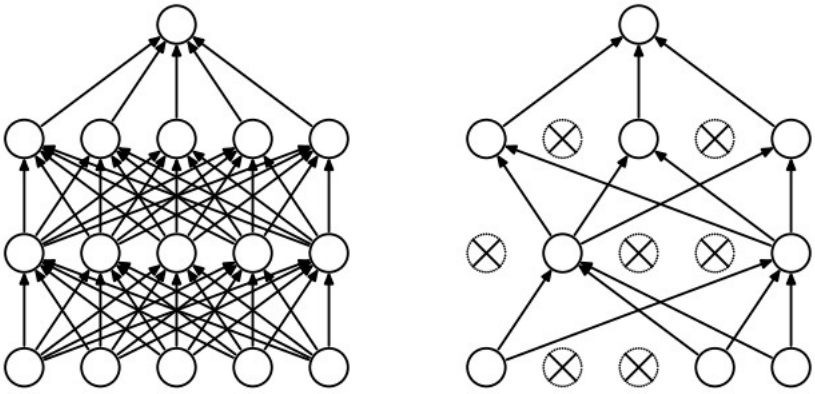
For large-scale machine learning approaches usually ADAM is the widely used optimizer as it requires less tuning and can be used out-of-the-box. However it appears that for some nonconvex optimization problems ADAM fails to converge to a minimum. To solve this problem a new variant of ADAM called AMSGRAD [23] has been developed. The paper [24] proves that the AMSGRAD optimizer manages to converge to a minimum for problems where ADAM does not converge. We compare both optimization methods applied on the data of the DCASE challenge.

5.3 Regularization

Overfitting on the training data leads to bad generalization. To avoid learning the training data by heart, additional regularization techniques can be included into the network, helping it to generalize better.

5.3.1 Dropout

The dropout technique shuts down specific nodes of a network according to a predefined rate. The dropped neurons can vary with each training epoch or in different layers. This introduces randomness in the layers and forces the network to not only rely on specific neighboring nodes for prediction. Figure 14 depicts a fully connected network to which dropout is applied.



(a) Standard Neural Net (b) After applying dropout.

Figure 14: Dropout applied to dense layers, taken from [25]

5.3.2 Batch Normalisation

Another method for helping the network to generalize is Batch Normalisation [26]. For training the data is divided into mini-batches. In the batch normalisation the mean and standard deviation of the mini batch is calculated and used for normalizing the output of a previous layer. As a consequence of that the weights of the next layer would not be optimal anymore and be adjusted by the optimizer. To prevent the change of all the weights due to that shift, two additional trainable parameters γ and *beta* are added to each batch normalization layer (depicted in Figure 15). This maintains the stability of the network and even improves generalization. A batch normalization is usually inserted

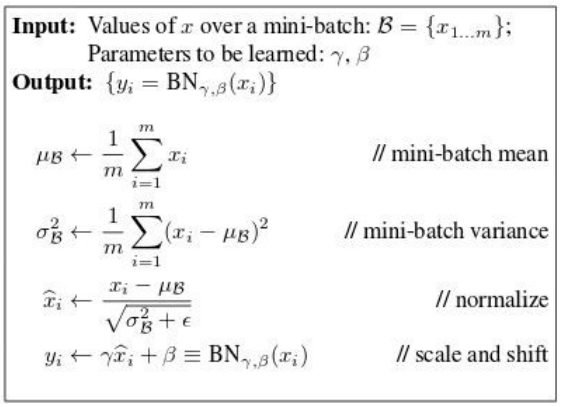


Figure 15: Batch Normalization algorithm, taken from [27]

either before or after the activation function of a layer.

5.4 Late Fusion

Late fusion is an approach to combine the predictions of several trained networks into one final prediction. The predictions of all the trained classifiers are concatenated to a score matrix and fed into the input of a simple network consisting of a single fully connected layer, thus a logistic regression. After cross-validation training of the models the logistic regression is trained on the validation set with the predictions of the trained models as input. This usually leads to an improvement in the classification accuracy. Figure 16 shows the idea of the late fusion of three classifiers.

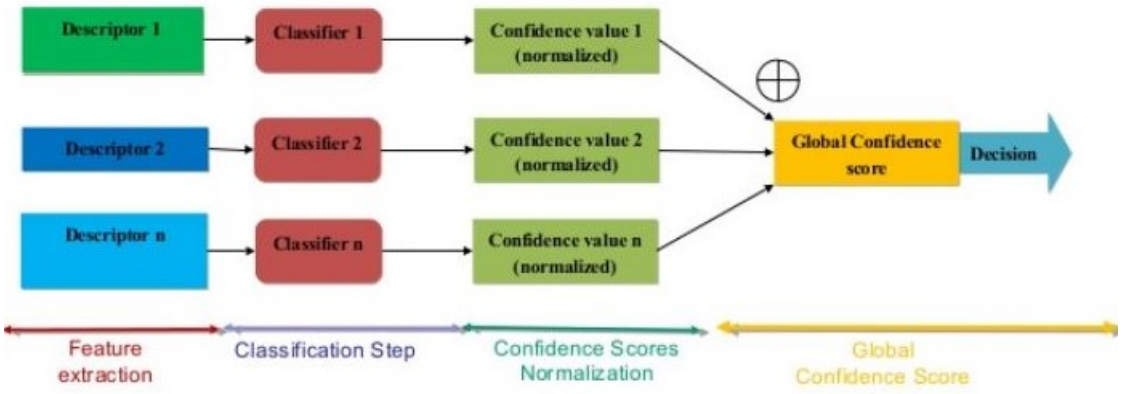


Figure 16: Late Fusion, taken from [28]

The normalization step shown in the figure above is not necessary as in our case, the output of the classifiers are vectors containing class probabilities, thus they are already normalized.

5.5 Keras and Lasagne

Keras [29] is a high-level API created for deep learning, written in python and can be run on top of Theano [30], Tensorflow [31], or CNTK [32]. In this work, Tensorflow with

GPU support was used. The initial implementation for the DCASE 2016 challenge [6] was done using the Lasagne framework [33]. Therefore the Lasagne code had to be rewritten in Keras, which showed some differences between the two APIs.

Lasagne is rather low-level, thus showing more similarities to Tensorflow than to Keras. Keras provides the Sequential API with which it is simple to create a deep learning model and to modify the training method. However to be able to access the model during training, specific callbacks have to be implemented. Also some default parameters in Keras differ from those in Lasagne. Those specifically being different are:

- **Initialization:**
  The Lasagne implementation for the DCASE 2016 challenge uses the HeNormal [34] initializer. This initializer is particularly suited for the ReLu activation. The weights are assigned a sample value of a truncated normal distribution. However the parameters of the distribution differ in both frameworks. Equation 1 shows the parameters of the initializer in Keras and equation 2 shows the parameters in Lasagne.
  Keras implementation: Lasagne implementation:

$$\mu = 0, \sigma = \sqrt{\frac{2}{fan_{in}}} \quad (1) \qquad \mu = 0, \sigma = gain\sqrt{\frac{1}{fan_{in}}} \quad (2)$$

  where fan_{in} is the amount of input units in the weight vector and $gain$ is a scaling factor. Usually for ReLu this factor is set to $\sqrt{2}$. Indeed the initialization of the convolutional layers led to changes in the performance of the model.

- **Batch Normalization:**
  The batch normalization can be easily added between layers, but it needs to be considered that the default parameters in Keras differ from those in Lasagne. Lasagne uses a parameter alpha which is initialized with 0.1, whereas Keras uses a parameter momentum which is set to 0.99. Although being named differently, those two parameters describe the fraction of the batch being used for calculation of the exponential moving average. In the backend of Keras the momentum parameter is assigned to (1 - momentum), therefore it is set to 0.99. To adjust the Keras implementation to the implementation of Lasagne the momentum parameter needs to be set to 0.9. Also the default values of the epsilon parameter differ in both frameworks.

- **Padding:**
  Padding is used to retain the size of the input of a convolutional layer by filling up the missing space with zeros. In both frameworks there are different modes which can be assigned to the padding parameter. In Keras, the parameter has to be one of *valid* or *same*, where *valid* means no padding at all, and *same* means preserving the input size of the layer by zero padding. In Lasagne padding can be one of *full*, *same*, *valid*, or an integer, where *full* means zero padding with one less than the filter size, *same* means padding with half of the filter size, *valid* means no padding at all, and if a single integer is supplied then a symmetric zero-padding corresponding to the integer size is performed.

Besides those three points, most of the other parameters can simply be taken from the Lasagne implementation and plugged in to the Keras implementation. However after

adjusting all the parameters there might still be differences in the training procedure performed by Keras.

5.6 Final Architecture

The network architecture used in this work (Figure 17) follows a VGG-style architecture proposed for object recognition. A modified version of VGG, adapted for audio applications was developed by the Institute of Computational Perception of the Johannes Kepler University in Linz and used for their submission for the DCASE 2016 and 2017 challenges [6, 7]. This architecture is further used in the submission which generalized much better on the unseen test set [8] in the DCASE 2017 challenge. This group reported improved results when exchanging the global average pooling layer to a flatten layer, although the increasing number of parameters due to the additional fully connected layer contradicts the assumption to reach better generalization. Moreover they used a late fusion approach of two models, one being trained on raw spectrograms and the other on CQT spectrograms, and obtained an increase of approx. 10 percentage points in classification accuracy on the unseen test set. In the following chapter the reason for this drastic improvement is investigated.

Input $1 \times 149 \times 149$
5×5 Conv(pad-2, stride-2)-32-BN-ReLu
3×3 Conv(pad-1, stride-1)-32-BN-ReLu
2×2 Max-Pooling + Drop-Out(0.3)
3×3 Conv(pad-1, stride-1)-64-BN-ReLu
3×3 Conv(pad-1, stride-1)-64-BN-ReLu
2×2 Max-Pooling + Drop-Out(0.3)
3×3 Conv(pad-1, stride-1)-128-BN-ReLu
3×3 Conv(pad-1, stride-1)-128-BN-ReLu
3×3 Conv(pad-1, stride-1)-128-BN-ReLu
3×3 Conv(pad-1, stride-1)-128-BN-ReLu
2×2 Max-Pooling + Drop-Out(0.3)
3×3 Conv(pad-0, stride-1)-512-BN-ReLu
Drop-Out(0.5)
1×1 Conv(pad-0, stride-1)-512-BN-ReLu
Drop-Out(0.5)
1×1 Conv(pad-0, stride-1)-15-BN-ReLu
Global-Average-Pooling
15-way Soft-Max

Figure 17: Network Architecture used in this work - (5x5) describes the kernel size, (pad, stride) describes padding and striding used, 32 is the number of filters of this layer, BN stands for Batch Normalization, which is applied after the activation of the layer, and ReLu describes the Rectified Linear Unit is used as activation

All of the convolutional layers of the final architecture are initialized by the HeNormal initializer [34]. The striding describes by how much the kernels overlap when sliding over the image. Also the input size of the model is set to 149x149. Usually the extracted spectrograms are much larger, therefore a sliding window approach is applied to the spectrogram, which divides the whole spectrograms into 149x149 patches. Afterwards these patches are used as training data. ADAM was used as optimizer with a learning rate set to 0.001. Also patience of 20 is applied, which means that if after 20 epochs the categorical cross-entropy loss did not decrease, the learning rate is halved. The cross-validation setup supplied by the DCASE challenge was used for training.

6 Results

The models were trained with the above mentioned training procedure. The cross-validation setup of the DCASE Challenge splits the development dataset into training and validation setup for each fold. For each epoch a model is trained on the training set and evaluated on the validation set. The measurement of the classification accuracy on the validation set is referred to as *Frame Validation Accuracy*. As we are interested to classify whole audio recordings we need to average the predictions of a model over all frames belonging to one recording. The final prediction is the highest averaged probability over all frames. The classification accuracy of entire recordings is referred to as *File Validation Accuracy*. Finally as the test set for the challenges is available we can also measure the classification accuracy for each test recording, which is referred to as *File Test Accuracy*. Figure 18 depicts the procedure of data preprocessing to training, hyperparameter tuning and collecting the results. After splitting the spectrograms into frames we split these frames up into training and validation set. In each epoch the model is trained on the training set and valiadated on the validation set. After training we can look at the file wise validation accuracies (in our case we can also look at the file-wise test accuracies) of the best model and tune the hyperparameters or adjust the network architecture and train a new model in order to be able to compare the trained models.

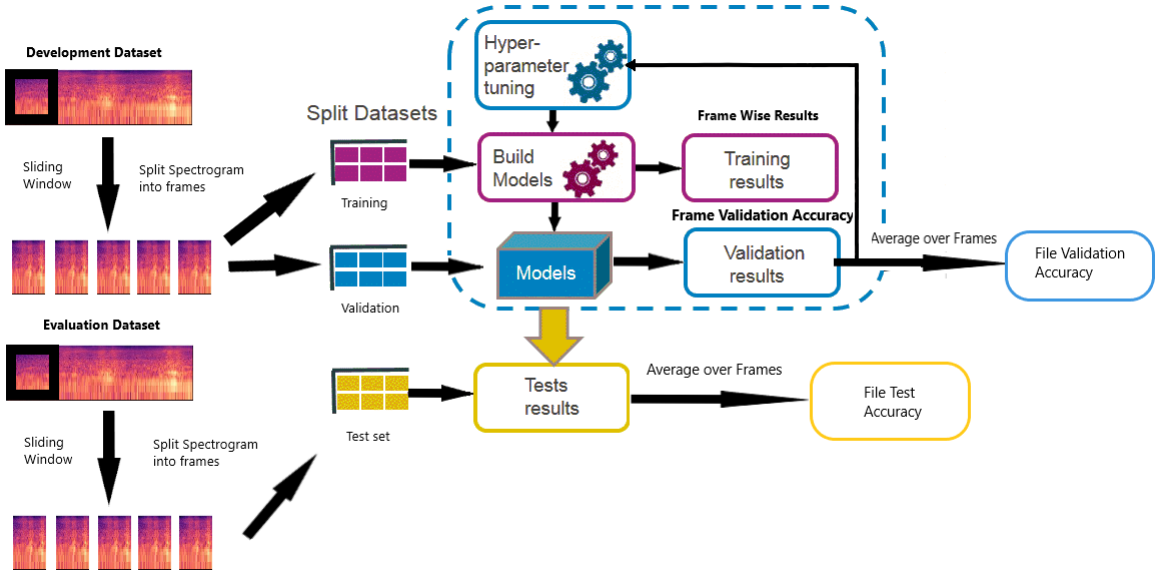


Figure 18: The spectrograms are split into frames, the frames are split into training and validation set, a new model is trained, results are collected

After training models on all 4 folds we need to fuse their predictions together to get an overall prediction of the model. In this work two different methods were compared, *Frame Wise Averaging* and *File Wise Averaging*. In the *Frame Wise Averaging* the average of the predictions of all 4 models is taken for each frame. Then a majority voting is conducted over all averaged frames to get the overall prediction of the model. On the other side for *File Wise Averaging* the average over all frames for each model is taken and after that a majority voting is conducted. This means that for *File Wise Averaging* the majority voting is conducted for 4 predictions, mainly the file wise prediction of each model. For *Frame*

Wise Averaging the majority voting is conducted for a vector containing the averaged predictions of 4 models for each frame, with the length of the vector being equal to the number of frames.

