**LOW-COMPLEXITY ACOUSTIC SCENE CLASSIFICATION IN DCASE 2022 CHALLENGE**

**CHAPTER -1**

**ABSTRACT**

This paper presents an analysis of the Low-Complexity Acoustic Scene Classification task in DCASE 2022 Challenge. The task was a continuation from the previous years, but the low-complexity requirements were changed to the following: the maximum number of allowed parameters, including the zero-valued ones, was 128 K, with parameters being represented using INT8 numerical format and the maximum number of multiply-accumulate operations at inference time was 30 million. The provided baseline system is a convolutional neural network which employs post-training quantization of parameters, resulting in 46.5 K parameters, and 29.23 million multiply-and-accumulate operations (MMACs). Its performance on the evaluation data is 44.2% accuracy and 1.532 log-loss. In comparison, the top system in the challenge obtained an accuracy of 59.6% and a log loss of 1.091, having 121 K parameters and 28 MMACs. The task received 48 submissions from 19 different

Teams, most of which outperformed the baseline system.

**Index Terms**— Acoustic scene classification, low-complexity, DCASE Challenge.

**CHAPTER-2**

**INTRODUCTION**

The task of acoustic scene classification is defined as classifying a short excerpt of audio into a class of a predefined set of classes, that indicates the context where the audio was recorded [1]. The task has been one of the main topics in the DCASE Challenge from its inception, and has developed from the original setup to include different additional problems, such as multiple devices and low-complexity conditions [2]. The current setup advances further towards real world applicability by defining the low-complexity constraints, in terms of maximum number of parameters and maximum number of operations permitted at inference time, typical of current IoT devices (or microcontrollers).For real-world applications, a classification method for acoustic scenes is expected to work in very diverse conditions, including audio captured with different devices and as short as possible inference time. The first task on low-complexity acoustic scene classification was defined in 2020 for only three classes and a single device [2], for which many submissions obtained very high performance. The solutions most commonly imposed restrictions on the model architectures, using slim models and depth-wise separable CNNs. In addition, pruning and post-training quantization of the model weights were popular choices [3]. The data mismatch between training and testing when dealing with multiple devices has been first introduced as a separate task in 2019, and then repeated in 2020. The majority of the systems handled the mismatch through data augmentation [2], with the best performance in the 2020 task being 76.5% accuracy and 1.21 log loss [4].The combination of multiple devices and low-complexity requirements was introduced in DCASE 2021 Challenge, with the model complexity limit being set to 128 K for the non-zero parameters. This created the situation in which most high-performing systems were very close to the allowed model size limit. Sparsity used in combination with quantization emerged as a popular and efficient way of reducing the model size, while the most popular system architectures submitted in 2021 were residual models. A few teams used modified versions of available residual models suitable for processing power-constrained devices, like Mobile Net [5] and Efficient Net [6]. The best performing system had an accuracy of 76.1% and log loss of 0.724 [7], while 18 submitted systems had an accuracy above 70%.The current edition formulates the problem of low-complexity acoustic scene classification by defining more concretely the low complexity limitations by selecting a class of target devices for which the developed system should be suitable. This results in the number of allowed parameters being maximum 128 K, counting all parameters, in contrast from DCASE 2021 when only the non-zero ones were counted. In addition, a limit of 30 million multiply-accumulate operations (MMACs) is approximated based on the computing power of the target device class. This paper introduces the results and analysis of the DCASE 2022 Challenge Task 1: Low-Complexity Acoustic Scene Classification with Multiple Devices. The paper is organized as follows: introduces the task setup, dataset, and baseline system present the challenge participation statistics and analysis of the submitted systems, respectively, presents conclusions and ideas for future development of this task.

**2.1. Dataset and performance evaluation**

The task uses TAU Urban Acoustic Scenes 2022, a newly released version of the previous acoustic scene datasets. The data consists of recordings from ten acoustic scenes which represent the target classes [8]: airport, indoor shopping mall, metro station, pedestrian street, public square, street with medium level of traffic, travelling by a tram, travelling by a bus, travelling by an underground metro and urban park. Data was recorded in multiple European cities, with recordings from ten cities available in the training set and 12 in the evaluation set (two new cities compared to the training).The audio files have been recorded simultaneously with four devices denoted A, B, C, and D, and another 11 devices denoted S1-S11 were simulated using the audio from device A. The development and evaluation sets consist of 64 and 22 hours of data, respectively. For complete details on the dataset creation and the exact amounts of data per device, we refer the reader to [2]. The difference from the previous datasets is that for this edition the audio data is presented in segments having a duration of 1 s, in order to comply with the inference time and computational limitations imposed by the considered target devices. The submissions were evaluated using multi-class cross entropy and accuracy. Accuracy was calculated as macro-average (average of the class-wise performance for each metric), but because the data is balanced, this corresponds to the overall accuracy, the systems were ranked based on the multi-class cross-entropy (log loss), for a ranking independent of the operating point. As in each edition of the challenge, the audio material in the evaluation data was released two weeks prior to the challenge deadline. The participants were expected to provide class predictions for the provided audio material, and submit the system output for evaluation, together with additional information on the methods. The reference annotation of the evaluation data is only available to task organizers and was used for scoring the submissions.

**CHAPTER-3**

**LITERATURE REVIEW**

**[1] E. Benetos, D. Stowell, and M. D. Plumbley:** This chapter presents state-of-the-art research and open topics for analyzing complex sound scenes in a single microphone case. First, the concept of sound scene recognition is presented, from the perspective of different paradigms (classification, tagging, clustering, segmentation) and methods used. The core section is on sound event detection and classification, presenting various paradigms and practical considerations along with methods for monophonic and polyphonic sound event detection. The chapter will then focus on the concepts of context and “language modeling” for sound scenes, also covering the concept of relationships between sound events. Work on sound scene recognition based on event detection is also presented. Finally the chapter will summarize the topic and will provide directions for future research.

**Summary:** Studied about Approaches to Complex Sound Scene Analysis.

**[2] T. Heittola, A. Mesaros, and T. Virtanen:**

This paper presents the details of Task 1: Acoustic Scene Classification in the DCASE 2020 Challenge. The task consists of two subtasks: classification of data from multiple devices, requiring good generalization properties, and classification using low-complexity solutions. Here we describe the datasets and baseline systems. After the challenge submission deadline, challenge results and analysis of the submissions will be added.

**Summary:**  Studied about generalization across devices and low complexity solutions,” in Proceedings of the Detection and Classification of Acoustic Scenes and Events.