6.1 DCASE 2016

The first step was to recreate the results from [6] by using the Keras implementation.

6.1.1 Spectrograms

6.1.1.1 24 Bin Spectrograms

Using raw spectrograms as features, led to the following results:

Fold	Frame Validation Accuracy	File Validation Accuracy	File Test Accuracy
1	0.7436	0.8034	0.7692
2	0.6944	0.73	0.80
3	0.7309	0.80	0.80
4	0.7419	0.83	0.78
Frame Wise Averaged Test accuracy: 0.8538 File Wise Averaged Test accuracy: 0.8435			

Table 1: Classification Accuracies using raw spectrograms as features

Two methods for averaging over the models of the 4 folds were compared for higher classification accuracy. The first method averages the predictions of the four models of each frame and conducts majority voting to get the final prediction. The second method firstly averages the frame wise predictions for each file and then conducts majority voting to get the final prediction. As we can see the final classification accuracies differ depending on which method we choose. To get an impression of the learning process, the learning curves have been plotted:

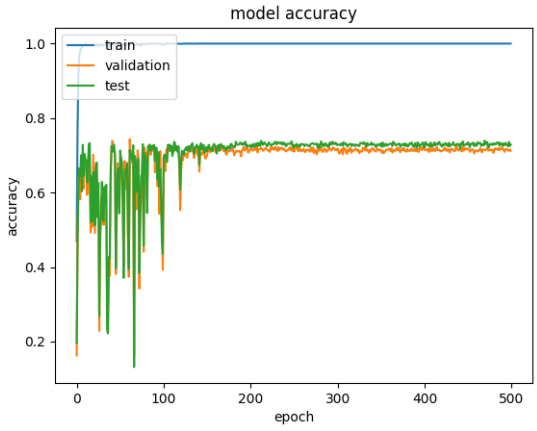


Figure 19: Accuracy Learning Curve Fold 1

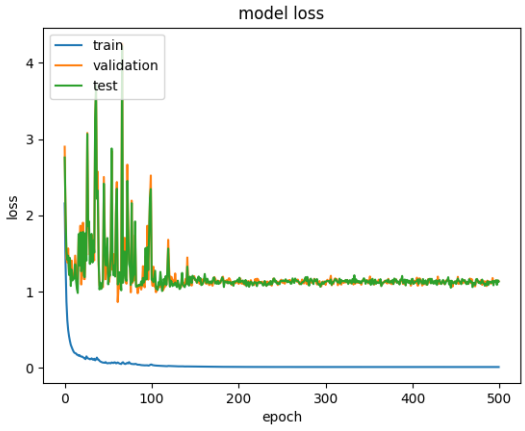


Figure 20: Loss Curve Fold 1

As the labels of the unseen test set of the challenge were made available we can even include the learning process with respect to the test set. As we can see in figure 19, the training accuracy converges to 100% as expected. The validation accuracy converges to

approximately 75%. Also the accuracy on the test set is slightly higher which implies that there is no overfitting of the training data. The file wise validation accuracy of fold 1 is slightly lower than the one stated in [6]. The model loss for fold 1 (Figure 20) also converges to its minimum. The learning curves for folds 2 to 4 can be seen in figure 21-26.

Although the file wise validation accuracies all differ from those stated in [6], the averaged accuracy over the four folds is higher than the one reached by the submitted model (83.3%). This might be due to differences in the training process in Keras, or another source of randomisation which might have been introduced by the GPU.

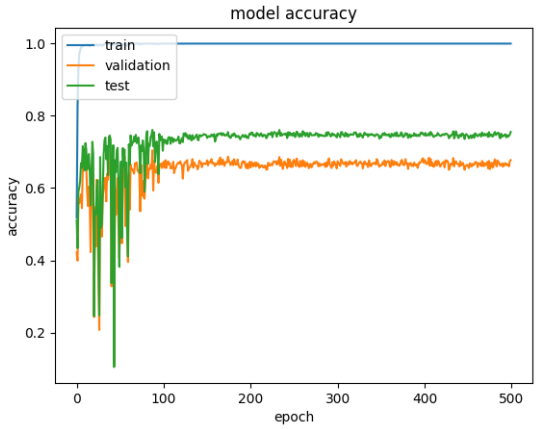


Figure 21: Accuracy Learning Curve Fold 2

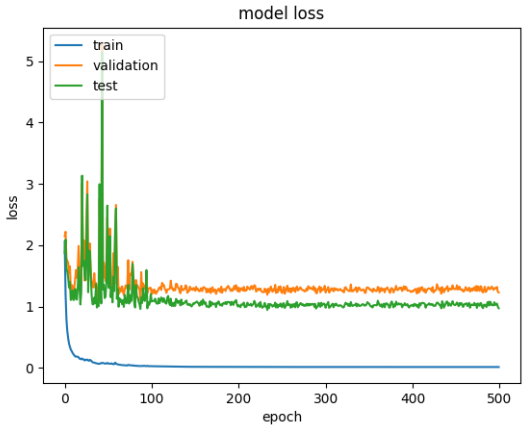


Figure 22: Loss Curve Fold 2

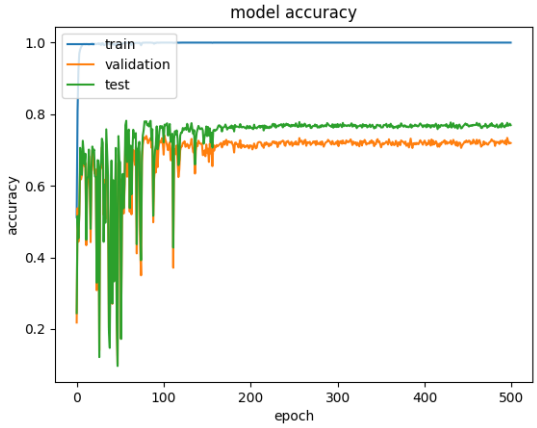


Figure 23: Accuracy Learning Curve Fold 3

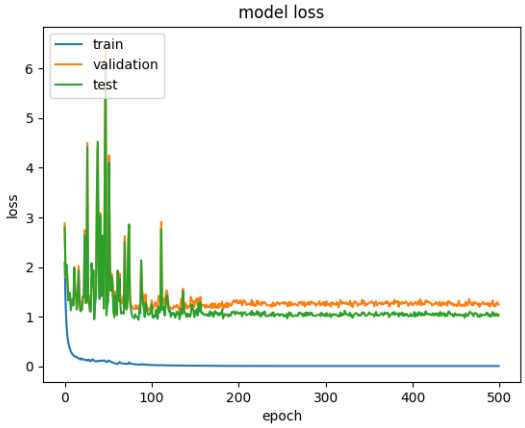


Figure 24: Loss Curve Fold 3

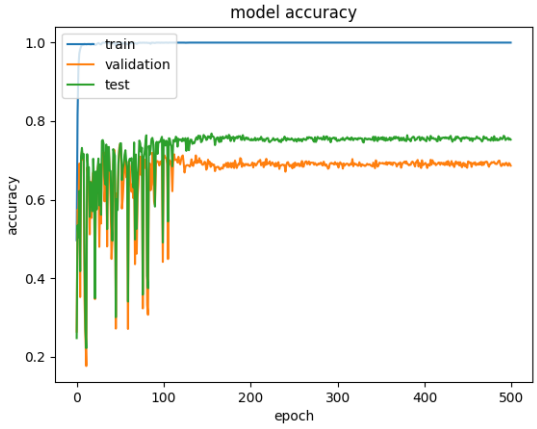


Figure 25: Accuracy Learning Curve Fold 4

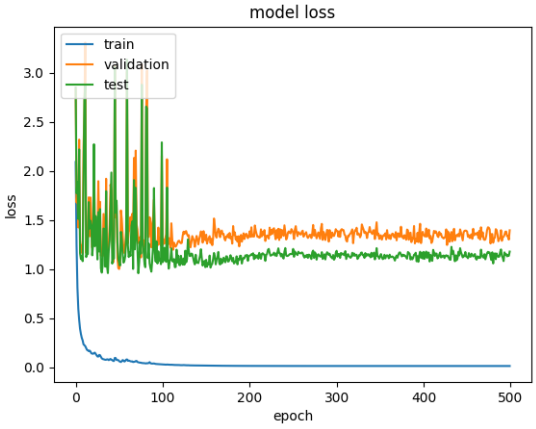


Figure 26: Loss Curve Fold 4

The learning curves for the other folds show that the models are generalizing very well. The accuracy on the test set is always higher than the validation accuracy. Generally the learning and loss curves for the other experiments are rather similar, therefore they are not depicted for further experiments.

6.1.1.2 24 Bin Spectrogram with AMSGRAD

As mentioned in section 4.2 a variant of the ADAM optimizer has been developed recently. In some experiments the use of this variant led to lower cross-entropy loss. To verify if that is the case on the DCASE data an experiment has been conducted utilizing the AMSGRAD optimizer. In Keras this can be easily implemented by setting the AMSGRAD option in the ADAM optimizer to true. The following results have been collected:

Fold	Frame Validation Accuracy	File Validation Accuracy	File Test Accuracy
1	0.7445	0.82	0.76
2	0.6799	0.71	0.80
3	0.7413	0.78	0.81
4	0.7413	0.80	0.81
Frame Wise Averaged Test accuracy: 0.8461 File Wise Averaged Test accuracy: 0.8487			

Table 2: Classification Accuracies using raw spectrograms as features and AMSGRAD

The file-wise test accuracies are slightly higher than in table 1. This implies that the model might be able to generalize better using the AMSGRAD optimizer. However the final classification accuracies are slightly lower than those when simply using ADAM as optimizer. Therefore the AMSGRAD variant of ADAM might be better for cases where ADAM does not converge, though it did not lead to a better generalization.

6.1.1.3 24 Bin Spectrogram with Random Normal Initialization

As the HeNormal initialization in Keras differs from the one in Lasagne, another experiment has been conducted using an initialization from a normal distribution with $\mu = 0.0$ and $\sigma = 0.02$. Table 3 shows the results for this experiment.

Fold	Frame Validation Accuracy	File Validation Accuracy	File Test Accuracy
1	0.7367	0.82	0.77
2	0.6759	0.71	0.80
3	0.7511	0.79	0.80
4	0.7472	0.81	0.76
Frame Wise Averaged Test accuracy: 0.8384 File Wise Averaged Test accuracy: 0.8487			

Table 3: Classification Accuracies using raw spectrograms as features and Random Normal Initialization

The accuracies slightly differ from the previous experiments, but there is no improvement in generalization. It is however noticeable that the file wise averaging of the models in this case leads to a higher classification accuracy. This was not the case in the previous experiments and shows that this method might also be a good choice in some cases. Also the HeNormal initialization seems to work better than the Random Normal initialization.

6.1.2 Constant Q Transform Spectrograms

In [8], different CQT spectrograms have been used and the submission performed better on the unseen test set than [7]. To find out on which spectrogram type the network performs better and if the CQT spectrograms lead to an improvement in generalization, experiments with 80 bin, 84 bin, and 149 bin CQTs have been conducted. For training models on 80, and 84 bins the input of the model was adjusted to (80x149), and (84x149), respectively. Also the CQT spectrograms are much larger than the raw spectrograms, meaning that there is more data to train on.

6.1.2.1 80 Bin CQT Spectrogram

Fold	Frame Validation Accuracy	File Validation Accuracy	File Test Accuracy
1	0.6426	0.76	0.65
2	0.5258	0.57	0.67
3	0.6287	0.67	0.66
4	0.5855	0.67	0.72
Frame Wise Averaged Test accuracy: 0.7512 File Wise Averaged Test accuracy: 0.7358			

Table 4: Classification Accuracies using CQT spectrograms and HeNormal Initialization

The frame - and file wise validation accuracy is much lower than the one compared to raw spectrograms. Also the performance on the unseen test set was rather bad. The file wise validation accuracy on fold 1 is much higher than the file wise test accuracy, implying overfitting on the training data. However the models of the remaining folds managed to generalize very well. The convergence to a minimum loss for CQT spectrograms takes place in the early epochs. This is due to the big amount of training data, therefore the number of epochs was reduced to 100 for experiments utilizing CQT spectrograms.

6.1.2.2 84 Bin CQT Spectrogram

Fold	Frame Validation Accuracy	File Validation Accuracy	File Test Accuracy
1	0.6332	0.74	0.70
2	0.5682	0.62	0.66
3	0.6519	0.75	0.73
4	0.5446	0.61	0.69
Frame Wise Averaged Test accuracy: 0.7743 File Wise Averaged Test accuracy: 0.7769			

Table 5: Classification Accuracies using CQT spectrograms and Random Normal Initialization

Using the 84 bin CQTs shows an improvement in the performance by almost 3 percentage points.

6.1.2.3 149 Bin CQT Spectrogram

Fold	Frame Validation Accuracy	File Validation Accuracy	File Test Accuracy
1	0.6628	0.77	0.72
2	0.6562	0.76	0.74
3	0.7068	0.78	0.75
4	0.5801	0.65	0.74
Frame Wise Averaged Test accuracy: 0.7743 File Wise Averaged Test accuracy: 0.7897			

Table 6: Classification Accuracies using CQT spectrograms and Random Normal Initialization

Again an improvement in the performance can be observed in table 6. Extracting spectrograms with more frequency bins leads to more information in the spectrogram. This might be an explanation for the improved classification accuracy.

Overall the raw spectrograms clearly outperformed the CQT spectrograms and there was no improvement in generalization observed. However as this work [8] was submitted to the challenge of 2017, this might be different for the dataset of the DCASE 2017 challenge.

6.1.3 Interpolation Algorithms

The next thing we are interested in, is how the interpolation algorithms (explained in section 3) influence the performance of the models. [8] stated that they resized the spectrograms to fit the input of the network, all of the interpolation algorithms mentioned in section 3 were applied to the CQT spectrograms to reshape them from (80x149), or (84x149) to (149x149).

6.1.3.1 Nearest Neighbor Interpolation

6.1.3.1.1 CQT 80 bins

Fold	Frame Validation Accuracy	File Validation Accuracy	File Test Accuracy
1	0.6550	0.73	0.77
2	0.5707	0.62	0.67
3	0.6358	0.72	0.68
4	0.6253	0.70	0.74
Frame Wise Averaged Test accuracy: 0.81025 File Wise Averaged Test accuracy: 0.8051			

Table 7: Classification Accuracies using upsampled CQT spectrograms and Random Normal Initialization

Comparing table 7 to table 4 (80 bin CQTs) we can already observe a difference in performance. Although the interpolation does not directly introduce new information into the spectrogram as it just repeats the nearest neighbors, it seems to perform better by even 6 percentage points.

6.1.3.1.2 CQT 84 bins

Fold	Frame Validation Accuracy	File Validation Accuracy	File Test Accuracy
1	0.6557	0.75	0.74
2	0.5960	0.65	0.75
3	0.6648	0.74	0.73
4	0.5956	0.70	0.70
Frame Wise Averaged Test accuracy: 0.7974 File Wise Averaged Test accuracy: 0.80			

Table 8: Classification Accuracies using upsampled CQT spectrograms and Random Normal Initialization

Also the performance of the 84 bin CQT improved (comparing table 8 and table 5) by approximately 3 percentage points.

6.1.3.2 Bilinear Interpolation

6.1.3.2.1 CQT 80 bins

Fold	Frame Validation Accuracy	File Validation Accuracy	File Test Accuracy
1	0.6625	0.77	0.72
2	0.5884	0.66	0.67
3	0.6195	0.69	0.70
4	0.6036	0.68	0.78
Frame Wise Averaged Test accuracy: 0.8051 File Wise Averaged Test accuracy: 0.782			

Table 9: Classification Accuracies using upsampled CQT spectrograms and Random Normal Initialization

The bilinear interpolation also improves the performance (compared to table 4) by around 5 percentage points.

6.1.3.2.2 CQT 84 bins

Fold	Frame Validation Accuracy	File Validation Accuracy	File Test Accuracy
1	0.6646	0.77	0.74
2	0.6425	0.73	0.74
3	0.6634	0.73	0.72
4	0.6335	0.72	0.74
Frame Wise Averaged Test accuracy: 0.8282 File Wise Averaged Test accuracy: 0.8358			

Table 10: Classification Accuracies using upsampled CQT spectrograms and Random Normal Initialization

Compared to table 5 the bilinear interpolation improved the performance of the 84 bin CQT by almost 7 percentage points.

6.1.3.3 Cubic Interpolation

6.1.3.3.1 CQT 80 bins

Fold	Frame Validation Accuracy	File Validation Accuracy	File Test Accuracy
1	0.6470	0.75	0.74
2	0.5948	0.68	0.66
3	0.6996	0.79	0.75
4	0.5969	0.68	0.73
Frame Wise Averaged Test accuracy: 0.7974 File Wise Averaged Test accuracy: 0.782			

Table 11: Classification Accuracies using upsampled CQT spectrograms and Random Normal Initialization

6.1.3.3.2 CQT 84 bins

Fold	Frame Validation Accuracy	File Validation Accuracy	File Test Accuracy
1	0.6743	0.76	0.76
2	0.6004	0.66	0.72
3	0.7066	0.80	0.76
4	0.6262	0.72	0.73
Frame Wise Averaged Test accuracy: 0.81025 File Wise Averaged Test accuracy: 0.81025			

Table 12: Classification Accuracies using upsampled CQT spectrograms and Random Normal Initialization

Also the cubic interpolation led to an improvement in generalization in both types of spectrograms.

6.1.3.4 Bicubic Interpolation

6.1.3.4.1 CQT 80 bins

Fold	Frame Validation Accuracy	File Validation Accuracy	File Test Accuracy
1	0.6512	0.74	0.74
2	0.5877	0.66	0.70
3	0.6940	0.79	0.76
4	0.6079	0.70	0.75
Frame Wise Averaged Test accuracy: 0.8128 File Wise Averaged Test accuracy: 0.81025			

Table 13: Classification Accuracies using upsampled CQT spectrograms and Random Normal Initialization

6.1.3.4.2 CQT 84 bins

Fold	Frame Validation Accuracy	File Validation Accuracy	File Test Accuracy
1	0.6447	0.78	0.64
2	0.6278	0.69	0.71
3	0.6794	0.76	0.76
4	0.6673	0.76	0.76
Frame Wise Averaged Test accuracy: 0.82051 File Wise Averaged Test accuracy: 0.81025			

Table 14: Classification Accuracies using upsampled CQT spectrograms and Random Normal Initialization

Bicubic interpolation made the models perform better than the cubic interpolation. This might be due to the fact that the cubic interpolation is originally designed for 1D sequences and not for 2D images.

6.1.3.5 Lanczos Interpolation

6.1.3.5.1 CQT 80 bins

Fold	Frame Validation Accuracy	File Validation Accuracy	File Test Accuracy
1	0.6232	0.69	0.73
2	0.6449	0.73	0.74
3	0.7002	0.76	0.76
4	0.5971	0.68	0.69
Frame Wise Averaged Test accuracy: 0.7846 File Wise Averaged Test accuracy: 0.7897			

Table 15: Classification Accuracies using upsampled CQT spectrograms and Random Normal Initialization

6.1.3.5.2 CQT 84 bins

Fold	Frame Validation Accuracy	File Validation Accuracy	File Test Accuracy
1	0.6298	0.72	0.65
2	0.5825	0.65	0.73
3	0.6681	0.74	0.72
4	0.6586	0.77	0.78
Frame Wise Averaged Test accuracy: 0.8153 File Wise Averaged Test accuracy: 0.7923			

Table 16: Classification Accuracies using upsampled CQT spectrograms and Random Normal Initialization

Remarkably all of the interpolated CQT spectrograms led to an improvement in performance and a better generalization. Also almost all of them performed better than the 149 bin CQT spectrogram (table 6), which proves that the size is not the decisive factor, but rather the application of the interpolation algorithm to the spectrogram. A possible reason for that might be a priori estimation via the interpolation.

6.1.3.6 Interpolated Spectrograms

The bilinear interpolation showed the best improvement, therefore this interpolation algorithm was also applied to the raw spectrograms, interpolating the frequency dimension from (149x149) to (175x149), and from (149x149) to (300x149). Again to prove that the size of the spectrogram is not the decisive factor for the improvement, raw spectrograms with dimension (298x149) using 60 frequency bins have been extracted and used as features. As we apply the sliding window over the time domain the input dimension of the model for the frequency has to be adjusted to the interpolated frequency dimension.

6.1.3.6.1 Spectrograms Stretched to 175

Fold	Frame Validation Accuracy	File Validation Accuracy	File Test Accuracy
1	0.7483	0.81	0.78
2	0.6853	0.72	0.80
3	0.7578	0.82	0.83
4	0.7774	0.85	0.82
Frame Wise Averaged Test accuracy: 0.8589 File Wise Averaged Test accuracy: 0.8589			

Table 17: Classification Accuracies using upsampled spectrograms and Random Normal Initialization

Again the results show an improvement in generalization compared to table 1.

6.1.3.6.2 Spectrograms Stretched to 300

Fold	Frame Validation Accuracy	File Validation Accuracy	File Test Accuracy
1	0.7567	0.80	0.80
2	0.7291	0.78	0.79
3	0.7666	0.82	0.82
4	0.7689	0.83	0.80
Frame Wise Averaged Test accuracy: 0.8461 File Wise Averaged Test accuracy: 0.8384			

Table 18: Classification Accuracies using upsampled CQT spectrograms and Random Normal Initialization

Although the stretched spectrogram to 175 frequency dimension showed a slight improvement, the models trained on spectrograms stretched to 300 frequency dimension actually performed even worse than the original extracted spectrograms.

6.1.3.6.3 Spectrograms using 60 Frequency Bins

Table 19 shows the results for the spectrograms which were extracted using a higher number of frequency bins. Extracting spectrograms using 60 frequency bins results in a frequency dimension of 298, which is comparable to the 300 frequency dimension of interpolated spectrograms. Remarkably the spectrograms stretched to the same dimension managed to generalize better. Therefore it might be that the interpolation works as noise removal or prior estimation.

Fold	Frame Validation Accuracy	File Validation Accuracy	File Test Accuracy
1	0.771	0.85	0.78
2	0.713	0.76	0.82
3	0.747	0.79	0.77
4	0.795	0.85	0.81
Frame Wise Averaged Test accuracy: 0.8359 File Wise Averaged Test accuracy: 0.8333			

Table 19: Classification Accuracies using upsampled CQT spectrograms and Random Normal Initialization

6.1.4 Network Architecture Adjustments

The model architecture depicted in figure 17 was originally designed for an input of (149x149). However using standard CQT spectrograms we needed to change the input dimensions. Therefore it might be the case that the kernel sizes and the striding do not fit the data anymore. Thus additional experiments in terms of architecture changes have been performed.

Fitting the input dimension of the model to the dimensions of the 84 bin CQT spectrograms requires a change from (149x149) to (84x149). Thus an input kernel of size (5x5) does not fit the dimension of the spectrogram anymore. However changing the input kernel to (4x4) or (3x3) would fit the frequency dimension of the CQT and might improve the overall performance of the model. Table 20 shows the results for an adjusted input kernel to (4x4) and table 21 for a (3x3) input kernel.

Fold	Frame Validation Accuracy	File Validation Accuracy	File Test Accuracy
1	0.6160	0.73	0.68
2	0.5692	0.61	0.68
3	0.6687	0.76	0.71
4	0.5866	0.68	0.72
Frame Wise Averaged Test accuracy: 0.7846 File Wise Averaged Test accuracy: 0.7641			

Table 20: Classification Accuracies using CQT 84 bin spectrograms, Random Normal Initialization, and an adjusted input kernel

Fold	Frame Validation Accuracy	File Validation Accuracy	File Test Accuracy
1	0.6393	0.73	0.69
2	0.5332	0.58	0.68
3	0.6616	0.77	0.71
4	0.5999	0.67	0.70
Frame Wise Averaged Test accuracy: 0.7667 File Wise Averaged Test accuracy: 0.7564			

Table 21: Classification Accuracies using CQT 84 bin spectrograms and Random Normal Initialization

Comparing the two tables above with table 5 we can see that reducing the input kernel from (5x5) to (4x4) results in an improvement, although if the input kernel gets smaller (3x3) the model performs worse than with a (5x5) input kernel. Also as the architecture uses padding to the original size the deeper kernels still fit the activations of the previous layers. Further, actually just the frequency dimension of the input dimension is changed and the time dimension stayed the same, therefore it makes sense to only adjust the frequency dimension of the input dimension of the model. Table 22 shows the results for a model with an input kernel of (3x5) and all other deeper kernels of size (2x3). We can observe that there is a slight improvement compared to when only adjusting the input kernel.

Fold	Frame Validation Accuracy	File Validation Accuracy	File Test Accuracy
1	0.63	0.75	0.70
2	0.5915	0.64	0.71
3	0.6967	0.77	0.74
4	0.6370	0.76	0.76
Frame Wise Averaged Test accuracy: 0.7794 File Wise Averaged Test accuracy: 0.7871			

Table 22: Classification Accuracies using CQT 84 bin spectrograms, Random Normal Initialization, and adjusted kernels

6.1.5 Additional Experiments

As the mentioned in section 5.1.1.3 the HeNormal initialization might lead to improvements in performance. Therefore the fundamental and best performing models have been selected and tested for improvement with the changed initialization.

The performances of the 149 bin CQT, bilinearly stretched 84 bin CQT, and 84 bin CQT with adjusted network structure using HeNormal initialization is shown in table 23, 24, and 25, respectively.

Fold	Frame Validation Accuracy	File Validation Accuracy	File Test Accuracy
1	0.6878	0.81	0.79
2	0.6679	0.70	0.71
3	0.7132	0.78	0.74
4	0.6297	0.72	0.80
Frame Wise Averaged Test accuracy: 0.8076 File Wise Averaged Test accuracy: 0.81025			

Table 23: Classification Accuracies using 149 bin CQT spectrograms and HeNormal Initialization

Fold	Frame Validation Accuracy	File Validation Accuracy	File Test Accuracy
1	0.6972	0.80	0.77
2	0.6485	0.70	0.76
3	0.7035	0.75	0.76
4	0.6516	0.77	0.77
Frame Wise Averaged Test accuracy: 0.8153 File Wise Averaged Test accuracy: 0.8230			

Table 24: Classification Accuracies using bilinearly stretched CQT spectrograms and HeNormal Initialization

Fold	Frame Validation Accuracy	File Validation Accuracy	File Test Accuracy
1	0.6845	0.77	0.74
2	0.5885	0.64	0.69
3	0.677	0.77	0.77
4	0.6384	0.71	0.71
Frame Wise Averaged Test accuracy: 0.7871 File Wise Averaged Test accuracy: 0.7974			

Table 25: Classification Accuracies using 84 bin CQT spectrograms, HeNormal Initialization, and adjusted architecture

Comparing the three tables to the previous experiments using random normal initialization we can see that in case of the standard extracted spectrograms the performance did improve. However the classification accuracy decreased for the bilinearly stretched spectrogram. The reason for this might be that the random normal initializer better approximates the real distribution of the interpolated spectrograms. To further investigate this assumption the HeNormal initializer was tested on the bilinearly stretched spectrograms to 175 in frequency dimension (Table 26).

Fold	Frame Validation Accuracy	File Validation Accuracy	File Test Accuracy
1	0.7297	0.79	0.77
2	0.6855	0.68	0.78
3	0.7461	0.76	0.81
4	0.7618	0.82	0.80
Frame Wise Averaged Test accuracy: 0.8589 File Wise Averaged Test accuracy: 0.8461			

Table 26: Classification Accuracies using bilinearly stretched spectrograms and HeNormal Initialization

The above table shows that for raw bilinearly stretched spectrograms no improvement is seen, but rather equals the accuracy when using random normal initializtaion. However because of the fact that network architecture changes improved generalization in previous experiments, more additional experiments with different network architectures on CQT spectrograms as well as on raw spectrograms have been carried out.

It is noticeable that if the raw spectrograms are stretched to 175 frequency dimension an improvement is observed, but this does not hold for spectrograms stretched to 300 frequency dimension. This implies that the network architecture again may not fit the input, therefore a change was made in the kernel sizes and the model was tested.

Fold	Frame Validation Accuracy	File Validation Accuracy	File Test Accuracy
1	0.7646	0.82	0.80
2	0.7103	0.74	0.79
3	0.7273	0.76	0.81
4	0.783	0.86	0.80
Frame Wise Averaged Test accuracy: 0.8487 File Wise Averaged Test accuracy: 0.8410			

Table 27: Classification Accuracies using bilinearly stretched spectrograms, HeNormal Initialization, and adjusted kernel

Table 27 shows the results for increasing the size of the input kernel to (6x5) and increasing all deeper kernels to (4x3). A slight improvement of classification accuracy can be observed. Changing the kernel sizes however does not change the activation sizes as the padding mode *same* is used and thus the input size is preserved by zero-padding. Therefore by changing the striding we would not need to explicitly change the deeper kernels as the activations sizes would be back to the sizes they were designed for. The following table shows the performance of a model trained on the spectrograms bilinearly stretched to 300 in frequency domain with adjusted striding of (4x2), meaning the striding is doubled in the frequency axis.

Fold	Frame Validation Accuracy	File Validation Accuracy	File Test Accuracy
1	0.7636	0.81	0.77
2	0.6674	0.69	0.80
3	0.745	0.81	0.74
4	0.7677	0.80	0.82
Frame Wise Averaged Test accuracy: 0.8410 File Wise Averaged Test accuracy: 0.8359			

Table 28: Classification Accuracies using bilinearly stretched spectrograms, HeNormal Initialization, and adjusted striding

Compared to table 26 the performance did not improve. As the input kernel of the network still has to slide over a bigger size than it was designed for, keeping striding and additionally adjusting the input kernel to fit the input shape of the spectrogram might lead to improvements.

Fold	Frame Validation Accuracy	File Validation Accuracy	File Test Accuracy
1	0.7432	0.79	0.75
2	0.7319	0.77	0.79
3	0.7318	0.78	0.80
4	0.7562	0.82	0.81
Frame Wise Averaged Test accuracy: 0.8256 File Wise Averaged Test accuracy: 0.8307			

Table 29: Classification Accuracies using bilinearly stretched spectrograms, HeNormal Initialization, adjusted striding, and adjusted input kernel of (8x5)

Looking at the table above we can see that the classification accuracy dropped even below those of table 28, where only striding was used. Therefore using a bigger striding does not positively affect the results for spectrograms of size (300x149). However this might not be the case for lowering the striding. The following table shows the performance of a model trained on 84 bin CQT spectrograms using adjusted striding and adjusted kernels.