**3] K. Koutini, F. Henkel, H. Eghbal-zadeh, and G. Wid-mer:**

This technical report describes the CP-JKU team’s submission for Task 1 – Subtask A (Acoustic Scene Classification with Multiple Devices) and Subtask B (Low-Complexity Acoustic Scene Classification) of the DCASE-2020 challenge [1]. For Subtask 1.A, we pro vide our Receptive Field (RF) regularized CNN model as a baseline, and additionally explore the use of two different domain adaptation objectives in the form of the Maximum Mean Discrepancy (MMD) and the Sliced Wasserstein Distance (SWD). For Subtask 1.B, we investigate different parameter reduction methods such as Pruning, while maintaining the receptive field of the networks. Additionally, we incorporate a decomposed convolutional layer that reduces the number of non-zero parameters in our models while only slightly

decreasing the accuracy, compared to the full-parameter baseline.

**Summary:** Studied about low-complexity cross-device acoustic scene classification with rf-regularized cnns.

**[4] S. Suh, S. Park, Y. Jeong, and T. Lee:** his technical report describes our Acoustic Scene Classification systems for DCASE2020 challenge Task1. For subtask A, we designed a single model implemented with three parallel ResNets, which is named Trident ResNet. We have confirmed that this structure is beneficial when analyzing samples collected from minority or unseen devices, and confirmed 73.7% classification accuracy for the test split. For subtask B, we used the Inception  
module to build a model named Shallow Inception that has fewer parameters than the CNN of the DCASE baseline system. Due to the sparse structure of the Inception module, we have enhanced the accuracy of the model up to 97.6%, while reducing the number of parameters.

**Summary:** Studied about Designing Acoustic Scene Classification Models with CNN Variants.

**5] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen:** In this paper we describe a new mobile architecture, MobileNetV2, that improves the state of the art performance of mobile models on multiple tasks and benchmarks as well as across a spectrum of different model sizes. We also describe efficient ways of applying these mobile models to object detection in a novel framework we call SSDLite. Additionally, we demonstrate how to build mobile semantic segmentation models through a reduced form of DeepLabv3 which we call Mobile DeepLabv3.

The MobileNetV2 architecture is based on an inverted residual structure where the input and output of the residual block are thin bottleneck layers opposite to traditional residual models which use expanded representations in the input an MobileNetV2 uses lightweight depthwise convolutions to filter features in the intermediate expansion layer. Additionally, we find that it is important to remove non-linearities in the narrow layers in order to maintain representational power. We demonstrate that this improves performance and provide an intuition that led to this design. Finally, our approach allows decoupling of the input/output domains from the expressiveness of the transformation, which provides a convenient framework for further analysis. We measure our performance on Imagenet classification, COCO object detection, VOC image segmentation. We evaluate the trade-offs between accuracy, and number of operations measured by multiply-adds (MAdd), as well as the number of parameters.

**Summary:** Studied about MobileNetV2: Inverted Residuals and Linear Bottlenecks.

**CHAPTER-4**

**EXISTING METHOD**

The task of acoustic scene classification was set up as a straightforward multi-class supervised classification problem, with class labels describing the acoustic scene. Labelled  
audio examples were provided for training the systems, with each audio example having a single label. For each test example, a system was expected to provide a label from the set  
of known labels.  
**2.1. Dataset**  
The task used the TUT Acoustic Scenes 2017 dataset, containing audio recorded in 15 different acoustic scenes; 3-5 minutes of audio was recorded in various locations for the  
following acoustic scenes: bus, cafe/restaurant, car, city centre, forest path, grocery store, home, lakeside beach, library, metro station, office, residential area, train, tram, and park.  
The development dataset has the same content as the complete TUT Acoustic Scenes 2016 dataset, but with original recordings being split into 10 s segments. The short audio segments provide less information for the decision making process in classification, thus increasing the task difficulty from the previous edition. This length is regarded as challenging for both human and machine recognition, based on the study in [2]. The development dataset contains 312 segments of 10 s per scene class (52 minutes). The evaluation  
dataset was recorded in similar locations approximately one year later than the development data, and contained 108 segments of 10 s per scene class (18 minutes). A detailed description of the data recording and annotation procedure is available in [13], while a more detailed description of the TUT Acoustic Scenes 2017 dataset can be found in [6].  
**2.2. Baseline system**  
The baseline system provided for this task uses a multilayer perceptron architecture (MLP) trained on log me band energies calculated in 40 Ms frames with a 50% overlap and 4050% mel bands. A 5-frame context was used, resulting in a feature vector length of 200. The MLP had two dense layers of 50 hidden units each, with 20% dropout, and an output layer of 15 softmax type neurons. Frame-based decisions from the network output were combined by majority voting to obtain the final class decision for one 10 s long test audio segment.  
The classification accuracy obtained by the system on the development set using the provided cross-validation setup is 73.8%, with class-wise performance ranging from 57% to  
99.7%. Performance on the evaluation dataset is 61%. A detailed description of the baseline system and its class-wise performance can be found in [6].  
  