Fold	Frame Validation Accuracy	File Validation Accuracy	File Test Accuracy
1	0.6953	0.74	0.76
2	0.6126	0.69	0.74
3	0.7194	0.78	0.77
4	0.6651	0.74	0.78
Frame Wise Averaged Test accuracy: 0.8282 File Wise Averaged Test accuracy: 0.8333			

Table 30: Classification Accuracies using 84 bin CQT spectrograms, HeNormal Initialization, and adjusted kernels, and striding

On the CQT spectrograms the adjusted striding improved the performance of the best 84 bin CWQT model so far (table 25). As the spectrograms bilenearly stretched to 175 dimension in frequency is the overall best performing model it might be possible to further improve this model by also applying architecture changes. The following two tables show experiments using once adjusted kernels with the HeNormal initializer (table 30) and once adjusted kernels with the random normal initializer (table 31).

Fold	Frame Validation Accuracy	File Validation Accuracy	File Test Accuracy
1	0.7661	0.82	0.78
2	0.7043	0.74	0.80
3	0.7309	0.76	0.80
4	0.7683	0.82	0.78
Frame Wise Averaged Test accuracy: 0.8307 File Wise Averaged Test accuracy: 0.8282			

Table 31: Classification Accuracies using bilinearly stretched spectrograms, HeNormal Initialization, and adjusted kernels

Fold	Frame Validation Accuracy	File Validation Accuracy	File Test Accuracy
1	0.7461	0.79	0.78
2	0.6962	0.74	0.81
3	0.719	0.77	0.82
4	0.7677	0.81	0.81
Frame Wise Averaged Test accuracy: 0.8410 File Wise Averaged Test accuracy: 0.8487			

Table 32: Classification Accuracies using bilinearly stretched spectrograms, HeNormal Initialization, and adjusted kernels

Looking at the above tables we can see that the random normal initializer works better for stretched spectrograms and also adjusting the kernel sizes does not positively influence the results.

6.1.6 Overview

After conducting all of the above experiments we can summarize the results and pick the best performing model. This can be done by looking at the mean and variance of the predictions on the unseen test set of the models. The best model is the one with the least variance and the highest mean. Figure 27 shows the means and variances over the 4 folds of the accuracies of the test set of the models trained on the CQT spectrgorams. Remarkably, the bilinearly stretched spectrograms show the least variance and high means. The worst performing model is the one trained on 80 bin CQT spectrograms.

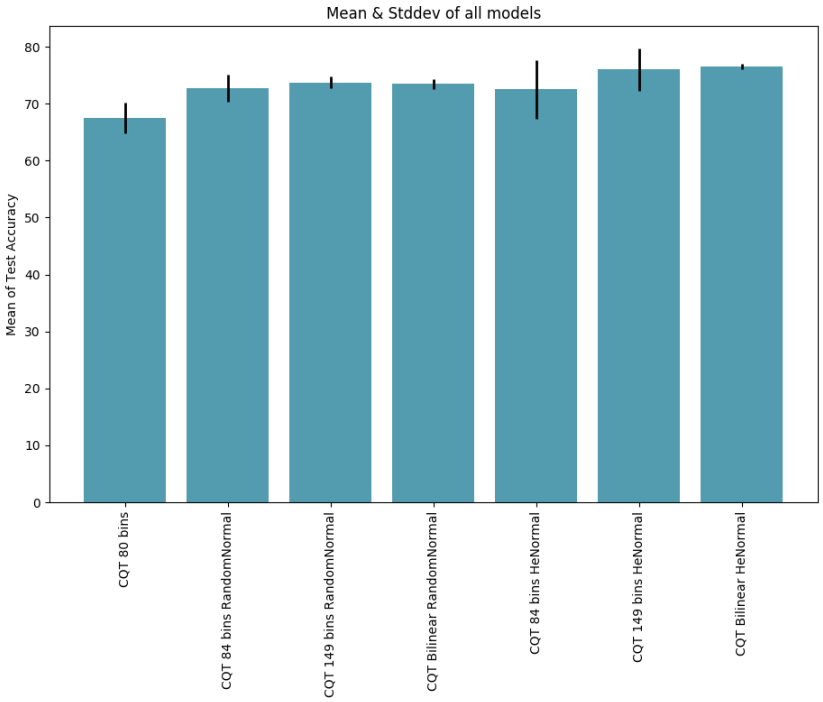


Figure 27: Error Plot showing the mean of the classification accuracies of the test set as bars and the standard deviation as error. The shown models are trained on CQT spectrograms.

Figure 28 depicts the means and variances of the models trained on raw spectrograms. The models trained on raw spectrograms are all performing almost equally well. The best

model however is trained on spectrograms bilinearly stretched to 175 in frequency dimension. Also noticeable is that the random normal initialization improved the performance in this model whereas in the models trained on the CQT spectrograms the HeNormal initialization led to improvements. Figure 29 shows the models trained on CQT spectrograms as well as those trained on raw spectrograms. Overall the best performance on the unseen test set was achieved by the bilinearly stretched spectrograms.

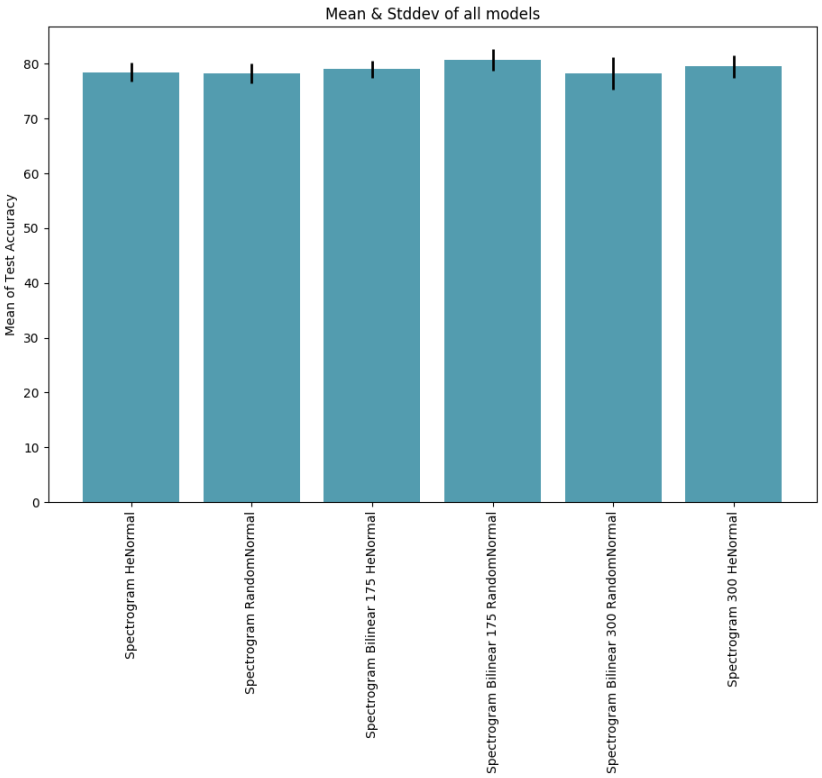


Figure 28: Error Plot showing the models which are trained on raw spectrograms.

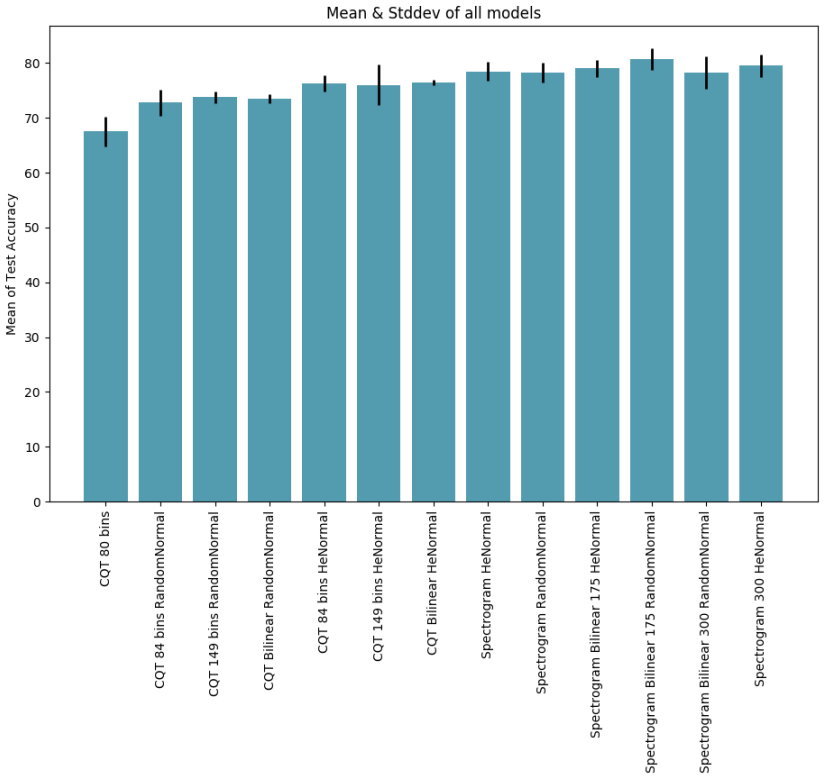


Figure 29: Error Plot of all models.

However in reality we obviously do not have the measure on the unseen test set. Therefore we have to consider the measures on the validation set. The next figure shows the performance of the best models on the validation set. Again the models trained on raw spectrograms outperform the models trained on CQT spectrograms. Also the best model on the unseen test set is one of the best performing models on the validation set. Table 33 shows the averaged final performance of all the models.

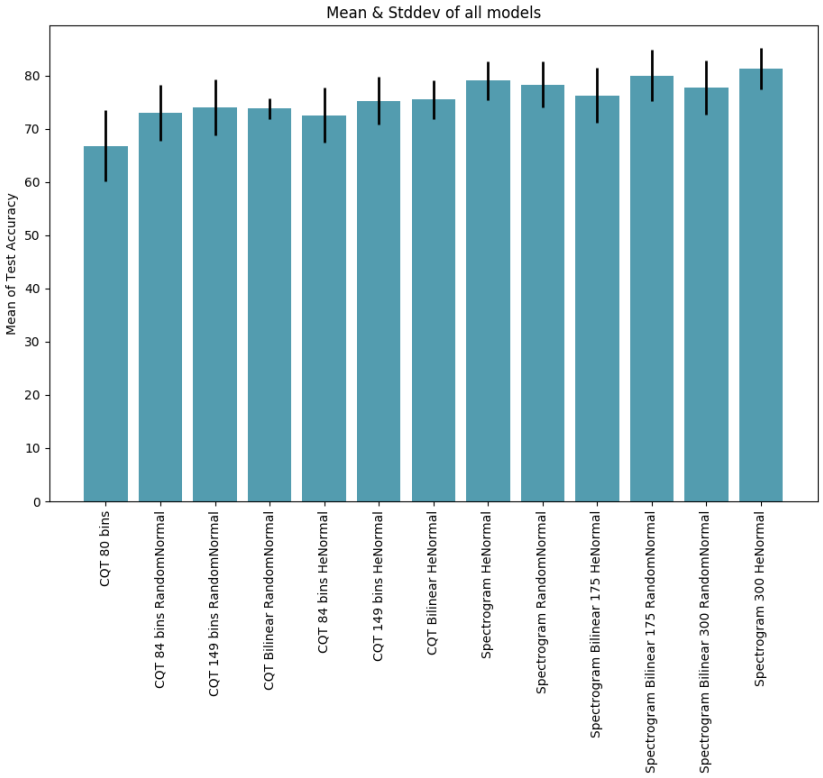


Figure 30: Error Plot showing the performance of all models on the validation set.

Spectrogram	Bins	Interpolation	Initializer	Validation Accuracy	Test Accuracy
CQT	80	None	RandomNormal	70%	80.51%
CQT	84	None	RandomNormal	73%	78.71%
CQT	149	None	RandomNormal	74%	78.97%
CQT	84	Bilinear	RandomNormal	73.75%	83.58%
CQT	84	None	HeNormal	72.5%	83.33%
CQT	149	None	HeNormal	75.25%	81.025%
CQT	84	Bilinear	HeNormal	75.5%	82.3%
STFT	24	None	RandomNormal	78.25%	83.84%
STFT	24	None	HeNormal	79%	85.38%
STFT	24	Bilinear	RandomNormal	80%	85.89%
STFT	24	Bilinear	HeNormal	76.25%	85.89%
STFT	24	Bilinear	HeNormal	79.5%	84.87%
STFT	60	None	HeNormal	81.25%	83.59%

Table 33: Overall Accuracies

In the above table we can see that the model with the highest accuracy on the validation set did not yield the best performance on the unseen test set. However it also shows that applying interpolation algorithms to spectrograms leads to better generalization.

6.2 DCASE 2017

In the DCASE 2017 challenge the training and the test set originated from different distributions. Therefore the need for generalization was much stronger. The best performing

models on the DCASE 2016 challenge were also tested on the DCASE 2017 challenge. As a measure of comparison the initial implementation [7] consisting of the late fusion of multiple DCNNs trained on raw spectrograms reached a file-wise accuracy of 64.8% on the unseen test set.

6.2.1 Constant Q Transform Spectrograms

The best performing CQT model on the DCASE 2016 challenge was the 84 bin CQT with adjusted network structure. The following table shows the performance of this model trained on the training set of the DCASE 2017.

Fold	Frame Validation Accuracy	File Validation Accuracy	File Test Accuracy
1	0.75	0.79	0.58
2	0.7341	0.78	0.61
3	0.7626	0.81	0.61
4	0.7595	0.83	0.61
Frame Wise Averaged Test accuracy: 0.658 File Wise Averaged Test accuracy: 0.6475			

Table 34: Classification Accuracies using 84 bin CQT spectrograms, HeNormal Initialization, and adjusted kernels

The above table shows that the model performs well on the validation set, but the accuracy drastically dropped on the test set. This implies that the model overfits the training data and fails to generalize. The next tables show the performance of the 149 bin CQT model and the bilinearly stretched CQT model.

Fold	Frame Validation Accuracy	File Validation Accuracy	File Test Accuracy
1	0.7361	0.78	0.59
2	0.7367	0.78	0.59
3	0.7484	0.79	0.62
4	0.7474	0.80	0.63
Frame Wise Averaged Test accuracy: 0.6321 File Wise Averaged Test accuracy: 0.6382			

Table 35: Classification Accuracies using 149 bin spectrograms, HeNormal Initialization

Fold	Frame Validation Accuracy	File Validation Accuracy	File Test Accuracy
1	0.7491	0.80	0.60
2	0.7380	0.76	0.60
3	0.7389	0.79	0.58
4	0.7572	0.83	0.59
Frame Wise Averaged Test accuracy: 0.6105 File Wise Averaged Test accuracy: 0.6253			

Table 36: Classification Accuracies using bilinearly stretched CQT spectrograms, HeNormal Initialization

The same overfitting pattern appears for the other models trained on CQT spectrograms.

6.2.2 Spectrograms

Table 37 and 38 show the performances of models trained on raw spectrograms and bilinearly stretched spectrograms.

Fold	Frame Validation Accuracy	File Validation Accuracy	File Test Accuracy
1	0.8045	0.80	0.61
2	0.8064	0.81	0.65
3	0.8118	0.81	0.67
4	0.841	0.85	0.68
Frame Wise Averaged Test accuracy: 0.6895 File Wise Averaged Test accuracy: 0.6882			

Table 37: Classification Accuracies using spectrograms, HeNormal Initialization

Fold	Frame Validation Accuracy	File Validation Accuracy	File Test Accuracy
1	0.8174	0.83	0.63
2	0.8121	0.82	0.63
3	0.8028	0.81	0.63
4	0.836	0.86	0.63
Frame Wise Averaged Test accuracy: 0.6654 File Wise Averaged Test accuracy: 0.67037			

Table 38: Classification Accuracies using bilinearly stretched spectrograms, RandomNormal Initialization

The models trained on raw spectrograms showed an improved performance compared to those trained on CQT spectrograms, although they are still overfitting heavily.

6.2.3 No Sliding Window

As the audio excerpts of the DCASE 2017 datasets are only 10 seconds long the sliding window over the spectrogram produces only 3 windows. Therefore omitting the sliding window and feeding the whole spectrograms into the network might help the model to generalize better. This was only done for raw spectrograms as the CQT spectrograms are much larger in the time domain.

Fold	Frame Validation Accuracy	File Validation Accuracy	File Test Accuracy
1	0.826	0.81	0.62
2	0.829	0.81	0.66
3	0.813	0.79	0.64
4	0.891	0.89	0.68
Frame Wise Averaged Test accuracy: 0.6846 File Wise Averaged Test accuracy: 0.6784			

Table 39: Classification Accuracies using raw spectrograms without sliding window and HeNormal Initialization

Using the whole spectrograms without the sliding window approach showed a slight difference in the performance.

Fold	Frame Validation Accuracy	File Validation Accuracy	File Test Accuracy
1	0.8427	0.83	0.63
2	0.825	0.81	0.66
3	0.827	0.82	0.65
4	0.878	0.87	0.67
Frame Wise Averaged Test accuracy: 0.6938 File Wise Averaged Test accuracy: 0.6827			

Table 40: Classification Accuracies using bilinearly stretched spectrograms without sliding window, and Random Normal Initialization

Models trained on bilinearly stretched spectrograms to 175 in the frequency dimension again performed slightly better. Table 41 shows the results of spectrograms stretched to 300 in the frequency dimension.

Fold	Frame Validation Accuracy	File Validation Accuracy	File Test Accuracy
1	0.856	0.85	0.67
2	0.86	0.85	0.66
3	0.8414	0.82	0.70
4	0.8863	0.88	0.70
Frame Wise Averaged Test accuracy: 0.7148 File Wise Averaged Test accuracy: 0.7092			

Table 41: Classification Accuracies using bilinearly stretched spectrograms without sliding window, and HeNormal Initialization

The stretching to 300 in the frequency domain led to even better results. To find out whether it is the interpolation that improved the results or if it was only the bigger frequency domain, another experiment using raw spectrograms with 60 frequency bins was conducted (Table 42). The 60 frequency bins led to a dimension of 298 in the frequency domain.

Fold	Frame Validation Accuracy	File Validation Accuracy	File Test Accuracy
1	0.8521	0.84	0.69
2	0.8619	0.86	0.65
3	0.852	0.84	0.66
4	0.8692	0.86	0.69
Frame Wise Averaged Test accuracy: 0.7154 File Wise Averaged Test accuracy: 0.70061			

Table 42: Classification Accuracies using bilinearly stretched spectrograms without sliding window, and Random Normal Initialization

The bigger spectrograms led to almost the same results as the bilinearly stretched spectrograms to equal size. Thus, bigger spectrograms help the model to generalize.

6.2.4 Additional Experiments

As stated in [8] they replaced the Global Average Pooling Layer with a Flatten Layer. Table 43 shows the results of using a flatten layer instead of the global average pooling layer and shows that this did not lead to any improvement.

Fold	Frame Validation Accuracy	File Validation Accuracy	File Test Accuracy
1	0.7867	0.80	0.57
2	0.7732	0.72	0.58
3	0.76	0.76	0.61
4	0.8075	0.81	0.59
Frame Wise Averaged Test accuracy: 0.6352 File Wise Averaged Test accuracy: 0.64136			

Table 43: Classification Accuracies using bilinearly stretched spectrograms, Random Normal Initialization, and a Flatten layer

When using bigger spectrograms it might be the case that the network architecture is inappropriate for that size. The best performing model so far was trained on interpolated spectrograms. By increasing the striding of the first layer in the frequency domain the original size of the activations is preserved for deeper layers. However this did not yield improvements (see table 44).

Fold	Frame Validation Accuracy	File Validation Accuracy	File Test Accuracy
1	0.8145	0.80	0.60
2	0.8065	0.80	0.64
3	0.8065	0.80	0.61
4	0.857	0.85	0.64
Frame Wise Averaged Test accuracy: 0.6784 File Wise Averaged Test accuracy: 0.6697			

Table 44: Classification Accuracies using bilinearly stretched spectrograms, Random Normal Initialization, and adjusted striding (4,2)

Another option to preserve the activation sizes in the deeper layers is to adjust the max pooling filter. By only slightly increasing the striding and the max pooling filter in the frequency domain the kernels of the deeper layers fit the input activations again. However this constellation did not lead to any improvements (table 45).

Fold	Frame Validation Accuracy	File Validation Accuracy	File Test Accuracy
1	0.8128	0.80	0.63
2	0.7792	0.77	0.65
3	0.797	0.78	0.61
4	0.846	0.84	0.66
Frame Wise Averaged Test accuracy: 0.6877 File Wise Averaged Test accuracy: 0.666			

Table 45: Classification Accuracies using bilinearly stretched spectrograms, Random Normal Initialization, adjusted striding (3, 2), and adjusted max pooling (3,2)

The max pooling layer usually results in the network focusing on significant details on an image. An average pooling layer does not extract the highest pixel of an activation but rather averages over the pixels. It could be the case that this helps the network to improve the ability to generalize. However table 46 denies that assumption.

Fold	Frame Validation Accuracy	File Validation Accuracy	File Test Accuracy
1	0.8328	0.83	0.62
2	0.8398	0.84	0.62
3	0.8155	0.82	0.64
4	0.856	0.85	0.64
Frame Wise Averaged Test accuracy: 0.6777 File Wise Averaged Test accuracy: 0.6598			

Table 46: Classification Accuracies using bilinearly stretched spectrograms, Random Normal Initialization, and average pooling

Omitting the sliding window approach leads to less training data as the whole spectrogram is used instead of cutting it into smaller patches. Therefore when less training data is available decreasing the batch size would make sense. The next 4 tables show the classification accuracies of models trained on differently sized spectrograms with a batch size of 50 instead of the initially used 100.

Fold	Frame Validation Accuracy	File Validation Accuracy	File Test Accuracy
1	0.827	0.82	0.63
2	0.851	0.84	0.65
3	0.819	0.81	0.64
4	0.873	0.87	0.65
Frame Wise Averaged Test accuracy: 0.70 File Wise Averaged Test accuracy: 0.6827			

Table 47: Classification Accuracies using standard spectrograms, HeNormal Initialization, and a batch size of 50

Fold	Frame Validation Accuracy	File Validation Accuracy	File Test Accuracy
1	0.8377	0.83	0.62
2	0.844	0.84	0.65
3	0.84	0.82	0.67
4	0.847	0.84	0.66
Frame Wise Averaged Test accuracy: 0.6987 File Wise Averaged Test accuracy: 0.6944			

Table 48: Classification Accuracies using bilinearly stretched spectrograms to 175, Random Normal Initialization, and a batch size of 50

Fold	Frame Validation Accuracy	File Validation Accuracy	File Test Accuracy
1	0.864	0.85	0.66
2	0.861	0.85	0.65
3	0.8329	0.82	0.70
4	0.8821	0.88	0.69
Frame Wise Averaged Test accuracy: 0.7185 File Wise Averaged Test accuracy: 0.7123			

Table 49: Classification Accuracies using bilinearly stretched spectrograms to 300, Random Normal Initialization, and a batch size of 50

Fold	Frame Validation Accuracy	File Validation Accuracy	File Test Accuracy
1	0.813	0.82	0.66
2	0.832	0.83	0.65
3	0.8396	0.84	0.69
4	0.8621	0.86	0.72
Frame Wise Averaged Test accuracy: 0.7216 File Wise Averaged Test accuracy: 0.7111			

Table 50: Classification Accuracies using standard spectrograms with 60 frequency bins, HeNormal Initialization, and a batch size of 50

Comparing these constellations with the previous ones where a batch size of 100 was used, all of the models trained on a smaller batch size generalized better. This implies that also the batch size contributes to the ability of generalization of the model. [35] confirms this assumption.

In [8] it was mentioned that for the raw spectrograms they extracted big spectrograms, but then resized them to fit them into the network. This also might be a data preprocessing step leading to a better generalization. Table 51, and 52 show the performance on models trained on spectrograms extracted with 60 frequency bins and then resized to 149, and spectrograms extracted with 60 frequency bins and resampled to half of the sample rate, respectively. Both of the experiments did not imply that this preprocessing has any positive effect on the performance of the model.

Fold	Frame Validation Accuracy	File Validation Accuracy	File Test Accuracy
1	0.8333	0.82	0.65
2	0.825	0.81	0.66
3	0.829	0.80	0.61
4	0.868	0.85	0.65
Frame Wise Averaged Test accuracy: 0.6815 File Wise Averaged Test accuracy: 0.67098			

Table 51: Classification Accuracies using spectrograms resized to 149, HeNormal Initialization, and a batch size of 50

Fold	Frame Validation Accuracy	File Validation Accuracy	File Test Accuracy
1	0.826	0.81	0.63
2	0.847	0.82	0.63
3	0.841	0.83	0.64
4	0.87	0.86	0.65
Frame Wise Averaged Test accuracy: 0.6747 File Wise Averaged Test accuracy: 0.66049			

Table 52: Classification Accuracies using spectrograms resampled to 149, HeNormal Initialization, and a batch size of 50

6.2.5 Late Fusion

The classification accuracy reached by [8] was 74.8% on the unseen test set. The model yielding this result consisted of a late fusion of two models, one of them trained on raw

spectrograms and the second one trained on CQT 84 bin spectrograms. To investigate if the predictions of the model trained on the CQTs influenced the final prediction to generalize better, an experiment was conducted fusing a model trained on bilinearly stretched spectrograms, and a model trained on CQT 84 bin spectrograms (Table 53). The results of this experiment suggest that the model trained on the CQT spectrograms did not significantly improve the generalization. In fact simply extracting raw spectrograms with more frequency bins yields higher accuracy (71.54%). The late fusions conducted in this work used the Stochastic Gradient Descent optimizer with an initial learning rate of 0.2 which is halfed every 50 epochs. The fusion layer is initialized with Random Normal initialization. During training a batch size of 50 is used and the model is trained for 500 epochs.

We always fuse predictions of models for the same fold together, thus in the end we get a fusion model for each fold. Again two methods for getting the final prediction over all 4 folds were compared: *Averaging Accuracy* and *Majority Voting*. The predictions of the fusion models are vectors containing probabilities for each class. Therefore the first method is to simply average over the predictions of the 4 folds of the fusion models. The second method is to conduct majority voting.

Fold	File Validation Accuracy	File Test Accuracy
1	0.894	0.6611
2	0.8883	0.6741
3	0.8986	0.6605
4	0.9154	0.6815
Averaged Accuracy over 4 folds of test set: 0.7105 Majority Voting over 4 folds of test set: 0.70		

Table 53: Late Fusion Accuracies of two models trained on bilinearly stretched spectrograms and 84 bin CQT spectrograms

As the models trained on the raw spectrograms in general performed better in all of the previous experiments, an approach was made fusing only two models trained on bilinearly stretched spectrograms (Table 54). In this case adding a third model to the fusion trained on 84 bin CQT spectrograms does improve performance (Table 55).

Fold	File Validation Accuracy	File Test Accuracy
1	0.8829	0.6777
2	0.8908	0.6870
3	0.8777	0.6938
4	0.918	0.7018
Averaged Accuracy over 4 folds of test set: 0.7265 Majority Voting over 4 folds of test set: 0.7117		

Table 54: Late Fusion Accuracies of two models trained on bilinearly stretched spectrograms

Fold	File Validation Accuracy	File Test Accuracy
1	0.9214	0.6827
2	0.9156	0.7068
3	0.919	0.7043
4	0.9307	0.7074
Averaged Accuracy over 4 folds of test set: 0.7358 Majority Voting over 4 folds of test set: 0.7246		

Table 55: Late Fusion Accuracies of models trained on bilinearly stretched spectrograms and 84 bin CQT spectrograms

Models trained on a batch size of 50 managed to generalize better. Therefore another late fusion approach was made fusing two models trained on bilinearly stretched spectrograms and a model trained on 84 bin CQT spectrograms. The accuracies of this experiment can be observed in table 56.

Fold	File Validation Accuracy	File Test Accuracy
1	0.913	0.6728
2	0.9187	0.6883
3	0.9204	0.7185
4	0.9358	0.7111
Averaged Accuracy over 4 folds of test set: 0.7364 Majority Voting over 4 folds of test set: 0.7247		

Table 56: Late Fusion Accuracies of bilinearly stretched spectrograms and 84 bin CQT spectrograms

Again the performance slightly improved. Table 57 shows the results of fusing models trained on bilinearly stretched spectrograms with batch size of 50 and 100 with a model trained on CQT spectrograms.

Fold	File Validation Accuracy	File Test Accuracy
1	0.9197	0.6777
2	0.9216	0.70
3	0.9241	0.72098
4	0.9368	0.72098
Averaged Accuracy over 4 folds of test set: 0.74074 Majority Voting over 4 folds of test set: 0.7315		

Table 57: Late Fusion Accuracies of bilinearly stretched spectrograms with different batch size and 84 bin CQT spectrograms

To finally resolve the question if fusing models trained on CQT spectrograms lead to an improvement in generalization, another three experiments were conducted. In the first experiment (table 58) models trained on raw spectrograms extracted with 60 frequency bins, bilinearly stretched spectrograms, standard spectrograms, spectrograms using a sliding window, and 84 bin CQT spectrograms were fused. Table 59 shows the results of a fusion of models trained on spectrograms extracted with 60 frequency bins, spectrograms bilinearly stretched to 300 and 175, and standard spectrograms with a batch size of 50. And the last experiment fused the same models as mentioned for table 59, but additionally adds another model trained on 84 bin CQT spectrograms.