**3.1. Submission statistics and ranking**  
A number of 97 systems were submitted for this task, corresponding to 39 teams and 129 authors. The number of participating teams is similar to previous edition (34 teams in 2016),  
but the number of submissions was much higher because each team was allowed to submit a maximum of 4 systems, even though not all of them did so. Most of the submitted systems  
outperformed the baseline system. A selection of top systems performance and 95% confidence interval is presented in Fig.2. Confidence intervals were calculated as a binomial proportion confidence interval for the classification output being correct or incorrect with respect to the ground truth. Based on Fig. 2, it can be seen that the confidence intervals for systems of neighbouring ranks overlap significantly.  
**3.2. Submissions analysis**  
A general analysis of the characteristics of the submitted systems reveals that the most popular classification approach was the convolutional neural network, with 55 of the 97 submission being based on a CNN architecture. In some cases the CNN was used as part of an ensemble, combined with a variety of techniques such as multilayer perceptron (MLP), recurrent neural networks (RNN), support vector machines (SVM),Gaussian mixture models (GMM), and i-vector. Recurrent network architectures were part of 18 systems, some being convolutional (CRNN), others LSTM and bi-LSTM. The CNNs are used in acoustic scene and generally in audio classification as a form of image processing, with their connectivity patterns exploiting regions in the time-frequency representations of signals, therefore being capable of capturing both time and frequency evolution of signals. On the other hand, RNNs are much better at capturing the long-term temporal characteristics, with the LSTM variants having very good internal memory capabilities for processing of time-series. Also MLP and  
SVM were popular choices, with 11 systems each, most often as part of an ensemble of classifiers. All systems in top 10 make use of CNNs in some way, while first non-CNN-based, ranked 14 and 16, and use MLP. Table 1 presents a selection of top systems and their characteristics, while Fig. 3 shows the confusion matrix of the top performing system.  
Most submissions were based on mel-scale representations, with log mel energies and MFCCs being used in 27 and 19 systems, respectively. Mel-scale representations are  
often used and generally work well in sound classification problems, their modelling of human perception making them a comfortable choice when no better assumptions on the data can be made. Other spectral representation include spectrogram and CQT [15], [16] with CQT probably made popular by previous edition runner-up system. CQT is often used in  
music analysis for its exponential frequency resolution and for preserving the relative positions of harmonics, but its use for environmental sound analysis is not as clearly motivated. While in 2016 CQT was used in three systems, this time there were 13, of which 9 relied solely on CQT, and others used it in combination with spectrogram or MFCC. There was also one system based on low-level features that included spectral centroid, roll off, zero-crossing rate and MFCCs and their derivatives, ranked only 54, at same level with the baseline.Many participants made use of binaural audio, with one third using the two channels separately instead of the averaged audio provided as example in the baseline system. This was mostly used as a way to obtain more data for the deep-learning methods, with the different channels having slight variations in the captured audio. Another new element was the use of specific data augmentation techniques, unnoticed in 2016: there was much use of block mixing, pitch shifting, time stretching, mixing files of the same class, and adding Gaussian noise, in some cases all the techniques being used in the same system. A novel and unique method in the challenge was the augmentation of the dataset using generative adversarial networks (GAN), by the system that also achieved the best performance [14]. All data augmentation techniques are motivated by the use of deep learning, for creating more data and adding more acoustic variability to allow better learning  
and generalization. A comparison of systems performance on the development and evaluation datasets reveals that most systems have a significant drop in performance for the evaluation dataset (10-20% in term of absolute accuracy). This is likely due to the mismatch in the data recording conditions, as the evaluation data was recorded one year later at similar or, in some cases, same locations. The situation was not intentional, being just a consequence of extending the previously available data with a new evaluation dataset, but it reveals the ease with which neural-network based systems over fit the data. As  
an observation, augmenting the dataset using GAN seemed to offer a more consistent performance in conjunction with the deep-learning methods, the corresponding system having only a 4% absolute drop in accuracy between development and evaluation sets. The Pearson correlation coefficient calculated between the development and evaluation performance for all systems is 0.42, which can be considered a medium strength of association between the two. This suggests that the performance of systems is somewhat consistent, and the gap in performance is due to data mismatch and not lack of generalization properties in the systems. Considering only the best system of each team, the correlation between the development and evaluation performance is 0.69, indicating very  
strong correlation. Based on this, we can assert that each team has produced at least one system that generalizes well for unseen data. .

**3. Statistical analysis of systems performance**  
the confidence intervals presented in Table 1 show that there is not a significant difference between performances of closely ranked systems, with only the top system being set apart from the others. To understand how much the different systems take similar or different decisions, the systems were compared in pairs using McNemar’s test [12]. McNemar’s test  
for comparing classifiers examines only the cases in which the prediction of one system is correct, and the prediction of the second system is wrong, therefore identifying if there is a  
difference in systems with respect to the test samples that are more difficult to classify. For systems with similar accuracy, this test indicates if the difference is statistically significant.  
The null hypothesis for the statistical test is that the two classifiers being compared perform similarly, while the alter- native hypothesis is that the difference is statistically significant. Figure 4 illustrates the results of this test using a significance level of 0.05. A red square in the illustration indicates a pair of systems for which the result does not allow rejecting the null hypothesis. For this comparison we considered only the best system of each team, plus the baseline, with the systems considered in order of their accuracy (team rank order).  
As expected, we notice that many systems on neighbouring ranks perform equivalently, with the indicators aligned close to the diagonal. The top four compared systems show sta tistically significant differences, while already between the fourth and the fifth the difference is not statistically significant. These are the same systems presented in Table 1, ranked  
1, 2, 6, 8, and 9, with accuracies of 83.3%, 80.4%, 77.7%,74.1%, and 73.8%, respectively. The second to fifth ranked submissions all belong to the same authors [17] and have  
accuracies from 80.4% to 79.6%, being based on the same method with very slight variations, with no significant difference detected using McNemar’s test.Using the information that the first three systems in our comparison are significantly different, we calculate the performance of their combined outputs with a majority vote rule.The obtained performance is only 84.69%, which is not much higher than the 83.3% accuracy of the top system, meaning that in many cases two of the three systems still misclassify the data. If we investigate the best case scenario between the three systems, by considering a correct item if at least one of the systems has classified it correctly, we obtain an accuracy of 96.05% - this indicates that most test items are indeed correctly classified by at least one of the three considered systems, and the possibility of improving performance by classifiers fusion exists, if suitable rules for fusion can be found. The average performance of all 97 systems is 64.33%, while a majority vote fusion of all systems obtains a performance of 73.52%. We contrast this with the human performance obtained on similar data [18], in which average human performance was 54.4% (87 participants), with participants from Finland, familiar with the recorded soundscape, scoring a better accuracy of 60%.