Fold	File Validation Accuracy	File Test Accuracy
1	0.9231	0.6772
2	0.9258	0.6988
3	0.925	0.6963
4	0.94103	0.7136
Averaged Accuracy over 4 folds of test set: 0.7247 Majority Voting over 4 folds of test set: 0.7123		

Table 58: Late Fusion Accuracies of multiple types of raw spectrograms and 84 bin CQT spectrograms

Fold	File Validation Accuracy	File Test Accuracy
1	0.91453	0.6765
2	0.9284	0.7019
3	0.9224	0.7173
4	0.9368	0.7216
Averaged Accuracy over 4 folds of test set: 0.7475 Majority Voting over 4 folds of test set: 0.7377		

Table 59: Late Fusion Accuracies of multiple types of raw spectrograms

Fold	File Validation Accuracy	File Test Accuracy
1	0.9368	0.6852
2	0.9344	0.7019
3	0.9446	0.7105
4	0.9487	0.7228
Averaged Accuracy over 4 folds of test set: 0.7457 Majority Voting over 4 folds of test set: 0.7346		

Table 60: Late Fusion Accuracies of bilinearly stretched spectrograms and 84 bin CQT spectrograms

In the last experiments the addition of the model trained on 84 bin CQT spectrograms did not improve the accuracies. The best performing fusion contained only models trained on raw spectrograms, therefore the inclusion of CQT spectrograms does not improve generalization.

6.2.6 Overview

Again to choose the best performing model we have a look at the means and variances of the predictions of the models for all folds. Figure 31 shows the means and variances of the best performing models on the test set.

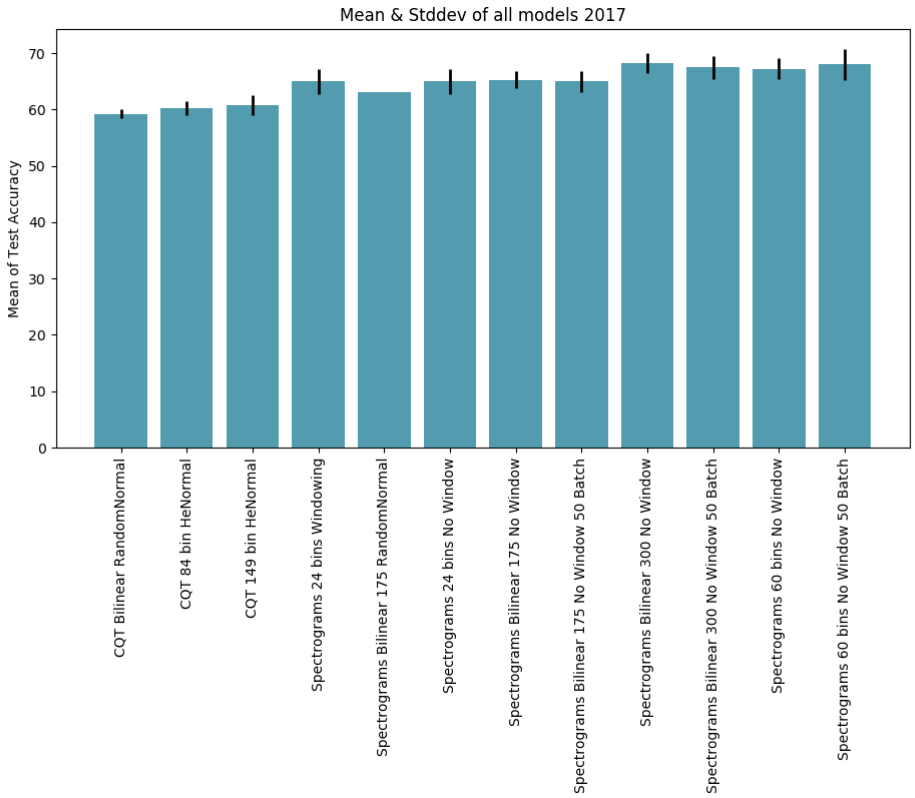


Figure 31: Means and Standard Deviations of the best models on the test set

As in section 5.1.6 the models trained on CQT spectrograms are outperformed by the models trained on raw spectrogram. Also the bigger the spectrograms the better the generalization ability of the network. The best performing model according to figure 32 would be the one trained on raw spectrograms bilinearly stretched to 300 in the frequency dimension. However as we do not have the measure on the unseen test set we have choose the best model according to the performance on the validation set (Figure 33).

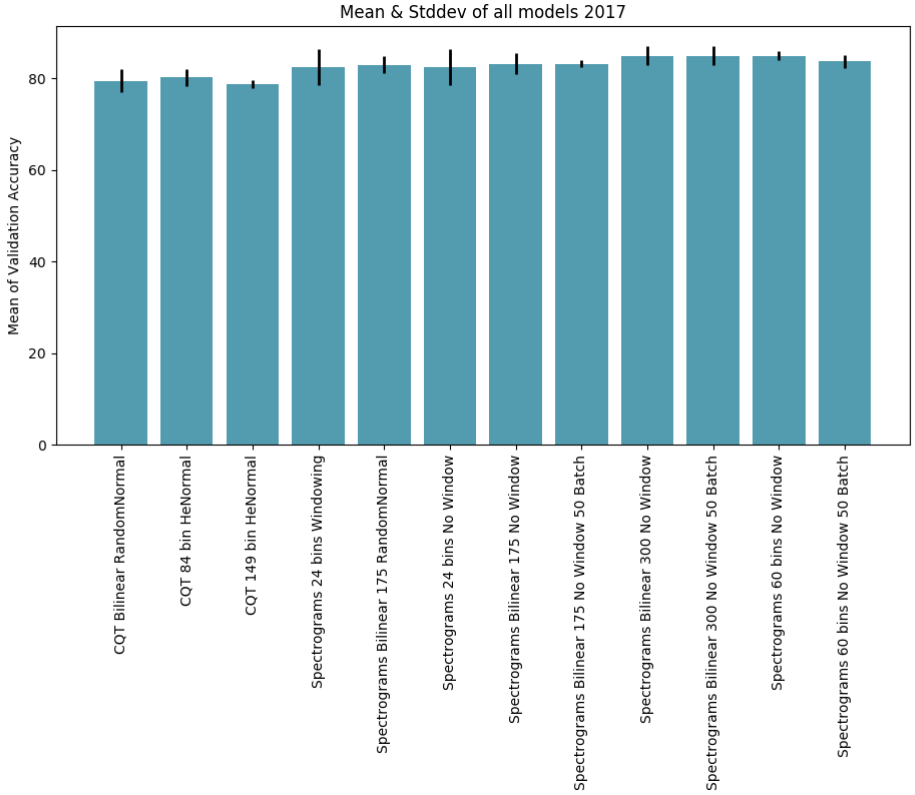


Figure 32: Means and Standard Deviations of the best models on the validation set

According to the performance on the validation set, the chosen model would be trained on raw spectrograms being extracted with 60 frequency bins which also performed very well on the unseen test set. As we use a late fusion approach we can choose some of the best performing models and fuse their predictions together. Table 61 depicts the different fusion approaches, their averaged validation accuracy, and the final accuracy on the test set. Here the fusion approach which yielded the highest validation accuracy was not the one to yield the highest accuracy on the test set. From the conducted experiments we know that including a model being trained on CQT spectrograms would not yield to improvements in the late fusion, therefore we would rather choose fusion approaches excluding CQT spectrograms in the future. The late fusion of models trained on multiple spectrograms did only yield a slight improvement of approximately 2 percentage points. This implies that in some cases it might be better to stick with simple model predictions.

6.3 Conclusion of Results

The experiments conducted on the DCASE 2016 challenge implied that interpolation algorithms lead to a better ability for the model to generalize. However according to the validation accuracies we would have chosen the model trained on spectrograms extracted using 60 frequency bins. This was also the best performing model in the DCASE 2017 on the validation and the test set implying that the size of the spectrograms in fact leads to better generalization. There was only a slight difference to the performance of the model trained on bilinearly stretched spectrograms, which again shows that models trained on interpolated spectrograms generalized very well in both competitions. In some cases also the adjustment of the network architecture showed an improvement in generalization.

Further the CQT spectrograms were outperformed by the raw spectrograms in both challenges. Also the inclusion of models trained on CQT spectrograms into the late fusion did not improve generalization as well as the resizing methods stated in [8]. Overall the best performing models showed a considerable improvement compared to the initial submission [7].

Spectrograms	Bins	Interpolation	Initializer	Validation Accuracy	Test Accuracy
STFT	24	Bilinear 175	RandomNormal	0.89075	0.7105
CQT	84	None	HeNormal		
STFT	24	Bilinear 300	RandomNormal	0.89235	0.7265
STFT	24	Bilinear 175	RandomNormal		
STFT	24	Bilinear 175	RandomNormal	0.91675	0.7358
STFT	24	Bilinear 300	RandomNormal		
CQT	84	None	HeNormal		
STFT*	24	Bilinear 175	RandomNormal	0.921975	0.7364
STFT*	24	Bilinear 300	RandomNormal		
CQT	84	None	HeNormal		
STFT*	24	Bilinear 175	RandomNormal	0.9292555	0.74074
STFT*	24	Bilinear 300	RandomNormal		
STFT	24	Bilinear 300	RandomNormal		
STFT	24	Bilinear 175	RandomNormal		
CQT	84	None	HeNormal		
STFT	24	None	HeNormal	0.92873	0.7247
STFT	24	Bilinear 175	RandomNormal		
STFT	24	Bilinear 300	RandomNormal		
STFT	60	None	RandomNormal		
CQT	84	None	HeNormal		
STFT*	24	Bilinear 175	RandomNormal	0.92553	0.7475
STFT*	24	Bilinear 300	RandomNormal		
STFT*	24	None	HeNormal		
STFT*	60	None	HeNormal		
STFT*	24	Bilinear 175	RandomNormal	0.941125	0.7457
STFT*	24	Bilinear 300	RandomNormal		
STFT*	24	None	HeNormal		
STFT*	60	None	HeNormal		
CQT	84	None	HeNormal		

Table 61: Averaged Validation Accuracies and Test Accuracies of the Fusion approaches

7 Visualizations

There are several packages available with which filters and activations can be visualized with Keras. As the only measure we have so far are the validation accuracies and the learning curves, the visualizations might also help us to find better models and adjust the network architecture appropriately. We can define a Keras backend function as a loss function that focuses on maximizing the activation of certain filters in certain layers and

*For spectrograms marked with an asterisk a batch size of 50 was used for training.

returns the gradient with respect to the pixels of a input image. By normalizing this gradient we avoid very small or very large gradients to assure a gentle gradient ascent procedure. The defined function can further be used to perform gradient ascent in an input space with respect to the filter activation loss. This particular algorithm is called Activation Maximization [37].

Using the keras-vis package [36] even attention maps can be visualized to see on which parts of the spectrogram the network focuses on, and how a perfect spectrogram for a particular class looks like. For visualizing attention maps a similar approach to the one of Activation Maximization is used. This time the gradient with respect to a particular input image is calculated. This tells us the change of the output value with respect to a change in the input and can be used to highlight regions in the input that most contribute to changes in the output [38]. The attention maps are also referred to as heatmaps, or saliency maps in the next sections. By utilizing the Activation maximization algorithm for the last layer in our network with respect to a particular index (referring to a class) we can visualize how a perfect image corresponding to this class would look like.

In the following sections models trained on CQT and raw spectrograms are compared with models trained on the bilinearly stretched spectrograms. Inputs that maximize the activations, saliency maps, and images that are considered to be perfect for a specific class by the network are shown.

7.1 DCASE 2016 Challenge

7.1.1 CQT Spectrograms

7.1.1.1 CQT 84 Bins

First of all a random spectrogram from the training set was chosen as sample and compared with a spectrogram of the test set (Figure 34). It can be observed that the spectrograms do not differ too much, at least to the human eye, hence the much better results of the models for the DCASE 2016 challenge.

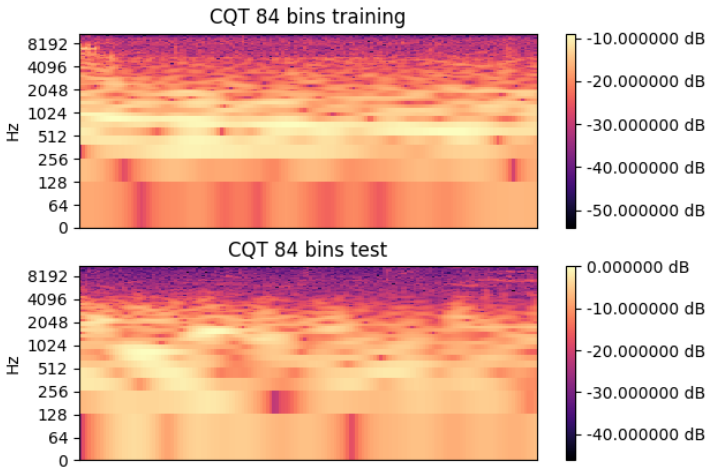


Figure 33: Comparison of a spectrogram from the training set and from the test set

Applying the Activation Maximization algorithm to the network we can extract for each layer the inputs that maximize the activations. The first convolutional layer encodes directions, colors, and rather simple patterns (Figure 35). Delving deeper into the network leads to more and more complex patterns. Figure 36 shows the inputs that maximize the filters in the third convolutional layer. Remarkably, already in the first layers a pattern can be observed referring to the zero padding. Filters which show patterns at the edges seem to focus heavily on the padding. Therefore it might be reasonable to neglect the padding parameter for further experiments. Advancing to even deeper layers the inputs that maximize the activations slowly begin to reconstruct patterns of a spectrogram (Figure 37). In the last layer of the convolutional network the complex patterns add up to what seems to be reconstructed spectrograms (Figure 38).

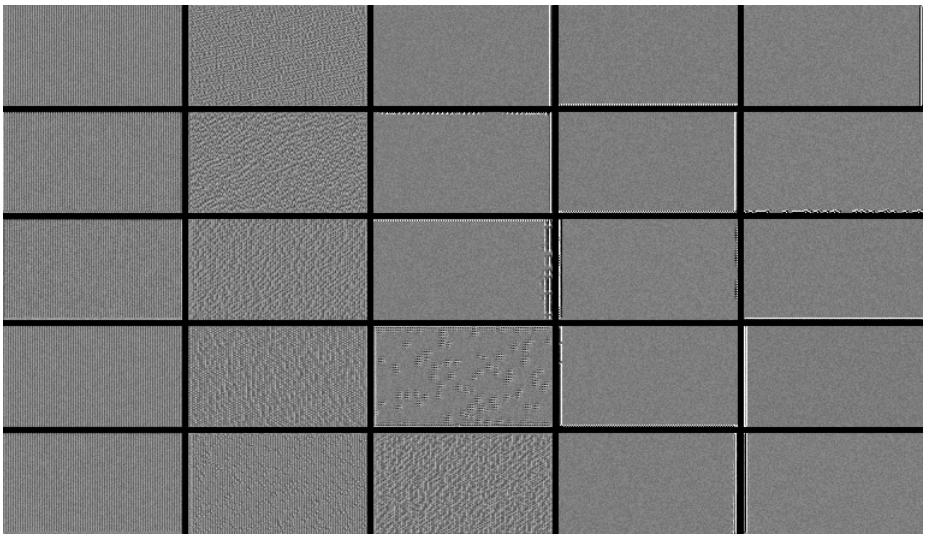


Figure 34: Inputs that maximize activations in the first convolutional layer.