**CHAPTER-4**

**PROPOSE METHOD**

**2. System complexity requirements**

The computational complexity is measured in terms of parameter count and MMACs (million multiply-accumulate operations) with the requirements modelled after Cortex-M4 devices (e.g.STM32L496@80MHz or Arduino Nano 33@64MHz). The maximum number of parameters is 128 K, with variable type fixed into INT8, and counting all parameters. This is a major difference from DCASE 2021 in which the 128 K model size limit was only for non-zero parameters, and there was no specific format imposed on the numerical representation. This change was made because in a real operational situation, even with a sparse model, the zero-valued parameters add to the number of MACs performed at inference, and produce additional computational overhead for handling sparsity.The maximum number of MACS per inference is 30 MMACs approximated based on the computing power of the target device  
class. This limit mimics the fitting of audio buffers into SRAM (fast access internal memory) on the target device for the analysis segment of 1 s, and allows some head space for feature calculation (e.g. FFT), assuming that the most commonly used features fit under this limit. In case learned features (embeddings) are used, e.g. VGGish [9], OpenL3 [10] or EdgeL3 [11], the network used to generate them contributes to the overall model size and complexity. Participants are required to provide full information about the model size and complexity in their technical report accompanying the submission. To facilitate model size calculation for the challenge participant, a script for calculating the number of parameters and the MMACs is provided for Keras, TFLite and PyTorch models1.  
**3. BASELINE SYSTEM**The baseline system has the same architecture as the 2021 one, being based on a convolutional neural network (CNN). The system consists of three CNN layers and one fully connected layer, followed by a softmax layer. The model is trained for 200 epochs with  
a batch size of 16. Complete details about the model and the parameters are provided with the code2. The feature extraction step follows a classical approach, where log mel-band energies are extracted every 40 ms with a 50% hop size. This results in an input shape of 40 × 51 for each 1 second audio file. Post-training quantization to 8 bits is used to reduce the model complexity. The quantization was done after training, using TFLite from TensorFlow 2.0,and setting the weights to INT8 type. The baseline system has a total number of parameters of 46512. The baseline system overall performance on the development data and system complexity information are provided in Table 1.  
4. CHALLENGE RESULTS  
The task received 48 submissions from a number of 19 teams. The number of participants in this edition is lower than in previous years,but similar to participation statistics of the other tasks. Only three of the 19 teams have lower performance than the baseline. The best  
system has a log loss of 1.091 and accuracy of 59.6%, with the four best spots belonging to team Schmid CPJKU [12].

**4.1. Performance analysis**  
The performance (log loss and accuracy) obtained by the top 10 teams, best system of each team, are presented in Table 2 and de- picted in Figure 2. The submission label was simplified to remove redundant information; submission number was kept for correspondence with the results on the website3. The 95% confidence intervals for log loss were calculated using the jackknife procedure.The ranking of the systems is based on log loss, where the top ranked one is the system of Schmid CPJKU [12] with a log loss of 1.091. Its accuracy of 59.6% is second-best accuracy among the 48 submissions. Team Chang HYU [13] is ranked second by log loss, but has the overall best accuracy among all submissions. Their accuracy is 60.8%, which appears to be significantly higher than Schmid CPJKU according to the 95% confidence interval, given the amount of data in the evaluation set. Compared to last edition, the top accuracy has decreased by 16%, and the log loss of the top systems is much higher. While in 2021 there were 21 systems with a log loss under 1, this year there is none. Top 10 systems have a log loss under 1.5, and an accuracy between 45.9%-60.8%. The decrease in performance is mostly a consequence of the data segment size being reduced from 10 to 1 second.Considering all submissions, the difference in performance between data belonging to devices seen or not in training is generally 10% in accuracy. However, the simulated unseen devices still have a better recognition rate than data from the real device D, which is the GoPro - it appears that its characteristics are very different from those of handheld devices developed for audio (mobile phones and tablets). Among the seen devices, the mobile devices have similar recognition rate, whether real (B, C) or simulated (S); systems have slightly better performance on device A. Given that most data in development set belongs to device A, the relatively small difference in performance among devices shows that the systems have very strong generalization properties which cover the device miss match. We also observe good generalization between seen and un-seen cities, with almost no difference in classification performance between them. Class-wise performance indicates that some acoustic scenes are more difficult overall: while scenes like bus or park obtain accuracies over 70-80% for many systems, the large majority of systems classify scenes from Pedestrian Street and Public Square with around 30% accuracy only.  
**4.2. Machine learning characteristics**  
regarding feature extraction, all the systems make use of log-Mel energies or Mel spectrogram, sometimes in combination with other features like deltas, spectral entropy/flatness, CQT or Gammatone. Augmentation techniques are used by most of the systems, only 5 teams do not report use of augmentation4. The most popular technique is mixup (used by 33 systems), followed by SpecAugment and pitch shifting (used by 16 systems). Only one team, Zou PKU [14] uses SpecAugment++, which is applied not only at the input but also at the hidden space of the neural network, to enhance also the intermediate feature representations. The system is ranked 7th based on the accuracy. The most popular architectures are CNNs (used by 34 systems); some report use of MobileNet [5] (still convolutions, but depthwise separable) or BC-ResNet [7]. The use of residual models is reported by five teams, a significant reduction compared to the 2021 edition when residual networks were the most popular architecture.The top team Schmid CPJKU uses a teacher-student setup, where the PaSST models pretrained on AudioSet are used as teacher, and the student model is a RF-regularized CNN [15]. The system is based on their previous submission’s system reducing its complexity to fit the current constraints. For data augmentation they use Frequency MixStyle, mixing frequency-wise statistics to enhance device generalization.  
**4.3. System complexity analysis**  
Almost all submissions are based on inverted residual blocks, or a slight variation of this convolutional block. This is mainly because the common pattern for all participants was to adapt state-of- the-art convolutional networks to meet the computational requirements. Among the adapted networks there are Mobile Nets and BC Resents, and one submission with Shuffle Net. Other notable solutions include the use of very involved feature extraction solutions coupled with very simple neural architectures. While the networks where only slightly modified or carefully designed to meet the computational requirements (without any particular trick), a lot of focus was put on the training and data augmentation strategies. In particular, to boost inference performance, quantization-aware training (QAT) was applied by most of the participants. Another common alternative was knowledge distillation with pertained bigger networks fine-tuned on the proposed task. Given the homogeneity in  
network topology of the submissions, the models proposed perform imilarly in acoustic sound classification without being to diverse in computational requirements.  
Three out of the top four performing models are based on architectures characterised by large receptive fields employing, respectively, a transformer architecture, coordinate attention and an encoder-decoder architecture. This proved to optimize the performance given the limited resources available, cleverly maximizing the working memory usage of the network, as this parameter was not limited in the task description. Another notable architecture is that proposed in AIT Essex [16], providing almost optimal performance but very limited MMACs and/or parameter usage. This is possible thanks to their optimized convolutional block, which resembles a grouped convolution whose inputs are a combination of the original input sequence. More standard approaches, based on BC-ResNet, inverted residual blocks or standard bi-dimensional convolutions proved less effective at solving the task with the very limited resources available. This highlights the necessity to develop and optimize neural networks specifically for different hardware platforms. In JH PM HYU [13], the authors used clever regularization techniques in order to improve the generalization capabilities of the network. In conclusion, it is clear that, despite clever architectural designs, neural networks trained with optimized pre-processing and training strategies outperform the other approaches. In the future, it would be nice to see such techniques applied to the less computationally expensive models. A performance versus computational cost plot containing the best performing system of each participating team is presented.

**CHAPTER-6**

**ADVANTAGES AND APPLICATIONS**

**Advantages:**

* Independently process the single images and thus even stationary objects can be detected
* Can be used with the moving or Fixed cameras.
* Fast to train the model
* Capable to deal with any type of the noise

**Applications:**

* machine hearing
* smart home
* scene monitoring
* biological signal analysis

**CHAPTER-7**

**MATLAB**

**7.1 INTRODUCTION TO MATLAB**

**What Is MATLAB?**

MATLAB is an elite dialect for specialized registering. It incorporates calculation, representation, and programming in an easy to-utilize condition wherein issues and preparations are communicated in herbal numerical documentation. Run of the mill utilizes comprise

• Math and calculation

• Algorithm advancement

• Data obtaining

• Modeling, re-enactment, and prototyping

• Data examination, investigation, and representation

• Scientific and designing illustrations

• Application advancement, including graphical UI building

MATLAB is an intuitive framework whose important statistics aspect is an show off that does not require dimensioning. This allows you to tackle several specialized processing issues, particularly those with framework and vector info, in a small quantity of the time it'd take to compose a program in a scalar non intuitive dialect, as an instance, C or FORTRAN.

The call MATLAB stays for grid studies facility. MATLAB changed into first of all composed to present easy access to framework programming created by way of the LINPACK and EISPACK ventures. Today, MATLAB motors fuse the LAPACK and BLAS libraries, inserting the cutting side in programming for network calculation.

MATLAB has advanced over a time of years with contribution from several customers. In university situations, it's far the usual academic apparatus for early on and propelled guides in mathematics, designing, and science. In enterprise, MATLAB is the tool of choice for excessive-profitability studies, advancement, and exam.

MATLAB highlights a collection of more utility-specific arrangements known as tool booths. Important to most clients of MATLAB, device kits permit you to learnandapply particular innovation. Tool compartments are exhaustive accumulations of MATLAB capacities (M-records) that reach out the MATLAB condition to take care of precise training of problems. Territories in which tool stash are reachable include flag coping with, manipulate frameworks, neural structures, fluffy reason, wavelets, pastime, and severa others.

**The MATLAB System:**

The MATLAB system consists of five main parts.

**Development Environment:**

 This is the set of tools and centres that help you operate MATLAB features and files. Many of that gear are graphical person interfaces. It includes the MATLAB desktop and Command Window, a command history, an editor and debugger, and browsers for viewing assist, the workspace, files, and the hunt direction.

**The MATLAB Mathematical Function:**

This is a great collection of computational algorithms ranging from standard capabilities like sum, sine, cosine, and complex arithmetic, to extra sophisticated features like matrix inverse, matrix eigen values, Bessel functions, and speedy Fourier transforms.

**The MATLAB Language:**

This is a high-level matrix/array language with control flow statements, functions, data structures, input/output, and object-oriented programming features. It allows both "programming in the small" to rapidly create quick and dirty throw-away programs, and "programming in the large" to create complete large and complex application programs.

**Graphics:**

MATLAB has considerable centres for displaying vectors and matrices as graphs, as well as annotating and printing those graphs. It consists of high-stage functions for 2-dimensional and 3-dimensional records visualization, photograph processing, animation, and presentation graphics. It also consists of low-stage capabilities that will let you absolutely customise the appearance of graphics as well as to construct complete graphical person interfaces for your MATLAB programs.

**The MATLAB Application Program Interface (API):**

This is a library that allows you to put in writing C and Fortran applications that have interaction with MATLAB. It consists of facilities for calling workouts from MATLAB (dynamic linking), calling MATLAB as a computational engine, and for studying and writing MAT-documents.

**7.2 MATLAB WORKING ENVIRONMENT:**

## MATLAB DESKTOP:

Matlab Desktop is the principle Matlab application window. The desktop consists of five sub windows, the summon window, the workspace program, the existing catalog window, the order records window, and at the least one figure home windows, which can be proven simply while the consumer suggests a sensible.

The order window is the area the customer sorts MATLAB orders and expressions at the initiate (>>) and wherein the yield of these fees is shown. MATLAB characterizes the workspace because the association of factors that the customer makes in a work session. The workspace software demonstrates these elements and some statistics approximately them. Double tapping on a variable within the workspace application dispatches the Array Editor, which may be applied to get data and salary instances modify sure homes of the variable.

The present Directory tab over the workspace tab demonstrates the substance of the existing registry, whose way is seemed within the present index window. 1For case, within the windows running framework the manner may be as consistent with the subsequent: C:MATLABWork, demonstrating that registry "paintings" is a subdirectory of the primary catalog "MATLAB", which is delivered in pressure C. Tapping on the bolt inside the present index window demonstrates a rundown of as of past due utilized approaches. Tapping at the seize to one aspect of the window enables the client to exchange the existing catalog.

MATLAB utilizes an inquiry way to discover M-data and different MATLAB related documents, which might be sort out in catalogs within the PC file framework. Any file keep strolling in MATLAB must dwell inside the ebb and go with the flow registry or in an index that is on are trying to find manner. Of direction, the statistics supplied with MATLAB and math works device kits are included into the inquiry way. The least stressful method to look which indexes are at the inquiry manner. The handiest method to peer which catalogs are soon the quest way, or to encompass or regulate an inquiry manner, is to pick set manner from the File menu the computer, and after that utilization the set way exchange container. It is exquisite exercise to add any typically utilized catalogs to the pursuit way to hold a strategic distance from again and again having the exchange the existing index.

The Command History Window contains a record of the orders a client has entered in the charge window, including both present and past MATLAB sessions. Already entered MATLAB orders can be chosen and re-executed from the charge history window by right

tapping on a summon or arrangement of orders. This activity dispatches a menu from which to choose different choices notwithstanding executing the orders. This is helpful to choose different choices notwithstanding executing the summons. This is a valuable component while trying different things with different orders in a work session

**Using the MATLAB Editor to create M-Files:**

The MATLAB manager is both a word processor unique for making M-statistics and a graphical MATLAB debugger. The proofreader can display up in a window without everybody else, or it could be a sub window in the laptop. M-facts are intended by means of the expansion .M, as in pixelup.M. The MATLAB editorial manager window has various draw down menus for errands, for instance, sparing, seeing, and troubleshooting documents. Since it plays out a few basic checks and furthermore utilizes shading to separate between exclusive additives of code, this content device is suggested as the equipment of selection for composing and changing M-capacities. To open the proofreader, sort regulate at the incite opens the M-report filename.M in a supervisor window, organized for altering. As referred to before, the record has to be inside the momentum catalog, or in an index within the pursuit manner.