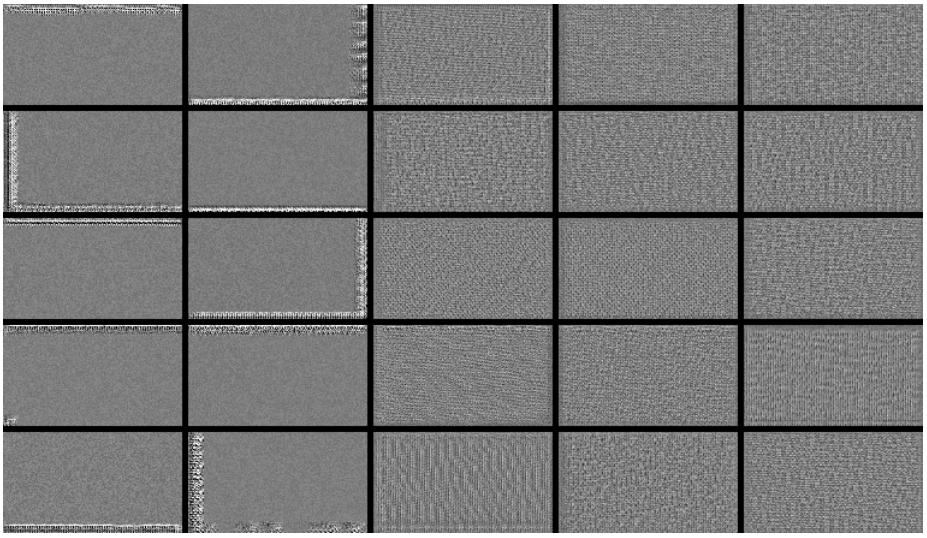


Figure 35: Inputs that maximize activations in the third convolutional layer.

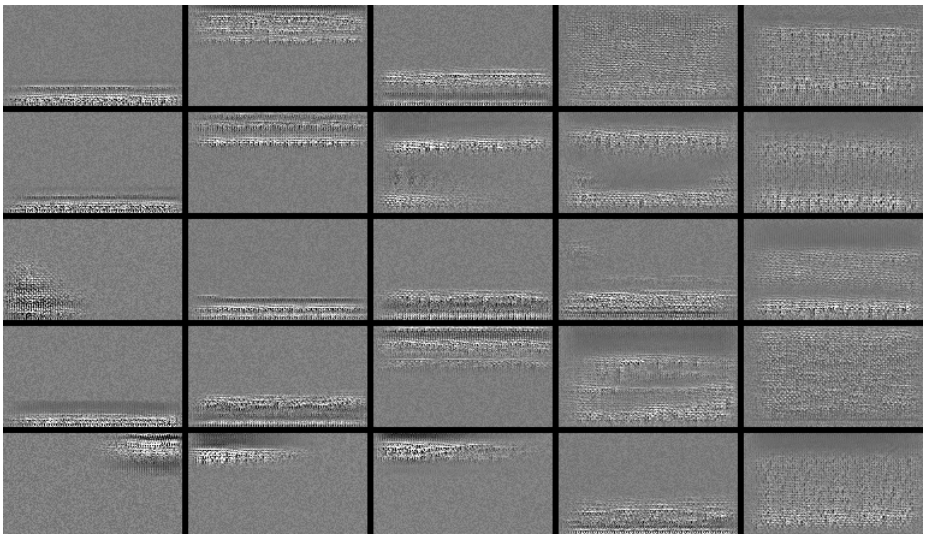


Figure 36: Inputs that maximize activations in the 7th convolutional layer.

Figure 37: Inputs that maximize activations in the last convolutional layer.

As mentioned in section 2.2 the CQT spectrograms are extracted to have a higher resolution for low frequencies. Therefore the network should rather focus on lower frequencies of the spectrogram in order to take advantage of this property of the CQTs. Comparing the heatmap in figure 39 to the sample spectrogram in figure 40, we can see that the network rather focuses on the lower frequencies. This might be a possible reason that the CQT spectrograms were outperformed by the raw spectrograms.

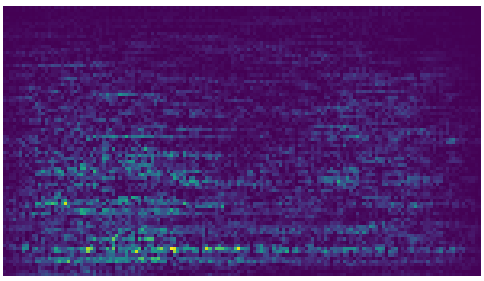


Figure 38: Attention Map of the sample CQT spectrogram of class *beach*

Finally we can recreate how a perfect spectrogram of the class *beach* would look like for the network (Figure 41).

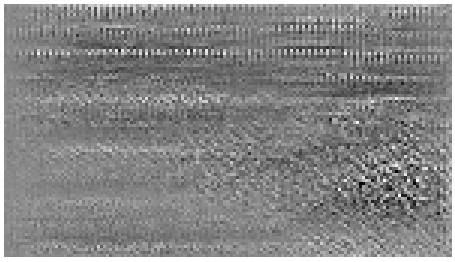


Figure 39: Perfect spectrogram for the class *beach*

7.1.1.2 CQT 84 bins bilinearly stretched

The same procedure was conducted for the best model being trained on bilinearly stretched CQT spectrograms. Figure 42 shows a comparison between a normally created 84 bin CQT spectrogram and a bilinearly stretched CQT spectrogram. There is not much difference to be observed.

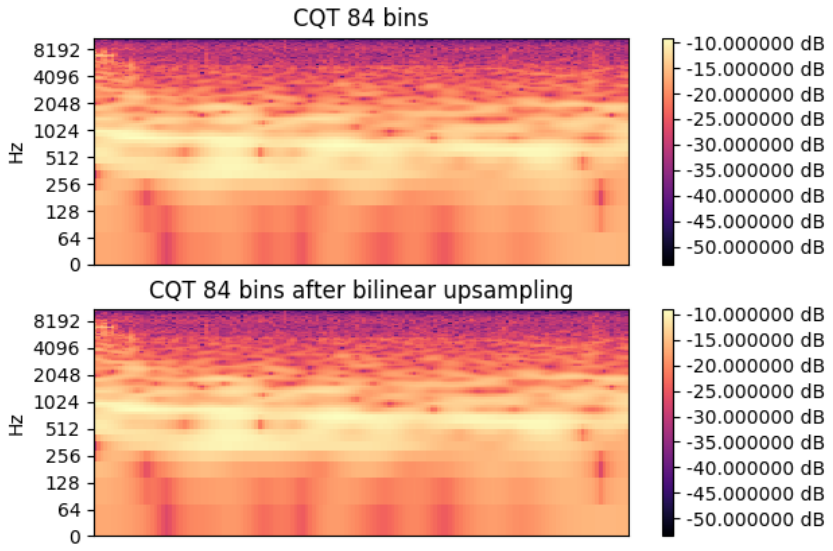


Figure 40: Comparison of normally created CQT spectrogram and upsampled spectrogram.

The following figures depict the outputs of the activation maximization algorithm applied to the first, third, 7th, and last convolutional layer of the network. Again some filters rather focus on the padding than the rest of the activations. The heatmap (figure 45) again shows that the network mainly focuses on the higher frequencies of the spectrogram. Figure 46 depicts the perfect input for the network to classify the sample as of class *beach*.

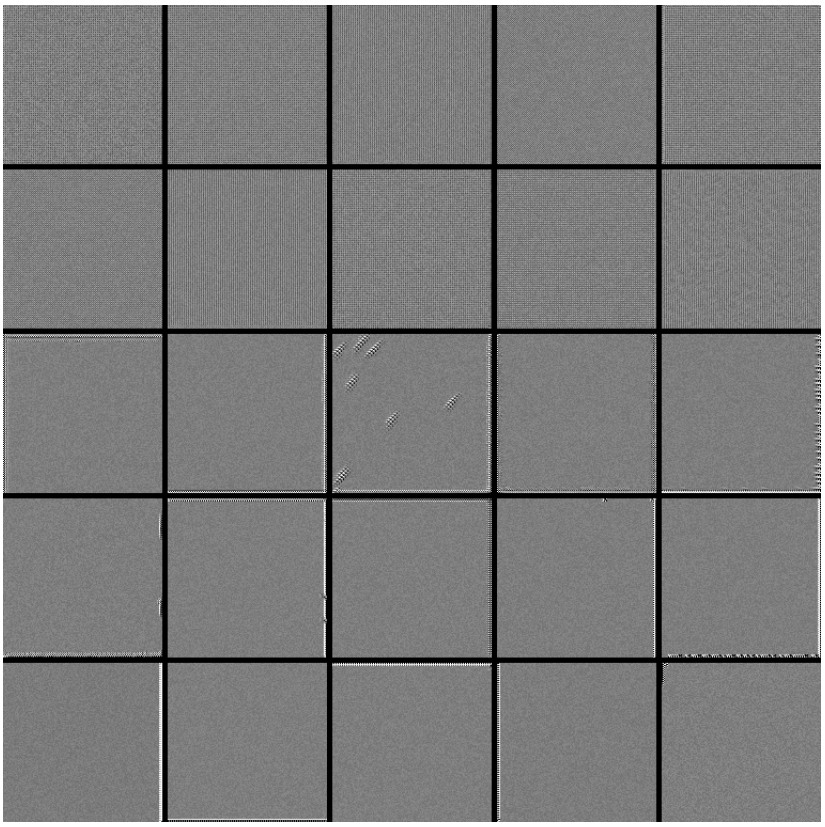


Figure 41: Inputs that maximize activations in the first convolutional layer.

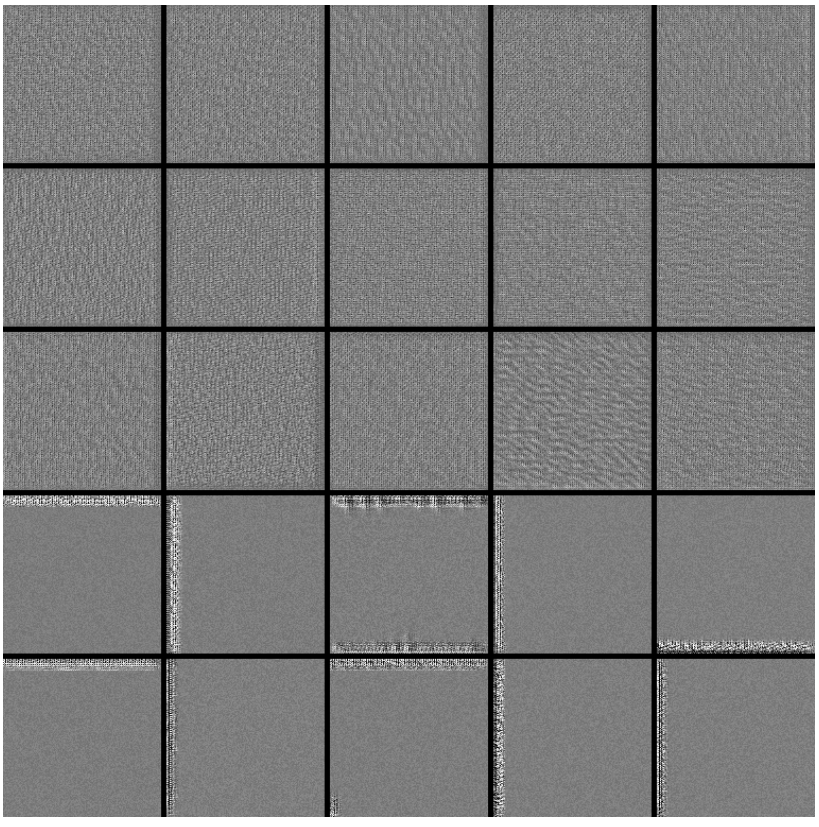


Figure 42: Inputs that maximize activations in the third convolutional layer.

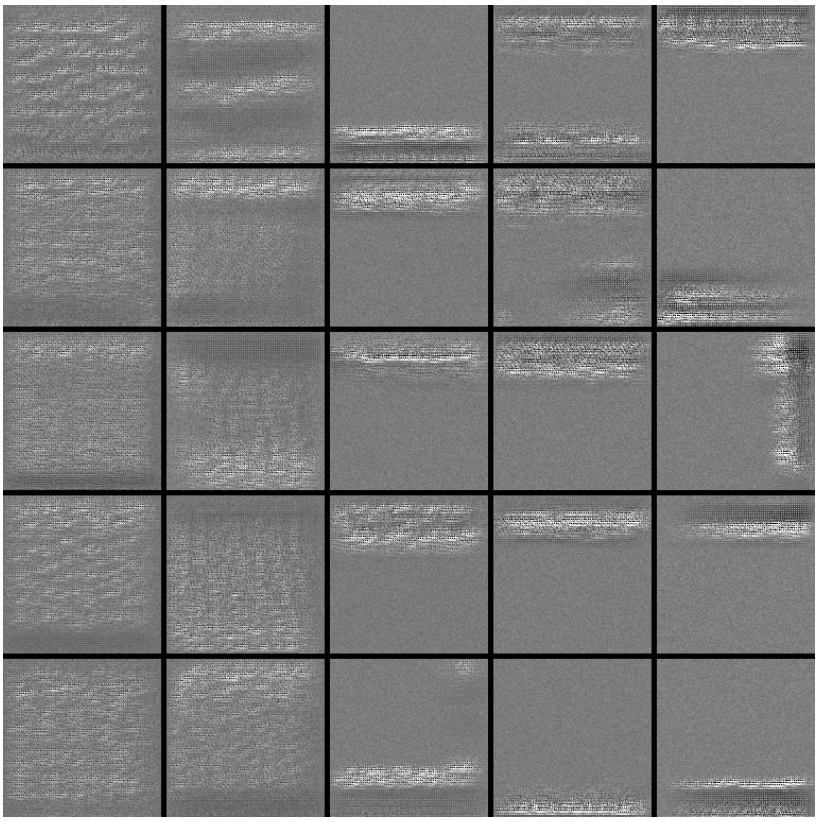


Figure 43: Inputs that maximize activations in the 7th convolutional layer.

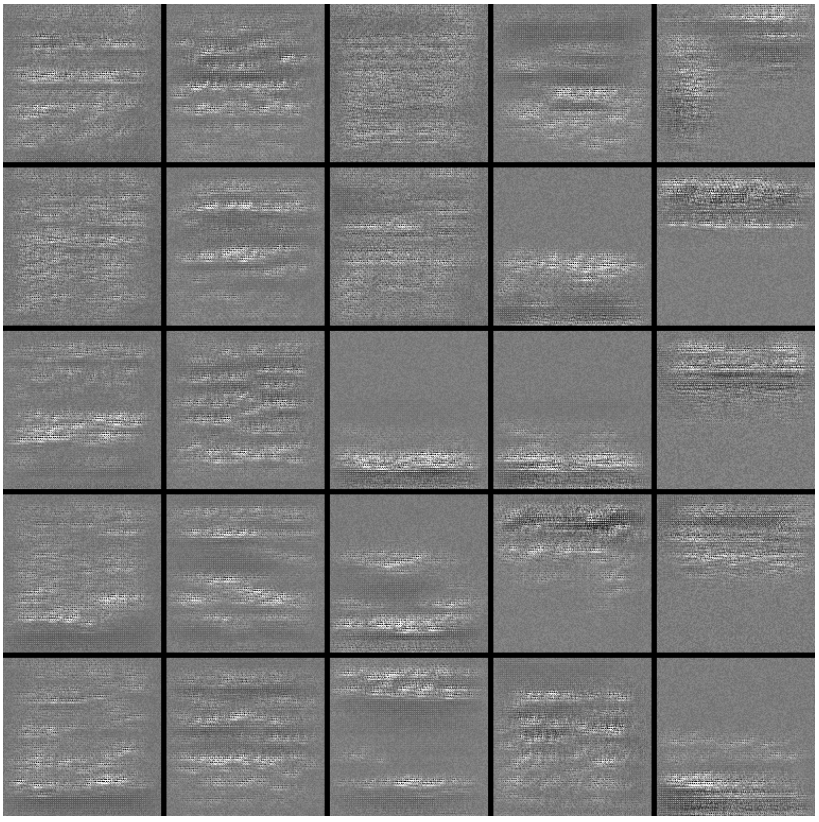


Figure 44: Inputs that maximize activations in the last convolutional layer.