**Getting Help:**

The important technique to get help on line is to utilize the MATLAB assist application, opened as a exclusive window both via tapping at the query mark image at the computing device toolbar, or by using writing help program on the provoke within the order window. The help Browser is an internet application coordinated into the MATLAB computing device that shows a Hypertext Markup Language (HTML) statistics. The Help Browser contains of two sheets, the assistance pilot sheet, used to find out data, and the show sheet, used to look the statistics. Clear as crystal tabs aside from pilot sheet are applied to play out a pursuit. Second, within the motion pictures taken via transferring camera setup, the state of affairs becomes extra complex because the heritage may additionally exchange by using shifting shot, we cannot tune item motion exactly inside the sum of distinction map. Therefore, in this situation, the purpose is executed through reusing the previous seam and applying it to the cutting-edge body. In order to discover the seams, we use the preceding seam from previous body to look the modern-day seam in contemporary frame. our method is using a seam computed in frame1 (in crimson) to go looking a comparable seam in frame2. For the pixels close by the area of previous seam, we decide how a lot the selected pixel might vary from the pixel of preceding seam. We use difference of the 2 pixels as the degree of temporal coherence. If the distinction value of first seam pixel is over the threshold, we can keep to go looking the next seam pixel on three feasible pixels (in yellow, blue and brown) in subsequent row, until we discover 5 consecutive pixels that also exceed the threshold.

When we can't search the matching seam, we recalculate the energy for a new seam. We assume a seam 𝑆l-1 has been calculated inside the previous body, and a seam must be calculated for the contemporary frame. For preserving the temporal coherence, we want to make a new seam close to the previous seam with the identical index. We use the distinction among preceding seam and all pixels at the current body as the measure

Thus we upload temporal coherence price Tc(i,j) to the strength map earlier than calculating a seam 𝑆L. The price Tc is zero while the body pixels have the equal fee as previous seam pixels. Using our temporal coherence price, we will calculate the seam which has least electricity and is more close to the preceding seam in previous frame. Consequently, we will decrease the jittery artifacts inside the films.

**COMMUNICATION:**

Communications System Toolbox™ offers algorithms and gear for the layout, simulation, and analysis of communications systems. These capabilities are furnished as MATLAB ® features, MATLAB System gadgets™, and Simulink ® blocks. The machine toolbox includes algorithms for source coding, channel coding, interleaving, modulation, equalization, synchronization, and channel modeling. Tools are supplied for bit blunders charge evaluation, producing eye and constellation diagrams, and visualizing channel characteristics. The machine toolbox additionally provides adaptive algorithms that allow you to version dynamic communications structures that use OFDM, OFDMA, and MIMO techniques. Algorithms support fixed-point facts arithmetic and C or HDL code era.

**Key Features**

▪ Algorithms for designing the physical layer of communications systems, which includes supply coding, channel coding, interleaving, modulation, channel fashions, MIMO, equalization, and synchronization

▪ GPU-enabled System objects for computationally intensive algorithms together with Turbo, LDPC, and Viterbi decoders

▪ Interactive visualization equipment, consisting of eye diagrams, constellations, and channel scattering capabilities

▪ Graphical tool for evaluating the simulated bit mistakes rate of a machine with analytical outcomes

▪ Channel models, consisting of AWGN, Multipath Rayleigh Fading, Rician Fading, MIMO Multipath Fading, and

LTE MIMO Multipath Fading

▪ Basic RF impairments, along with nonlinearity, section noise, thermal noise, and section and frequency offsets

▪ Algorithms available as MATLAB features, MATLAB System objects, and Simulink blocks

▪ Support for fixed-point modeling and C and HDL code technology

**System Design, Characterization, and Visualization:**

The layout and simulation of a communications gadget requires analyzing its reaction to the noise and interference inherent in real-world environments, reading its behavior the usage of graphical and quantitative manner, and determining whether the resulting overall performance meets requirements of acceptability. Communications System Toolbox implements a selection of obligations for communications machine layout and simulation. Many of the functions, System objects™, and blocks inside the device toolbox perform computations associated with a specific thing of a communications gadget, consisting of a demodulator or equalizer. Other talents are designed for visualization or evaluation.

**System Characterization**

The system toolbox offers several standard methods for quantitatively characterizing system performance:

▪ Bit error rate (BER) computations

▪ Adjacent channel power ratio (ACPR) measurements

▪ Error vector magnitude (EVM) measurements

▪ Modulation error ratio (MER) measurements

Because BER computations are fundamental to the characterization of any communications system, the system toolbox provides the following tools and capabilities for configuring BER test scenarios and accelerating BER simulations:

**BER tool**— A graphical user interface that enables you to analyze BER performance of communications systems. You can analyze performance via a simulation-based, semi analytic, or theoretical approach.

**Error Rate Test Console** — A MATLAB object that runs simulations for communications systems to measure error rate performance. It supports user-specified test points and generation of parametric performance plots and surfaces. Accelerated performance can be realized when running on a multi core computing platform.

**Multi core and GPU acceleration** — A capability provided by Parallel Computing Toolbox™ that enables you to accelerate simulation performance using multi core and GPU hardware within your computer.

**Distributed computing and cloud computing support** — Capabilities provided by Parallel Computing Toolbox and MATLAB Distributed Computing Server™ that enable you to leverage the computing power of your server farms and the Amazon EC2 Web service. Performance Visualization. The system toolbox provides the following capabilities for visualizing system performance:

**Channel visualization tool** — For visualizing the characteristics of a fading channel

**Eye diagrams and signal constellation scatter plots** — for a qualitative, visual understanding of system behavior that enables you to make initial design decisions

**Signal trajectory plots** — for a continuous picture of the signal’s trajectory between decision points

**BER plots** — for visualizing quantitative BER performance of a design candidate, parameterized by metrics such as SNR and fixed-point word size

**Analog and Digital Modulation**

Analog and digital modulation strategies encode the facts circulation into a sign this is appropriate for transmission. Communications System Toolbox presents some of modulation and corresponding demodulation abilities. These talents are available as MATLAB features and gadgets, MATLAB System Modulation sorts provided by the toolbox are:

**Source and Channel Coding**

Communications System Toolbox affords source and channel coding talents that can help you develop and compare communications architectures fast, enabling you to discover what-if eventualities and avoid the need to create coding competencies from scratch.

**Source Coding**

Source coding, also referred to as quantization or signal formatting, is a manner of processing facts a good way to lessen redundancy or prepare it for later processing. The system toolbox offers a diffusion of styles of algorithms for imposing source coding and interpreting, inclusive of:

▪ Quantizing

▪ Companding (*µ*-law and A-law)

▪ Differential pulse code modulation (DPCM)

▪ Huffman coding

▪ Arithmetic coding

**Channel Coding**

▪ orthogonal area-time block code (OSTBC) (encoder and decoder for MIMO channels)

▪ Turbo encoder and decoder examples

The gadget toolbox offers application functions for developing your personal channel coding. You can create generator polynomials and coefficients and syndrome deciphering tables, in addition to product parity-take a look at and generator matrices.

The system toolbox additionally presents block and convolutional interleaving and deinters leaving functions to reduce facts errors as a result of burst mistakes in a conversation machine:

**Block,** including General block interleaver, algebraic interleaver, helical scan interleaver, matrix interleaver, and random interleaver.

**Convolutional,** including General multiplexed interleaver, convolutional interleaver, and helical interleaver

**Channel Modeling and RF Impairments**

Channel Modeling

Communications System Toolbox provides algorithms and tools for modeling noise, fading, interference, and different distortions which might be commonly found in communications channels. The system toolbox supports the subsequent styles of channels:

▪ Additive white Gaussian noise (AWGN)

▪ Multiple-enter multiple-output (MIMO) fading

▪ Single-enter single-output (SISO), Rayleigh, and Rician fading

▪ Binary symmetric

A MATLAB channel object provides a concise, configurable implementation of channel models, enabling you to

specify parameters such as:

▪ Path delays

▪ Average path gains

▪ Maximum Doppler shifts

▪ K-Factor for Rician fading channels

▪ Doppler spectrum parameters

For MIMO systems, the MATLAB MIMO channel object expands these parameters to also include:

▪ Number of transmit antennas (up to 8)

▪ Number of receive antennas (up to 8)

▪ Transmit correlation matrix

▪ Receive correlation matrix

To combat the effects noise and channel corruption, the system toolbox provides block and convolutional coding and decoding techniques to implement error detection and correction. For simple error detection with no inherent correction, a cyclic redundancy check capability is also available. Channel coding capabilities provided by the system toolbox include:

▪ BCH encoder and decoder

▪ Reed-Solomon encoder and decoder

▪ LDPC encoder and decoder

▪ Convolutional encoder and Viterbi decoder

****

**RF Impairments**

To model the effects of a non-ideal RF front end, you can introduce the following impairments into your communications system, enabling you to explore and characterize performance with real-world effects:

▪ Memory less nonlinearity

▪ Phase and frequency offset

▪ Phase noise

▪ Thermal noise

You can include more complex RF impairments and RF circuit models in your design using SimRF™.

****

**Equalization and Synchronization**

Communications System Toolbox lets you discover equalization and synchronization strategies. These techniques are usually adaptive in nature and tough to design and symbolize. The machine toolbox affords algorithms and tools that will let you swiftly select the proper approach on your communications machine. Equalization To compare one-of-a-kind techniques to equalization, the device toolbox offers you with adaptive algorithms which include:

▪ LMS

▪ Normalized LMS

▪ Variable step LMS

▪ Signed LMS

▪ MLSE (Viterbi)

▪ RLS

▪ CMA

These adaptive equalizers are available as nonlinear decision feedback equalizer (DFE) implementations and as

Linear (symbol or fractionally spaced) equalizer implementations.

**Synchronization**

The device toolbox provides algorithms for each service segment synchronization and timing phase synchronization. For timing section synchronization, the machine toolbox presents a MATLAB Timing Phase Synchronizer object that offers the following implementation techniques:

▪ Early-late gate timing method

▪ Gardner’s method

▪ Fourth-order nonlinearity method

**Stream Processing in MATLAB and Simulink**

Most verbal exchange structures cope with streaming and frame-primarily based statistics using a aggregate of temporal processing and simultaneous multi frequency and multichannel processing. This form of streaming multidimensional processing can be visible in superior communication architectures consisting of OFDM and MIMO. Communications System Toolbox enables the simulation of advanced communications structures via helping move processing and frame-based simulation in MATLAB and Simulink. In MATLAB, circulate processing is enabled by way of System items™, which use MATLAB objects to symbolize time-based and facts-driven algorithms, sources, and sinks. System objects implicitly manipulate many information of flow processing, including information indexing, buffering, and management of set of rules state. You can mix System gadgets with fashionable MATLAB functions and operators. Most System items have a corresponding Simulink block with the identical abilities. Simulink handles circulation processing implicitly with the aid of coping with the float of information thru the blocks that make up a Simulink model. Simulink is an interactive graphical environment for modeling and simulating dynamic systems that uses hierarchical diagrams to symbolize a machine version. It includes a library of widespread-reason, predefined blocks to represent algorithms, resources, sinks, and device hierarchy.

**Implementing a Communications System**

Fixed-Point Modeling Many communications systems use hardware that requires a fixed-point representation of your design.

Communications System Toolbox supports fixed-point modeling in all relevant blocks and System objects™ with tools that help you configure fixed-point attributes.

Fixed-point support in the system toolbox includes:

▪ Word sizes from 1 to 128 bits

▪ Arbitrary binary-point placement

▪ Overflow handling methods (wrap or saturation)

▪ Rounding methods: ceiling, convergent, floor, nearest, round, simplest, and zero

Fixed-Point Tool in Simulink Fixed Point™ facilitates the conversion of floating-point data types to fixed point. For configuration of fixed-point properties, the tool tracks overflows and maxima and minima.

**Code Generation**

Once you've got advanced your set of rules or communications device, you can robotically generate C code from it for verification, rapid prototyping, and implementation. Most System gadgets, functions, and blocks in Communications System Toolbox can generate ANSI/ISO C code the use of MATLAB Coder™, Simulink Coder™, or Embedded Coder™. A subset of System gadgets and Simulink blocks also can generate HDL code. To leverage present highbrow belongings, you can choose optimizations for specific processor architectures and integrate legacy C code with the generated code.

You can also generate C code for both floating-point and fixed-point data types.