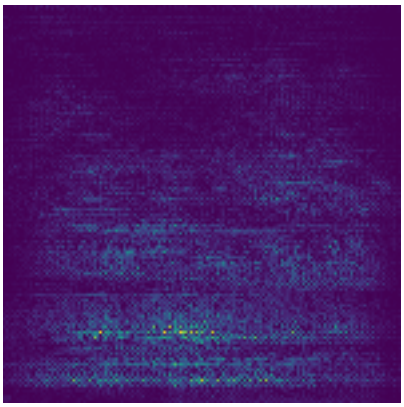


Figure 45: Attention Map of the sample CQT spectrogram of class *beach*

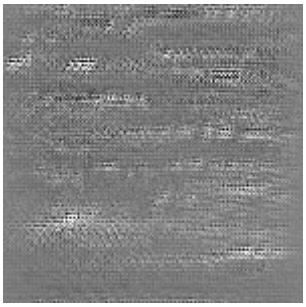


Figure 46: Perfect spectrogram for the class *beach*

7.1.2 Raw Spectrograms

The comparison of the raw spectrogram and the upsampled spectrogram can be observed in figure 47. For the raw spectrograms only the heatmap of the best performing spectrogram was visualized to see on what parts of the spectrogram the network focuses on (Figure 48).

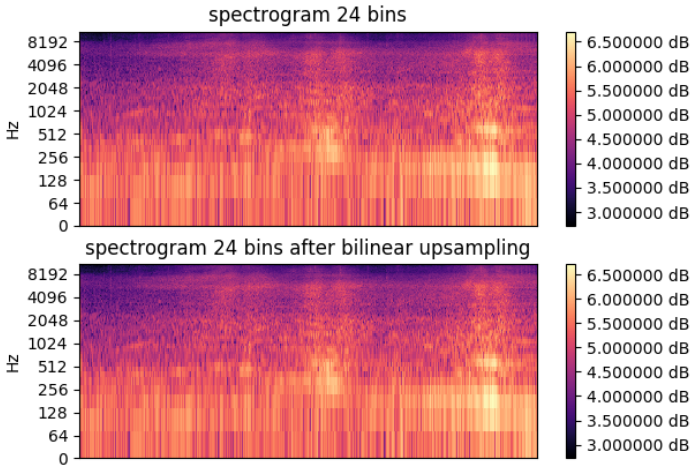


Figure 47: Comparison of training sample and test sample of same class.

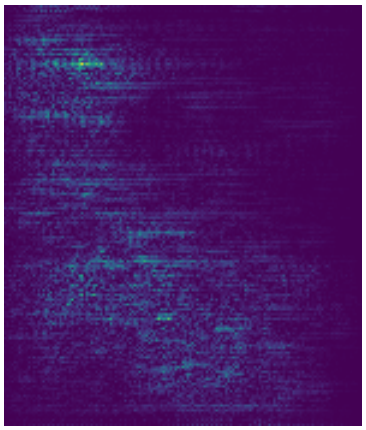


Figure 48: Attention Map of the raw sample spectrogram of class *beach*

In the above attention map we can see that for the raw spectrogram the network considers almost the whole frequency domain of the sample image.

7.2 Best Model of DCASE 2017 Challenge

The best performing model on the unseen test set of the DCASE 2017 challenge was trained on spectrograms using 60 frequency bins for extraction and a batch size of 50. In Figure 49 we can see the comparison of two spectrograms using 60 bins for extraction with one of them originating from the training set and the other from the test set. We can observe that there are big differences in the two spectrograms, especially in the higher frequencies. The heatmap in figure 50 shows that the model focuses rather on the higher frequencies.

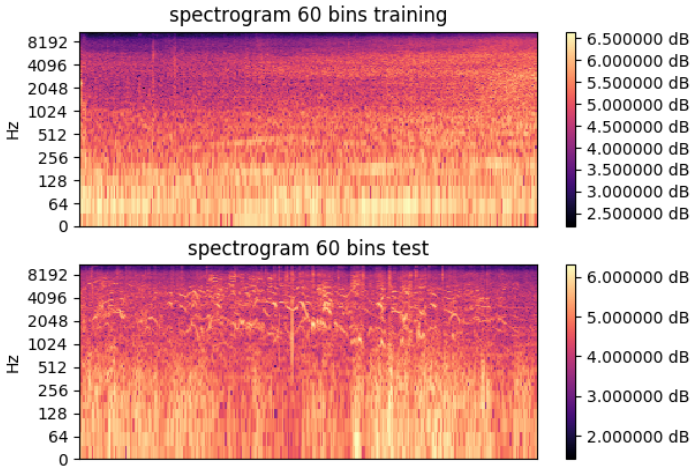


Figure 49: Comparison of training sample and test sample of same class.

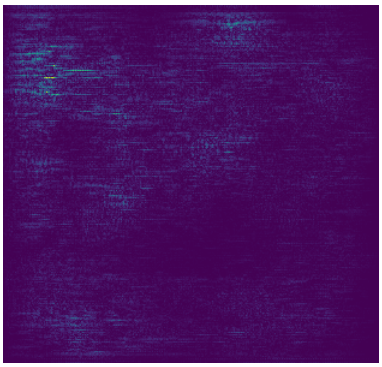


Figure 50: Attention Map of the raw sample spectrogram of class *beach*

7.3 Summary

In this section we gained a short insight into what a network sees and focuses on. Via the Activation Maximization algorithm we can observe the more and more complex patterns of the filters that maximize the activations as we delve deeper into the network. Some filters strongly focused on the zero padding, therefore it might help the model to generalize when omitting the padding. Further by looking at what parts of the spectrogram the network focuses on we might be able to choose appropriate features as input, for example if a model focuses rather on lower frequencies it might be better to choose CQT spectrograms as features as they have higher resolution for lower frequencies.

8 Conclusion

Acoustic Scene Classification remain a very complex task with many approaches seeming to be suitable. Deep Convolutional Neural Networks became one of the most popular of such approaches recently. As it turned out at the DCASE 2016 challenge they can be a very powerful tool, with the winner's submission containing a late fusion of DCNNs and binaural I-vectors [6]. However the submission that won the DCASE 2016 challenge could not generalize as well in the challenge of the following year [7]. Most of Deep Learning approaches suffered from this issue since the distribution of the unseen test set was different from training and validation set. Another submission [8] for the DCASE 2017 challenge however managed to generalize better although they used the same architecture and similar fusion model proposed in[6] . This approach contained a fusion of two models which were trained on different types of spectrograms. With the results collected in this thesis we can deny that the used CQT spectrograms were the source of better generalization. Although they have a higher resolution for lower frequencies, most of the networks mainly focused on the higher frequencies in the spectrograms, as evaluated in section 6. Also the exchange of the Global Average Pooling layer with a Flatten layer (as stated in [8]) does not have any effect on the ability of the model to generalize. However another approach, mainly applying interpolation algorithms onto the spectrograms, led to a better generalization in all cases. Therefore this approach might be a consideration for future tasks. Also extracting spectrograms of bigger size and adjusting the network architecture to fit this size can lead to improvements. The late fusion of multiple DCNNs mostly increased the classification accuracy slightly, however this might not always be the case. In the end generalization remains to be one of the biggest challenges in machine learning.

References

[1] D. Stowell, D. Giannoulis, E. Benetos, M. Lagrange, and M.D. Plumbley. Detection and classification of acoustic scenes and events. Multimedia, IEEE Transactions on, 17(10):1733–1746, Oct 2015.

[2] Mesaros, Annamaria, Heittola, Toni, & Virtanen, Tuomas. (2016). TUT Acoustic scenes 2016, Development dataset [Data set]. Zenodo. http://doi.org/10.5281/zenodo.45739

[3] Mesaros, Annamaria, Heittola, Toni, & Virtanen, Tuomas. (2016). TUT Acoustic scenes 2016, Evaluation dataset [Data set]. Zenodo. http://doi.org/10.5281/zenodo.165995

[4] Mesaros, Annamaria, Heittola, Toni, & Virtanen, Tuomas. (2017). TUT Acoustic scenes 2017, Development dataset [Data set]. Zenodo. http://doi.org/10.5281/zenodo.400515

[5] Mesaros, Annamaria, Heittola, Toni, & Virtanen, Tuomas. (2017). TUT Acoustic scenes 2017, Evaluation dataset [Data set]. Zenodo. http://doi.org/10.5281/zenodo.1040168

[6] Hamid Eghbal−Zadeh, Bernhard Lehner, Matthias Dorfer, Gerhard Widmer, CP-JKU submissions for DCASE 2016: A hybrid apporach using binaural i-vectors and Deep Convolutional Neural Networks, http://www.cs.tut.fi/sgn/arg/dcase2016/documents/challenge_technical_reports/DCASE2016_Eghbal-Zadeh_1028.pdf

[7] Bernhard Lehner, Hamid Eghbal-zadeh, Matthias Dorfer, Filip Korzeniowski, Khaled Koutini, Gerhard Widmer, Classifying short acoustic scenes with i-vectors and CNNs: challenges and optimisations for the 2017 DCASE ASC Task, http://www.cs.tut.fi/sgn/arg/dcase2017/documents/challenge_technical_reports/DCASE2017_Lehner_142.pdf

[8] Zheng Weiping1, Yi Jiantao, Xing Xiaotao1, Liu Xiangtao, Peng Shaohu, Acoustic Scene Classification using Deep Convolutional Neural Network and multiple spectrograms fusion, http://www.cs.tut.fi/sgn/arg/dcase2017/documents/challenge_technical_reports/DCASE2017_Xing_158.pdf

[9] Jan Schlüter, 2017, Deep Learning for Event Detection, Sequence Labelling and Similarity Estimation in Music Signals, Doctoral Thesis, Chapter 2 & 3

[10] Böck, Sebastian and Korzeniowski, Filip and Schlüter, Jan and Krebs, Florian and Widmer, Gerhard, madmom: a new Python Audio and Music Signal Processing Library, Proceedings of the 24th ACM International Conference on Multimedia, October 2016, 1174–1178, 10.1145/2964284.2973795

[11] McFee, Brian, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. "librosa: Audio and music signal analysis in python." In Proceedings of the 14th python in science conference, pp. 18-25. 2015

[12] Ken Turkowski and Steve Gabriel (1990), Filters for Common Resampling Tasks, In Andrew S. Glassner, Graphics Gems I, Academic Press, pp. 147–165, CiteSeerX 10.1.1.116.7898, ISBN 978-0-12-286165-9

[13] D.W. Ruck, S.K. Rogers, M. Kabrisky, M.E. Oxley, B.W. Suter, The multilayer perceptron as an approximation to a Bayes optimal discriminant function, Dec 1990, IEEE Transactions on Neural Networks, Volume: 1, 296 - 298

[14] Y. LeCun, K. Kavukcuoglu and C. Farabet, "Convolutional networks and applications in vision," Proceedings of 2010 IEEE International Symposium on Circuits and Systems, Paris, 2010, pp. 253-256. doi: 10.1109/ISCAS.2010.5537907, http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5537907&isnumber=5536941

[15] Scherer, Dominik and Müller, Andreas and Behnke, Sven, Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition, 2010, Artificial Neural Networks – ICANN 2010, 92–101

[16] Y-Lan Boureau2, Jean Ponce1, Yann LeCun, A Theoretical Analysis of Feature Pooling in Visual Recognition, http://yann.lecun.org/exdb/publis/pdf/boureau-icml-10.pdf

[17] Karen Simonyan, Andrew Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition, 4 Sep 2014, arXiv:1409.1556

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In NIPS, 2012.

[19] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, Going deeper with convolutions, arXiv, 2014.

[20] Zifeng Wu, Chunhua Shen, Anton van den Hengel, Wider or Deeper: Revisiting the ResNet Model for Visual Recognition, 30 Nov 2016, arXiv:1611.10080

[21] Convolutional Neural Network MathWorks, https://de.mathworks.com/solutions/deep-learning/convolutional-neural-network.html

[22] Bottou L. (2010) Large-Scale Machine Learning with Stochastic Gradient Descent. In: Lechevallier Y., Saporta G. (eds) Proceedings of COMPSTAT'2010. Physica-Verlag HD

[23] Diederik P. Kingma, Jimmy Ba, Adam: A Method for Stochastic Optimization, 30 Jan 2017, arXiv:1412.6980

[24] Sashank J. Reddi, Satyen Kale, Sanjiv Kumar, On the Convergence of Adam and Beyond, 15 Feb 2018, ICLR 2018 Conference Blind Submission

[25] Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting", JMLR 2014

[26] Ari S. Morcos1, David G.T. Barrett, Neil C. Rabinowitz, & Matthew Botvinick, On the importance of signle directions for generalization, 15 Feb 2018, ICLR 2018 Conference Blind Submission, https://openreview.net/pdf?id=r1iuQjxCZ

[27] Sergey Ioffe, Christian Szegedy, Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, 2 Mar 2015, arXiv:1502.03167

[28] An In-Depth Evaluation of Multimodal Video Genre Categorization, CBMI 2013, Nov 2, 2014

[29] Keras, Chollet, François and others, 2015, https://keras.io

[30] Theano Development Team, Theano: A Python framework for fast computation of mathematical expressions, May 2016, arXiv e-prints, abs/1605.02688

[31] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Mike Schuster, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[32] Frank Seide, Amit Agarwal, CNTK: Microsoft's Open-Source Deep-Learning Toolkit, 2016, 978-1-4503-4232-2, 2135–2135, http://doi.acm.org/10.1145/2939672.2945397

[33] Sander Dieleman, Jan Schlüter, Colin Raffel, Eben Olson, Søren Kaae Sønderby, Daniel Nouri and others, Lasagne: First release, August 2015, 10.5281/zenodo.27878, http://dx.doi.org/10.5281/zenodo.27878

[34] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification, arXiv:1502.01852

[35] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, Ping Tak Peter Tang, On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima, 15 Sep 2016, arXiv:1609.04836

[36] keras-vis, Kotikalapudi, Raghavendra and contributors, 2017, https://github.com/raghakot/keras-vis

[37] Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent, Visualizing Higher-Layer Features of a Deep Network, June 9th, 2009, https://www.researchgate.net/profile/Aaron_Courville/publication/265022827_Visualizing_Higher-Layer_Features_of_a_Deep_Network/links/53ff82b00cf24c81027da530.pdf

[38] Karen Simonyan, Andrea Vedaldi, Andrew Zisserman, Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps, 19 Apr 2014, arXiv:1312.6034v2 [cs.CV]