DSP Proto typing DSPs are used in communication system implementation for verification, rapid prototyping, or final hardware implementation. Using the processor-in-the-loop (PIL) simulation capability found in Embedded Coder, you can verify generated source code and compiled code by running your algorithm’s implementation code on a target processor. FPGA Prototyping

FPGAs are used in communication systems for implementing high-speed signal processing algorithms. Using the FPGA-in-the-loop (FIL) capability found in HDL Verifier™, you can test RTL code in real hardware for any existing HDL code, either manually written or automatically generated HDL code.

**CHAPTER -8**

**HARDWARE & SOFTWARE REQUIREMENTS:**

**Software:**

• Matlab R2018a.

**Hardware:**

**Operating Systems:**

• Windows 10

• Windows 7 Service Pack 1

• Windows Server 2019

• Windows Server 2016

**Processors:**

Minimum: Any Intel or AMD x86-64 processor

Recommended: Any Intel or AMD x86-64 processor with four logical cores and AVX2 instruction set support

**Disk:**

Minimum: 2.9 GB of HDD space for MATLAB only, 5-8 GB for a typical installation

Recommended: An SSD is recommended a full installation of all Math Works products may take up to 29 GB of disk space

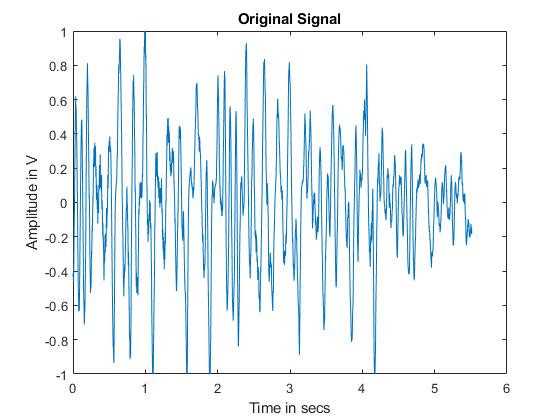
**RAM:**

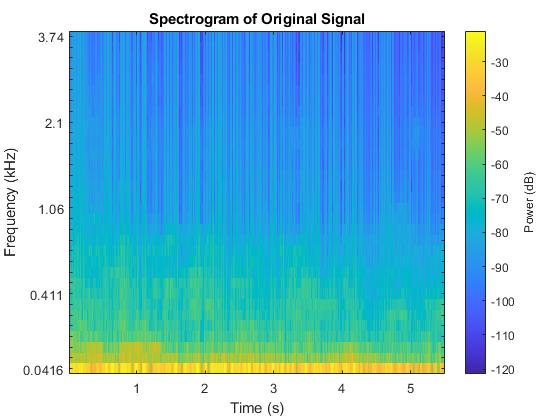
Minimum: 4 GB

Recommended: 8

**CHAPTER-9**

**RESULTS**





**CHAPTER-10**

**CONCLUSION**

This paper presented an analysis of the Low-Complexity Acoustic Scene Classification task in DCASE 2022 Challenge. The task was modelled after devices to bring the research problem closer to real-world applications. The number of multiply-and-accumulate operation set to 30 M and the total maximum number of parameters set to 128 K have been a sufficient constraint to receive a variety of interesting techniques, even though most systems were close to the imposed limits. The top systems employed large receptive fields, coordinate attention and transformer architectures to optimize performance, while quantization-aware training was the most used technique among participants to fulfil the complexity constrains. The number of submissions has decreased slightly from previous years, which may reflect the increased complexity of the task. However, the use of different devices for context-awareness is a sought after direction for applicability, therefore solutions suitable for limited computational power are needed. Moreover, the task could consider steering development towards solutions where room for improvement is still needed, like minimizing the working memory usage.

**CHAPTER-11**

**REFERENCES**

[1] E. Benetos, D. Stowell, and M. D. Plumbley, Approaches to Complex Sound Scene Analysis. Cham: Springer International Publishing, 2018, pp. 215–242.

[2] T. Heittola, A. Mesaros, and T. Virtanen, “Acoustic scene classification in DCASE 2020 challenge: generalization across devices and low complexity solutions,” in Proceedings of the Detection and Classification of Acoustic Scenes and Events 2020 Workshop, Tokyo, Japan, November 2020, pp. 56–60.

[3] K. Koutini, F. Henkel, H. Eghbal-zadeh, and G. Widmer, “CP-JKU submissions to DCASE’20: Low-complexity cross-device acoustic scene classification with rf-regularized CNNs,” DCASE2020 Challenge, Tech. Rep., June 2020.

[4] S. Suh, S. Park, Y. Jeong, and T. Lee, “Designing acoustiscene classification models with CNN variants,” DCASE2020 Challenge, Tech. Rep., June 2020.

[5] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “Mobilenetv2: Inverted residuals and linear bottle-necks,” in 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 4510–4520.

[6] M. Tan and Q. V. Le, “Efficientnet: Rethinking model scaling for convolutional neural networks,” in 36th Int. Conf. on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA, ser. Proceedings of Machine Learning Re-search, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 2019, pp. 6105–6114.

[7] B. Kim, Y. Seunghan, K. Jangho, and C. Simyung, “QTI sub-mission to DCASE 2021: Residual normalization for device-imbalanced acoustic scene classification with efficient de- sign,” DCASE2021 Challenge, Tech. Rep., June 2021.

[8] A. Mesaros, T. Heittola, and T. Virtanen, “Acoustic scene classification in DCASE 2019 challenge: closed and open set classification and data mismatch setups,” in Proc. of the DCASE 2019 Workshop, New York, Nov 2019.

[9] S. Hershey, S. Chaudhuri, D. P. W. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, M. Slaney, R. J. Weiss, and K. Wilson, “CNN ar-

chitectures for large-scale audio classification,” in 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2017, pp. 131–135.

[10] J. Cramer, H.-H. Wu, J. Salamon, and J. P. Bello, “Look, listen and learn more: Design choices for deep audio embeddings,” in IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP), Brighton, UK, May 2019, pp. 3852–3856.

[11] S. Kumari, D. Roy, M. Cartwright, J. P. Bello, and A. Arora, “EdgeL3: Compressing L3-net for mote scale urban noise monitoring,” in 2019 IEEE International Parallel and Dis- tributed Processing Symposium Workshops (IPDPSW), May 2019, pp. 877–884.

[12] F. Schmid, S. Masoudian, K. Koutini, and G. Widmer, “CP- JKU submission to dcase22: Distilling knowledge for low- complexity convolutional neural networks from a patchout audio transformer,” DCASE2022 Challenge, Tech. Rep., June2022.